

CS511

Notes for final-term

by “Muhammadians”

For Your kind information (read well):

Asslam-o-Alaikum!

Virtual university's students “*Junaid Waris and Farhan khan*” are here. As there are no official handouts given by the virtual university, the university provides a **reading section** after every module. We've just copied that and pasted text as well as images. After completion of the midterm notes, (CS511 midterm notes by muhammadians) we are going to compile final term notes which named as now “CS511 final term notes W#7-13”.

Now this is the pdf file of 7 (W# 7th to 13) weeks of final -term only. Because still only these weeks are opened. These notes will be updated every week until final-term and shared in groups. Only this file will be updated and shared in groups. Before final-term exams, this will update completely. Try to find out where you got these notes the first time, and if you don't find them, then you can contact us in our **official group as well as for current papers** (the group link is given below).

We hope you will find it best because it's difficult to read every module's reading material from LMS, so we tried to save your time and headache as well. We compiled this for the welfare of virtual university students. All credits go to only Virtual University, who provide us the material.

Don't print out this pdf file. This file has half blank pages or 3rd of 4th blank page (because of module ending). Only going through, just reading bold material, and remembering only SQ, LQ, and quizzes; these do not ensure the preparation for good marks, although they do ensure passing marks. Well, I wish you the best of luck for your future! Remember us in your prayers. Thank you so much!

Regards:

- © “Junaid Waris”
- © “Farhan Khan”

Whats App Group Link:

<https://chat.whatsapp.com/GkX5GZnrnWDDwWk4QB1ec>

(We may respond slowly but respond to you effectively.)

Module No.81

Event handling

In every language of computer science event has its own significance, but in JavaScript, which is utilized in web development, it is very significant. An event is an action that occurs as a result of the user or another source, such as a mouse click. Let's see how it is handled in JavaScript.

Introduction:

- JavaScript event is an action that can be detected by JavaScript.
- Many of them are initiated by user actions, some are generated by the browser itself.
- We say that an event is triggered and then is handled by JavaScript functions.

There are multiple ways to handle an event. Let's have a look at them.

The given figure no.80.1 depicts it in detail.

Event Handling Approaches-Inline Hook

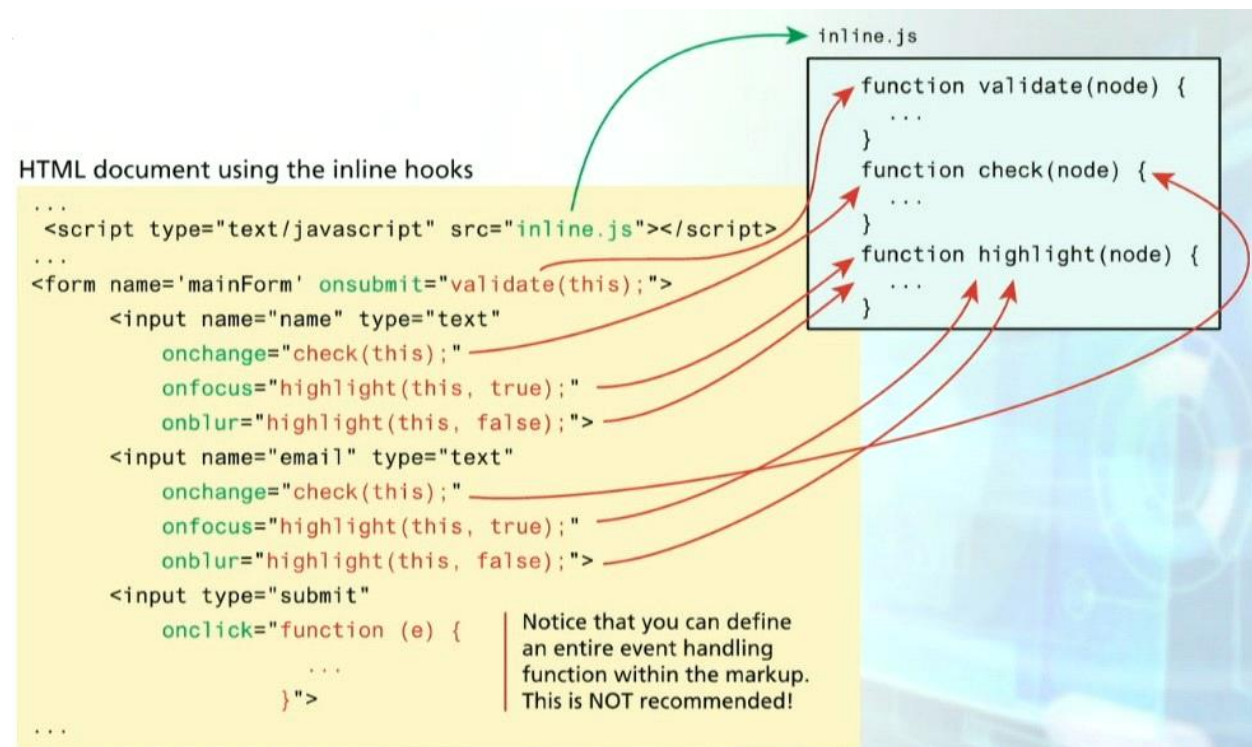


Figure no.80.1

The very first line portrays that a JavaScript code has been stored in an external file which includes the some functions such as

```
function validated (node){...}
```

```
// validate the node
```

```
function check (node){....} // check whether the node is or not an
appropriate one.
```

```
function highlight (node){....} // highlight text
```

The figure 80.1 clearly depicts the events on which these functions are invoked.

Another way of handling event is.

Event –Handling Approaches-Event property Approach:

- `var mybutton=document.getElementById ('example');`//In JavaScript, get an element from the DOM tree

(Any approach, such as query Selector or get element ById, can be used). After that we write function name on its appropriate event. As shown below.

- `myButton.onclick=alert ('some message');`
- `myButon.addEventListener ('click', alert ('some message'));`// it take two parameter
- `myButton.addEventListener ('mouseout', funcName);`

Similarly, we may define several events on a single object using this method.

The key benefit of this method is that JavaScript code is segregated from HTML. The script/code may be in an external file or an embedded JavaScript. The code of HTML will be very clean and clear, while making changes in HTML you need not worry about the functionality and vice versa.

Another way to define event.

Event-Handling Approach-Event Listener Approach (anon function)// anonymous function is pass on.

```
myButton.addEventListener ('click', function ()
```

```
{
```

```
    var d = new Date ();
```

```
    alert ("You clicked this on" +d.toString ());
```

```
}); // in this piece of code the whole function is pass as a parameter to
```

AddEventListener when a click event is generated.

The last method to handle an event is,

Event Object

When an event is triggered, the browser will construct an event object that contains information about the event.

```
div.addEventListener ('click', function(e)
```

```
{ //findout where the user clicked
```

```
var x=e. clientX;
```

and via **e. Target** can be used to manipulate elements of **div**.

Module No.82

Capturing and Bubbling.

Let's see how JavaScript handle events.

Event Handling:

Events are handled in three phases.

- Capturing
 - Event propagation from parent towards the most nested child.
- Target
 - Identifying the target.
- Bubbling
 - Elements registered for the events are triggered from the most nested child towards the outermost.

In JavaScript, events progress from the immediate object that caused the event to the document root. This is referred to as event propagation or event bubbling.

While this may appear to be a difficult concept, the underlying notion is extremely straightforward. When you click on an element, you're also clicking on all the ascendants in the DOM tree, as seen in Figure 82.1

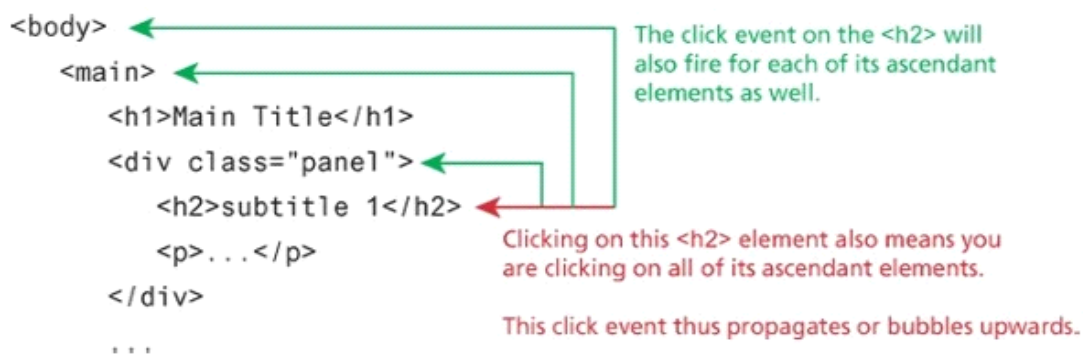


Figure 82.1

Module No.83

Event Types – mouse, keyboard and touch.

Figure taken from: Fundamentals of Web Development (Second Edition, Pearson), Authors: Randy Connolly, Ricardo Hoar

The click event is perhaps the most visible, but JavaScript and the DOM support a variety of others. In reality, the W3C specifies various classes of event, each having multiple types of events inside each class. Mouse events, keyboard events, touch events, form events, and frame events are some of the most popular event types.

Let's have a look at few types of events which are supported by JavaScript.

Mouse Events: Event related to mouse is called mouse events.

Mouse events are used to describe a variety of mouse-driven interactions. Mouse click and mouse move events are two types of mouse events. Many mouse events can be sent at once, which is unusual. The user may move the mouse from one div to another at the same time, activating the mouseon and mouseout events as well as the mousemove event. To deal with these difficulties, use the Cancelable and Bubbles properties. Following is the list of mouse events.

- **click** The mouse was clicked on an element.
- **dblclick** The mouse was double clicked on an element. // double click
- **mousedown** The mouse was pressed down over an element.// pressed button but did not release yet.
- **mouseup** The mouse was moved (not clicked) over an element.// when the button is released.
- **mouseover** The mouse was moved (not clicked) over an element.
- **mouseout** The mouse was moved off of an element.
- **mousemove** The mouse was moved while over an element.

These are different types of events which can be used in JavaScript.

Note: click event // in this both up and down will be invoked.

And **mouseup** is used only for up and likewise **mousedown** is used only for down.

Keyboard Events: the events related to keyboard is called keyboard events.

Within input fields, these events are most useful. With each key hit, we could, for example, validate an email address or submit an asynchronous request for a dropdown list of suggestions with each key press. <input type="text" id="key">

For this input box, we could listen for key press events and replay each pushed key back to the user as shown below.

```
document.getElementById("key").addEventListener("keydown", function (e) {  
  
    var keyPressed=e.keyCode;           // get the raw key code  
  
    var character=String.fromCharCode(keyPressed); // convert to string
```

```
alert ("Key " + character + " was pressed");  
});
```

Note: Regrettably, different browsers handle keyboard properties differently. For example, the `keyCode` property is not available for the `keypress` event in FireFox. If we used the same callback method for the `keypress` event in code above, we'd have to update the code to retrieve the key press as follows:

```
// use either the which or the keyCode property  
  
var keyPressed = e.which || e.keyCode;
```

Instead of clogging our code with these kind of browser testing conditional statements, we can use something like the jQuery framework to handle these quirks.

Types of keyboard events as shown below.

- **Keydown** The user is pressing a key (this happens first)
- **Keypress** The user presses a key (this happens after keydown)
- **KeyUp** The user releases a key that was down (this happens last)

Touch events: these are relatively new events which are especially used in mobile devices.

- Touch events are a relatively new category of events that can be triggered by devices with touch screens.
- The different events (e.g., `touchstart`, `touchmove`, and `touchend`) are analogous to some of the mouse events (`mousedown`, `mousemove`, and `mouseup`).
- Touch events are only available by default in Chrome and iOS Safari at the time of writing.
- In Edge and FireFox, the user must allow touch events. Pointer events, a new standard specification that combines mouse, touch screen, and pen input into a single event type, are supported by Microsoft Edge. However, pointer events are only supported in Edge at the time of writing.

A form has been created in HTML page, to test a keyboard event.

```
<form id = "info" method= "post" action= "#">  
  
  <div>  
  
    <label> Name</label>  
  
    <input type = "text" id = "name">  
  
  </div>  
  
</div>
```

<label> Email</label>

<input type = "text" id = " email">

</div>

<input type = "submit"> </form>

```
<time>November 15, 2012</time>
<div>
  <p>By Ricardo on <time>September 15, 2021</time>
  <p class="comment">Easy on the HDR buddy.</p>
</div>
<div>
  <p>By Susan on <time>October 12, 2012</time>
  <p class="comment last">I love Central Park.
</div>
<form id="info" method="post" action="#">
  <div>
    <label>Name</label>
    <input type="text" id="name">
  </div>
  <div>
    <label>Email</label>
    <input type="text" id="email">
  </div>
  <input type="submit">
</form>
```

- [Canada](#)
- [Germany](#)
- [United States](#)

Comments as of Nov

By Ricardo on Septem

Easy on the HDR bud

By Susan on October

I love Central Park.

Name

Email

- [Home](#)
- [Browse](#)

Figure 83.1

//comment out the div that was built with the event listener as shown in figure 83.2

```
var divs = document.querySelectorAll('div')
for (var div of divs) {
  // div.addEventListener('click', div_cl)
}
```

Figure 83.2

```
<script>
```

```
var uname=document.querySelector('name')// reserved word can't be used as variable.
```

```
uname.addEventListener('keypress', function(e){console.log(e.keyCode)
```

```
if (e.keyCode==95)
```

```
e.preventDefault();} )
```

The event listener is working fine, because we get the ASCII code for the keys which are being pressed, and when the key code is equal to 95 then prevent the default behavior, it implies that the user is not allowed to type underscore in the input box as shown in figure 83.3,

```
<ul>
  <li><a href="#">Home</a> | </li>
  <li><a href="#">Browse</a> | </li>
</ul>
</footer>
<script>
  var uname = document.querySelector('#name')
  uname.addEventListener('keypress',function(e){
    console.log(e.keyCode)
    if (e.keyCode == 95)
      e.preventDefault();
  })
  function div_click(e) {
    var r = Math.floor(Math.random()*255)
    var g = Math.floor(Math.random()*255)
    var b = Math.floor(Math.random()*255)
    var col = `background-color: rgb(${r},${g},${b})`;
    this.style = col;
    e.stopPropagation();
  }
```

Figure 83.3

Module No.84

Form Validation; Client side input validation

Form validation is the process of determining whether or not the user has filled out the form correctly. This activity can be done at various stages. E.g

- **HTML Level**

// using built-in HTML element we can do validation e.g type of an email address must be email likewise specify the range for a specific number.

- **JavaScript Level**

// this is also called client side validation.

Note: the user may disable JavaScript. And submit the form. Because similar situations may arise, it is vital to validate the form at the server level as well to prevent erroneous data from being stored in the database. So it good to have validation at all the stages.

- **Webserver Level**

Form Events:

Events related to form are called form events. Forms are the primary means of collecting and transmitting user input to the server. Form events allow us to perform some real-time processing in response to user input. The submit event is the most popular JavaScript listener for forms. We use preventDefault() to prevent submitting to the server and inform the user if the password field (with id pw) is blank. Otherwise, nothing is done, allowing the default event to occur (submitting the form).

Catching the submit event and validating a password to not be blank

```
document.getElementById("loginForm").addEventListener('submit', function(e)
```

```
{ var pass = document.getElementById("pw").value;
if (pass=="") { alert ("enter a password");
e.preventDefault (); } });
```

Form events

- submit// When the form is submitted this event is triggered. We can do some prevalidation of the form in JavaScript before sending the data on to the server.
- reset // HTML forms have the ability to be reset. This event is triggered when that happens.
- formdata

Textfield events

- blur // Triggered when a form element has lost focus (that is, control has moved to a different element), perhaps due to a click or Tab key press.
- change// some <input>, <textarea> or <select> field had their value change. This could mean the user typed something, or selected a new choice.
- focus // Complementing the blur event, this is triggered when an element gets focus (the user clicks in the field or tabs to it).
- select // When the users selects some text. This is often used to try and prevent copy/paste.
-

Let's look at an example where these events will be used to validate a form.

```
uname.addEventListener ('change', check_name);
email.addEventListener ('change', check_email);
form.addEventListener ('submit', validate);
function check_name () {
if (this.value.length<5)
{
this.style= 'background-color: pink';
} else {
this.style= 'background-color: white';
```

```

}

}

function check_email () {

    var re= / [^@]+@[\.\.]+\.[\.\.]+/

    if (! re.test (this.value)){

        This.style= 'background-color: pink';

    } else {

this.style= 'background-color: white;

}

}

function validate (e) {

    var re=/ [^@]+@[\.\.]+\.[\.\.]+/

    if (uname.value.length<5||re.test (email.value) ==false {

        e.preventDefault ();

    }}

```

Module No.85

Introduction to jQuery

jQuery is one of the JavaScript libraries that are used to assist us with web development. jQuery is a popular framework that gives developers access to cross-browser animation tools, user interface elements, and DOM manipulation functions, among other things. Learning about frameworks, especially jQuery, can help you improve your development abilities by allowing you to use ever-evolving frameworks that provide new and increasingly expected functionality.

The Most popular Framework

- Makes working with JavaScript and DOM easy// [jQuery also makes it easy to select an element and to perform different operation on it](#)
- Cross browser compatibility // [it makes sure that the code written is executable on all the different browsers.](#)
- Most popular JavaScript library (as of 2021). According to w3techs.com
 - Used by 78% of all website
 - 96% market share

How to include JQuery in our code, there are multiple ways to include it.

Including JQuery

- Use a Content Delivery Network (CDN)

```
<script src = https://code.jquery.com/jquery-3.6.0.min.js> </script>  
//can include directly from CDN, min.js is best as it has minimum size.
```

Can also be

- Download and used on your web server
- Installed via package manager(npm, yarn, etc)

Note: JQuery can be included in the head, or before body in HTML file.

Let's have a look at an example that how it can be included in HTML file.

Figure 85.1 shows that how it can be included when we have an internet connection.

```
<head>  
  <meta charset="utf-8">  
  <title>Share Your Travels -- New York - Central Park</title>  
  <style>  
    .grey {  
      background-color: grey;  
    }  
  </style>  
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
</head>  
<body>
```

Figure no 85.1

But if we have no internet connection then download and save it in folder where you have HTML file.

And then include it using absolute path as shown in figure 85.2

```
<head>  
  <meta charset="utf-8">  
  <title>Share Your Travels -- New York - Central Park</title>  
  <style>  
    .grey {  
      background-color: grey;  
    }  
  </style>  
  <script src="jquery-3.6.0.min.js"></script>  
</head>  
<body>
```

Figure no. 85.2

Module No.86

jQuery selectors

The programmatic selection of DOM elements is one of the most common JavaScript jobs. The element selection method in jQuery is both strong and easy. JavaScript has a variety of functions for selecting an element, such as `getElementById()` and `querySelector()`. Although the `querySelector()` and `querySelectorAll()` capabilities now let you pick DOM elements using CSS selectors, developers have only been able to assume that these functions are available in most users' browsers since around 2014 (IE 8 and earlier didn't fully support these techniques). One of the reasons for jQuery's early popularity was that it gave programmers with a straightforward cross-browser way to programmatically select an element using regular CSS selectors.

- The power of jQuery resides in the function named `jQuery()`
- There's also an alias for this function named `$()`
- You can combine CSS selectors with the `$()` notation to select DOM objects that match CSS attributes

JavaScript versus jQuery selection

```
<p id="here">hello <span>there</span></p>
<ul>
  <li>France</li>
  <li>Spain</li>
  <li>Thailand</li>
</ul>
<script>
/* selecting using regular JavaScript */
var node = document.getElementById("here");
var link = document.querySelectorAll("ul li");
/* equivalent selection using jQuery */
var node = $("#here");
var link = $("ul li");
</script>
```

So, what is the result of the `$()` function? The `getElementById()` function, as you may recall, returns an Element object, whereas `querySelectorAll()` returns a NodeList object. The jQuery set object, which is an array-like structure that holds a set of DOM elements that match the selector, is returned instead by the `$()` function.

Instead of returning a single object, the `$()` function always returns a group of results. This is simple to overlook and might cause unanticipated frustration. For example, the following code can be used to obtain something that is singular (such as the `body` element):

```
$('#body') = temp
```

Due to the fact that the `$()` function always produces a set of results, you must refer to this element as the 0th array element, as illustrated below:

```
var b = temperature[0];
```

Nonetheless, the fact that jQuery always produces an array is one of its strengths: regardless of whether a selector returns a single value or numerous values, all of its methods work the same.

Basic Selectors

The four basic selectors were defined back in Chapter 4, and include the universal selector, class selectors, id selectors, and elements selectors. Selectors in jQuery are designed to closely resemble the CSS specification, which is especially useful given that CSS is something you've learnt and used throughout this book:

- `$("*")`—Universal selector matches all elements (and is slow).
- `$("tag")`—Element selector matches all elements with the given element name.
- `$(".class")`—Class selector matches all elements with the given CSS class.
- `$("#id")`—Id selector matches all elements with a given HTML id attribute.

For example to pick the single `div` element with `id="grab"`, write:

```
var singleElement = $("#grab");
```

To get a set of all the `<a>` elements the selector would be:

```
var allAs = $("a");
```

Other CSS selectors mentioned in Chapter 4: attribute selectors, pseudo-element selectors, and contextual selectors can be used in addition to these basic selectors, as shown in Figure 86.1. The rest of this section goes over a few of these selections and how they work with jQuery.

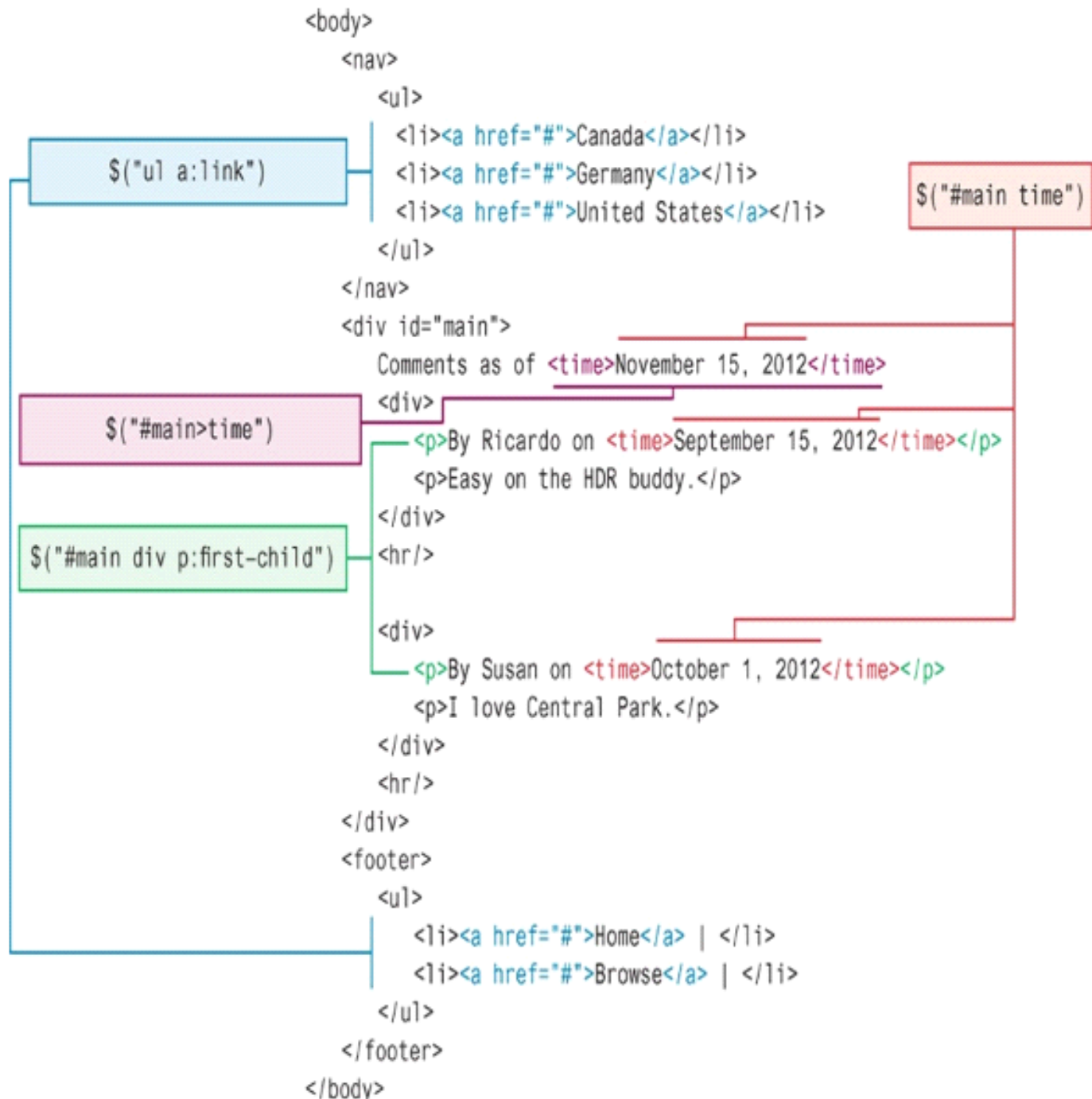


Figure 86.1: figure taken from: Fundamentals of Web Development (Second Edition, Pearson), Authors: Randy Connolly, Ricardo Hoar

jQuery's Content Filter Selection

The jQuery content filters provide you more selection options. You can use:has() to choose elements that have a specific child,:empty to select elements that have no children, and:contains to match a specific piece of text (). Consider the following illustration:

```
var allWarningText = $( "body *:contains ('warning')");
```

It will provide a list of all DOM elements that include the term warning. You can see how we would want to draw attention to those DOM elements by colouring the backdrop red, as seen in Figure 86.2.

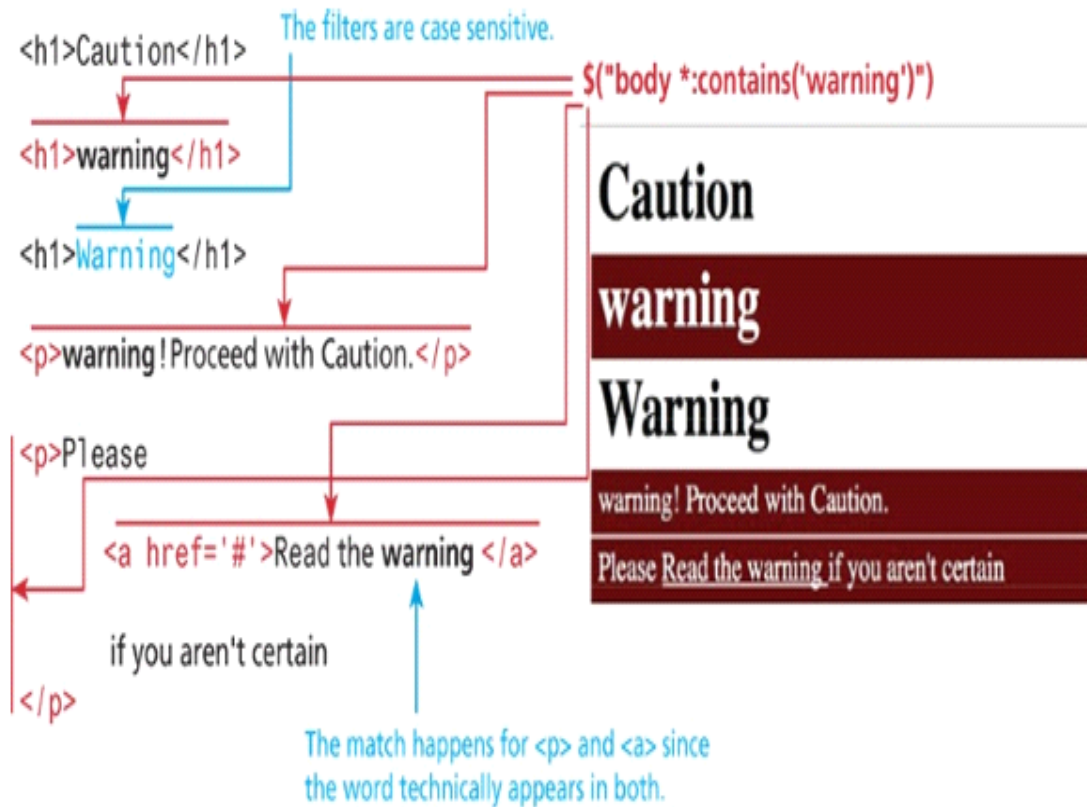


Figure 86.2: figure taken from: Fundamentals of Web Development (Second Edition, Pearson), Authors: Randy Connolly, Ricardo Hoar

Module No.87

jQuery Manipulators

//manipulator can be used for both read and write.

Common Element Manipulations in jQuery

You can choose any set of components from a web page using all of the selectors mentioned in this chapter. You can then alter them in a variety of ways once they've been picked.

- We can both set and get innerHTML through the `html()` method

//link is assigned the html code contained by the first p

```
var html = $("p").html (); // change the innerHTML of an element
```

```
$("#p#message").html("<b> Invalid Email</b>"); // change innerHTML of all matching elements
```

```
$(p.warning").html("No warning");
```

`.text()` can be used in similar manner for innerText

Common Element Manipulations----- HTML Attributes

The attributes defined in the HTML tags detailed in Chapter 3 are the fundamental collection of attributes associated to DOM elements. You've probably used key attributes like an `<a>` tag's href attribute, an `<img>` src attribute, and most elements' class attribute before. Using the `attr()` method on any element returned by a selector in jQuery, we can both set and obtain an attribute value. The attribute name is specified by the first parameter, and the optional second parameter allows you to edit the attribute by specifying a value. If no second parameter is provided, the procedure returns the attribute's current value. The following are some examples of possible applications:

We can both set and get an attribute value by using the `attr()` method

// link is assigned the href attribute of the first `<a>` tag

```
var link = $("a").attr("href"); // change all links in the page to http://funwebdev.com
```

```
$("#a").attr("href", http://funwebdev.com"); //change the class for all images on the page to fancy  
$("#img").attr("class", "fancy");
```

Common Element Manipulations----- HTML Properties

The checked property of a radio button or checkbox is the most common example of an HTML tag that includes both properties and attributes. The `attr()` method could be used to set HTML properties in earlier versions of jQuery. However, because properties are not actually characteristics, strange behavior ensued. Although `attr()` may return some (less useful) values, the `prop()` method is now preferred for retrieving and setting the value of a property.

To illustrate this subtle difference, consider a DOM element defined by

```
<input class="meh" type="checkbox" checked="checked">
```

The value of the `attr()` and `prop()` functions on that element differ as shown below.

```
var theBox = $(".meh");
```

```
theBox.prop("checked"); // evaluates to TRUE
```

```
theBox.attr("checked"); // evaluates to "checked"
```

`val()` used for getting/setting values of elements having the value property (e.g., `<input type="text">`...)

Common Element Manipulations----- Changing CSS

Modifying a CSS style is fairly similar to changing attributes in terms of syntax. The `css()` technique in jQuery is incredibly user-friendly. This method has two versions (with two different method signatures), one for getting the value and the other for setting it. The first version takes a single parameter holding the value you want for the CSS attribute and returns it.

```
var color = $("#element").css ("background-color"); // get the color
```

To set a CSS attribute, you use the second version of `css ()`, which takes two parameters: the first being the CSS attribute, and the second the value.

```
// set color to red
```

```
$("#element").css ("background-color", "red");
```

Module No.88

jQuery Manipulators – Example

The background color of name field has been changed using jQuery, as shown in below figure 88.1.

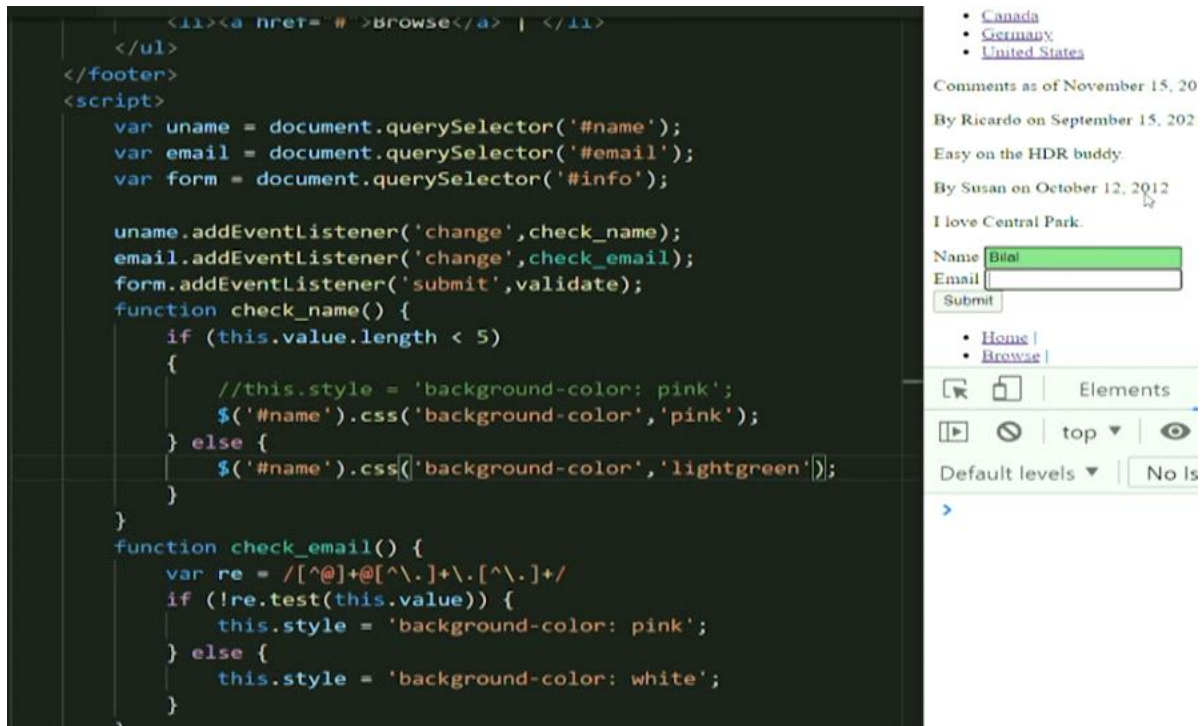


Figure 88.1

You can check out the length as well as the value typed in the name filed as illustrated in figure 88.2.

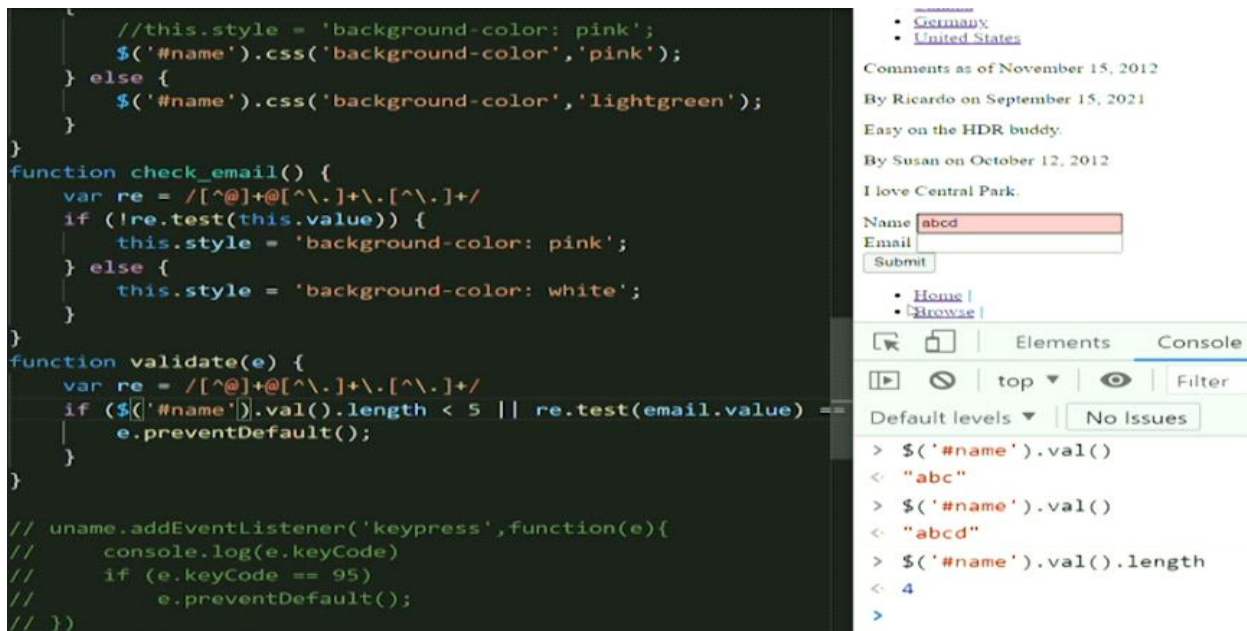


Figure 88.2

Note: You can also do it in the console as demonstrated below.

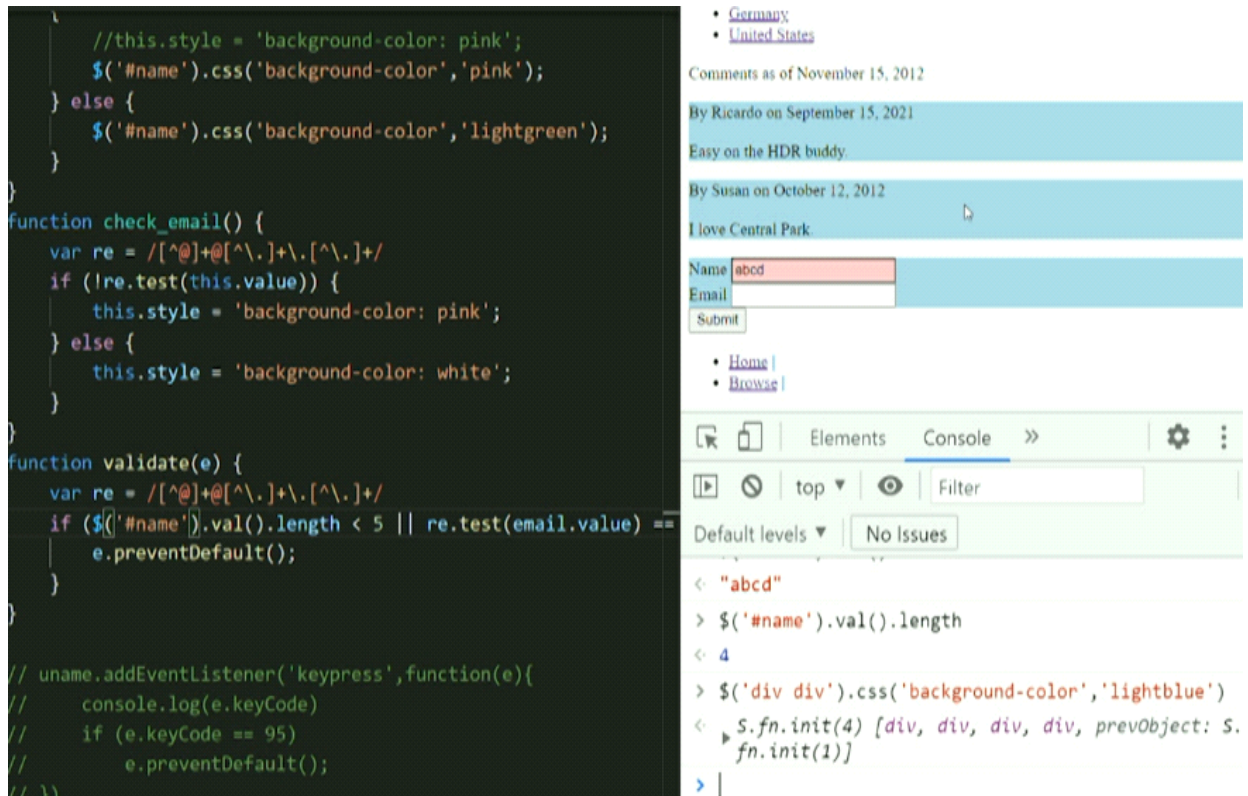


Figure 88.3

Using the code shown in the console, you can easily alter the background color of div and inner dive.

Module No.89

Events in jQuery

Introduction:

jQuery, like JavaScript, allows you to create and manage JavaScript event listeners/handlers. With a few minor syntax modifications, these events are used in the same way as JavaScript.

Listeners

For jQuery Setting up listeners for specific events is similar to how JavaScript works. While `addEventListener()` is used in pure JavaScript, jQuery includes `on()` and `off()` methods as well as shortcut methods for attaching events.

Looking at figure 89.1. we can see that jQuery is substantially less verbose than JavaScript once again. Is that, however, all it has to offer? When working with events, jQuery also makes many typical chores easier. Figure shows a simple example with three different mouse events. The click event handler, in particular, makes advantage of the jQuery `off()` function to stop listening to the mouse move event. This is far more straightforward than the JavaScript-only solution.

Binding and unbinding events

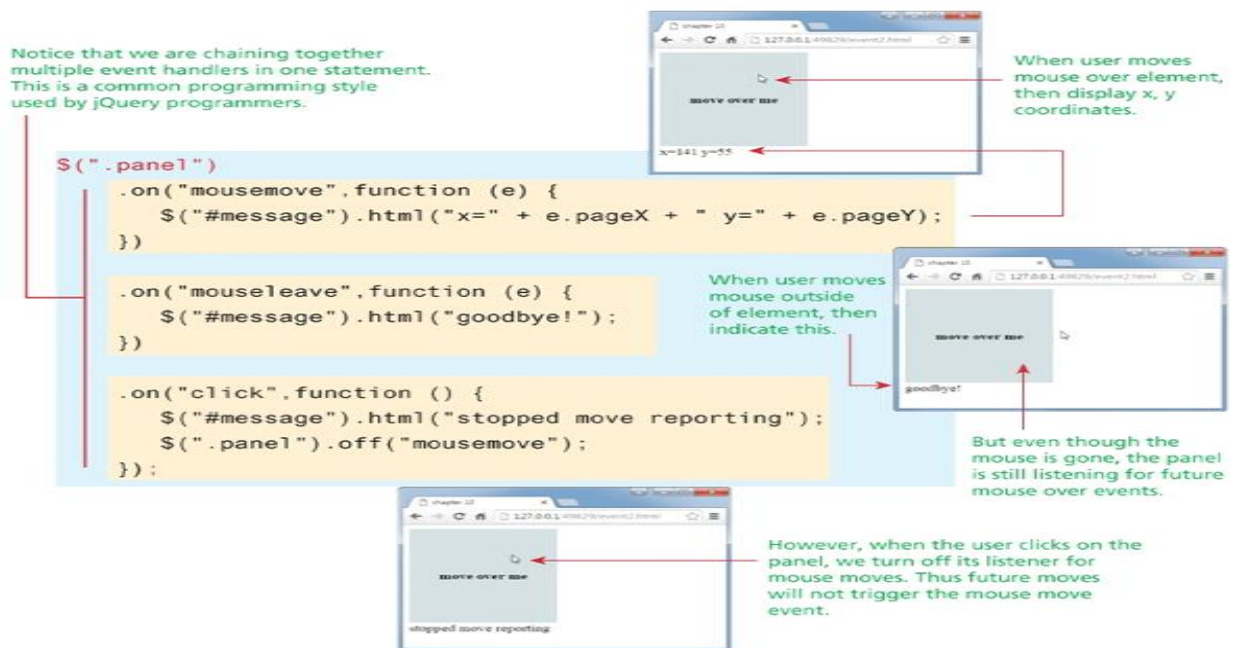


Figure 89.1.

Page loading

You learnt why having your listeners inside the window is important in JavaScript. It was a fantastic idea to use the `.addEventListener ("load",...)` event.

It did this by ensuring that the complete page and all DOM elements were loaded before attempting to connect listeners to them. With jQuery we do the same thing but use the `$(document).ready ()` event

```

$(document).ready (function () {
// set up listeners knowing page loads before this runs
$("#example").click (function () {
$("#message").html ("you clicked");
}); });
// or the even simpler
$(function () {
$("#example").click (function () {
$("#message").html ("you clicked");
}); });

```

Example:

```

//var divs =document.querySelectorAll('div');
//for (var div of divs) {
//    .. div.addEventListener ('click', div_click);
//}

```

The code above was created in java scripts, and all of the divs were accessed using java script, which was then iterated using for loop, and listeners were added to each individual listener.

Note: the above bunch of code were replaced by a single line of code in jQuery.

```

$('div').on ('click', div_click); // click event has been registered.

```

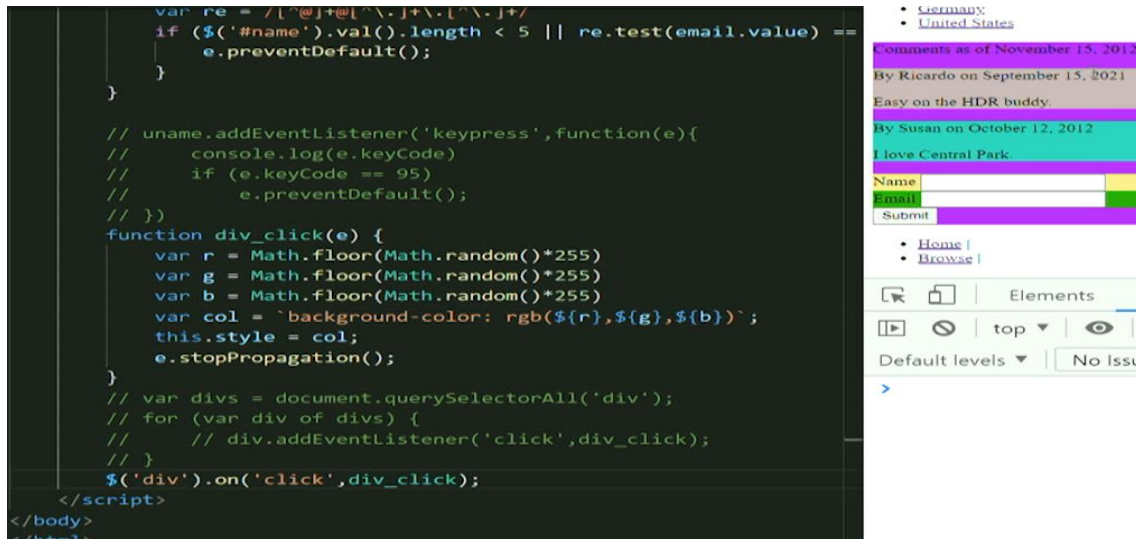


Figure 89.2: figure taken from: Fundamentals of Web Development (Second Edition, Pearson), Authors: Randy Connolly, Ricardo Hoar

Module No.90

DOM manipulation using jQuery

jQuery has a number of helpful methods for manipulating DOM elements. We've already seen how to edit the inside contents of a DOM element with the `html()` function, as well as how to modify the internal attributes and styles of an existing DOM element using the `attr()` and `css()` methods. You may recall learning several JavaScript DOM modification methods like `appendChild()`, `createElement()`, and `createTextNode()` in Chapter 9. A similar set of DOM manipulation techniques is offered by jQuery, which extends the capabilities of these native JavaScript DOM functions.

Creating Nodes

If you choose to think of your page as a DOM object, you'll want to work with the tree structure rather than just the strings. Fortunately, jQuery can automatically translate strings containing correct DOM syntax into DOM objects.

Remember that the `createElement()` method is used to create a DOM node in JavaScript:

```
var element = document.createElement('div'); //create a new DOM n
```

However, because the jQuery DOM manipulation functions accept an HTML string, jQuery objects, or DOM objects as parameters, you could opt to describe your element as follows:

```
var element = $(""); // create new DOM node
```

Instead of relying on pure JavaScript, which has fewer shortcuts, you may apply all of the jQuery functions to the object this way. If we consider creation of a simple <a> element with multiple attributes, you can see the comparison of the JavaScript and jQuery techniques in be

Comparison of node creation in JS and jQuery

Example no.90.1.

// pure JavaScript way

```
var jsLink = document.createElement("a");
jsLink.href = "http://www.funwebdev.com";
jsLink.innerHTML = "Visit Us";
jsLink.title = "JS";
```

// jQuery version 1

```
var link1 = $('<a href="http://funwebdev.com" title="jQuery">Visit Us</a>');
```

// jQuery version 2

```
var link2 = $('<a></a>');
link2.attr("href", "http://funwebdev.com");
link2.attr("title", "jQuery verbose");
link2.html("Visit Us");
```

// version 3 ... also not creating a temporary variable which // will be more typical once we start chaining methods (see next // section)

```
 $('<a>', {
    href: 'http://funwebdev.com',
    title: 'jQuery',
    text: 'Visit Us'
  });
```

As you can see, jQuery offers a variety of ways to create an element. It is important to note that running the code in above example no.90.1. has no observable effect on the browser window. The code has only done one thing: it has created DOM nodes. You must also add the elements to the DOM tree, which is discussed next, in order for them to be visible.

Adding DOM Elements

When an element is defined in any of the ways described in example no.90.1. , it must be inserted into the existing DOM tree. You can also insert the element into several places at once if you desire, since selectors return a set of DOM elements.

Inserting DOM Elements

The append () method accepts as a parameter an HTML string, a DOM object, or a jQuery object as an argument. The item is subsequently added to the element(s) being selected as the last child.

In Figure 90.1.1, we can see the effect of an append () method call. As can be seen in the figure, jQuery has a variety of additional methods for adding content to the DOM tree.

```
<div class="dest">
existing content
</div>
```

```
var link = $('<a href="http://funwebdev.com">Fun</a>');
```

```
$(".dest").append(link);
```

```
<div class="dest">
existing content
<a href="http://funwebdev.com">Fun</a>
</div>
```

```
$(".dest").prepend(link);
```

```
<div class="dest">
<a href="http://funwebdev.com">Fun</a>
existing content
</div>
```

```
link.appendTo($(".dest"));
```

```
<div class="dest">
existing content
<a href="http://funwebdev.com">Fun</a>
</div>
```

```
link.prependTo($(".dest"));
```

```
<div class="dest">
<a href="http://funwebdev.com">Fun</a>
existing content
</div>
```

```
$(".dest").before(link);
```

```
<a href="http://funwebdev.com">Fun</a>
<div class="dest">
existing content
</div>
```

```
$(".dest").after(link);
```

```
<div class="dest">
existing content
</div>
<a href="http://funwebdev.com">Fun</a>
```

```
link.insertBefore($(".dest"));
```

```
<a href="http://funwebdev.com">Fun</a>
<div class="dest">
existing content
</div>
```

```
link.insertAfter($(".dest"));
```

```
<div class="dest">
existing content
</div>
<a href="http://funwebdev.com">Fun</a>
```

Figure 90.1.1

The `appendTo()` method is similar to `append()`, except it is used in the opposite syntactic order.

If we were to use `appendTo()`, we would have to switch the object making the call and the parameter to have the same effect as `append()`.

The variable `link` is used in the example in the picture, but a more frequent technique that is functionally comparable is as follows:

```
 $('(<a href="http://funwebdev.com">Fun</a>').appendTo($(".dest"));
```

The `prepend()` and `prependTo()` methods operate in a similar manner except that they add the new element as the first child rather than the last. The diagram also illustrates the `before()` and `after()` methods for adding content before or after a specified element.

Wrapping Existing DOM in New Tags

Adding new HTML elements as needed to support some jQuery methods is one of the most common ways to improve a website that supports JavaScript. Imagine for illustration purposes our art galleries being listed alongside some external links as described by the HTML in example 90.2. below.

```
<div class="external-links">
<div class="gallery">Uffuzi Museum</div>
<div class="gallery">National Gallery</div>
```

```
<div class="link-out">funwebdev.com</div>
</div>
```

If we wanted to wrap all the gallery items in the whole page inside, another

<div> (perhaps because we wish to programmatically manipulate these items later) with class galleryLink we could write:

```
$(".gallery").wrap('<div class="galleryLink"><div>');
```

This modifies the HTML to that shown in example 90.3 below Note how each and every link is wrapped correctly in a <div> that uses the galleryLink class.

```
<div class="external-links">
  <div class="galleryLink">
    <div class="gallery">Uffuzi Museum</div>
  </div>
  <div class="galleryLink">
    <div class="gallery">National Gallery</div>
  </div>
</div>
<div class="link-out">funwebdev.com</div>
</div>
```

In a related demonstration of how succinctly jQuery can manipulate HTML, consider the situation where you wanted to add a title element to each <div> element that reflected the unique contents inside. To achieve this more sophisticated manipulation, you must pass a function as a parameter rather than a tag to the wrap() method, and that function will return a dynamically created <div> element as shown in example 90.3. below.

Using wrap() with callback to create a unique div for each element

```
$(".gallery").wrap (function () {
return "<div class='galleryLink' title='Visit ' + $(this).html() + "<div>";
});
```

You might have noticed the use of \$(this), and wondered what \$(this) meant in example 90.3. above Among jQuery developers, this is a common idiom. It's used in functions to provide a jQuery wrapper for the this keyword. You may remember from Chapter 8, that this is often a point of confusion for JavaScript developers since its meaning is completely dependent upon the context in which it is used. In example 90.3. above, the \$(this) refers to whatever \$(".gallery") object is passed to the wrap () function. The wrap () method is a callback function, which is called for each element in a set (often an array). Each element then becomes this for the duration of one of the wrap() function's executions, allowing the unique title attributes as shown in example 90.4. below.

```
<div class="external-links">
  <div class="galleryLink" title="Visit Uffuzi Museum">
    <div class="gallery">Uffuzi Museum</div>
  </div>
  <div class="galleryLink" title="Visit National Gallery">
    <div class="gallery">National Gallery</div>
  </div>
</div>
<div class="link-out">funwebdev.com</div>
</div>
```

There is an inverse method to perform the opposite objective, as there is with practically anything in jQuery.

In this case, `unwrap()` is a method that does not take any parameters and whereas `wrap()` added a parent to the selected element(s), `unwrap()` removes the selected item's parent. `WrapAll()` and `wrapInner()`, for example, provide you more control over wrapping DOM elements. The online jQuery documentation has more information on certain methods.

Module No.91

Animations using jQuery

A very powerful feature of jQuery is with the reference of Animation. Different elements could be added through animation in html document.

Animation Effects:

Some of the generic animation methods are as show and fade In methods.

Show() and fadeIn():

.show(duration)

This function will display the image gradually over given duration in milli seconds.

For example;

If image was hidden then **.show(1000)** function will gradually show the image in 1000 milli seconds (1 second).

You can also pass string parameters like slow or fast to show function.

.fadeIn(duration)

This function gradually fade the element by decreasing its opacity to given duration.

slide()

This function will move the element from one position to another.

For example;

A menu item will slide down from a button when mouse hover or enter on button element and when mouse leave the button ,sliding menu disappers.

```
<button id="menuBtn">Menu</button>
<ul id="menu">
  <li><a href="#">Menu item 1</a></li>
  <li><a href="#">Menu item 2</a></li>
  <li><a href="#">Menu item 3</a></li>
  <li><a href="#">Menu item 4</a></li>
</ul>
```

```
$(function () {
  $("#menu").hide();    When page loads, hide the list.

  $("#menuBtn").on("mouseenter", function () {
    $("#menu").slideDown(500);
  });                  Slide list down in 0.5 sec when mouse
                        hovers over it.

  $("#menuBtn").on("mouseleave", function () {
    $("#menu").slideUp(300);
  });                  Slide list up faster when mouse is no
                        longer hovering over it.

});
```

Example-1: slide() function (taken from Fundamentals of Web Development second edition 2017)



Figure-1: slide() function (taken from Fundamentals of Web Development second edition 2017)

Use animation to a limited extent, because animations take time and if users want to do work fast, animations will annoy users because of their durations.

Raw Animations in jQuery:

Along with generic animation methods of jQuery, there are some raw animations which could be used for custom type requirements. You can animate any numeric CSS property.

animate()

```
$("#box").animate({left:'495px'});
```

Function animate will move the box about 495 pixels on left side. Box animation will be visible from one position to other. The position of object must be relative not statis.

For example:

Following code example shows a box enter on the page from left side, then move upwards, and by clicking on the button in that dialogue box, the dialogue box get fade.

```

$(function() {
    $('#notifyBtn').on("click", function () {

        $('#notification')
            .animate({right: '0px', opacity: "1"}, 500)
            .animate({top: "0"});

        window.setTimeout(function() {
            dismissNotification();
        }, 4000);

    });

    $('#dismissBtn').on("click", function () {
        dismissNotification();
    });
});

```

3 Over 0.5 sec, first animate these two properties

These are the **after** values

4 and then animate this property

After 4 seconds, call the dismiss notification function

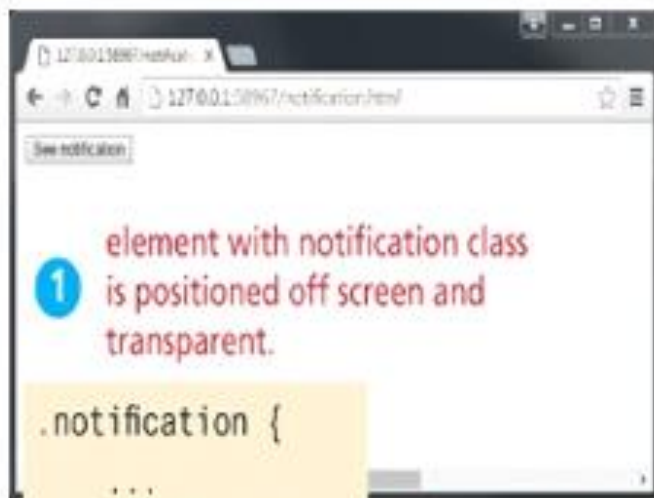
```

function dismissNotification() {
    $('#notification').fadeOut(500);
    $('#notifyBtn').fadeOut(500);
}

```

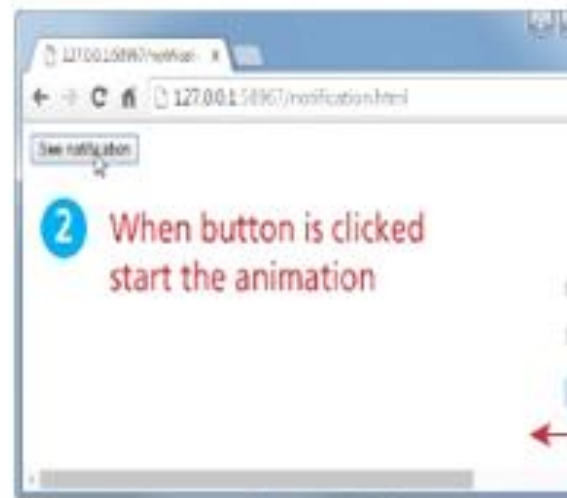
5

Example-1: animate function code (taken from Fundamentals of Web Development second edition 2017)



These are the three properties that will be animated.

These are the **before** values.



Example-1: animate function output (taken from Fundamentals of Web Development second edition 2017)

Week # 08

Module No.92

Introduction to Ajax

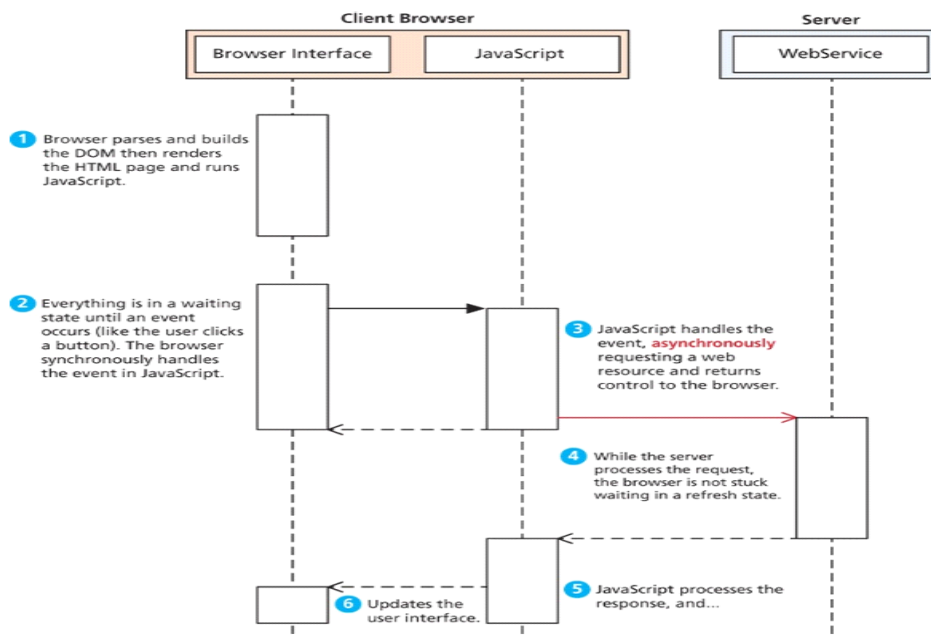
Ajax is a very important feature of JavaScript through which we fetch data from server while staying on client side.

AJAX is known as Asynchronous JavaScript and XML. Earlier it was originally using XML, but now it is using JSON format.

General Sequence of Events:

As shown in Figure 1;

- In browser Dom will be created.
- HTML document will be shown and html page will be listening for events(scroll events, click events, etc).



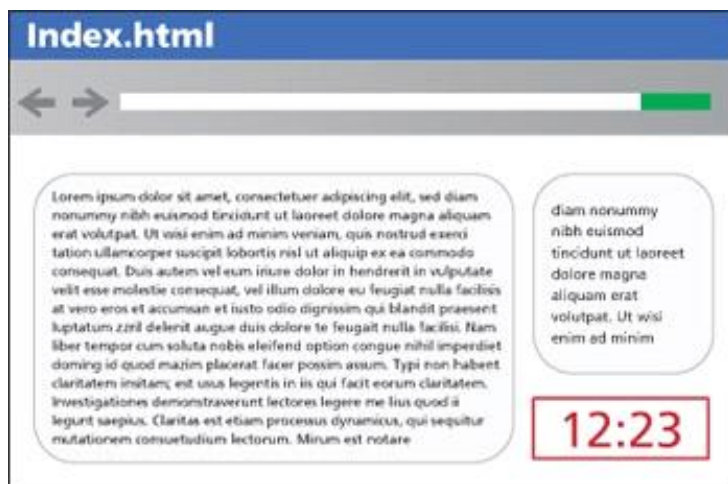
Example-1: UML sequence diagram of an AJAX request (taken from Fundamentals of Web Development second edition 2017)

Asynchronous Request:

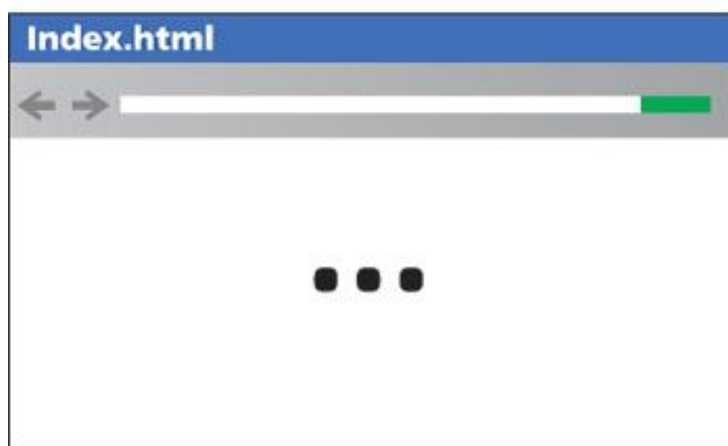
When event occur and there is the need to fetch some data from server, then we sends asynchronous request to server and server send back the request with data. We will place that data in appropriate place and then html page layout will be updated.

Synchronous Request:

The page loaded is again sent back to server with query string or other way to get information. Server creates a new web page and send it back to us. For example; we want to change the time on a web page and send a request for it to server, server change the time along with recreate all the content of web page.

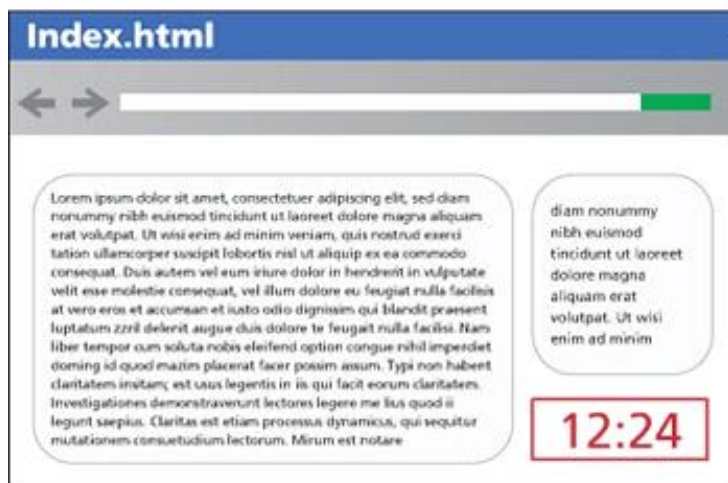


- 1 The page loads and shows the current server time as a small part of a larger page.



- 2 A synchronous JavaScript call makes an HTTP request for the "freshest" version of the page.

While waiting for the response, the browser goes into its waiting state.



- 3 The response arrives, so the browser can render the new version of the page, and the functionality in the browser is restored.

<html

</htm

Figure 10.13 Illustration of a synchronous implementation of the server time web page (taken from Fundamentals of Web Development second edition 2017)

Ajax Asynchronous Example(What's Changed?):

In ajax we send request to change time on web page to server and it only sends us the updated time. We update that time on our web page and remaining content of web page are not

changed. Ajax is very powerful in web pages because it is not updating whole page to change one element.

The mail pages, social media pages etc. are using ajax in their web pages. Just required data is fetched and updated on the browser page in ajax.

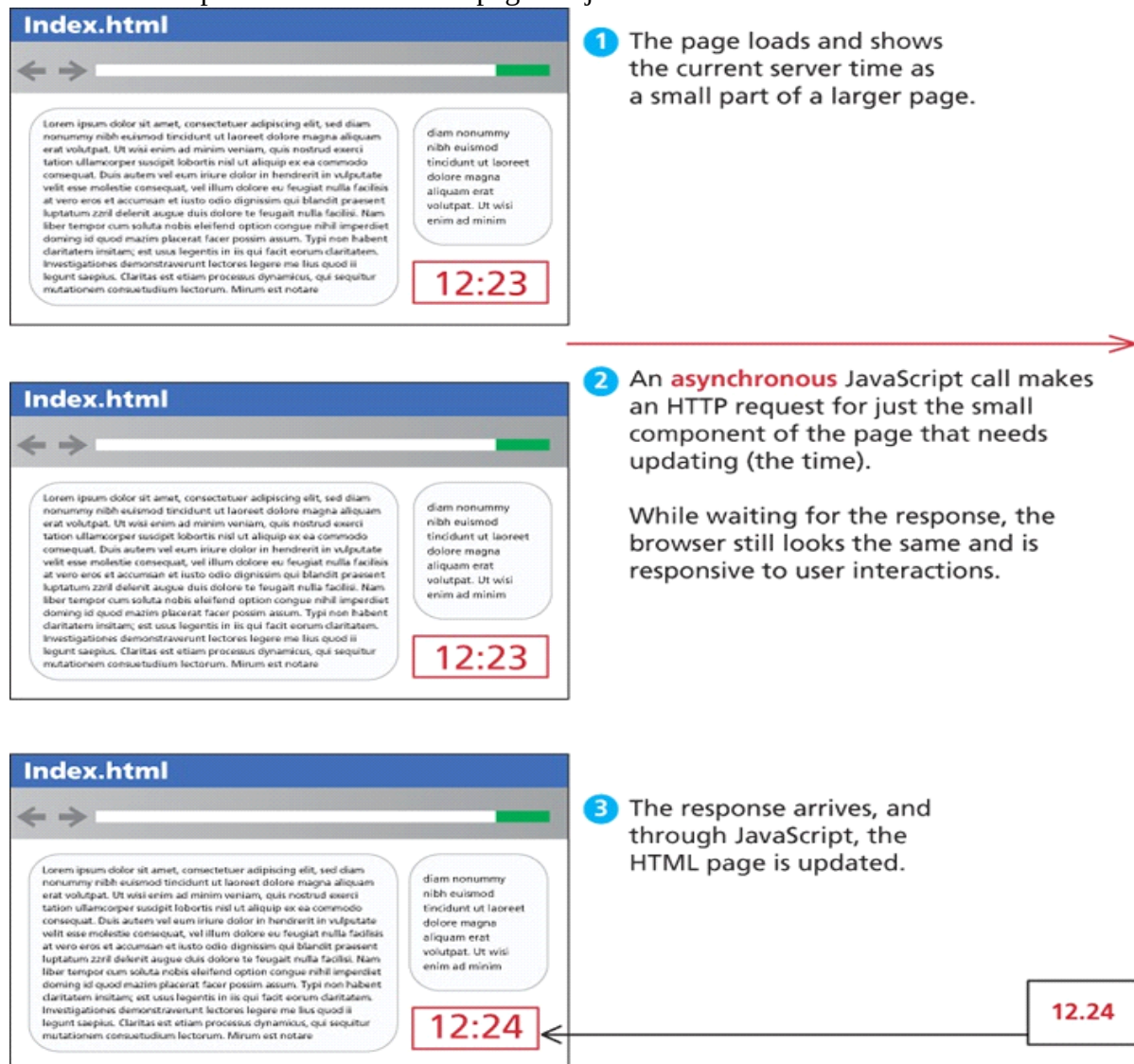


Figure 10.14 Illustration of an AJAX implementation of the server time web page
(taken from Fundamentals of Web Development second edition 2017)

Making Asynchronous Requests-load():

How we can we send asynchronous request through jQuery ?

load():

load() function is used to update content that we get from server.

```
$("#timeDiv").load("currentTime.php");
```

Here in JQuery load function will get the updated time from server-side script named currentTime.php in a single string and load that value asynchronously into the <div id="timeDiv"> element.

Making Asynchronous Requests-GET:

\$.get()

\$.get() function is used to send request to get data from server. There is a dollar sign and dot before get, that will hint as it is not an element rather it is a jQuery Ajax function.

For example;

```
$.get("serviceTravelCountries.php?name=Italy");
```

Here a url and country name is giving as parameter in query string.

The Post Method:

\$.post()

This method is used to send data on server and data is sent in parameter.

The POST Method-Form Serialization:

To send data from form to server using Ajax, we use serialization. In given code example, the serialize() method converts(encoded) the data of form into query string that would be stored in postData variable. And that variable is passed as parameter to post method to send data on server.

```
var postData = $("#someForm").serialize();
```

```
$.post("formHandler.php",postData);
```

Complete Control over Ajax:

Many things can be customized using Ajax. For example, which URL you want to access, what data is to send, whether it is asynchronous or synchronous type of request. In synchronous request you have to wait for the response of server then next statement could be executed. In asynchronous request, next statements could be run without waiting server response. In given code HTTP headers could be set in Ajax query. For example headers and referrer etc properties could be set.

```
$.ajax({ url: "vote.php",  
        data: $("#voteForm").serialize(),  
        async: true,  
        type: post,  
        headers: {"User-Agent": "Homebrew Vote Engine",  
                 "Referer": "http://funwebdev.com"  
        }  
});
```

Code 1-Listing10.16 Adding headers to an AJAX post in jQuery

Cross-Origin Resource Sharing:

The Hackers sometimes use cross origin resource sharing for hacking. By default requests related to cross origin resource sharing are disabled and server does not entertain those requests. To deal with these requests servers require allowing Access-Control-Allow-Origin requests. Through this we can fetch data from different websites using Ajax. I on server side this request is not enabled then on client side error will be shown to not have permission to access data. Such type of error could be shown when you try to fetch data from different domains or even on same domain with different urls.

Module No.93

Asynchronous file transmission

File transmission

The default way a file gets uploaded to the server is through the synchronous mode. In this mode, the user browses and select the file and upload it by pressing the **submit** button. But in Ajax, a file can be uploaded to a server in asynchronous mode as shown below

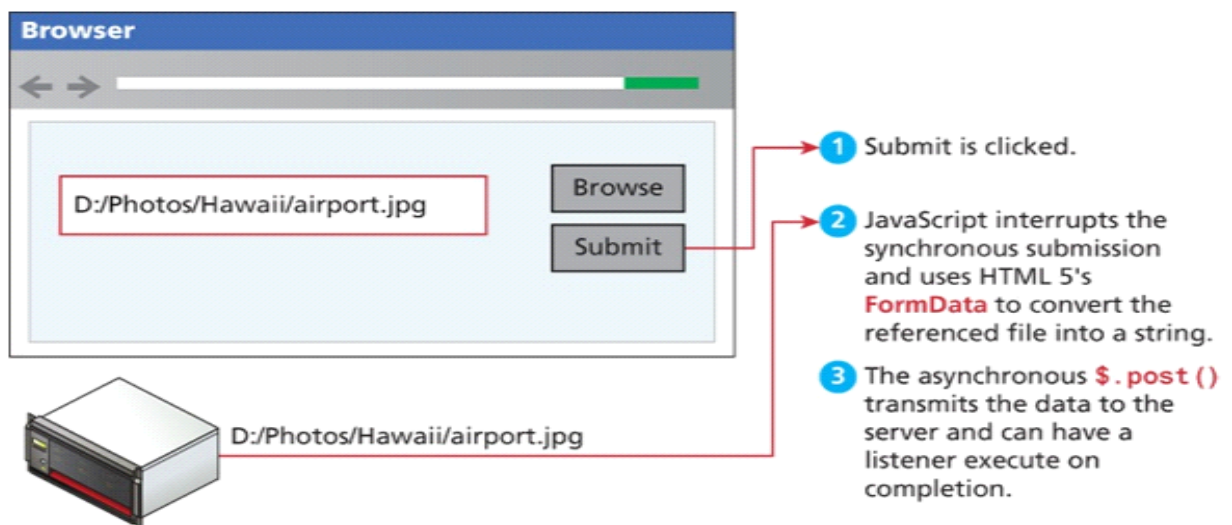


Figure: Posting a file using Form data
(Taken from fundamentals of Web Development Second Edition)

The following code uses the form data interface to post files asynchronously.

```

function uploadFile () {
    // get the file as a string
    var formData = new FormData($("#fileUpload")[0]);
    var xhr = new XMLHttpRequest();
    xhr.addEventListener("load", transferComplete, false);
    xhr.addEventListener("error", transferFailed, false);
    xhr.addEventListener("abort", transferCanceled, false);
    xhr.open('POST', 'upload.php', true);
    xhr.send(formData); // actually send the form da
    function transferComplete(evt) { // stylized upload complet
        $("#progress").css("width", "100%");
        $("#progress").html("100%");
    }
    function transferFailed(evt) {
        alert("An error occurred while transferring the file.");
    }
    function transferCanceled(evt) {
        alert("The transfer has been canceled by the user.");
    }
}

```

**Example: Using the form data interface to post files asynchronously
(Taken from fundamentals of Web Development Second Edition)**

In the upload File() function, a built-in JavaScript form object Form Data () is called which receives the name of the file to be uploaded.

Module No.94

Introduction to Server-Side Development; Comparison of Server-side technologies

Introduction:

Up till now we have covered only the client-side server technologies like HTML, CSS, JavaScript, etc. whose primary focus was on creating static web pages. Now in the server-side technology like PHP, ASP.NET, Python, JSP, etc. focuses shift on creating dynamic web pages and web sites from which the user can interact.

Client Site Scripts:

Following figure shows the execution of client site scripts

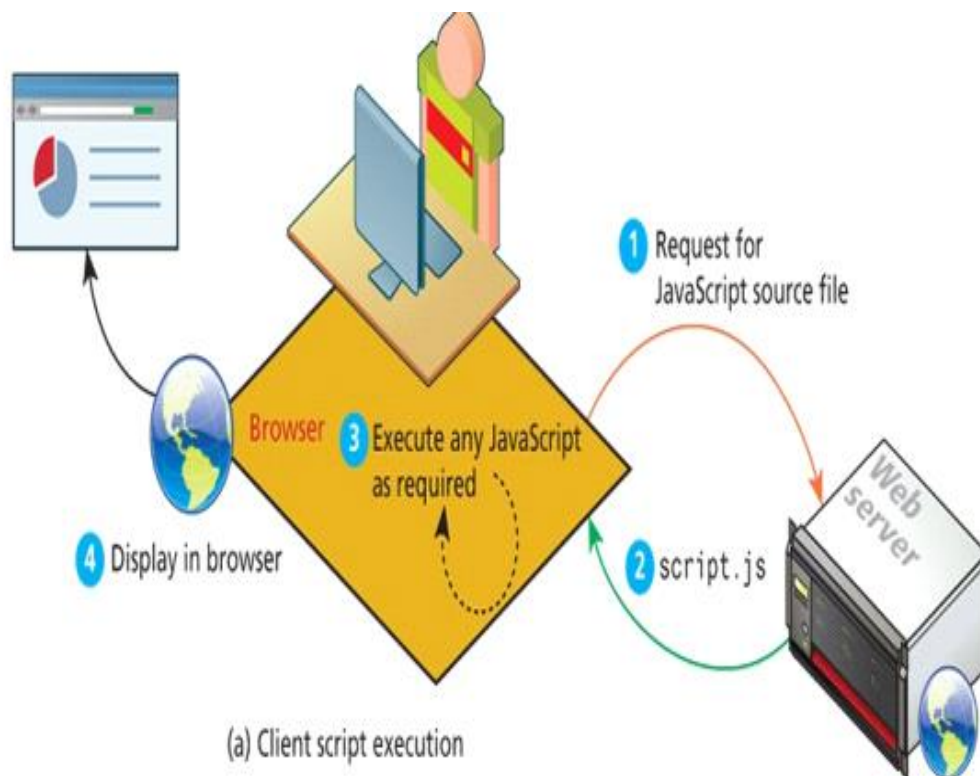


Figure: Execution of client site scripts
(Taken from fundamentals of Web Development Second Edition)

Server Site Scripts:

Similarly, the execution of the server site scripts has been illustrated in the following figure.

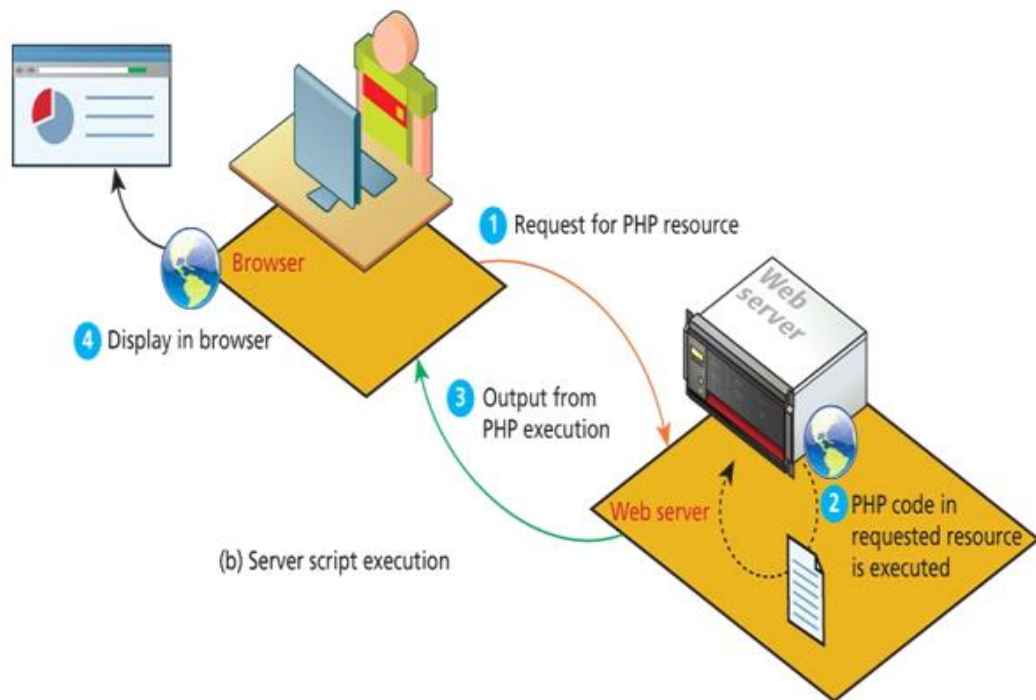


Figure: Execution of server site scripts

(Taken from fundamentals of Web Development Second Edition)

Server -Side Script Resources:

A server-side script running on a webserver can use or access multiple resources made available to it to complete its process. These resources may include a database, local files, webservices, mail server, or a different software, etc.

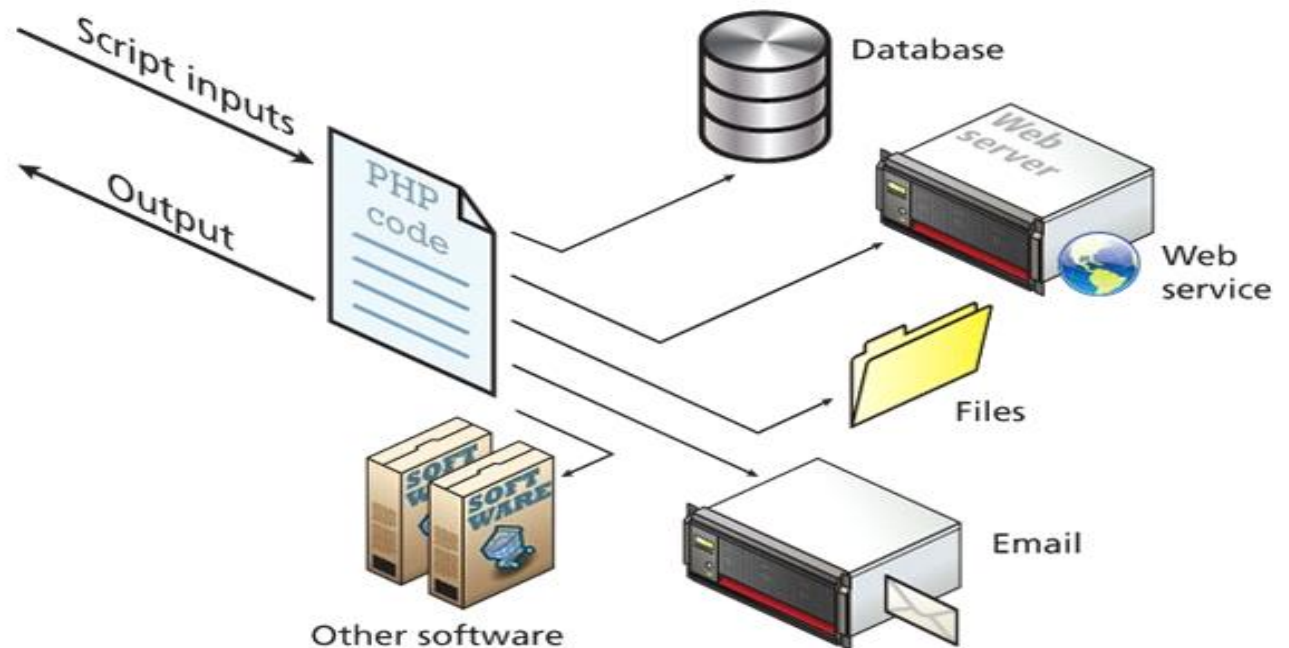


Figure: Server scripts may have access to many resources

(Taken from fundamentals of Web Development Second Edition)

Comparing Server-Side Technologies:

ASP.NET

It is a framework in which C# language is used to create web sites.

JSP

Java Server Pages uses java language as its core to create dynamic contents.

Perl

Perl is historically one of the earliest languages to create dynamic web contents.

Python

Python is an object-oriented programming language and uses framework like Django, flask to create dynamic web page contents

Ruby on Rails

Ruby is a programming language which uses Ruby on Rails a web development framework for developing websites.

Node.js

Node.js is an open-source platform that uses java script language to develop dynamic contents.

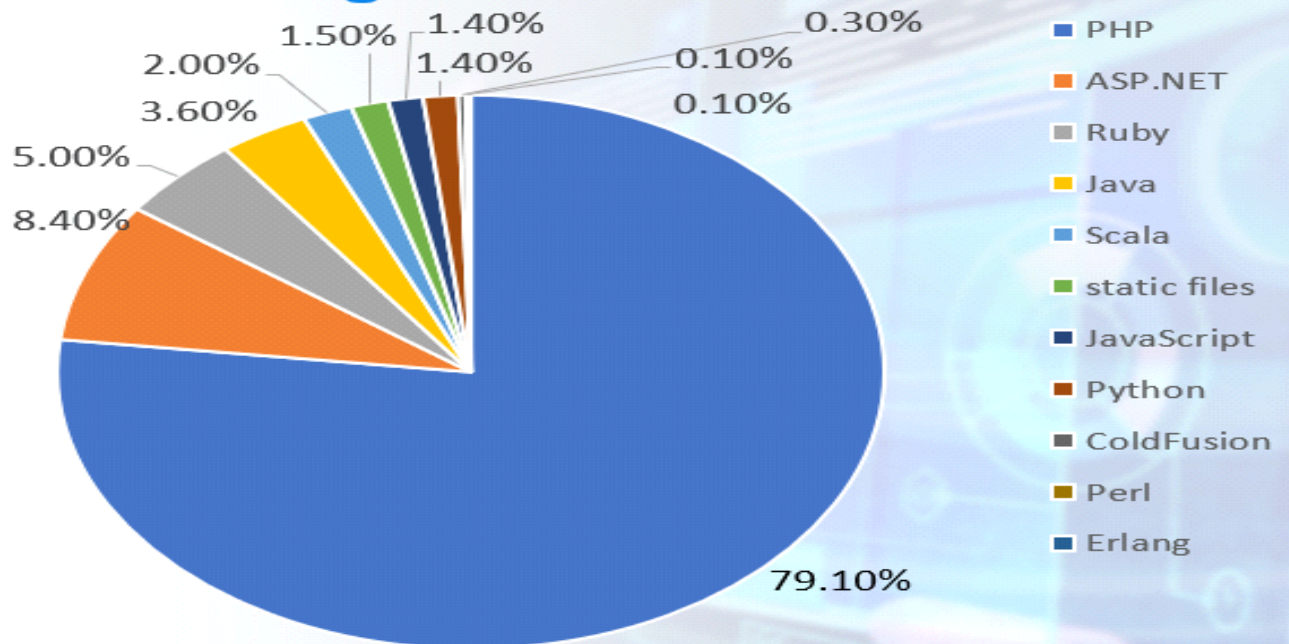
PHP

PHP (Hyper Text Preprocessor) is one of the most popular scripting languages that used to generate dynamic web contents. In this course we will be using the PHP language to create dynamic webpages.

Popularity of PHP:

In terms of popularity, the most popular programming language in creating dynamic websites is PHP. In fact, if we looked into the world wide about 79 percent of the websites uses PHP language.

Comparing Server-Side Technologies



Source: <https://w3techs.com/>

Figure: Popularity of PHP language

Taken from .ppt slides

Comparison of two Server-side scripting languages:

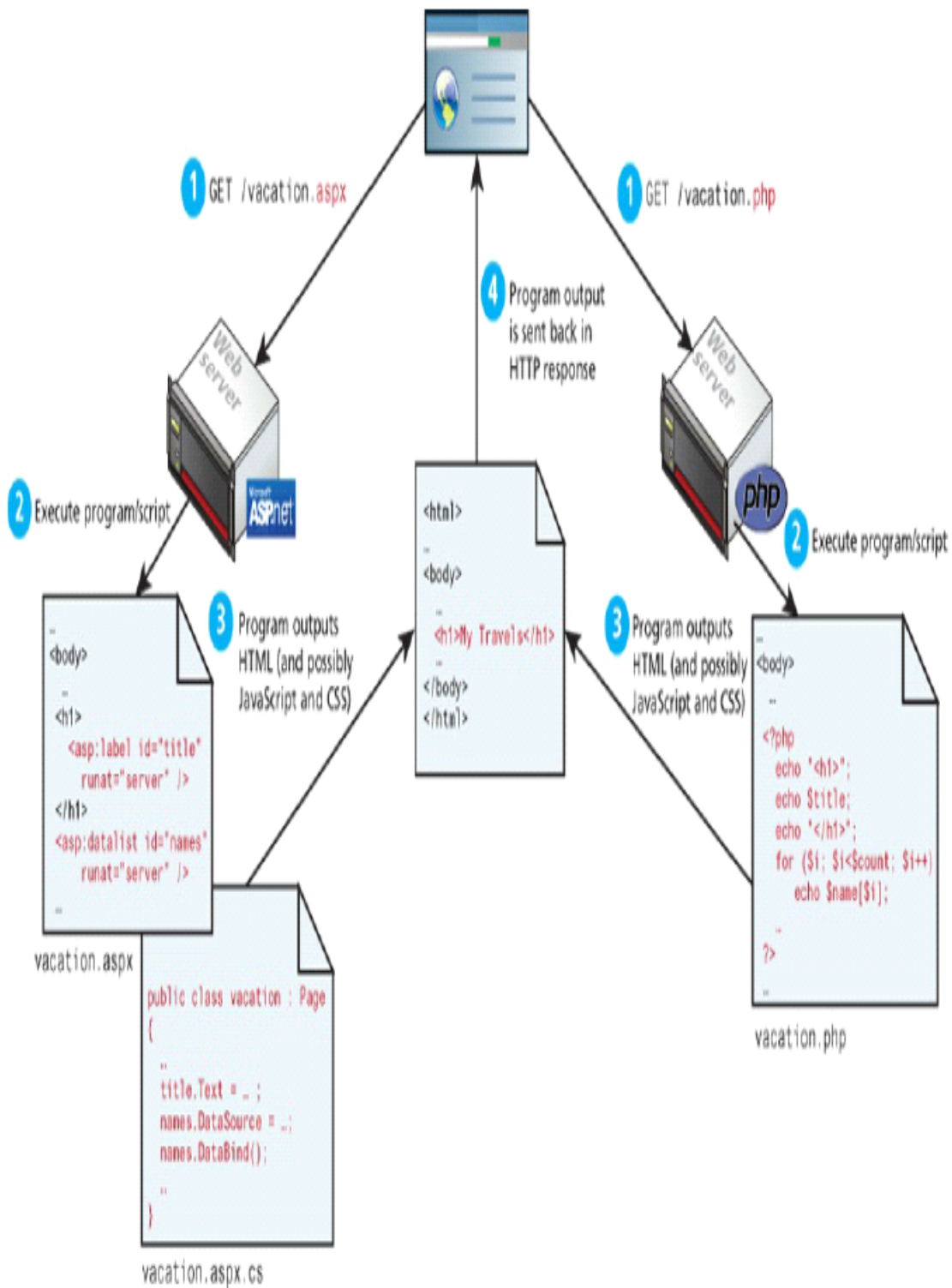


Figure: Comparison of two server-side scripting language
(Taken from Websites fundamentals of Web Development Second Edition)

Module No.95

Installation of Apache webserver, PHP and MySQL for local development

When we want to start server-side development, a set of software's are required which is also called as software stack. Minimum software's included in the Software stack are Operating System, Webserver, Programming language, Database. etc.

In this course we will be using the XAMMP server which includes the minimum number of software's to develop websites.

The XAMMP server can be downloaded from the following link:

<https://www.apachefriends.org/download.html>

Module No.96

Introduction to PHP

Introduction:

PHP is a very simple programming language and like JavaScript it is a dynamically typed language. Dynamically typed means that values of the variables in the programming language are checked at runtime. PHP also supports the object-oriented language features like C++, C#, Java etc. It provides object-oriented features like classes which is also the back bone of object-oriented programming.

PHP Tags:

PHP language provides a facility that its code can be embedded directly with in an HTML file. However, instead of having an **.html** extension, a PHP file will usually have the extension **.php** HTML, CSS, JavaScript code can also be inserted into the PHP file. PHP programming code must be contained within an opening `<?php` tag and a matching closing `?>` tag in order to differentiate it from the HTML. The programming code within the `<?php` and the `?>` tags is interpreted and executed, while any code outside the tags is echoed directly out to the client.

Variables:

PHP Variables are dynamically typed, which means that the programmer does not have to declare the data type of variable and it is determined at run time. PHP variables are also loosely type and can be assigned different data types over time.

To declare a variable in PHP a dollar sign (\$) must be placed before the name of the variable indicating to the PHP engine to handle it as a variable. Variables in PHP are assigned value from right to left i.e.

```
<?php
$var = 10;
?>
```

It should also be noted that name of the PHP variables is case sensitive i.e., variables \$var and \$Var are treated as two different variables in PHP. In PHP, variable names can also contain the underscore character, which is useful for readability reasons.

Constant:

In PHP, a constant is somewhat similar to a variable, except a constant's value never changes, in other words it says constant. A constant in PHP can be defined using the **define()** function and it should be declared at the top of the PHP file. The define() function normally takes two parameters; the name of the constant and second its value. The standard programming convention for naming constant should be in uppercase. It should also be noted that constants in PHP should be used without quotes or (\$) sign.

```
<?php
define ("DATABASE_LOCAL", "local host");
echo DATABASE_LOCAL;
?>
```

Data types:

- **Boolean** A logical true or false value
- **Integer** Whole Numbers
- **Float** Decimal Number
- **String** Letters
- **Array** A collection of data of any type
- **Object** Instances of classes

Escaping Characters:

String literals in PHP can be defined using either the single quote or the double quote character. Single quotes define everything exactly as is, and no escape sequences are expanded. If you use double quotes, then you can specify escape sequences using the backslash. For instance, the string "Good\n Morning" contains a newline character between the two words since it uses double quotes, but would actually output the slash n were it enclosed in single quotes.

The following table shows the String Escape Sequences

Sequence	Description
\n	Line Feed
\t	Horizontal tab
\\	Back Slash
\\$	Dollar Sign
\"	Double quote
\'	Single quote

Table: String Escape Sequence
(Taken from Websites fundamentals of Web Development Second Edition)

Output using echo and print:

In PHP, pages are programs that output HTML. To output something that will be seen by the browser, you can use the echo () function.

```
<?php
    echo ("hello");
?>
```

There is also an equivalent shortcut version that does not require the parentheses.

```
<?php
    echo "hello";
?>
```

Similarly, output can also be generated through the use of print statement.

```
<?php
                                                    print("Hello");
?>
```

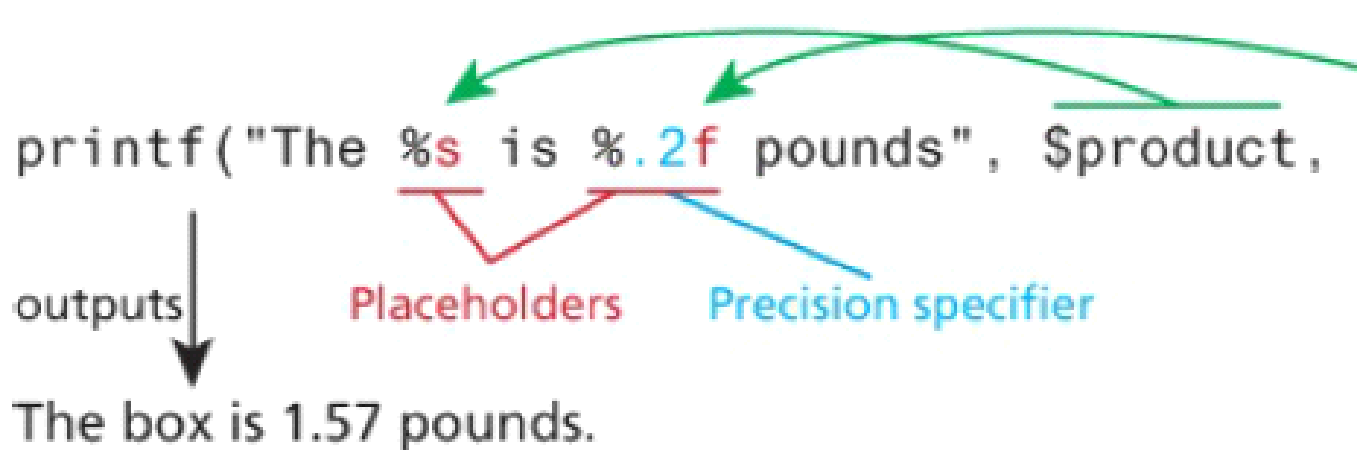
Output using printf:

In PHP, output can also be displayed using printf() function. The printf() function is derived from the same-named function in the C programming language. The function takes at least one parameter, which is a string and that string optionally take references parameters, which are then integrated into the first string by placeholder substitution.

The printf() function also allows a developer to apply special formatting, for instance, specific date/time formats or number of decimal places.

```
$product = "box";
$weight = 1.56789;

printf("The %s is %.2f pounds", $product,
```



outputs

The box is 1.57 pounds.

Figure: Illustration of components in a printf statement and output
(Taken from Websites fundamentals of Web Development Second Edition)

Module No.97

Introduction to PHP – String Concatenation

In PHP, strings can easily be appended together using the concatenation operator, which is the period (.) symbol. For example in the following code

```
<?php
```

```
$username = "Ahmed";
```

```
echo "Hello" . $username;
```

```
// outputs Hello Ahmed
```

```
?>
```

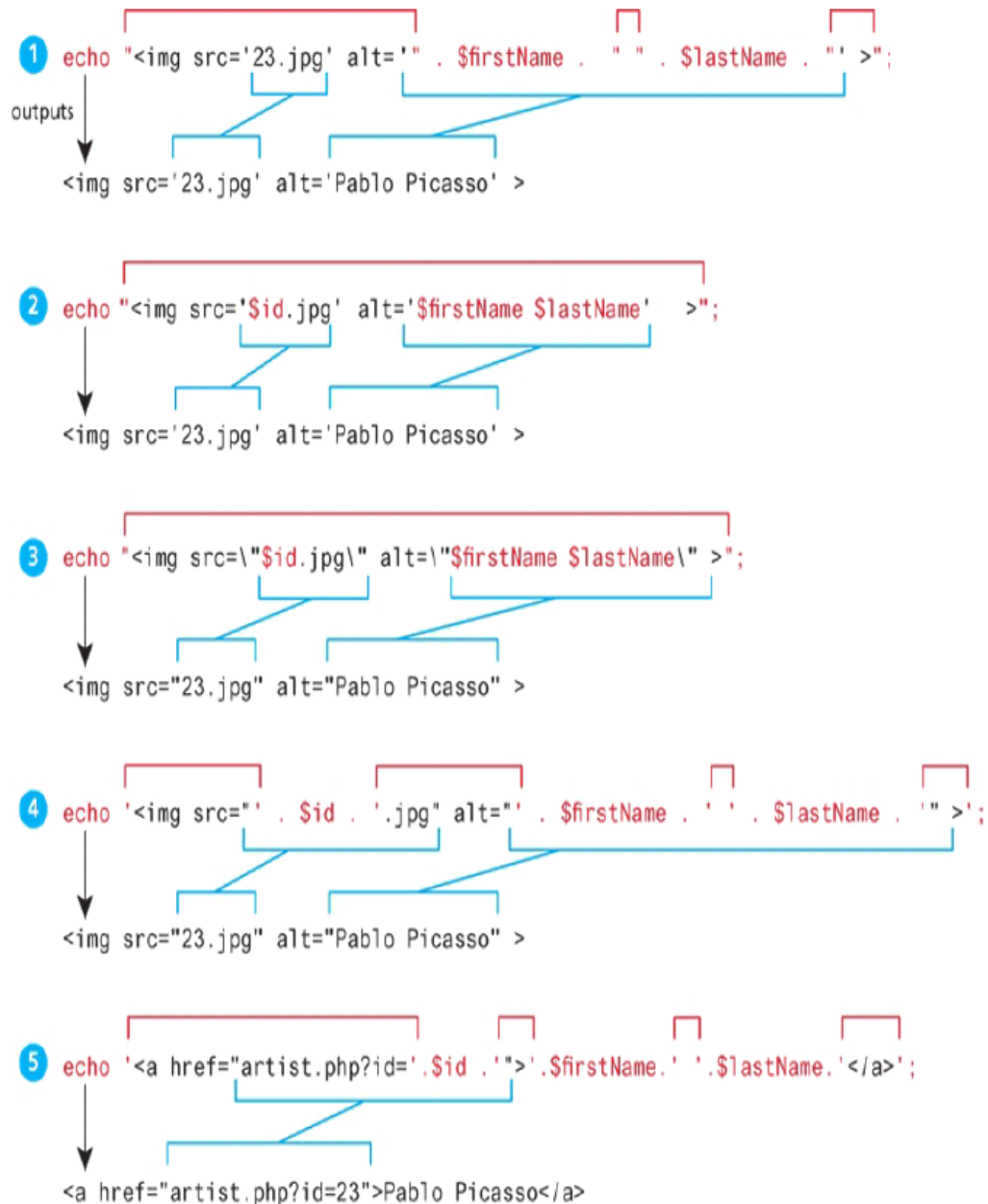


Figure: More Complicated Concatenation Examples explain
(Taken from Websites fundamentals of Web Development Second Edition)

Module No.98

Program Control – Conditionals

Just like in any other programming language PHP also uses the conditional statements in it except for few differences.

Comparison Operators:

In PHP, when doing comparison, either you can use the Equal to operator “==” or the Identical operator “===”.

- In the Equal to operator “==”, two equal operators are used i.e., `5 == '5'; // true`

In PHP, the equal to operator always check the equality of two operands. So, in the above example, PHP will internally treat number 5 without quotes as integer and number 5 with quotes as string and its comparison will be resulted as true which might lead to some logical error.

- In the Identical operator “===”, three equal operators are used i.e. `5 === '5' // false`

In PHP, the Identical operator check the equality of both the operands and their data types So, In the above example, the comparison will return as false.

if ..else:

The syntax for conditionals statements in PHP is identical to that of JavaScript. In this syntax the condition to test is contained within () brackets with the body contained in {} blocks. Optional else if statements can follow, with an optional else ending the branch.

```
// if statement
if ( $hourOfDay > 6 && $hourOfDay < 12) {
    $greeting = "Good Morning";
}
else if ($hourOfDay == 12) { // optional else if
    $greeting = "Good Noon Time";
}
else { // optional else branch
    $greeting = "Good Afternoon or Evening";
}
```

Example: Conditional snippet of code using if ..else

(Taken from Websites fundamentals of Web Development Second Edition)

Alternative Syntax for Control Structure:

It is also possible to place the body of an if or an else outside of PHP. For instance, in the below figure, an alternate form of an if ... else is illustrated (along with its equivalent PHP-only form).

```
<?php  if ($userStatus == "loggedin") {  ?>
    <a href="account.php">Account</a>
    <a href="logout.php">Logout</a>
<?php  } else {  ?>
    <a href="login.php">Login</a>
    <a href="register.php">Register</a>
<?php  }  ?>
```

Example: Combining PHP and HTML in the same script

(Taken from Websites fundamentals of Web Development Second Edition)

switch ... case:

The switch statement is similar to a series of if ... else statements. An example using switch statement is given below:

```
switch ($artType) {
    case "PT":
        $output = "Painting";
        break;
    case "SC":
        $output = "Sculpture";
        break;
    default:
        $output = "Other";
}
```

Example: Conditional statement using switch statement

(Taken from Websites fundamentals of Web Development Second Edition)

match:

PHP 8 introduces a new match expression statement in which multiple conditionals are evaluated and returns a value. Match expression statement is somewhat similar to switch

statement in PHP, except for few differences. In match statement, break statement is not required, it uses identical operator also no default case is required.

```
$output = match ($artType) {  
    "PT" => "Painting",  
    "SC" => "Sculpture",  
};
```

Figure: match statement
(Taken from .ppt slides)

Program Control – Loops

PHP also supports the same type of loops in any other programming languages. i.e., while loop, for loop and do while loop.

While and do while loop:

The while loop and the do ... while loop are quite similar. Both will execute nested statements repeatedly as long as the while expression evaluates to true. In the while loop, the condition is tested at the beginning of the loop; in the do ... while loop the condition is tested at the end of each iteration of the loop.

```
$count = 0;
while ($count < 10)
{
    echo $count;
    $count++;
}
```

```
$count = 0;
do
{
    echo $count;
    // this one increments the count by 2 each time
    $count = $count + 2;
} while ($count < 10);
```

Example: while and do while loop

(Taken from Websites fundamentals of Web Development Second Edition)

for loop:

The for loop in PHP has the same syntax as the for loop in JavaScript. The for loop contains the same loop initialization, condition, and post loop operations as in JavaScript.

```
// this one increments the count by 1 each time
for ($count=0; $count < 10; $count++)
{
    echo $count;
}
// this one increments the value by 5 each time
```

Example: the for loops

(Taken from Websites fundamentals of Web Development Second Edition)

Include Files:

PHP does have one important facility that is unlike most other non-web programming languages, namely, the ability to include or insert content from one file into another. Almost every PHP page beyond simple practice exercises makes use of this include facility. Include files provide a mechanism for reusing both markup and PHP code, as shown in the following figure.

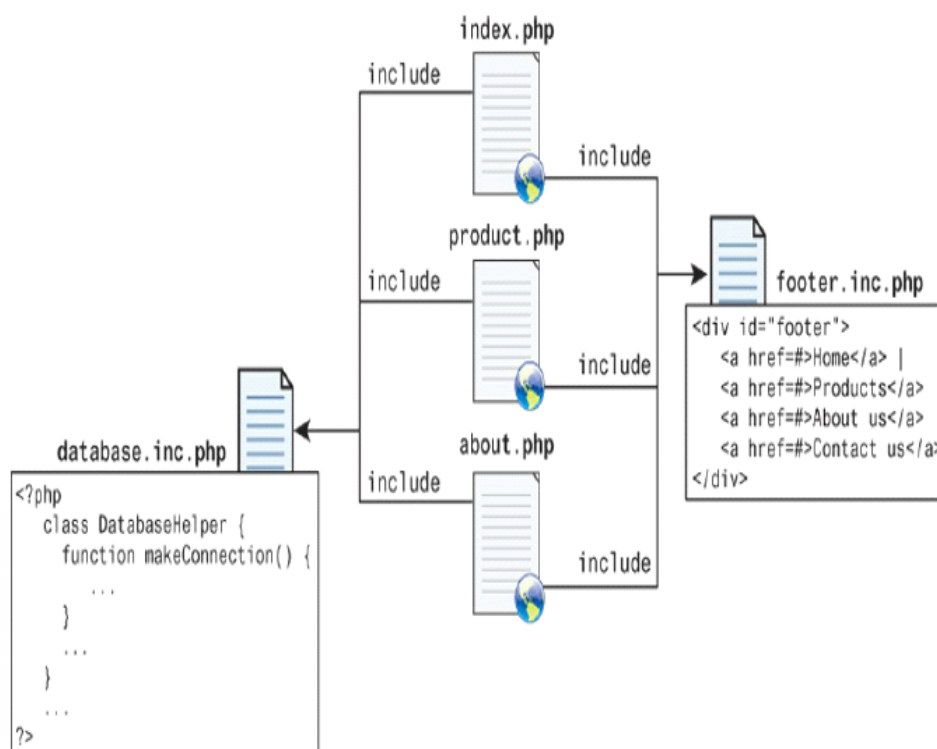


Figure: The include files

(Taken from Websites fundamentals of Web Development Second Edition)

Examples of Include files

- **Include** "somefile.php"
- **Include_once** "somefile.php"

Module No.100

Functions:

- Function starts with the keyword **function**
- Can take zero or more parameters
- May or may not return a value

```
function square($num) {  
    return $num*$num;  
}
```

```
$sq = square(5);
```

Function Syntax – Return Type Declaration

- A Return Type Declaration explicitly defines a function's return type by adding a colon and the return type after the parameter list when defining a function

```
function mustReturnString() : string {  
    return "hello";  
}
```

Parameters – Passing Parameters by Reference

- By default, arguments passed to functions are **passed by value** in PHP
- PHP also allows arguments to functions to be passed by reference, which will allow a function to change the contents of a passed variable
- The mechanism in PHP to specify that a parameter is passed by reference is to add an ampersand (&) symbol next to the parameter name in the function declaration

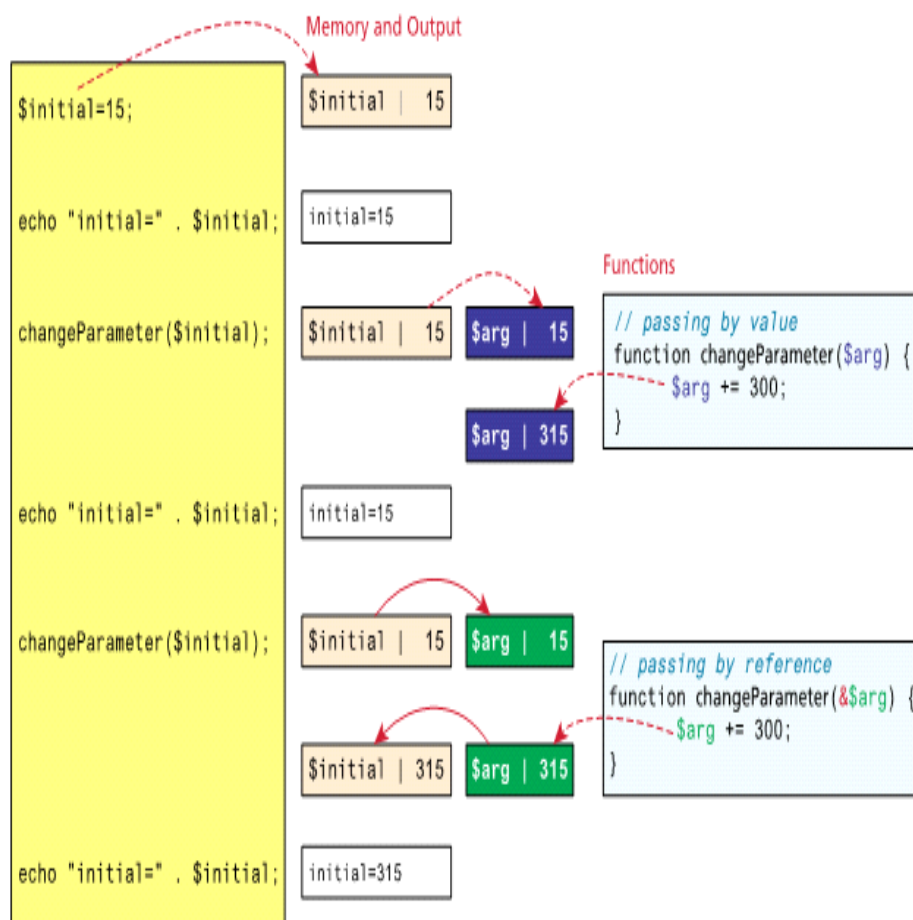


Figure taken from: Fundamentals of Web Development (Second Edition, Pearson), Authors: Randy Connolly, Ricardo Hoar

Module No.101

Functions:

Parameters – Default Values

Parameters can have default values for example consider the following function. Here default values of variables a, b, and c are 1, 2 and 3. So if the function is called without any parameter, these default values will be used.

```
function func($a=1, $b=2, $c=3) {  
    echo $a." ".$b." ".$c;  
}  
func(5,10,15); //5 10 15  
func(5,10);    //5 10 3  
func();        //1 2 3
```

Variable Scope within Functions

Variables defined within a function have function scope. While variables defined in the main script are said to have global scope, by default, not available within functions. PHP does allow variables with global scope to be accessed within a function using the **global** keyword.

```
$x=10;  
function func() {  
    global $x;  
    echo $x;  
}
```

Module No.102

Arrays:

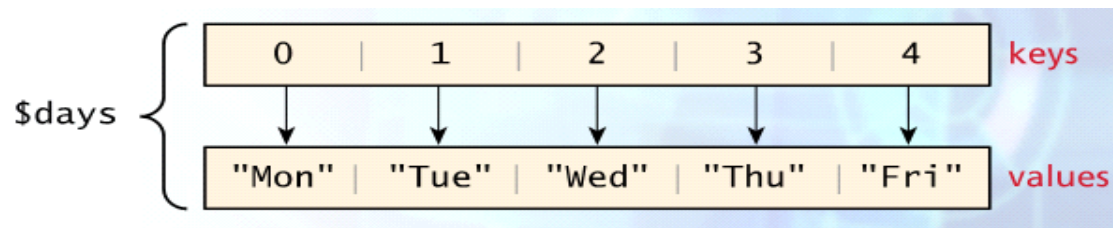
Defining and Accessing an Array

Using array function:

```
$days = array("Mon","Tue","Wed","Thu","Fri");
```

Using array notation:

```
$days = ["Mon","Tue","Wed","Thu","Fri"];
```



- For debugging, use `var_dump($days)` or `print_r($days)` to see contents of the array (or any variable). Place output in `<pre></pre>`

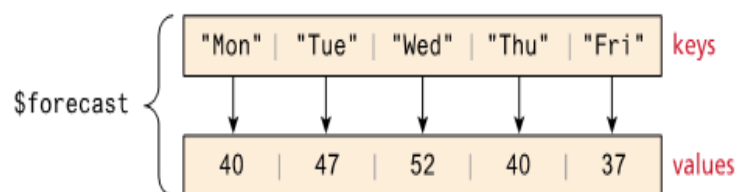
```
<pre><?php var_dump($days); ?></pre>
```

All arrays in PHP are generally referred to as **associative arrays**

```
$days = array(key0 => value"Mon", 1 => "Tue", 2 => "Wed", 3 => "Thu", 4 => "Fri");
```

You can use integer and string keys, not necessarily in order

```
$forecast = array(key"Mon" => value40, "Tue" => 47, "Wed" => 52, "Thu" => 40, "Fri" => 37);
```



```
echo $forecast["Tue"]; // outputs 47
echo $forecast["Thu"]; // outputs 40
```

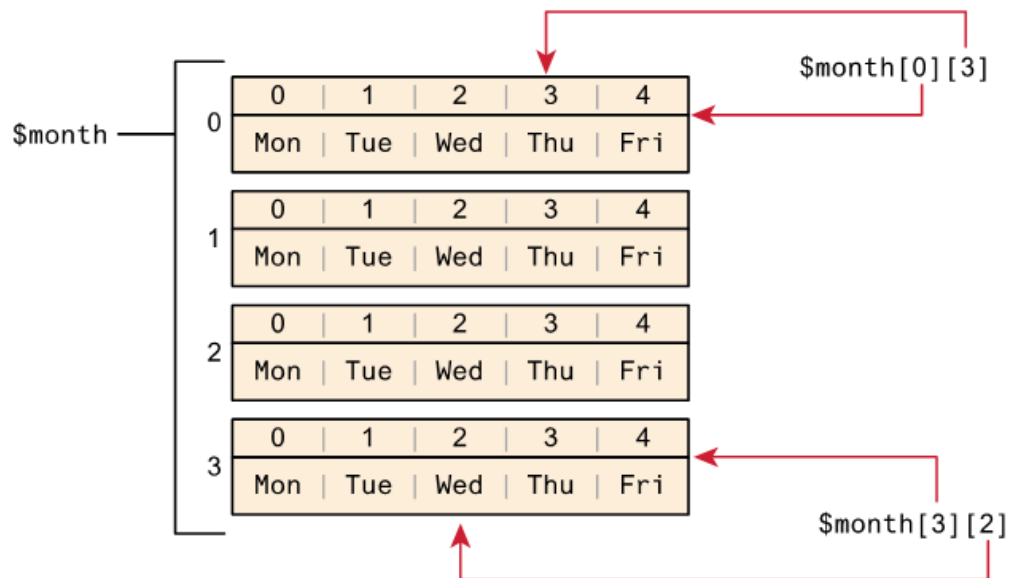
Multidimensional Arrays

A multidimensional array is an array of arrays as follows:

```
$month = array
```

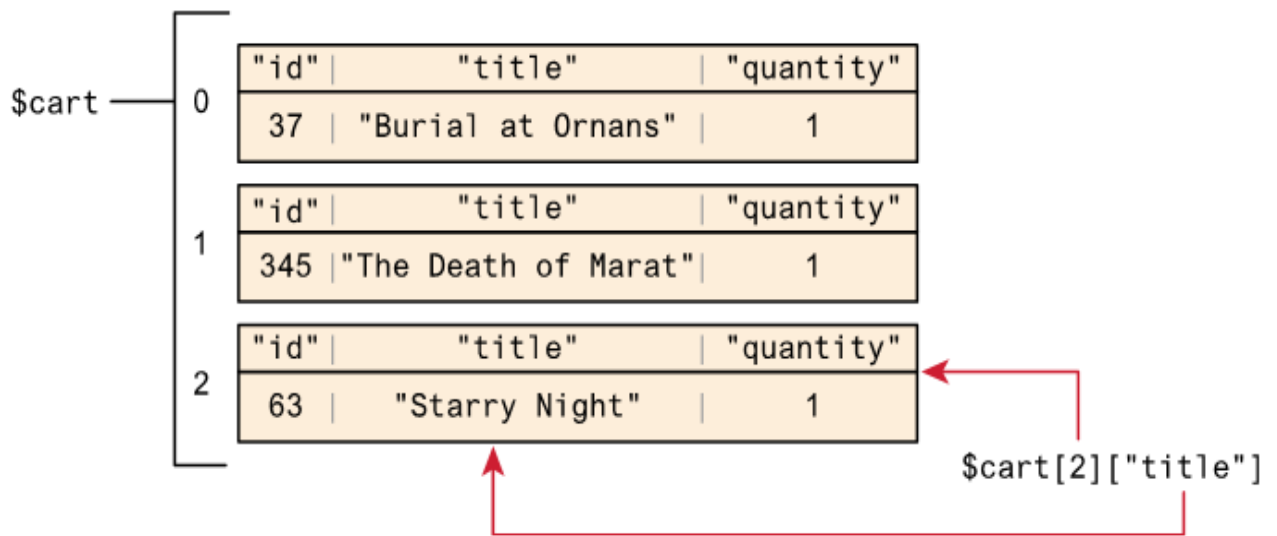
```
(
array("Mon","Tue","Wed","Thu","Fri"),
array("Mon","Tue","Wed","Thu","Fri"),
array("Mon","Tue","Wed","Thu","Fri"),
array("Mon","Tue","Wed","Thu","Fri")

```



```
);
```

```
$cart = array();
$cart[] = array("id" => 37, "title" => "Burial at Ornans", quantity" => 1);
$cart[] = array("id" => 345, "title" => "The Death of Marat", "quantity" => 1);
$cart[] = array("id" => 63, "title" => "Starry Night", "quantity" => 1);
```



Iterating through an Array - for

for loop can be used to iterate through an array as follows:

```
for ($i=0; $i<count($days); $i++) {
    echo $days[$i] . "<br>";
}
```

Iterating through an Array - foreach

Similarly can also be used to iterate through an array as follows:

//iterating through the values

```
foreach ($forecast as $value) {
    echo $value . "<br>";
}
```

// iterating through the values AND the keys

```
foreach ($forecast as $key => $value) {
    echo "day[" . $key . "]=" . $value;
}
```

Adding and Deleting Elements

An element can be added to an array simply by using a key/index that hasn't been used, as shown below:

```
$days[5]= "Sat";
```

As an alternative to specifying the index, a new element can be added to the end of any array using empty square brackets after the array name, as follows:

```
$days[]= "Sun";
```

Delete with `unset()`

Module No.103

Arrays Sorting

Arrays can be sorted using “sort” and “assort” as follows:

sort(\$days);

As the values are all strings, the resulting array would be:

Array ([0] => Fri [1] => Mon [2] => Sat [3] => Sun [4] => Thu [5] => Tue [6] => Wed)

asort(\$days);

The resulting array in this case keeps associations so is:

Array ([4] => Fri [0] => Mon [5] => Sat [6] => Sun [3] => Thu [1] => Tue [2] => Wed)

Here are some more arrays related functions:

array_keys(\$someArray)

array_values(\$someArray)

array_rand(\$someArray, \$num=1)

shuffle(\$someArray)

array_reverse(\$someArray)

array_walk(\$someArray, \$callback, \$optionalParam)

in_array(\$needle, \$haystack, \$optionalStrict)

Module No.104

Arrays – Super Globals

PHP uses special predefined associative arrays called **superglobal variables** that allow the programmer to easily access HTTP headers, query string parameters, and other commonly needed information.

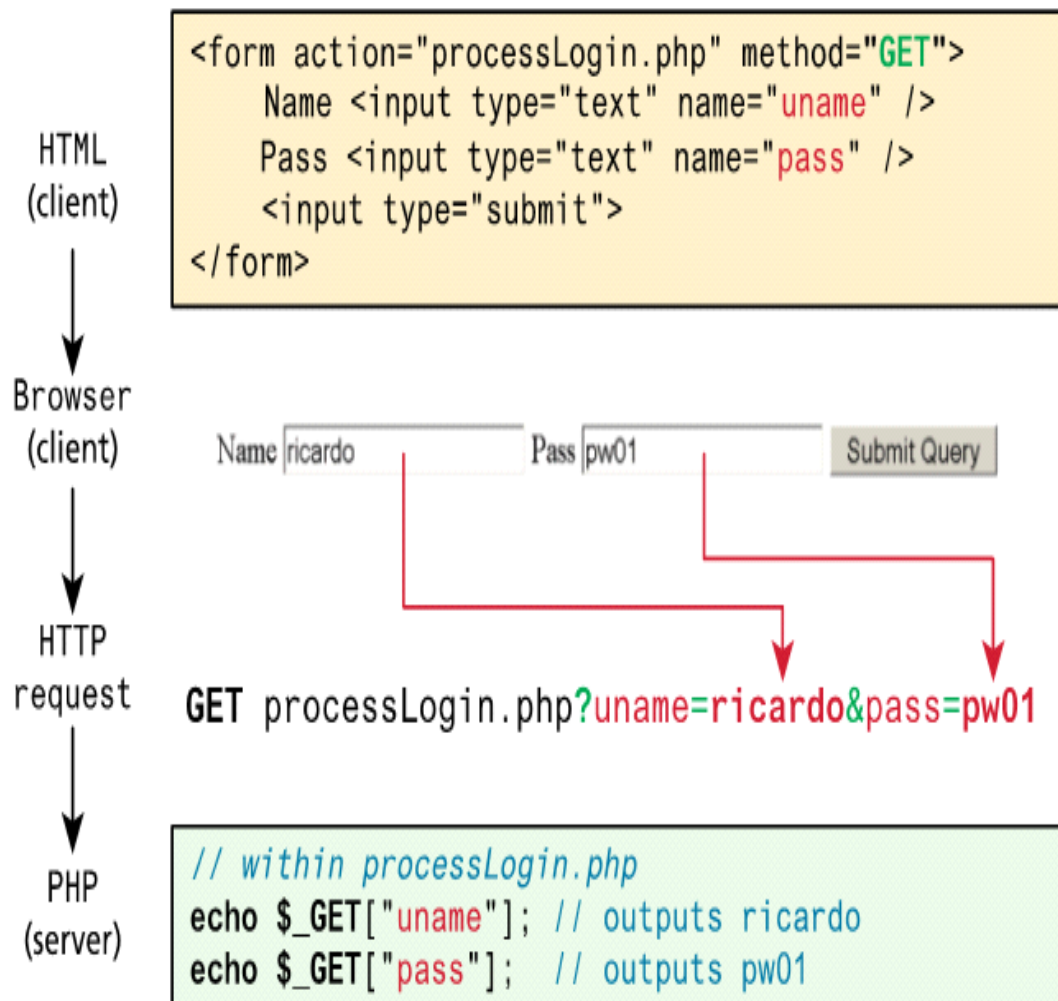
Here are some super global arrays along with the task it is used for:

- `$GLOBALS`
 - Array for reading and storing data that needs superglobal scope
- `$_COOKIE`
 - Array of cookie data passed to page via HTTP request
- `$_SESSION`
 - Array that contains session data
- `$_ENV`
 - Array of server environment data (mostly OS related and paths)
- `$_SERVER`
 - Array containing information about the request and the server
- `$_ENV`
 - Array of server environment data (mostly OS related and paths)
- `$_SERVER`
 - Array containing information about the request and the server

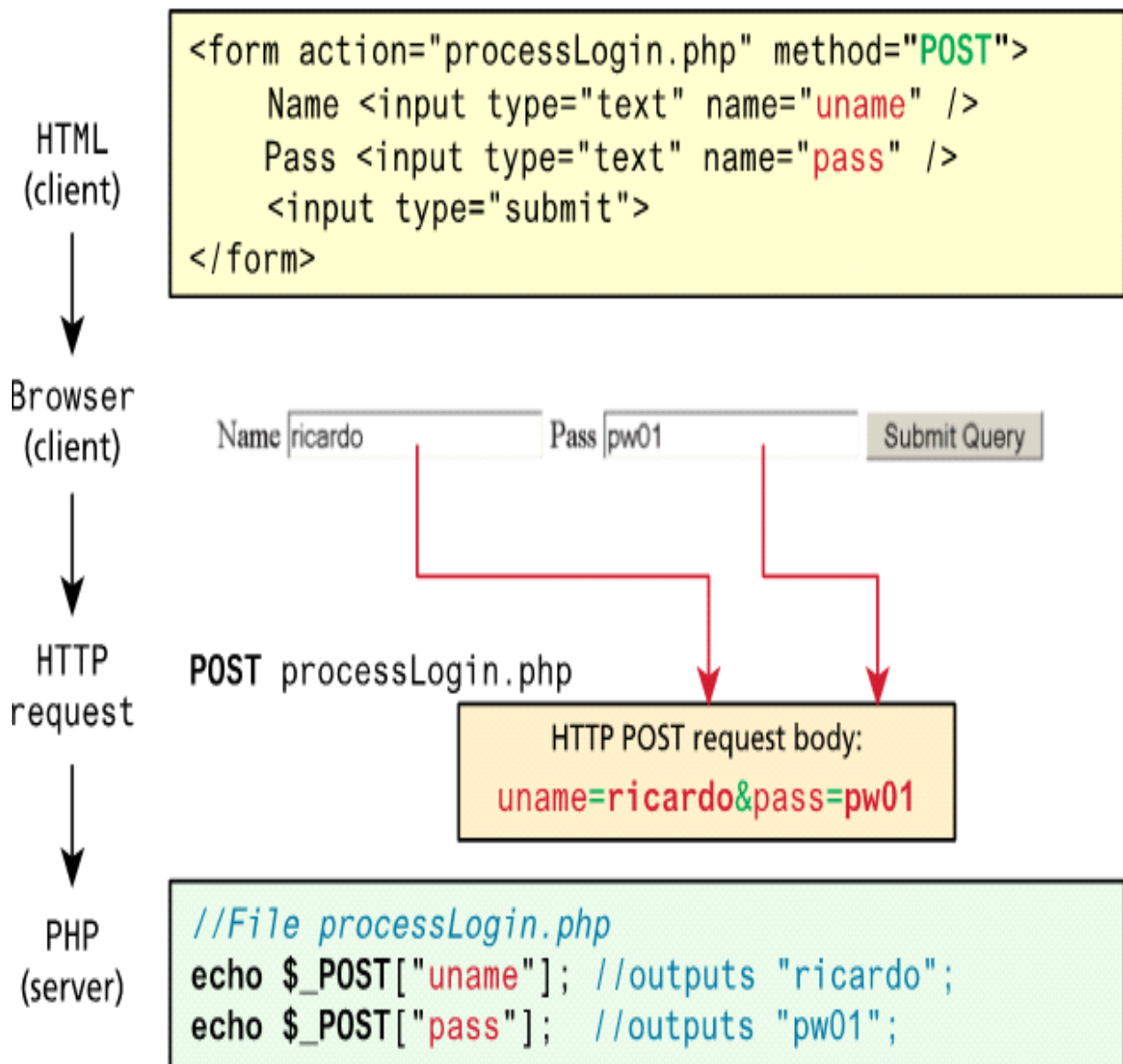
Module No.105

`$_GET` and `$_POST` superglobal arrays

Both `$_GET` and `$_POST` superglobal arrays can be used to get the data (submitted through HTML form) for further processing as shown in figure below:



Above figure shows about how we can extract data using `$_GET` array. Whereas the figure below shows the use of `$_POST` for getting the data.



Sanitizing Query Strings

Note that just because you are expecting a proper query string, it doesn't mean that you are going to get one. Your program must be able to handle:

- - If query string parameter doesn't exist
 - If query string parameter doesn't contain a value
 - If query string parameter value isn't the correct type or is out of acceptable range
 - If value is required for a database lookup, but provided value doesn't exist in the database table

Module No.106

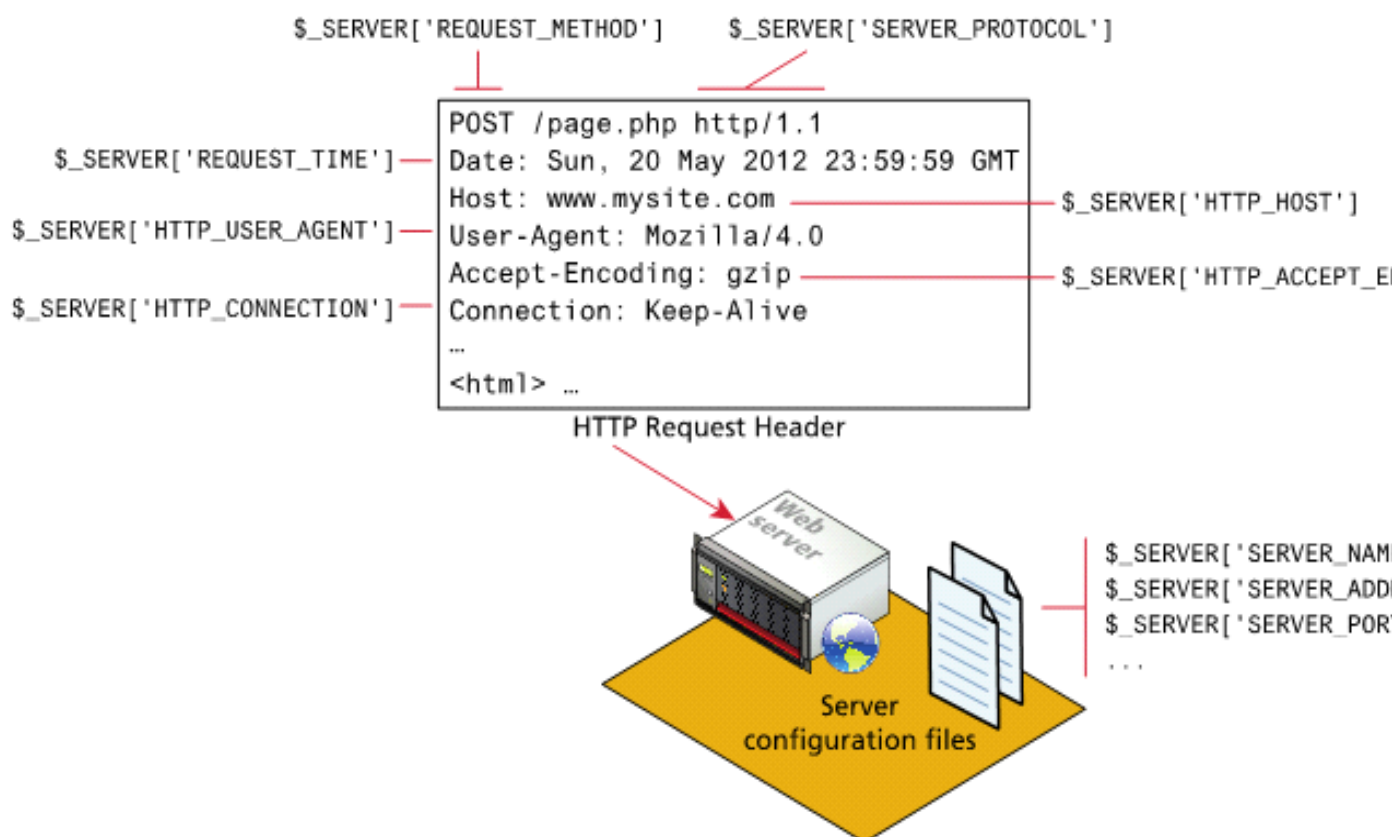
\$_GET and \$_POST hands on practice:

Please refer to video tutorial of topic 106.

Module No.107

\$_SERVER Array

Contains all the necessary information related to the current server. Lets look at the figure below



Reference: figure taken from: Fundamentals of Web Development (Second Edition, Pearson),
Authors: Randy Connolly, Ricardo Hoar

Request Header Information Keys:

Note that for advanced browser detection we need to get browscap.ini and configure php.ini [browscap] to use it.

```
<?php
echo $_SERVER['HTTP_USER_AGENT'];
$browser = get_browser($_SERVER['HTTP_USER_AGENT'], true);
print_r($browser);
?>

$previousPage = $_SERVER['HTTP_REFERER'];
// Check to see if referer was our search page
if (strpos($previousPage,"search.php") != 0) {
    echo "<a href='search.php'>Back to search</a>";
}
// Rest of HTML output
```

Module No.108

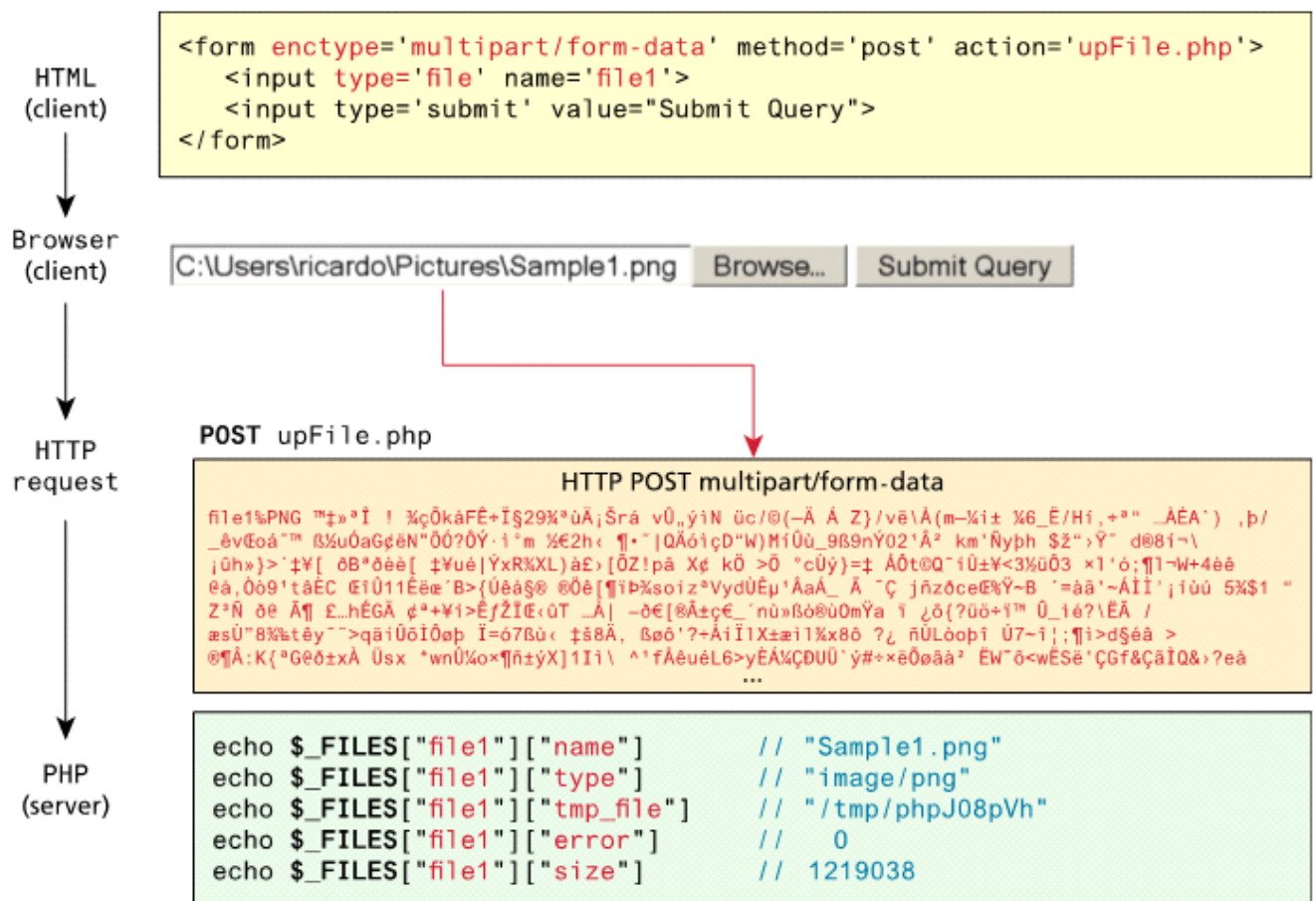
\$_FILES array

Sometimes you require uploading files through web pages. It can be done as follows:

First, you must ensure that the HTML form uses the HTTP POST method. Second, you must add the `enctype="multipart/form-data"` attribute to the HTML form that is performing the upload. Finally you must include an input type of file in your form.

```
<form enctype='multipart/form-data' method='post'>
  <input type='file' name='file1' id='file1'>
  <input type='submit' value='Submit Query'>
</form>
```

Figure below shows the flow of how the files are uploaded.



Reference: figure taken from: Fundamentals of Web Development (Second Edition, Pearson),

Authors: Randy Connolly, Ricardo Hoar

The code below shows about how we can check the error in file handling:

```
foreach ($_FILES as $fileKey => $fileArray) {
  if ($fileArray["error"] != UPLOAD_ERR_OK) { // error
    echo "Error: " . $fileKey . " has error" . $fileArray["error"] . "<br>";
  }
  else { // no error
```

```
    echo $fileKey . "Uploaded successfully ";}}
```

We can also restrict the file size that should be uploaded. It can be limited in multiple ways e.g. by using HTML form attributes in inputs (browser) or JavaScript (browser) or PHP validation (server).

The following code is an example of limiting the file type.

```
$validExt = array("jpg", "png");
$validMime = array("image/jpeg", "image/png");
foreach($_FILES as $fileKey => $fileArray ){
    $extension = end(explode(".", $fileArray["name"]));
    if (in_array($fileArray["type"], $validMime)
        && in_array($extension, $validExt)) {
        echo "All is well. Extension and mime types valid"; }
    else {
        echo $fileKey." has an invalid mime type or extension"; }}
```

Moving the file:

```
$fileToMove = $_FILES['file1']['tmp_name'];
$destination = "./upload/" . $_FILES["file1"]["name"];
if (move_uploaded_file($fileToMove,$destination)) {
    echo "The file was uploaded and moved successfully!";
}
else {
    echo "There was a problem moving the file.";}

```

Module No.109

Reading and writing files:

Mostly, when we have to save the user's data, or data we are getting from forms, we save it in the database. Because, in the database we are saving data in extracted form and it becomes easy for us to revert the data efficiently and easily. However, there are certain situations in which you have to create the files on the server side by writing and reading the data in such files. For this, we have different functions in PHP.

Before getting into these functions, let's first know about the ways of reading files. Mainly, there are two basic techniques for reading/writing files in PHP.

First is "Streaming" in which we open the file and read the data line by line, after reaching at the end of the file after reading close the file. This method is used when we have large files and loading them in memory at once is not possible.

Another method is "All -In- Memory access" in which we read the complete file in the memory at once and write the data residing in memory at once in the file. It does make the processing of files extremely easy.

How Streaming works?

Stream access works in the same way as 'fopen(), fgets() and fclose()' works in the C/C++ programming language. First, we open the file using the fopen() function and then by thorough loop read the data from the file line by line using the fgets() function, till we reach the end of the file. After reading all data from the file, close the file by fclose() function.

How In-Memory File Access works?

If you have a file of size which can be fit into memory, then we have such functions to read/write the complete file at once. The function "file()" reads the entire file and returns the file content as an array, with each array element corresponding to one line in the file, such as index [0] of the file having the first line, index [1] have the second line and so on.

Other functions are "file_get_contents()" and "file_put_contents()". file_get_contents() reads the file and stores its all content into a string variable. If we have data in string form and we want to save it into the file then we will use the function file_put_content () function and pass that string to the function to write all the data in the file.

Module No.110

Object oriented programming facilitate us to think in terms of object and program coding. So, in program we keep the object in mind and then implement the properties and methods related to object. By this way, our code becomes maintainable, making logic and program/code becomes easy as well.

To define a class, we use the keyword “class” then write the class name, after that list down the methods and properties. For example, we have a class named “Artist” and have some related information, we will write this as follows;

```
class Artist {  
    public $firstName;  
    public $lastName;  
    public $birthDate;  
    public $birthCity;  
    public $deathDate;  
}
```

Instantiating Objects:

Use the new keyword:

After making class, we create the instances of the class called object. Creating object is simple, like we use the “new” keyword with class name. By this an object is created in memory, which we save to a variable. Example is given as follows:

```
$picasso = new Artist ();  
$dali = new Artist ();
```

Here, two objects are created of class “Artist”. In OO programming we can use as many objects as required.

Properties:

Once you have instances of an object, you can access and modify the properties using the object’s variable name and an arrow (->)

```
$picasso = new Artist();  
$dali = new Artist();  
$picasso->firstName = “Pablo”;  
$picasso->lastName = “Picasso”;  
$picasso->birthCity = “Malaga”;  
$picasso->birthDate = “October 25 1881”;  
$picasso->deathDate = “April 8 1973”;
```

After creating object, its properties are accesses through “->” sign. If the property is public, we can directly store value in it, like “Pablo”. Similarly, to retrieve the value we write the object name following -> sign and then property name like \$picasso->firstName.

Module No.111

Constructor:

Before method calling, we have to initialize the object, and in OO programming, to initialize the value constructor is used. In PHP, defining of constructor is bit different as compared to C++ or Java. Here, to define the constructor of any class, we will define as functions and its name will always be “**__construct**”. By this, function will be called automatically while creating object. Constructor can be parameterized; its parameters can be taken from user or some values can be default.

```
class Artist {  
function __construct ($firstName, $lastName, $city, $birth, $death=null) {  
$this->firstName = $firstName;  
$this->lastName = $lastName;  
$this->birthCity = $city;  
$this->birthDate = $birth;  
$this->deathDate = $death;  
}  
}
```

In the above example, \$death default value is null, while all other values are taken as parameter and saving them in properties of class' objects. Here you can see that parameter of function is \$firstName and same is the property of the class, it doesn't make much difference and doesn't have to be same always.

In PHP, \$this is used for object property which represent the current object, then -> sign is used, after that property name of object is used.

```
$picasso = new Artist("Pablo","Picasso","Malaga","Oct 25,1881",  
"Apr 8,1973");  
$dali = new Artist("Salvador","Dali","Figures","May 11 1904",  
"Jan 23 1989");
```

For example, \$this. firstName = \$firstName; here any name passed through constructor will becomes property of this object. Parameter name can be different from property name, the difference between them is that the local parameter can be accessed as it is while object property is accessed through "\$this". An object is created through "new" keyword, then write class name, after that pass the parameters which the constructor is taking.

Methods:

Methods define the tasks each instance of a class can perform and are useful since they associate behavior with objects. For example, if we have information about artist and want to display it in HTML table format then we will write its method as follows:

```
class Artist {  
// ...  
public function outputAsTable() {  
// ...  
}  
}
```

To call the above method:

```
$a-> outputAsTable ();
```

Here \$a is the object of the class Artist.

Module No.112

In PHP we define the visibility of properties and methods of class. Visibility means that some properties of objects are accessible and some should be visible only to function. Similarly, some functions or methods should be accessible through object and some should not be accessible.

To define all this, we use the visibility keywords which are public and private. When we set a variable, property or method as public, we can access it through object. While if we set a variable, property or method as private, we can't access it through object. However, we can access variable or method inside the class but cannot access directly through object.

For example, we have class named "Painting", having public property "\$title" and private property "\$profit". Similarly, we have a public function "doThis()" and private function "doSecretThat()". In the class code we can access all these public and private methods and properties, like we are accessing public property "title" and private property "profit" in the following way:

```
$a = $this->profit;
```

```
$b = $this->title;
```

Similarly, method can be accessed. However, when we look towards object side that which thing we can access or not. Let say, we have created an object of painting as "\$p1 = new Paining ();" by this we can access public properties/methods like "title" and "dothis()" in following way:

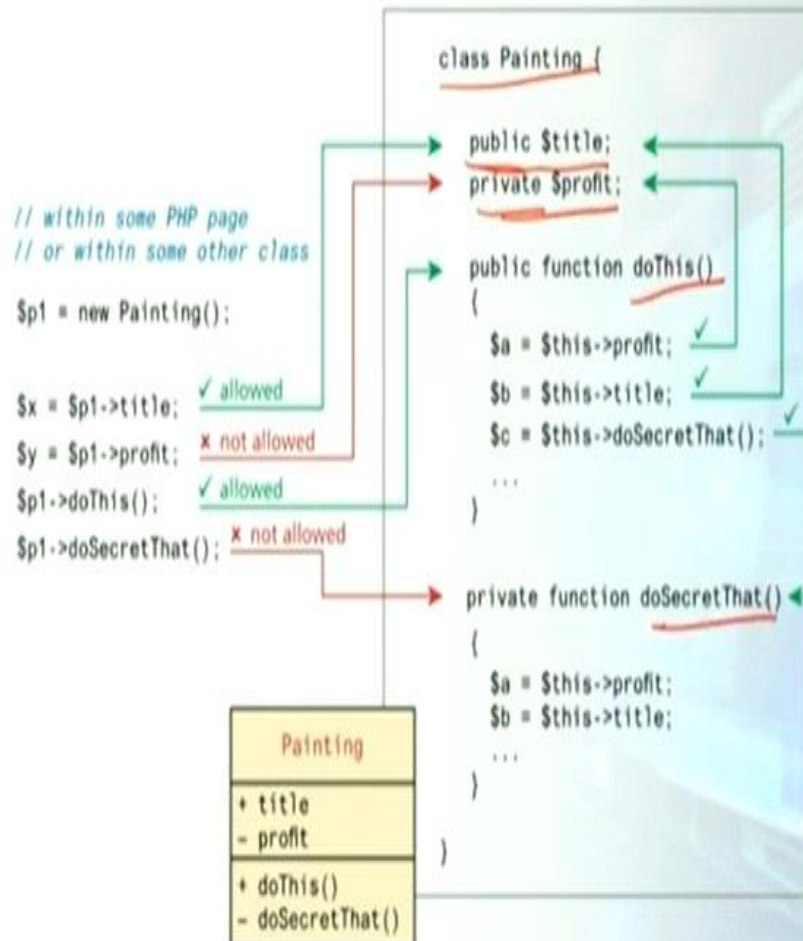
```
$x = $p1->title;
```

```
$p1->doThis();
```

But we can not access private properties or methods, such as if we try to access "profit" which is a private property, we will get an exception. Similarly, private method like "doSecretThat()" is not accessible.

Classes and Objects in PHP

Visibility



Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

So, it is advised that whenever you write program, set the visibility of methods/properties properly, it will reduce the chance of accidental bugs. Such as if you set the most of the properties as private, there is no need to access them through object, it can be accessed through functions where required. The benefit of accessing properties through function is that you will get the exact logic about the use of property in class. If you set each property as public and you make changes in the code related to object, then there is a chance that logical bugs arise accidentally. So, make sure to set the visibility of properties and methods appropriately.

Module No.113

In OO programming, there is a concept of static variable, any property or method which is set as static is shared among all objects. In fact, we can set the variable or methods as static without creating object. In PHP, if we have to set a property as static, the syntax will be as follows:

```
public static variable name = initializing value;
```

For example, in class “Artist” we will write the static property like this:

```
public static $artistCount = 0;
```

By this we can access this static property without creating an object. Similarly, the mechanism to define static function will be as follows:

```
public static method name() {  
    Statement;  
}
```

For class “Artist” we will write the static function like this:

```
public static function addArtist() {  
    self::$artistCount++;  
}
```

If we have to access the above static method, we can access this without creating any object, by using class name directly like this:

```
Artist :: addArtist();
```

And to access static variable we will write it as follows:

```
Print(Artist::$artistCount);
```

We can change its value directly, if required.

Class Constants:

Constants is another similar concept like static variable in class, the only difference is that you can change the value of static variable while constant values can't be changed.

Constants can also be accessed directly through class name like static variables. To define the constant in the class body, syntax will be as follows:

```
const constant name = 'constant value';
```

For example,

```
Const EARLIEST_DATE = 'January 1, 1200';
```

By convention write the constant name in capital letters. These can be accessed both inside and outside the class, to access in the body of the class we will write it as follows:

```
self::EARLIEST_DATE
```

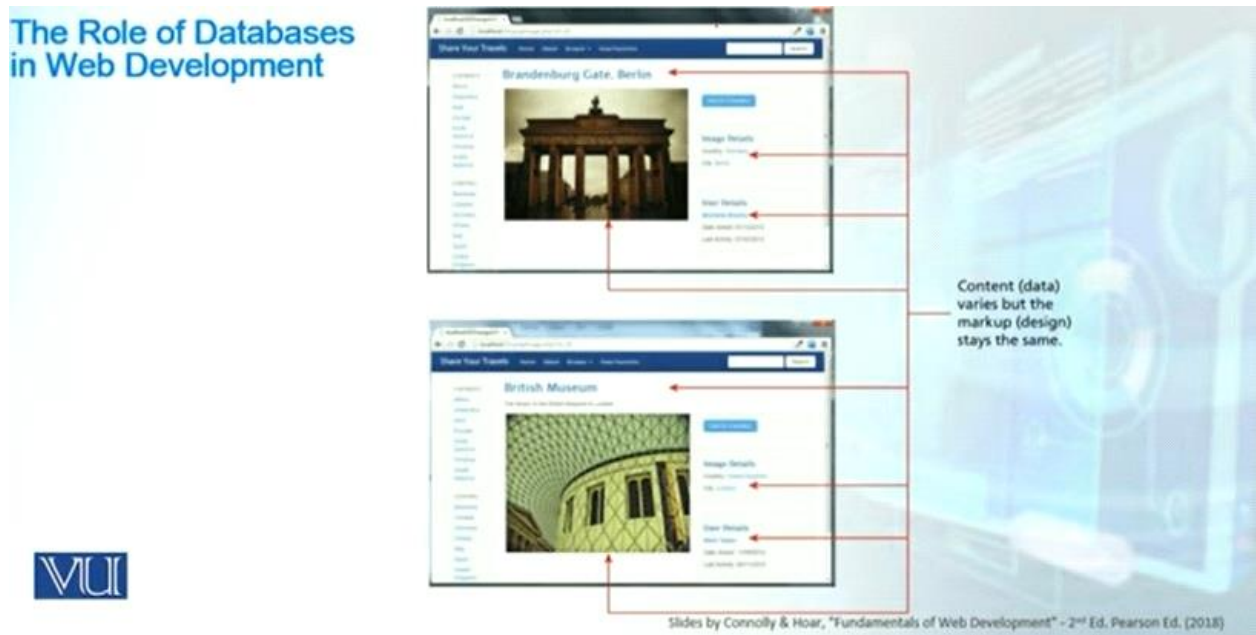
while to access it outside the class write as follows:

```
classReference::EARLIEST_DATE
```

Module No.114

The Role of Database in Web Development:

Database plays very important role in web development; all dynamic website is associated with database in some way. If we talk about only HTML or CSS pages they are static i.e., their content remains same. If we want to make our HTML pages dynamic, we can do this by server side by using any language like PHP. PHP will fetch the data from database and fit it into HTML blocks which we want to make dynamic.



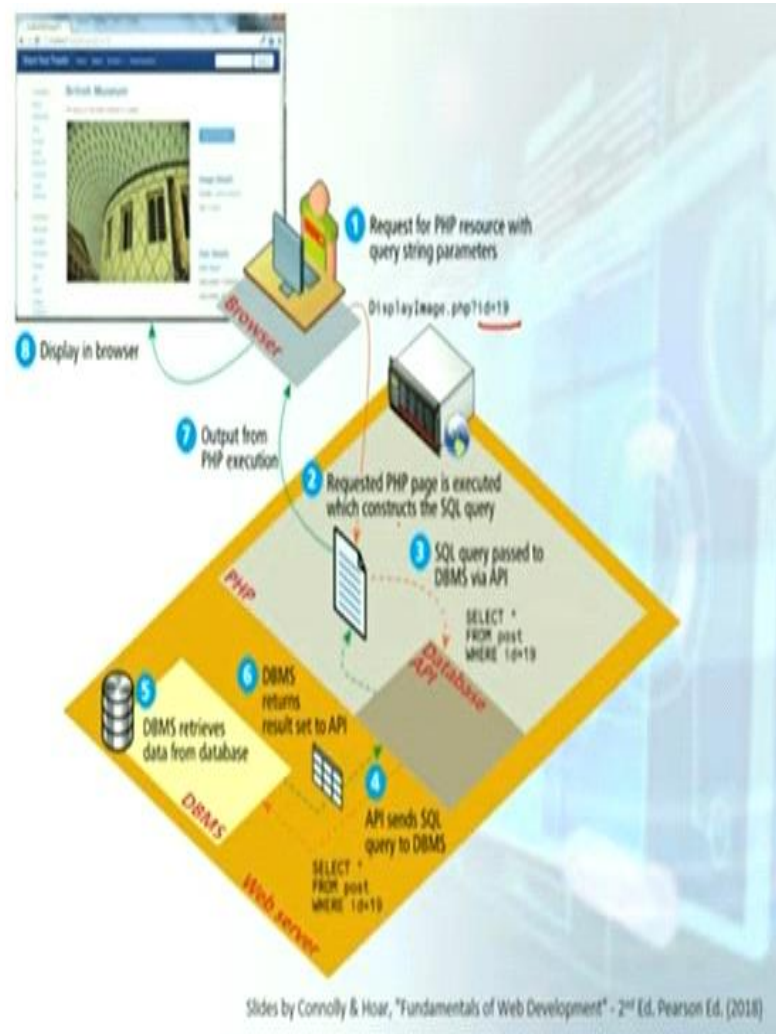
For example, above is the image of a website which have information about different places. For this the HTML, CSS and JAVA scripts would be the same for all the palaces. The static or constant part would be covered in HTML part, while dynamic contents will be fetched from database and embedded in HTML part. Here in the example website some parts are constant like header, nav bar, and layout, but there are some differences as well. The above both pages are generated by the same script.

The main idea behind is that we make a HTML template and then save the information about different places into database. After that we will fetch the information of a particular place and plugin it into the HTML template. By this, the user will see the information of different places in the same way which is generated by same script.

How Websites Use Database?

Let see how websites used database. Our user will send the request to the server site through PHP script. Mostly, one script deals with the different types of objects like in the above example, one script is displaying information of different places, with this we will pass some information to the query string.

How websites use databases

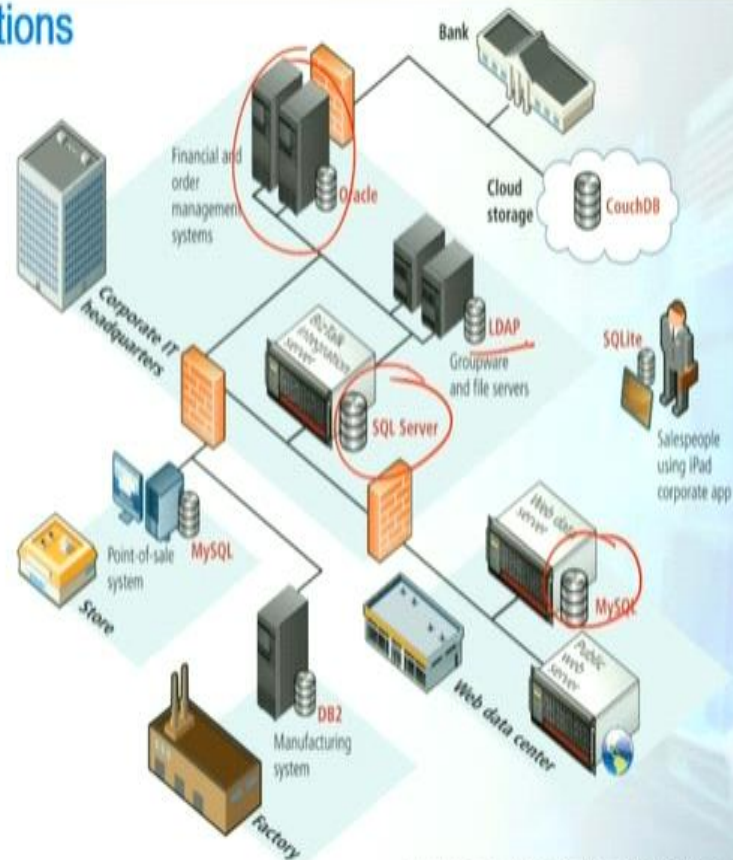


For example, here in the image given above explains display of the information about place having id “19”, we will pass information of place id as “DisplayImage.php?id=19”. This request will go to PHP script which will access the database. To access the database the server site programming language will use database API which is used to access the database. So, we will pass the required information to the API, then API will send the request to the actual database to fetch the required data. The fetched data will be passed to the script, which then embed the dynamic information into HTML, so the page will be opened at client side at browser. The user will think that all web pages are different, but at the backend script generating these webpages with different information would be the same.

Database Options:

When we talk about database we have many options, such as different organizations are using different options and at times one organizations uses multiple options. For example, some database use DBMS Oracle, some use MySQL or SQL server. If organization is using SQL server, then the information about access grant is usually kept in LDAP server.

Database Options



Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

There are other options as well, like if some staff of your organization is working on remote places where internet access is not available, then the apps of their devices will use local database, such as SQLite which is most popular and can be used in mobile apps. In certain situations, database can be placed in cloud then insert or retrieve information in database through cloud. So, there are many options when we are dealing or working with database.

Module 115

When we interact with database, most of the time its language is SQL. In SQL we use different statements.

SELECT Statement:

Most basic statement of database is “SELECT”, it is used when we have to fetch some data from the database. For example, we have table having information related to book like book title, ISBN, author information and price etc. This all information is organized columns or fields. Let’s say we want to retrieve information of each book title and ISBN, we will follow the following syntax:

SELECT filed name FROM table name;

SELECT ISBN10, Title FROM Books;

If we want to retrieve data from all fields of data then “*” is used, the query will be as follows:

SELECT * FROM Books;

If want to sort the fetched data on the bias of any filed, like we want to sort the data according to title of book then “ORDER BY” is used, the query will be as follows:

SELECT ISBN10, Title FROM Books ORDER BY title;

Sorting can be done on multiple criteria, like sorting some fields on descending order or ascending order. For example, to show the copyright in descending order and title in ascending order, the SQL query will be:

SELECT ISBN10, Title FROM Books ORDER BY CopyrightYear DESC, Title ASC;

Here, you can see that one of the filed” CopyrightYear” is not selected in the SELECT statement, this shows that you can do sorting on the basis of field if it is not selected in SELECT statement.

SELECT Statement

SQL keyword that indicates the type of query (in this case a query to retrieve data)

SQL keyword for specifying the tables

`SELECT ISBN10, Title FROM Books`

Fields to retrieve

Table to retrieve from

`SELECT * FROM Books`

Wildcard to select all fields

Note: While the wildcard is convenient, especially when testing, for production code it is usually avoided; instead of selecting every field, you should select just the fields you need.

`select iSbN10, title
FROM BOOKS
ORDER BY title`

SQL keyword to indicate sort order

Field to sort on

Note: SQL doesn't care if a command is on a single line or multiple lines, nor does it care about the case of keywords or table and field names. Line breaks and keyword capitalization are often used to aid in readability.

`SELECT ISBN10, Title FROM Books
ORDER BY CopyrightYear DESC, Title ASC`

Keywords indicating that sorting should be in descending or ascending order (which is the default)

Several sort orders can be specified: in this case the data is sorted first on year, then on title

VUI

Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

Use of WHERE Clause:

If we want to fetch data from database on some conditions then WHERE clause is used. For example, if we want to fetch the ISBN and title of books which have copy right year is after 2010 and on ward. So, the query will be:

```
SELECT isbn10, title FROM Books WHERE copyrightYear > 2010;
```

Similary , we can apply multiple conditions, for example, to fetch the record of books of maths which have copy right year 2014.

```
SELECT isbn10, title FROM Books WHERE category = 'Math' AND copyrightYear = 2014;
```

Use the WHERE Clause

```
SELECT isbn10, title FROM books  
WHERE copyrightYear > 2010
```

SQL keyword that indicates
to return only those records
whose data matches the
criteria expression

Expressions take form:
field operator value

```
SELECT isbn10, title FROM books  
WHERE category = 'Math' AND copyrightYear = 2014
```

Comparisons with strings require string
literals (single or double quote)

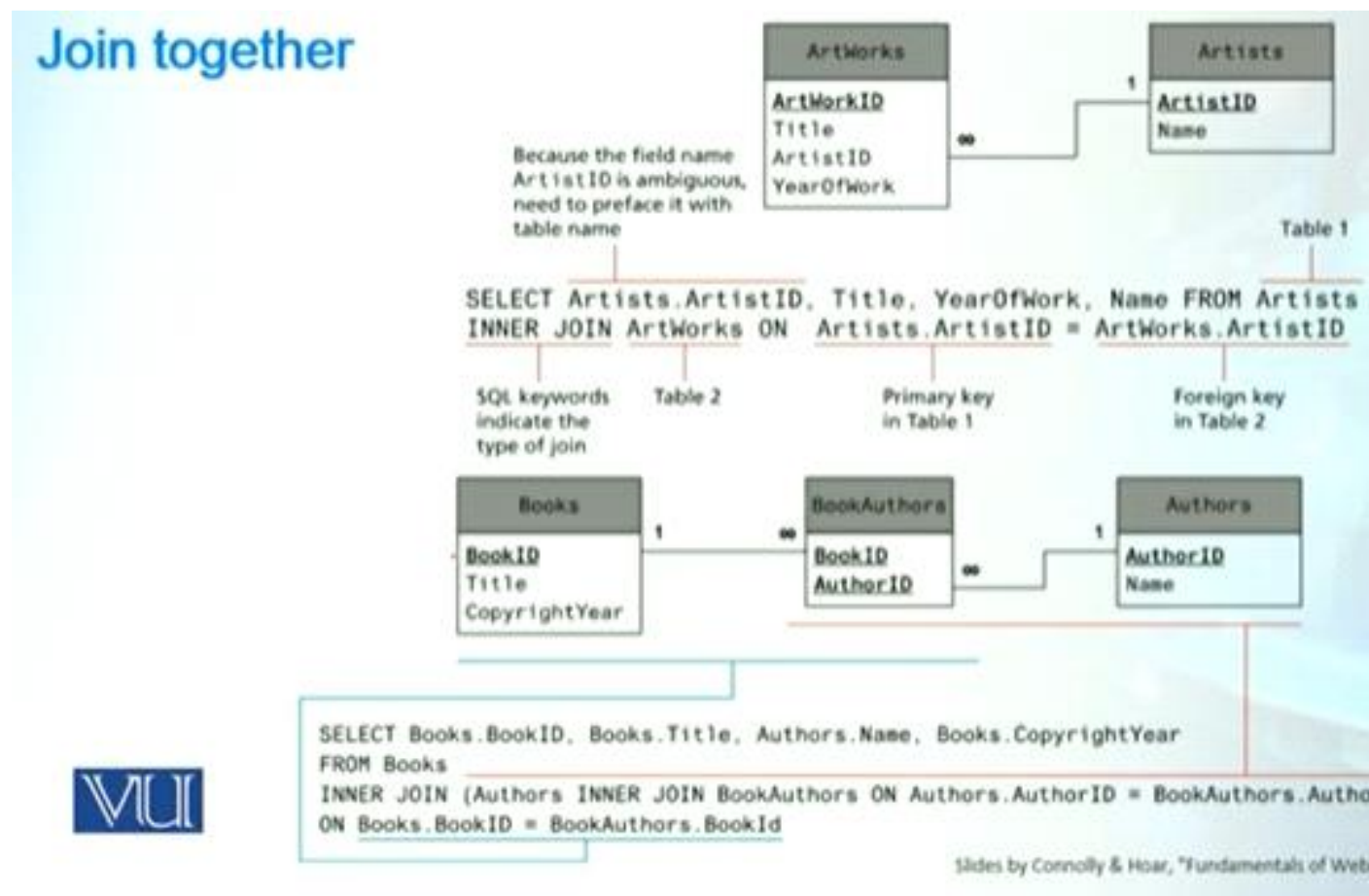
Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

Join in Database:

Sometimes, information is distributed in multiple tables. Lets say we have books , their informaiton is in Books table, book authors informations in in Authors table, then which author has wrritten which book, this informaiton will be in third table BookAuthors. Another example is that the informaiton of artist is in a table Artists, and informaiton about artwork of these artists in another table called Artworks.

In normalization we will do this to resolve some problems. If we want to fetch the information from multiple tables then in databases we use concept of joins. In this we will select the tables from where we have to fetch the data. Lets say, to fetch the data form artist table and join it to the artworks table, for this we will use INNER JOIN on Artwork table and select the common filed of both tables by using ON keyword. Like in these tables the commons field in ArtistID, so the query will be:

```
SELECT Artist.ArtistID, Title, YearOfWork, Name FROM Artists INNER JOIN ArtWorks ON
Artists.ArtistID = ArtWorks.ArtistID;
```



Simliary, join can be on two or more tables. For example, to join three table related to books, the Books and Book authors will join on the bais of book ID, while Book authos and authors joins on the bais of author ID. The query for this will be like:

```
SELECT Books.BookID, Books.Title, Authors.Name, Books.CopyrightYear FROM Books INNER
JOIN (Authors INNER JOIN BookAuthors ON Authors.AuthorID = BookAuthors.AuthorId) ON
Books.BookID = BookAuthors.BookId;
```

Member Group By:

Another important concpet in database is of grouping. Lets say, we want to see that that how many artworks in the table of ArtWorks are made after year 1900. For this we will write SELECT statement, count is used as agggregation function to count the number of paintings. The query will be:

```
SELECT Count(ArtWorksID) AS NumPaintings FROM ArtWorks WHERE YearOfWork > 1900;
```

If we want to make this search specific to artist that which artist has produce how many artworks then we will use GROUP BY clause. The query will be:

```
SELECT Nationality, Count(ArtistID) AS NumArtists FROM Artists GROUP BY Nationality;
```

Here, grouping is done on the basis of nationality, so all records are grouped in the base of nationality so we will get the number of artworks of each nation.

Member Group By

This aggregate function returns a count of the number of records

Defines an alias for the calculated value

```
SELECT Count(ArtWorkID) AS NumPaintings
FROM ArtWorks
WHERE YearOfWork > 1900
```

Count number of paintings after year 1900

Note: This SQL statement returns a single value with a single value.

NumPaintings
745


```
SELECT Nationality, Count(ArtistID) AS NumArtists
FROM Artists
GROUP BY Nationality
```

SQL keywords to group output by specified fields

Note: This SQL statement returns many records as there are unique values in the group-by field.

Nationality	NumArtists
Belgium	4
England	15
France	36
Germany	27
Italy	53



Slides by Connolly & Hoar, "Fundamentals of W"

INSERT, UPDATE, and DELETE Statements:

To store the information in database, INSERT statement is used. The syntax for this query will be:

```
INSERT INTO table name (name of fields) VALUES (values to be inserted);
```

For example,

```
INSERT INTO ArtWorks (Title, YearOfWork, ArtistID) VALUES ('Night Watch', 1642, 105);
```

It is not mandatory to mention all field, only mention those fields name in which you want to insert data. Another syntax for this query is as follows:

```
INSERT INTO table name SET filed name = vlaue, filed name = vlaue....;
```

For example,

```
INSERT INTO ArtWorks SET Title = 'Night Watch', YearOfWork=1642, ArtistID=105;
```

Similarly, to update the record, UPDATE statement is used, syntax is:

UPDATE table name SET field name = vlaue, filed name = vlaue WHERE record to be updated.

WHERE in used to specify that for which record, we want this updation. It is mandatory to use WHERE clause, if you don't use this clause all related record in your database will be updated.

For Example,

UPDATE ArtWorks SET Title='Night Watch', YearOfWork=1642, ArtistID =105 WHERE ArtWorkID=54;

INSERT, UPDATE, and DELETE Statements

INSERT INTO ArtWorks (Title, YearOfWork, ArtistID) VALUES ('Night Watch', 1642, 105)

SQL keywords for inserting (adding) a new record: INSERT INTO
Table name: ArtWorks
Fields that will receive the data values: (Title, YearOfWork, ArtistID)
Values to be inserted. Note that string values must be within quotes (single or double): ('Night Watch', 1642, 105)
Note: Primary key fields are often set to AUTO_INCREMENT which means the DBMS will set it to a unique value when a new record is inserted.

INSERT INTO ArtWorks SET Title='Night Watch', YearOfWork=1642, ArtistID=105

Nonstandard alternate MySQL syntax, which is useful when inserting record with many fields (less likely to insert wrong data into a field).

UPDATE ArtWorks SET Title='Night Watch', YearOfWork=1642, ArtistID=105 WHERE ArtWorkID=54

It is essential to specify which record to update, otherwise it will update all the records!
Specify the values for each updated field.
Note: Primary key fields that are AUTO_INCREMENT cannot have their values updated.

DELETE FROM ArtWorks WHERE ArtWorkID=54

It is essential to specify which record to delete, otherwise it will delete all the records!

Slides by Connolly & Hoar, "Fundamentals of Web"

To delete a specific record, DELETE statement is used. For Example,

DELETE FORM ArtWorks WHERE ArtWordID=54.

Similar to UPDATE statement, you have to use the WHERE clause in DELETE statement, otherwise all your records will be deleted.

Transactions:

Another important concept in SQL is of transactions, specially when you are working with critical systems like banking's or with collection of statements where successful execution of all statements is important. Mechanism of transaction is simple, you start the transaction and then write statements, and successful execution is must, if any of the statements fails to execute then you will stop this

transaction and none of the statement will be executed. If you don't use transaction this will results in inconsistency of data as some statements will be executed and some will fail to execute.

Transactions

- By starting the transaction, all database modifications within the transaction will only be permanently saved in the database if they all work

START TRANSACTION

INSERT INTO orders . . .

INSERT INTO orderDetails . . .

UPDATE inventory . . .

/* if we have made it here everything has worked so commit changes */

COMMIT

/* if we replace **COMMIT** with **ROLLBACK** then the three database changes would be "undone" */

Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

In transaction if all statements executes successfully then you will COMMIT this and if all statement doesn't execute then you will use ROLLBACK.

Module 116

In web development, we have different options for databases, sometimes we are given option to choose the database, while sometime we have to use the given database. Most popular database used in web development is MySQL, which is now known as MariaDB. But the software stack used for this, it has MariaDB at backend, but the tool to access this or the code we write in PHP to interact with database is the same as used in MySQL.

Command-Line Interface:

There are multiple ways to access the MySQL, when you install the database, a command line utility will be installed with this which is called “Command-Line Interface”. This utility works irrespective of Window, Linux or MAC. Through this utility we connect with database and using command interact with database.

To launch an interactive MySQL command-line session, you must specify the host, username, and database name to connect. The command will be as follows:

mysql -h 192.168.1.14 -u bookUser -p

In windows to connect with database you have to access file named “mysql.exe”, path to access the “mysql” file is **Xamp>MySQL>Bin>mysql.exe**. In the same folder, there will be another file name “mysqld.exe”, it is server file so it cannot be used for connection. To connect with database, you need to provide some parameters like host such as “-h 192.168.1.14”. This host can be an IP address where the DB server is hosted, it can be local host or 127.0.0.1 if database server is installed on the same machine from where you are accessing it.

Another parameter is user name of database like “-u bookUser” and then password after “-p” but it is not recommended, if you give password, it will become part of command line history. However, if you write only “-p”, then utility on command prompt will ask for password, in case of valid password you will be connected to database and a command line interface will be shown.

Pre-defined statements can also be executed through command line utility. Let's say these statements are saved in “commands.sql”, which is the text file having SQL queries. To execute the queries, write the “<” after passing parameters and then after a space write the name of file having statements to be executed on server. For example,

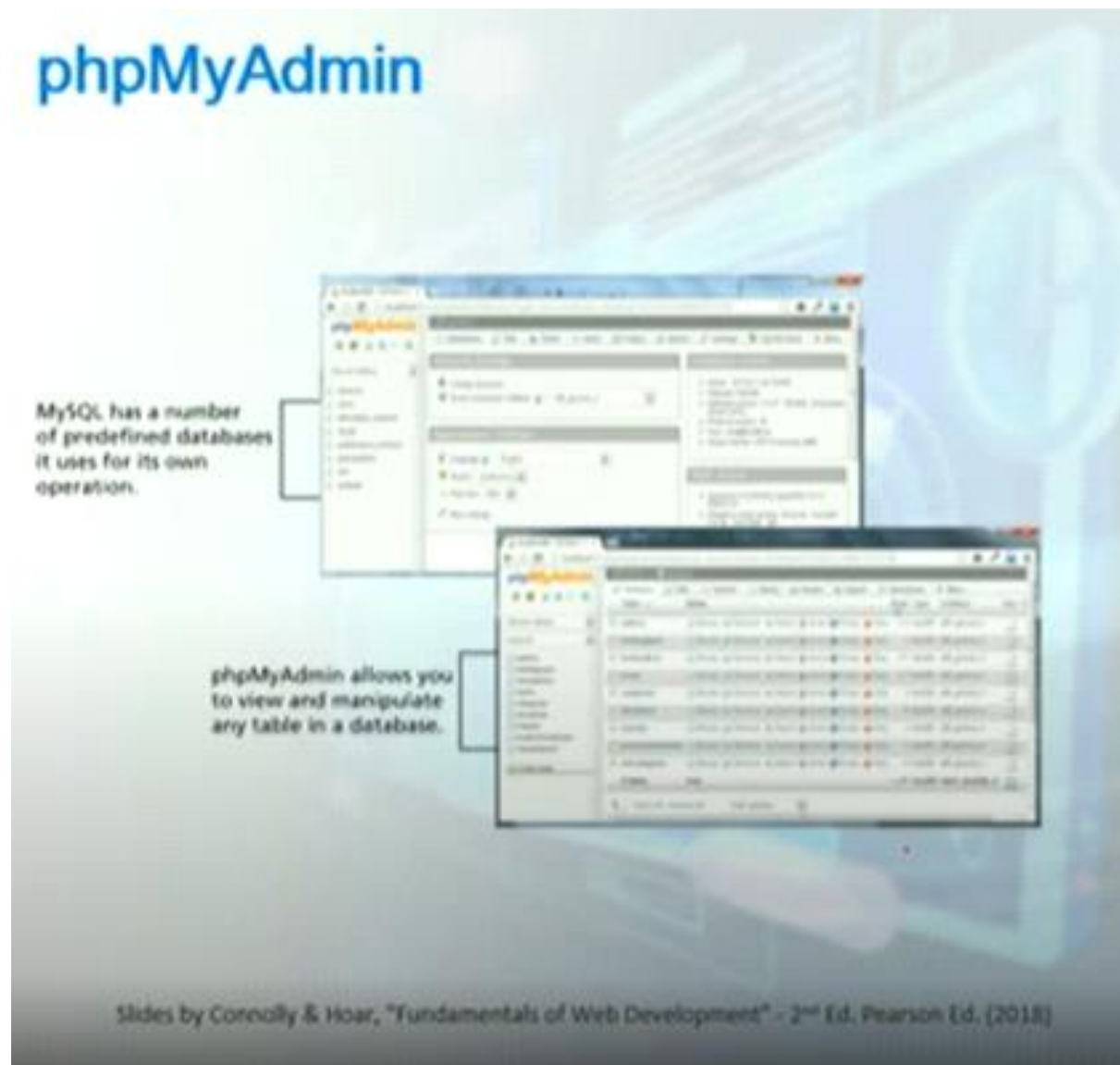
mysql -h 192.168.1.14 -u bookUser -p < commands.sql

After valid authentication of above command all statements written in the file will be executed one by one.

phpMyAdmin:

Sometimes, use of command line interface becomes difficult as you have to do all work by typing and memorize the commands as well. Alternative to command line interface is some graphical interface

like in MySQL, most popular utility is phpMyAdmin. It is web-based utility but used to manage the MySQL, it is installed with XAMPP.



Its interface is very interactive, you can see all databases, by selecting a database you will see its tables. You can also create new table, execute queries, can save the database and you can also import/export database.

MySQL Workbench:

It is desktop based graphical interface with MySQL. In this you can see the table connections if the primary key/foreign keys are set appropriately. You can also see database schema, create database, generate query, create tables and can perform all other activities related to database.

Module 117

On installing XAMPP, the utility to manage the database in PHP i.e., phpMyAdmin will install by default.

Command-Line Utility/Console based Utility:

To access the MySQL, command-line or console-based utility is used. If you want to access MySQL through the console, write 'cmd' in Windows, while in Linux terminal it is used. The steps are as follows:

1. Go to the folder in which you have installed MySQL via XAMPP.
2. Next there will be a folder of SQL and then a Bin folder.
3. In the bin folder, there is a file mysql.exe.
4. To access the mysql.exe file write this command "mysql -h localhost -u root"
5. After this, it will be connected to mariaDB.

Some other commands which are used are:

- By writing "help" you can view different commands that are available.
- To check the available database, write the command "show databases;"
- To connect with any database, write "use database name;"
- We can see tables and can make queries as well.

Graphical Interface for database:

Alternative to the command-line interface is phpMyAdmin. To open phpMyAdmin, open the browser, if XAMPP is installed then through the local host you can open it. There will be a link on the top right "phpMyAdmin ". Before clicking this link make sure that the database server is started. To check this, go to control panel of XAMPP, here MySQL must be in green color and status is "started", if it is started then you will see "start" which means database is running, otherwise it will show "stop" button.

If the button is green that means the database is running, so click on phpMyAdmin to access the database, here you can see all databases on the left side of the panel. To create a new database "New" option is also available. By selecting an existing database, you can create new tables using the create table interface.

For example, to view the "mysql" database, there are already many tables to manage the dbms. By clicking on "SQL" you can write any query and its result will be displayed. Moreover, you can export/import the tables, can set privileges for users and perform many other operations.

Module 118

In web development where we have to store the information in the database then, we have to connect the PHP script with the database. Let's see, how can we fetch the data from MySQL while using PHP?

PHP MySQL APIs:

Earlier, PHP had an extension named “MySQL extension” to connect with the database. This was the original extension to the PHP for working with MySQL and has been replaced with the newer “mysqli” extension.

mysqli Extension:

This extension provides both a procedural and an object-oriented approach. This extension also supports most of the latest features of MySQL. But the disadvantage is that if your website changes from MySQL to any other database, the code will not work for the new database as it was specifically developed for MySQL. As an alternative to this, another generic approach is used which is called “PHP data objects”.

PHP data objects (PDOs):

This provides an abstraction layer that with the appropriate drivers can be used with any database, and not just MySQL databases. However, it is not able to make use of all the latest features of MySQL. In PDPOs, the PHP code will not change with change of database, only database connection will change i.e., connection string and parameters will be modified.

Deciding on a Database API:

While PDO is unable to take advantage of some features of MySQL, there is a lot of merit to the fact that PDO can create database-independent PHP code. But there could be a scenario where such features are required of MySQL which are not available as PDOs, so you can use “mysqli”, otherwise use of PDOs are recommended. Like many things in the web world, there is not a single best choice. As the chapter (and book) proceed, we will standardize on the object-oriented, database-independent PDO approach.

Basic Connection Algorithm:

To connect with database there are some simple steps which are as follows:

1. Connect to the database
2. Handle connection errors
3. Execute the SQL query
4. Process the results
5. Free resources and close connection

Module 119

Accessing MySQL in PHP Fetching Data:

Let's see how we can fetch data from database systems (MySQL) into PHP. To fetch data in PHP two mechanisms are available i.e., procedural method through mysqli and other is object oriented.

Executing the Query:

For both of the above-mentioned methods we have to write query which we save in a variable like this:

```
$sql = "SELECT * FROM Categories ORDER BY CategoryName";
```

In case of using mysqli, query function will be used like this:

```
$result = mysqli_query (&connection, $sql)
```

In this function we will pass the connection object and sql query.

This query will execute on the server and return the result. In case of using PDO, no need to pass the connection object as it follows the object-oriented approach. Ther query will be like this:

```
$result = $pdo->query($sql);
```

Executing the Query

```
$sql = "SELECT * FROM Categories ORDER BY CategoryName";  
// returns a mysqli_result object  
$result = mysqli_query($connection, $sql);  
OR  
$result = $pdo->query($sql);
```



Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (20

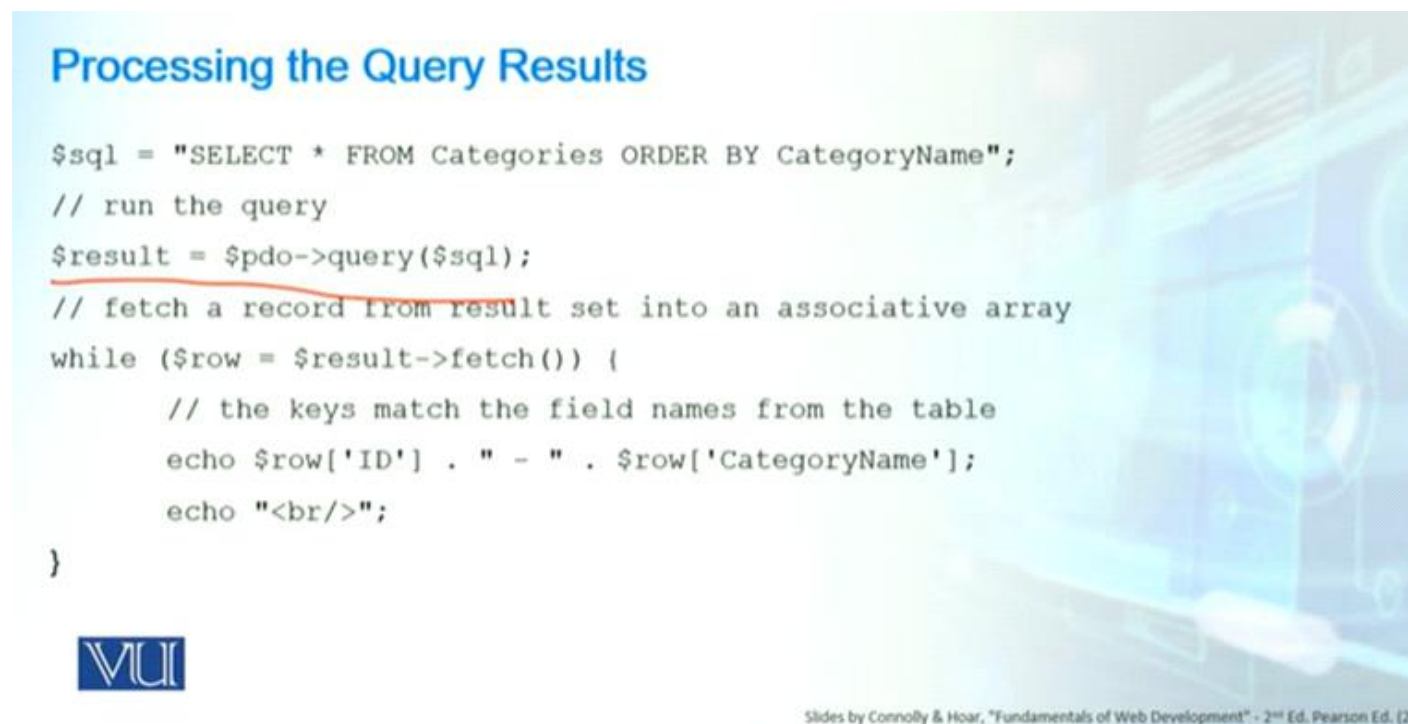
The PDO object will execute the query function and pass the SQL statement, this statement will execute on the server and return the results.

Processing the Query Results:

After retrieving the result, we have to process the results. In the result object there is a fetch () function which is basically a result set, on calling fetch () function it will return the first-row form retrieved results.

```
while ($row = $result->fetch ())
```

The retrieved result will save in \$row. Suppose, there is no result or we have processed all results then \$result->fetch () will return a null object and \$row=null will get us out of the loop. However, if there are results save in \$row, as it is an associative array so its index will be column name. Like in the database we have column name ID so we will get the value of the first row by writing \$row ['ID'], same is for category name. In the same fashion all rows will be processed and we will get out of loop.



Processing the Query Results

```
$sql = "SELECT * FROM Categories ORDER BY CategoryName";  
// run the query  
$result = $pdo->query($sql);  
// fetch a record from result set into an associative array  
while ($row = $result->fetch()) {  
    // the keys match the field names from the table  
    echo $row['ID'] . " - " . $row['CategoryName'];  
    echo "<br/>";  
}
```

VUI

Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2

Result is basically a result set, i.e., type of the cursor to retrieve the data. By this we access all rows one by one.

\$row = \$result->fetch ().

This is an associative array, for one record we will get one row and, in this record, the index will be column name.

Processing the Query Results

```
$sql = "select * from Paintings";  
$result = $pdo->query($sql);
```

\$result
Result set is a type
of cursor to the
retrieved data

ID	Title	Artist	Year
345	The Death of Marat	David	1793
400	The School of Athens	Raphael	1510
408	Bacchus and Ariadne	Titian	1520
425	Girl with a Pearl Earring	Vermeer	1665
438	Starry Night	Van Gogh	1889

```
$row = $result->fetch()
```

\$row
Associative
array

ID	Title	Artist	Year
345	Death of Marat	David	1793

keys

values



Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (20

Freeing Resources and Closing Connection:

It is important that when you are done with fetching data from the database, then close the connection of the database in the following way.

Freeing Resources and Closing Connection

```
//closes the connection  
mysqli_close($connection);  
  
// closes connection and frees the  
resources used by the PDO object  
$pdo = null;
```

Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

Working with Parameters:

When we execute the query on the server, we have to pass the parameters. There are many ways to pass parameters. For example, to change the name of the category name in the following query.

```
$sql = "UPDATE Categories SET CategoryName = 'Web' WHERE CategoryName = 'Business'";
```

Using PDO, call the `exec()` function and pass the query to it, it will return the count of rows which get affected as a result of this execution.

Working with Parameters

```
$sql = "UPDATE Categories SET  
CategoryName='Web' WHERE  
CategoryName='Business'";  
$count = $pdo->exec($sql);  
echo "<p>Updated " . $count . "  
rows</p>";
```

Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

Working with Parameters – Technique 1? Placeholders:

When you are storing any value in a database or taking value from a user to write SQL query, it is recommended to use placeholders. The placeholders are empty spaces used for passing values, if you don't use placeholders your website gets prone to hacking and specifically for attacks of SQL injections. Using placeholders, you can prevent them to some extent.

For example, we have a table named "books" and want to place values in some fields like:

```
$sql = INSERT INTO books (ISBN0, Title, CopyrightYear, ImprintId, ProductionStatusId, TrimSize,  
Description) VALUES (?, ?, ?, ?, ?, ?, ?);
```

The question marks here are placeholders that need to be filled later on. Before execution of the statement, call the prepare () function and pass the query to this.

```
$statement = $pdo->prepare ($sql);
```

by this prepared statement will get to know how many placeholders are using and how many values it has to accept. After this, for each value we use the bindValue() function like, \$statement ->bindValue (1, \$_POST ['isbn']);

Do the same for remaining values. The values will be replaced by question marks (placeholder) and upon execution will save in the database.

Working with Parameters – Technique 1 ? Placeholders

```
$sql = "INSERT INTO books (ISBN10, Title,
CopyrightYear, ImprintId, ProductionStatusId,
TrimSize, Description) VALUES (?, ?, ?, ?, ?, ?, ?)";

$stmt = $pdo->prepare($sql);

$stmt->bindValue(1, $_POST['isbn']);
$stmt->bindValue(2, $_POST['title']);
$stmt->bindValue(3, $_POST['year']);
$stmt->bindValue(4, $_POST['imprint']);
$stmt->bindValue(5, $_POST['status']);
$stmt->bindValue(6, $_POST['size']);
$stmt->bindValue(7, $_POST['desc']);
$stmt->execute();
```

Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

Calling function for each place holder severalty will be time consuming, to solve this issue instead of binding individual values make an array for all values. For this, the main mechanism will remain the same, call the execute function after prepare statement and pass the array to the execute () function.

Working with Parameters – Technique 1 ? Placeholders with Array

```
/* can pass an array, to be used in order */
```

```
$sql = "INSERT INTO books (ISBN10, Title, CopyrightYear, ImprintId, ProductionStatusId, TrimSize, Description) VALUES (?, ?, ?, ?, ?, ?, ?)";  
$statement = $pdo->prepare($sql);  
$statement->execute  
(array($_POST['isbn'],  
$_POST['title'], $_POST['year'],  
$_POST['imprint'], $_POST['status'],  
$_POST['size'], $_POST['desc']));
```

Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

Working with Parameters- Technique 2 – Named Parameters:

Named parameters can also be used with placeholders. For this, in query write: with the name of value, you want to store like this:

```
$sql = "INSERT INTO books (ISBN10, Title, CopyrightYear, ImprintId, ProductionStatusId, TrimSize, Description) VALUES (: isbn, :title, :year, :imprint, :status, :size, :desc)";
```

Working with Parameters – Technique 2 - Named Parameters

```
$sql = "INSERT INTO books (ISBN10, Title, CopyrightYear, ImprintId,
ProductionStatusId, TrimSize, Description) VALUES (:isbn, :title, :year,
:imprint, :status, :size, :desc) ";
$stmt = $pdo->prepare($sql);
$stmt->bindValue(':isbn', $_POST['isbn']);
$stmt->bindValue(':title', $_POST['title']);
$stmt->bindValue(':year', $_POST['year']);
$stmt->bindValue(':imprint', $_POST['imprint']);
$stmt->bindValue(':status', $_POST['status']);
$stmt->bindValue(':size', $_POST['size']);
$stmt->bindValue(':desc', $_POST['desc']);
$stmt->execute();
```



Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

After this pass the name of the placeholder in bindValue() function, instead of question marks. In case of using "?" There is a chance of producing a logical bug if the order of values changes, while using the named parameter as placeholder only the appropriate value will go to its place. You can use an array with the named parameters, the main mechanism will remain the same, while in the execute function, mention the value of the named parameters.

Getting User Input into a Query:

HTML forms are used to get the input from the user, this example is just for learning. In real production you have to take care of the values that are valid or not.

Getting User Input into a Query

Browser - Rename Category Form

Category to change:

New category name:

```
<form method="post" action="rename.php">
  <input type="text" name="old" /><br/>
  <input type="text" name="new" /><br/>
  <input type="submit" />
</form>
```

\$_POST['new']
Communications

\$_POST['old']
English

```
UPDATE Categories SET CategoryName='Communications' WHERE CategoryName='English'
```

Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

Module 120

Accessing MySQL in PHP Performing Transactions:

Transaction plays an important role in database management systems, particularly when we are dealing with a group of statements. If you want that group of statements to remain related to each other while executing, and if one statement fails to execute, the remaining statements will stop. For this every database system provides the facility of transactions.

Using Transactions in MySQL:

To use a transaction PDO object is used, which establishes the connection with the database. Once a connection is established, call the function “`$pdo->beginTransaction();`”, it will initiate the transactions.

After initiating transactions, we can execute multiple queries, let's say we have executed some queries and called the commit function as “`$pdo->commit ();`”.

Commit function call will execute all the queries and upon successful execution the program logic will carry on. But if any error occurs in execution of even a single statement/query then an exception will be thrown. Exceptions will be caught by the catch block and will call the roll back function like “`$pdo->rollback();`”. Roll back function will reverse all statements and the database will have no impact.

Suppose, we have two statements, one is used to insert the category of “Philosophy” and the other is “Art”. Let's say the first statement executes successfully, while the second statement will give an error so an exception will be thrown, control will shift to catch block and roll back function will be called. Now, the roll back function stops execution of the first statement as well. The value of category “Philosophy” is stored in the database only in a case when there is no exception at the time of calling the commit function.

Using Transactions

```
$pdo = new PDO($connString,$user,$pass);  
  
try (  
    // begin a transaction  
    $pdo->beginTransaction();  
    // a set of queries: if one fails, an exception will be thrown  
    $pdo->query("INSERT INTO Categories (CategoryName) VALUES ('Philosophy')");  
    $pdo->query("INSERT INTO Categories (CategoryName) VALUES ('Art')");  
    // if we arrive here, it means that no exception was thrown  
    $pdo->commit();  
} catch (Exception $e) {  
    // we must rollback the transaction since an error occurred with insert  
    $pdo->rollback();  
}
```

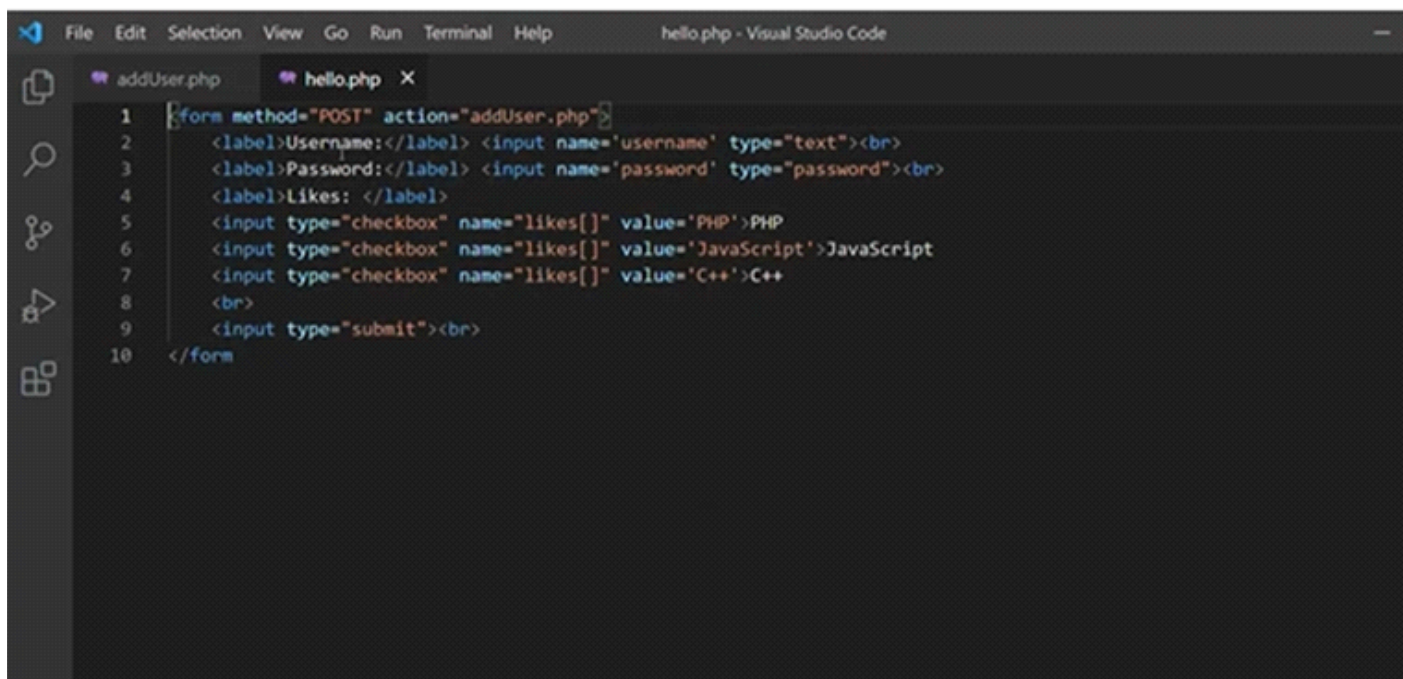


This process is very useful particularly when you are working with a bank database. For example, you have drawn some amount from an account and transferred it into another account. In this case execution of both statements is necessary, in case of error, amount will be deducted from an account and will not be transferred to another account. If you are using a transaction then if an error occurs in the second statement then the first statement will also be rolled back and that deducted amount will be undo.

Module 121

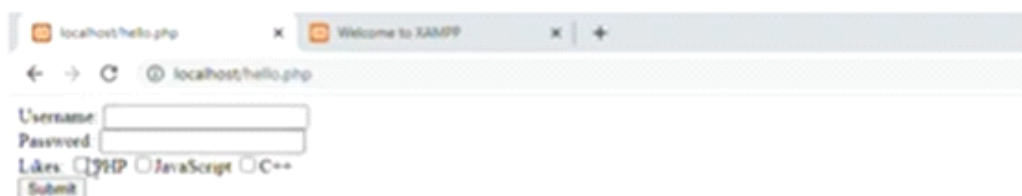
Example, Database Creation:

Let's work on an example, in which we will create a database and insert values in it. In the previous lecture, we created a form and took input for username and password. After this, in action, user information in PHP was sent to the server, there was a script "addUser.php" that took that information or form data and then displaying on browser.

A screenshot of the Visual Studio Code editor interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar shows 'hello.php - Visual Studio Code'. The editor has two tabs: 'addUser.php' and 'hello.php'. The 'hello.php' tab is active, displaying the following HTML code:

```
1 <form method="POST" action="addUser.php">
2   <label>Username:</label> <input name='username' type="text"><br>
3   <label>Password:</label> <input name='password' type="password"><br>
4   <label>Likes: </label>
5   <input type="checkbox" name="likes[]" value='PHP'>PHP
6   <input type="checkbox" name="likes[]" value='JavaScript'>JavaScript
7   <input type="checkbox" name="likes[]" value='C++'>C++
8   <br>
9   <input type="submit"><br>
10 </form>
```

This page in browser will look like this:



Let's say if we want to save this information in a database. For this, we have to access MySQL using phpMyAdmin. Open the local host to access phpMyAdmin, you can open this directly by typing "localhost/phpmyadmin/". After opening phpMyAdmin interface follows these steps:

- To create a database, go to Databases.
- After creating the database, by default it will ask to add a table.
- To add go to the “Crate table” tab just write the name of the table.
- Whenever you save the record in the database, you add one column for id which is called primary key. You can not take this id form user; it will be created automatically and the database will set it as auto increment.
- So here we need three columns, id, user name, and password so we set Number of columns as 3.
- After creation of table, set the data type for each column such as for id, set the type as INT (integer), Default value as None, and check the “auto increment” box in Index and set PRIMARY, it will become unique.
- For column “username” set the length 50, data type as VARCHAR, and index as UNIQUE, so that every user name should be unique to avoid multiple users with the same name. After that, set the index name as “username”.
- For column “password”, type is VARCHAR, length as 50. This type is just for practice purposes.
- Create another table for the “likes” of the user programming language by following the same pattern. Remember here to avoid using the word “like” as it is a keyword in SQL.

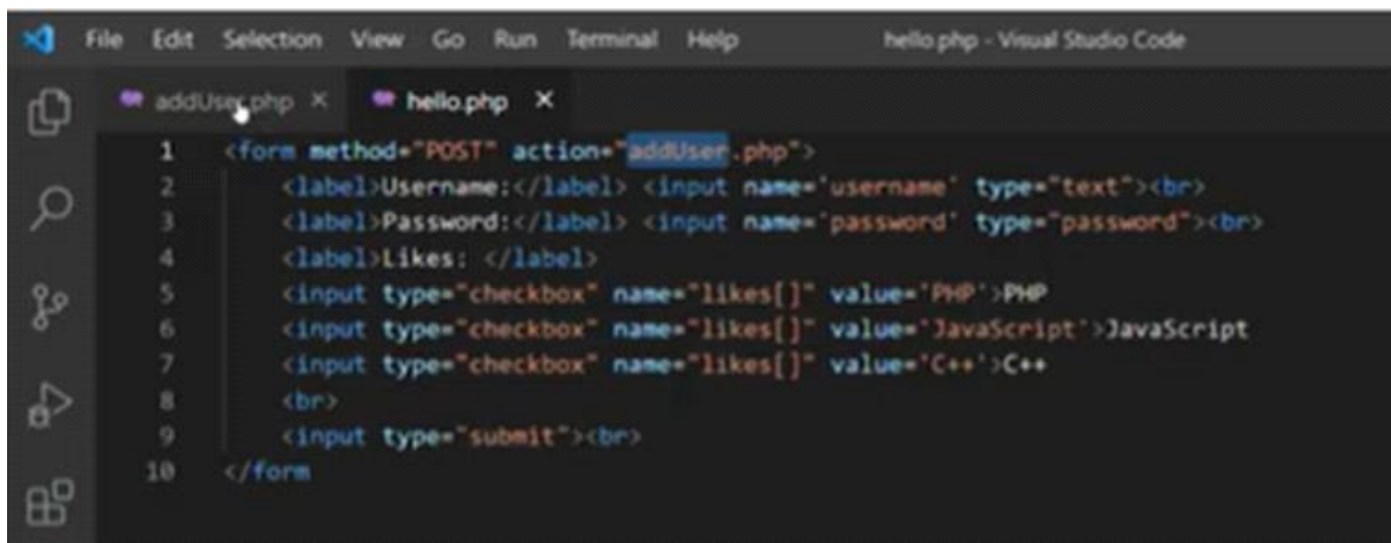
In development, avoid setting the password as text in the database. Before saving the password, hash it using MD5 and store that password hash in the database. The benefit of this is that if you want to recover a password or user forgot password you cannot access the existing password. So, you have to reset the password always.

Module 122

Example – Database Insertion:

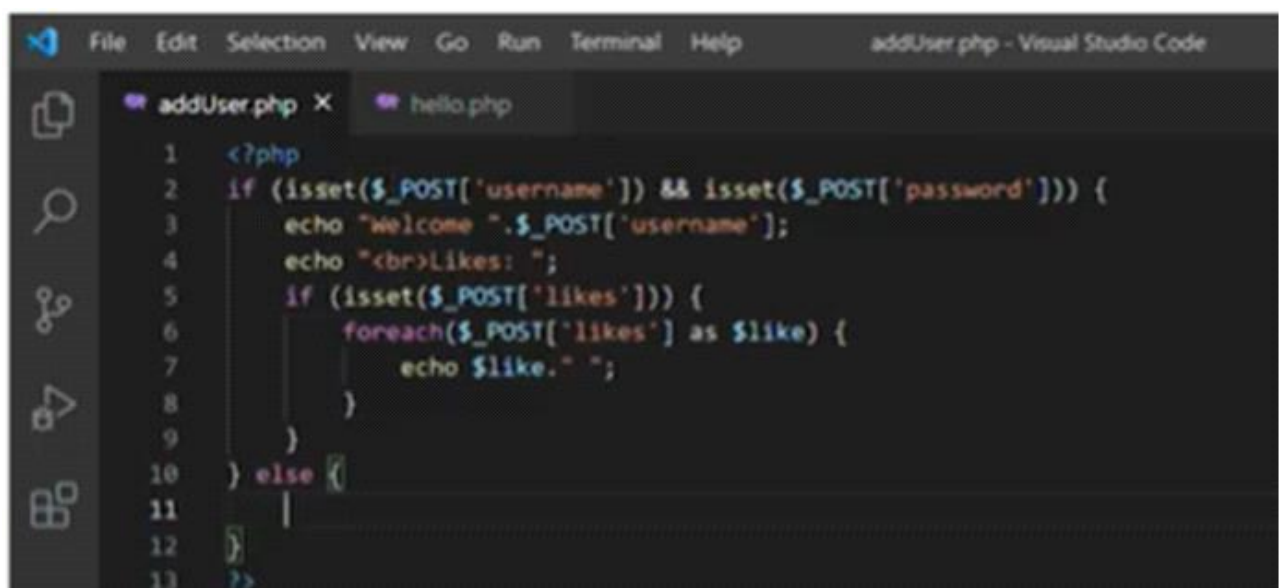
After creating the database, read the data from the HTML forms, and send that data to a PHP script to access the data, then insert those values in the database.

Here we have form containing username and password and “addUser.php” is mentioned in its action, when we click on the submit button, the information provided by the user will be submitted to the script.

A screenshot of the Visual Studio Code editor with the file 'addUser.php' open. The code is an HTML form with a POST method and an action of 'addUser.php'. It contains three labels: 'Username:', 'Password:', and 'Likes:'. The 'Username' and 'Password' fields are text inputs. The 'Likes' field has three checkboxes with values 'PHP', 'JavaScript', and 'C++'. A 'submit' button is at the bottom.

```
1 <form method="POST" action="addUser.php">
2   <label>Username:</label> <input name="username" type="text"><br>
3   <label>Password:</label> <input name="password" type="password"><br>
4   <label>Likes: </label>
5   <input type="checkbox" name="likes[]" value='PHP'>PHP
6   <input type="checkbox" name="likes[]" value='JavaScript'>JavaScript
7   <input type="checkbox" name="likes[]" value='C++'>C++
8   <br>
9   <input type="submit"><br>
10 </form>
```

The following is the script of the “addUser.php”.

A screenshot of the Visual Studio Code editor with the file 'addUser.php' open. The code is a PHP script that checks if 'username' and 'password' are set in the \$_POST array. If they are, it echoes a welcome message and the likes. If not, it echoes an error message.

```
1 <?php
2 if (isset($_POST['username']) && isset($_POST['password'])) {
3     echo "Welcome " . $_POST['username'];
4     echo "<br>Likes: ";
5     if (isset($_POST['likes'])) {
6         foreach($_POST['likes'] as $like) {
7             echo $like . " ";
8         }
9     }
10 } else {
11     |
12 }
13 ?>
```

In this script the very first thing we have to check is whether the user's name and password exists or not? So, make sure whenever you take data from the form that data must exist at the php end before putting it into the database. If the data exists then save the values in the variable and save them in the database.

Here, we will create the variable for username and pass and then establish the database connection to insert the values in the database. While creating the database connection, in connection string first mention the type of database which you are accessing like here we are using "pdo", if you are using any other DBMD then mention that name in place of "sql". After that mention about the server hosting database, if your database and webserver is on the same machine then write "local host". After that, write the name of the database. In addition, you have to provide the database user and password which is root for username and password is an empty string at the local host, but in a real development environment you have to create a proper database user with appropriate privileges with a strong password. After mentioning all requirements, use \$pdo for establishing connection.

```
<?php
if (isset($_POST['username']) && isset($_POST['password'])) {
    $conn = 'mysql:host=localhost;dbname=users';
    $dbuser = 'root';
    $dbpass = '';
    $pdo = new PDO($conn,$dbuser,$dbpass);
    $username = $_POST['username'];
    $password = $_POST['password'];
```

There is a chance that connection doesn't establish, to check this we will write all code in a try catch block like this:

```

<?php
if (isset($_POST['username']) && isset($_POST['password'])) {
    try {
        $conn = 'mysql:host=localhost;dbname=users';
        $dbuser = 'root';
        $dbpass = '';
        $pdo = new PDO($conn,$dbuser,$dbpass);

        $username = $_POST['username'];
        $password = $_POST['password'];
        echo "<br>Likes: ";
        if (isset($_POST['likes'])) {
            foreach($_POST['likes'] as $like) {
                echo $like." ";
            }
        }
    } catch (PDOException $e) {
        die($e->getMessage());
    }
} else {
}
?>

```

Using try catch we will be able to identify issues causing error, as in case of error it will throw an exception. In the catch block we will catch the PDO exception and use a die or exit method like “die(\$e->getMessage())”. This getMessage() method will read the exception if thrown, display it and exit from the PHP script.

After establishing connection, use the insert statement to insert values in the database. Just make sure to use placeholders in insert statements if you are getting data from the form, it will help you to prevent certain attacks on your website. You can insert direct values but it is never recommended. After writing a sql statement, prepare it through the pdo object. This will give us a statement type object which is used to execute the query by giving values of named placeholders. In case of more than one value in the form of an array.

```

$sql = 'INSERT INTO user(username,password) VALUES(:user,:pass)';
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user' => $username, ':pass' => $password));

```

This insertion step is performed in three steps, it can be done in a single step but not recommended.

If you are not using named placeholder, using question marks (?) only then the values will be given like this:

```

$sql = 'INSERT INTO user(username,password) VALUES(?,?)';
$stmt = $pdo->prepare($sql);
$stmt->execute(array($username,$password));

```

In positional placeholder if the order of columns or values get changed it will create a problem, while in using named placeholder it will not create any issue if the order of values/column changed.

Execution of above code will display this page where we input the values which need to be inserted in database.



A screenshot of a web browser showing a form titled 'localhost/addUser.php'. The form has three input fields: 'Username:' with the value 'bilal', 'Password:' with masked characters '*****', and 'Likes:' with three checkboxes: 'PHP' (checked), 'JavaScript' (unchecked), and 'C++' (checked). Below these fields is a 'Submit' button.

On submitting the form, we got this error.



This error occurs because we have updated our code with positional placeholder while we are inserting through names placeholders, so after correction code execute it again which will give correct out like this:



To check the values insertion in the database, check the database using phpMyAdmin. Here user id is generated automatically, remember to save the password in hash format.



In the next step we want to insert the values in the “likes” table about the user’s choice in programming languages. But the issue here is that we don’t know the user id while inserting values in php. To resolve this issue, the sql provides us the last insert id so on successful execution of insert statement we can get the id of the user which is inserted last.

If we have already set the likes variable then we can insert each like in the database. Before this user id is required, to get it we will use a pdo object like “\$usrId = \$pdo->lastInsertId();” . The insert statement for like table:

```
$usrId = $pdo->lastInsertId();
echo "<br>Likes: ";
if (isset($_POST['likes'])) {
    foreach($_POST['likes'] as $like) {
        $sql = 'INSERT INTO likes(userid,likes) VALUES(:userid,:likes)';
        $stmt = $pdo->prepare($sql);
        $stmt->execute(array(':userid' => $usrId, ':likes' => $like));
    }
}
```

On executing the above code, it will display the following page where we insert data for user “ahmad”.



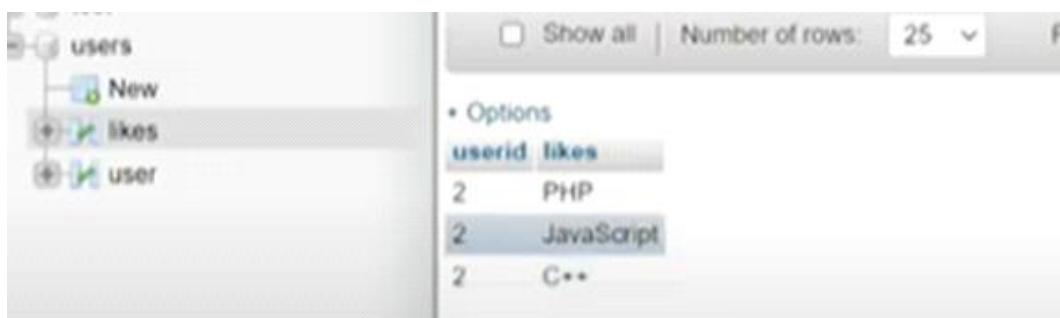
Username: ahmad
Password: *****
Likes: ☒ PHP ☒ JavaScript ☒ C++
Submit

On submitting it displays this, which no means no exception is thrown and values are inserted in the database.



Likes:

Open the database and check the user table first to check the user id of “ahmad” then browse the table “likes” where values are inserted.

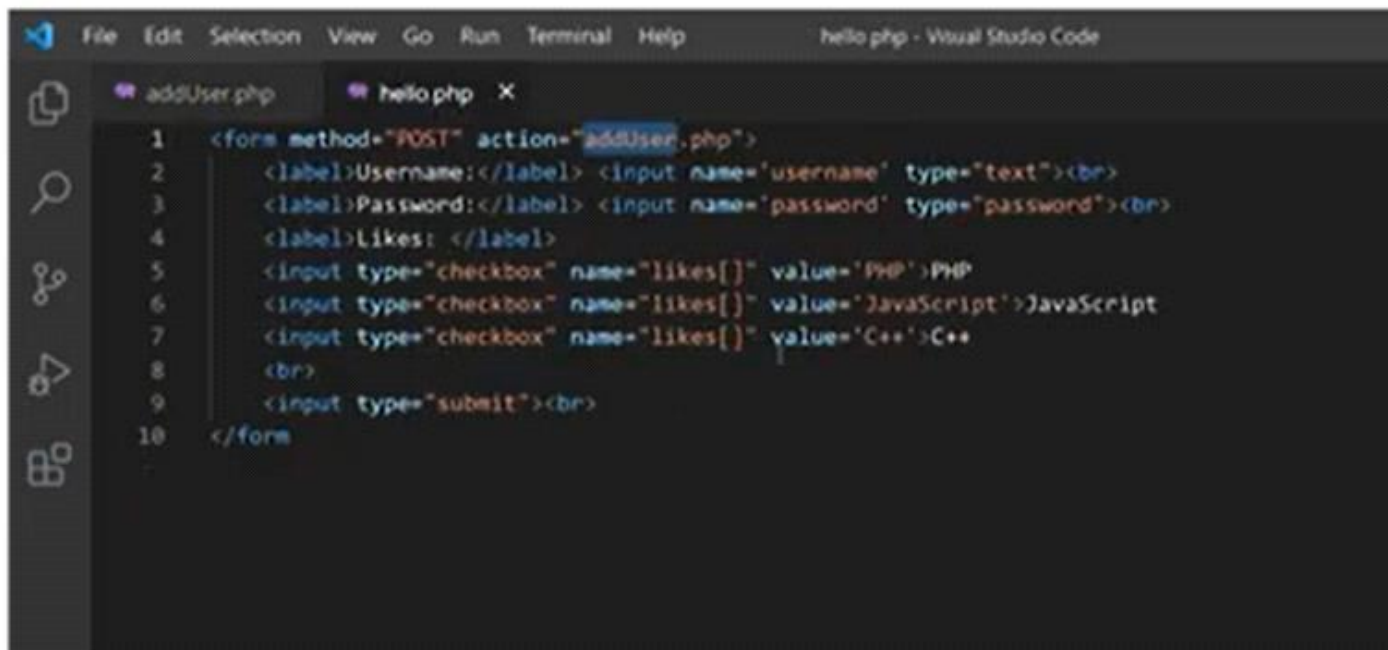


userid	likes
2	PHP
2	JavaScript
2	C++

In a normalized database, there will be a separate table for “likes” and have many to many relationships with the “user” table.

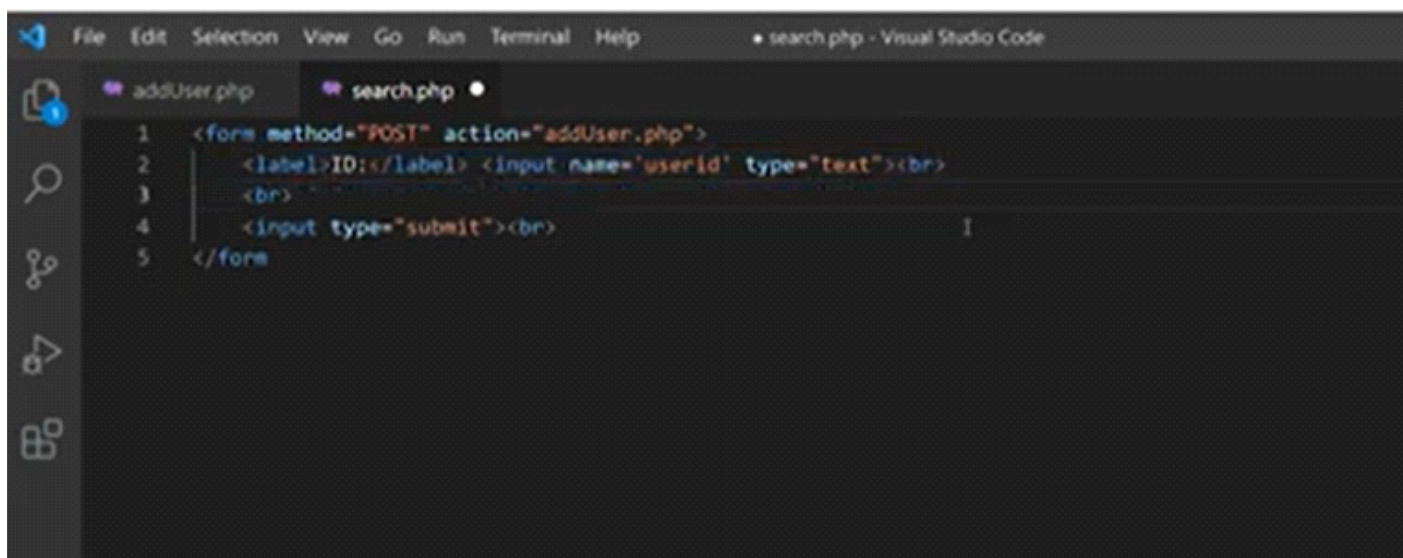
Module 123

After inserting values into the database using PHP, let's see how we can fetch the values from the database and display them in PHP. For this we will make a form for “search”, save it as search.php.



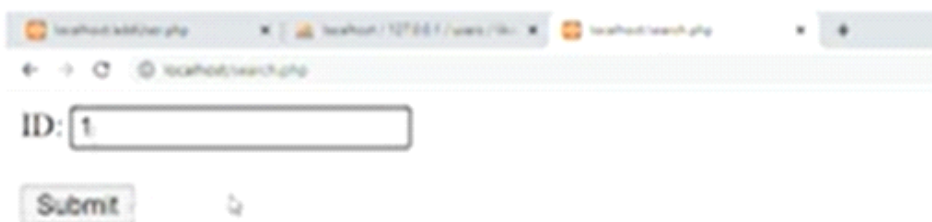
```
1 <form method="POST" action="addUser.php">
2   <label>Username:</label> <input name='username' type="text"><br>
3   <label>Password:</label> <input name='password' type="password"><br>
4   <label>Likes: </label>
5   <input type="checkbox" name="likes[]" value='PHP'>PHP
6   <input type="checkbox" name="likes[]" value='JavaScript'>JavaScript
7   <input type="checkbox" name="likes[]" value='C++'>C++
8   <br>
9   <input type="submit"><br>
10 </form>
```

Let's say pass the user's name, if it exists then display the user's name and id. For this create the form and ask for user id as follows:



```
1 <form method="POST" action="addUser.php">
2   <label>ID:</label> <input name='userid' type="text"><br>
3   <br>
4   <input type="submit"><br>
5 </form>
```

For example, we will pass id "1" here, which will display the name of the user having id "1".

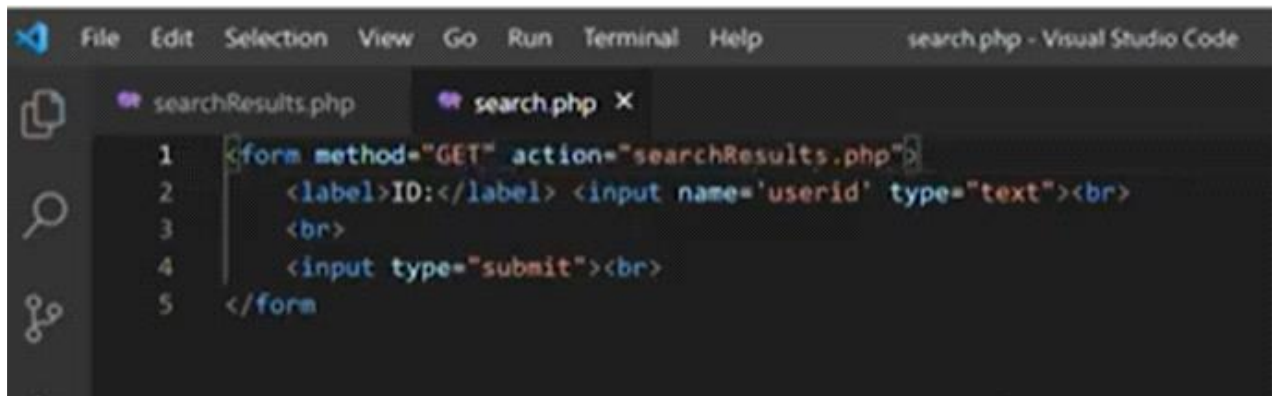


localhost/addUser.php | localhost:127.0.0.1/users/100 | localhost/search.php

localhost/search.php

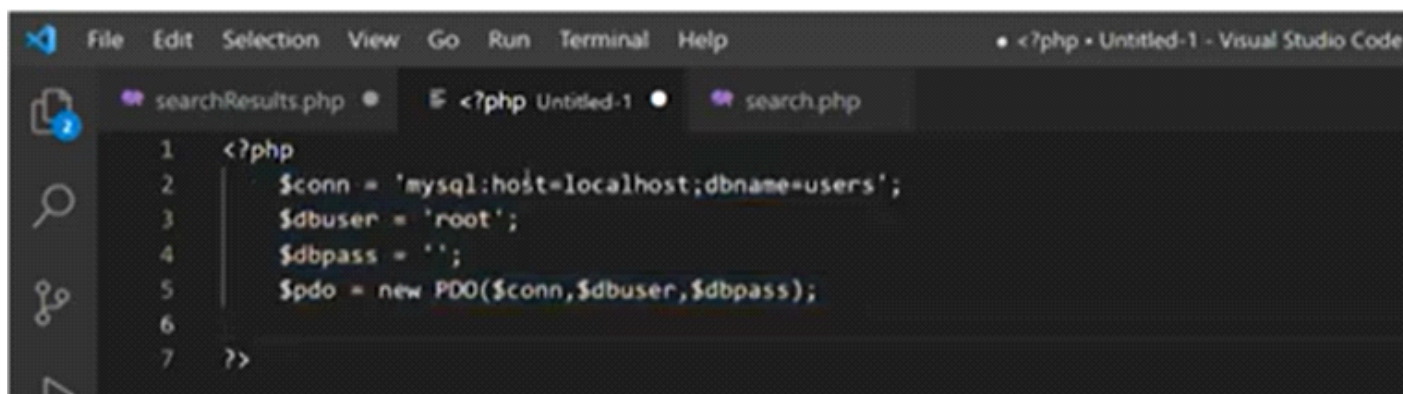
ID:

Set the action page as “search_results.php”, also set the method as “GET” and save it. The form we have is “search.php” so when the form is submitted its data goes to action page.



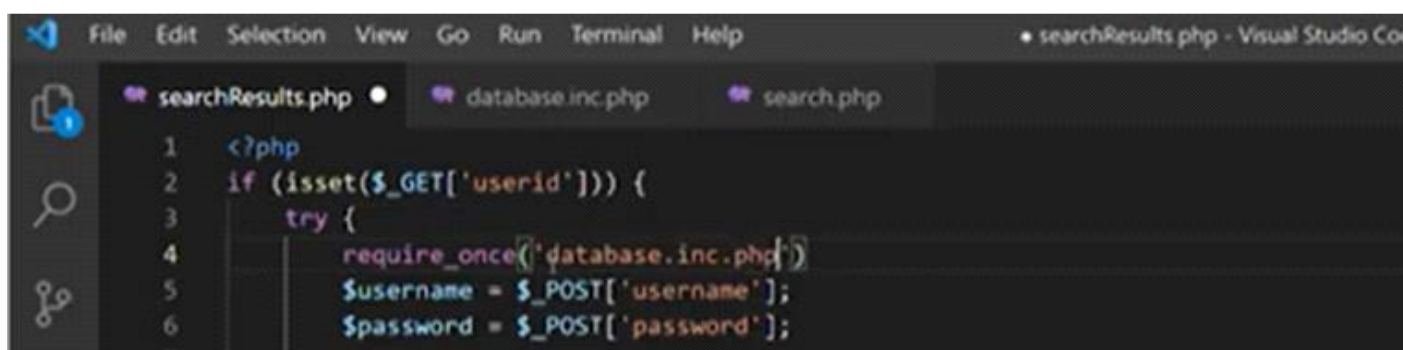
```
1 <form method="GET" action="searchResults.php">
2   <label>ID:</label> <input name='userid' type="text"><br>
3   <br>
4   <input type="submit"><br>
5 </form>
```

If user id exists and set then create the database connection and specify the user id in this to get the results. The issue of creation of database connection arises if we are dealing with multiple scripts at a time to interact with the database. For this we have to reproduce database connection code for each script, let's say we have to make some changes in db connection then we have to change the complete script. To overcome this issue, create a new file and save the database connectivity code and save it with extension “inc” like “database.inc.php”



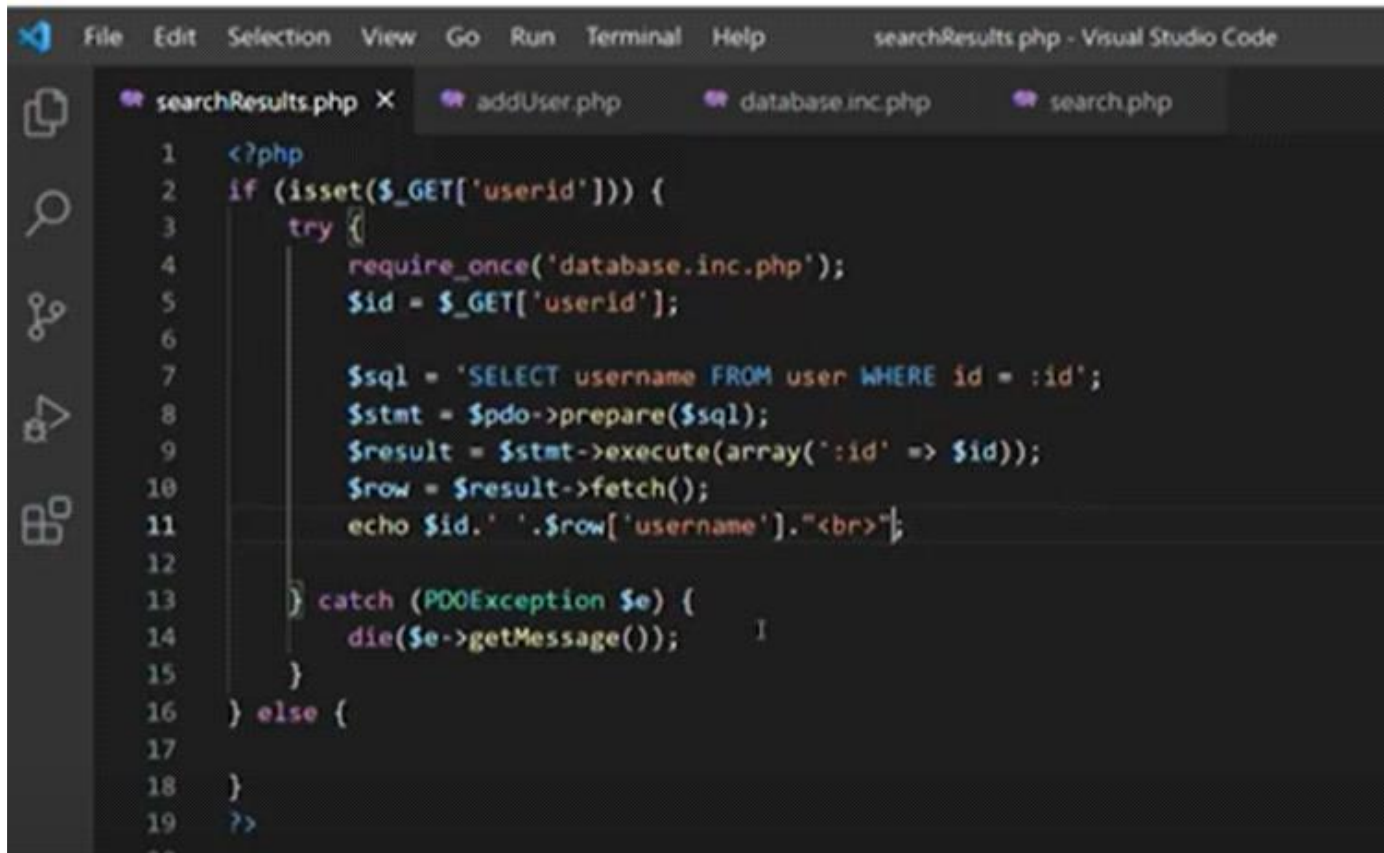
```
1 <?php
2   $conn = 'mysql:host=localhost;dbname=users';
3   $dbuser = 'root';
4   $dbpass = '';
5   $pdo = new PDO($conn,$dbuser,$dbpass);
6
7 >>
```

After this, include this file where you have to establish the database connection using **require_once()**, so that file is included only once. Do the same with other scripts as well. Remember, if you have a code which is repeating in multiple files, keep that in a single file and include that file where required.



```
1 <?php
2 if (isset($_GET['userid'])) {
3   try {
4     require_once('database.inc.php');
5     $username = $_POST['username'];
6     $password = $_POST['password'];
7   }
```

After establishing database connection and getting user id, write the query to fetch the data where id is equal to user id.

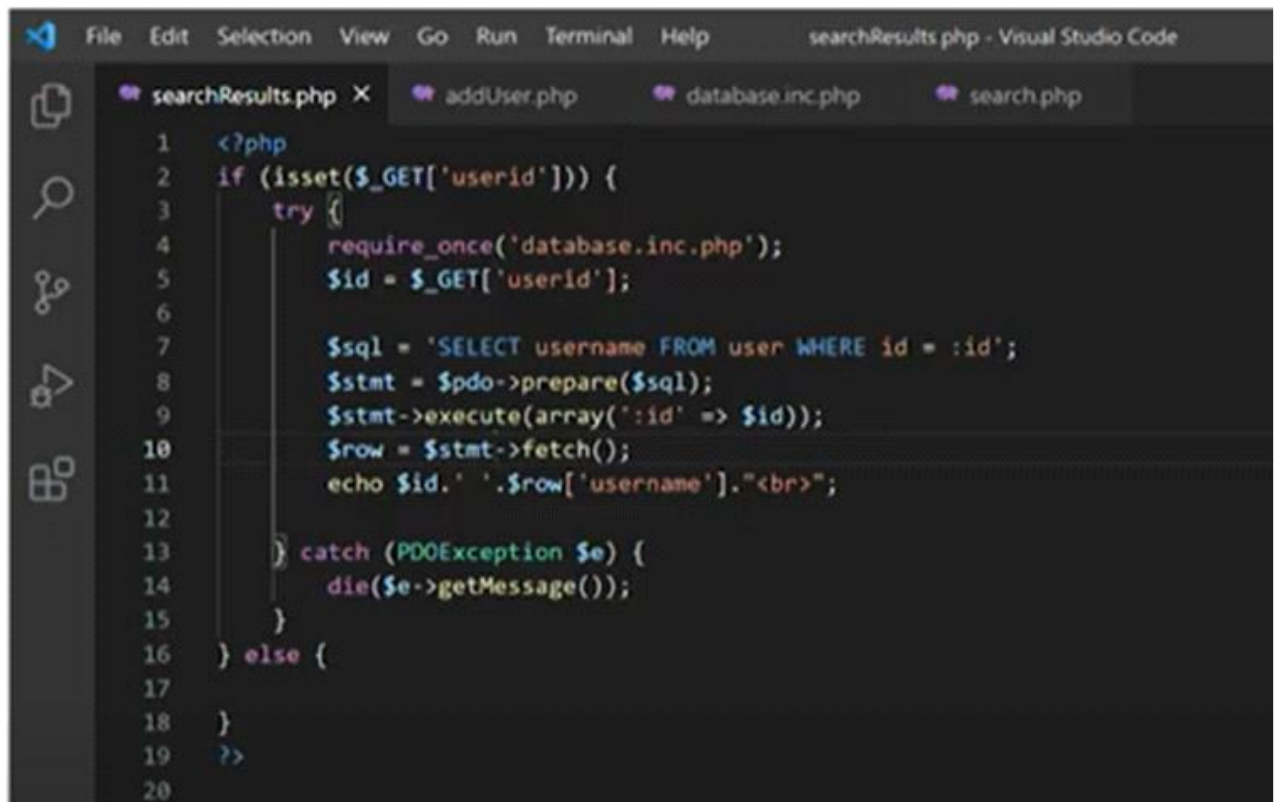


```
1 <?php
2 if (isset($_GET['userid'])) {
3     try {
4         require_once('database.inc.php');
5         $id = $_GET['userid'];
6
7         $sql = 'SELECT username FROM user WHERE id = :id';
8         $stmt = $pdo->prepare($sql);
9         $result = $stmt->execute(array(':id' => $id));
10        $row = $result->fetch();
11        echo $id.' '.$row['username'].'<br>';
12    } catch (PDOException $e) {
13        die($e->getMessage());
14    }
15 } else {
16 }
17 }
18 }
19 ?>
```

On execution of this code, enter the user id as “1”, but it gives a fatal error as result, which is most likely due to the error in the fetch statement. Let’s figure it out.



So, when we are using a fetch statement, we have two options, `pdo.query` directly or `pdo.prepreate` or `stmt.execute`. Here we have used `execute`, so it will return only true or false value. So, here we will explicitly fetch the result by directly using a statement.



```
1 <?php
2 if (isset($_GET['userid'])) {
3     try {
4         require_once('database.inc.php');
5         $id = $_GET['userid'];
6
7         $sql = 'SELECT username FROM user WHERE id = :id';
8         $stmt = $pdo->prepare($sql);
9         $stmt->execute(array(':id' => $id));
10        $row = $stmt->fetch();
11        echo $id.' '.$row['username'].'<br>';
12    } catch (PDOException $e) {
13        die($e->getMessage());
14    }
15 } else {
16 }
17 }
18 }
19 }
20 }
```

Displaying the code again by giving user id as 1, it will display the following result.



In the same fashion, we can get the information about all users. We can also get information about which user likes which programming language. Let's say if we are fetching records of all users then make the following changes in the code.



```
$sql = 'SELECT username FROM user WHERE id = :id';
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':id' => $id));
while ($row = $stmt->fetch()) {
    echo $id.' '.$row['username'].'<br>';
}

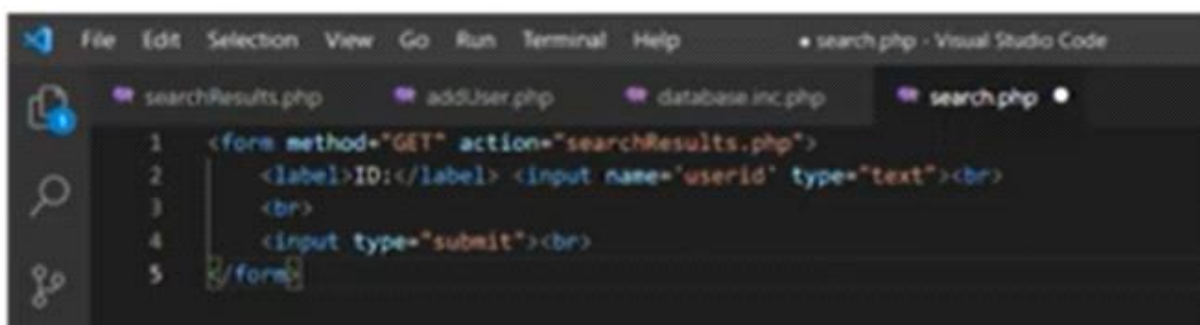
} catch (PDOException $e) {
    die($e->getMessage());
}
```

Remember, if you have to take input from using through form then always use placeholders, although you can access it directly but in that case security issues will arise.

Module 124

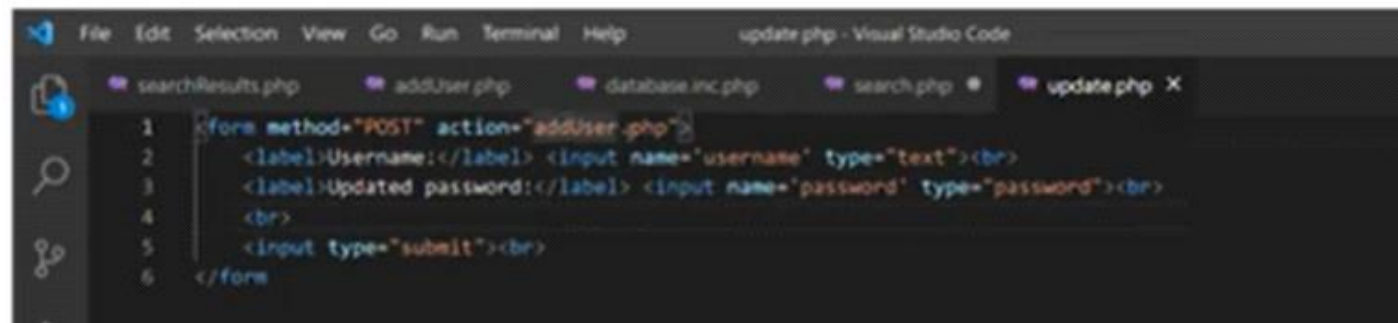
Example – Database:

After inserting values in the database lets learn how to update the stored values in the database. Main idea is that the user gives the values in the form which needs to be updated and then these values will be saved in the database. Let's work on the example to understand this. Suppose we have a user and we want to update its password.



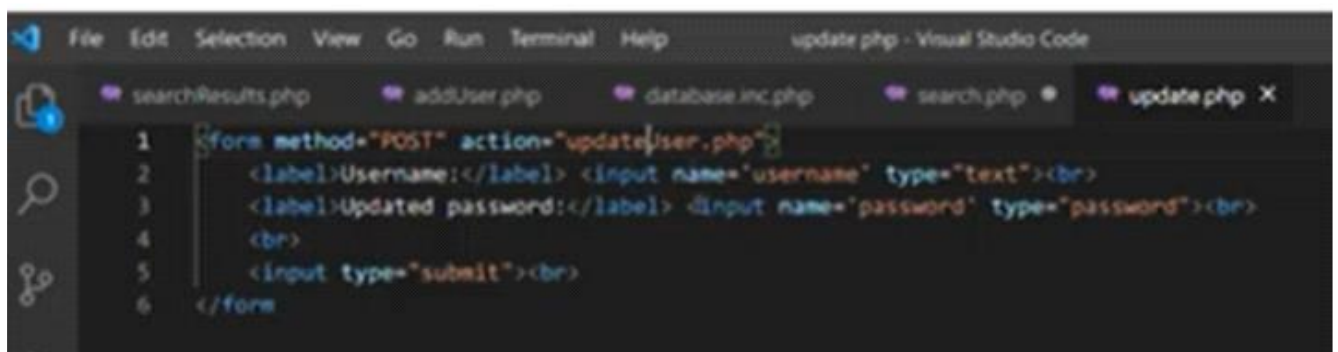
```
1 <form method="GET" action="searchResults.php">
2   <label>ID:</label> <input name='userid' type="text"><br>
3   <br>
4   <input type="submit"><br>
5 </form>
```

Set the field updated password as follows.



```
1 <form method="POST" action="addUser.php">
2   <label>Username:</label> <input name='username' type="text"><br>
3   <label>Updated password:</label> <input name='password' type="password"><br>
4   <br>
5   <input type="submit"><br>
6 </form>
```

After this user will submit this form and call the update user script.



```
1 <form method="POST" action="updateUser.php">
2   <label>Username:</label> <input name='username' type="text"><br>
3   <label>Updated password:</label> <input name='password' type="password"><br>
4   <br>
5   <input type="submit"><br>
6 </form>
```

It will show the following result on screen.

Username:

Updated password:

On giving data in the form and submitting, it will go to updateuser.php and it will update the password of the user given in the input. After this, save the addUser.php with the name of updateUser.php, include the database connectivity in the form of pdo. Now write the update query as follows:

```
searchResults.php  updateUser.php  database.inc.php  search.php  update.php
1  <?php
2  if (isset($_POST['username']) && isset($_POST['password'])) {
3      try {
4          require_once('database.inc.php');
5
6          $username = $_POST['username'];
7          $password = $_POST['password'];
8
9          $sql = 'UPDATE user SET password = :pass WHERE username = :user';
10         $stmt = $pdo->prepare($sql);
11         $stmt->execute(array(':user' => $username, ':pass' => $password));
12         $userid = $pdo->lastInsertId();
13         echo "<br>Likes: ";
```

Execute this code, the return type will show how many records are updated, for this set the count which should be one.

```
$sql = 'UPDATE user SET password = :pass WHERE username = :user';
$stmt = $pdo->prepare($sql);
$count = $stmt->execute(array(':user' => $username, ':pass' => $password));
```

If no record is updated in the result of this execution that means that no record exists against this user. If one record is updated that means the user exists and its password is updated, to check this write the following line of code.

```

$count = $stmt->execute(array(':user' => $username, ':pass' => $password));
if ($count == 0) {
    echo 'User not found <br>';
} else {
    echo "Record for user $username updated successfully <br>";
}

```

The working of the above code will give following result.



Now check this in the database to verify.



Let say we give the name of the user who doesn't exists, such "khurram" doesn't exist in record then it will display message like this:



We should not have this, so let's figure out why we have got this message? To check this, we will display the \$count to check its value, it will be the following output.



In this we are using a prepare and execute statement, if we remove this i.e., directly run the query the count will return us the number of records updated. So, we will use another function like:

```

$sql = 'UPDATE user SET password = :pass WHERE username = :user';
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user' => $username, ':pass' => $password));
$count = $stmt->rowCount();
if ($count == 0) {
    echo 'User not found <br>';
} else {
    echo "Record for user $username updated successfully <br>";
}

```

This function will tell us how many rows are updated. So, after this function when we try to update the record of user “khurram” which does exist, it will return us output as “User not found” and if the user exists it will show output as “Record for user ahmad updated successfully”.

Remember that the execute statement will return true or false, and to check the number of updated records the rowCount() function will be used.

Module 125

Type of Errors:

There are different types of errors we faced during web development.

Expected errors:

The most common errors expected errors, these errors occur mostly on the base of user input. Like users have not provided the required value or values are not in the required format. Such errors can be checked at runtime and handled logically.

Warnings:

The issues that arise during programming are called warnings and we can deal with them during programming development.

Fatal errors:

The errors which occur after deployment and stop the further execution of program/script are called fatal errors. For example, your web server and database server are running on two different machines, let's say database server went down so when you try to connect with a database it will throw an exception and the program will crash. If the program is handled appropriately, you will give appropriate measures like you can send a message to the user that “Please try again, we are facing some issues”. Meanwhile, an email alert can be generated for the database administrator. However, if a fatal occurs you can execute the program further until it is resolved.

As a program it is your responsibility to look into possible errors and check the user's input that it is given in the correct format. For example, if we are taking data from a user either it is in GET form or POST. Here name1 has no value, valid name is given in name2, and in name3 some space like “%20”, it is hexadecimal value for space.

Types of Errors

Example query string: `id=0&name1=&name2=smith&name3=%20`

Notice that this parameter has no value.

This parameter's value is a space character (URL encoded).

<code>isset(\$_GET['id'])</code>	returns	true	
<code>isset(\$_GET['name1'])</code>	returns	true	Notice that a missing value for a parameter is still considered to be <code>isset</code> .
<code>isset(\$_GET['name2'])</code>	returns	true	
<code>isset(\$_GET['name3'])</code>	returns	true	
<code>isset(\$_GET['name4'])</code>	returns	false	Notice that only a missing parameter name is considered to be not <code>isset</code> .
<code>empty(\$_GET['id'])</code>	returns	true	Notice that a value of zero is considered to be empty. This may be an issue if zero is a "legitimate" value in the application.
<code>empty(\$_GET['name1'])</code>	returns	true	
<code>empty(\$_GET['name2'])</code>	returns	false	
<code>empty(\$_GET['name3'])</code>	returns	false	Notice that a value of space is considered to be not empty.
<code>empty(\$_GET['name4'])</code>	returns	true	

Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

To check the above value, we can use “isset()” and “empty()” functions. The function “isset()” indicates whether the value we want to check or read is available or not, while “empty()” function tells us that if the value exists then it is empty or not. For example, if we pass any variable like id, name1, name2 and name3 to the “isset ()” function it will return true. This function will not check the value that it exists or not, it is valid or not, it only checks that the variable exists. So, if we pass the variable name4 to the function it will return false as this variable doesn’t exist in the query string.

Similarly, if we pass these variables to “empty ()” function, in case of id it will return true as “id” has zero (0) value and zero stands for false or empty. Name1 is also empty so it will return true, while in name2 and name3 it will return false, and for name4 it will return true as it doesn’t exist.

Module 126

Type of Errors:

There are different types of errors we faced during web development.

Expected errors:

The most common errors expected errors, these errors occur mostly on the base of user input. Like users have not provided the required value or values are not in the required format. Such errors can be checked at runtime and handled logically.

Warnings:

The issues that arise during programming are called warnings and we can deal with them during programming development.

Fatal errors:

The errors which occur after deployment and stop the further execution of program/script are called fatal errors. For example, your web server and database server are running on two different machines, let's say database server went down so when you try to connect with a database it will throw an exception and the program will crash. If the program is handled appropriately, you will give appropriate measures like you can send a message to the user that "Please try again, we are facing some issues". Meanwhile, an email alert can be generated for the database administrator. However, if a fatal occurs you can execute the program further until it is resolved.

As a program it is your responsibility to look into possible errors and check the user's input that it is given in the correct format. For example, if we are taking data from a user either it is in GET form or POST. Here name1 has no value, valid name is given in name2, and in name3 some space like "%20", it is hexadecimal value for space.

Types of Errors

Example query string: `id=0&name1=&name2=smith&name3=%20`

Notice that this parameter has no value.

This parameter's value is a space character (URL encoded).

<code>isset(\$_GET['id'])</code>	returns	true	
<code>isset(\$_GET['name1'])</code>	returns	true	Notice that a missing value for a parameter is still considered to be <code>isset</code> .
<code>isset(\$_GET['name2'])</code>	returns	true	
<code>isset(\$_GET['name3'])</code>	returns	true	
<code>isset(\$_GET['name4'])</code>	returns	false	Notice that only a missing parameter name is considered to be not <code>isset</code> .
<code>empty(\$_GET['id'])</code>	returns	true	Notice that a value of zero is considered to be empty. This may be an issue if zero is a "legitimate" value in the application.
<code>empty(\$_GET['name1'])</code>	returns	true	
<code>empty(\$_GET['name2'])</code>	returns	false	
<code>empty(\$_GET['name3'])</code>	returns	false	Notice that a value of space is considered to be not empty.
<code>empty(\$_GET['name4'])</code>	returns	true	

Slides by Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

To check the above value, we can use “isset()” and “empty()” functions. The function “isset()” indicates whether the value we want to check or read is available or not, while “empty()” function tells us that if the value exists then it is empty or not. For example, if we pass any variable like id, name1, name2 and name3 to the “isset ()” function it will return true. This function will not check the value that it exists or not, it is valid or not, it only checks that the variable exists. So, if we pass the variable name4 to the function it will return false as this variable doesn’t exist in the query string.

Similarly, if we pass these variables to “empty ()” function, in case of id it will return true as “id” has zero (0) value and zero stands for false or empty. Name1 is also empty so it will return true, while in name2 and name3 it will return false, and for name4 it will return true as it doesn’t exist.

Module 127

PHP Error and Exception handling

When a fatal PHP error occurs, program execution will eventually terminate unless it is handled. The PHP documentation provides two mechanisms for handling runtime errors: procedural error handling and object-oriented exception handling.

Procedural Error Handling

In the procedural approach to error handling, the programmer needs to explicitly test for error conditions after performing a task that might generate an error.

```
$connection = mysqli_connect(DBHOST, DBUSER, DBPASS, DBNAME);

$error = mysqli_connect_error();

if ($error != null) {

    // handle the error

    ...

}
```

The advantage of the try ... catch mechanism is that it allows the developer to handle a wider variety of exceptions in a single catch block. Yet, even with explicit testing for error conditions, there will still be situations when an unforeseen error occurs. In such a case, unless a custom error handler has been defined, PHP will terminate the execution of the application.

Object-Oriented Exception Handling

When a runtime error occurs, PHP throws an exception. If an exception is not caught, then eventually the PHP environment will handle it by terminating execution with an “Uncaught Exception” message. Like other object-oriented programming languages, PHP uses the try ... catch programming construct to programmatically deal with exceptions at runtime.

Example of try ... catch block

```
// Exception throwing function

function throwException($message = null,$code = null) {

    throw new Exception($message,$code);

}

try {

    // PHP code here

    $connection = mysqli_connect(DBHOST, DBUSER, DBPASS, DBNAME)

    or throwException(“error”);

    ...

}
```

```
catch (Exception $e) {  
    echo ' Caught exception: ' . $e->getMessage();  
    echo ' On Line : ' . $e->getLine();  
    echo ' Stack Trace: '; print_r($e->getTrace());  
} finally {  
    // PHP code here that will be executed after try or after catch  
}
```

The finally block is optional. Any code within it will always be executed after the code in the try or in the catch blocks, even if that code contains a return statement. It is typically used if the developer wants certain things done regardless of whether an exception occurred, such as closing a connection or removing temporary files. However, the finally block is only available in PHP 5.5 and later, which was released in June 2013.

Module 128

Regular Expressions

A regular expression is a set of special characters that define a pattern. They are a type of language that is intended for the matching and manipulation of text. In web development they are commonly used to test whether a user's input matches a predictable sequence of characters, such as those in a phone number, postal or zip code, or email address.

Regular Expression Syntax

A regular expression consists of two types of characters: literals and metacharacters. A literal is just a character you wish to match in the target (i.e., the text that you are searching within). A metacharacter is a special symbol that acts as a command to the regular expression parser. There are 14 metacharacters described in the php implementation. To use a metacharacter as a literal, you will need to escape it by prefacing it with a backslash (\).

Table : Common Regular Expression Patterns

Pattern	Description
^	If used at the very start and end of the regular expression, it means that the entire string (and not just a substring) must match the rest of the regular expression contained between the ^ and the \$ symbols.
\$	
\t	Matches a tab character.
\n	Matches a new-line character.
.	Matches any character other than \n.
[qwerty]	Matches any single character of the set contained within the brackets.
[^qwerty]	Matches any single character not contained within the brackets.
[a-z]	Matches any single character within range of characters.
\w	Matches any word character. Equivalent to [a-zA-Z0-9_].
\W	Matches any nonword character.
\s	Matches any white-space character.
\S	Matches any nonwhite-space character.
\d	Matches any digit.
\D	Matches any nondigit.
*	Indicates zero or more matches.
+	Indicates one or more matches.
?	Indicates zero or one match.
{n}	Indicates exactly n matches.
{n,}	Indicates n or more matches.
{n, m}	Indicates at least n but no more than m matches.
 	Matches any one of the terms separated by the character. Equivalent to Boolean OR.
()	Groups a subexpression. Grouping can make a regular expression easier to understand.

Module 129

Regular Expressions – Examples

In PHP, regular expressions are contained within forward slashes. So, for instance, to define a regular expression, you would use the following:

```
$pattern = '/ran/';
```

It should be noted that regular expression pattern checks are case sensitive. Regular expressions can be complicated to visually decode; to help, this section will use the convention of alternating between red and blue to indicate distinct subpatterns in an expression and black text for literals.

This regular expression will find matches in all three of the following strings.

```
'randy connolly'
```

```
'Sue ran to the store'
```

```
'I would like a cranberry'
```

To perform the pattern check in PHP, you would write something similar to the following:

```
$pattern = '/ran/;  
$check = 'Sue ran to the store';  
if ( preg_match($pattern, $check) ) {  
echo 'Match found!';  
}
```

To perform the same pattern check in JavaScript, you would write something similar to the following:

```
var pattern = /ran/;  
if ( pattern.test('Sue ran to the store') ) {  
document.write('Match found!');  
}
```

In JavaScript a regular expression is its own data type. Just as a string literal begins and ends with quote characters, in JavaScript, a regular expression literal begins and ends with forward slashes.

Module 130

PHP Filters

In PHP, filters are used to validate the input against pre-defined patterns. There are two types of filters: validate filters and sanitize filters. Following function is used for aforementioned task:

```
Filter_var($value,FILTER)
```

Validate Filters

- Check input against a pre-defined pattern
- Validate email (FILTER_VALIDATE_EMAIL)

```
$email = filter_var('abc@mail.com', FILTER_VALIDATE_EMAIL);
```

```
// $email = abc@mail.com
```

```
$email = filter_var('abc@.com', FILTER_VALIDATE_EMAIL);
```

```
// $email=null
```

Other validate filters

FILTER_VALIDATE_BOOL, FILTER_VALIDATE_FLOAT, FILTER_VALIDATE_INT
FILTER_VALIDATE_MAC

FILTER_VALIDATE_IP

FILTER_VALIDATE_URL

FILTER_VALIDATE_REGEX

Validate against a regular expression (FILTER_VALIDATE_REGEX)

```
$cnic = filter_var('12345-1234567-8', FILTER_VALIDATE_REGEX, array("options" =>  
array("regex" => "/^\d{5}-?\d{7}-?\d{8}$/")));
```

```
if($cnic == null) {
```

```
    echo "Invalid CNIC";
```

```
} else {
```

```
    echo "Valid CNIC $cnic";
```

```
}
```

Sanitize Filters

Firstly, Remove or replace specific characters

FILTER_SANITIZE_EMAIL

Removes all characters except letters, digits and !#\$%&'*+,-=?@{1}7©.[]

```
Filter_var('my(email)@mail/.com/', SANITIZE) == 'myemail@mail.com'
```

`FILTER_SANITIZE_STRING`

Strips HTML tags

```
filter_var('<h1>A" B</h1>',FILTER_SANITIZE_FULL_SPECIAL_CHARS)=='A&#34; &#39; B '
```

`FILTER_SANITIZE_FULL_SPECIAL_CHARS`

```
filter_var('<h1>A"&#39;B</h1>',FILTER_SANITIZE_FULL_SPECIAL_CHARS)=='&#39;h1&gt;A&quot; &#39; B&lt;/h1 &gt;'
```

Following are few sanitize filters:

`FILTER_SANITIZE_ADD_SLASHES`

`FILTER_SANITIZE_ENCODED`

`FILTER_SANITIZE_NUMBER_FLOAT`

`FILTER_SANITIZE_NUMBER_INT`

`FILTER_SANITIZE_SPECIAL_CHARS`

`FILTER_SANITIZE_URL`

Module 131

Validating User Input

The following list indicates most of the common types of user input validation.

Required information: Some data fields just cannot be left empty. For instance, the principal name of things or people is usually a required field. Other fields such as emails, phones, or passwords are typically required values.

Correct data type. While some input fields can contain any type of data, other fields, such as numbers or dates, must follow the rules for its data type in order to be considered valid.

Correct format: Some information, such as postal codes, credit card numbers, and social security numbers have to follow certain pattern rules. It is possible, however, to go overboard with these types of checks.

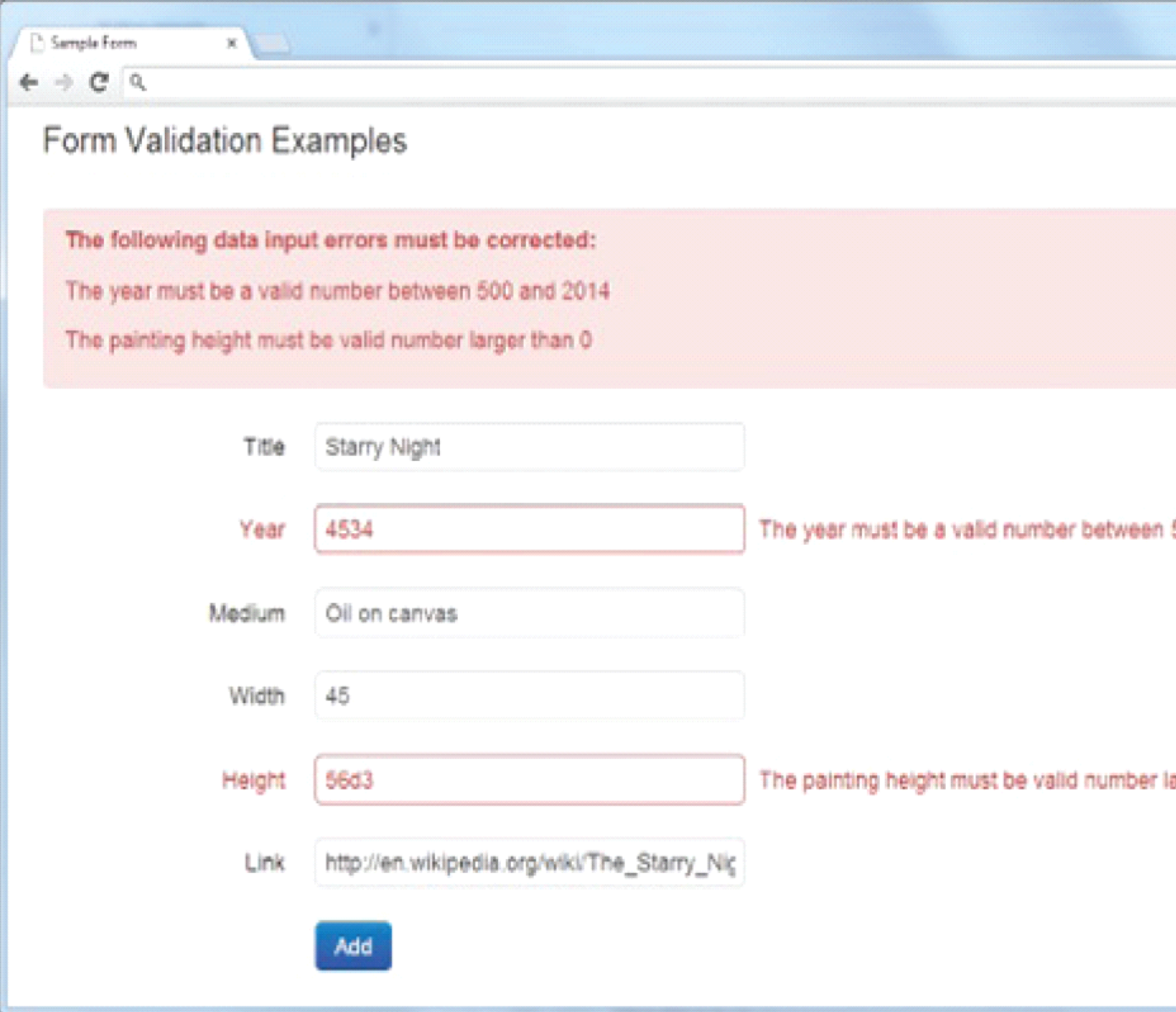
Comparison: Some user-entered fields are considered correct or not in relation to an already-inputted value. Perhaps the most common example of this type of validation is entering passwords: most sites require the user to enter the password twice and then a comparison is made to ensure the two entered values are identical.

Range check: Information such as numbers and dates have infinite possible values: However, most systems need numbers and dates to fall within realistic ranges. For instance, if you are asking a user to input her birthday, it is likely you do not want to accept January 1, 214 as a value; it is quite unlikely

she is 1800 years old! As a result, almost every number or date should have some type of range check performed.

Custom: Some validations are more complex and are unique to a particular application. Some custom validations can be performed on the client side.

Notifying the User



The screenshot shows a web browser window with a tab titled "Sample Form". The address bar contains navigation icons and a search icon. The main content area is titled "Form Validation Examples". Below the title, there is a red-shaded box containing the following text:

The following data input errors must be corrected:

- The year must be a valid number between 500 and 2014
- The painting height must be valid number larger than 0

Below the error messages, there are several input fields:

- Title:
- Year: (This field has a red border and a red error message to its right: "The year must be a valid number between 500 and 2014".)
- Medium:
- Width:
- Height: (This field has a red border and a red error message to its right: "The painting height must be valid number larger than 0".)
- Link:

At the bottom of the form, there is a blue button labeled "Add".

Sample Form

Form Validation Examples

Title

Enter the painting title

The title is required (it cannot be blank)

Year

Enter the year of the painting

The year must be a valid number between 500 and 2014

Medium

Oil on canvas

Width

45

Height

Enter the height in cm of the painting

The painting height must be valid number larger than 0

Link

Enter Wikipedia link for painting

Add

Static textual hints

Sample Form

Form Validation Examples

Title
Required

Year
The year of the painting must be a valid number between 500 and 2014

Medium
The painting medium (e.g., oil on board, acrylic on canvas)

Width
The optional painting height must be valid number larger than 0

Height
The optional painting height must be valid number larger than 0

Link
If there is a wikipedia page for this painting, enter its URL here

Placeholder text
(visible until user enters a value into field)

```
<input type="text" ... placeholder="Enter the height ..." >
```

Pop-up tool tip
(appears when mouse
hovered over icon)

The screenshot shows a web browser window with a form titled "Form Validation Examples". The form contains the following fields:

- Title**: "Enter the painting title" with a "required" tooltip and a question mark icon.
- Year**: "Enter the year" with a question mark icon.
- Medium**: "Enter the medium type of the painting" with a question mark icon.
- Width**: "Enter the width in cm of" with a question mark icon.
- Height**: "Enter the height in cm of" with a question mark icon.
- Link**: "Enter Wikipedia link for painting" with a question mark icon and a "Hint" tooltip.

Below the fields is an "Add" button. Red arrows point from the text annotations to the "required" tooltip, the question mark icons, and the "Hint" tooltip.

Pop-over

The screenshot shows a web browser window with a form titled "Form Masking Examples". The form contains the following fields:

- Phone**: An input field with a mask pattern of "() _ _ _ _ _".
- Date of Birth**: An input field with a mask pattern of " _ / _ / _ _".
- Credit Card**: An input field with a mask pattern of " _ _ _ _ _ _ _ _".

Below the fields is an "Add" button.

Input fields with masks

The screenshot shows a web browser window with a form titled "Form Masking Examples". The form contains the following fields:

- Phone**: An input field with the value "(403) 444-6666".
- Date of Birth**: An input field with the value "23/May/2013".
- Credit Card**: An input field with the value " _ _ _ _ _ _ _ _".

Below the fields is an "Add" button.

The screenshot shows a web browser window with a form titled "Form Masking Examples". The form contains the following fields:

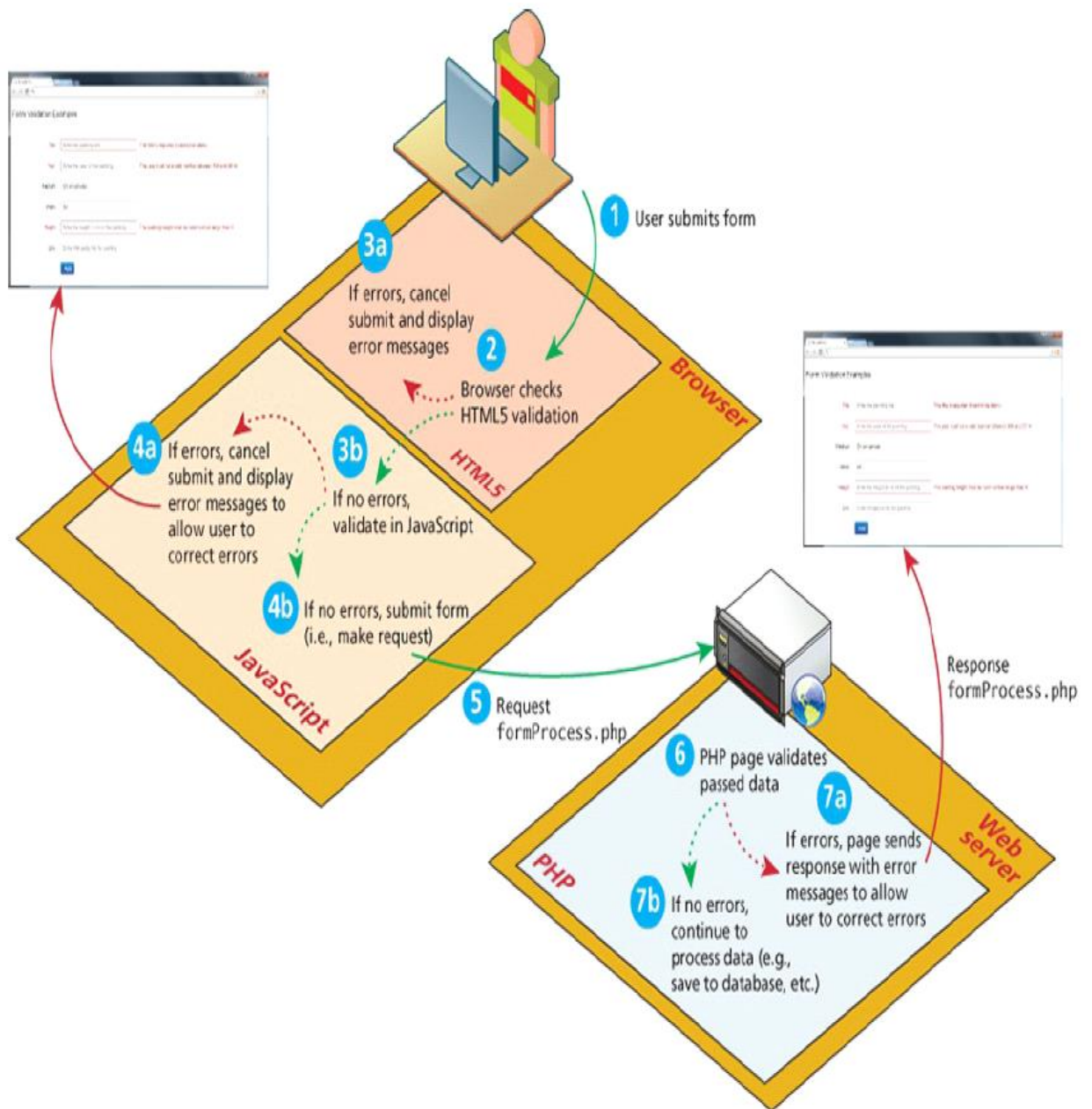
- Phone**: An input field with the value "(403) 444-6666".
- Date of Birth**: An input field with the value " _ / _ / _ _".
- Credit Card**: An input field with the value " _ _ _ _ _ _ _ _".

Below the fields is an "Add" button.

Module 132

Where to perform validation

Validation can be performed at three different levels. With HTML5, the browser can perform basic validation with no need for any JavaScript. However, since the validation that can be achieved in HTML5 is quite basic, most web applications also perform validation in the browser using JavaScript. The advantage of validation using JavaScript is that it reduces server load and provides immediate feedback to the user. Unfortunately, JavaScript validation cannot be relied on: for instance, it might be turned off on the user's browser. For these reasons, validation must always be done on the server side. Indeed, you should always perform the same validity checks on both the client in JavaScript and on the server in PHP, but server-side validation is the most important since it is the only validation that is guaranteed to run. Below figure illustrates the interaction of the different levels of validation.



Validation at the PHP Level

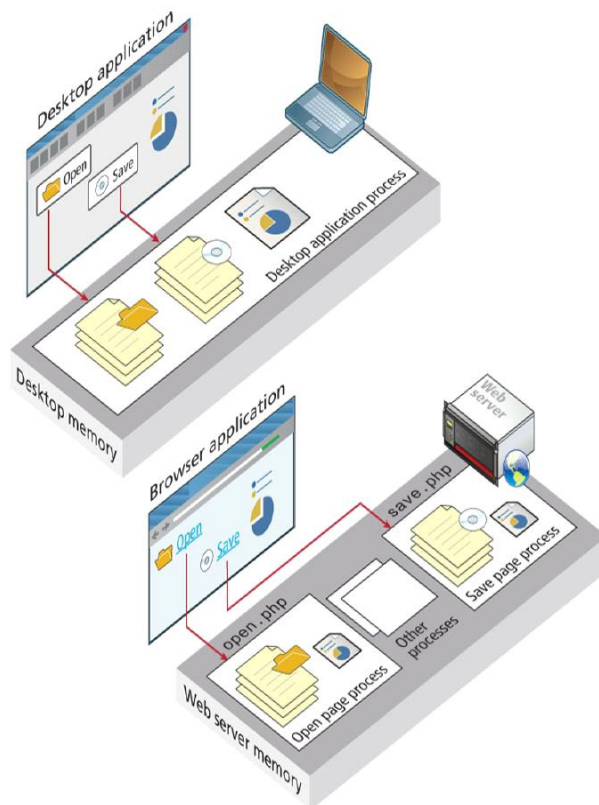
No matter how good the HTML5 and JavaScript validation, client-side prevalidation can always be circumvented by hackers, or turned off by savvy users. Validation on the server side using PHP is the most important form of validation and the only one that is absolutely essential. In this case, we will be validating the query string parameters rather than the form elements directly as with JavaScript.

Module 133

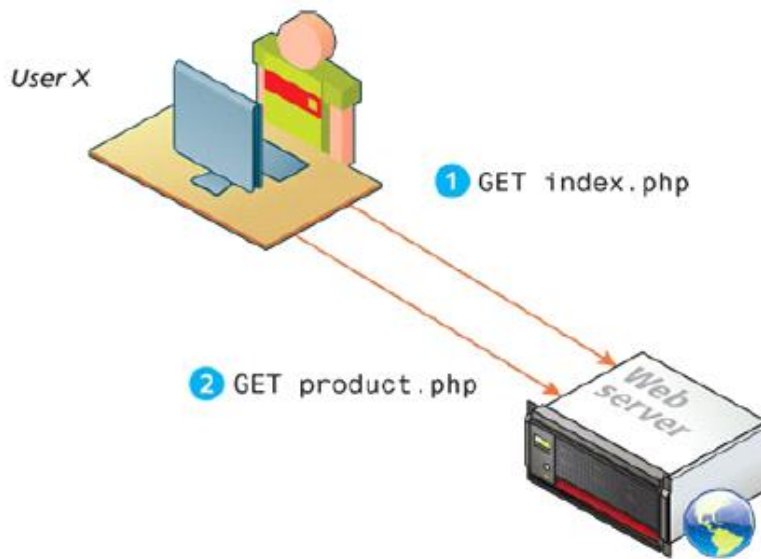
The Problem of State Management

Almost all applications need to process user inputs, output information, and read and write from databases or other storage media. But in this chapter we will be examining a development problem that is unique to the world of web development: how can one request share information with another request?

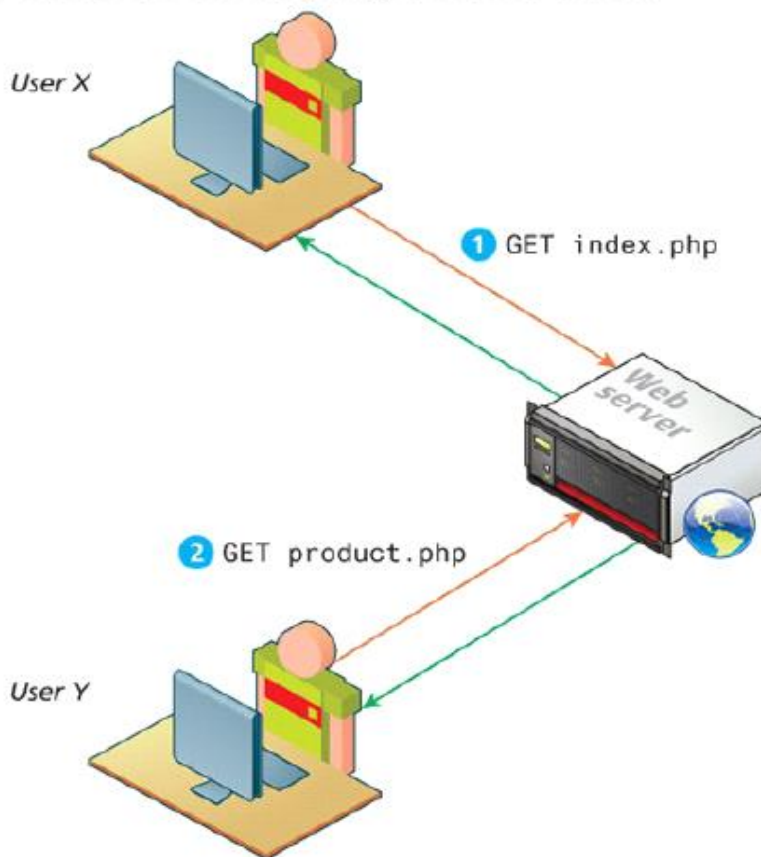
Single-user desktop applications do not have this challenge at all because the program information for the user is stored in memory and can thus be easily accessed throughout the application. Yet one must always remember that web applications differ from desktop applications in a fundamental way. Unlike the unified single process that is the typical desktop application, a web application consists of a series of disconnected HTTP requests to a web server where each request for a server page is essentially a request to run a separate program, as shown in below figure.



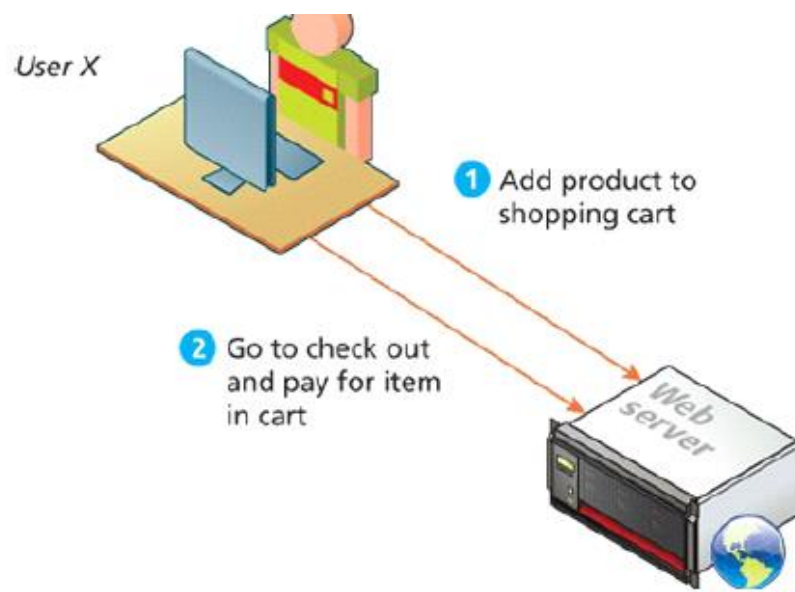
Furthermore, the web server sees only requests. The HTTP protocol does not, without programming intervention, distinguish two requests by one source from two requests from two different sources, as shown in below figure



... is for the server not really any different than ...

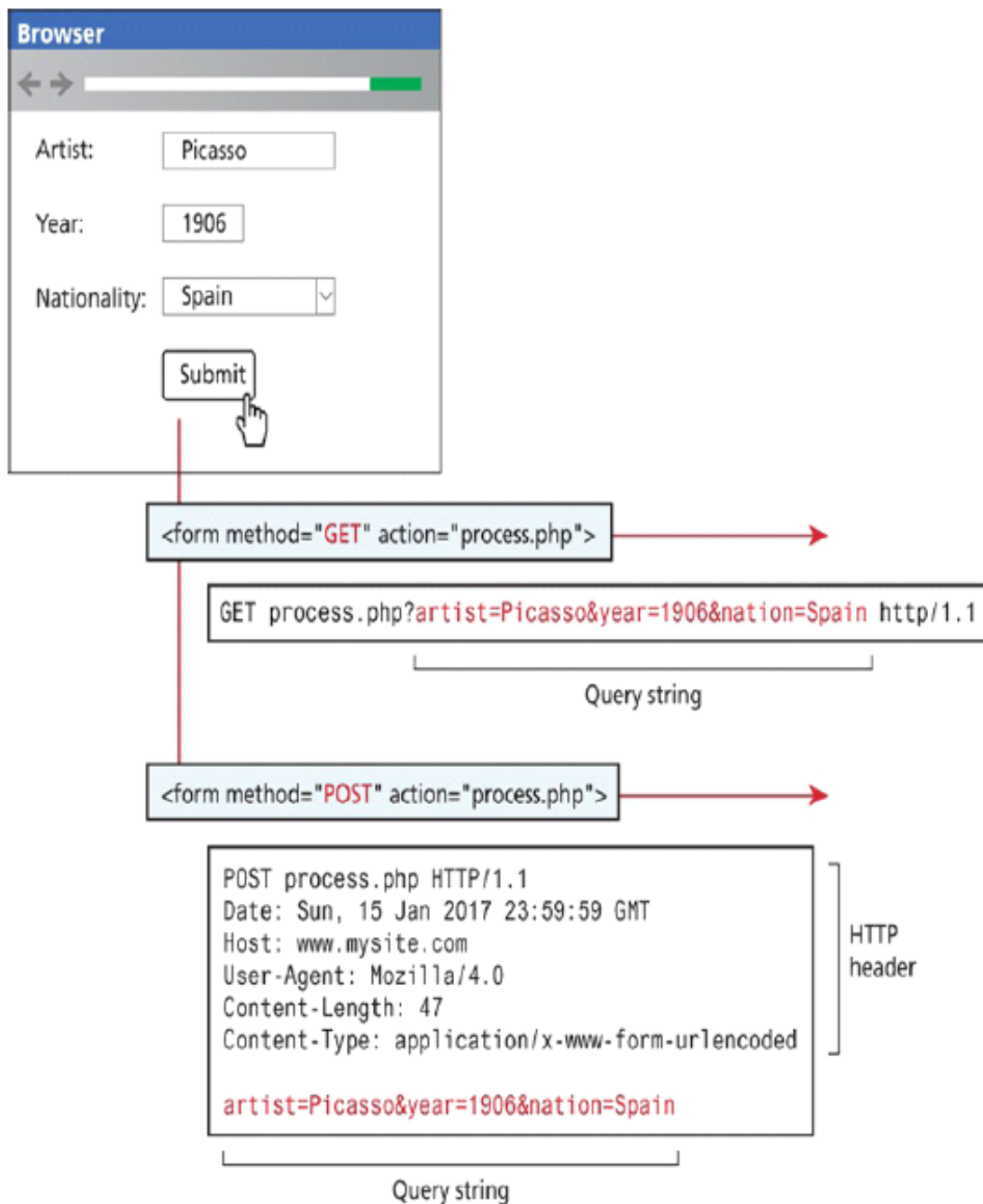


While the HTTP protocol disconnects the user's identity from his or her requests, there are many occasions when we want the web server to connect requests together. Consider the scenario of a web shopping cart, as shown in the below Figure. In such a case, the user most certainly wants the server to recognize that the request to add an item to the cart and the subsequent request to check out and pay for the item in the cart are connected to the same individual.



Module 134

A web page can pass query string information from the browser to the server using one of the two methods: a query string within the URL (GET) and a query string within the HTTP header (POST). Below Figure reviews these two different approaches.



Dynamic vs Static URLs

- Dynamic URLs (i.e., query strings)

A pretty essential part of web application development

- Static URLs

The appearance of search terms within the URL does seem to improve its relative position. Another benefit to static URLs is that users tend to prefer them.

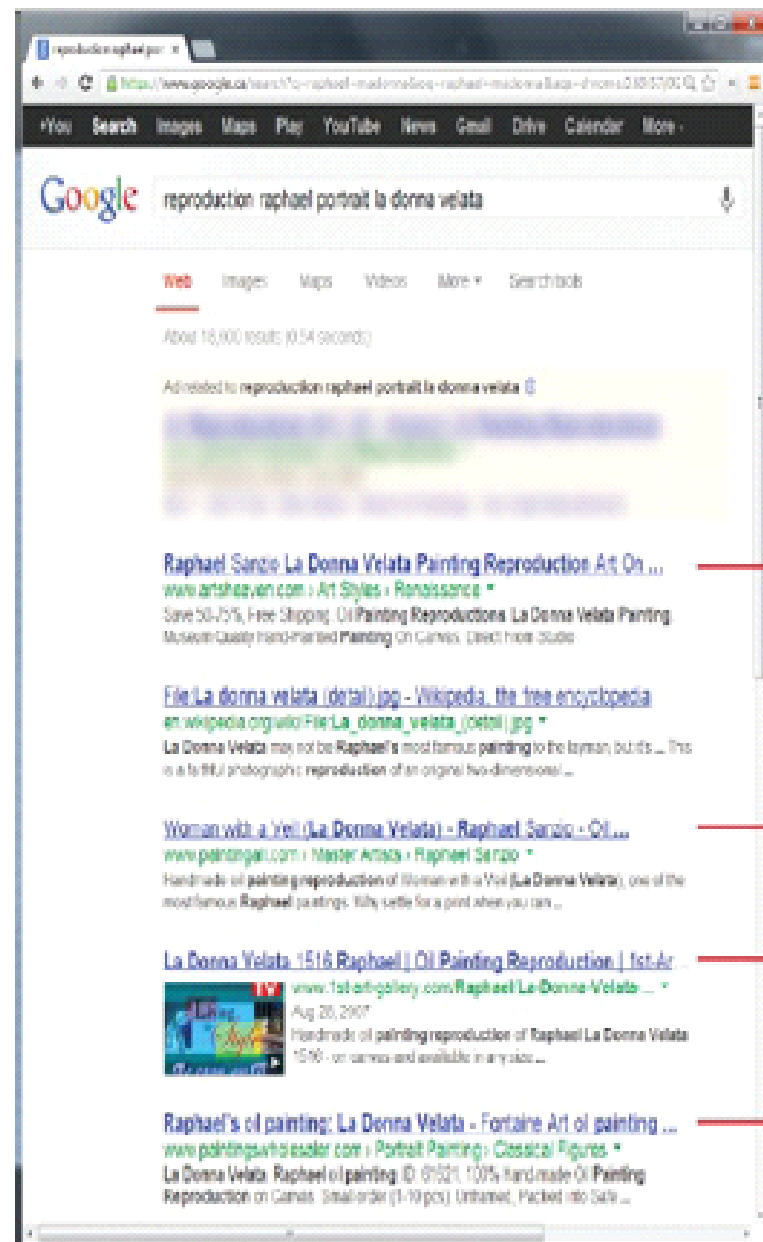
- Can use URL rewriting to make static URLs be dynamic

Depending on your web development platform, there are different ways to implement URL rewriting. On web servers running Apache, the solution typically involves using the `mod_rewrite` module in Apache along with the `.htaccess` file. The `mod_rewrite` module uses a rule-based rewriting engine that utilizes Perl-compatible regular expressions to change the URLs so that the requested URL can be mapped or redirected to another URL internally.

<http://www.1st-art-gallery.com/Raphael/La-Donna-Velata-1516.html> ←

<http://www.paintingall.com/raphael-sanzio-woman-with-a-veil-la-donna-velata>

<http://www.artsheaven.com/raphael-la-donna-velata.html>



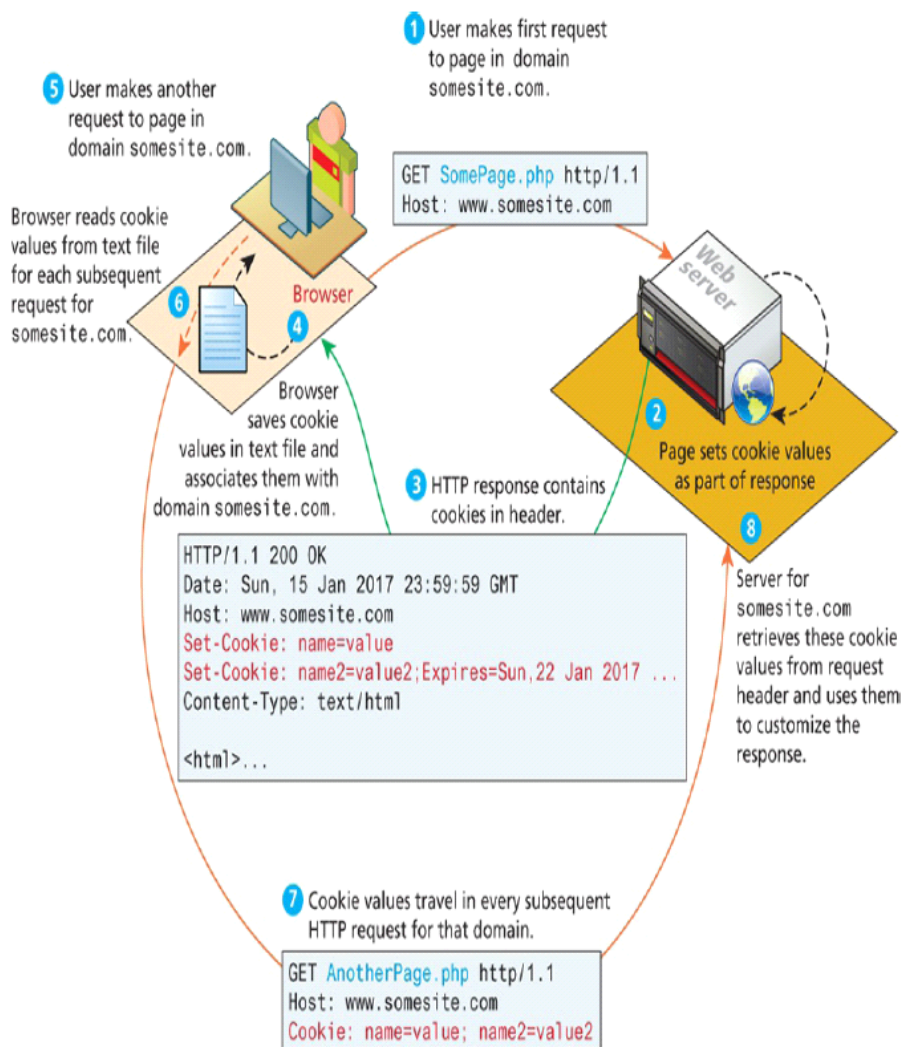
<http://www.paintingswholesaler.com/detail.asp?vcode=6umd7krr1yqi161c&title=La+D>

Module 135

Cookies are a client-side approach for persisting state information. Cookies were intended to be a long-term state mechanism. They provide website authors with a mechanism for persisting user-related information that can be stored on the user's computer and be managed by the user's browser. Cookies are not associated with a specific page but with the page's domain, so the browser and server will exchange cookie information no matter what page the user requests from the site. The browser manages the cookies for the different domains so that one domain's cookies are not transported to a different domain.

How Do Cookies Work

While cookie information is stored and retrieved by the browser, the information in a cookie travels within the HTTP header. Below figure illustrates how cookies work.



Due to the limitations of cookies (both in terms of size and reliability), your site's correct operation should not be dependent upon cookies. Nonetheless, the user's experience might be improved with the judicious use of cookies.

It is important to note that cookies must be written before any other page output.

```
<?php
// add 1 day to the current time for expiry time
$expiryTime = time()+60*60*24;

// create a persistent cookie
$name = "Username";
$value = "Ricardo";

setcookie($name, $value, $expiryTime);

?>
```

The `setcookie()` function also supports several more parameters, which further customize the new cookie. Notice that when we read a cookie, we must also check to ensure that the cookie exists. In PHP, if the cookie has expired (or never existed in the first place), then the client's browser would not send anything, and so the `$_COOKIE` array would be blank.

Module 136

Reading a cookie

```
<?php
if( !isset($_COOKIE['Username']) ) {

//no valid cookie found

}

else {

echo "The username retrieved from the cookie is:";

echo $_COOKIE['Username'];

}

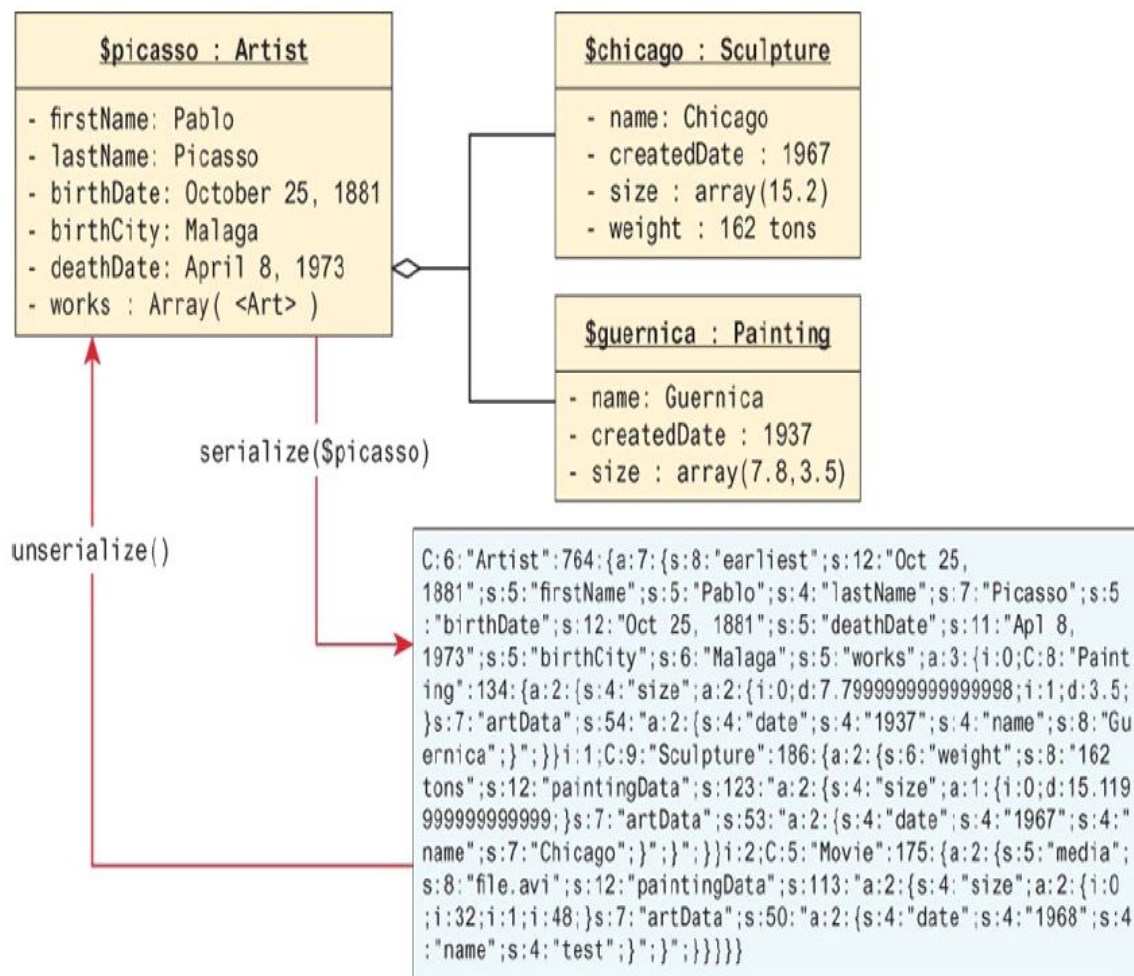
?>
```

Many sites provide a “Remember Me” checkbox on login forms, which relies on the use of a persistent cookie. This login cookie would contain the user's username but not the password. Instead, the login cookie would contain a random token; this random token would be stored along with the username in the site's back-end database. Every time the user logs in, a new token would be generated and stored in the database and cookie. Another common use of cookies is to track a user's browsing behavior on a site. Some sites will store a pointer to the last requested page in a cookie; this information can be used by the site administrator as an analytic tool to help understand how users navigate through the site.

Module 137

Serialization

Serialization is the process of taking a complicated object and reducing it down to zeros and ones for either storage or transmission. Later that sequence of zeros and ones can be reconstituted into the original object as illustrated in the below Figure.



Artist class modified to implement the Serializable interface

```
class Artist implements Serializable {  
    //...  
    // Implement the Serializable interface methods  
    public function serialize() {  
        // use the built-in PHP serialize function
```

```

return serialize(
    array("earliest" => self::$earliestDate,
        "first" => $this->firstName,
        "last" => $this->lastName,
        "bdate" => $this->birthDate,
        "ddate" => $this->deathDate,
        "bcity" => $this->birthCity,
        "works" => $this->artworks
    )
);
}

public function unserialize($data) {
    // use the built-in PHP unserialize function

    $data = unserialize($data);

    self::$earliestDate = $data['earliest'];
    $this->firstName = $data['first'];
    $this->lastName = $data['last'];
    $this->birthDate = $data['bdate'];
    $this->deathDate = $data['ddate'];
    $this->birthCity = $data['bcity'];
    $this->artworks = $data['works'];
}

//...
}

```

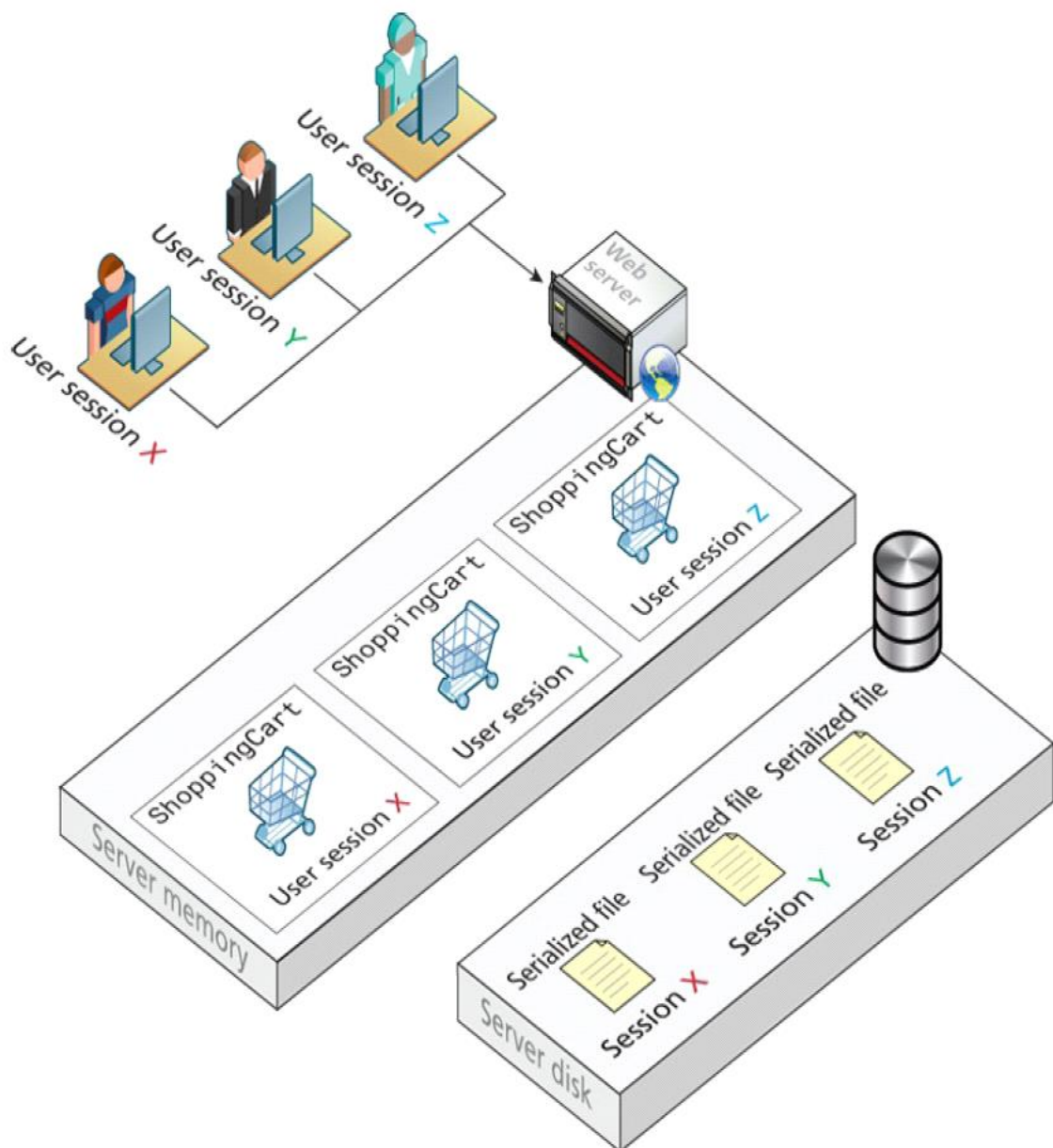
Note that in order for our Artist class to save successfully, the Art, Painting, and other classes must also implement the Serializable interface. It should be noted that references to other objects stored at the same time will be preserved while references to objects not serialized in this operation will be lost. Although nearly unreadable to most people, this data can easily be used to reconstitute the object by passing it to unserialize(). If the data above is assigned to \$data, then the following line will instantiate a new object identical to the original:

```
$picassoClone = unserialize($data);
```

Module 138

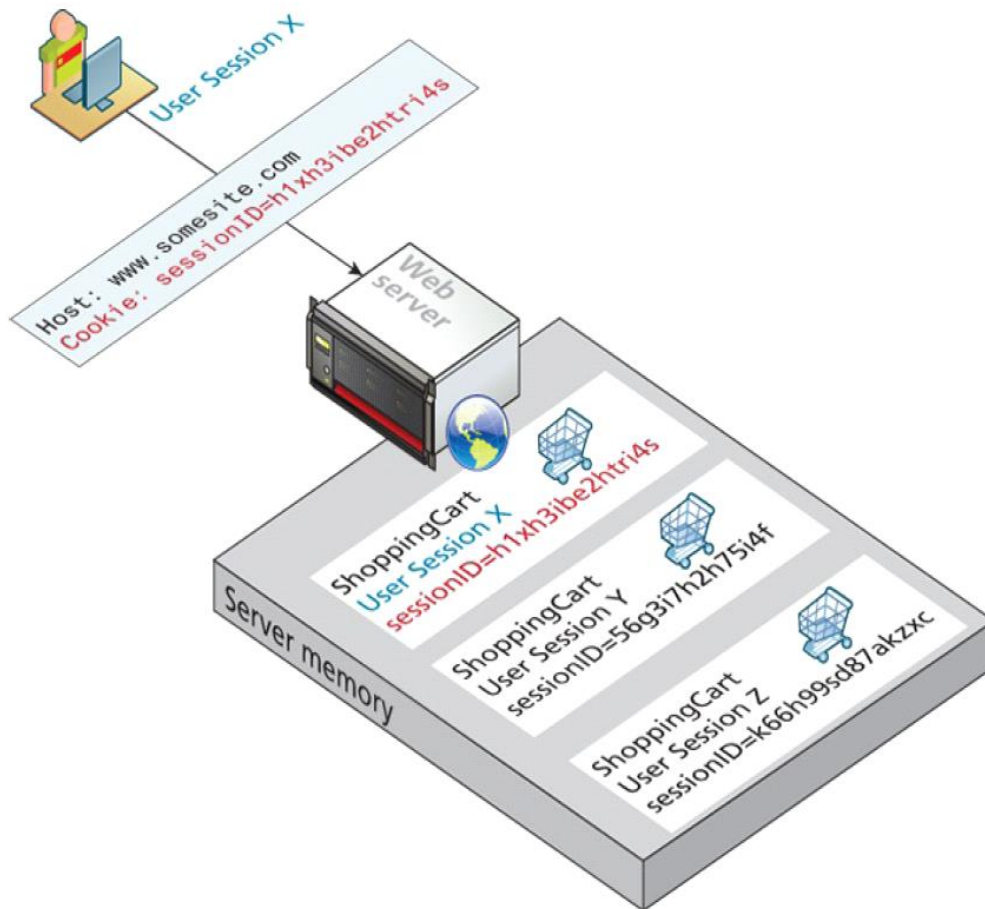
Session State using sessions

All modern web development environments provide some type of session state mechanism. Session state is a server-based state mechanism that lets web applications store and retrieve objects of any type for each unique user session. That is, each browser session has its own session state stored as a serialized file on the server, which is deserialized and loaded into memory as needed for each request, as shown in the below figure.



How Does Session State Work?

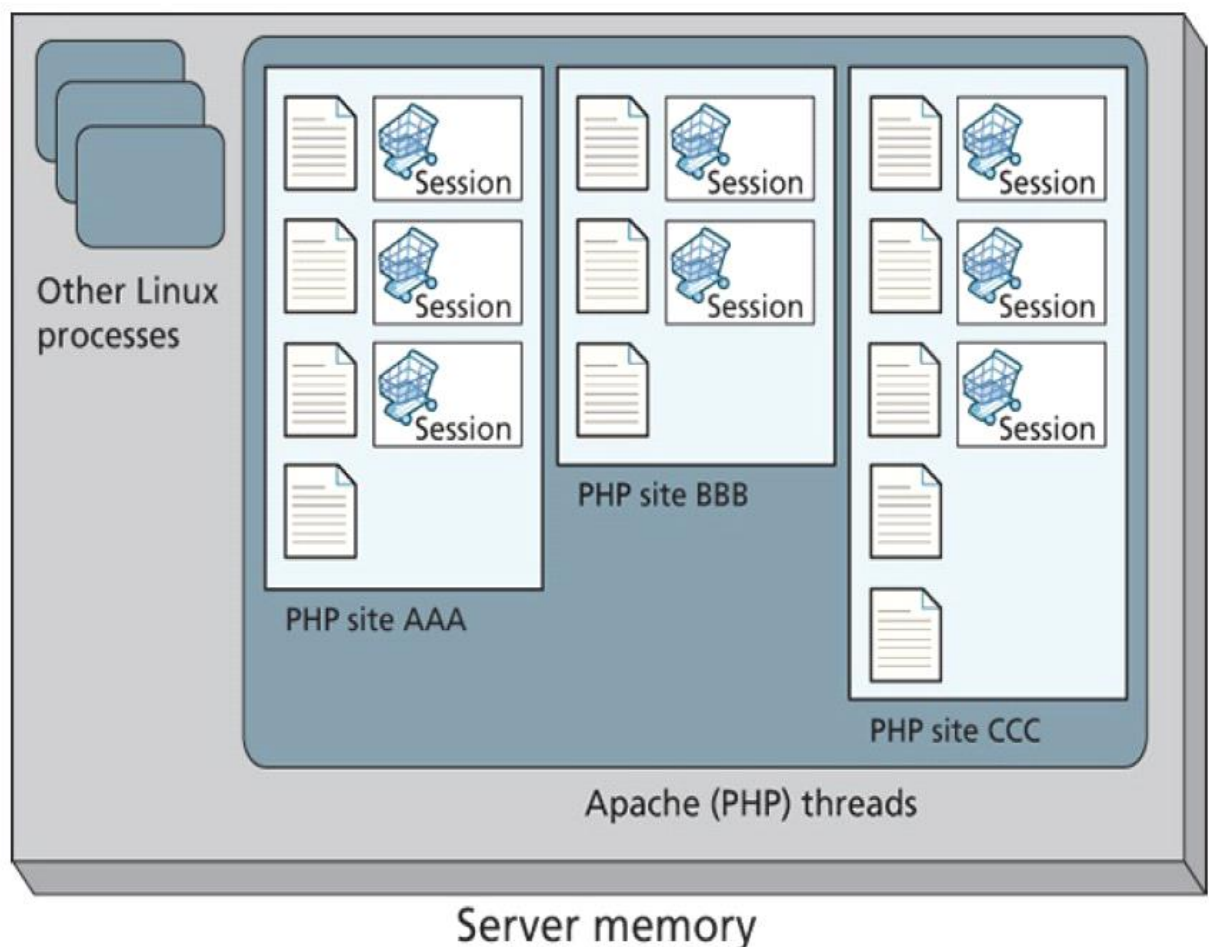
Modern development environments such as ASP.NET and PHP make session state remarkably easy to work with, it is tempting to see session state as a one-stop solution to all web state needs. However, if we take a closer look at how session state works, we will see that session state has the same limitations and issues as the other state mechanisms. The first thing to know about session state is that it works within the same HTTP context as any web request. The server needs to be able to identify a given HTTP request with a specific user request. Since HTTP is stateless, some type of user/session identification system is needed. Sessions in PHP (and ASP.NET) are identified with a unique session ID. In PHP, this is a unique 32-byte string that is by default transmitted back and forth between the user and the server via a session cookie, as shown in below figure.



Module 139

Session Storage and Configuration

Each user's session information is kept in serialized files, one per session. It is possible to configure many aspects of sessions including where the session files are saved. For a complete listing refer to the session configuration options in `php.ini`. The decision to save sessions to files rather than in memory addresses the issue of memory usage that can occur on shared hosts as well as persistence between restarts. Many sites run in commercial hosting environments that are also hosting many other sites.



Session state is ideal for storing more complex objects or data structures that are associated with a user session. The classic example is a shopping cart. While shopping carts could be implemented via cookies or query string parameters, it would be quite complex and cumbersome to do so. In PHP, session state is available to the developer as a superglobal associative array, much like the `$_GET`, `$_POST`, and `$_COOKIE` arrays.³ It can be accessed via the `$_SESSION` variable, but unlike the other superglobals, you have to take additional steps in your own code in order to use the `$_SESSION` superglobal. To use sessions in a script, you must call the `session_start()` function at

the beginning of the script.

Accessing session state

```
<?php
session_start();

if ( isset($_SESSION['user']) ) {
    // User is logged in
}

else {
    // No one is logged in (guest)
}

?>
```

Session state is typically used for storing information that needs to be preserved across multiple requests by the same user. Since each user session has its own session state collection, it should not be used to store large amounts of information because this will consume very large amounts of server memory as the number of active sessions increase. As well, since session information does eventually time out, one should always check if an item retrieved from session state still exists before using the retrieved object..

Checking session existence

```
<?php
include_once("ShoppingCart.class.php");
session_start();

// always check for existence of session object before accessing it
if ( !isset($_SESSION["Cart"]) ) {
    // session variables can be strings, arrays, or objects, but
    // smaller is better
    $_SESSION["Cart"] = new ShoppingCart();
}

$cart = $_SESSION["Cart"];

?>
```

Module 140

HTML5 web storage

Web storage is a new JavaScript-only API introduced in HTML5.4. In addition, web storage is not limited to the 4K size barrier of cookies; the W3C recommends a limit of 5MB but browsers are allowed to store more per domain. Currently web storage is supported by current versions of the major browsers, including IE8 and above. However, since JavaScript, like cookies, can be disabled on a user's browser, web storage should not be used for mission-critical application functions.

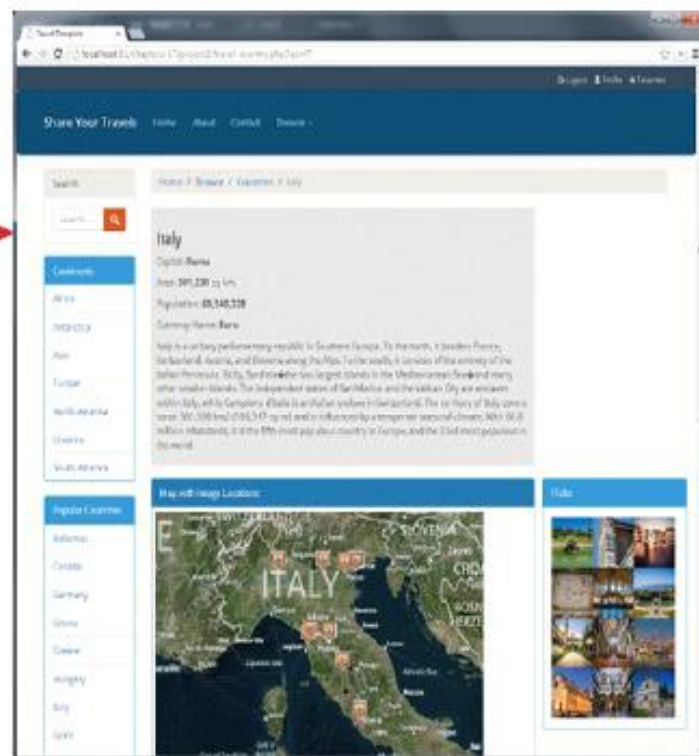
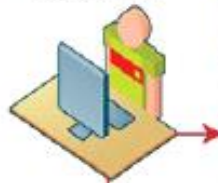
Why Would We Use Web Storage?

Cookies have the disadvantage of being limited in size, potentially disabled by the user, vulnerable to XSS and other security attacks, and being sent in every single request and response to and from a given domain. On the other hand, the fact that cookies are sent with every request and response is also their main advantage: namely, that it is easy to implement data sharing between the client browser and the server. Unfortunately with web storage, transporting the information within web storage back to the server is a relatively complicated affair involving the construction of a web service on the server and then using asynchronous communication via JavaScript to push the information to the server.

A better way to think about web storage is not as a cookie replacement but as a local cache for relatively static items available to JavaScript. One practical use of web storage is to store static content downloaded asynchronously such as XML or JSON from a web service in web storage, thus reducing server load for subsequent requests by the session.

Below figure illustrates an example of how web storage could be used as a mechanism for reducing server data requests, thereby speeding up the display of the page on the browser, as well as reducing load on the server.

1 User requests page (PHP)



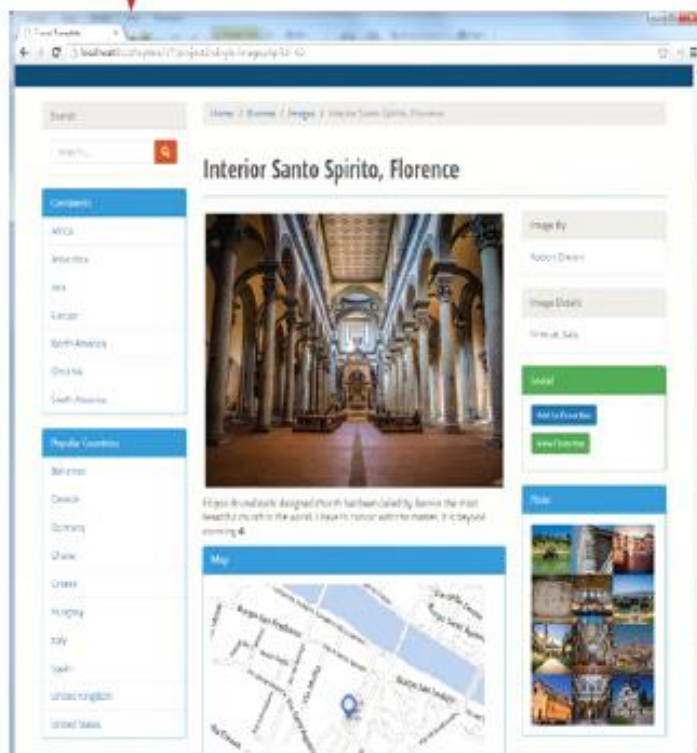
2 XML retrieved from flickr REST web service (JavaScript)



Web service server

4 User requests a related page (PHP)

3 XML saved in browser's web storage (JavaScript)



5 XML retrieved from browser's web storage (JavaScript) ...

6 ... and browser displays XML data (JavaScript), saving a second request to the flickr REST web service

Writing web storage

```
<form ... >

<h1>Web Storage Writer</h1>

<script language="javascript" type="text/javascript">

if (typeof (localStorage) === "undefined" ||
typeof (sessionStorage) === "undefined") {

alert("Web Storage is not supported on this browser...");

}

else {

sessionStorage.setItem("TodaysDate", new Date());

sessionStorage.FavoriteArtist = "Matisse";

localStorage.UserName = "Ricardo";

document.write("web storage modified");

}

</script>

<p><a href="WebStorageReader.php">Go to web storage reader</a></p>

</form>
```

Reading web storage

```
<form id="form1" runat="server">

<h1>Web Storage Reader</h1>

<script language="javascript" type="text/javascript">

if (typeof (localStorage) === "undefined" ||
typeof (sessionStorage) === "undefined") {

alert("Web Storage is not supported on this browser...");

}

else {

var today = sessionStorage.getItem("TodaysDate");

var artist = sessionStorage.FavoriteArtist;

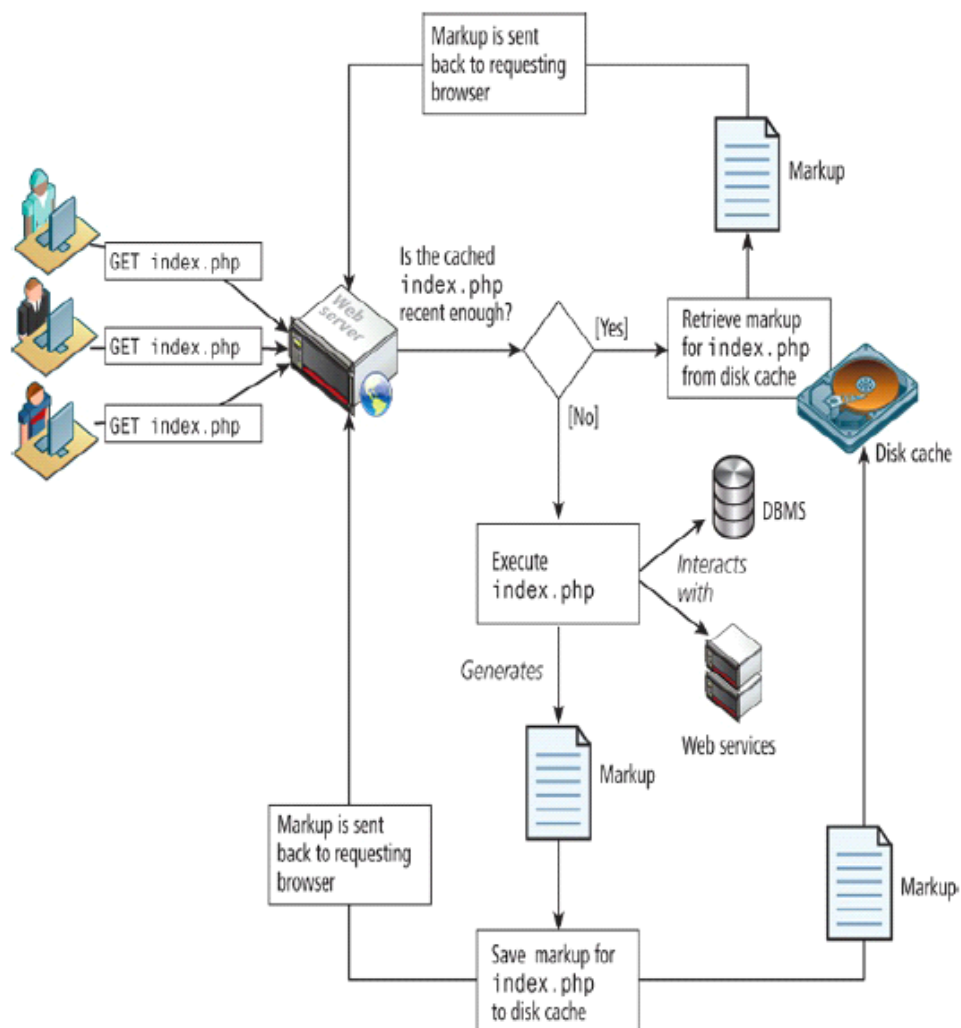
var user = localStorage.UserName;
```

```
document.write("date saved=" + today);  
document.write("<br/>favorite artist=" + artist);  
document.write("<br/>user name = " + user);  
}  
</script>  
</form>
```

Module 141

Caching

Caching is a vital way to improve the performance of web applications. Your browser uses caching to speed up the user experience by using locally stored versions of images and other files rather than re-requesting the files from the server. While important, from a serverside perspective, a server-side developer only has limited control over browser caching. In this type of caching, the contents of the rendered PHP are written to disk for fast retrieval. This can be particularly helpful because it allows PHP to send a page response to a client without going through the entire page processing life cycle again. Page output caching is especially useful for pages whose content does not change frequently but which require significant processing to create.



Application Data Caching

One of the biggest drawbacks with page output caching is that performance gains will only be had if the entire cached page is the same for numerous requests. However, many sites customize the content on each page for each user, so full or partial page caching may not always be possible. An alternate strategy is to use application data caching in which a page will programmatically place commonly used collections of data that require time intensive queries from the database or web server into cache memory, and then other pages that also need that same data can use the cache version rather than re-retrieve it from its original location.

Module 142

PHP with Ajax

AJAX requests are initiated on client side using JavaScript. The request is received by the server-side script (PHP), just like form data is received. The result is sent to the client side. The client receives the input and update the page.

JavaScript Side

An asynchronous (non-blocking) XML Http Request (XHR) is sent

- States of XHR object

0 UNSET

1 OPENED

2 HEADERS RECEIVED

3 LOADING

4 DONE

Example - JavaScript Request

```
var xhr = new XMLHttpRequest();

xhr.onreadystatechange = function() {
    if (this.readyState == XMLHttpRequest.DONE
        &&(this.status>=200&&this.status < 400)) {
        document.getElementById("updates").innerHTML = this.responseText;
    }
    xhr.open("GET", "http://localhost/getupdates.php?q=" + str, true);
    xhr.send();
}
```

Example - jQuery Request

```
var xhr = new XMLHttpRequest();

$.get("http://localhost/getupdates.php?q=" + str,
```

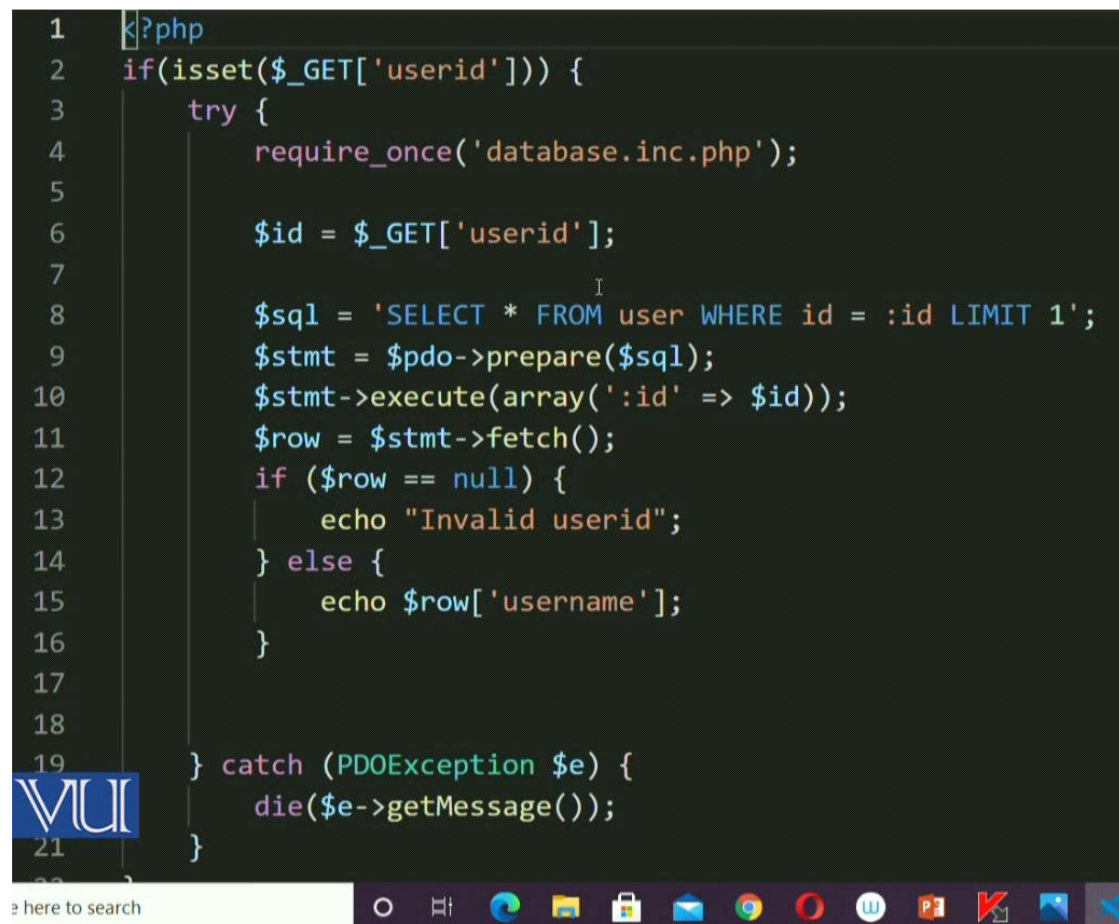
```

function(data, status){
$("#updates").html(data);
}
)
.done(function() {...} )
.fail(function() {...} )
.always(function() {...} )
;

```

Module 143

PHP with Ajax-Example

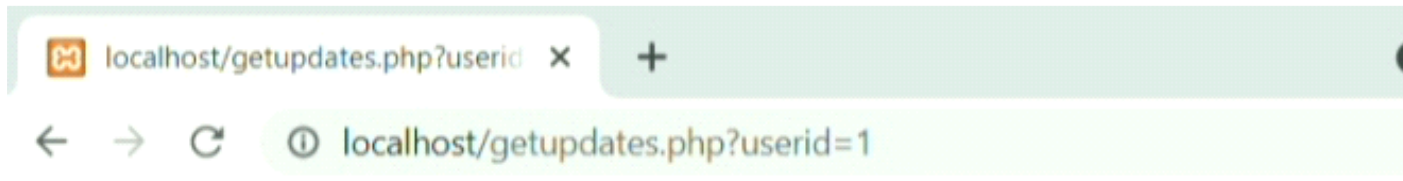


```

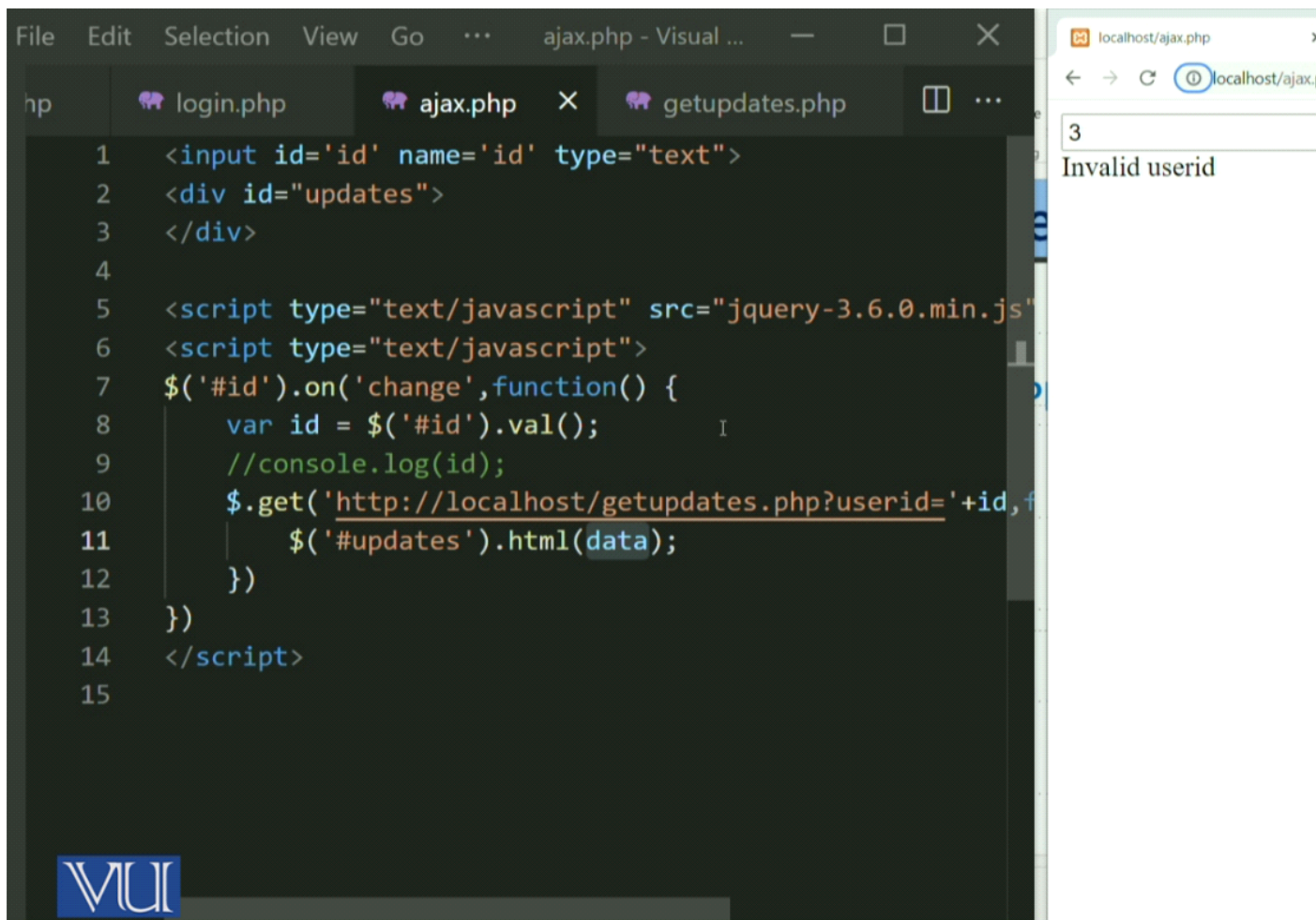
1  <?php
2  if(isset($_GET['userid'])) {
3      try {
4          require_once('database.inc.php');
5
6          $id = $_GET['userid'];
7
8          $sql = 'SELECT * FROM user WHERE id = :id LIMIT 1';
9          $stmt = $pdo->prepare($sql);
10         $stmt->execute(array(':id' => $id));
11         $row = $stmt->fetch();
12         if ($row == null) {
13             echo "Invalid userid";
14         } else {
15             echo $row['username'];
16         }
17
18
19     } catch (PDOException $e) {
20         die($e->getMessage());
21     }
22 }

```

Output:



bilal



Module 144

Web Application Design

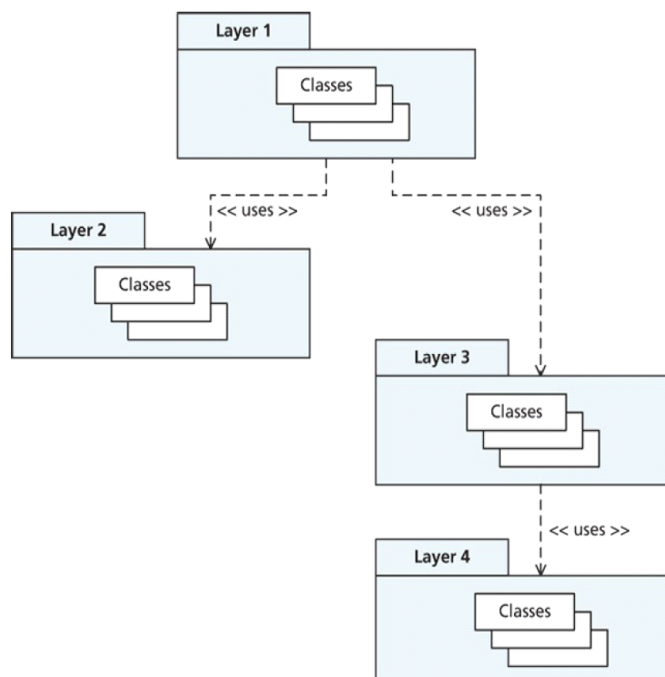
Challenges in Designing Web Applications

It is quite possible to create complex web applications with little to no class design. The page-oriented development approach is such that each page contains most of the programming code it needs to perform its operations. Rapidly thought-out systems are rarely able to handle unforeseen changes in an elegant way. A well-designed application infrastructure up front can make your web application easier to modify and maintain, easier to grow and expand in functionality, less prone to bugs, and thus, ultimately, in the long run easier to create.

What Is a Layer?

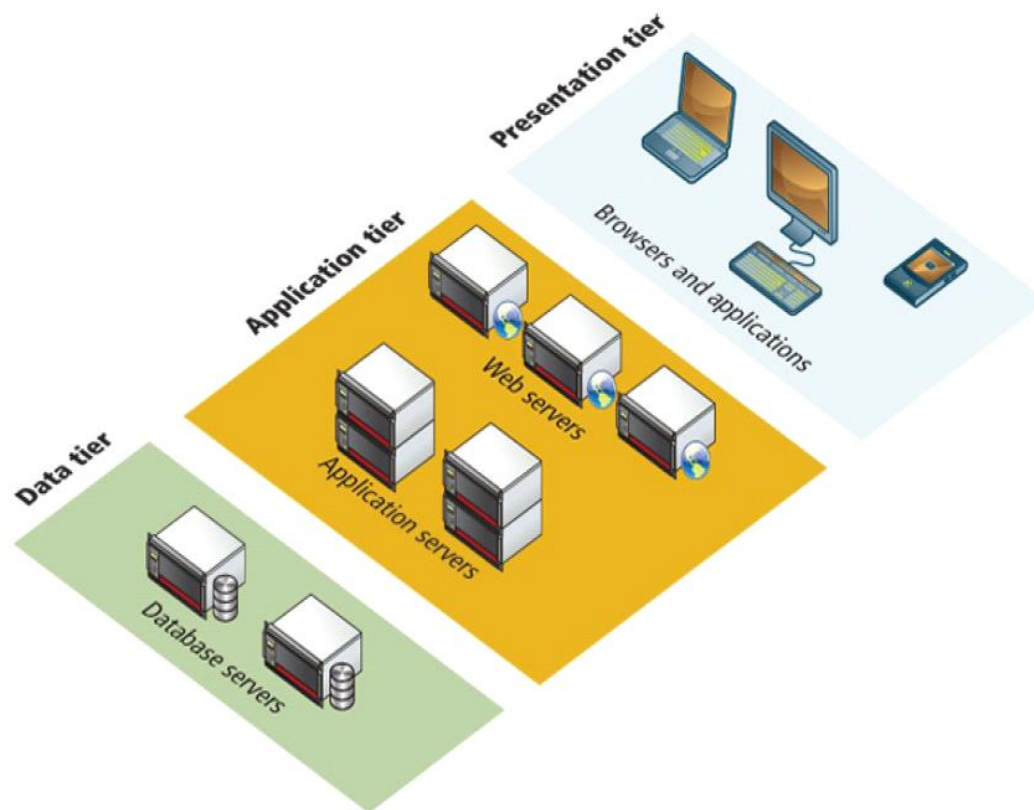
A layer in the context of application development, is simply a group of classes that are functionally or logically related. Each layer in an application should demonstrate cohesion. Distribute the functionality of your software among classes so that coupling is minimized. A dependency is a relationship between two elements where a change in one affects the other.

In the layered design approach, each class within the layer has a limited number of dependencies. A dependency (also referred to in UML as the uses relationship) is a relationship between two elements where a change in one affects the other. In the illustration given in below figure, the various layers have dependencies with classes only in layers “below” them, that is, with layers whose abstractions are more “lower level” or perhaps more dependent upon externalities such as databases or web services.



These different tiers most often refer to different places in a network. For example, a typical web application can be considered a three-tier architecture: the user's workstation is the presentation tier,

the web server is the application tier, and the DBMS running on a separate data server is the database tier, as shown in below Figure. The rest of the chapter will use the word tier in this latter sense, and use the word layer when referring to the conceptual grouping of classes within an application.



Common Layering Schemes

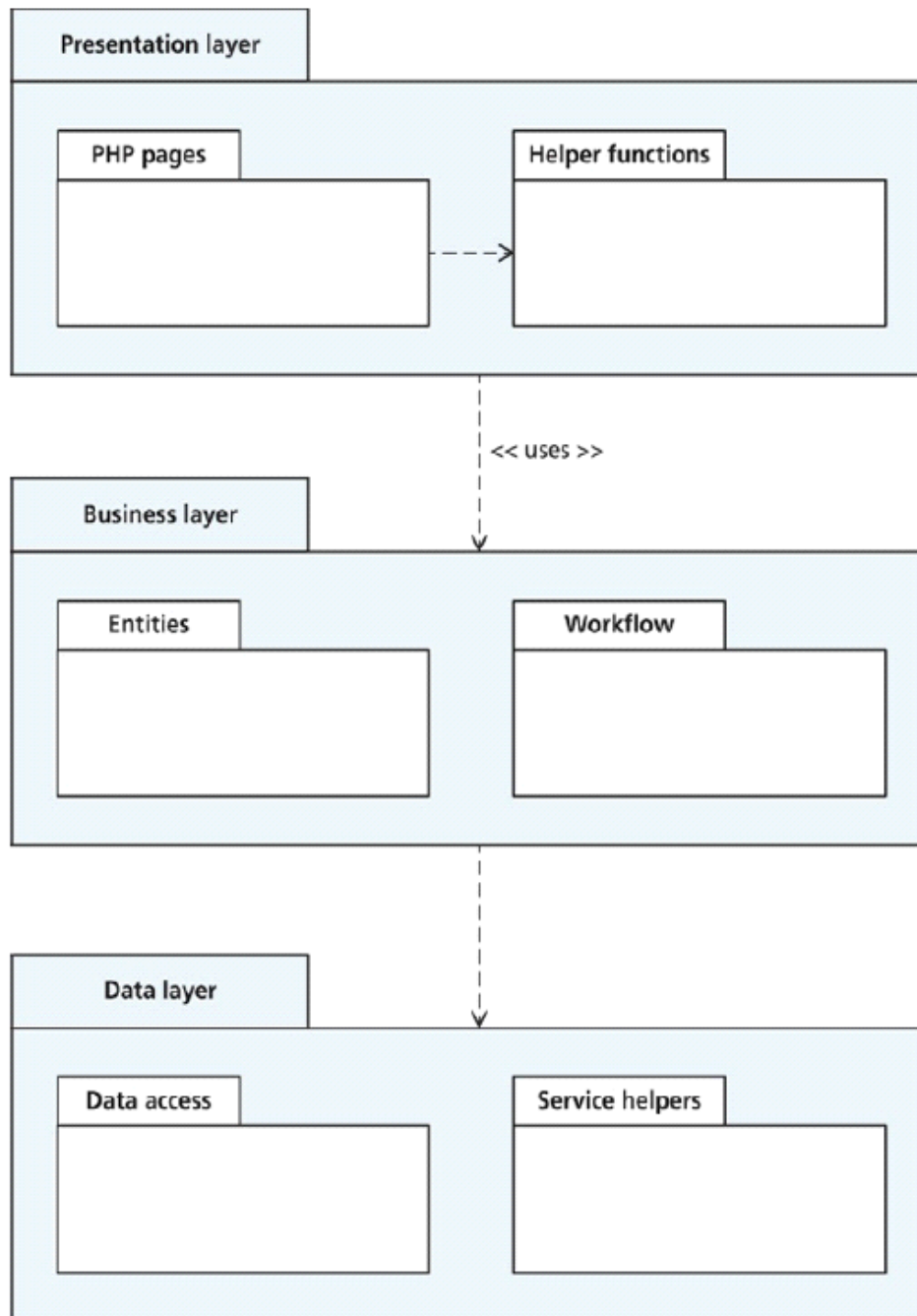
Presentation: Principally concerned with the display of information to the user, as well as interacting with the user.

Domain/Business: The main logic of the application. Some developers call this the business layer since it is modeling the rules and processes of the business for which the application is being written.

Data Access: Communicates with the data sources used by the application. Often a database, but could be web services, text files, or email systems.

Module 145

The two-layer paradigm has various flaws, thus many developers prefer a three-layer model in which a business layer (also known as a domain layer) is in charge of executing business rules and processes. The high-level design of a three-layer model is shown in the diagram below.

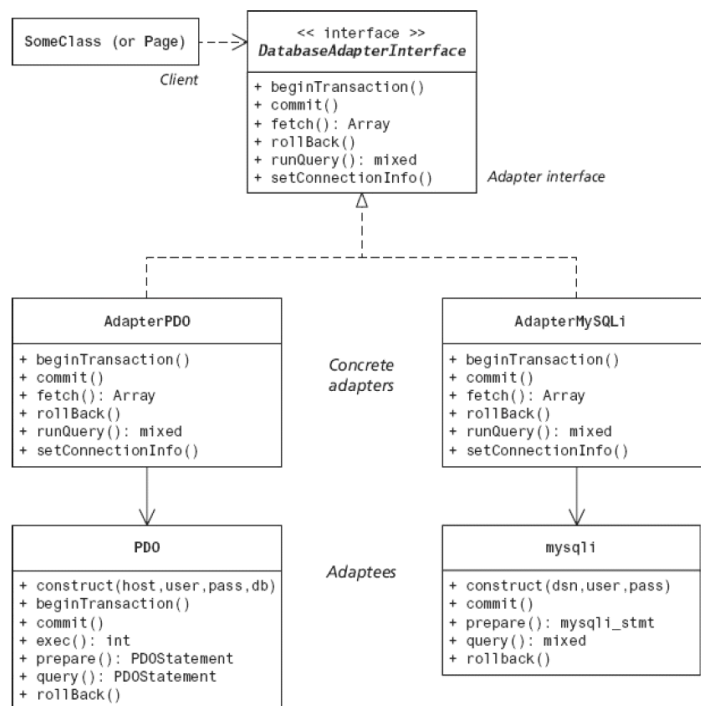


Some authors refer to the classes in a three-layer model's "middle" layer as business objects, while others refer to them as entities or domain objects. Business objects, whatever they're called, represent the data and behaviour of things that correspond to the business's conceptual domain. A simple domain layer would consist of domain/business objects that are very similar to the database table.

Module 146

Adapter Pattern

The Adapter pattern is used to change a set of classes' interfaces to a different but preferable interface. The key advantage of this pattern is that it separates the client (in the context of discussing patterns, the term client refers to the classes that use the pattern classes) from the consuming class's interface. The Adapter pattern is commonly used in web applications to leverage a database API (such as PDO or mysqli) without having to keep connecting the pages to that database API. Real-world websites, as discussed previously in the chapter, update the database or the API used to access it as the site increases in complexity or the size of its data or requests. The majority of the programme is protected against future change by using an Adapter. Indeed, writing (or reusing) a database API adaptor is one of the first things several designers do when starting a new web application project. The design of a sample database adapter is shown in the diagram below.



So, how would this adapter's code look like? The Adapter pattern, as shown in the diagram above, must first create an interface. We want the adapter to represent the capabilities that any database adapter will require in this case. Not only can you open and delete connections, but you can also conduct SELECT, UPDATE, INSERT, and DELETE queries and manage transactions.

Simple Factory Pattern

The Adapter pattern was used in the previous section to eliminate a dependent on a changing interface. Unfortunately, during the instantiation of the concrete adapter, a sort of dependency was introduced into the client code. The Simple Factory pattern is used as a solution to this problem. A factory is a custom class that creates subclasses (or concrete implementations of an interface) so that

clients aren't tied to specific subclasses or implementations. Factory patterns come in a variety of shapes and sizes. The Factory Method and the Abstract Factory are two patterns identified in the Design Patterns book. The Simple Factory pattern is a simplified version of the other two factories, as its name implies. A Factory Method with early binding can be developed in programming languages like C# or Java using conditional logic similar to the following pseudo-code:

```
If requested == 'PDO' Then
```

```
Return new PDOAdapter()
```

```
Else If requested == 'oracle' Then
```

```
Return new OracleAdapter()
```

```
Else If requested 'odbc' then
```

```
Return new OdbcAdapter()
```

```
etc.
```

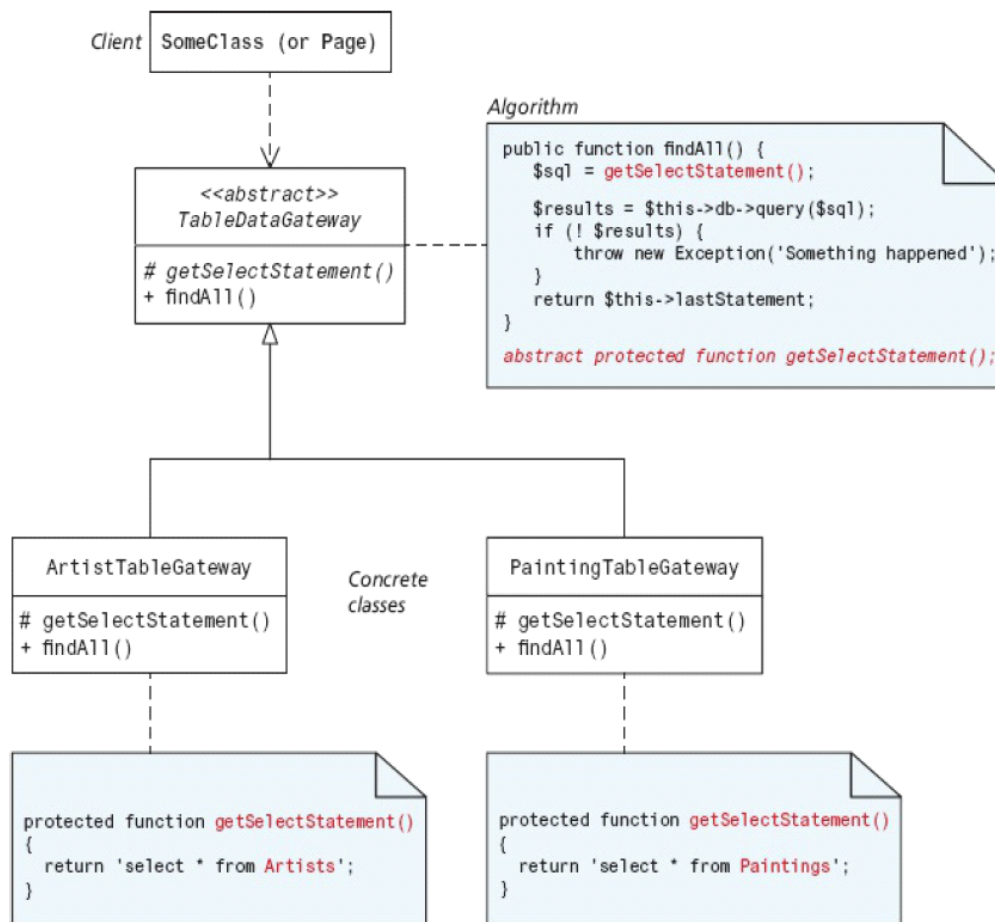
Because PHP is a late-binding language, you can design a factory class that avoids conditional logic by supplying the specific class name to instantiate dynamically at run time.

Module 147

Template Method Pattern

One of the most important of the 23 traditional design patterns is the Template Method. Many object-oriented developers, in fact, utilise this technique without even realising it. This technique involves

defining an algorithm in an abstract superclass and deferring to the subclasses the algorithm stages that can vary. The following Figure, for example, depicts the design of a sample data access layer that employs the Template Method pattern.



Dependency Injection

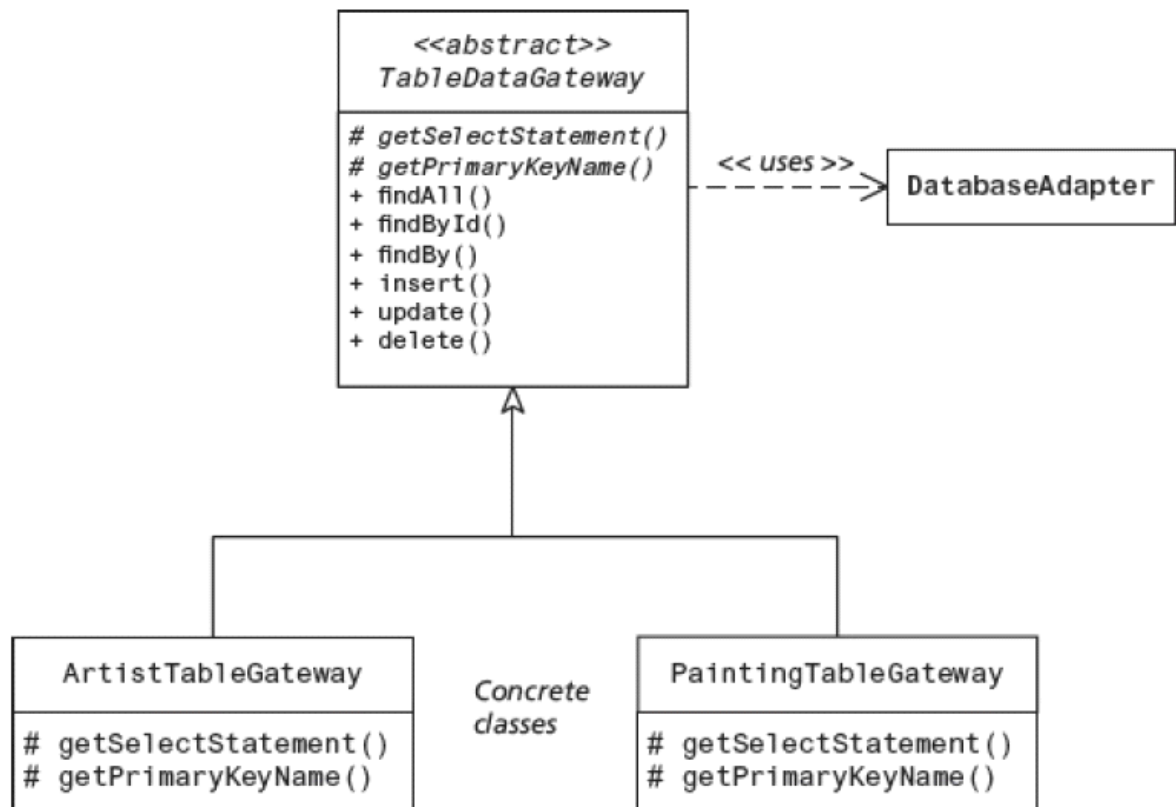
Although Dependency Injection is not one of the original 23 design patterns identified in the Design Patterns book, it has become one of the most essential software design patterns (and thankfully one of the simplest). It was first identified and named by Martin Fowler⁵; its purpose is to reduce the number of dependencies within a class, by passing (injecting) potential dependencies into a class rather than hard-coding them into the class.

Module 148

Table Data Gateway Pattern

The Table Data Gateway pattern by Fowler is similar to a data access object. A gateway is essentially an object that encapsulates external resource access. As a result, a table data gateway allows CRUD

access to a database table (or perhaps joined tables). The diagram below shows how this approach can be used to build the foundations of a data access layer. Notice how the superclass contains the majority of the common code (and uses the Template Method pattern), while each subclass defines the code specific to that table.



Domain Model Pattern

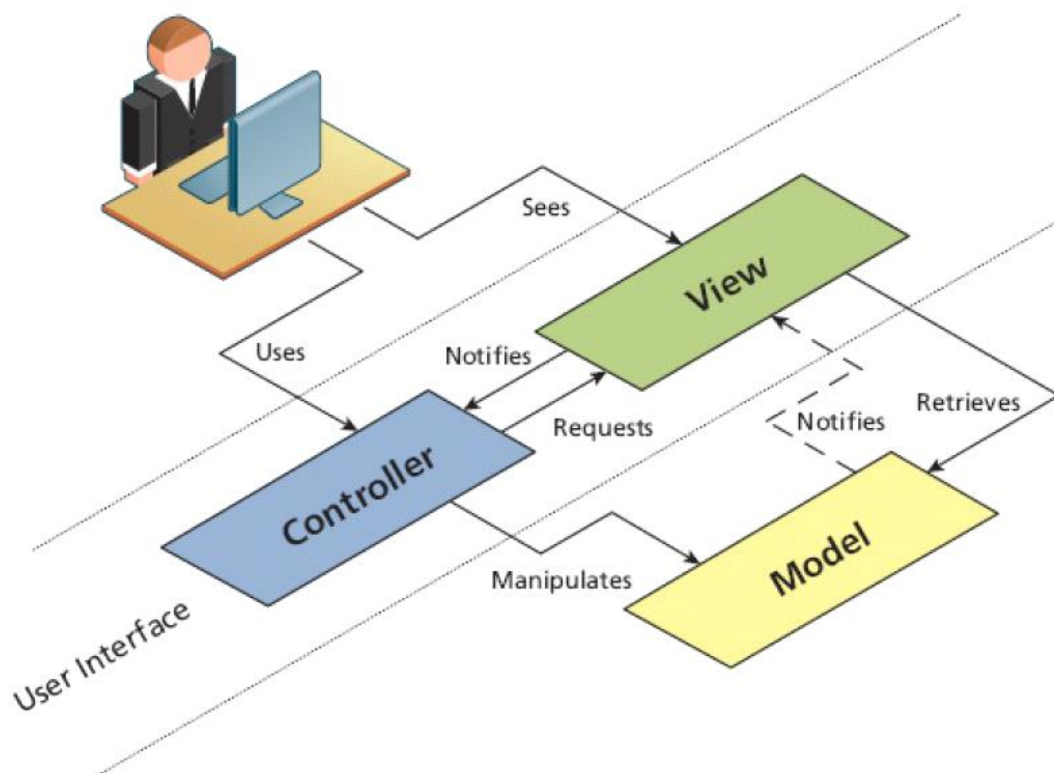
The Domain Model pattern is a natural fit for programmers who are experienced with object-oriented design. The developer creates an object model, which is a collection of connected classes that represent items in the application's issue domain. A domain model's classes will contain both data and behaviour, making them the ideal place to apply business rules.

Module 149

Model-View-Controller (MVC) Pattern

The Model-View-Controller (MVC) pattern predates the entire pattern movement, as it was first developed as a user-interface framework for the SmallTalk (early object-oriented language) platform in the 1970s. It influenced the concept and design of many future user interface frameworks

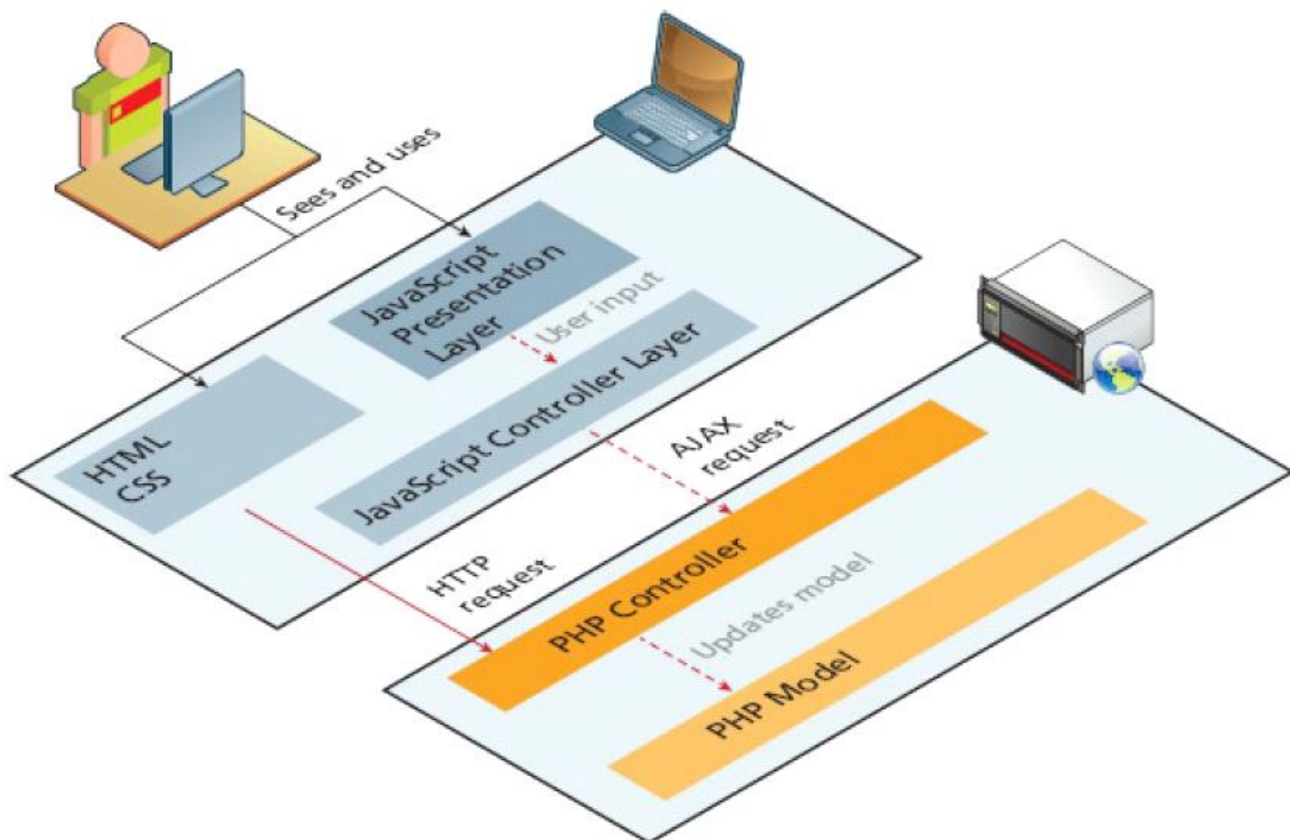
significantly. The MVC design comes in a variety of subtle (and not so subtle) versions, including several for PHP and JavaScript. The MVC design separates an application into three classes that serve three functions: model, view, and controller. The Model-View-Controller (MVC) pattern was created in the 1970s as a user-interface framework for the SmallTalk (early object-oriented language) platform. It has a considerable impact on the concept and design of many future user interface frameworks. The MVC design is available in a variety of subtle (and not so subtle) variations, including PHP and JavaScript implementations. The model, view, and controller (MVC) design divides an application into three classes that serve three functions.



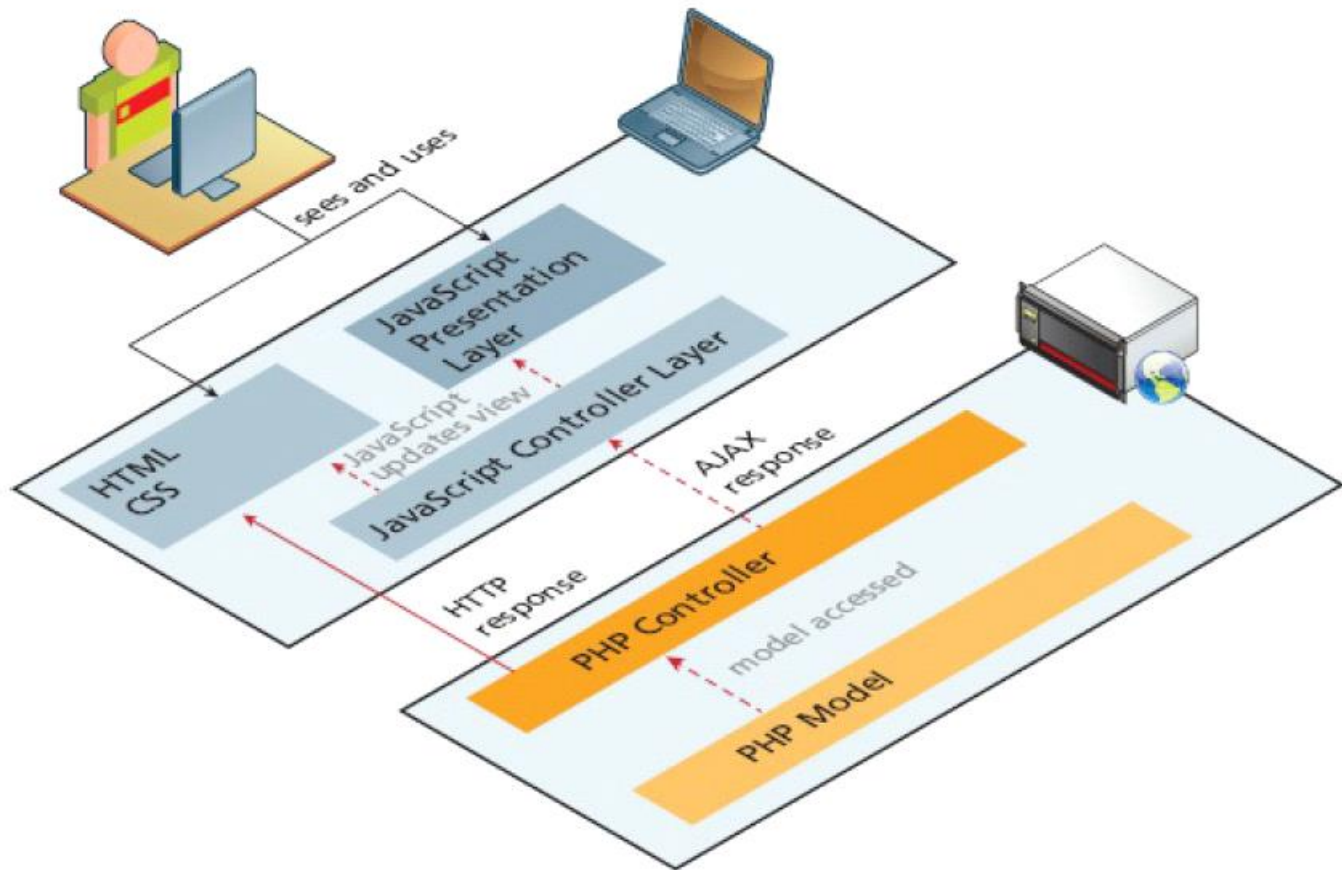
Module 150

It should be remembered that the MVC design was created for desktop applications in which the views could utilise the Observer pattern (or something similar like event listeners) to update themselves anytime the model changed. When the MVC pattern is used in a web environment, things

become more challenging. The model in MVC is relatively straightforward: it's very similar to the domain model covered in the previous section (though it could also be just the gateway classes). However, some features of the paradigm may also be implemented in JavaScript on modern AJAX-based websites. The more difficult question is: what is the relationship between the View and the Controller? The View includes not only HTML and CSS, but also the PHP that creates it and presentation-oriented JavaScript. As seen in Figure , the Controller is most likely partially developed in JavaScript and partially in PHP.



Other distinctions exist between a web MVC and a desktop MVC. Because the model is primarily (or totally) on a different computer than the views, there is no mechanism for the views to listen for changes in the model as in the standard MVC approach. Another distinction is that with desktop applications, the model is a collection of objects that remain populated during the application's lifespan. These objects appear in a PHP application just when the script is running and disappear once the request is handled. When developing a web MVC application, one of the most important design decisions is whether the controller will be a server-side PHP controller or a client-side JavaScript controller. It is possible for the controller to be both as illustrated in following Figure.



Module 151

Laravel Introduction:

Laravel is a php based framework. It is quite a complex framework that provides a lot of features and tools. You can say that it has its own ecosystem. Here we will cover some basics of Laravel. Here are some features of Laravel:

It is a Php based framework.

It is Open source

It follows MVC architecture

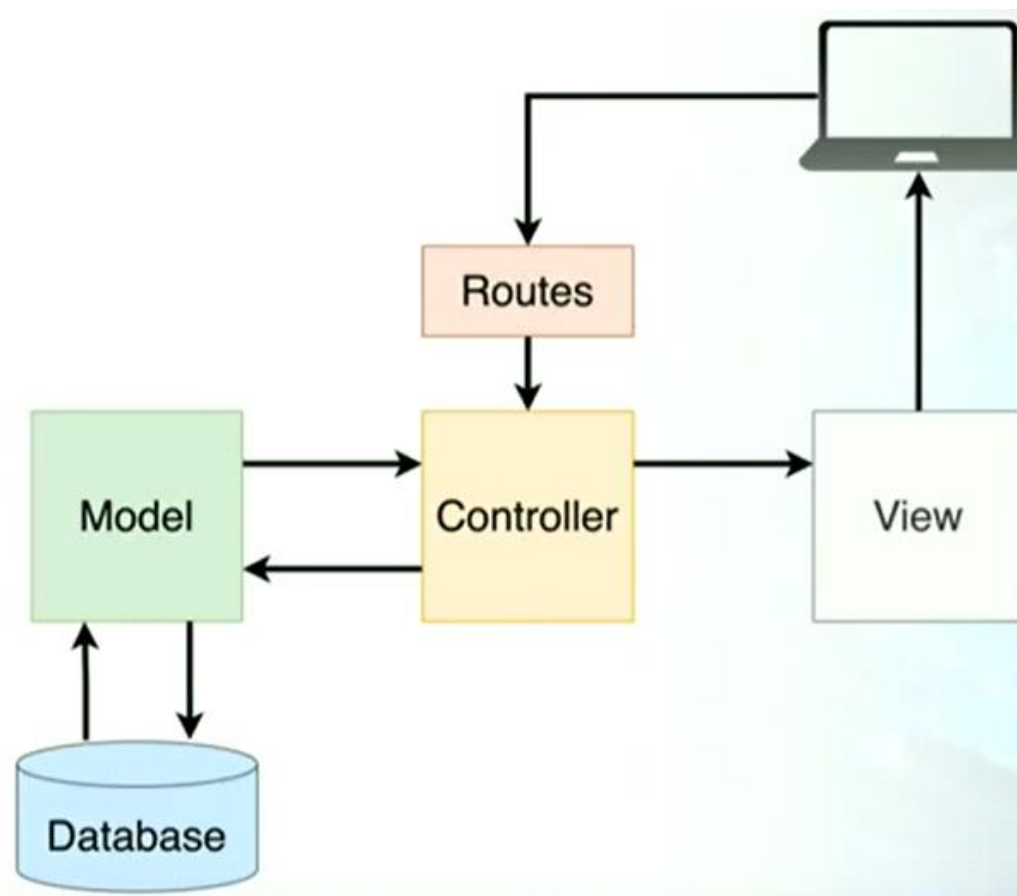
It uses template engine for layouts/views

It also provides command line utility (artisan) to do plenty of tasks.

You can easily integrate it with database

It uses composer as dependency and package manager

Figure below shows the Laravel architecture



User will send a URL request e.g. user can ask about “a list of products”. We have a “Route” class in Laravel framework that will manage this request and will decide the controller that will be invoked on the basis of the user request. For example, if user asked for the list of the products, the “Product Controller” will be activated which will send a request to “Product model” to have a list of all the products. Model will then fetch the products from the database, make their list and will send back to controller. Controller will then pass the list to the view. The view is a template that defines how the

data will be displayed to the user. The data will then be sent to the user in HTML format which will be rendered in the browser

Module 152

Laravel Installation:

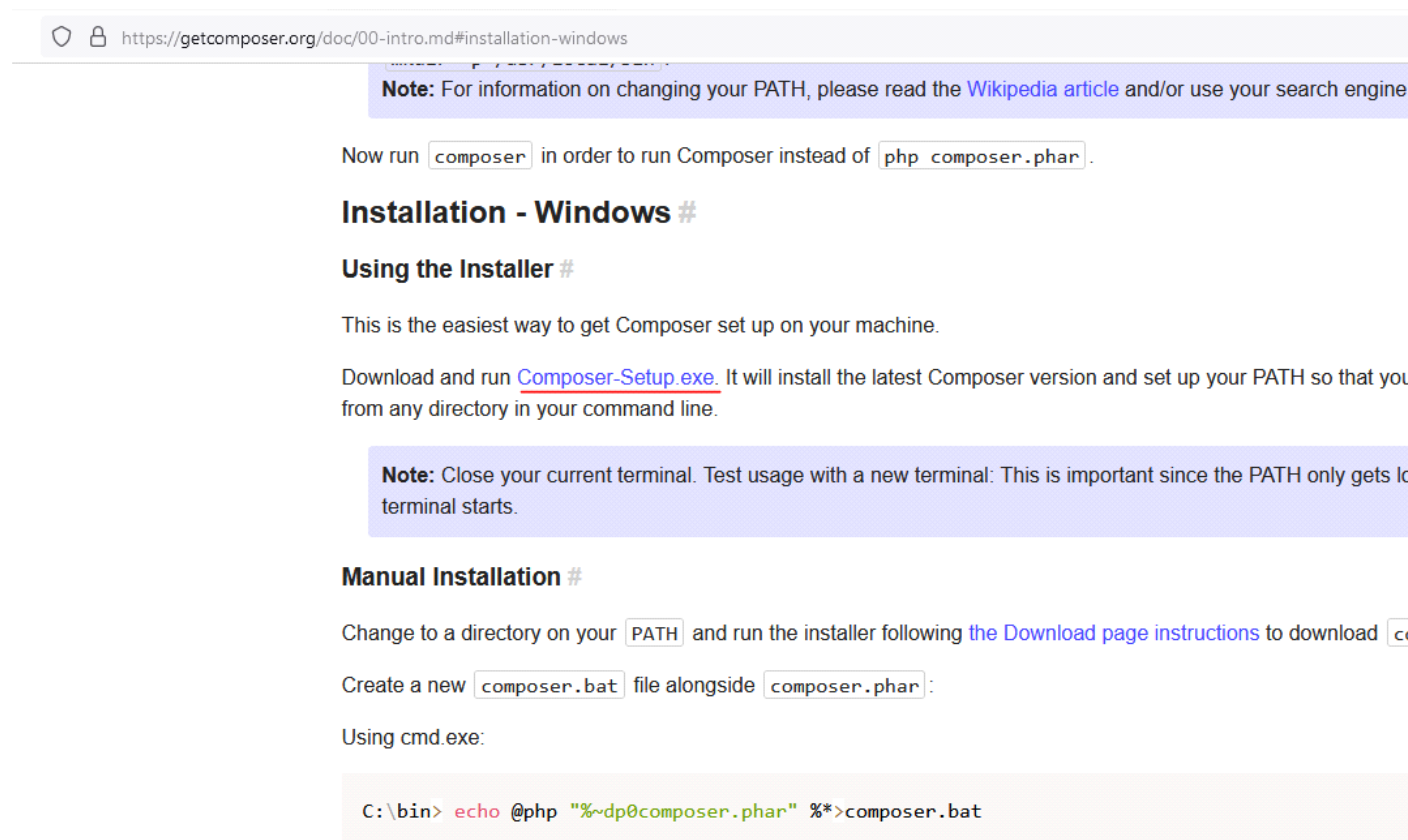
Normally Laravel can be downloaded in zip format. However, Laravel being a PHP framework, depends on some PHP scripts that may not be present in the zip file. For this purpose you will use a dependency manager called the “Composer”. When you will install Laravel through composer, then all the dependencies will also be installed on your machine.

For installation of composer do the following:

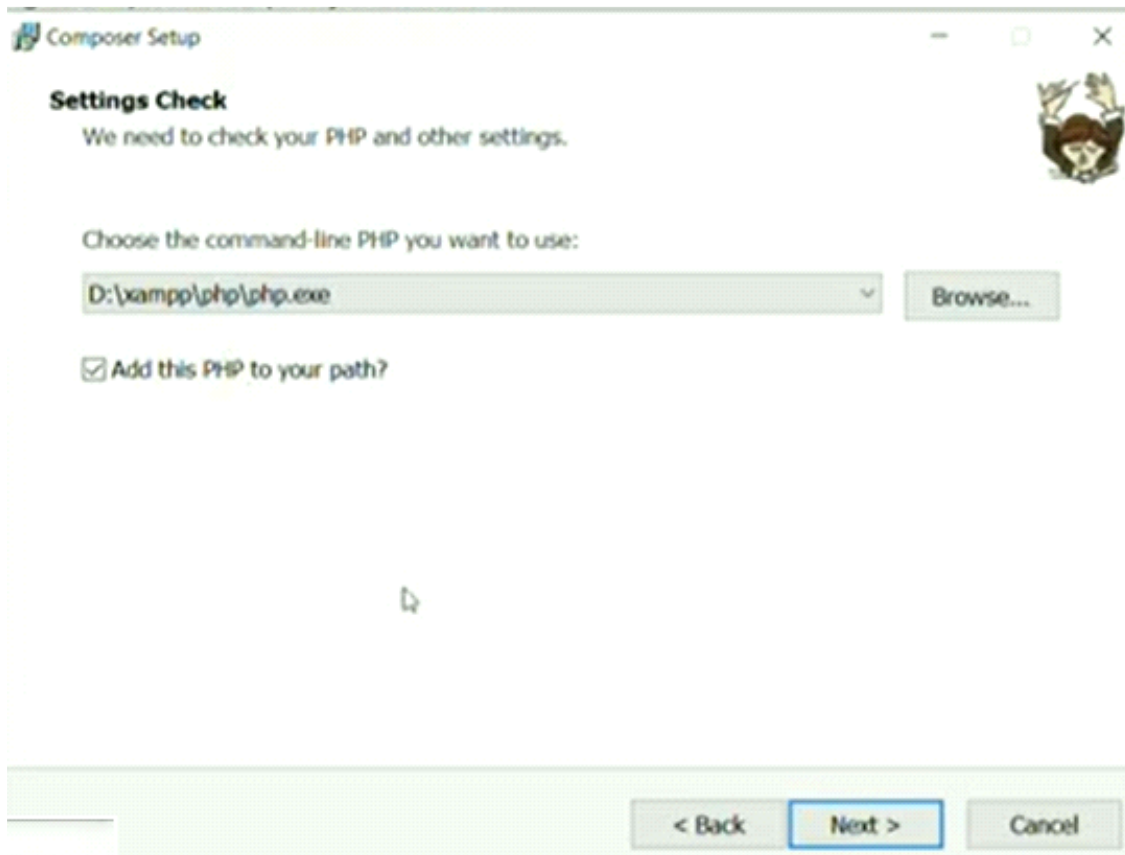
Open the link:

<https://getcomposer.org/doc/00-intro.md#installation-windows>

From the page (that is opened), click “Composer-Setup.exe” as shown in the figure below:



Composer-Setup.exe will be downloaded. Execute this file by accepting the default options. Note that, at an intermediate stage, you will have to enter the path of the PHP file as shown below:



Note: your path may be different.

Once the composer is installed, you can verify the installation by typing “composer” at the command prompt. If the installation was successful, you will see the composer related information on command prompt else you will get “Command not found” message.

Once the composer is installed, you will use the composer to install Laravel. For this purpose type the following command, that will install the Laravel and create a Laravel project named “example-app”.

```
C:\Windows\system32\cmd.exe
show           Shows information about packages.
status         Shows a list of locally modified packages.
suggests       Shows package suggestions.
u              Upgrades your dependencies to the latest version according to composer.
composer.lock file.
update         Upgrades your dependencies to the latest version according to composer.
composer.lock file.
upgrade        Upgrades your dependencies to the latest version according to composer.
composer.lock file.
validate       Validates a composer.json and composer.lock.
why            Shows which packages cause the given package to be installed.
why-not        Shows which packages prevent the given package from being installed.

C:\Users\NEW-B>d:
D:\>composer create-project laravel/laravel example-app
```

Once the project is created, you can navigate the installation folder to check different of the directories and files in the example-app project.

Now to run the Laravel, go to the command prompt, go to the project folder (in our case it was D:\example-app) and type the following command:

```
D:\example-app>php artisan serve
```

You will get URL of the website (your project). You can open the URL in the browser and see the default webpage.

Module 153

Laravel: Introduction to Routes

Whenever the user sends a request, it is received by the Route class. However, before discussing the Routes, let's discuss the Representational State Transfer (REST) architecture. REST is a way of implementing web services. REST is comprised of the objects which we call resources. We can perform different operations on these objects using the HTTP methods. Mainly the following four operations are performed:

POST (Create)

GET (Retrieve)

PUT (Update)

DELETE (Delete)

For example, the following URL will return the product having product id=1

<http://www.myshop.com?pid=1>

As you can see that we have sent the request using the get method (you can see the string query above), so the retrieve method will be called.

In Laravel, the Routes are handled through the Route class. Routes are defined in the folder named “routes/” in Laravel project.

You can check the “routes” folder in the previously installed “example-app” project. Open the main project folder and you will see the “routes” folder. The folder contains many files. The main file that we will be using is “web.php” file. The file contains the routes that are used when the project is accessed through the web.

Now open the “web.php” file and you will see the following code:

```
Route::get('/',function() {  
    return view('welcome');  
});
```

Enter the following URI in browser:

227.0.0.1:8000

(Note: we assume that this is the URI of the project)

You will see the welcome view in the browser. You can verify the view in the “views” folder.

‘/’ in the above code refers to the main (root) folder of the project.

Now let’s define a new Route named “about”. You will define it as follows:

```
Route::get('/about',function() {  
    return 'About Us!';  
});
```

Now enter the URI:

227.0.0.1:8000/about

You will see the “About Us!” message in the browser.

You can create the navigation between “home” and the “about” pages as follows:

```
Route::get('/',function() {  
    return 'Home<br><a href="'.url('/about').'"> About Us</a>';  
});  
  
Route::get('/about',function() {  
    return 'About Us<br><a href="'.url('/').'"> Home</a>';  
});
```

Module 154

Laravel – Routes with Variables

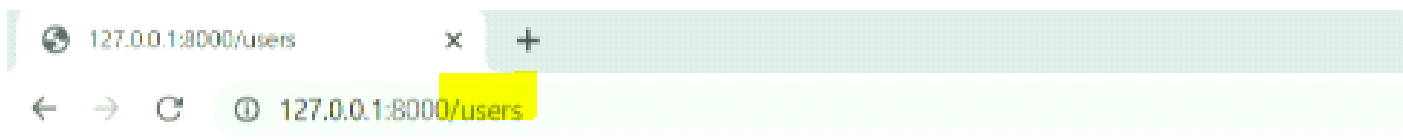
We create routes using URLs. If we want to handle variable using routes.

Let's define a route for Users:

Creating a simple Route “/users”

```
}):  
    return 'List of users<br><a href="'.url('/').">Home</a>';  
Route::get('/users', function () {
```

Output:



List of users
Home

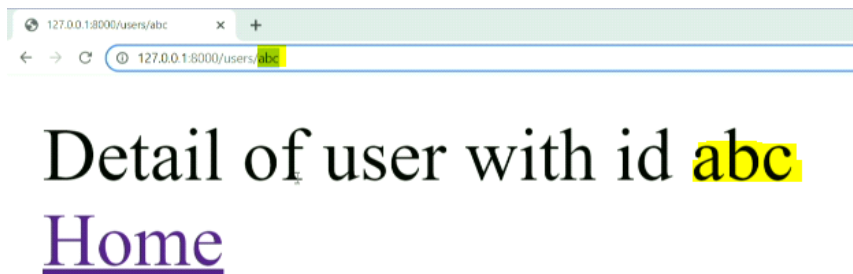
Routes with Variable:

```
Route::get('/users/{id}', function ($id) {  
    return 'Detail of user with id '.$id.'<br><a href="'.url('/'  
});
```

Output:



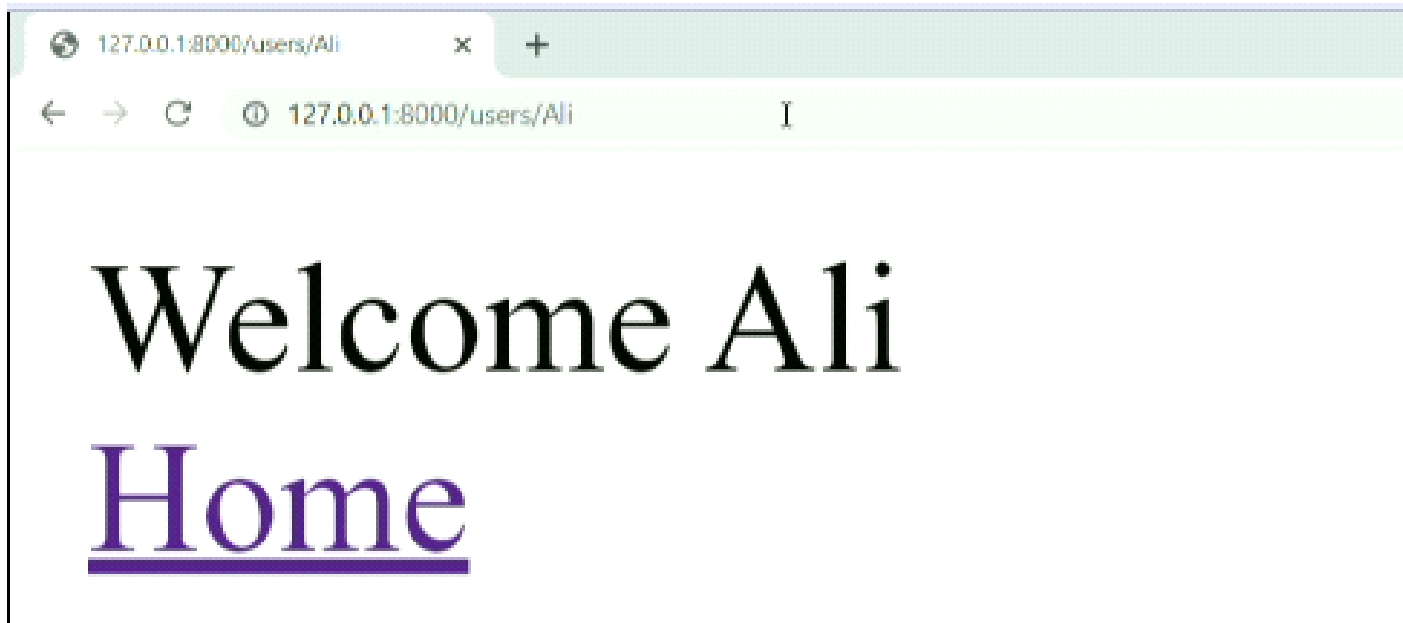
This variable will be passed, it doesn't matter it's a number or a string. But we can also define a pattern for this variable.



Define different patterns for the variable:

```
Route::get('/users/{id}', function ($id) {  
    return 'Details of user with id '.$id.'<br><a href="'.url('/').'">Home</a>';  
})->where('id', '[0-9]+');  
  
Route::get('/users/{name}', function ($name) {  
    return 'Welcome '.$name.'<br><a href="'.url('/').'">Home</a>';  
})->where('id', '[a-zA-Z ]+');
```

If we pass a string in the URL:



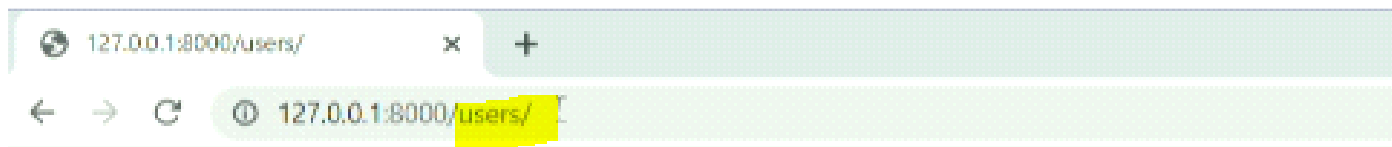
If we write a number in the URL:



Optional Variable:

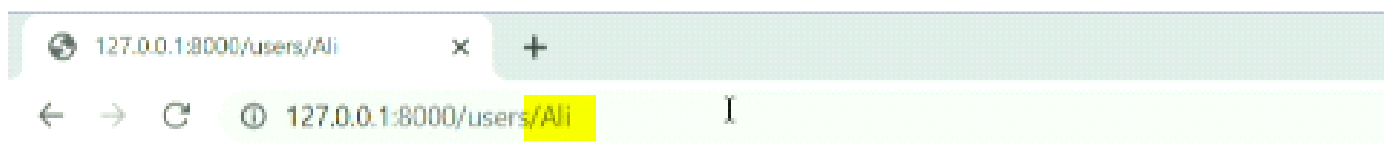
```
Route::get('/users/{rname?}', function ($name=null) {  
    if ($name == null)  
        $name = 'Guest';  
    return 'Welcome ' . $name . '<br><a href="'.url('/').'  
})->where('id', '[a-zA-Z ]+');
```

If we will no pass any string so, “Guest” will be passed:



Welcome Guest

Home



Welcome Ali

Home

Receive Multiple Variables:

```
Route::get('/users/{id}/{favid}', function ($id,$favid) {  
    return 'Details of user with id '.$id.' and favoriate id <br><a href="'.  
})->where('id','[0-9]+');
```

Module 155

Laravel – Views:

Views are located in resources/view

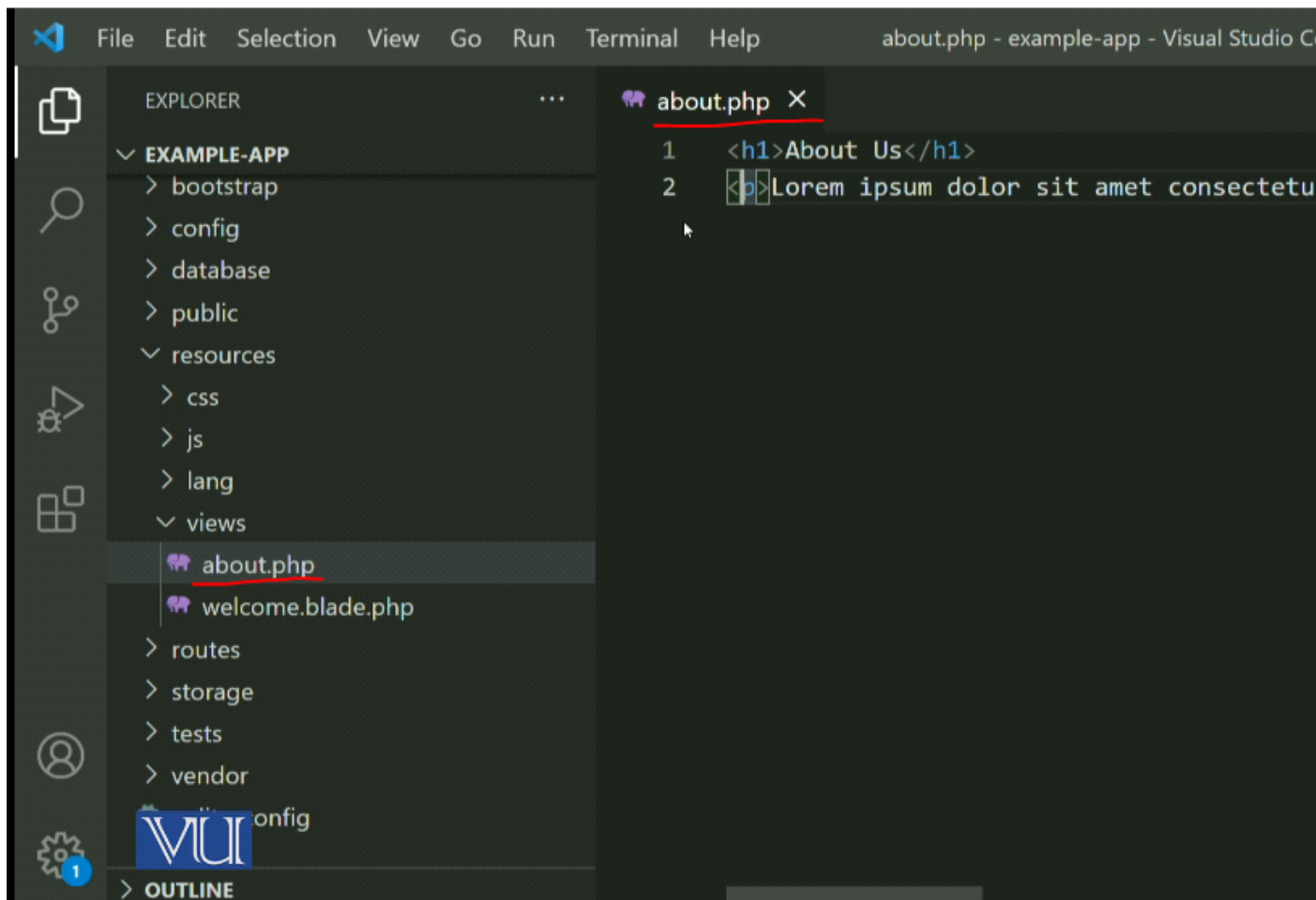
Types:

Plain PHP

Blade Template

To directly access

Create a file “about.php” under Views folder.



Open Web.php and write the following code:

```
Route::get('/about', function () {  
    return view('about');  
});
```

Output:



About Us

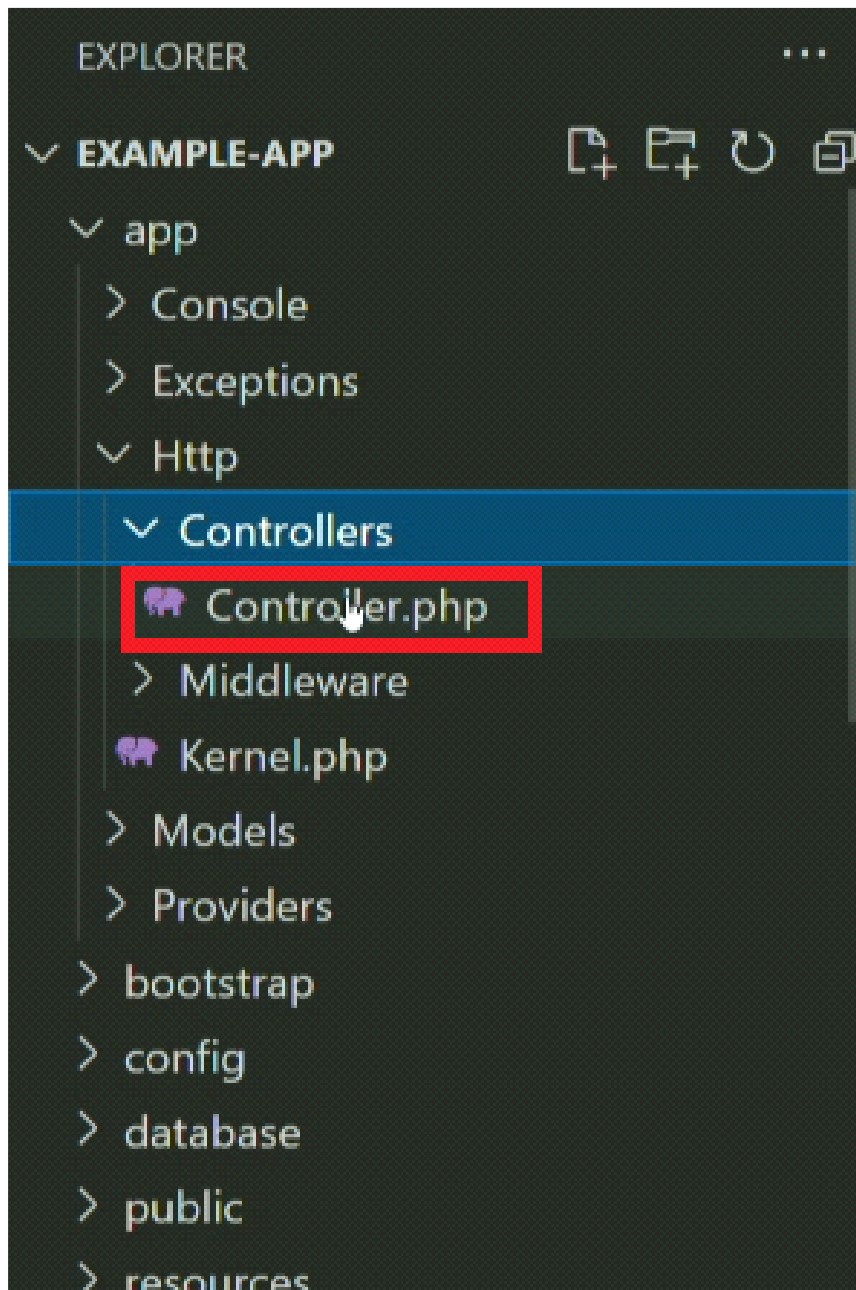
Lorem ipsum dolor sit amet consectetur adipisicing elit. Deleniti distinctio minima, numquam natus possimus asperiores impedit! Vitae, dolores corporis a, voluptatum mollitia molestiae atque dolorem accusamus quos nihil at enim!

You can also use View Function:

```
Route::view('/about', 'about');
```

Module 156

Laravel – Controller



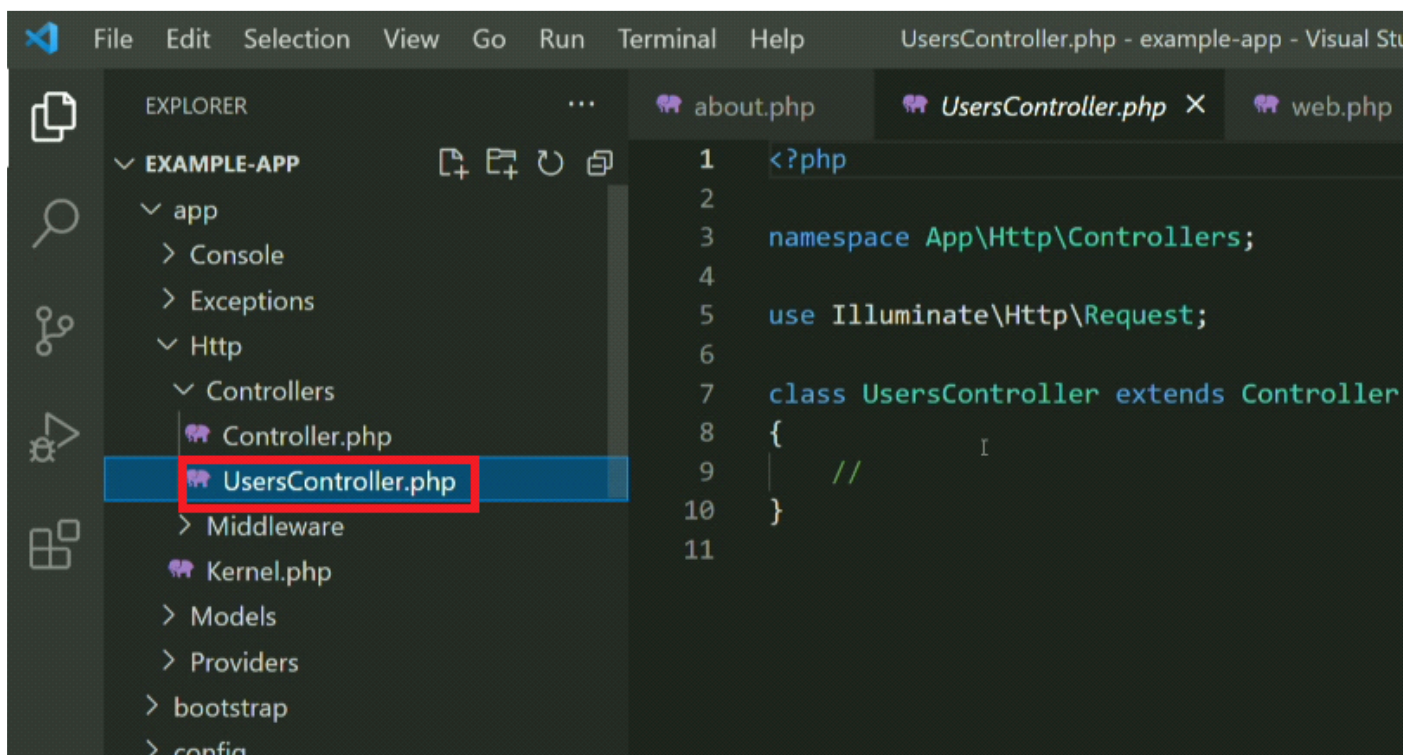
Open cmd.exe. The following command will create a controller named “UserController”.

```
C:\Windows\system32\cmd.exe

D:\example-app>php artisan make:controller UsersController
Controller created successfully.

D:\example-app>
```

A new controller file is created.



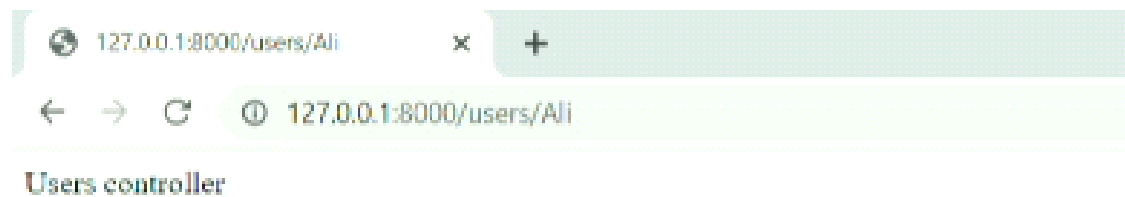
UsersController.php

```
class UsersController extends Controller
{
    //
    public function index() {
        return 'Users controller';
    }
}
```

Controller in Web.php

```
Route::get('/users/{id}/{favid}', function ($id,$favid) {  
    return 'Details of user with id '.$id.' and favoriate id '.$favid.'  
    }->where('id','[0-9]+');  
  
Route::get('/users/{name?}', 'App\Http\Controllers\UsersController@index')->where('id',
```

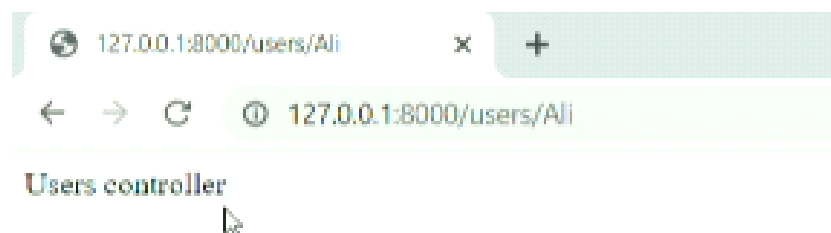
Output:



You can also invoke the controller using the following code:

```
use App\Http\Controllers\UsersController;  
Route::get('/users/{name?}',[UsersController::class,'index'])->where('id','[a-zA-Z ]+
```

Output:



Module 157

Laravel – Controller with C\$RUD Operations

Some operations are redundant, for example, Create, Read, Update and Delete. Let's see how to make a controller for these operators.

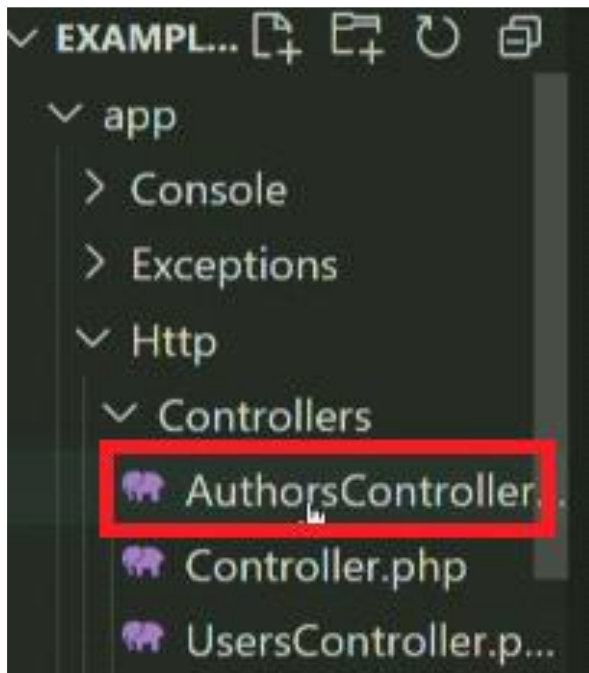
Open cmd.exe and write the following Command for the CRUD operations controller.

```
C:\Windows\system32\cmd.exe

D:\example-app>php artisan make:controller AuthorsController --resource
Controller created successfully.

D:\example-app>
```

A new controller will be created under the Controller folder, having all the functions created for Read, Write, Update and Delete.



If we want to check, how many Controllers are created. We can use the following command. Its shows how many routes we defined and how many of them are attached with the Controller.

```
D:\example-app>php artisan make:controller AuthorsController --resource
Controller created successfully.
```

```
D:\example-app>php artisan route:list
```

Domain	Method	URI	Name	Action	Middleware
	GET HEAD	/		Closure	web
	GET HEAD	about		Closure	web
	GET HEAD	api/user		Closure	api auth:api
	GET HEAD	users/{id}		Closure	web
	GET HEAD	users/{id}/{favid}		Closure	web
	GET HEAD	users/{name?}		App\Http\Controllers\UsersController@index	web

```
D:\example-app>
```

Module 158

Laravel – Redirects

Sometimes, we need to redirect a user. We can direct a user using the following code. There are different methods to redirect in Laravel. The following code will redirect “redirect/helper to the Hokme page.

```
Route::get('redirect/helper', function() {  
    return redirect()->to('/');  
});
```

Another way to redirect the web page:

```
Route::get('redirect/shortcut', function() {  
    return redirect('/');  
});
```

Another way to redirect the web page:

```
Route::get('redirect/facade', function() {  
    return Redirect::to('/');  
});
```

```
Route::redirect('redirect/route', '/');
```

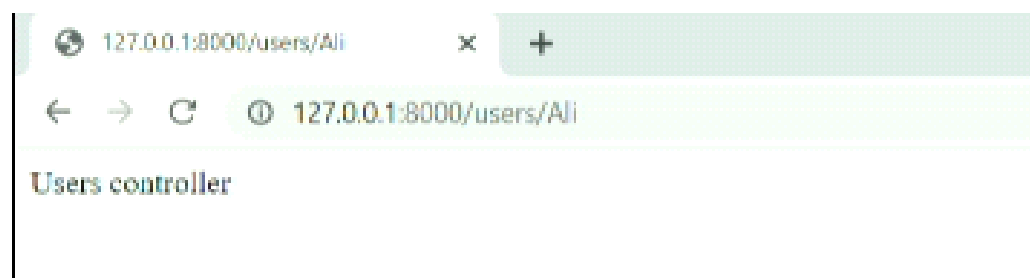
Redirect with a parameter:

```
Route::get('redirect/params', function() {  
    return redirect()->route('users.show', ['name' => 'Ali']);  
});
```

We haven't created any view for the user. So, we are going to create a view for the user at first.

```
use App\Http\Controllers\UsersController;  
Route::get('/users/{name?}', [UserController::class, 'show'])  
->where('id', '[a-zA-Z ]+')  
->name('users.show');
```

Here name “Ali” will be passed as a parameter.

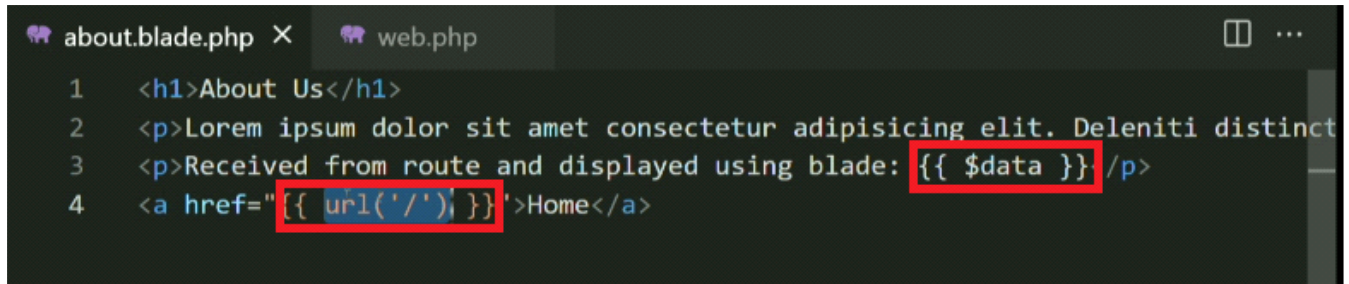


Module 159

Laravel – Blade Templates

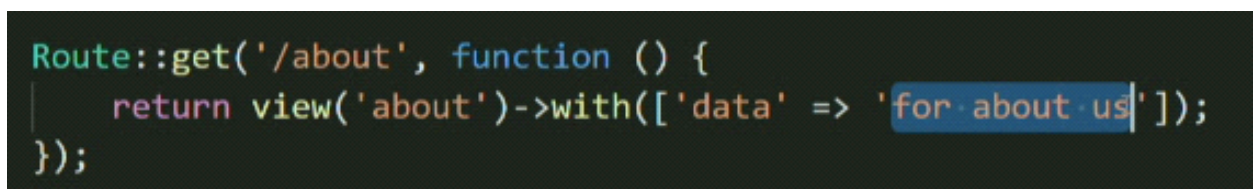
We will use our previous file “about.php”. Just rename the file to “about.blade.php”.

Edit the file and whatever needs to be dynamic, will be used in Blade Template using `{{variable_name}}`



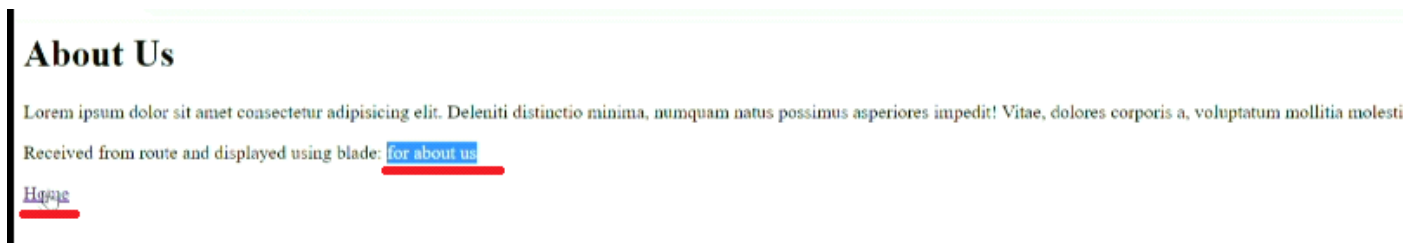
```
1 <h1>About Us</h1>
2 <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Deleniti distinct
3 <p>Received from route and displayed using blade: {{ $data }} /p>
4 <a href="{{ url('/') }}">Home</a>
```

Route in web.php:

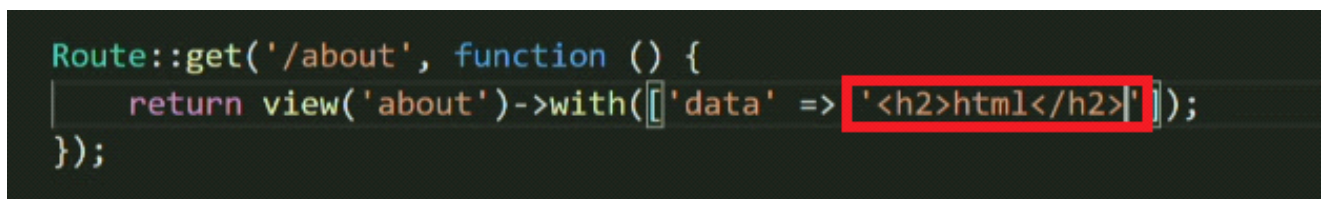


```
Route::get('/about', function () {
    return view('about')->with(['data' => 'for about us']);
});
```

Output: “for about us” used as data.



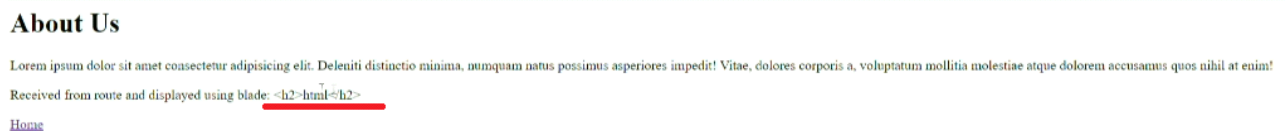
Let’s pass an HTML tag as a parameter and see the output:



```
Route::get('/about', function () {
    return view('about')->with(['data' => '<h2>html</h2>']);
});
```

Output:

Here `<h2>` is processed as a html tag.



We can also show the data as it is, on the webpage using the {!! and the data will be shown here as h2.

About Us

Lorem ipsum dolor sit amet consectetur adipisicing elit. Deleniti distinctio minima, numquam natus possimus asperiores impedit! Vitae, dolores corporis a, voluptatum mollitia molestiae atque dolorem accusamus quos nihil at enim!

Received from route and displayed using blade:

html

[Home](#)

```
1 <h1>About Us</h1>
2 <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Deleniti distinct
3 <p>Received from route and displayed using blade: {!! $data !!}</p>
4 <a href="{{ url('/') }}">Home</a>
```

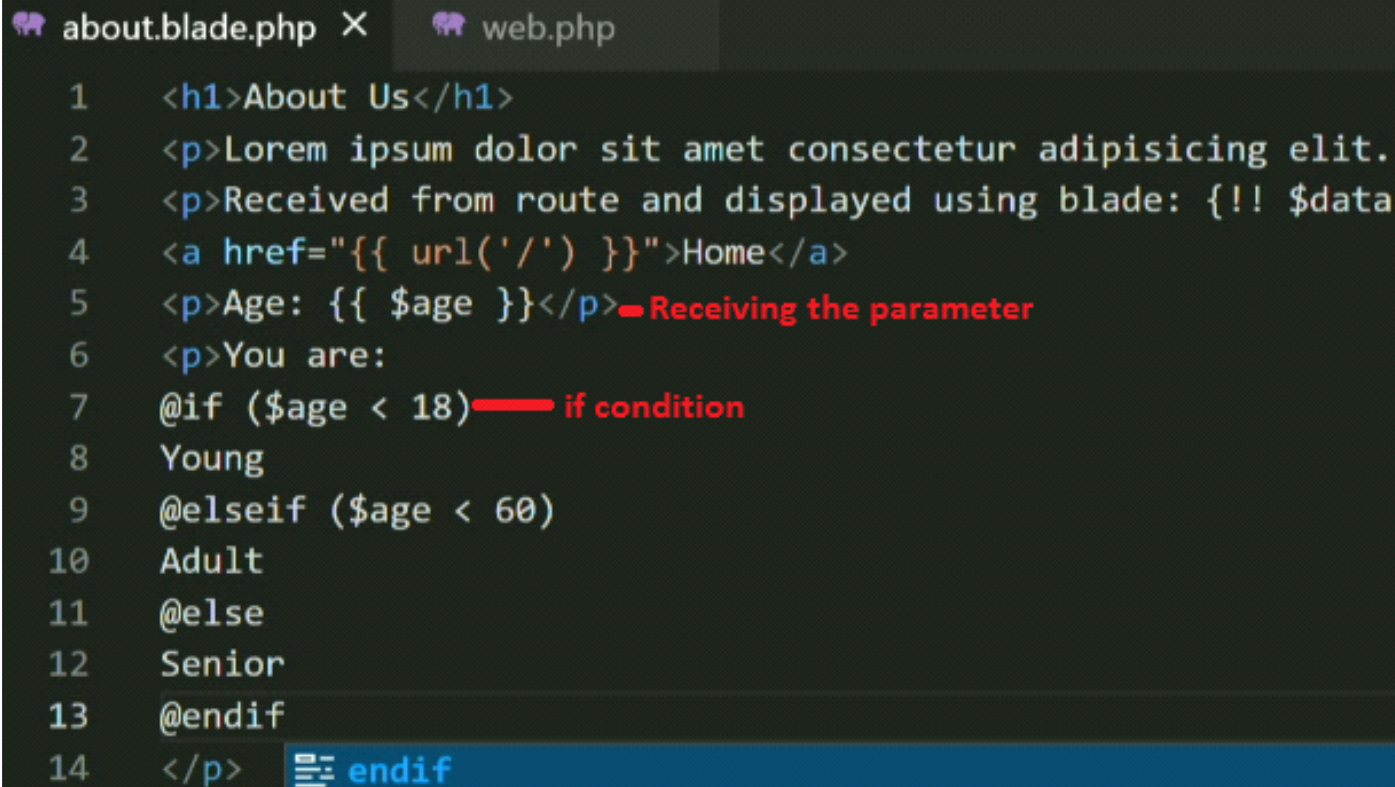
Module 160

Laravel – Conditions in Blade

We can use different conditions in Laravel Blade Template. Let's see how can we use them.

Passing a parameter age with value 21 in Web.php

```
Route::get('/about', function () {  
    return view('about')->with(['data' => '<h2>html</h2>']);  
});
```



```
about.blade.php X web.php  
1 <h1>About Us</h1>  
2 <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.  
3 <p>Received from route and displayed using blade: {!! $data  
4 <a href="{{ url('/') }}">Home</a>  
5 <p>Age: {{ $age }}</p> Receiving the parameter  
6 <p>You are:  
7 @if ($age < 18) if condition  
8 Young  
9 @elseif ($age < 60)  
10 Adult  
11 @else  
12 Senior  
13 @endif  
14 </p> endif
```

Output:



About Us

Lorem ipsum dolor sit amet consectetur adipisicing elit. Deleniti distinctio minima, numquam natus possi Vitae, dolores corporis a, voluptatum mollitia molestiae atque dolorem accusamus quos nihil at enim!

Received from route and displayed using blade:

html

[Home](#)

Age: 21

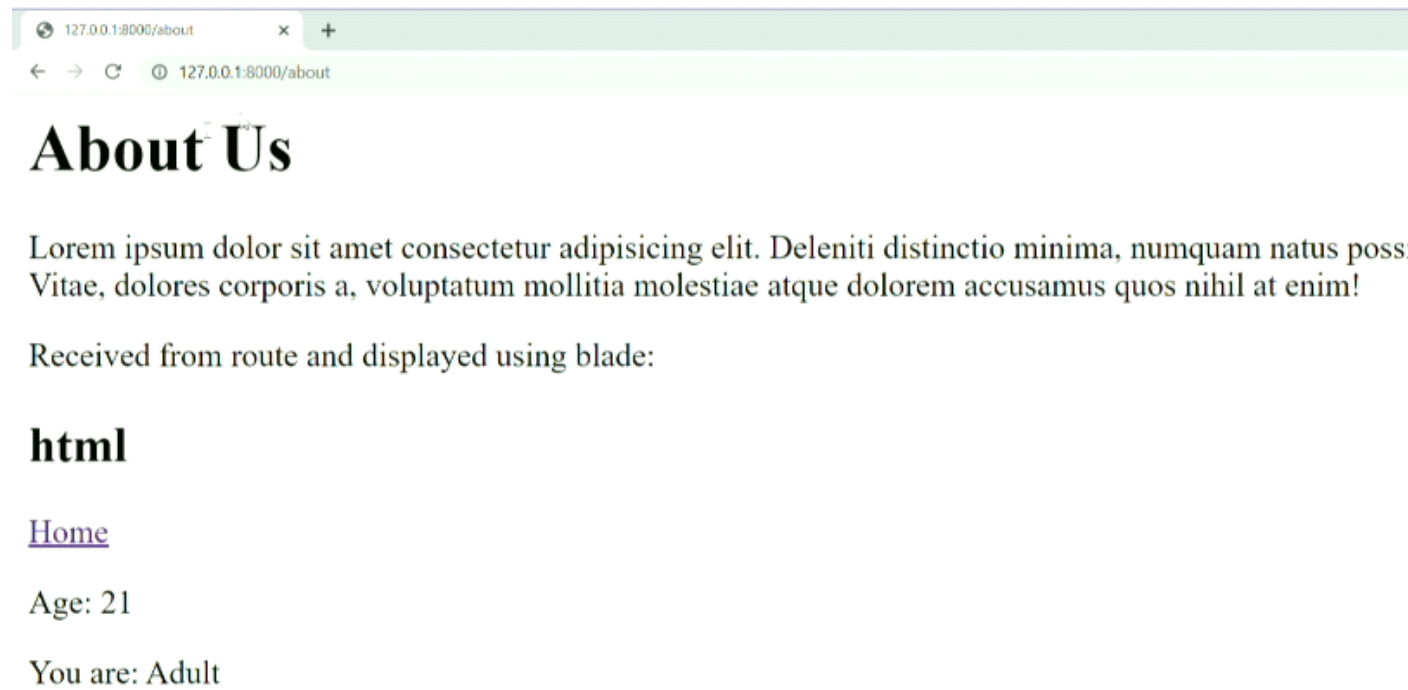
You are: Adult

Use of @unless:

This can be used as NOT operator.

```
about.blade.php X web.php
1 <h1>About Us</h1>
2 <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Deleniti
3 <p>Received from route and displayed using blade: {!! $data !!}</p>
4 <a href="{ { url('/') } }">Home</a>
5 <p>Age: { { $age } }</p>
6 <p>You are:
7 @if ($age < 18)
8 Young
9 @elseif ($age < 60)
10 Adult
11 @else
12 Senior
13 @endif
14
15 @unless($age >= 18)
16 You are eligible.
17 @endunless
18 </p>
```

Output:



Module 161

Laravel – Loops in Blade

Create a Route on about with an array parameter in “web.php”.

```
Route::get('/about', function () {  
    return view('about')->with(  
        ['data' => '<h2>html</h2>',  
         'age' => 15,  
         'languages' => ['PHP', 'JavaScript', 'C++', 'Java', 'C#']  
        ]  
    );  
});
```

Just to check of the array is passed or not in “About.blade.php”.

```
about.blade.php • web.php  
1 <h1>About Us</h1>  
2 <pre>  
3     {{ print_r($language )}}  
4 </pre>
```

@For loop:

```
@for ($i=0;$i<count($languages);$i++)  
    {{ $languages[$i] }}<br>  
@endfor
```

Output:



@For Each:

```
<h2>Foreach</h2>
@foreach($languages as $language)
|   {{ $language }}<br>
@endforeach
```

Foreach

PHP
JavaScript
C++
Java
C#

@While: This loop will empty the array at the end. So, this loop will be used in such situation we want to empty the array at the end.

```
<h2>While</h2>
@while ($language = array_pop($languages))
| {{ $language }}<br>
@endwhile
```

While

C#

Java

C++

JavaScript

PHP

@Forelse:

```
<h2>Forelse</h2>
@forelse($languages as $language)
| $language<br>
@empty
| No element in the array<br>
@endforelse
```

Output:

Forelse

No element in the array

As @While left no element in this array. So, this @empty section will be displayed on the screen.

Module 162

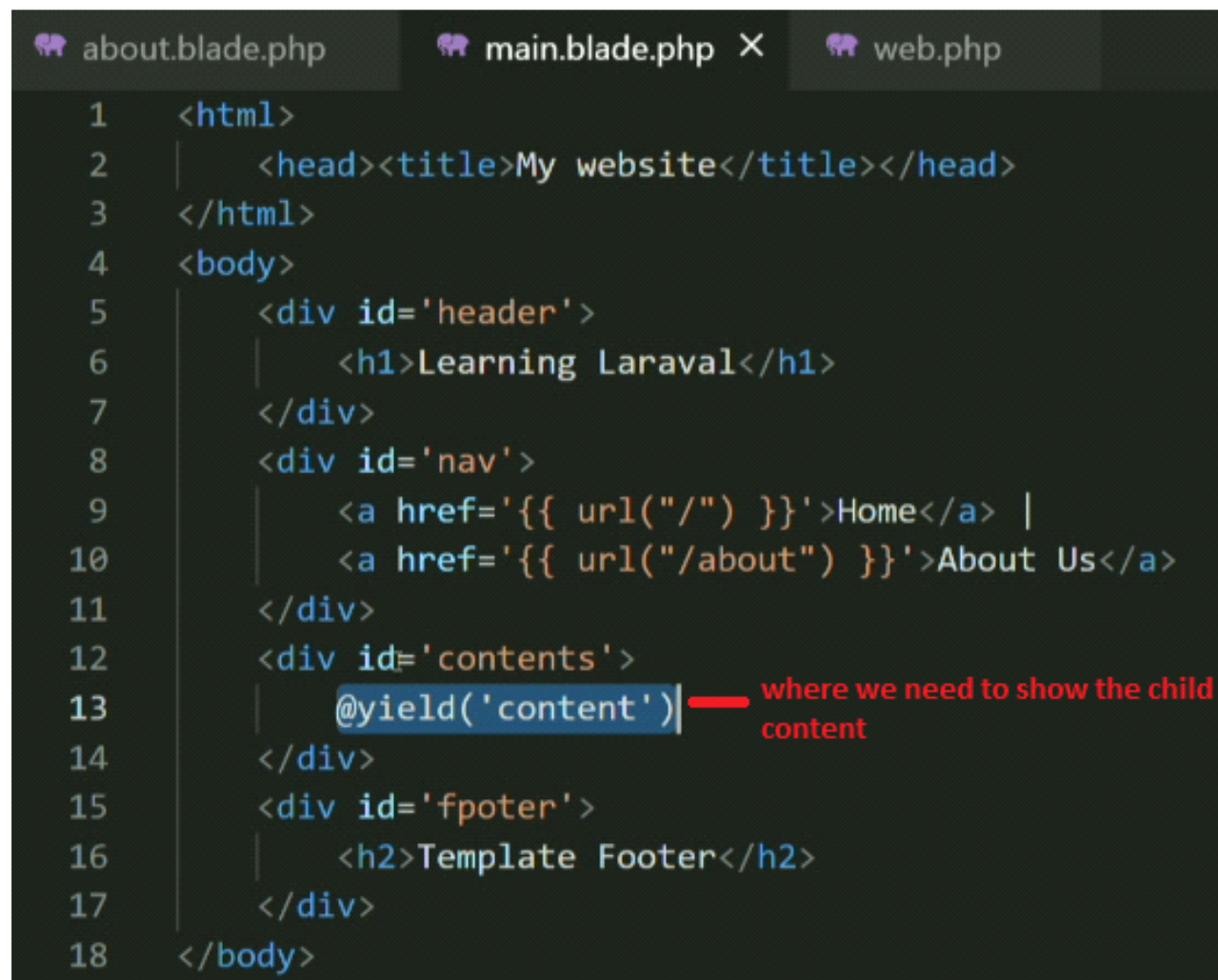
Laravel – Parent Template

We basically define the basic template in the Parent Template. For example, header, footer, navigation bar, left bar, right bar and main contents section.

We will specify in the child template which one is its parent and will use the variables accordingly.

Create a “layouts” folder under “Views”.

Create a file “main.blade.php” in “layouts” folder.



```
1 <html>
2 |   <head><title>My website</title></head>
3 </html>
4 <body>
5 |   <div id='header'>
6 |     <h1>Learning Laraval</h1>
7 |   </div>
8 |   <div id='nav'>
9 |     <a href='{{ url("/") }}'>Home</a> |
10 |    <a href='{{ url("/about") }}'>About Us</a>
11 |   </div>
12 |   <div id='contents'>
13 |     @yield('content') — where we need to show the child
14 |   </div>
15 |   <div id='fpoter'>
16 |     <h2>Template Footer</h2>
17 |   </div>
18 </body>
```

Module 163

Laravel Child Templates

You have already studied the parent templates. Suppose the following directive in the parent template:

```
@yield('content')
```

Now we will discuss how to create the child template to fill the contents in the parent template. Note that the contents of the child template are used to fill the parent template.

For this you will have to first mention that which layout this template is using. You can mention the layout as follows:

```
@extends('layouts.main')
```

Next you will define the section that will fill the parent template. For this you will use the “sections” as follows:

```
@section('content')
```

```
<h1>Contents</h1>
```

```
<P> These are sample contents</P>
```

```
@endsection
```

Note that the sample contents mentioned above will be placed at the location where you have used `@yield('content')` directive.

Furthermore, you can define the title of the page as well. This can be done as follows:

```
@section('content','Books')
```

```
<h1>Contents</h1>
```

```
<P> These are sample contents</P>
```

```
@endsection
```

Now the title of the page will be “Books”.

Module 164

Laravel Working with database

Almost every modern web application interacts with a database. Laravel makes interacting with databases extremely simple across a variety of supported databases using raw SQL, a fluent query builder, and the Eloquent ORM. Currently, Laravel provides first-party support for five databases:

MariaDB 10.2+ (Version Policy)

MySQL 5.7+ (Version Policy)

PostgreSQL 9.6+ (Version Policy)

SQLite 3.8.8+

SQL Server 2017+ (Version Policy)

Configuration of Laravel Database

The configuration for Laravel's database services is located in your application's `config/database.php` configuration file. In this file, you may define all of your database connections, as well as specify which connection should be used by default. Most of the configuration options within this file are driven by the values of your application's environment variables. Examples for most of Laravel's supported database systems are provided in this file.

Configuration Using URLs

Typically, database connections are configured using multiple configuration values such as host, database, username, password, etc. Each of these configuration values has its own corresponding environment variable. This means that when configuring your database connection information on a production server, you need to manage several environment variables.

Environment Configuration

It is often helpful to have different configuration values based on the environment where the application is running. For example, you may wish to use a different cache driver locally than you do on your production server.

To make this a cinch, Laravel utilizes the DotEnv PHP library. In a fresh Laravel installation, the root directory of your application will contain a `.env.example` file that defines many common environment variables. During the Laravel installation process, this file will automatically be copied to `.env`.

While configuring database also check database setting in `.env` file for the following variables

`DB_CONNECTION`, `DB_HOST`, `DB_PORT`, `DB_DATABASE`, `DB_USERNAME`,
and `DB_PASSWORD`.

Laravel Migrations

Migrations are like version control for your database, allowing your team to modify and share the application's database schema. Migrations are typically paired with Laravel's schema builder to build your application's database schema. If you have ever had to tell a teammate to manually add a column to their local database schema, you've faced the problem that database migrations solve.

Generating Migrations

To create a migration, use the make:migration Artisan command:

```
php artisan make:migration create_users_table
```

The new migration will be placed in your database/migrations directory. Each migration file name contains a timestamp, which allows Laravel to determine the order of the migrations.

Module 165

Creating Tables

To create a table we will run migration command first

```
php artisan make:migration create_users_table
```

The new migration will be placed in your database/migrations directory. Each migration file name contains a timestamp, which allows Laravel to determine the order of the migrations.

Creating Tables

To create a new database table, use the create method on the Schema facade. The create method accepts two arguments: the first is the name of the table, while the second is a closure which receives a Blueprint object that may be used to define the new table:

```
use Illuminate\Database\Schema\Blueprint;
```

```
use Illuminate\Support\Facades\Schema;
```

```
Schema::create('users', function (Blueprint $table) {
```

```
$table->id();
```

```
$table->string('name');
```

```
$table->string('email');
```

```
$table->timestamps();
```

```
});
```

When creating the table, you may use any of the schema builder's column methods to define the table's columns.

Module 166

Running Migrations

To run all of your outstanding migrations, execute the migrate Artisan command:

```
php artisan migrate
```

If you would like to see which migrations have run thus far, you may use the migrate:status Artisan command:

```
php artisan migrate:status
```

Note: If you receive a "class not found" error when running migrations, try running the composer dump-autoload command.

Rolling Back Migrations

To roll back the latest migration operation, you may use the rollback Artisan command. This command rolls back the last "batch" of migrations, which may include multiple migration files:

```
php artisan migrate:rollback
```

You may roll back a limited number of migrations by providing the step option to the rollback command. For example, the following command will roll back the last five migrations:

```
php artisan migrate:rollback --step=5
```

The migrate:reset command will roll back all of your application's migrations:

```
php artisan migrate:reset
```

Module 167

Adding a Foreign key

Foreign Key Constraints

Laravel also provides support for creating foreign key constraints, which are used to force referential integrity at the database level. For example, let's define a `user_id` column on the `posts` table that references the `id` column on a `users` table:

```
use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

Schema::table('posts', function (Blueprint $table) {

    $table->unsignedBigInteger('user_id');

    $table->foreign('user_id')->references('id')->on('users');});
```

Since this syntax is rather verbose, Laravel provides additional, terser methods that use conventions to provide a better developer experience. When using the `foreignId` method to create your column, the example above can be rewritten like so:

```
Schema::table('posts', function (Blueprint $table) {

    $table->foreignId('user_id')->constrained();

});
```

The `foreignId` method creates an UNSIGNED BIGINT equivalent column, while the `constrained` method will use conventions to determine the table and column name being referenced. If your table name does not match Laravel's conventions, you may specify the table name by passing it as an argument to the `constrained` method:

```
Schema::table('posts', function (Blueprint $table) {

    $table->foreignId('user_id')->constrained('users');

});
```

You may also specify the desired action for the "on delete" and "on update" properties of the constraint:

```
$table->foreignId('user_id')

->constrained()

->onUpdate('cascade')
```

Module 168

Running Raw SQL

You can insert data in 2 ways:

Method 1:

Open phpMyAdmin

Select the database you'd like to run a query on

You must select a database from the left column first

Select SQL

Look for the database name again above the text field, confirm this is the correct database you intend to run a query on

Write or paste your query in the text field

Click GO

Some queries will require a second confirmation in order to run.

Method 2:

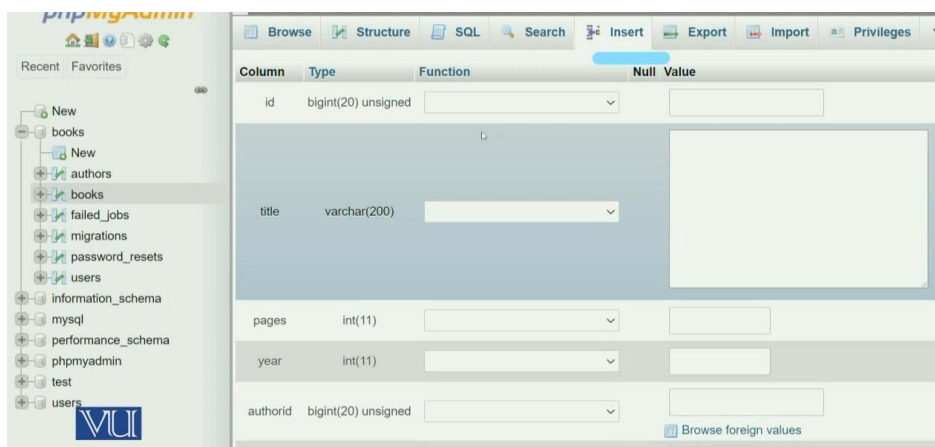
Open phpMyAdmin

Select the database you'd like to run a query on

You must select a database from the left column first

Select Insert

Look for the database name again above the text field, confirm this is the correct database table in which you intend to add data



Insert data in input field and Click GO

Resource Controller:

If you think of each Eloquent model in your application as a "resource", it is typical to perform the same sets of actions against each resource in your application.

Laravel resource routing assigns the typical create, read, update, and delete ("CRUD") routes to a controller with a single line of code. To get started, we can use the make:controller Artisan command's --resource option to quickly create a controller to handle these actions:

```
php artisan make:controller BookController --resource
```

Module 169

Laravel Views Introduction

It's not practical to return entire HTML documents strings directly from your routes and controllers. Thankfully, views provide a convenient way to place all of our HTML in separate files. Views separate your controller / application logic from your presentation logic and are stored in the resources/views directory.

A simple view may look like this

<!-- View stored in resources/views/greeting.blade.php --

```
<html>

  <body>

    <h1>Hello, {{ $name }}</h1>

  </body>

</html>
```

You may create a view by placing a file with the .blade.php extension in your application's resources/views directory. The .blade.php extension informs the framework that the file contains a Blade template. Blade templates contain HTML as well as Blade directives that allow you to easily echo values, create "if" statements, iterate over data, and more.

Once you have created a view, you may return it from one of your application's routes or controllers using the global view helper:

```
Route::get('/', function () {

return view('greeting', ['name' => 'VU']);});
```

To fetch data for view from database in controller you may use this method given below. It

class UserController extends Controller

```
{/**

  * Show a list of all of the application's users.

  * @return \Illuminate\Http\Response

  */

public function index()

{ $users = DB::table('users')->get();

  return view('user.index', ['users' => $users]); }}
```

Here “return view('user.index', ['users' => \$users]);” will pass data to view user after fetching data from data base.

Module 170

Laravel Query Builder

Laravel's database query builder provides a convenient, fluent interface to creating and running database queries. It can be used to perform most database operations in your application and works perfectly with all of Laravel's supported database systems.

The Laravel query builder uses PDO parameter binding to protect your application against SQL injection attacks. There is no need to clean or sanitize strings passed to the query builder as query bindings.

Retrieving All Rows From A Table

You may use the table method provided by the DB facade to begin a query. The table method returns a fluent query builder instance for the given table, allowing you to chain more constraints onto the query and then finally retrieve the results of the query using the get method:

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use App\Http\Controllers\Controller;
```

```
use Illuminate\Support\Facades\DB;
```

```
class UserController extends Controller
```

```
{
```

```
/**
```

```
 * Show a list of all of the application's users.
```

```
 *
```

```
 * @return \Illuminate\Http\Response
```

```
 */
```

```
public function index()
```

```
{
```

```
    $books= DB::table('books')->get();
```

```
    return view('books.index', [books=> $books]);
```

```
}  
  
}
```

If you don't need an entire row, you want to apply a condition that only those books which are entered after or in 2021 then this method will return the only those values.

```
$books= DB::table('books')->where('year','>=', 2021)->get();
```

Joins

The query builder may also be used to add join clauses to your queries. To perform a basic join, you may use the join method on a query builder instance. The first argument passed to the join method is the name of the table you need to join to, while the remaining arguments specify the column constraints for the join. You may even join multiple tables in a single query:

```
*/  
public function index()  
{  
    // $books = DB::select('SELECT * FROM books');  
    $books = DB::table('books')  
        ->join('authors', 'books.authorid', '=', 'authors.id')  
        ->select('books.*', 'authors.name')  
        ->get();  
    return view('books.index', [ 'books' => $books]);  
}
```

Module 171

Eloquent ORM Introduction

Laravel includes Eloquent, an object-relational mapper (ORM) that makes it enjoyable to interact with your database. When using Eloquent, each database table has a corresponding "Model" that is used to interact with that table. In addition to retrieving records from the database table, Eloquent models allow you to insert, update, and delete records from the table as well.

Table Names

After glancing at the example above, you may have noticed that we did not tell Eloquent which database table corresponds to our Flight model. By convention, the "snake case", plural name of the class will be used as the table name unless another name is explicitly specified. So, in this case, Eloquent will assume the Flight model stores records in the flights table, while an AirTrafficController model would store records in an air_traffic_controllers table.

If your model's corresponding database table does not fit this convention, you may manually specify the model's table name by defining a table property on the model:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Flight extends Model
{
    /**
     * The table associated with the model.
     *
     * @var string
     */
    protected $table = 'my_flights';
}
```

Retrieving Models

Once you have created a model and its associated database table, you are ready to start retrieving data from your database. You can think of each Eloquent model as a powerful query builder allowing you to fluently query the database table associated with the model. The model's all method will retrieve all of the records from the model's associated database table:

```
use App\Models\Flight;
```

```
foreach (Book::all() as $book) {  
  
    echo $book->name;  
  
}
```

Building Queries

The Eloquent all method will return all of the results in the model's table. However, since each Eloquent model serves as a query builder, you may add additional constraints to queries and then invoke the get method to retrieve the results:

```
$books= Book::where('active', 1)  
  
->orderBy('name')  
  
->take(10)  
  
->get();
```

For Insert, Update, Delete

Insert

```
$book = new Book;  
  
$book->title = "Introduction to Www"  
  
$book->save ();
```

Update

```
$book = Book: :find (1);  
  
$book->title = "Introduction to CS";  
  
$book->save ());
```

Delete

```
$book = Book: :find (1);  
  
$book->delete ();
```

Module 172

Laravel Eloquent (Getting data)

You may generate various other types of classes when generating a model, such as factories, seeders, policies, controllers, and form requests.

To get started, let's create an Eloquent model. Models typically live in the app\Models directory and extend the Illuminate\Database\Eloquent\Model class. You may use the make:model Artisan command to generate a new model:

Create BookController first using following command

```
php artisan make:controller BookController --resource
```

Now create eloquent model with --factory

```
php artisan make:model Book --factory
```

Previously we were fetching data by conventional laravel query builder now by fetching data by Eloquent room we will be using following method



```
public function index()
{
    // $books = DB::select('SELECT * FROM books');
    // $books = DB::table('books')
    //     ->join('authors', 'books.authorid', '=', 'authors.id')
    //     ->select('books.*', 'authors.name')
    //     ->get();
    $books = Book::all();
    return view('books.index', ['books' => $books]);
}
```

This will return all data in books table and pass it to the view books.index using Eloquent.

Module 173

Laravel Eloquent (Viewing data)

Make new controller as per Eloquent naming conventions and fetch data in your controller first as shown below in figure

```

public function index()
{
    $authors = Author::all();
    return view('authors.index', ['authors' => $authors]);
}

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    //
}

```

After fetching data from data base using Eloquent, data is now passed to the index view.

Alternatively `::all()` can also be used in model, when model is associated with database table

, you are ready to start retrieving data from your database. You can think of each Eloquent model as a powerful query builder allowing you to fluently query the database table associated with the model. The model's `all` method will retrieve all of the records from the model's associated database table:

```
use App\Models\Author;
```

```

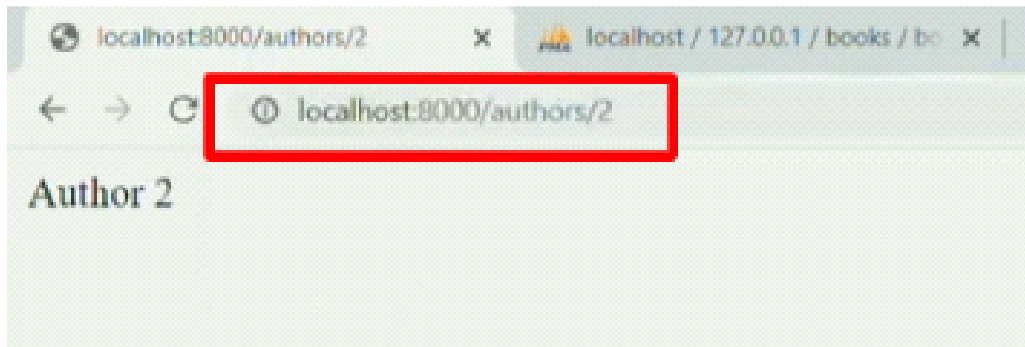
foreach (Author::all() as $author) {
    echo $author->name;
}

```

View data of specific ID

To view data of specific author will provide id of the the author of which data we need.

Resource controller will detect ID from the URL.



ID will be passed to the show function in our resource controller and controller will fetch data from database using model and pass it to blade view.

```
*/  
public function show($id) → View data of specific  
{  
    $author = Author::find($id);  
    return view('authors.show', ['author' => $author]);  
}
```

Module 174

Laravel - Eloquent (Relations)

Eloquent relationships are defined as methods on your Eloquent model classes. Since relationships also serve as powerful query builders, defining relationships as methods provides powerful method chaining and querying capabilities. For example, we may chain additional query constraints on this posts relationship:

```
$user->posts()->where('active', 1)->get();
```

But, before diving too deep into using relationships, let's learn how to define each type of relationship supported by Eloquent.

One To Many

A one-to-many relationship is used to define relationships where a single model is the parent to one or more child models. For example, one author has many books. Like all other Eloquent relationships, one-to-many relationships are defined by defining a method on your Eloquent model:

```
class Author extends Model
{
    use HasFactory;
    public function books() {
        return $this->hasMany(Book::class, 'authorid');
    }
}
```

Module 175

Laravel Form Request (Create)

Blade is the simple, yet powerful templating engine that is included with Laravel. Unlike some PHP templating engines, Blade does not restrict you from using plain PHP code in your templates. In fact, all Blade templates are compiled into plain PHP code and cached until they are modified, meaning Blade adds essentially zero overhead to your application. Blade template files use the `.blade.php` file extension and are typically stored in the `resources/views` directory.

CSRF Field

Anytime you define an HTML form in your application, you should include a hidden CSRF token field in the form so that the CSRF protection middleware can validate the request. You may use the `@csrf` Blade directive to generate the token field:

```
<form method="POST" action="/profile">
```

```
    @csrf
```

```
    ...
```

```
</form>
```

Method Field

Since HTML forms can't make PUT, PATCH, or DELETE requests, you will need to add a hidden `_method` field to spoof these HTTP verbs. The `@method` Blade directive can create this field for you:

```
<form action="/foo/bar" method="POST">
```

```
    @method('PUT')
```

```
</form>
```

Blade views may be returned from routes or controller using the global view helper. Of course, as mentioned in the documentation on views, data may be passed to the Blade view using the view helper's second argument:

```
Route::get('/', function () {  
    return view('greeting', ['name' => 'Finn']);  
});
```

In our lecture video we will be writing these two function for form view and for form store .

```

public function create()
{
    return view('authors.create');
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $author = new Author;
    $author->name = $request->input('name');
    $author->email = $request->input('email');
    $author->category = $request->input('category');
    $author->save();
}

```

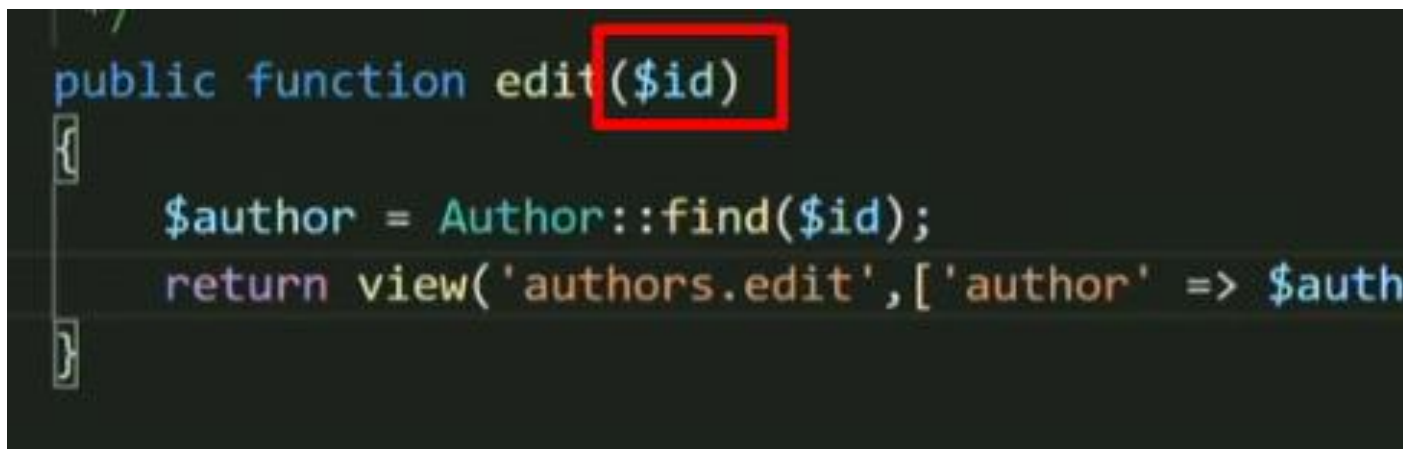
Form create will pass the control of application blade view template to view the form. After for submission it route will direct the control to store controller function to store the form.

Module 176

Laravel Form Request (Update)

This controller method will process the edit form. It is very similar to store(). We will validate, update, and redirect.

Create a new function 'edit' in controller, this function will be getting an id from view, on the basis of that id function edit will request data from author table only of that specific id.



```
public function edit($id)
{
    $author = Author::find($id);
    return view('authors.edit', ['author' => $author]);
}
```

After getting data from database, controller function will pass data object to view where data will be represented in editable form.

After submission of form data will be submitted to controller function update which will update data in database accordingly using controller function given below.

```
public function update(Request $request, $id)
{
    $author = Author::find($id);
    $author->name = $request->input('name');
    $author->email = $request->input('email');
    $author->category = $request->input('category');
    $author->save();
    return view('authors.show', ['author' => $author]);
}
```

Module 177

Laravel Form Request (Delete)

When Form is displayed with delete button, on pressing delete button we can pass id of instance which needs to be deleted to the controller function destroy directly. This could also be done using JavaScript.

When Id is passed to controller function destroy then function after loading author model will execute delete() function. On execution of delete() function in model delete query will run and instance will be deleted.

```
public function destroy($id)
{
    $author = Author::find($id);
    $author->delete();
    return redirect(url('authors'));
}
```

Module 178

Laravel CRUD Operations

You have already studied about the Create, Update and Delete operations in previous topics. Here we will continue the same code and will add the links to let users perform these operations.

Consider the code given below:

```
@foreach ($authors as $author)

<tr>

    <td> {{ $author->id }}</td>

    <td> {{ $author->name }}</td>

    <td> {{ $author->email }}</td>

    <td> {{ $author->category }}</td>

    <td>

        <a href='{{ url("authors/".$author->id) }}'>Details</a> |

        <a href='{{ url("authors/".$author->id."/edit") }}'>Edit</a> |

        <a href='{{ url("authors/".$author->id."/delete") }}'>Delete</a> |

    </td>

</tr>

@endforeach
```

In the above code, we have executed “foreach” loop to display the information of each author. For each author we will display the author id, name, email and category. Furthermore, now we will add the following links against each author:

Details: Clicking this link will display the details of each author. The information will be displayed using author id.

Edit: Click this link will help us edit the details of the author.

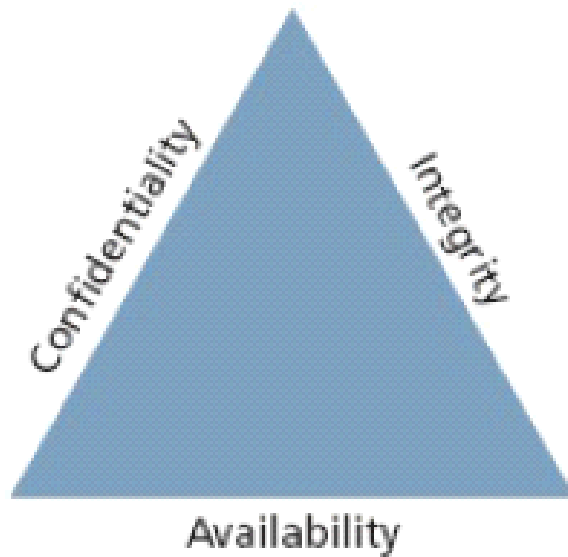
Delete: Clicking this link will help us delete a particular author.

Note: As all these tasks are performed in loop, the links will appear against each author record.

Module 179

Security principals

Information security is normally mentioned in the form of a triangle having three dimensions as shown below:



We need to consider all of these three dimensions for implementing the security of a webpage.

Confidentiality refers to the privacy of the data. It means that the private data of a user should only be accessible to that user. No other user should have the access to the private data of other users.

Integrity means that the data should be correct and secure. This means that the unauthorized access and modifications should be prevented.

Availability means that the website and the data should always be available to the authorized users whenever requested.

Risk Assessment and Management:

It is important to assess the risk in the web development domain. Risk assessment means that we measure that how the risk is likely to occur and what will be the impact of the risk

We can discuss the security principals from different angles. We will discuss the following aspects w.r.t. risks:

Actors

Impact

Threat

Vulnerabilities

Actors:

Generally there are three categories of Actors.

Internal actors are those people that work within the organization e.g. cashier, IT staff, CEO etc.

External actors are the people outside the organization. Majority of the attacks are launched by these people. Normally these people have bad intentions and high skills to penetrate within the system of the organization

Partner actors Patterns are the people an organization works with. Partners may also be source of an attack and may result in consequences as the partners have access to the data of the organization (system) in majority of the cases.

Impact:

Impact refers to the cost and loss that we will have to bear if attack is successful, for example, which systems were affected and which data was stolen. Impact relates to loss of the availability, confidentiality and/or integrity.

A loss of availability: It prevents users from accessing some or all of the systems.

A loss of confidentiality: It means your data is exposed to unauthorized users.

A loss of integrity: it means that your data is changed and you don't have the accurate data now e.g. placing fake orders, changing the home address of the user etc.

Threats:

There are different categories of threats. Normally threats are categorized using the "STRIDE" mnemonic.

Spoofing—The attacker uses someone else's information to access the system.

Tampering—The attacker modifies some data in non-authorized ways.

Repudiation—The attacker removes all trace of their attack, so that they cannot be held accountable for other damages done.

Information disclosure—The attacker accesses data they should not be able to.

Denial of service—The attacker prevents real users from accessing the systems.

Elevation of privilege—The attacker increases their privileges on the system thereby getting access to things they are not authorized to do.

Vulnerabilities:

Vulnerabilities refer to the security holes in your system. Following are top five classes of vulnerability from the Open Web Application Security Project:

1. Injection

2. Broken authentication and session management
3. Cross-site scripting
4. Insecure deserialization
5. Security misconfiguration

Security Policy:

For each system in an organization, there should be a well-defined policy related to the usage and security of the system. A policy can be defined with respect to the following perspectives:

Usage policy means that what systems are available for the users to use and under which circumstances.

Authentication policy defines that how the users are granted access to the systems.

Legal policies includes a number of things e.g. data retention and backup policies, accessibility requirements etc.

Whatever the circumstances are, the business should continue. So, it is better to make plans and follow the good practices to keep the system going in case of hard security circumstances.

Following are some good practices that should be considered:

Admin Password Management

Backups and Redundancy

Geographic Redundancy

Stage Mock Events

Auditing

Reference:

Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

Module 180

Security by design

Security needs to be implemented in each phase of software development e.g. when you are collecting the requirements, you need to check your security requirements at the same stage, what should be your security policy etc. similarly, you need to find out and implement the security requirements at design phase e.g. you need assess the threats and risks, you need to devise the plans for redundancy management etc. The same applies to the web development

Following are some points that you will have to consider at each phase:

Requirements:

- Privacy needs

- Security Policy

- CIA Triad

Design

- Threat assessment

- Risk Assessment

- Redundancy Planning

Implementation

- Pair Programming

- Code Reviews

- Defensive Programming

Testing

- Security Unit Tests

- Vulnerability Tests

- Test Cases

Deployment

- Penetration Testing

- Attach thyself

- Default Values

Module 181

Authentication:

To maintain confidentiality and integrity, the identity of an individual must be verified. The process of verifying individual identity is known as authentication. Authentication is the process to confirm what someone says they are. On successful authentication the user is allowed to access the required resources.

Authentication Factors:

The things that you ask from someone to verify their identity are known as authentication factors. There are three factors of authentication factors

Knowledge: the things that you know like username and passwords, Security questions, among others. This information can be found by someone else and can be shared easily.

Ownership: the things you have that are certificates, access card, USB keys, software tokens, email, SMS among others. These factors are vulnerable to be stolen, Snatched like other possession. Moreover, some of these factors can be duplicated and can be misused.

Inherence: something you are like typing speed, Key pattern intervals, Biometrics (Retinas, Facial recognition, fingerprints among others). These authentication factors are less vulnerable and are difficult to forge and share.

Authentication Factors (usage perspective):

Authentication can be Single factor or Multi-factor authentication. In single-factor authentication, a single one of the above factors is used to validate the user's claim. In case of single factor authentication, the knowledge factor is comparatively more vulnerable than ownership and inherence factors. Single-factor based authentication can be implemented as knowing the password and relies on the strength of the password. Password can be easily learnt by the person looking over the shoulder during the password entry and can be retrieved from the browser or through phishing scams and other attacks. In the scenario where more authentication confidence is required like bank transactions or accessing sensitive information, multi-factor authentication needs to be considered.

Multi-factor authentication requires at least two distinct factors like one from knowledge and other from ownership. This kind of authentication increases security and reduces attack threats. This is because to break multifactor authentication, the attacker needs to have two different attack vectors. ATM machine is an example of multi-factor authentication where the customer needs to use the ATM card and enter PIN as well.

Type of authentication

There are various ways a user can be authenticated - some of which include, Session-based authentication, Cookie-based authentication, Token-based authentication, Claims-based authentication and Hypertext Transfer Protocol (HTTP) authentication mechanisms.

HTTP authentication:

HTTP authentication is one of the simplest ways to identify users logged into a system. Common types of HTTP authentication include: Basic, Digest and Form Based

HTTP basic authentication: The users need to provide username and password while requesting to the server and are sent in the authorization header. The user's name and passwords are concatenated into a single string username: password and encoded in Base64.

HTTP Digest authentication: In this technique all the authentication credentials like http method, username, password, and requested URIs are encrypted using MD5. However, this authentication is still vulnerable to main-in-the-middle attack.

Form based authentication: In Form-based authentication, the user credentials (username and password) are sent in a plain text (like Basic authentication) using forms, which can lead to exposure of usernames and passwords except connecting via HTTPS/SSL. One benefit is that, the developer is allowed to customize the authentication and error pages sent by the browser.

Third party authentication:

To store and maintain authentication credentials even if it is username and password is not so easy. Moreover, direct log in through username and password is not safe and secure. Therefore, most of the larger organizations use third party authentication services for authentication of users. Third party authentication service providers are the organization who have already invested in the authentication and allows you to use their system to authenticate your users.

Open Authentication (OAuth) is one of the most popular third party authentication services. OAuth is “An open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications”. OAuth allows users to authenticate a site by using credentials from another site and have many versions. OAuth allows users to share specific data with an application while keeping their usernames, passwords, and other information private. For example, an application can use OAuth services to obtain permission from users to store files in their Google Drives.

References:

<https://www.oauth.com/>
<https://developers.google.com/identity/protocols/oauth2>
<https://iq.opengenus.org/user-authentication-techniques-types/>

Module 182

Cryptography:

Cryptography is an information security tactic used to protect enterprise information and communication from cyber threats through the use of codes. It is considered the art of hiding information to prevent unauthorized access to your data.

This practice refers to secure information and communication techniques derived from mathematical concepts and a set of rule-based calculations, called algorithms, to transform messages in ways that are hard to decipher. These algorithms are then used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on the internet, and confidential communication like credit card transactions and emails.

Cryptography achieves several information security-related objectives including confidentiality, integrity, and authentication, and non-repudiation.

Basic Concepts:

Cryptography: The art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible, and then retransforming that message back to its original form

Plain text: The original intelligible message

Cipher text: The transformed message

Cipher: An algorithm for transforming an intelligible message into one that is unintelligible by transposition and/or substitution methods

Key: Some critical information used by the cipher, known only to the sender & receiver

Encipher (encode): The process of converting plaintext to cipher text using a cipher and a key

Decipher (decode): The process of converting cipher text back into plaintext using a cipher and a key

Cryptanalysis: The study of principles and methods of transforming an unintelligible message back into an intelligible message without knowledge of the key.

Code: An algorithm for transforming an intelligible message into an unintelligible one using a code-book

Ciphers: All the encryption algorithms are based on two general principles: substitution, in which each element in the plaintext is mapped into another element, and transposition, in which elements in the plaintext are rearranged.

Public Key Cryptography:

There are two main methods to encrypt and decrypt a message that need to be securely communicated between sender and receiver.

Symmetric Key: In symmetric key algorithms, the encryption and decryption keys are known both to sender and receiver. The encryption key is shared and the decryption key is easily calculated from it. The big challenge in the symmetric key ciphers is the exchange of secret before starting the communication.

Public key cryptography:

The public key ciphers, the issue of exchanging the secret has been resolved by using two different keys i.e., public key used for encryption and private key used for decryption.

In public key cryptography, encryption key is made public, but it is computationally infeasible to find the decryption key without the information known to the receiver.

Digital Signature:

Digital signature is a secure method used to validate the digital document send by the sender from three perspective:

To validate the sender who claim the digital document also known as authenticity.

The received document is same as send by the sender i.e., not changed in the middle which is termed as integrity of the document

The sender cannot deny the sent document which is termed as nonrepudiation.

Digital document can be signed by using the concept of public key cryptography. It can be done by encrypting the hash of the transmitted message and the receiver apply the same technique to get the hash and decrypting the signature.

References:

<https://www.triskelelabs.com/>

https://www.vssut.ac.in/lecture_notes/lecture1428550736.pdf

Module 183

HTTPS:

Hyper Text Transfer Protocol Secure (HTTPS) is an HTTP protocol running on the top of Transport Layer Security (TLS). TLS is an improved form of Secure Socket Layer (SSL).

Secure Handshakes:

The foundation for establishing a secure link happens during the initial handshake. This handshake must occur on an IP address level, so while you can host multiple secure sites on the same server, each domain must have its own IP address in order to perform the low-level handshaking as shown in the following figure.

Certificates:

The certificate that is transmitted during the handshake is actually an X.509 certificate, which contains many details including the algorithms used, the domain it was issued for, and some public key information. The complete X.509 specification can be found in the International Telecommunication Union's directory of public key frameworks.

From HTTP to HTTPS:

Despite all the advantages of a secure site (including a modest boost from some search engines in ranking, and an increasing trend to serve all websites over https), there are many considerations to face when migrating or setting up a secure site. Coordinating the migration of a website can be a complex endeavor involving multiple divisions of a company. In addition to marketing materials being updated in the physical world to use the new URL, there are some nontechnical issues that need to be addressed like the annual budget to purchase and renew a certificate from a certificate authority. In addition to the business questions there are also some technical considerations in migrating to HTTPS.

References:

R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

Module 184

Security Best Practices:

With all our previous discussion of security thinking, cryptographic principles, and authentication in mind, it's now time to discuss some practical things you can do to harden your system against attacks.

A system will be targeted either purposefully or by chance. The majority of attacks are opportunistic attacks where a scan of many systems identifies yours for vulnerabilities. Targeted attacks occur less often, but are by their nature more difficult to block. Either way there are some great techniques to make your system less of a target.

Auditing:

Auditing is the process by which a third party is invited (or required) to check over your systems to see if you are complying with regulations and your claims. Auditing happens in the financial sector regularly, with a third-party auditor checking a company's financial records to ensure everything is as it should be. Oftentimes, simply knowing an audit will be done provides

incentive to implement proper practices.

The practice of logging, where each request for resources is stored in a secure

log, provides auditors with a wealth of data to investigate. Linux, by default,

stores logs related to ssh (Secure Shell) and other network access. You can exert control

over these and the logging of your Apache server.

Another common practice is to use databases to track when records are edited

or deleted by storing the timestamp, the record, the change, and the user who was logged in. These logs are often stored in separate, audit tables.

References:

R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

Module 185

Common threats vectors:

These are the common factors that an attacker uses to penetrate to a web application.

Brute force attacks:

A brute force attack uses trial-and-error to guess login info, encryption keys, or find a hidden web page. Hackers work through all possible combinations hoping to guess correctly.

This is an old attack method, but it's still effective and popular with hackers. Because depending on the length and complexity of the password.

Automated tools help with brute force attacks. These use rapid-fire guessing that is built to create every possible password and attempt to use them. Brute force hacking software can find a single dictionary word password within one second.

Protection from Brute force attacks:

- Limit number of guesses

- CAPTCHA

- High encryption

- Two-factor authentication (2FA)

- Salt the hash

SQL injection:

An SQL injection is a type of cyber-attack in which a hacker uses a piece of SQL (Structured Query Language) code to manipulate a database and gain access to potentially valuable information.

It's one of the most prevalent and threatening types of attack because it can potentially be used against any web application or website that uses an SQL-based database.

Protection from SQL injection attacks:

- Skip quotes while entering user entered form data into the database

- Use place holders

Cross-site scripting (XSS):

Cross-site scripting (XSS) is a code injection attack that allows an attacker to execute malicious JavaScript in another user's browser.

The attacker does not directly target his victim. Instead, he exploits a vulnerability in a website that the victim visits, in order to get the website to deliver the malicious JavaScript for him. To the victim's browser, the malicious JavaScript appears to be a legitimate part of the website, and the website has thus acted as an unintentional accomplice to the attacker.

If an attacker can use your website to execute arbitrary JavaScript in another user's browser, the security of your website and its users has been compromised.

Stored Cross-Site Scripting (XSS)

In Stored XSS attacks the attacker is capable of saving content on the webpage which is reflected every time someone visits the website. It does not require someone to click a link necessarily. By using XSS and stored XSS, the attackers can do many things like:

- Stealing cookies which can be used for authentication

Defacing the website, presenting content which the webserver did not intend to

Phishing users in leaving credentials in fake login forms

Protection from SQL injection attacks:

Safe encoding: HTML encoding allows the web application to return typically unsafe characters in a safe manner. For example, the special characters "<" can be encoded as "<".

Insecure Directory Object Reference (IDOR):

IDOR vulnerabilities happen when developers have not implemented authorization requirements to access resources.

Eve, by simply changing an identifier, e.g., the document Rest parameter, she can access Alice's documents. This happens when the web application does not enforce authorization between objects, allowing attackers to enumerate values and test access to other points of data.

Protection from IDOR attacks:

Avoiding the use of magic numbers: An application should avoid using sequences of numbers when referencing data. For example, the documents had identifiers from 1000 to 1002. An attacker can check all document identifiers from 0 all the way to 10000.

Authorization should be properly implemented, it is also helpful to use GUID ("Globally Unique Identifier") or UUID ("Universally Unique Identifier") when referencing data

DOS attacks:

It is an attack on the computer or network that restricts, reduces, or prevents the system from restoring accessibility to its legitimate users. It is a kind of attack in which an attacker or intruder tries to deprive system users or authorized users of accessing their computers, networks, or sites. In DOS, the attacker focuses on the bandwidth of the victim to perform this attack.

A DoS attack is reframed with the name of the Distributed Denial of Service (DDoS) attack when a multitude of hacked systems target a single system (computer, website, or network). In other words, when a Denial-of-Service attack is performed using several compromised devices to attack a particular system, then that distributed nature of the attack is termed as a Distributed Denial of Service (DDoS) attack.

Protection from DOS/DDOS attacks:

Unfortunately, there is no 100% successful ways to protect a victim from falling under the prey of malicious DoS/DDoS attackers. But, users can apply some prevention tactics to reduce the likelihood an attacker may use a computer to attack other computers. These prevention tactics are:

Install Antivirus software with the latest updates.

Install a firewall and try to configure it with the most recent updates to restrict traffic.

Apply filtering of emails to manage unwanted traffic.

Security misconfiguration:

Improper configuration of servers' results in a wide range of errors that can be exploited by the attackers and is known as security misconfigurations. These errors include out-of-date software or patches, open and virtual mail relays, input-coupled control, arbitrary program execution among others.

Consider, for example, that most websites use an HTML form to allow users to contact the website administrator or other users. If the form allows users to select the recipient from a dropdown, then what is being transmitted is crucial since it could expose your mail server as a virtual open mail relay. By transmitting the email address of the recipient, the contact form is at risk of abuse since an attacker could send to any email they want.

References:

<https://www.kaspersky.com/resource-center/definitions/brute-force-attack>

https://www.w3schools.com/cybersecurity/cybersecurity_web_applications_attacks.php

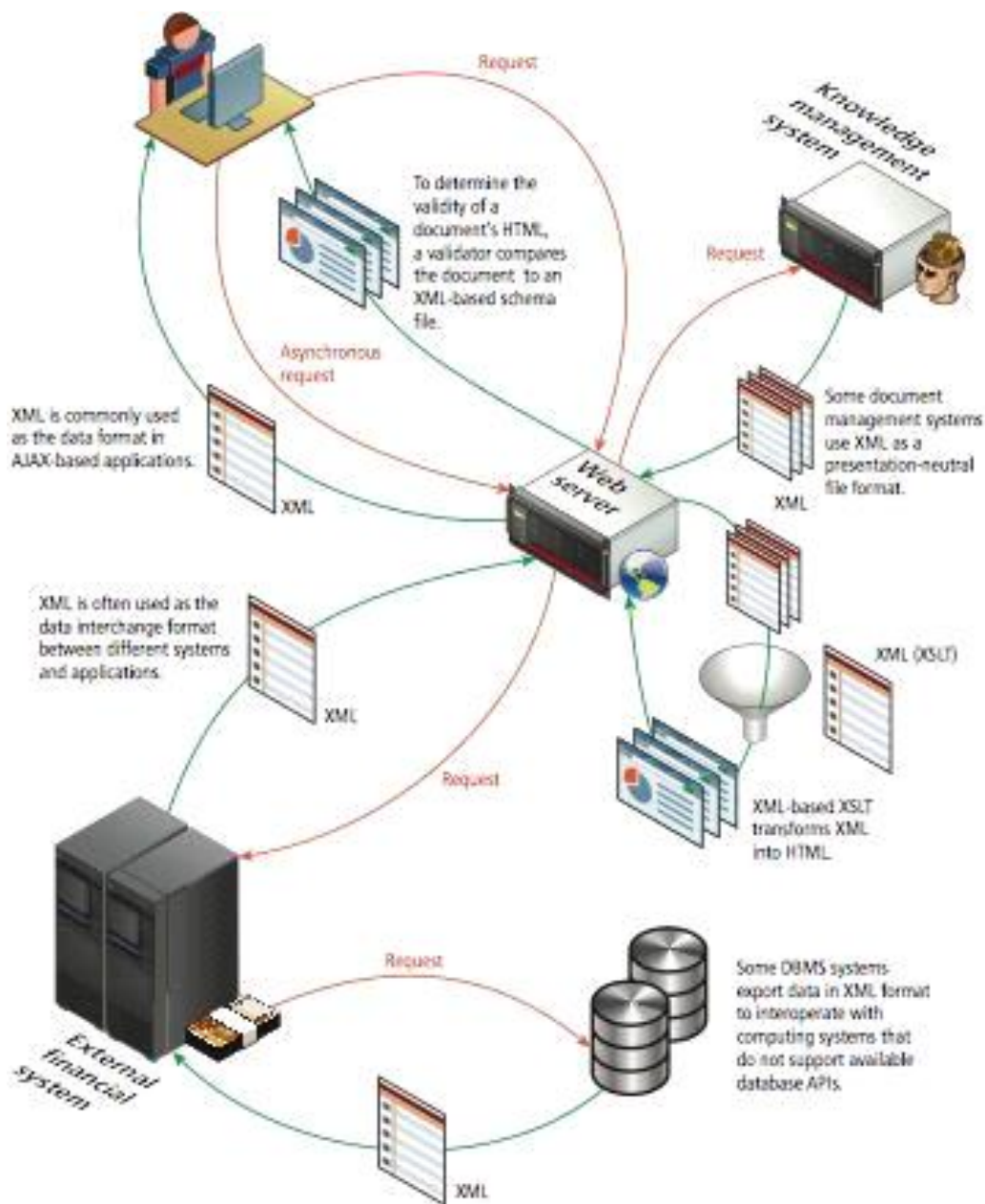
<https://www.w3schools.in/ethical-hacking/>

R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

Module 186

XML:

XML stands for eXtensible Markup Language and was designed to store and transport data. It is a markup language, but unlike HTML, XML can be used to mark up any type of data. One key benefit of XML data is that as plain text, it can be read and transferred between applications and different operating systems. XML is used on the web server to communicate asynchronously with the browser. Used as a data interchange format for moving information between systems over the Internet.



Well-Formed XML

Element names are composed of any of the valid characters

Element names can't start with a number

There must be a single-root element

All elements must have a closing element (or be self-closing)

Elements must be properly nested

Elements can contain attributes

Attribute values must always be within quotes

Element and attribute names are case sensitive

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<art>
  <painting id="290">
    <title>Balcony</title>
    <artist>
      <name>Manet</name>
      <nationality>France</nationality>
    </artist>
    <year>1868</year>
    <medium>Oil on canvas</medium>
  </painting>
</art>
```

Valid XML:

A valid XML document is one that is well formed and whose element and content conform to the rules of either its document type definition (DTD) or its schema

DTDs tell the XML parser which elements and attributes to expect in the document as well as the order and nesting of those elements

A DTD can be defined within an XML document or within an external file

Document Type Definition (DTD):

<pre><?xml version="1.0" encoding="ISO-8859-1"?> <art> <painting id="290"> <title>Balcony</title> <artist> <name>Manet</name> <nationality>France</nationality> </artist> <year>1868</year> <medium>Oil on canvas</medium> </painting> </art></pre>	<pre><?xml version="1.0" encoding="ISO-8859-1"?> <!DOCTYPE art [<!ELEMENT art (painting*)> <!ELEMENT painting (title,artist,year,medium)> <!ATTLIST painting id CDATA #REQUIRED> <!ELEMENT title (#PCDATA)> <!ELEMENT artist (name,nationality)> <!ELEMENT name (#PCDATA)> <!ELEMENT nationality (#PCDATA)> <!ELEMENT year (#PCDATA)> <!ELEMENT medium (#PCDATA)>]></pre>
---	---

Example of XML Schema:

XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<art>
  <painting id="290">
    <title>Balcony</title>
```

```

<artist>
  <name>Manet</name>
  <nationality>France</nationality>
</artist>
<year>1868</year>
<medium>Oil on canvas</medium>
</painting>
</art>

```

Corresponding XML Schema

```

<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
  <xs:element name="art">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="painting" maxOccurs="unbounded" minOccurs="0">
          <xs:complexType>
            <xs:attribute type="xs:integer" name="id"/>
            <xs:sequence>
              <xs:element type="xs:string" name="title"/>
              <xs:element name="artist">
                <xs:complexType>
                  <xs:choice>
                    <xs:element type="xs:string" name="name"/>
                    <xs:element type="xs:string" name="nationality"/>
                  </xs:choice>
                </xs:complexType>
              </xs:element>
              <xs:element type="xs:short" name="year"/>
              <xs:element type="xs:string" name="medium"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

```
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

References:

R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

Module 187

Xpath:

XML Path Language

Used for accessing elements and attributes in an XML document

Widely used in JavaScript for accessing certain elements/attributes of HTML documents

XPath expressions:

Expression	Description
nodename	All nodes with the name "nodename"
/	From root
//	Anywhere in the document
.	Current node
..	Parent node

@	Attribute
[]	Predicate

XPath Examples

```
<?xml version="1.0" encoding="ISO-
8859-1"?>
<art>
  <painting id="290">
    <title>Balcony</title>
    <artist>
      <name>Manet</name>
      <nationality>France</nationality>
    </artist>
    <year>1868</year>
    <medium>Oil on
canvas</medium>
  </painting>
</art>
```

XML

Corresponding XPath

```
/art
  Selects element "art"

//year
  Selects element "year"

//name/..
  Selects element "artist"

//painting/@id
  Selects attribute "id" in painting

//painting[year>1800]
  Selects "painting" elements with year > 1800
```

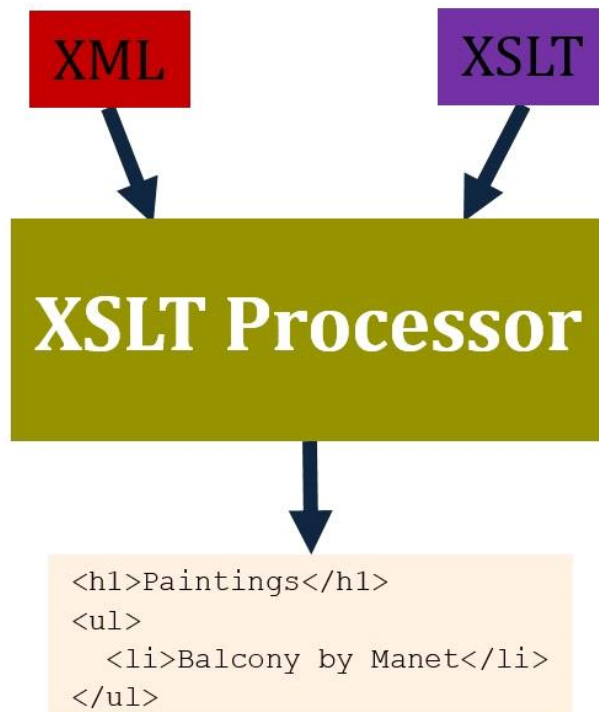
XSLT:

eXtensible Stylesheet Language Transformation (XSLT)

Used for transforming an XML document into another form

Defined as an XML document

Uses XPath for selecting elements and built-in functions for processing/transforming them



```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/art">
  <h1>Paintings</h1>
  <ul>
    <xsl:for-each select="painting">
      <li>
        <xsl:value-of select="title"/> by
        <xsl:value-of select="artist/name"/>
      </li>
    </xsl:for-each>
  </ul>
</xsl:template>
```

</xsl:stylesheet>

References:

R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

Module 188

XML Processing:

Every language has some libraries to process the XML document.

XML processing in Javascript:

XML processing using JavaScript: JavaScript provides two ways to process the XML document. 1) The approach that used to process the entire XML document is known as in-memory approach. The second method is to extract/pull the few lines or elements at a time is known as event or pull approach.

Let say we are receiving an XML document “art.xml” through ajax query. Request is sent to the server through send() method and object to the XML document “xmlDoc” will be received as response from the server. Further, all elements with name “painting” will be pulled and will be store in an object named “paintings” which can be used for further processing.

```
var xmlhttp = new XMLHttpRequest()

xmlhttp.open("GET","art.xml",false);

xmlhttp.send();

var xmlDoc = xmlhttp.responseXML;

var paintings = xmlDoc.getElementsByTagName("painting");
```

XML processing in jQuery:

Let suppose the XML document is in string form, first string is converted into the XML document and stored in the DOM object. After that the DOM object is converted into jQuery object and then all elements with name “painting” will be extracted and stored in the “paintings” which can be used for further processing.

```
var art = '<?xml ...';

// create the DOM object

var xmlDoc = $.parseXML(art);
```

```
// convert DOM object to jQuery object
var xml = $(xmlDoc);

// find all the painting elements
var paintings = xml.find("painting");
```

XML processing in PHP:

There are two ways to process XML document in PHP i.e., SimpleXML and XMLReader.

SimpleXML

Let suppose there is an XML file “art.xml”. The if condition is used to check whether the file is available or not. If file is available then XML file is loaded to an object, then first element is accessed and printed using “echo” command.

```
<?php

//xml file with name art.xml

$filename = 'art.xml';

// condition to check the file is available
if (file_exists($filename)) {

//if yes then file is loaded to “$art”

    $art = simplexml_load_file($filename);

    // access a single element
    $painting = $art->painting[0];

    // display elements
    echo '<h2>'.$painting->title.'</h2>';}
```

XMLReader:

```
$art = simplexml_load_file($filename);

$titles = $art->xpath('/art/painting/title');

foreach ($titles as $t) {

    echo $t . '<br/>';

}

$names = $art->xpath('/art/painting[year>1800]/artist/name');

foreach ($names as $n) {

    echo $n . '<br/>';
```

```
}
```

References:

R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

Module 189

JSON

JSON stands for JavaScript Object Notation and is a most commonly used text format for storing and transporting data. JSON is "self-describing" and easy to understand. JSON objects are in the key value pair form.



As compared to XML, JSON is quicker to read and write, and easy to parse. For AJAX applications, JSON is faster and easier than XML

Using JSON in Javascript:

JSON is parsed into a ready-to-use JavaScript object.

Example of JSON in Javascript:

//Let JSON is in text document

```
var text =
```

```
'{"artist": {  
    "name": "Manet",  
    "nationality": "France"  
}}';
```

```
//converting text into javascript object  
var a = JSON.parse(text);  
  
// can access value using the object  
alert(a.artist.nationality);  
  
//or a['artist']['nationality']
```

Using JSON in PHP:

```
<?php  
  
// convert JSON string into PHP object  
  
$text =  
  
    '{"artist": {  
        "name": "Manet",  
        "nationality": "France"  
    }  
}';  
  
$anObject = json_decode($text);  
  
// check for parse errors  
  
if (json_last_error() == JSON_ERROR_NONE) {  
    echo $anObject->artist->nationality;  
}  
  
?>
```

References:

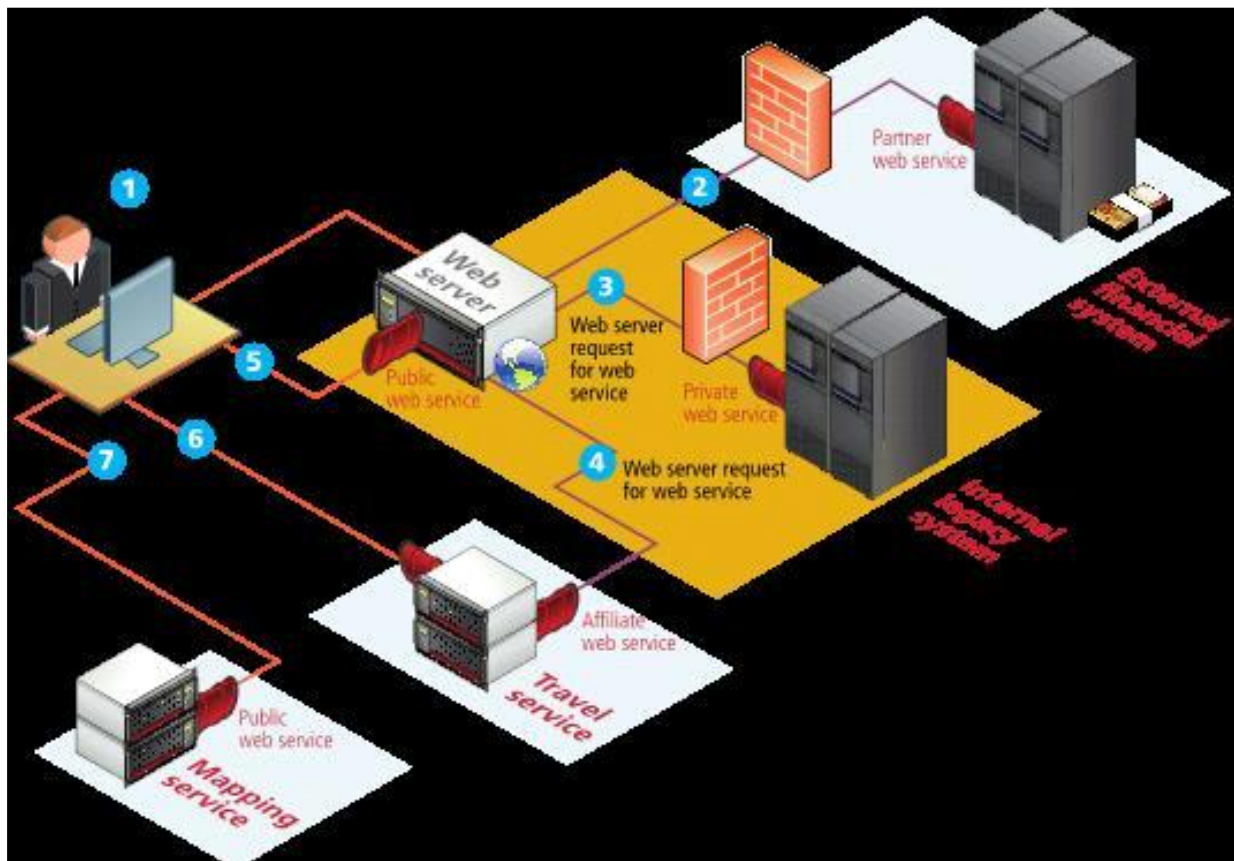
R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

<https://www.w3schools.com/js>

Module 190

Overview of web services:

When you are developing a website, most of time you need to use some services from an external or internal source which is known as web services. These sources may include external finance system, internal legacy system, travel service, and map services among others as shown in the following figure.



SOAP base web services:

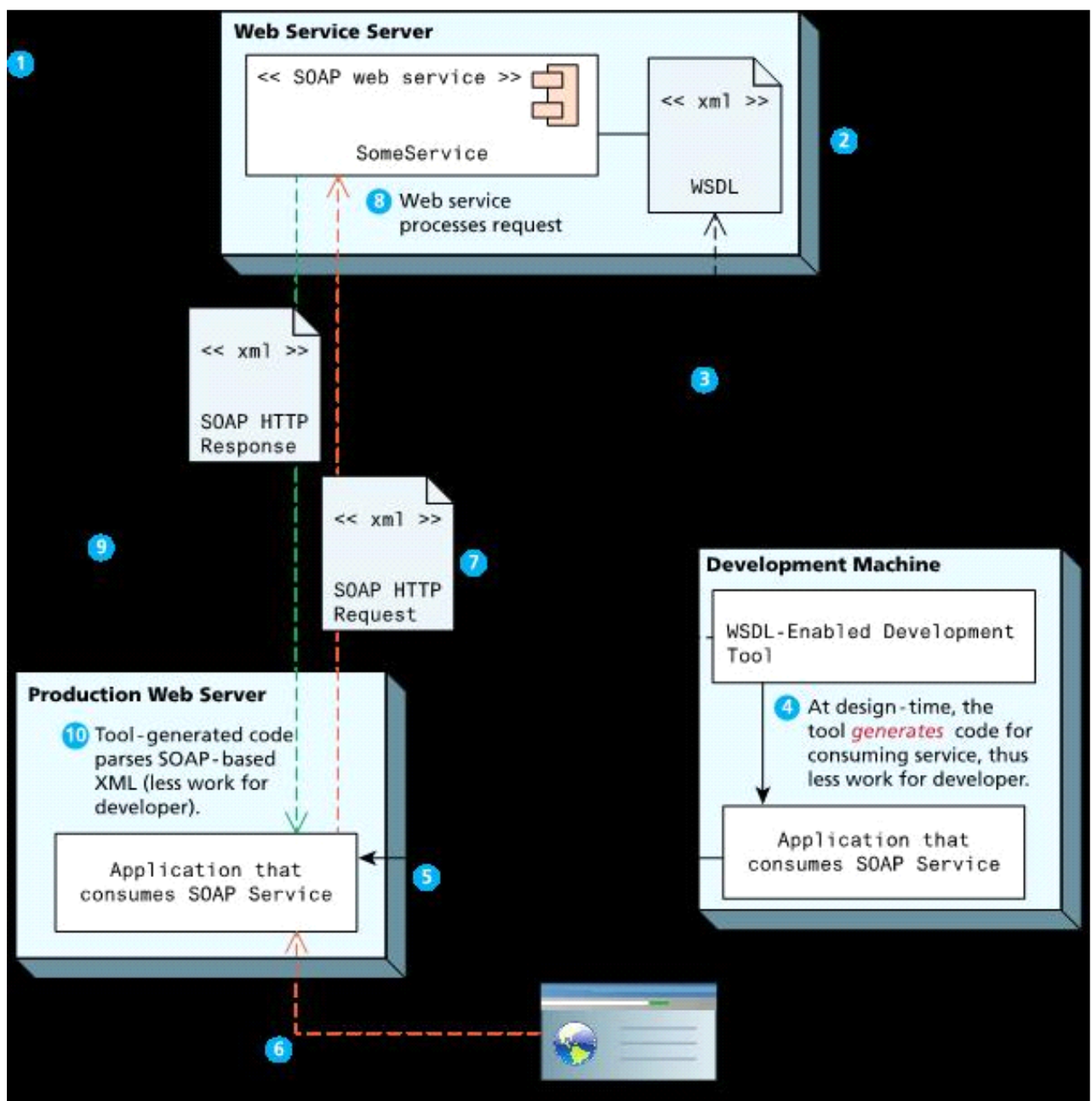
SOAP stands for Simple Object Access Protocol which is an XML-based protocol having the main benefit of implementing the SOAP Web Service as its security. SOAP offers a wrapper for sending web service-based messages over the Internet with the help of HTTP protocol. All its messages are usually in XML format. The SOAP message consists of:

SOAP document has a root element termed as the <Envelope> element. This element is the initial element used in an XML document.

SOAP document has a root element termed as the <Envelope> element. This element is the initial element used in an XML document.

Then next to the "envelope," which is categorized into two parts. The former is the 'header,' and the later is the 'body.' The header includes the routing-data that is essentially the information telling the XML document to whom or for which the client needs to be sent.

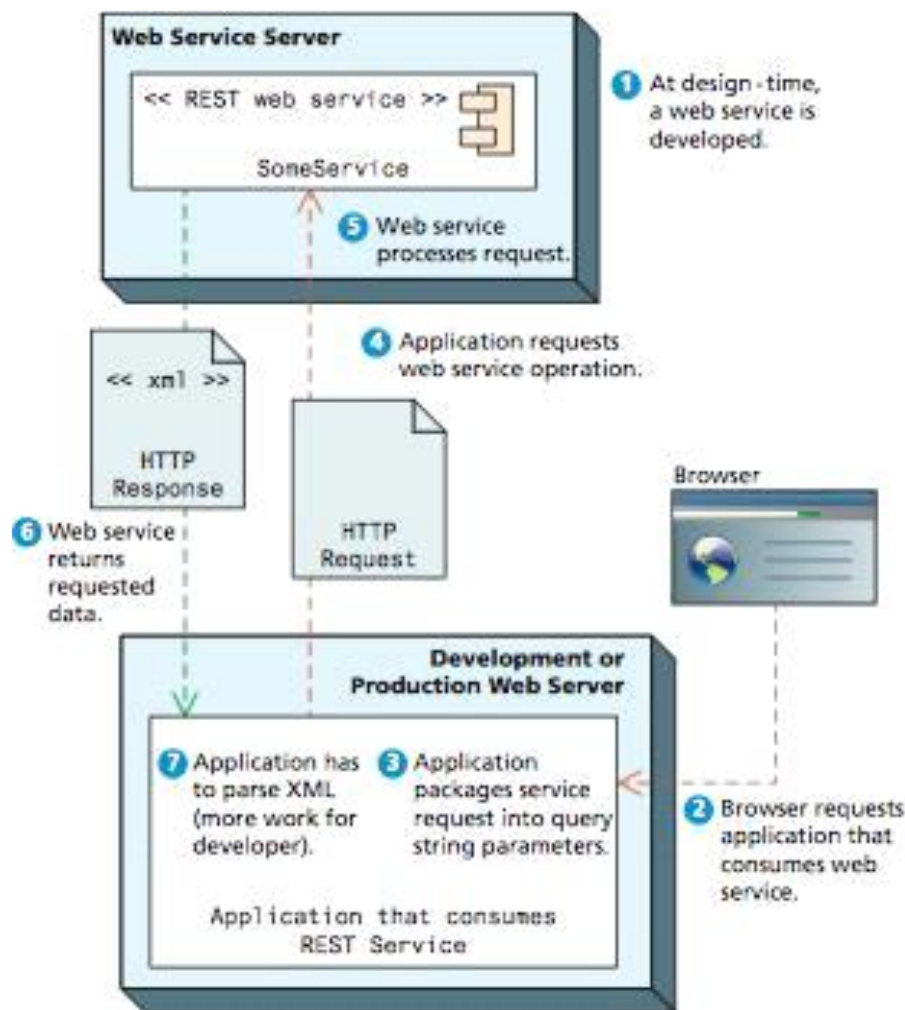
Lastly, the body includes the actual message



Rest web services:

REST service is not a collection of a paradigm or specific rules; it is instead the style architecture for software. Those apps that are designed using this architecture are collectively termed as RESTful web services. Contrasting SOAP, which aims at actions, REST deals majorly on the resources.

It establishes the resources through the use of URL as well as depends on the nature of the transport protocol (like HTTP's: GET, PUT, POST, DELETE, etc.) used to perform the resources. The allocation of resources in REST depends on the URL. It is more like conventions-based application.



Identity. Each web service request must identify who is making the request.

Authentication. Each web service request must provide additional evidence that they are who they say they are.

API Keys

[https://dev.virtualearth.net/REST/v1/Locations?o=json&query=British%20Museum,+Great+Russell+Street,+London,+WC1B+3DG,+UK&key=\[BING API KEY HERE\]](https://dev.virtualearth.net/REST/v1/Locations?o=json&query=British%20Museum,+Great+Russell+Street,+London,+WC1B+3DG,+UK&key=[BING API KEY HERE])

References:

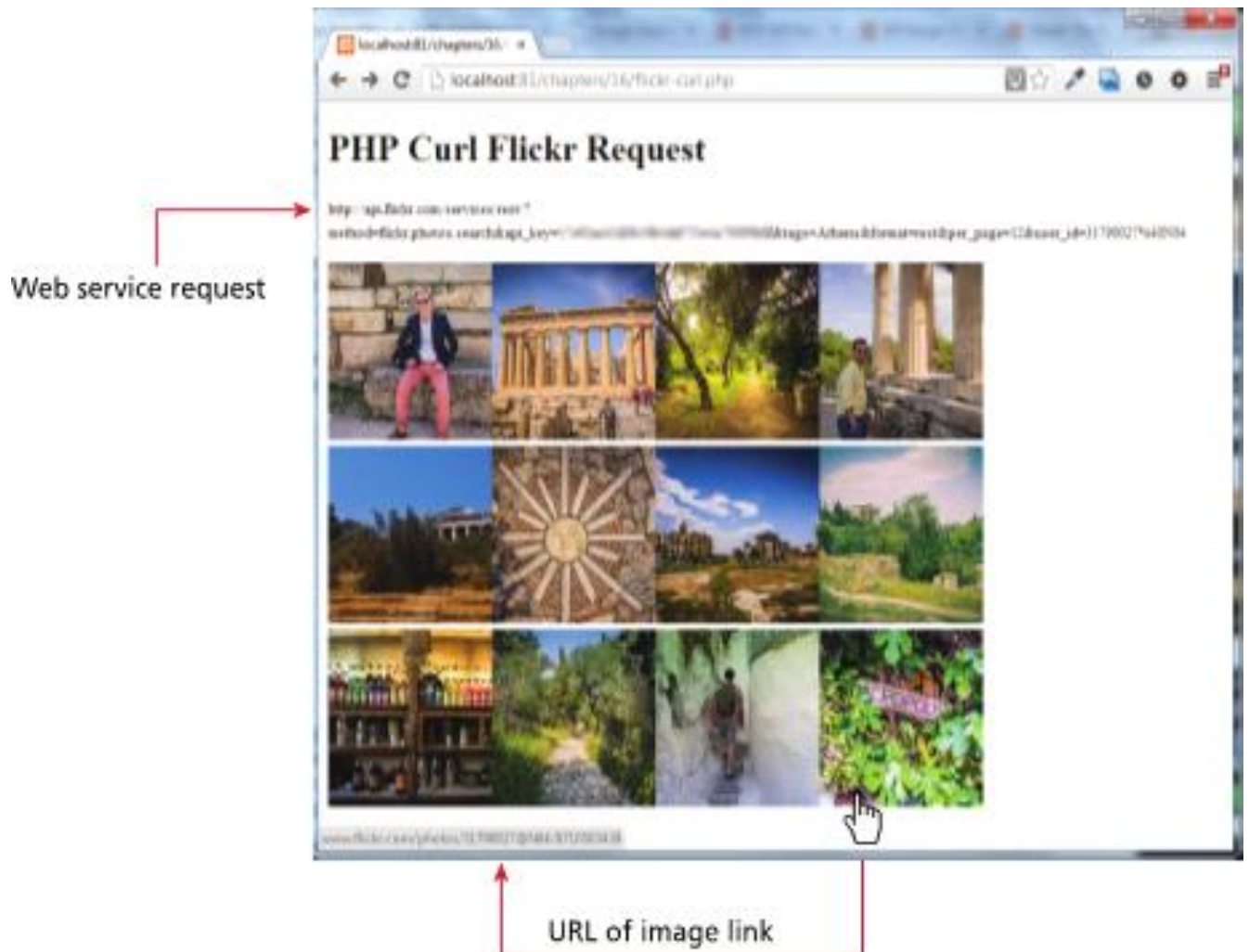
R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

<https://www.w3schools.in/restful-web-services/types-of-web-services>

Module 191

Consuming Web services in PHP:

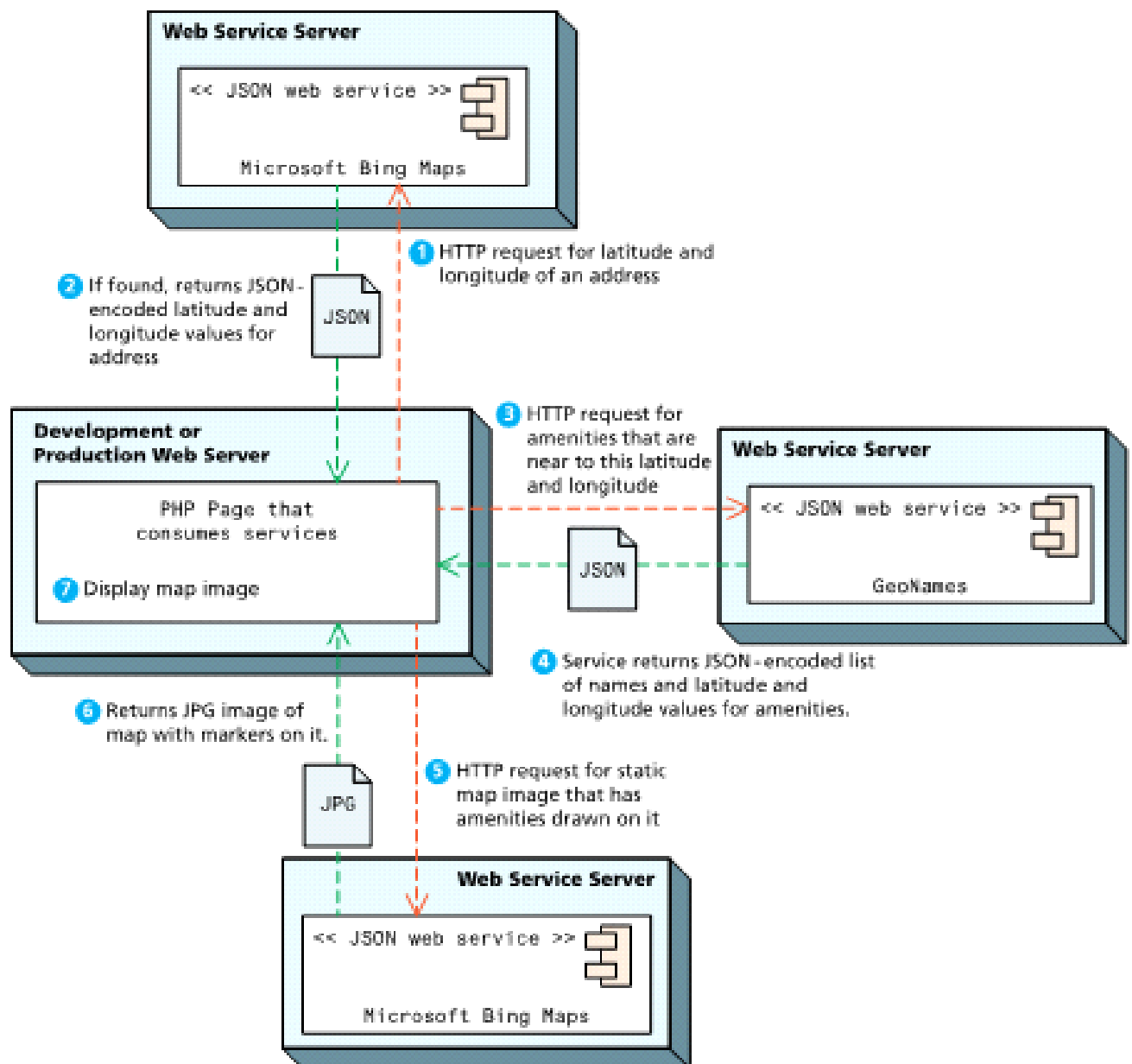
Webservice can be used for different purposes using the URI provided by that webservice. Every webservice can provide multiple URIs, functionalities, resources that one need to consume. For example, you want to use the image sharing website named “flicker” which provide high quality images and API for accessing those images. For this, you need to generate and send a query and results will be fetched based on that query which are actually the images which can be displayed on your own web page. In this way you are displaying images on your own web page that are provided by another source (shown in the following figure).



Consuming JSON based web service:

Example 1:

Let suppose we want to use a map related JSON based webservice. For this purpose, there is need to provide latitude and altitude that particular location that we want to display. If location found, JSON the name of location for encoded latitude and altitude values will be returned. To find the surrounding location of the already provided latitude and altitude values, another webservice will used. The JSON encoded list of names of the surrounding will be returned. In this way, you will get maps related to the provided latitude and altitude and can display in your web page. Similarly, you can interact with multiple webservices and use them in your web page.



Example 2:

URL of service request for static road map image

Zoom level (between 1 and 21)

`http://dev.virtualearth.net/REST/v1/Imagery/Map/Road/43.65163,-79.40853/16?`

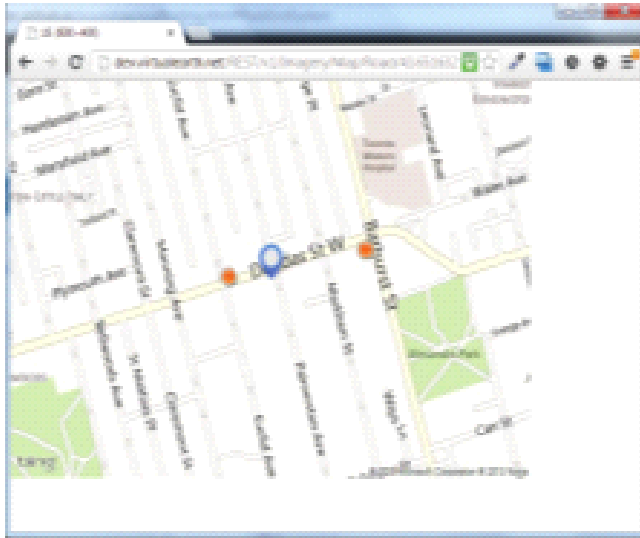
`key=[your api key]`

`&mapSize=600,400` Width and height of map in pixels

`&pp=43.65163,-79.40853;66;` Location of marker (marker 66 = blue circle)

`&pp=43.65208,-79.40618;34;` Location of other markers (amenities) with marker 34 = orange circle

`&pp=43.65166,-79.40958;34;`



References:

R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

<https://www.w3schools.com/js>

Module 192

Creating a JSON Web Service

Consider the URL and format of requests

Tell the browser to expect JSON rather than HTML

Header('Content-Type: application/json');

Use json_encode() to format.

Implement JsonSerializable

Example:

There is Class named “Country” which contains country related information. This class extends “DomainObject” class and implements “JsonSerializable” interface. To generate some response, a jsonSerialize() method will be implemented. In this function key value pairs will be passed in the form of array and will be converted into the Json and those who request these web service will receive these information as shown in below figure.

class Country extends DomainObject implements JsonSerializable

{

/*

This method is called by the json_encode() function that is part of PHP

*/

public function jsonSerialize() {

return ['iso' => \$this->ISO,

'name' => \$this->CountryName,

'value' => \$this->CountryName,

'area' => \$this->Area,

'population' => \$this->Population,

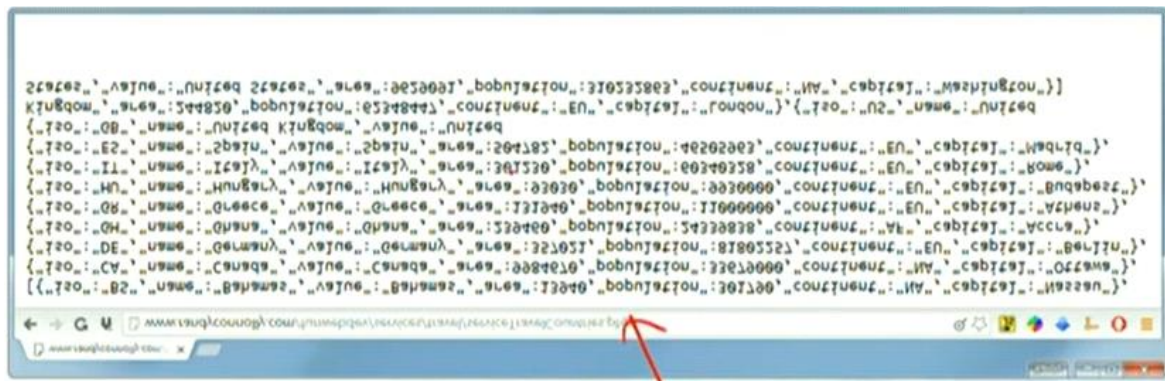
'continent' => \$this->Continent,

'capital' => \$this->Capital

]; }

}

Output:



References:

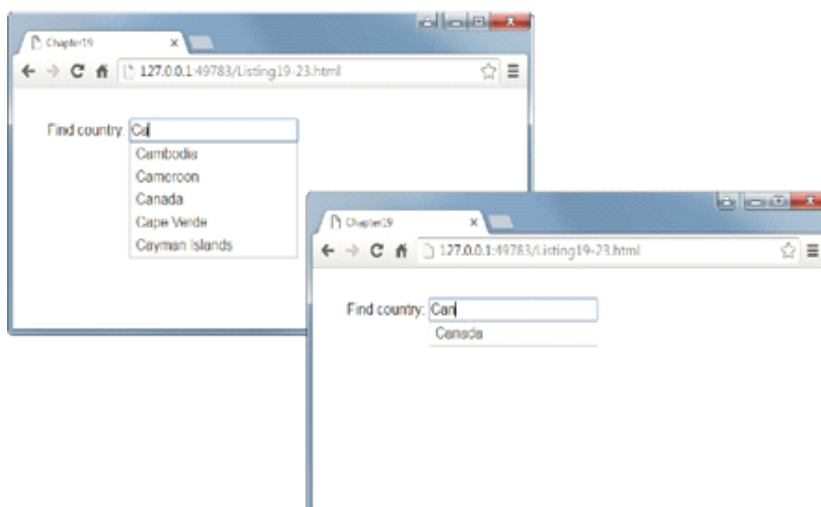
R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

Module 193

Web services using AJAX:

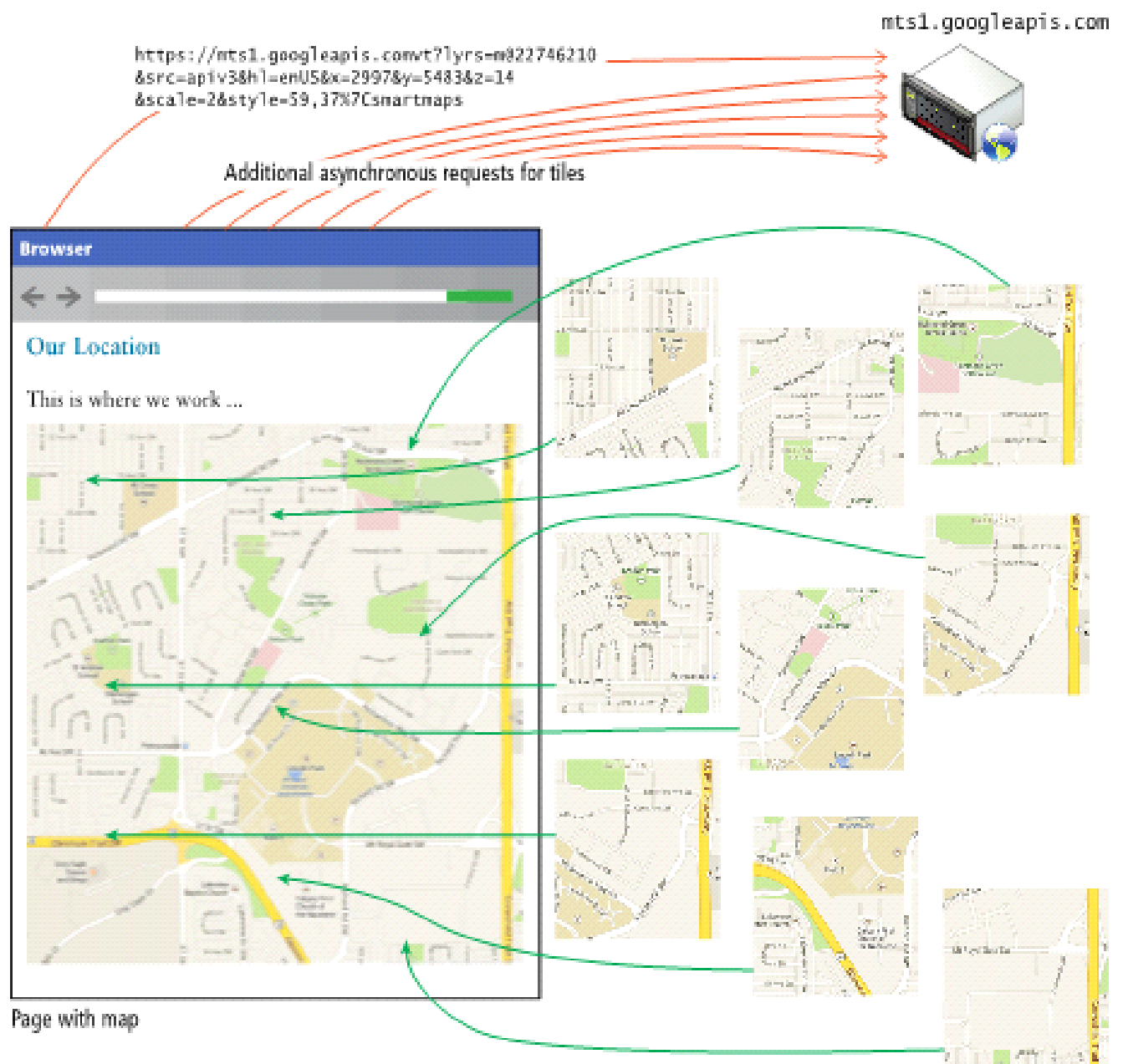
Let there some webservices which can be our own developed or external webservice and we want to use those web service in Javascript using AJAX.

The following figure shows a simple webservice where user type first two letters of a country, a list of countries name having first two letters as “ca” will appear. The user selects the target name or type them.



Similarly, if some want to used google map you need to use the required API. For this, you need to

use the URI of google map API and mention the information you need to access. In this way you get the required data in the form images or in embedded map object form as shown in the following figure.



Code example:

The HTML page used for displaying the map having an element map with required height and width which is 500px and 600px respectively. In the script the URI and required key need to be defined in the script tag. To use the Map service, a javascript function is defined with required parameters like latitude and altitude and the previously defined element "map" with some added properties like alignment, zoom among others.

```
<!doctype html>

<html>

<head>

  <style>

    #map {

      height: 500px;

      width: 600p}

  </style>

  <script

src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY" async>

  </script>

  <script>

    $(function() {

      // hard-coded latitude and longitude for demonstration purposes

      var ourLatLng = {lat: 51.011179 , lng: -114.132866 };

      var ourMap = new  google.maps.Map(document.getElementById('map'),

      {

        center: ourLatLng,

        scrollwheel: false,

        zoom: 14 });});

  </script>

</head>

<body>

  <h2>Our Location</h2>

  <h3>This is where we work ... </h3>

  <div id="map"></div>

</body>

</html>
```

References:

R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

Module 194

JavaScript Frameworks

JavaScript frameworks provide structure for developing JavaScript applications (front-end)

Popular frameworks

Angular

Ember

Vue

React

...

JavaScript Front-End Frameworks

Ember forces developers to adopt a known and well-regarded approach to structuring and implementing a web application. It uses a variant of the MVC pattern

Angular has many similarities to Ember (i.e., models, templates, and routing), and has the added advantage of being partially maintained by Google.

React is developed by Facebook. Unlike Ember and Angular, React is not a complete MVC-like framework; instead it focuses on the view.

JavaScript for Backend (Node.js)

Alternative to LAMP stack

MEAN stack

MongoDB-Express-Angular-Node.js

MERN stack

MongoDB-Express-React-Node.js

References:

R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

Module 195

Introduction to Node.js:

Started in 2009

Nod.js is an open-source server environment that runs JavaScript outside the browser, can be run on every platform (windows, Linux, mac os x among others).

Node.js eliminates the waiting, and simply continues with the next request.

Node.js runs single-threaded, non-blocking, asynchronous programming, which is very memory efficient.

Also used for non-web development, e.g., VS Code

Prominent Nod.js based web applications:

LinkedIn switched from Ruby on Rails to Node.js

- Up to 20 times faster

- Servers reduced from 30 to 3

PayPal switched from Java to Node.js

- Developed in less time, with less developers

- 33% less lines of code

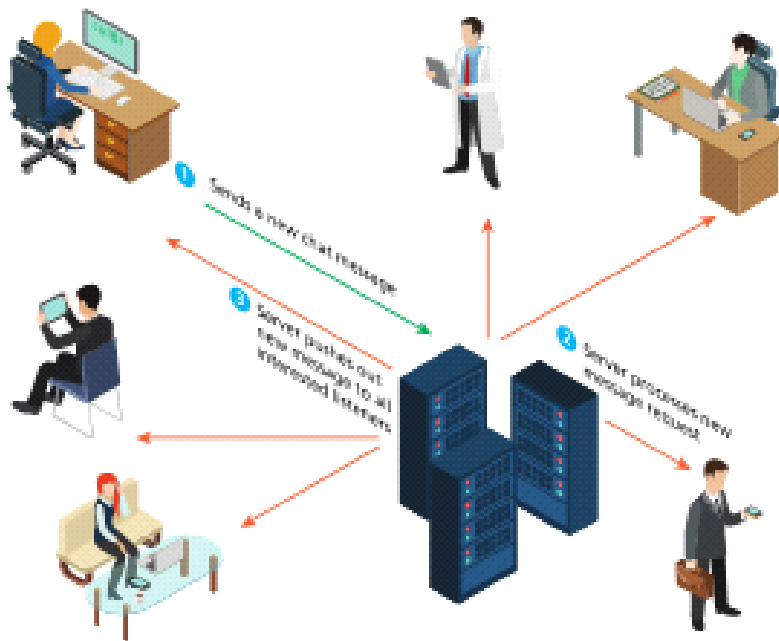
- 40% less files

- Doubled performance (requests/second), with less processors

- Fast response time, 35% less response time

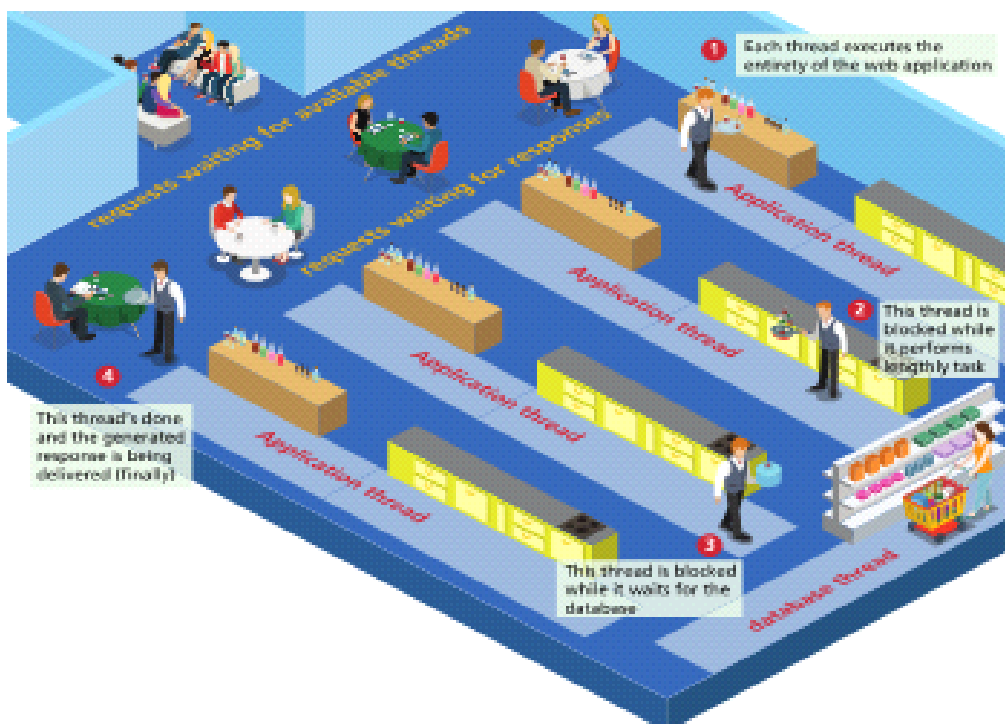
Push based applications:

Node.js basically used for push-based applications. In push-based applications a user generates a message and send to the server. The server processes the received message and then pushes the processed message to all the concern listeners.



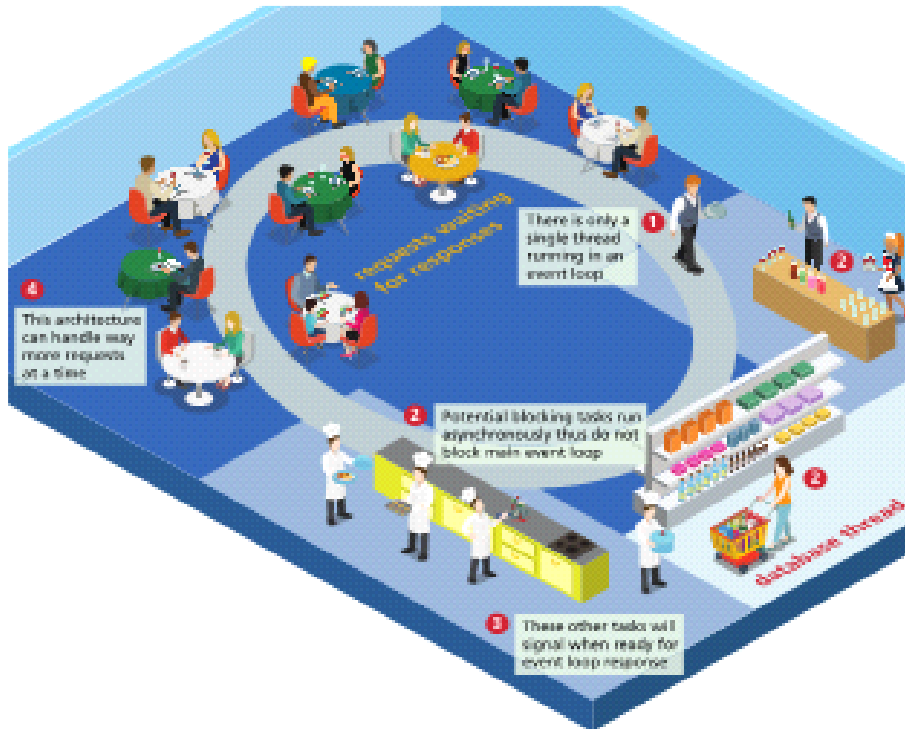
Blocking thread-based architecture:

In traditional server-based web development languages like php, Asp.net among others are based on blocking threaded architecture. In these languages, there are limited number of threads which executes task/process in parallel. Every thread performs a single task at a time means when a single completes their execution only then new task/process can be served. In this way when there more task in the queue to be executed on limited number of threads. In that case tasks in the queue will have to wait until the threads get free by completing execution of already assigned tasks. So, this results in high waiting time users' task in queue.



Node.js Single-Thread Architecture:

The Node.js support single-threaded architecture. In this architecture, tasks are executed in a loop which serve the new tasks in the queue without waiting for the completion of task which is already under process and their execution not completed yet. In this way, Node.js reduce the waiting time of the task in the queue.



Node Package Manager (NPM)

Package manager for JavaScript

Used for installing JavaScript libraries and frameworks

Can handle dependencies

Usage

`npm install` // command to install npm package

`npm install packagename` // To install a particular (by mentioning their name) npm package

`npm install packagename@version` // To install a particular version of npm package

`npm update` // To update the npm package

`npm list` // To show all the packages

References:

R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

<https://www.w3schools.com/js>

Module 196

Simple Node.js program1:

This a simple Node.js program (hello.js) that listen to the request and display a hello message as response to that message.

```
// Load the http module to create an HTTP server

var http = require('http'); // including the required package

// Configure HTTP server to respond with Hello World to all requests

// “function (request, response)” is the call back function that execute
//when process completes.

var server = http.createServer(function (request, response) {

    response.writeHead(200, {"Content-Type": "text/plain"});

    response.write("Hello this is our first node.js application\n");

    response.end();

});

// Listen on port 7000 on localhost

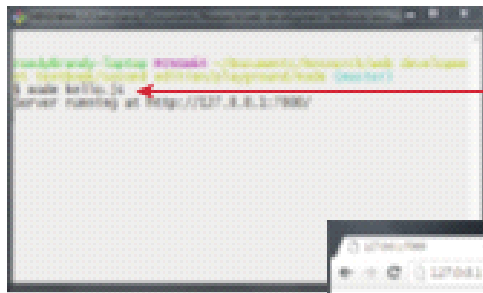
server.listen(7000, "localhost");

// display a message on the terminal

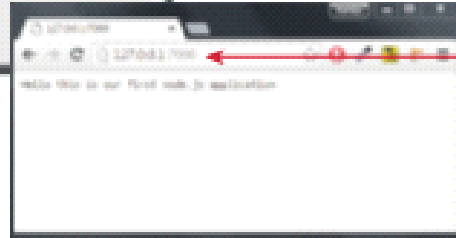
console.log("Server running at http://127.0.0.1:7000/");
```

How to execute the “hello.js” program?

You have to open terminal or command prompt and write the command like: node hello.js, the program will run. The message will be displayed by adding the IP address in the browser as shown in the following figures.



1 First you have to run the program via node command
(You can stop the program via Ctrl-C)



2 Then use browser to request URL and port

Note: every time you make a change to your Node source file, you will have to stop the program and re-run it.

Simple Node.js program 2:

To save a text message into a file

data is written in the file 'message.txt'

When writing is complete, anonymous function is "called back"

```
const fs = require('fs') // include package "fs"
```

```
// file name: "message.txt", call back function: "function (err)"
```

```
fs.writeFile('message.txt', 'Hello World!', function (err) {
```

```
  if (err) { // if error exist show error message or display message
```

```
    // "writing is done"
```

```
    console.error(err);
```

```
    process.exit(1);
```

```
  } else {
```

```
    console.log('Writing is done.');
```

```
  }
```

```
})
```

References:

R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

Module 197

Express.js:

It's a bit difficult to handle web requests using Node.js only, therefore, to resolve this difficulty, a number of libraries can be used to facilitate the Node.js development. One of these libraries is the Express.js library, which is also known as a Minimalist web application framework for Node.js.

Express.js has support for handling:

- Routes

- Requests/Responses

- Views

Example:

```
const express = require('express') // including express package
const app = express() // create express object
const port = 3000 // mention port number

// for get request, get method is called with name and call back function
app.get('/:name', function(req, res) {
  // reading user name using "req.params.name", concatenate with word "Hello", and will send through
  // response object.
  res.send('Hello ' + req.params.name)
})

// To run the server which will listen through port "port"
app.listen(port, function () {
  console.log("http://localhost:." + port)
})
```

References:

R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

Module 198

Node.js with MySQL:

Mostly Node.js is used with NoSQL databases like MongoDB

Can be used with traditional DBMS like mysql

Example:

```
var mysql = require('mysql'); // include "mysql" package
// select query to select all records from books table

var sql = 'SELECT * FROM books';

// create connection using mysql object with parameters like host, user, and password
var con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: ""
});

/*connecting to the database by passing call back function to the connect method. If the connection is
not established, error message will be displayed otherwise the "connected" message will be displayed.
In the next statement the "con.query(...)", another call back function will be called which will either
display an error message or the query result. */
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query(sql, function (err, result)
  {
    if (err) throw err;
    console.log("Result: " + result);
  });
});
```

References:

R. Connolly and R. Hoar, Fundamentals of Web Development, 2nd ed., Pearson, 2017.

Module 199

Background:

There are many JavaScript's front-end frameworks that exist that we use to develop the interfaces. The main idea behind it is to generate interfaces by using JavaScript or HTML. We can generate interfaces through HTML and then JavaScript will be used to fill the data. The question arises here how can we get data from the server-side. The Client-side sends the request through the browser to the server and renders the fetched data from the server at the appropriate place. There are many JavaScript front-end frameworks available but we are going to discuss the relatively early days framework which is Angular.

Introduction:

Angular is a popular browser-based and open-source framework. Although it is open source, many of its lead developers are employees of Google. It follows the MVC framework. It also stands for "A" in the terms of MEAN Stack but it does not mean that you have to use angular whenever there is MEAN in full-stack. As we have discussed in the previous lecture that there are many other options which can be used like react, view or ember can be used. It is independent of other components of Stack and can be used without any of them.

Angular 1 was written in JavaScript. Angular 2 is not an upgrade of angular 1 but it is completely rewritten. Angular 2 required developers to switch from JavaScript to TypeScript which is a superset of JavaScript. In TypeScript, we can statically assign the types of data variables but this is not possible in JavaScript because JS is managing the types of data variables at run time. Angular not restrict the developer to use the TypeScript only but it also allows them to write in Dart whose syntax is like C language. AngularJS uses JavaScript and we are going to discuss AngularJS because it uses the regular JavaScript. It has a large user base of committed developers.

Why Angular JS?

One of the reasons for the developer's interest in AngularJS was that it is well suited in fact angular and front-end frameworks are well suited for single-page applications. Single-page applications are web applications that typically consist of only a single page that is updated dynamically. It means all the functionality is available on a single page and record components fetched the required data from the server and fill it in appropriate places. For example, when you open Gmail, the interface has been generated. A new page is not going to open when you click on the inbox. It only shows the list of emails.

Example:

Here is a very simple angular application. Angular can be included by saving in the local system or you can include it through the content delivery network.

```
<html ng-app>
```

//Here we are telling that HTML document is basically an angular app. ng stands for angular. ng-app is a directive for designating the root AngularJS element.

```
<head>
```

```
<title> Chapter 20 </title>
```

```
<script src=https://code.angularjs.org/1.5.0/angular.min.js>
```

```
</script>
```

```
</head>
```

```
<body>
```

Enter your name: <input type="text" ng-model="name" />

// Text field is created here and angular model has been defined for this text field and "name" is the given name of the model. The model could be used for giving data. Ng-model="name" is a directive for saving the field value in the model.

```
<p> You entered: {{ name }} </p>
```

//we are going to access the model here. Model name should be written in the double curly brackets and it displays the data stored in that model. {{ name }} is a data binding expression.

```
<hr>
```

Enter your city: <input type="text" ng-model="city" />

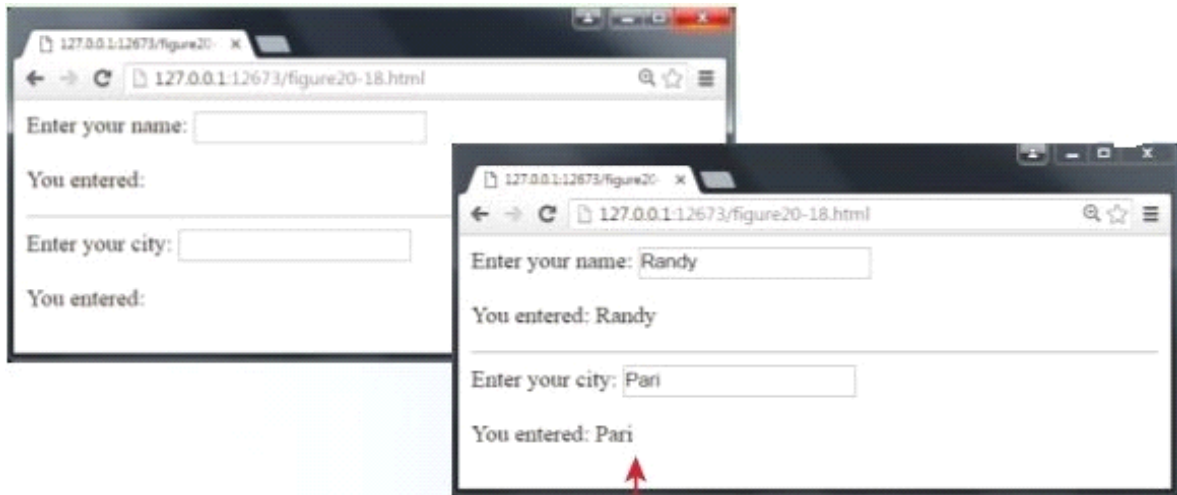
```
<p> You entered: {{ city }} </p>
```

//One more angular model has been defined and the name of the model is city. Model can be accessed by using its name in the double curly bracket.

```
</body>
```

```
</html>
```

You have created the models on the input field and linked them to the paragraph tag. Paragraph tag will display the value stored in the model. As soon as you are going to write anything in the input field, it will display in paragraph.



Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

Let's extend this example by adding in an AngularJS controller. Angular is MVC framework and it has three components i.e., Model, View, and Controller. Controller is basically managing the request. Let's write the code for simple controller, fetch the data from controller and display it.

Adding a controller

```
<html ng-app="demo">
```

```
//Now this directive is specifying the module used in the application.
```

```
<body ng-controller="myController">
```

```
//This element is going to use a controller (myController) to get its data.
```

```
<div id="search">
```

```
    City Search: <input type="text" ng-model="search" /->
```

```
//Text field has been generated and search model has been defined for this text field which
saves the user's input in a model property named search
```

```
</div>
```

```
<table>
```

```
<tr ng-repeat="city in cities | filter:search | orderBy: 'name' ">
```

```
//
```

A directive to loop through a collection named cities (which is defined in the controller). We have cities array or the collection. This repeat is going to iterate all cities present in cities collection one by one. This is just like the for each loop. Then we are going to filter the data on a search basis. The value of search we are getting from the text field. Then we are going to sort the data in order on the basis of cities name. It uses filters to alter how this element works. In the example, the filter filter and the orderBy filter are used to modify how the ng-repeat works. Here the search refers to data items in the model.

```

        <td> {{ city.name }} </td>

        <td> {{ city.country }} </td>

//It bind the data to values in the collection.

    </tr>

</table>

</body>

</html>

```

How controller works?

Controller has a collection of cities. We are going to fetch cities from the controller and filter those values and display them in the form of a table.

```
Var myapp = angular.module('demo', []);
```

//First we are going to create the app and then create the module using angular.module. A module is an AngularJS container for the different components used in the application.

```
Myapp.controller('myController, function ($scope)
```

// We can create controller through the app. Add a controller to the module named myController. The \$scope variable is passed or injected into the controller by AngularJS. \$scope is going to use the access the variable stored in the controller.

```

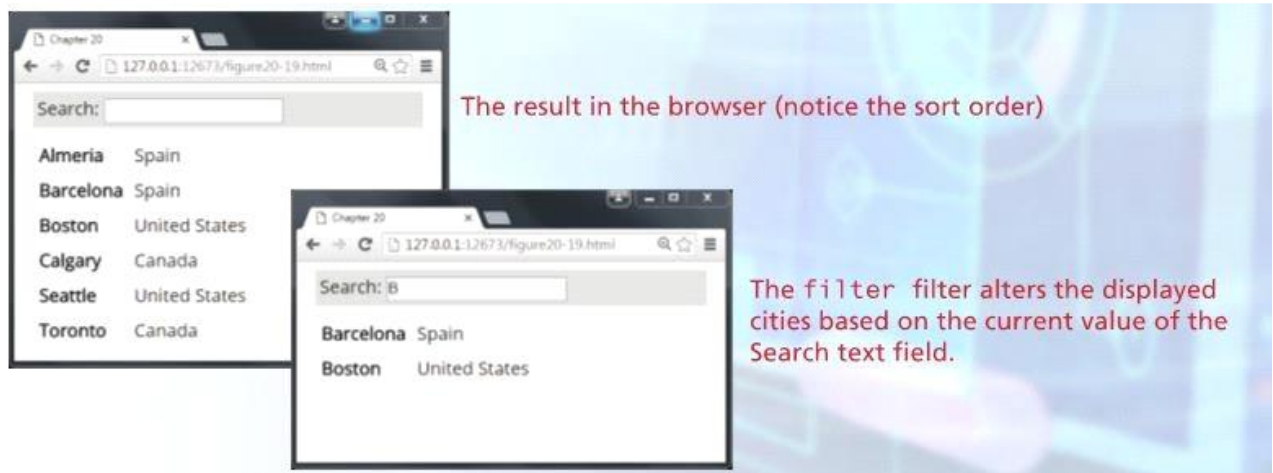
{

$scope.cities = [{name: 'Calgary', country: 'Canada'},
                  {name: 'Toronto', country: 'Canada'},
                  {name: 'Boston', country: 'United States'},
                  {name: 'Seattle', country: 'United States'},
                  {name: 'Almeria', country: 'Spain'},
                  {name: 'Barcelona', country: 'Spain'}];

```

//\$scope.cities variable is used to store the model (data). Here we are defining an array of object literal named cities. The body tag of controller access the all data available in \$scope.cities.

```
});
```



Connolly & Hoar, "Fundamentals of Web Development" - 2nd Ed. Pearson Ed. (2018)

Getting Data from a Web Service

```
var myapp = angular.module('demo',[]);
```

```
//First create the app and create the module using angular.module.
```

```
// notice that our callback function is passed the $http parameter.
```

```
// parameters are injected by the AngularJS system "behind-the-scenes"
```

```
myapp.controller('myController', function ($scope, $http) {
```

```
// Create the controller through app and pass the anonymous function. Angular injects two parameters in this function. One parameter is scope that stores the data and the other parameter is HTTP which is going to deal with the HTTPS requests and responses.
```

```
var url = 'http://www.randyconnolly.com/funwebdev/services/travel/countries.php?continent=EU;
```

```
// we are going to specify the URLs in the function and fetch the data using jQuery from that URL. It retrieves country data from web services.
```

```
$http.get(url).then(function (response) {
```

```
$scope.countries = response.data
```

```
// get will get the data from the URL and after the completion of get command, it will execute the function. Data fetched from the web services if going to store in scope.countries variable. Use the body element to display the data.
```

```
});
```

```
});
```

Module 200

Web Sockets:

As we have discussed that we can fetch the data from the server-side while staying at the browser through JavaScript by using ajax.

Ajax is HTTP-based and HTTP is a half-duplex protocol which means data can flow back and forth between two devices, but not simultaneously. The sender can send data at one time and the receiver can receive data at another time, they both can send and receive the data but only one can request will be performed at a time. Both requests cannot be performed simultaneously.

We have one other protocol which is Web Sockets and it is a full-duplex communication protocol. Enables interaction between a web browser and web server. It allows an application program to send and receive information simultaneously. We can send the request and fetch the data at the same time.

Let's see through examples. We will send the data from the server and get the data from the client on the other side. We are going to use node.js to implement the server-side but you can use any other scripting language too.

The idea behind is that the server listens to the requests of web sockets and then responds accordingly. We are going to develop an application that will return the same message to the client which we receive from the client and concatenate the string "message received" with the message.

Server-Side Example (Node.js)

```
const express = require('express');

const app = express();

// We need express and app will be created on its basis.

const http = require('http');

const server = http.createServer(app);

// We need HTTP for request response and will create server through it.

const {server} = require("socket.io");

//We need socket.io library. It is used for implementing the sockets.

const io = new Server(server);

// We will create server of this library and create the object io on the basis of this server. Then the HTTP server will be passed to the socket server.

// This object has different event handler on it.

io.on ('connection', (socket) => {

    console.log('A user connected');});
```

//The first event handler is to on the connection. When the connection of the web socket will establish it will print the message (user connected). As you can see, we have the call-back function. We can give the call back function by giving the arrow notation => and you can also write it like function(socket).

```
io.on ('message', function(message) => {  
    console.log('message from client: ' + message);  
    io.emit('Message received ' + message)});
```

//Second event is used to define the message. As soon as a message will be received, the call back function will be called and inject the message as a parameter and display it on the console. This message will be displayed on the server-side. To display this message on the client-side we are going to use the io.emit event handler.

```
server.listen(7000, () => {  
    console.log('localhost:7000');});
```

Client-Side Example (JavaScript)

//Now we are going to communicate with the server through the client-side. Create the object of the web socket and pass the IP address of the server. Usually, WSS replaces the HTTP protocol. WS is WebSocket and WSS is web socket secure connection.

```
var client = new WebSocket ("http://localhost:7000");
```

//Then define the event handler. When the connection will open send the message to the server.

//sending message on socket open

```
client.onopen = function() {  
    client.send ("Message sent to the server");  
}
```

// This message will be sent to the server and server sends the message back. To receive the message, onmessage event handler will be used.

```
client.onmessage = function (event) {  
    console.log(event.data);}
```

//Now end the communication with the server and don't want to send and receive the message, close connection.

```
client.close();
```

Module 201

Web Server Hosting

When we finished the development of the website, the next step is to host the website. Website hosting means we have to copy the website on a web server so that the users can visit our website through the web server.

The hosting of the website is very important and crucial because users will be interacting with your website through the server (host). Users will consider your website slow, unresponsive, and unavailable if the hosting is poor. It is not necessary to buy expensive hosting. The solution is to understand the different types of hosting and choose the best option according to your website.

The three broad categories of web hosting are shared hosting, collocated hosting, and dedicated hosting. Within each of these categories, there are subcategories, which altogether provide you with more than enough choices to make a selection that works for your situation.

Shared Hosting:

The most common and most cheap hosting is shared web hosting. This hosting will provide the space on the server for your website that will host many websites on a single machine. All those websites use the resources of the same machine. At times, there might be hundreds or thousands of websites hosted on a single machine. This hosting is suitable especially for small websites or websites that have less traffic. Shared web hosting is further divided into two categories: Simple shared hosting and virtualized shared hosting.

Simple shared hosting

In simple shared hosting, all the websites are deployed on a single machine and it usually has a single Database management system. All the websites use the resources i.e., CPU, RAM, bandwidth, or hard disk space of the single machine. In simple hosting clients receive access to a webserver but they cannot increase their privileges to any part of the machine.

Lack of control and poor performance can be disadvantages of simple shared hosting. Static HTML sites or default WordPress installations don't have the problems of control. But if you want to install some software, shared hosts do not permit it. You cannot use a particular version of the software, but rather use what everybody is using.

Poor performance is a more common problem with shared hosting. As all the websites are using a single machine and sharing the same resources, performance problems will arise. Shared servers work fine if they have a small number of websites hosted. Some shared servers host thousands of websites on a single machine and which results in poor performance.

Virtualized shared hosting

We have one other mechanism to host the website virtualized shared hosting. In this hosting, we create virtual machines on a single machine. Each virtual machine is assigned its own resources from the resources of a single machine. In virtualized hosting, each website will have its own virtualized machine.

The advantage of virtualized shared hosting is that each website has its own virtual machine, they are given the freedom to install or configure the software. Disadvantage+ is still the same in the end all the resources are shared.

Dedicated Hosting:

In dedicated hosting, a physical machine is rented or allocated to you, and that web hosting service is going to manage that machine. You can install software of your choice on that machine and all the resources of the machine are going to be used by your website. It has an advantage over shared hosting is that you have given the complete physical machine to control and you don't have to handle the issues that arise due to shared resources.

Hardware is normally standardized by the web hosting service provider and the host is going to take care of hardware issues. The disadvantage of dedicated hosting is the lack of control over the hardware and a restriction on accessing the hardware. It might not be suitable for your particular needs.

Collocated Hosting:

Collocated hosting is almost like dedicated hosting, but you can purchase, build, and manage the machine yourself. In collocation, you can use the infrastructure of a web hosting service but you can manage the hardware by yourself, unlike the dedicated hosting. We have physical access to the computer. Web hosting services provide the services like energy requirements, internet connection, electricity, security requirements, fire suppression system, climate control, or air conditioning requirements.

The advantage of collocated hosting goes beyond a dedicated server with not only full control over the operating system, software version, firewalls, and policies but also the physical machine.

The disadvantage is that you must control everything yourself, with little or no support from a third party. These web hosting service providers are costly and they are not going to provide services for the hardware issues.

In-House Hosting:

In in-house hosting, you have to manage the webserver yourself entirely in-house. You will use your personal machine to host the website and your website is going to use all the resources of that machine.

For example, if you are using a DSL connection and you asked your service provider to assign you the static IP address that is not going to change whenever you connect to the internet. You can enter it in the domain name server and host your website here. It has an advantage in terms of control. It may seem appealing to host your website by using your own home-based machine but there are many issues that can arise. You have to manage all the requirements by yourself. Most of the time, the bandwidth of home-based resources will be less, it has limited resources, and you have to keep your machine online 24/7 to be accessible. This option is good for small websites and for websites where a 24/7 online presence is not important.

Cloud Hosting:

The recent hosting mechanism is cloud-based hosting. In cloud-based hosting, your website is hosted on multiple machines and a third party manages it. For example, Amazon and many other companies provide the services to run your website on multiple machines.

The website has thousands or millions of users, you can manage all its request on one machine and that's why you have to distribute its request on multiple machines to keep your website scalable.

Module 202

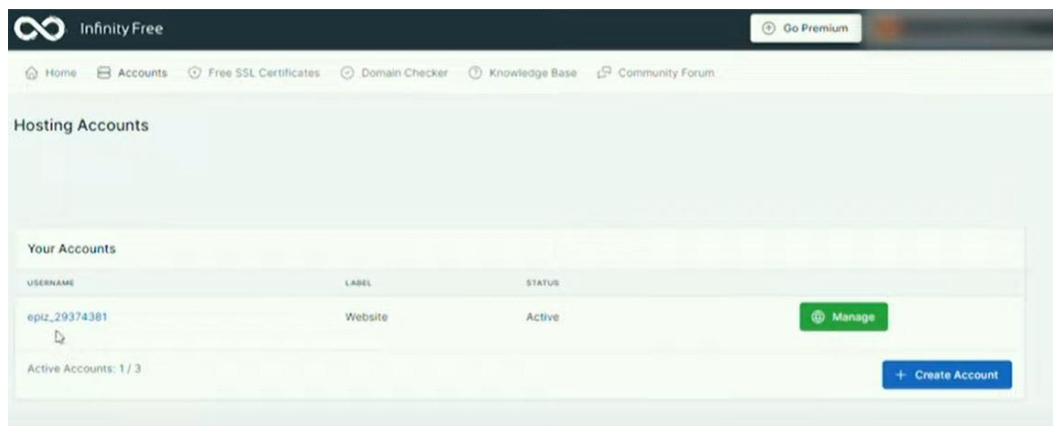
There are different options to host the website. Let's see how to use a free web hosting service to host the website. This is just to get an idea to show how the website can be hosted. But you cannot host a professional website on a free web hosting service because it has many limitations. It will show the ads and free website hosting services cannot give the support of secure connections like HTTPS or SSL. If the website is dealing with users' data, then the data will not be transferred securely and will be exposed. So only use free hosting service for demonstration and learning purposes.

Let's go through one example to demonstrate how the website can be hosted on web hosting.

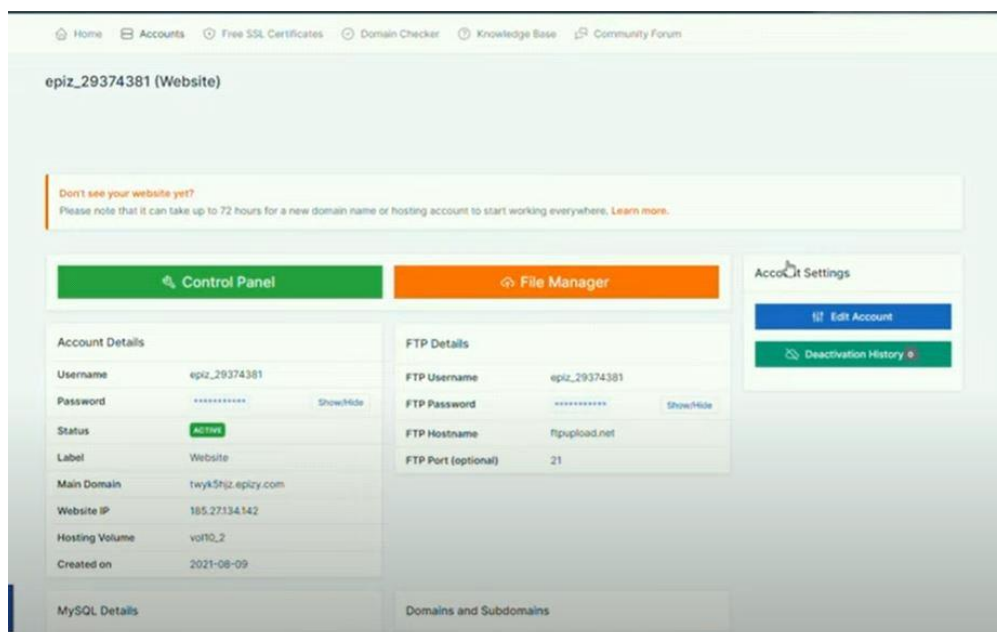
We are using a free web hosting service (infinityfree). This is free website hosting and it has different limitations.



You will sign up for it and after signing up, you are going to log in to your account. When you log in the system will display the screen, which shows the active accounts, and right now it will show the 1 active account out of 3 accounts. We have created the website already because it can take 2-3 days to enter your website in DNS. A username will be assigned to your website and you can manage your website.

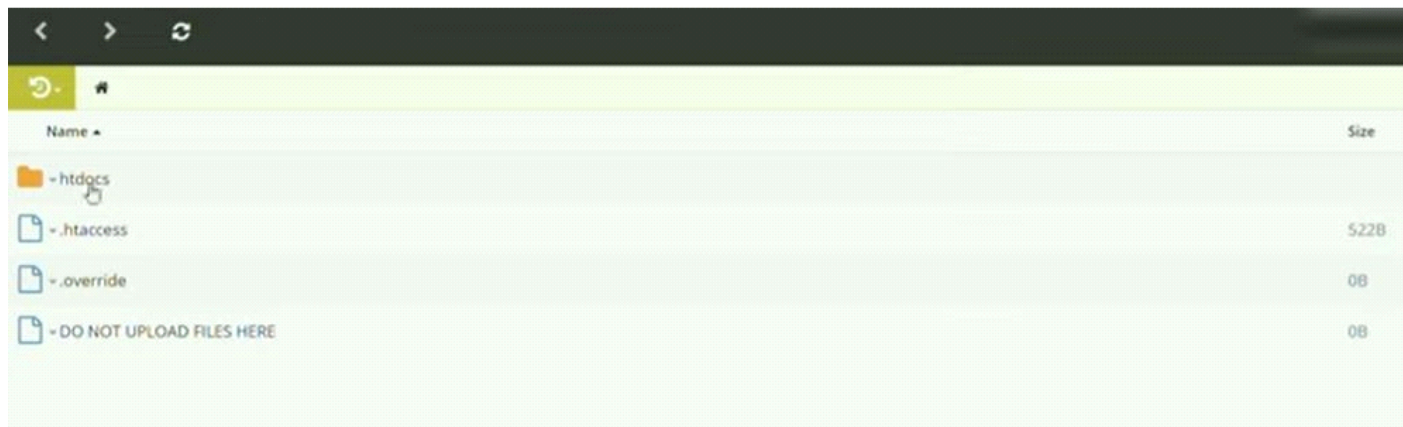
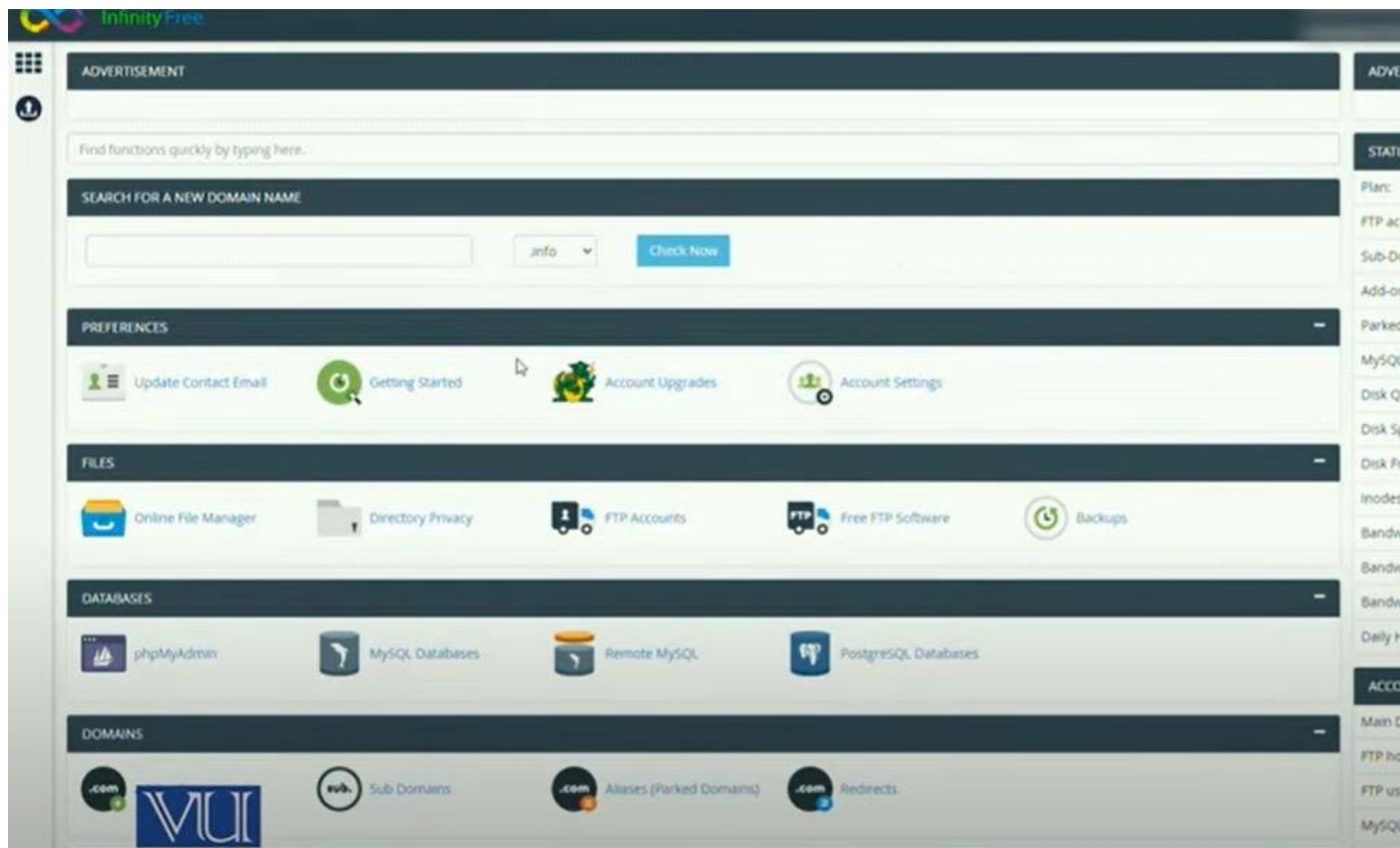


You can see the control panel information and see the information (username and password) related to establishing a database connection.



Open the control panel and you can access different tools here from which you can upload the files to a web server. You can perform many other activities through the control panel. InfinityFree panel is a C panel but this C panel is a very basic version and many features are disabled because this is a free service. When you pay for some good web hosting service and buy paid account then it has many additional options.

In this control panel, you have a preferences panel in which you can perform different settings for your website. There are different options to upload the file. You can upload the file through an online file manager or you can use the option of FTP client. If you want to use the FTP client option then you have to go for File Zilla which is a free and open client for accessing the FTP sites. It has a database-related section and you can create the different databases. It also has the support of phpMyAdmin through which you can manage the database.



Let's upload the file to the webserver and see how it will be displayed on the website. Click on the file manager. It will show the basic interface to upload the file. We have httpd docs folder here and you can upload your files in this folder. These files will be hosted on the server and will be displayed on your website. At the bottom of the page, it has an upload file option choose the file from local storage and upload it. After uploading the file be accessible on the URL which you used while creating the website. You can see the file on your website. We can also change the file name with the option to rename it in the hosting service. You can upload your website, PHP scripts, establish a database connection, enter the data, or retrieve the data.

I would suggest you develop the home page or the personal website and upload the dynamic content.

It is obvious that when you are going to use the free hosting service, you will face limitations. You cannot install the Laravel in infinityfree but you can deal with the PHP scripts and establish a database connection.

Module 203

Domain and Name Server Administration

Introduction:

Internet functioning is based on the IP Addresses. Each website has IP address and we can access the website through that IP address. It is very difficult to remember the IP addresses because they keep changing. We have domain name system to access the website in which we link the IP address to the URL and all this mechanism done through the domain name system.

We can register the URL of our website through Domain name system. We have different domain name registrar that register our website. We have some popular registrars i.e., GoDaddy, TuCows, Network solutions and many more.

Most of the time registrar also provide the hosting options with the domain name registration although they are two different things. You can host your website on some other web server and use other domain name registrar to register the website. Most of the time, web hosting provider also assign the domain name along with hosting the website.

Module 204

Search Engines

Search engine has an important role in the world wide web through which users search the websites. Before getting into the details of search engines, let's look into what happened before the search engines.

In the days before search engines, there was no capacity to search the entire world wide web. There were very few websites when the world wide web was created and this number increased to a thousand or more than ten thousand websites. It was not easy to keep the records of this large number and browse them. We had directories to browse. The first directory was “vlib” which was created by Tim Berners-Lee. Websites were listed down in vlib and you can access these websites. The most well-known directory is the Yahoo directory, which listed down and categorized the websites. For example, if you want to visit the university website, you have to select the education category and then the university category and here you find the websites of the universities.

There was one more option which is an open directory project. Directory websites were good but they lacked the ability to search quickly. “dmoz” is the open directory project in which websites are listed down as per categories. Users can share the website information through email, message boards, and chats to inform each other about the websites.

Now, if you have to search the specific websites let's say you want a website related to automobiles, you can visit only those websites which are listed in the directory and those about them you know already.

Search engines solve the problem using the web crawler. Web crawler is the first component of the search engine. Search engines looked for the available websites using a web crawler and then indexed them. If someone searches for the website of automobiles, the search engine will look for the websites that are related to automobiles or have the term automobiles in their content and then show the result.

Search Engine Overview

Components of search engine

Search engine has multiple components but the main component is a web crawler. Web crawl crawls the existing websites. There are the anchor tags in the website through which we users move from website to website. It downloads the page, identifies the anchor tags, and adds them to the database.

Crawlers are also known as web spiders, worms, or robots. Around 50% of the websites can be accessed through this component.

URL submission is another mechanism to search the website. A new website has been uploaded and it doesn't have anchor text right now. The other websites cannot point to the new website. Search engines cannot find the existence of the website. Here comes the URL submission where you add the URL of your website which tells the search engine about the existence of the website. Now, it should be made part of search engine index and users can access them.

Indexes are also the mechanism to search the website. Search engine saves the content of the website and creates the index on it. These indexes will be stored in the database and find the keywords in the content of the website. After that, queries can be executed quickly on those indexes to recover the complete records.

The query server handles requests from the users for a particular query. Search engine shows the results of the query. Now, it is very important to rank the result of that query. For example, there are more than 10 million websites on automobiles, which of these will be the top 10 from them. Search engines solve this problem by using ranking algorithms.

Module 205

Page Rank

Page rank is an algorithm that is used for ranking websites. It was published by Google's founder. This algorithm is basic for search engine ranking. In fact, search engines use many other factors for ranking websites as well but the most important factor is the page rank algorithm. It works on the graphs that model all the websites and their connection with other websites. The graph looks at the hyperlinks between websites. Let's look at the example and create the graph for the example. We have four websites i.e., HEC, VU, QAU, and MOE.

We have four websites so we will have 4 vertices in the graph. Then edges will be created for the vertices on the basis of connection with other websites. One website has a link to another website. For

example, “VU and QAU” websites have the link of “HEC” and it will be shown in a graph by drawing the edge from VU towards HEC and from QAU towards HEC. The ‘HEC” website has the links for the links VU, QAU, and MOE, and connections will be created from “HEC” to these three websites. The following diagram shows the graph of the example. Each website becomes a vertex and each connection to another website becomes the edge.

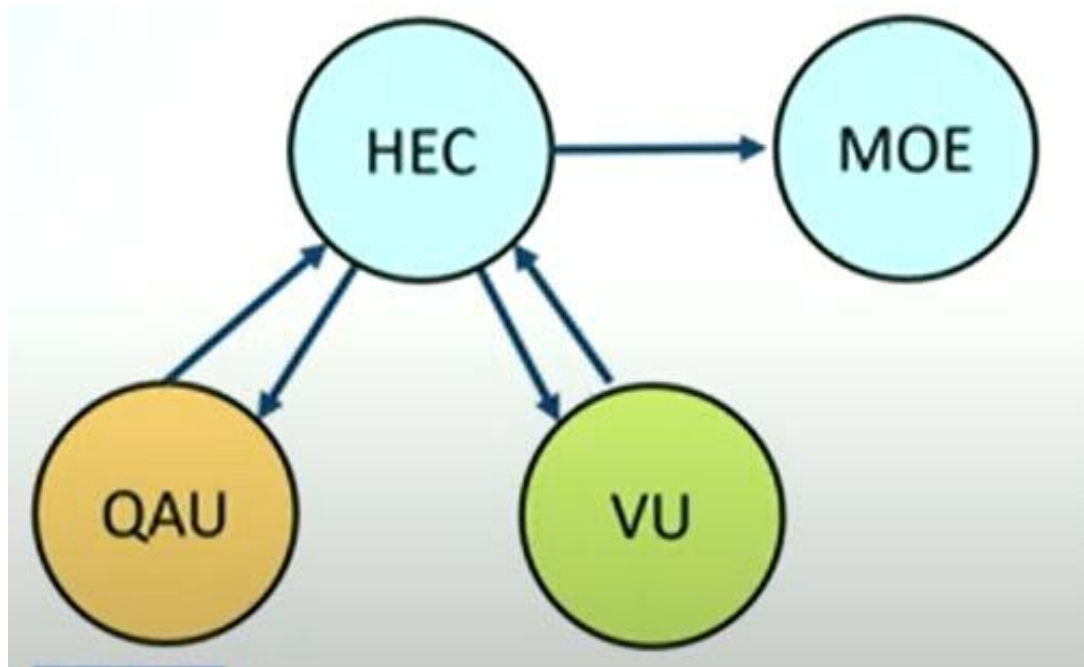


Figure 6: Example

How the page rank algorithm interprets the graph? The main idea is that authentic websites are pointed by authentic websites. Now the question arises of how to identify the authentic website and which authentic website is pointing to which websites? We can see from the graph that one website is pointing to which website but how to calculate the authenticity and importance. In the following diagram, you can see the different websites pointing to other websites. Now look at the website named as 1 and is linked by the one website. Look at the website named 2 which is also linked by the one website. But both websites have different page ranks although they have the same number of websites pointing to it. Page rank is represented by the size of the vertex for this particular website. Website 2 is linked by a yellow vertex and its size is larger than the website pointing towards website 1. The bigger size of the vertex represents more importance. Website 2 has a larger page rank than website 1 because it is linked by the important vertex.



Figure 7: Example

How it works?

We can calculate the authenticity and importance through iterations. In iteration 0, we assign the same page rank to all websites i.e., HEC, VU, QAU, and MOE. Each vertex is assigned the ranking of $1/n$. We have four websites for the current example so "n" will be 4 and the page rank for all websites will be $1/4$.

In iteration 1 each vertex will be visited and its page rank will be distributed among its outgoing vertices. HEC vertex is in consideration, its page rank is $1/4$ and it has three outgoing links towards VU, QAU, and MOE. This page ranking of $1/4$ will be distributed among these three vertices. Each outgoing vertex of HEC will have a ranking of $1/12$. Now consider the QAU vertex, it has the page rank of $1/4$ and it has one outgoing link towards HEC. Its page rank of $1/4$ is going to be distributed to one outgoing link and HEC will have the page rank of $1/4$ from QAU. MOE vertex has no outgoing link and its page rank will not be distributed.

The sum of all the page ranks should be 1. In the first iteration, the HEC vertex has a ranking of $1/4$ assigned from QAU and $1/4$ assigned from VU, and page rank will be $1/2$ for HEC. Other vertices have a $1/12$ page rank. This process is a repetitive process. In this next iteration, consider HEC which has $1/2$ page ranking. This rank will be distributed among 3 vertices and each vertex will be awarded $1/6$ page rank from HEC. This process will be repeated and page rank will stop changing at one point. We cannot perform more iterations because page ranks have been fixed. They won't change anymore and we call it page rank convergence. The page rank algorithm runs when either page ranks are

supposed to be converged or a certain number of iterations will be completed. For example, we would find page rank for 1000 iterations and after 1000 iterations the score will be fixed.

The basic version of page rank and the problem is that if the vertex has no outgoing link, then ultimately the page rank of that website will be 1 and 0 page rank for other websites. This problem will be solved using a scaled version of page rank. In the scaled version in each iteration, each vertex has been assigned the page rank. Some page ranks are assigned on the basis of calculation and other page ranks are equally distributed among the vertices. This version does not create the sync holes and page rank cannot be accumulated at one point. This is the one mechanism to rank the websites and it is not easy to manipulate them. We can create the links to other websites on our websites but it would not be easy to add the link of your website to the important or authentic websites.