CS521

WEB SYSTEMS AND TECHNOLOGIES

SLIDES WEEK [7 TO 16]

MADE BY MAH JABEEN

0321 2711298
mahjabeen97869@gmail.com

WEEK 7

TOPIC 55 TO 63

Web Systems and Technologies

- Modifying DOM Elements
- Creating DOM elements
- Event Handling
- Capturing and Bubbling
- Event Types
- Form Validation; Client Side Input Validation
- Introduction to jQuery
- Query Selectors
- jQuery Manipulators

Modifying the DOM

Changing an Element's Style

```
<style>
    .box {
        margin: 2em; padding: 0;
        border: solid 1pt black;
    }
   .yellowish { background-color: #EFE63F; }
   .hide { display: none; }
</style>
<main>
   <div class="box">
    ...
   </div>
</main>
```

```
var node = document.querySelector("main div");
```

1. `node.className = "yellowish";` — This replaces the existing class specification with this one. Thus the no longer has the box class

   Equivalent to: `<div class="yellowish">`

2. `node.classList.remove("yellowish");`
   `node.classList.add("box");` — Removes the specified class specification and adds the box class

   Equivalent to: `<div class="">` `<div class="box">`

3. `node.classList.add("yellowish");` — Adds a new class to the existing class specification

   Equivalent to: `<div class="box yellowish">`

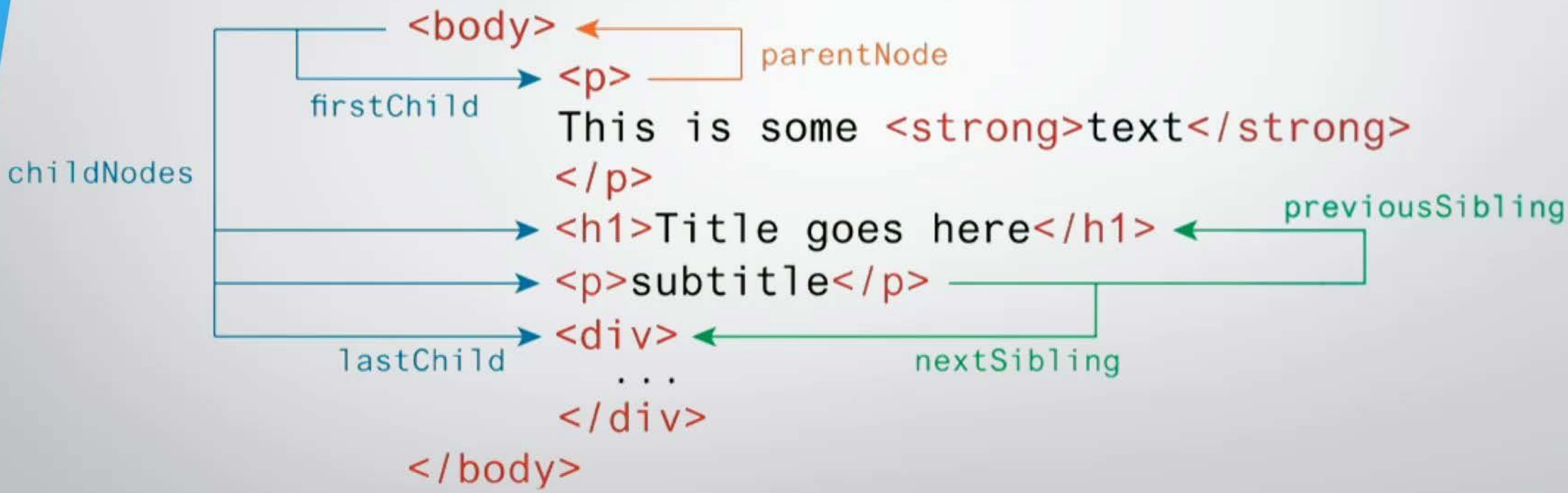
4. `node.classList.toggle("hide");` — If it isn't in the class specification, then add it

   Equivalent to: `<div class="box yellowish hide">`

5. `node.classList.toggle("hide");` — If it is in the class specification, then remove it

   Equivalent to: `<div class="box yellowish">`

Dom Family


<body>
parentNode
firstChild

This is some text

childNodes
previousSibling
<h1>Title goes here</h1>
subtitle

lastChild
nextSibling
...

</body>

Changing an Element's Content

```
node.innerHTML = "foo<em>bar</em>";
```

Creating DOM Elements

Creating DOM elements

```
<div id="first">
    <h1>DOM Example</h1>
    <p>Existing element</p>
</div>
```

Visualizing the DOM elements

```
<div>
    <h1> "DOM Example" </h1>
    <p> "Existing element" </p>
</div>
```

1. Create a new text node

```
var text = document.createTextNode("this is dynamic");
```

`"this is dynamic"`

2. Create a new empty <p> element

```
var p = document.createElement("p");
```

`<p></p>`

3. Add the text node to new <p> element

```
p.appendChild(text);
```

`<p> "this is dynamic" </p>`

4. Add the <p> element to the

```
var first = document.getElementById("first");
first.appendChild(p);
```

Creating DOM elements

```
<div id="first">
    <h1>DOM Example</h1>
    <p>Existing element</p>
</div>
```

Visualizing the DOM elements

```
<div>
    <h1> "DOM Example" </h1>
    <p> "Existing element" </p>
</div>
```

1. Create a new text node

```
"this is dynamic"
```

```
var text = document.createTextNode("this is dynamic");
```

2. Create a new empty <p> element

```
var p = document.createElement("p");
```

```
<p></p>
```

3. Add the text node to new <p> element

```
p.appendChild(text);
```

```
<p> "this is dynamic" </p>
```

4. Add the <p> element to the

```
var first = document.getElementById("first");
first.appendChild(p);
```

```
<div id="first">
    <h1>DOM Example</h1>
    <p>Existing element</p>
    <p>this is dynamic</p>
</div>
```

```
<div>
    <h1> "DOM Example" </h1>
    <p> "Existing element" </p>
    <p> "this is dynamic" </p>
</div>
```

Events

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 it is *handled* by JavaScript functions

Event Handling Approaches- Inline Hook

HTML document using the inline hooks

```
...
 <script type="text/javascript" src="inline.js"></script>
...
<form name='mainForm' onsubmit="validate(this);">
      <input name="name" type="text"
          onchange="check(this);"
          onfocus="highlight(this, true);"
          onblur="highlight(this, false);">
      <input name="email" type="text"
          onchange="check(this);"
          onfocus="highlight(this, true);"
          onblur="highlight(this, false);">
      <input type="submit"
          onclick="function (e) {
                    ...
                  }">
```

inline.js

```
function validate(node) {
  ...
}
function check(node) {
  ...
}
function highlight(node) {
  ...
}
```

Notice that you can define an entire event handling function within the markup. This is NOT recommended!

Event-Handling Approaches – Event Property Approach

- var myButton = document.getElementById('example');
- myButton.onclick = alert('some message');

Event-Handling Approaches – Event Listener Approach

- var myButton = document.getElementById('example');
- myButton.addEventListener('click', alert('some message'));
- myButton.addEventListener('mouseout', funcName);

Event-Handling Approaches – Event Listener Approach (anon function)

```
myButton.addEventListener('click', function() {
        var d = new Date();
        alert("You clicked this on "+
d.toString());
});
```

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)
{
        // find out where the user clicked
        var x = e.clientX;
        …
```

Capturing and Bubbling

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

Event Types – Mouse, Keyboard, Touch

EVENT TYPES

- Mouse events
- Keyboard events
- Touch events
- Form events
- Textfield events

Mouse Events

- **click** The mouse was clicked on an element
- **dblclick** The mouse was double clicked on an element
- **mousedown** The mouse was pressed down over an element
- **mouseup** The mouse was released over an element
- **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

Keyboard Events

- **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

- 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).

Form Validation

Form Validation

- HTML Level
- JavaScript Level
- Webserver Level

Form Events

- Form events
  - submit
  - reset
  - formdata
- Textfield events
  - blur
  - change
  - focus
  - select
  - …

Introduction to jQuery

The Most Popular Framework

- Makes working with JavaScript and DOM easy
- Cross browser compatibility
- Most popular JavaScript library (as of 2021). According to w3techs.com
  - Used by 78% of all website
  - 96% market share

Including jQuery

- Use a Content Delivery Network (CDN)
  - <script src="https://code.jquery.com/jquery-3.6.0.min.js"> </script>
- Can also be
  - Downloaded and used on your web server
  - Installed via package manager (npm, yarn, etc.)

jQuery Selectors

jQuery Selector

- 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

jQuery Selector

- 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");
```

- The $() function always returns a set of results

Basic Selectors

- $("*")—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 element with a given HTML id attribute

Examples

```
<body>
    <nav>
        <ul>
            <li><a href="#">Canada</a></li>
            <li><a href="#">Germany</a></li>
            <li><a href="#">United States</a></li>
        </ul>
    </nav>
    <div id="main">
        Comments as of <time>November 15, 2012</time>
        <div>
            <p>By Ricardo on <time>September 15, 2012</time></p>
            <p>Easy on the HDR buddy.</p>
        </div>
        <hr/>

        <div>
            <p>By Susan on <time>October 1, 2012</time></p>
            <p>I love Central Park.</p>
        </div>
        <hr/>
    </div>
    <footer>
        <ul>
            <li><a href="#">Home</a> | </li>
            <li><a href="#">Browse</a> | </li>
        </ul>
    </footer>
</body>
```

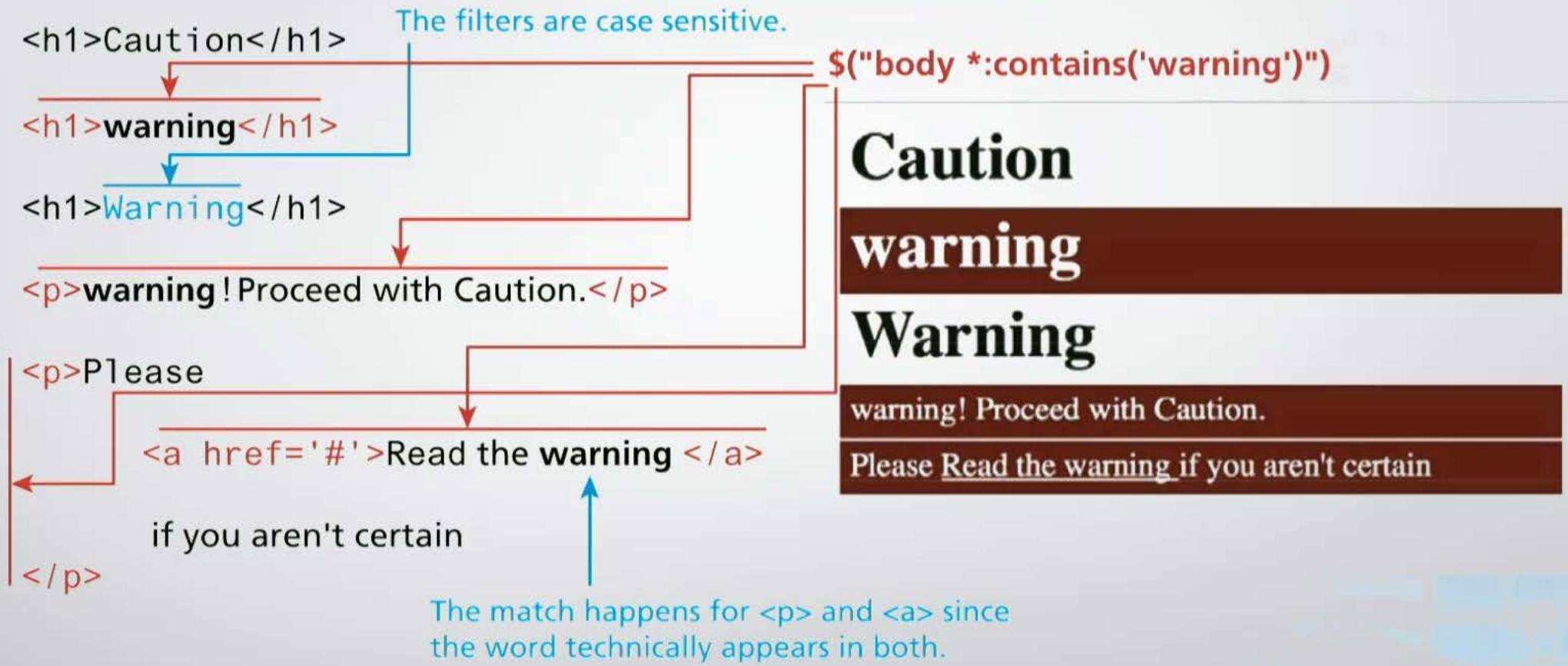
$("ul a:link")

$("#main time")

$("#main>time")

$("#main div p:first-child")

jQuery's Content Filter Selection



jQuery Manipulators

Common Element Manipulations – HTML

- 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

- 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 prop() method retrieves and set the value of a property, often used for form elements

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

```
var theBox = $(".meh");
theBox.prop("checked");
// evaluates to TRUE
```

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

Common Element Manipulations – Changing CSS

- jQuery provides the extremely intuitive css() method

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

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

WEEK 8

TOPIC 64 TO 72

Only audio/video lectures

Web Systems and Technologies

- Query Manipulators - Example
- Events in Query
- DOM Manipulation using jQuery
- Animations using jQuery
- Introduction to Ajax
- Asynchronous File Transmission
- Introduction to server side development, Comparison of server side technologies
- Installation of Apache webserver, PHP and MySQL for Local Development
- Introduction to PHP

WEEK 9

TOPIC 73 TO 81

Web Systems and Technologies

- 73 -Introduction to PHP (String Concatenation)
- 74 - Program Control - Conditional Program Statements
- 75 Program Control - Loops
- 76 Functions in PHP (1)
- 77 Functions in PHP(2)
- 78 Arrays in PHP
- 79 Sorting Arrays
- 80 Arrays - Super Global
- 81 $_GET and $_POST Arrays

Introduction to PHP – String Concatenation

String Concatenation

- Strings can be concatenated using (.) operator

```
$username = "Ricardo";
echo "Hello". $username;
//outputs Hello Ricardo
```

More concatenation examples

1 `echo "<img src='23.jpg' alt='" . $firstName . " " . $lastName . "' >";`

outputs

`<img src='23.jpg' alt='Pablo Picasso' >`

2 `echo "<img src='$id.jpg' alt='$firstName $lastName' >";`

`<img src='23.jpg' alt='Pablo Picasso' >`

3 `echo "<img src=\"$id.jpg\" alt=\"$firstName $lastName\" >";`

`<img src="23.jpg" alt="Pablo Picasso" >`

More concatenation examples

4

```
echo '<img src="' . $id . '.jpg" alt="' . $firstName . ' ' . $lastName . '" >';
```

```
<img src="23.jpg" alt="Pablo Picasso" >
```

5

```
echo '<a href="artist.php?id='.$id .'">'.$firstName.' '.$lastName.'</a>';
```

```
<a href="artist.php?id=23">Pablo Picasso</a>
```

Program Control – Conditionals

Comparison Operators

- Equal to operator ==

```
5 == '5'; //true
```

- Identical operator ===

```
5 === '5'; //false
```

if ...else

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

Alternate Syntax for Control Structure

```
<?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 endif; ?>
```

switch . . . case

- Uses equal to operator

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

match

- Introduced in PHP 8
  - Uses identical operator
  - Can return a value
  - Break not required
  - No default case (throws exception if not matched) or use conditions for default

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

Program Control – Loops

while and do . . . while

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

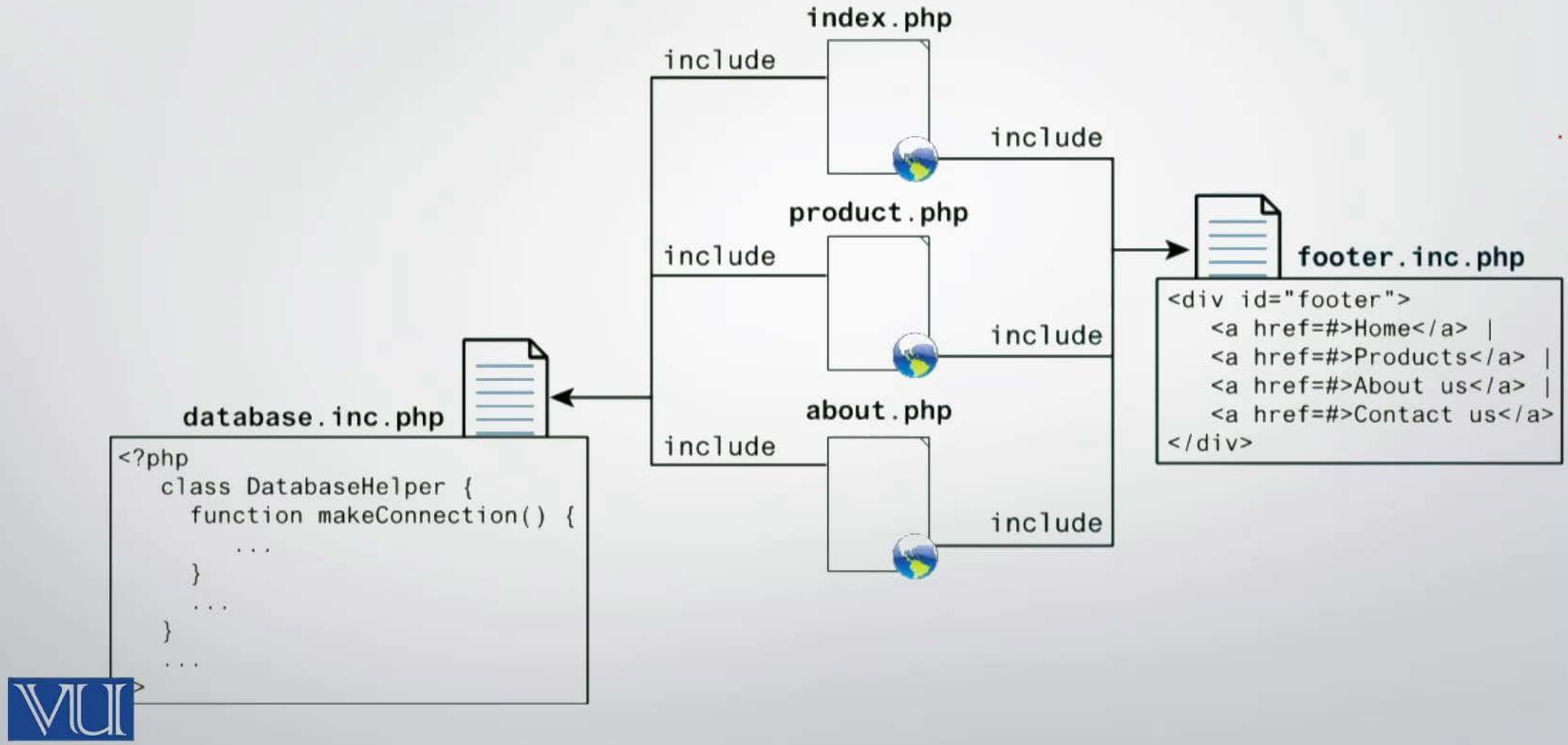
while and do . . . while

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

for

```
for ($count=0; $count < 100; $count+=5)
{
   echo $count;
}
```

Include Files



Include Files

- include "somefile.php";
- include_once "somefile.php";

Functions (1)

Function Syntax

- 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

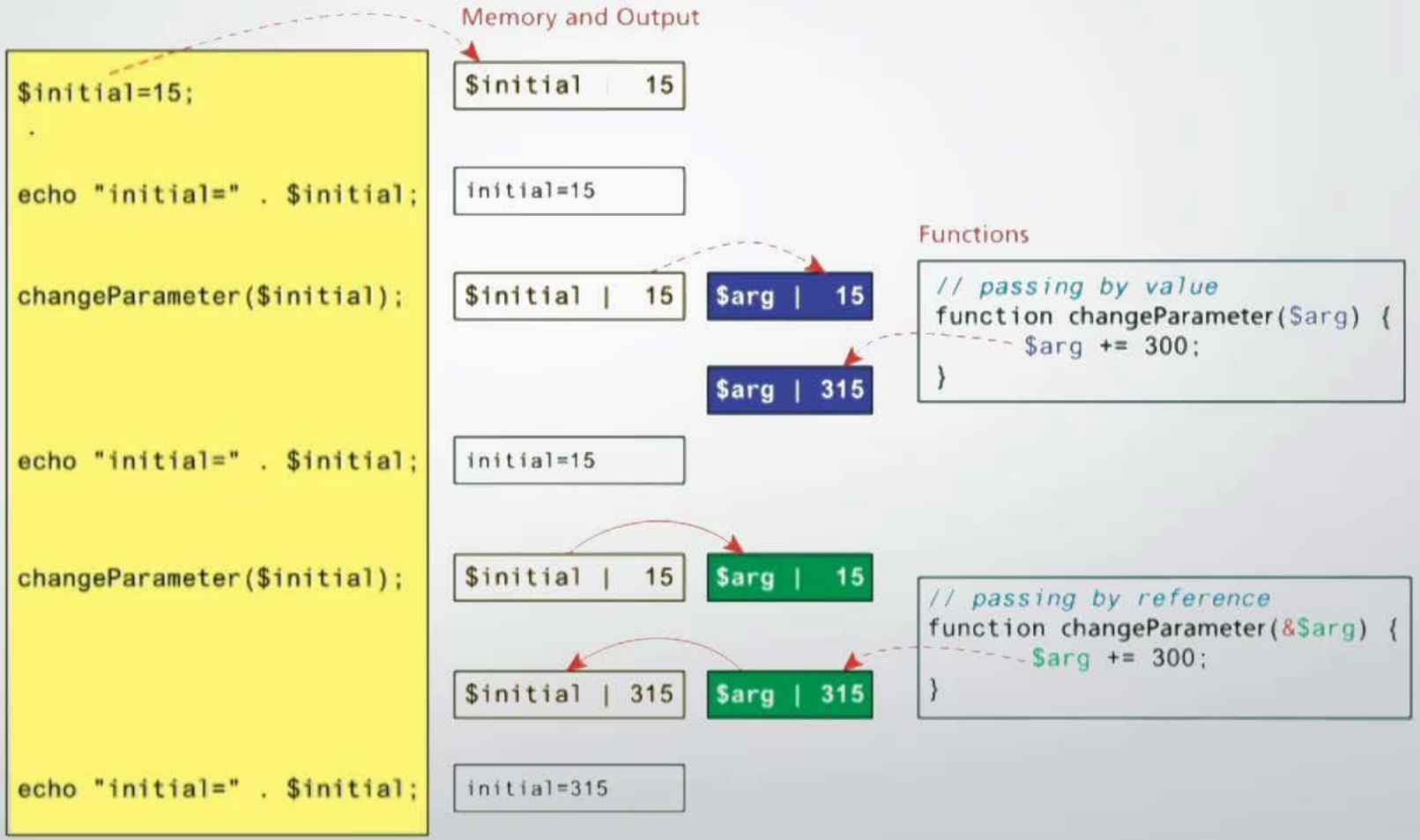
- A Return Type Declaration explicitly defines a function's return type by adding a colon and the return type after the parameter list when defining a function

```
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

Parameters – Passing Parameters by Reference



Parameters – Default Values

- Parameters can have default values

```
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
```

Functions (2)

Parameters – Default Values

- Parameters can have default values

```
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
```

Parameters – Named Arguments

- Named arguments supported in PHP 8

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

func(5,10,15);   //5 10 15
func();          //1 2 3
func(c: 5);      //1 2 5
```

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;
}
```

Arrays

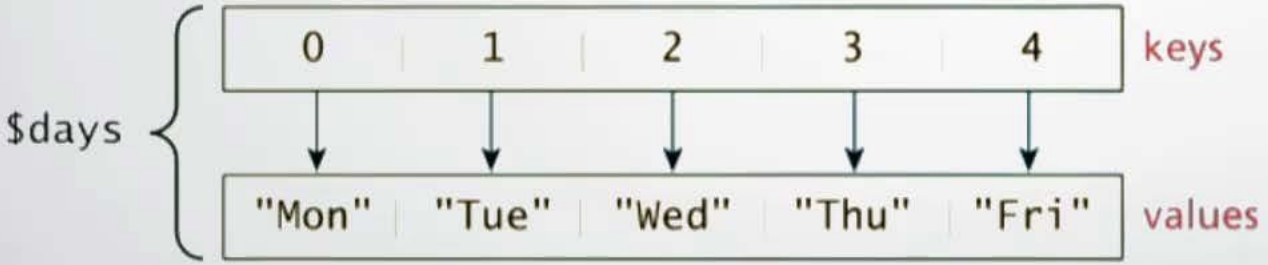
Defining and Accessing an Array

- Using array function

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

- Using array notation

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

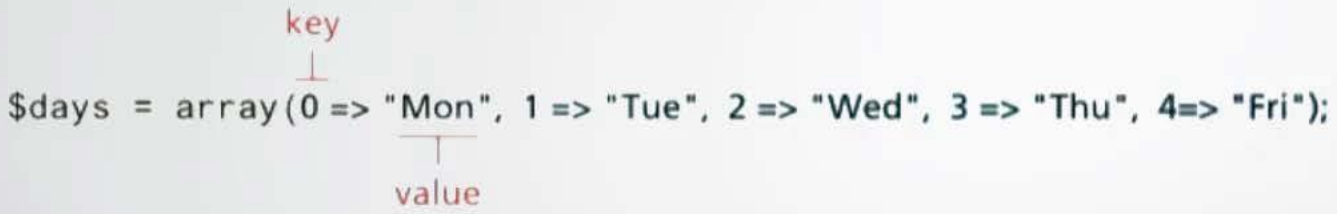


- For debugging, use *var_dump($days)* or *print_r($days)* to see contents of the array (or any variable). Place output in <pre></pre>

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

Defining and Accessing an Array

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



Arrays

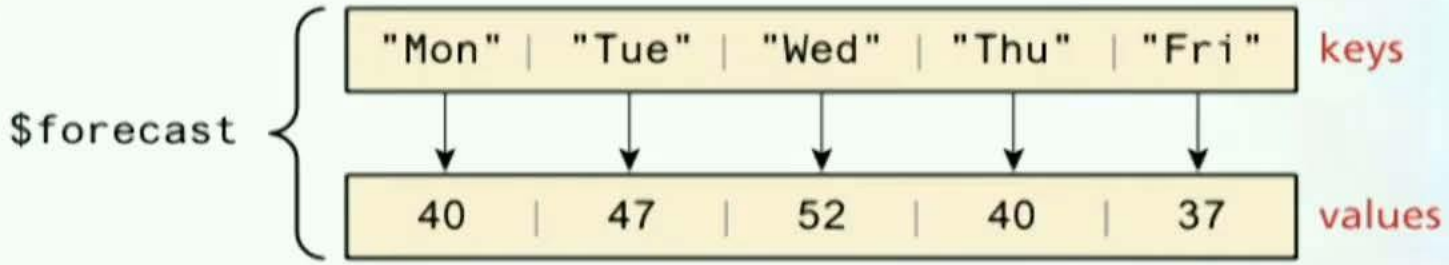
Defining and Accessing an Array

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

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

key: "Mon", "Tue", "Wed", "Thu", "Fri"

value: 40, 47, 52, 40, 37

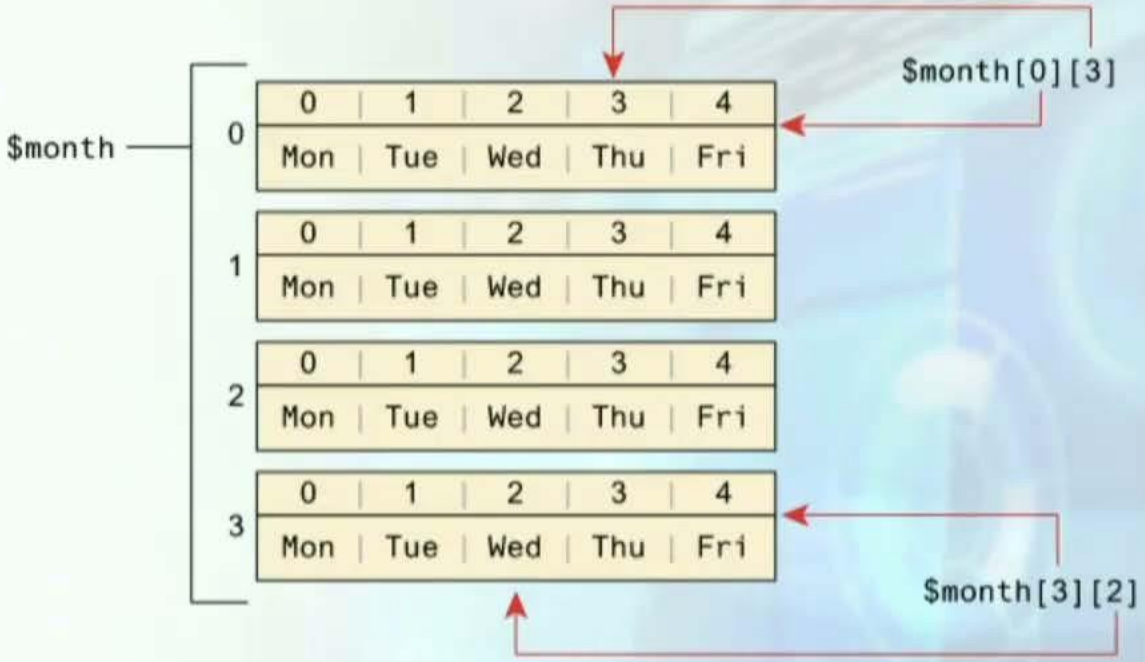



```
echo $forecast["Tue"];   // outputs 47
echo $forecast["Thu"];   // outputs 40
```

Arrays

Multidimensional Arrays

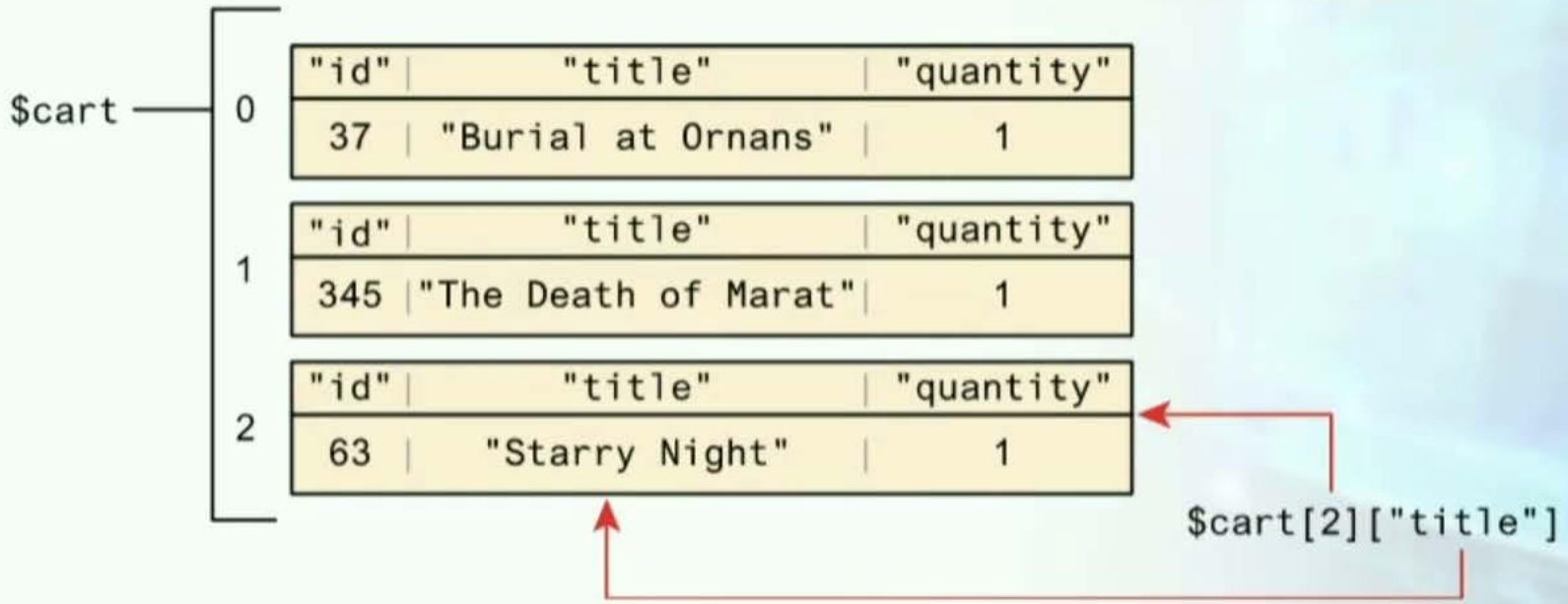
```
$month = array
(
   array("Mon","Tue","Wed","Thu","Fri"),
   array("Mon","Tue","Wed","Thu","Fri"),
   array("Mon","Tue","Wed","Thu","Fri"),
   array("Mon","Tue","Wed","Thu","Fri")
);
```



Arrays

Multidimensional Arrays

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

Iterating through an Array - foreach

```
//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()

Arrays – Sorting

Array Sorting

- 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)
```

More Array Operations

```
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)
```

Arrays – Super Globals

Superglobal Arrays

- 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

Superglobal Arrays

- $GLOBALS
  - Array for reading and storing data that needs superglobal scope
- $_COOKIES
  - Array of cookie data passed to page via HTTP request
- $_SESSION
  - Array that contains session data

Superglobal Arrays

- $_ENV
  - Array of server environment data (mostly OS related and paths)
- $_SERVER
  - Array containing information about the request and the server

Superglobal Arrays

- $_GET
  - Array of query string data passed to the server via the URL
- $_POST
  - Array of query string data passed to the server via the HTTP header
- $_REQUEST
  - Array containing the contents of $_GET, $_POST, and $_COOKIES
- $_FILES
  - Array of file items uploaded to the server

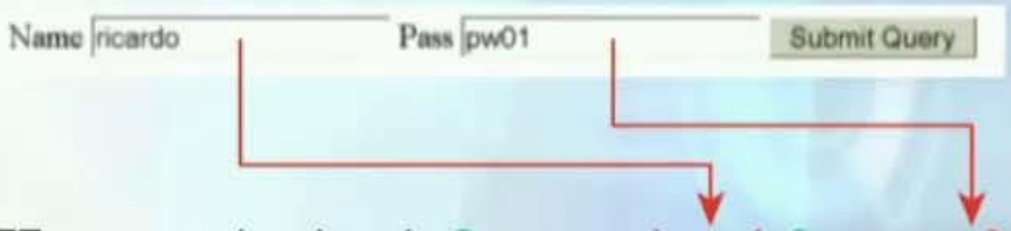
$_GET and $_POST Superglobal Arrays

Reading Form Data – GET

HTML (client)

```
<form action="processLogin.php" method="GET">
    Name <input type="text" name="uname" />
    Pass <input type="text" name="pass" />
    <input type="submit">
</form>
```

Browser (client)



HTTP request

```
GET processLogin.php?uname=ricardo&pass=pw01
```

PHP (server)

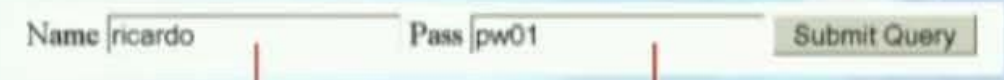
```
// within processLogin.php
echo $_GET["uname"]; // outputs ricardo
echo $_GET["pass"];  // outputs pw01
```

Reading Form Data – POST

HTML (client)

```
<form action="processLogin.php" method="POST">
    Name <input type="text" name="uname" />
    Pass <input type="text" name="pass" />
    <input type="submit">
</form>
```

Browser (client)



HTTP request

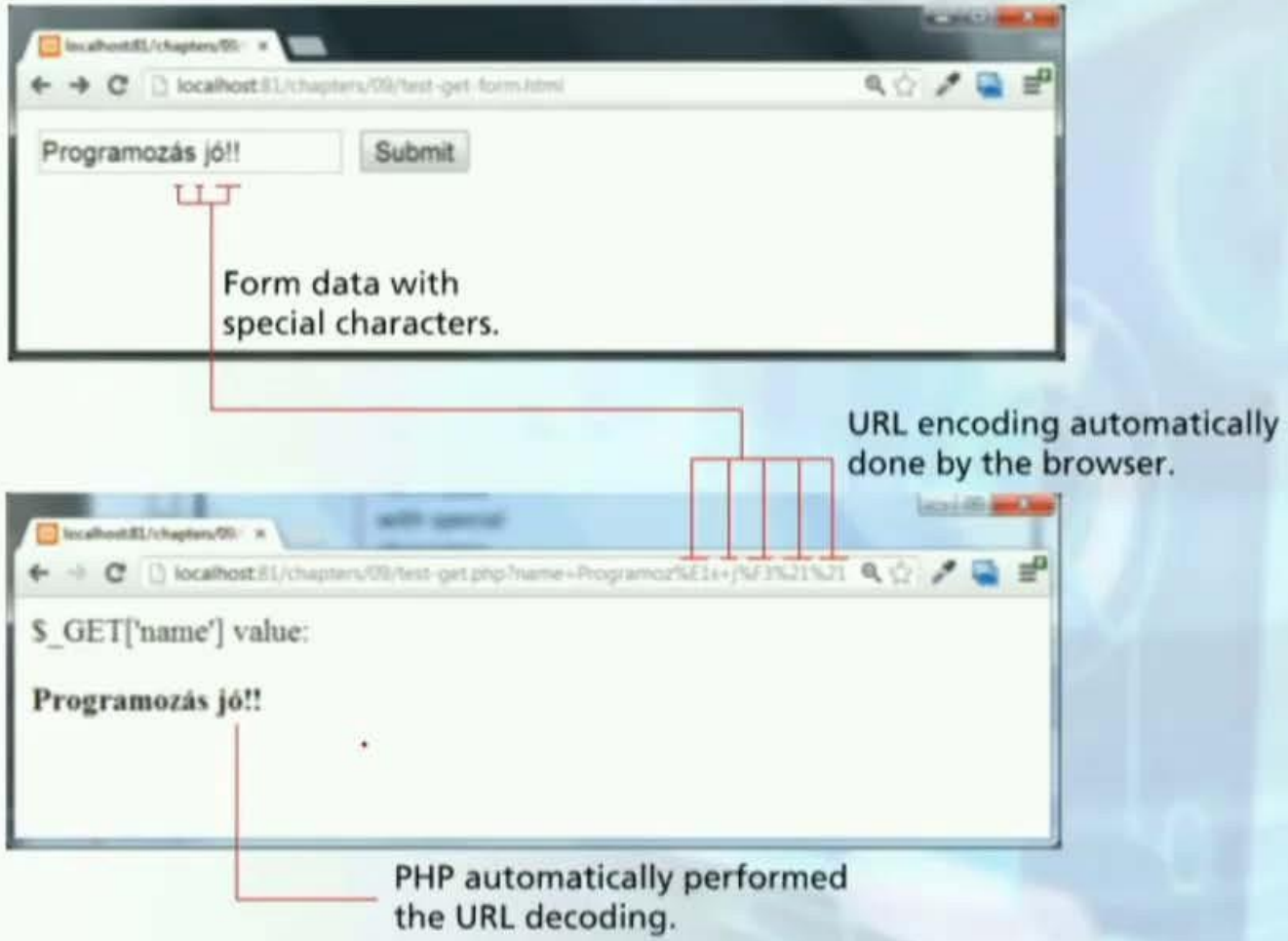
POST processLogin.php

HTTP POST request body:
uname=ricardo&pass=pw01

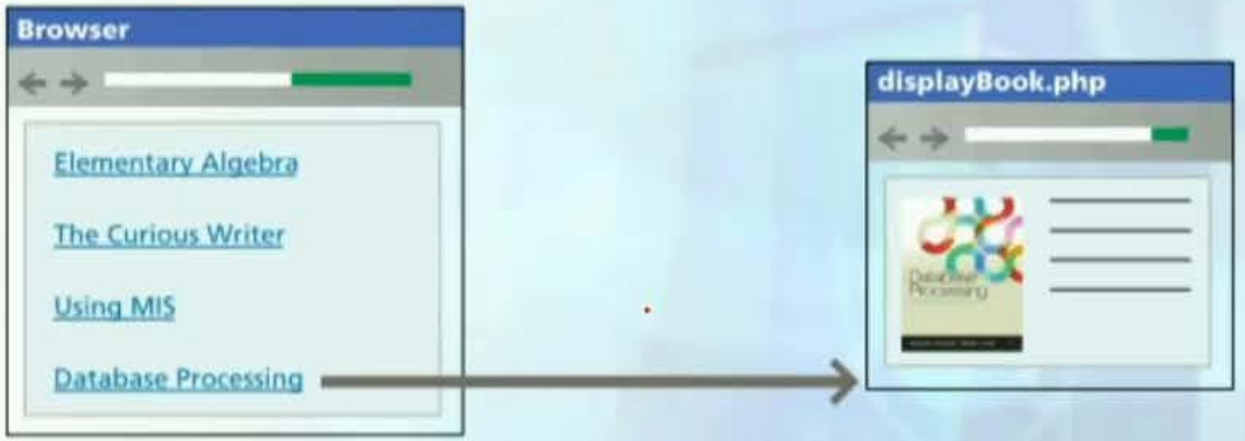
PHP (server)

```
//File processLogin.php
echo $_POST["uname"]; //outputs "ricardo";
echo $_POST["pass"];  //outputs "pw01";
```

Note URL Encoding and Decoding



Using Query Strings in Hyperlinks



Sanitizing Query Strings

- That is, 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

WEEK 10

TOPIC 82 TO 90

Web Systems and Technologies

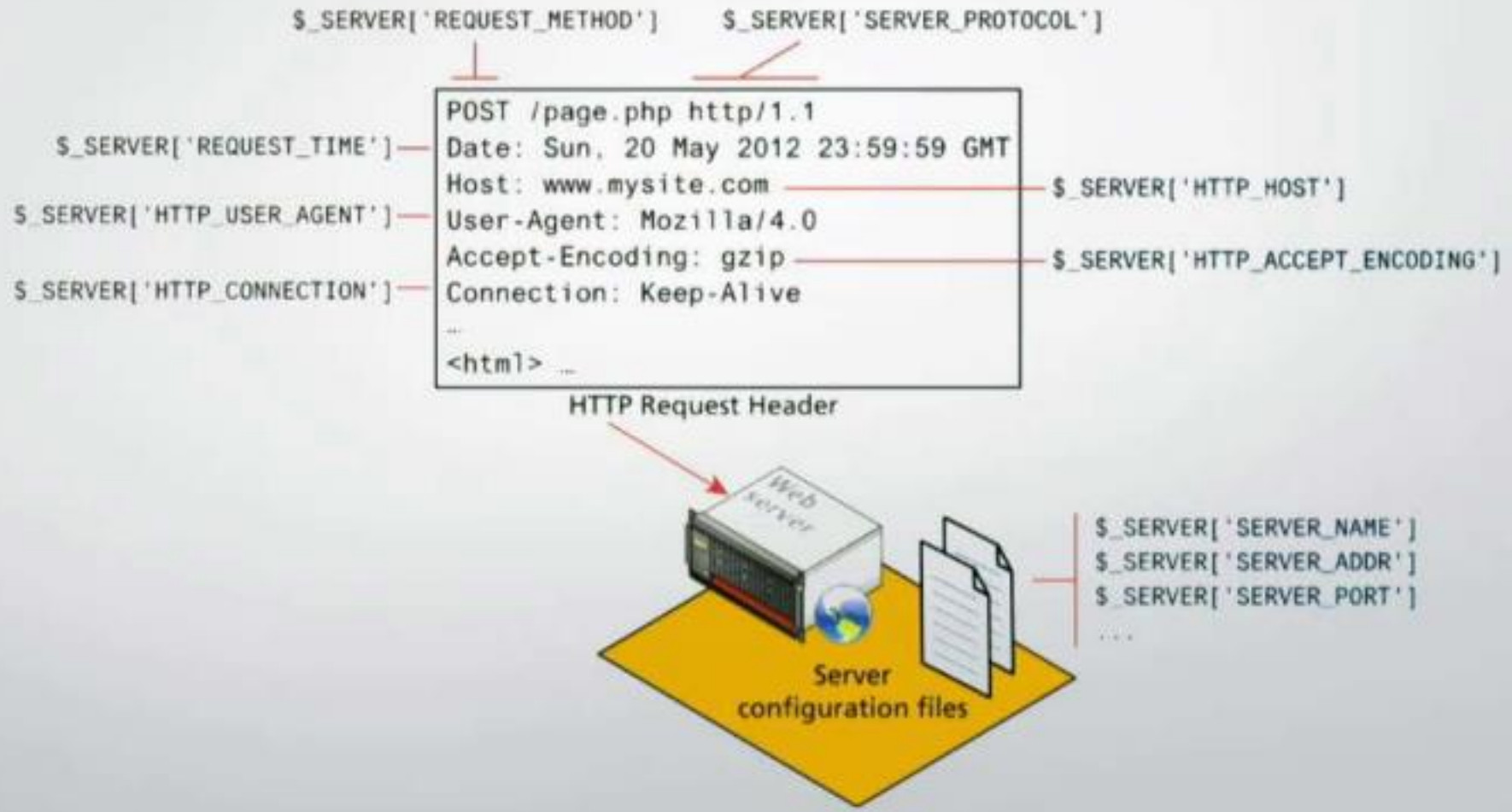
- 82 $_GET and $_POST arrays - Example
- 83 $_SERVER Array
- 84 $_FILES Array
- 85 Reading and Writing Files
- 86 Classes and Objects in PHP; Object Properties
- 87 Constructor; Methods
- 88 Visibility
- 89 Static members; Constants
- 90 Databases and Web Development

$_GET and $ _POST Superglobal Arrays - Example

$_SERVER Array

$_SERVER Array

Server Information Keys



Request Header Information Keys

- Advanced browser detection, 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);
?>
```

Request Header Information Keys

```
$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
```

$_FILES Array

HTML Required for File Uploads

- 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.

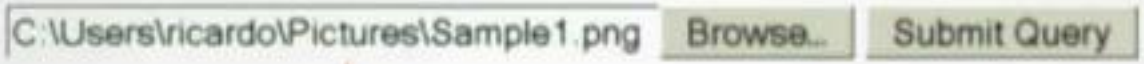
```
<form enctype='multipart/form-data'
method='post'>
  <input type='file' name='file1'
id='file1'>
  <input type='submit'>
</form>
```

Handling the File Upload in PHP

HTML (client)

```
<form enctype='multipart/form-data' method='post' action='upFile.php'>
   <input type='file' name='file1'>
   <input type='submit' value="Submit Query">
</form>
```

Browser (client)



HTTP request

POST upFile.php

HTTP POST multipart/form-data

```
file1%PNG "‡›*Ì ! %çÔkáFÊ÷Ì§29%*úÅ¡Šrá vÛ,ÿiN üc/©(–Å Å Z)/vè\Å(m~%i± %6_Ê/Hi.+*" _ÆÊA`) .þ/
_êvßoá'™ ß%uÔaGé®N^ÔÔ?ÔŸ·ì'm ¾Æ2h‹ ¶÷`|QÅóiçD"W)MìÛü_9ß9nŸ02'Å' km'Ñyþh $2`›Ÿ` d®ß1¬\
¡ûh»)>´‡¥[ ðB*ðéè[ ‡¥uè|ŸxR%XL)áE›[ÕZ!pâ X¢ kÔ >Ô 'cÛÿ]¤‡ ÅÔt©Q`ìÜ±¥<3%üÔ3 ×1'ó¡¶1¬W+4éê
®á,Ôó9'tâÊC ¢lÛ11Êée`ß>{Üèâ§® ®Ôè|¶ì℗%soiz*VydÜÊµ'ÅøÅ_ Å ˆÇ jñzôce¢®Ÿ-ß ¯=àã'~ÀII'¡íüü 5%$1 "
Z³Ñ ðº Å¶ £_hÉGÀ ¢°÷¥ì>É/ƶÏ¢‹ûT _Å| -ð€[®Å±ç€_¨nü»ßð®üÐmŸa ì ¿ô{?üö÷1™ Û_ìê?\ÉÅ /
æsÚ"ß%%tëy¯`>qàìÜõÎÔøþ Ì÷ó7ßü‹ ‡ßßÅ, ßøó'?÷ÀìÎ]Xtæ11%xßõ ?¿ ñÜLòoþ1 Ü7~1¦;¶ì>d§éà >
®¶Å:K(*Gëð±xÅ Üsx *wnÜ%o×¶ñ±ýX]1Ii\ ^'fÅéuéL6>yÉÅ%ÇÐUÜ`ý#÷÷eÔøàa° ÊW`ó<wÉ5ë'ÇGf&ÇàÎQß›?eà
...
```

```
echo $_FILES["file1"]["name"]      // "Sample1.png"
echo $_FILES["file1"]["type"]      // "image/png"
echo $_FILES["file1"]["tmp_file"]  // "/tmp/phpJ08pVh"
echo $_FILES["file1"]["error"]     //   0
echo $_FILES["file1"]["size"]      // 1219038
```

PHP (server)

Checking for Errors

```
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 ";
  }
}
```

File Size Restrictions

- You can limit in multiple ways
- HTML form attributes in inputs (browser)
- JavaScript (browser)
- PHP validation (server)

Limiting the Type of File Upload

```
$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.";
}
```

Reading/Writing Files

Two ways

- There are two basic techniques for read/writing files in PHP:

  Stream access. Just read a small portion of the file at a time

  All-In-Memory access. Read the entire file into memory (i.e., into a PHP variable). While not appropriate for large files, it does make processing of the file extremely easy.

Stream Access

- To those of you familiar with functions like fopen() , fclose() , and fgets() from the C programming language, this first technique will be familiar

- Open fopen()
- Read data fgets()
- Close the file fclose()

In-Memory File Access

- file() Reads the entire file and returns an array, with each array element corresponding to one line in the file.
- file_get_contents() Reads the entire file and returns a string variable.
- file_put_contents() Writes the contents of a string variable out to a file.

Classes and Objects in PHP

Defining Classes

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

Instantiating Objects

- Use the new keyword

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

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";
```

Web Systems and Technologies

TOPIC 87

Classes and Objects in PHP Constructor & Methods

Classes and Objects in PHP

Constructor

```
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;
}
}
```

Classes and Objects in PHP

Instantiating using Constructors

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

Classes and Objects in PHP

Methods

- Methods define the tasks each instance of a class can perform and are useful since they associate behavior with objects

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

$a->outputAsTable()
```

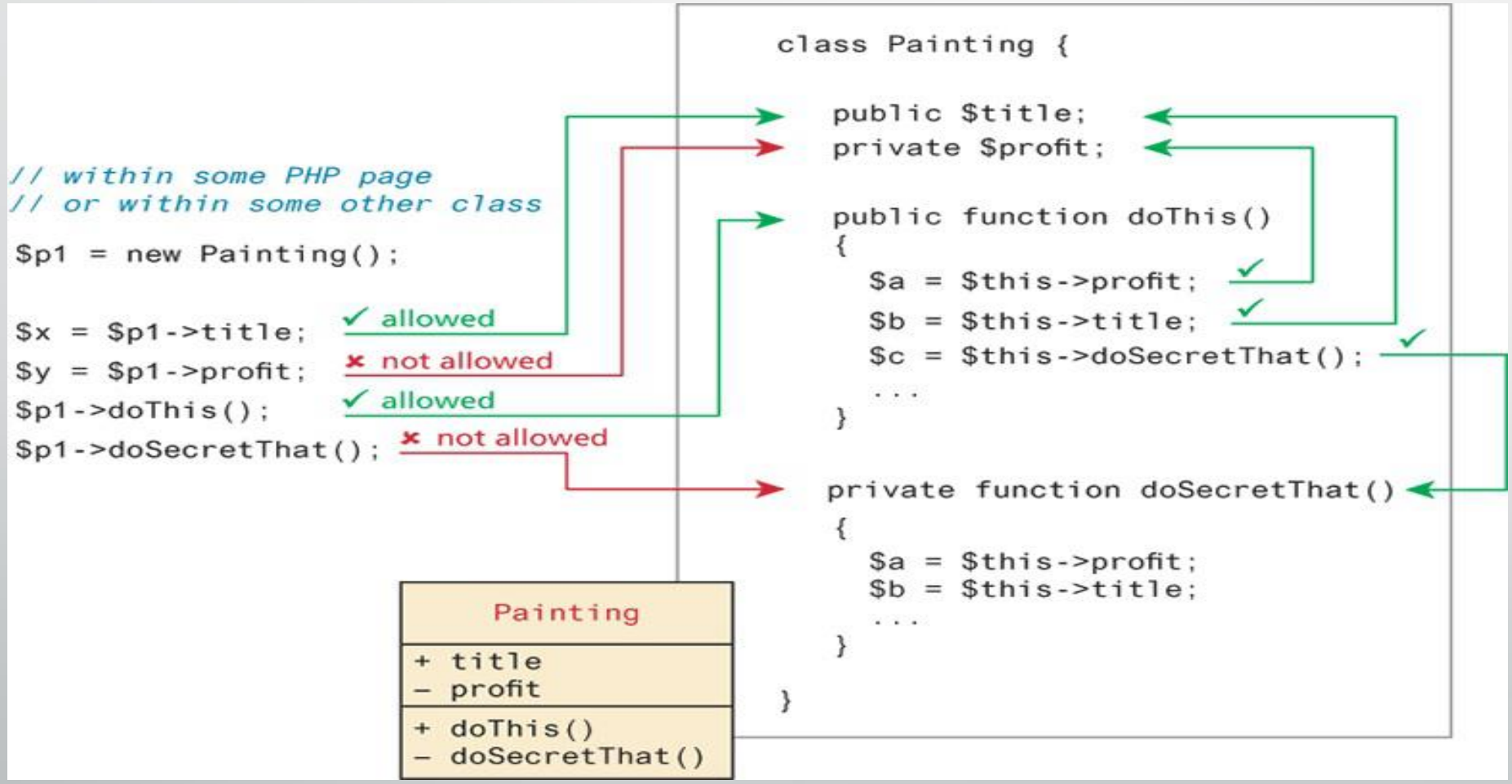
Web Systems and Technologies

TOPIC 88

Classes and Objects in PHP Visibility

Classes and Objects in PHP

Visibility



Web Systems and Technologies

TOPIC 89

Classes and Objects in PHP Static Members & Constants

Classes and Objects in PHP

Static Members

- A **Static member** is a property or method that all instances of a class share
- Static value is shared across all instances

```
class Artist {
public static $artistCount = 0;
public static function addArtist() {
self:: $artistCount++;
}
}
Artist::addArtist();
print(Artist::$artistCount);
```

Classes and Objects in PHP

Class Constants

- **Constant Values** can be stored more efficiently as class constants so long as they are not calculated or updated

```
Const   EARLIEST_DATE ='January1, 1200';
```

- They can be accessed both inside and outside theclass using self::EARLIEST_DATE in the class and classReference::EARLIEST_DATE

Web Systems and Technologies

TOPIC 90

Databases and Web Development

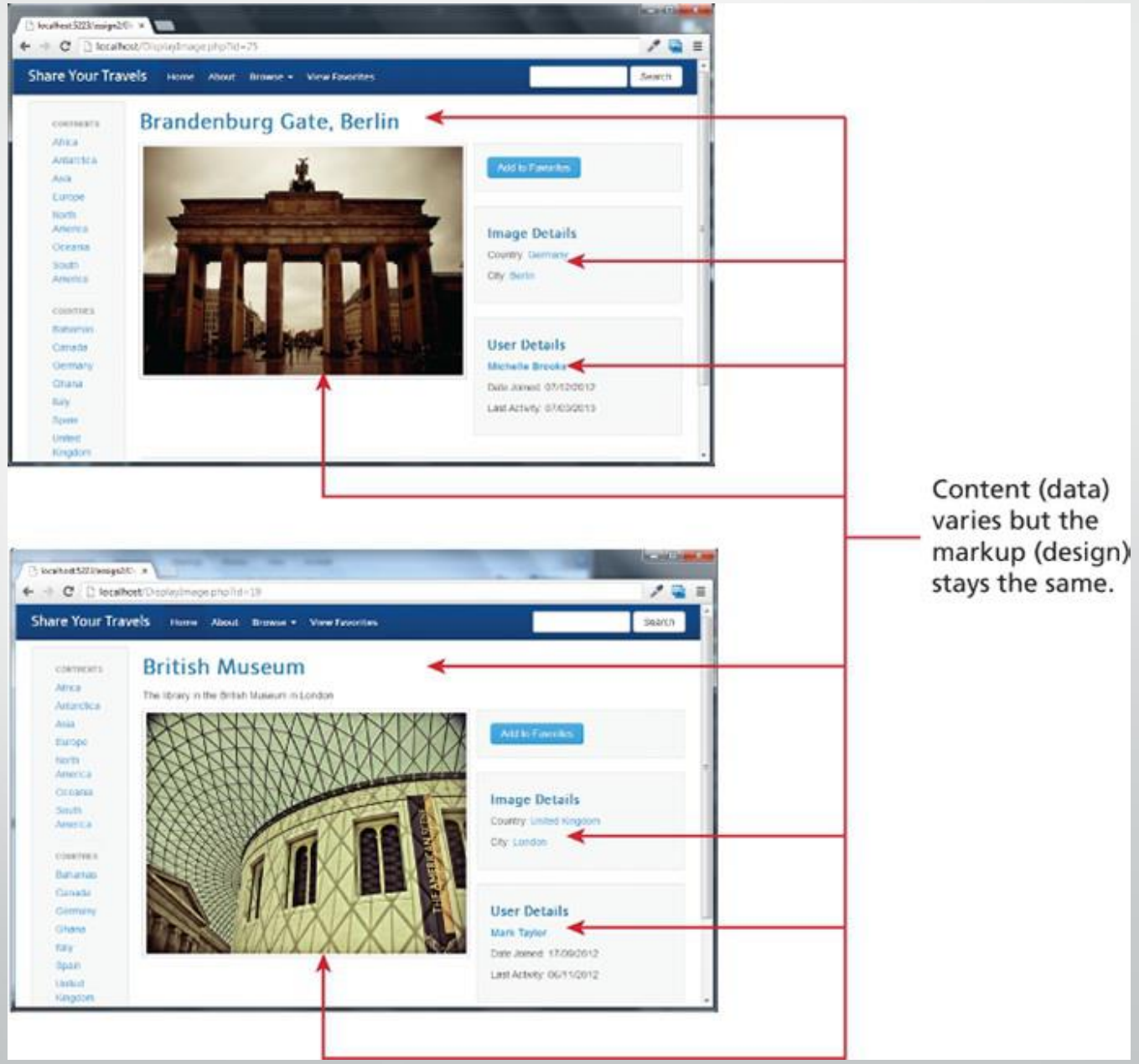
Databases and Web Development

The Role of Databases in Web Development

- HTML pages remain constant
- Using databases with PHP allows to make parts of HTML dynamic(variable)

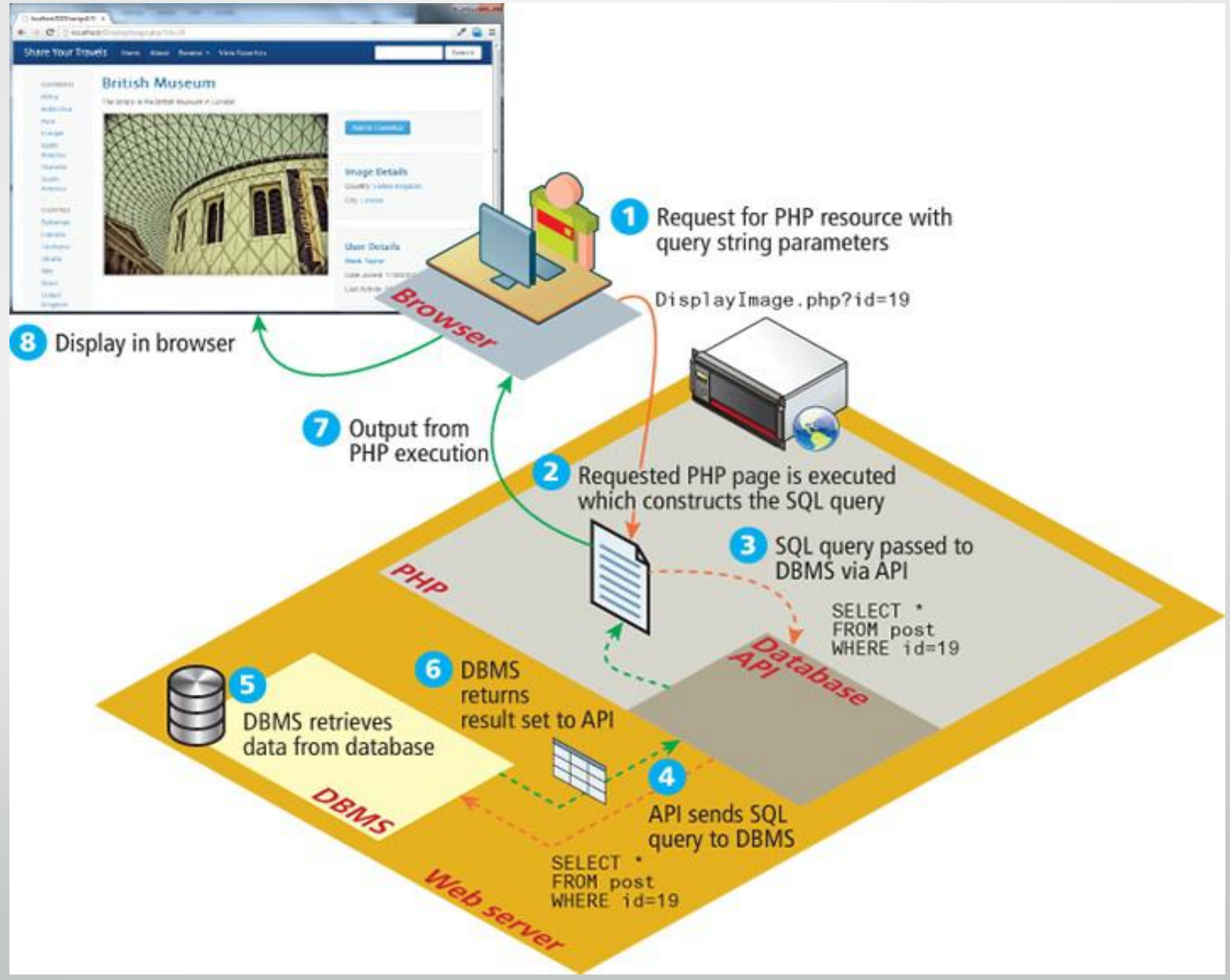
Databases and Web Development

Role of databases in web development



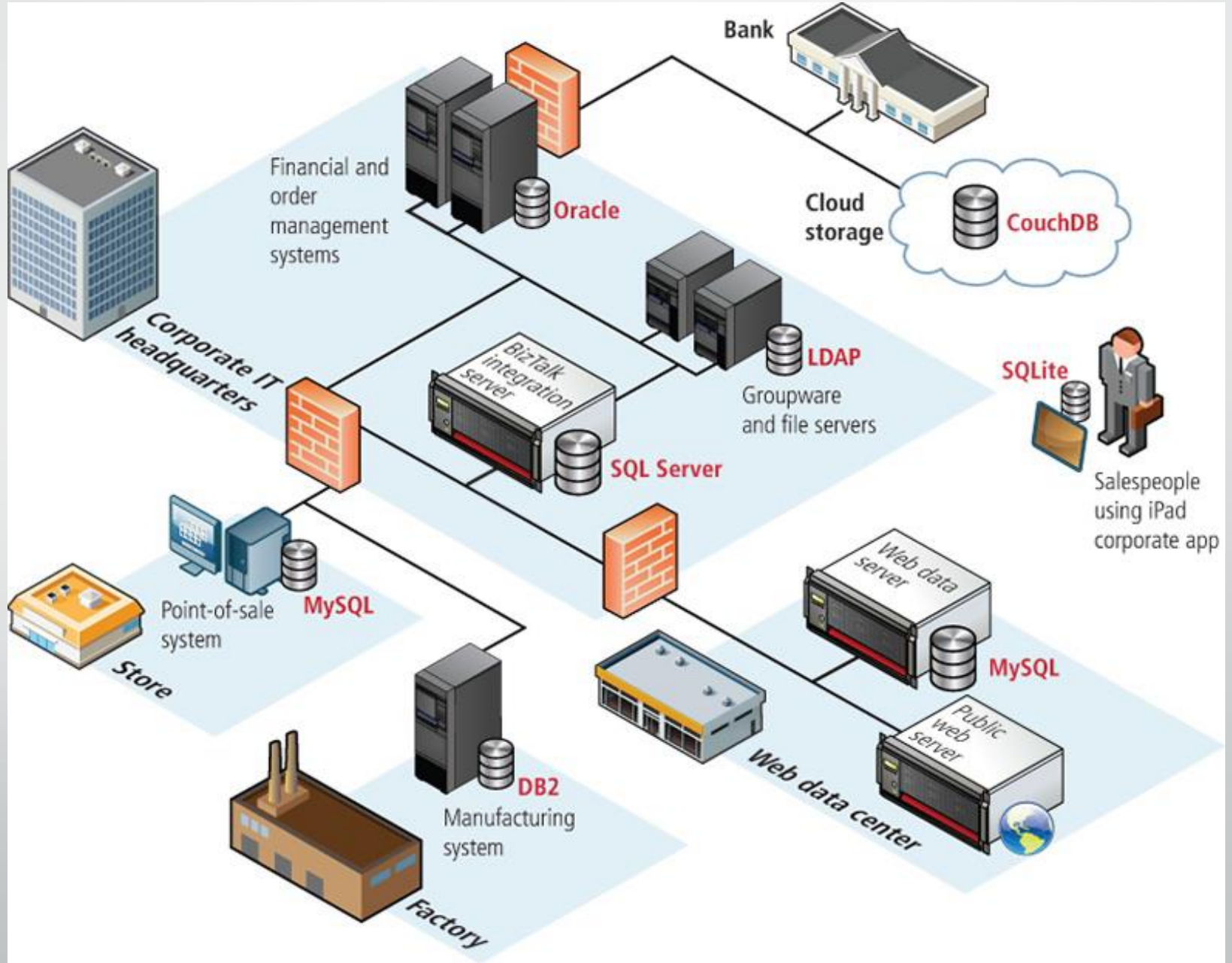
Databases and Web Development

How websites use databases



Databases and Web Development

Database options



WEEK 11

TOPIC 91 TO 99

SQL,Database,PHP

- 91 SQL - Select, Insert, Update, Delete
- 92 Introduction to MySQL
- 93 phpMyAdmin
- 94 Accessing MySQL in PHP
- 95 Fetching Data
- 96 Performing Transactions
- 97 Example: Database Creation
- 98 Example: Database Insertion
- 99 Example: Database Search

SQL – Select, Insert, and Update

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

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)

Join together

Because the field name ArtistID is ambiguous, need to preface it with table name

Table 1

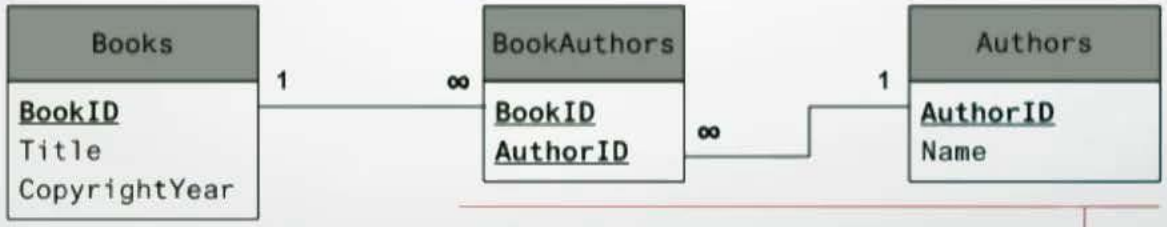
```
SELECT Artists.ArtistID, Title, YearOfWork, Name FROM Artists
INNER JOIN ArtWorks ON  Artists.ArtistID = ArtWorks.ArtistID
```

SQL keywords indicate the type of join

Table 2

Primary key in Table 1

Foreign key in Table 2




```
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

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 record with a single value in it.

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 as many records as there are unique values in the group-by field.

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

INSERT, UPDATE, and DELETE Statements

SQL keywords for inserting (adding) a new record

Table name

Fields that will receive the data values

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

Values to be inserted. Note that string values must be within quotes (single or double).

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!

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" */

Introduction to MySQL

Command-Line Interface

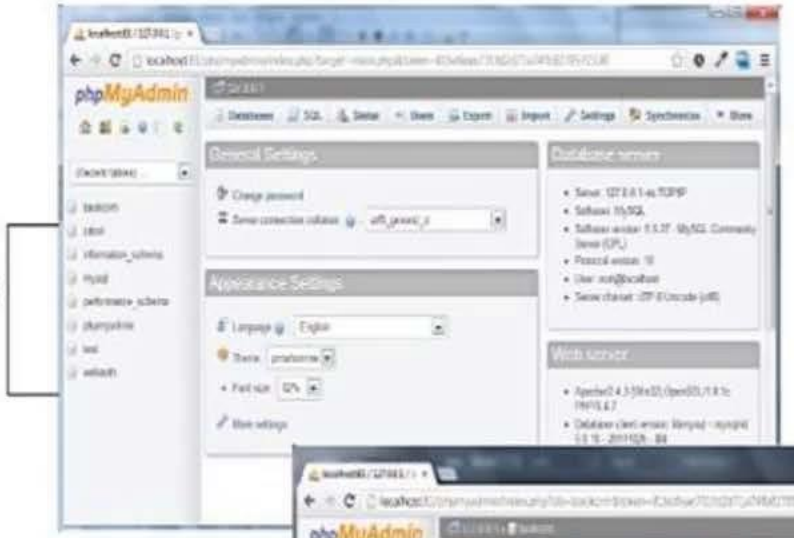
```
Database changed
mysql> SHOW TABLES;
+--------------------------+
| Tables_in_book_database |
+--------------------------+
| authors                  |
| bindingtypes             |
| bookauthors              |
| books                    |
| categories               |
| disciplines              |
| imprints                 |
| productionstatuses       |
| subcategories            |
+--------------------------+
9 rows in set (0.00 sec)

mysql> SHOW COLUMNS IN authors;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| ID          | int(11)      | NO   | PRI | NULL    | auto_increment |
| FirstName   | varchar(255) | YES  |     | NULL    |                |
| LastName    | varchar(255) | YES  |     | NULL    |                |
| Institution | varchar(255) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM authors WHERE FirstName LIKE "A%";
+-----+-------------+-----------+-----------------------------------------------------+
| ID  | FirstName   | LastName  | Institution                                         |
+-----+-------------+-----------+-----------------------------------------------------+
|   2 | Andrew      | Abel      | Wharton School of the University of Pennsylvania    |
|  25 | Allen       | Center    | NULL                                                |
|  37 | Allen       | Dooley    | Santa Ana College                                   |
|  40 | Andrew      | DuBrin    | Rochester Institute of Technology                   |
|  56 | Allan       | Hambley   | NULL                                                |
|  57 | Arden       | Hamer     | Indiana University of Pennsylvania                  |
|  82 | Arthur      | Keown     | Virginia Polytechnic Instit. and State University   |
| 102 | Annie       | McKee     | NULL                                                |
| 119 | Arthur      | O'Sullivan| NULL                                                |
| 172 | Allyn       | Washington| Dutchess Community College                          |
| 194 | Anne Frances| Wysocki   | University of Wisconsin, Milwaukee                  |
| 198 | Alice M.    | Gillam    | University of Wisconsin-Milwaukee                   |
| 214 | Anthony P.  | O'Brien   | Lehigh University                                   |
| 216 | Alvin C.    | Burns     | NULL                                                |
| 225 | Abbey       | Deitel    | NULL                                                |
| 252 | Alvin       | Arens     | Michigan State University                           |
| 258 | Ali         | Ovlia     | NULL                                                |
| 270 | Anne        | Winkler   | NULL                                                |
| 275 | Alan        | Marks     | DeVry University                                    |
+-----+-------------+-----------+-----------------------------------------------------+
19 rows in set (0.00 sec)

mysql>
```

Command-Line Interface

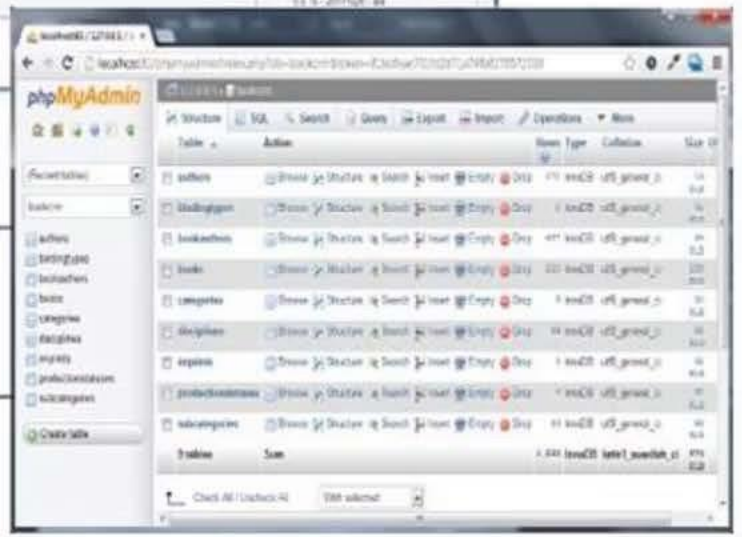
- To launch an interactive MySQL command-line session, you must specify the host, username, and database name to connect to as shown below:
- **mysql -h 192.168.1.14 -u bookUser –p**
- To import commands from a file called commands.sql , for example, we would use the < operation:
- **mysql –h 192.168.1.14 –u bookUser –p < commands.sql**

phpMyAdmin

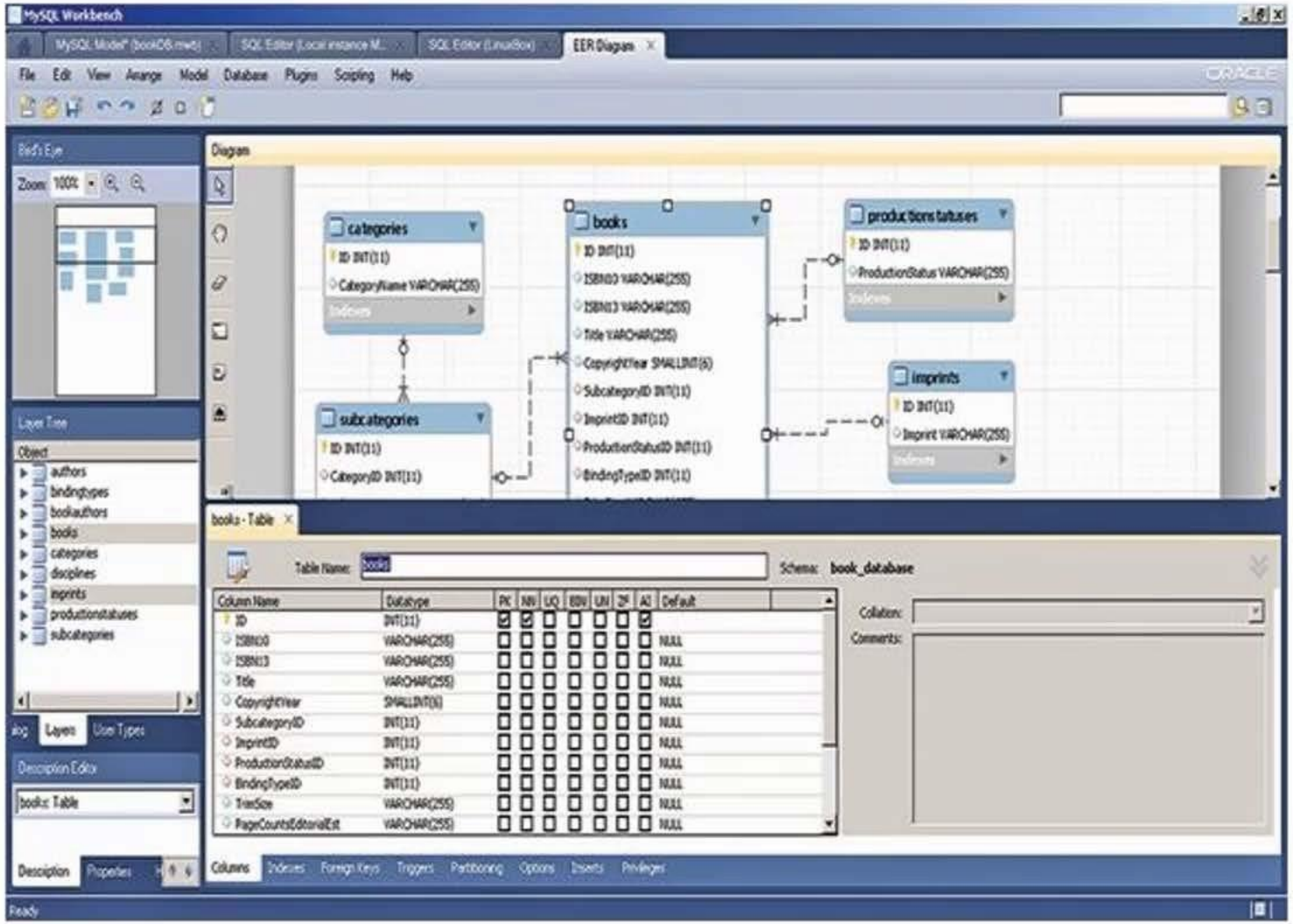
MySQL has a number of predefined databases it uses for its own operation.



phpMyAdmin allows you to view and manipulate any table in a database.



MySQL Workbench



Using phpMyAdmin

Accessing MySQL in PHP

PHP MySQL APIs

- **MySQL extension**. This was the original extension to 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
- **PHP data objects (PDOs)**. 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

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
- Like many things in the web world, there is no single best choice.
- As the chapter (and book) proceed, we will standardize on the object-oriented, database-independent PDO approach.

Basic Connection Algorithm

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

Basic Connection Algorithm

```php
<?php

try {
   $connString = "mysql:host=localhost;dbname=bookcrm";
   $user = "testuser";
   $pass = "mypassword";

   $pdo = new PDO($connString,$user,$pass);
   $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

   $sql = "SELECT * FROM Categories ORDER BY CategoryName";
   $result = $pdo->query($sql);

   while ($row = $result->fetch()) {
      echo $row['ID'] . " - " . $row['CategoryName'] . "<br/>";
   }
   $pdo = null;
}
catch (PDOException $e) {
   die( $e->getMessage() );
}

?>
```

Connecting to a Database (mysqli peocedural)

```
// modify these variables for your
installation

$host = "localhost";

$database = "bookcrm";

$user = "testuser";

$pass = "mypassword";

$connection = mysqli_connect($host,
$user, $pass, $database);
```

Connecting to a Database (PDO Object-oriented)

```
// modify these variables for your
installation

$connectionString =
"mysql:host=localhost;dbname=bookcrm";

$user = "testuser";

$pass = "mypassword";

$pdo = new PDO($connectionString,
$user, $pass);
```

Handling Connection Errors - mysqli

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

// mysqli_connect_errno returns the
last error code

if ( mysqli_connect_errno() ) {

        die( mysqli_connect_error() );
        // die() is equivalent to exit()

}
```

Handling Connection Errors - PDO

```
try {
        $connString =
"mysql:host=localhost;dbname=bookcrm";
        $user = DBUSER;
        $pass = DBPASS;
        $pdo = new
PDO($connString,$user,$pass);
        . . .
}
catch (PDOException $e) {
        die( $e->getMessage() );
}
```

Web Systems and Technologies

Accessing MySQL in PHP
Fetching Data

Accessing MySQL in PHP

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);
```

Accessing MySQL in PHP

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/>”;
}
```

Accessing MySQL in PHP

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	values

keys

Accessing MySQL in PHP

Freeing Resources and Closing Connection

```
// close the  connection

mysqli_close($connection);

// closes connection and frees the resources used by the PDO
object

$pdo = null;
```

Accessing MySQL in PHP

Working with Parameters

```
$sql = “UPDATE Categories SET CategoryName='Web' WHERE

CategoryName='Business'”;

$count = $pdo->exec($sql);

echo "<p>Updated ” . $count . “ rows</p>”;
```

Accessing MySQL in PHP

Working with Parameters – Technique 1 ? Placeholders

```
$sql = "INSERT INTO books (ISBN10, Title, CopyrightYear, ImprintId,
ProductionStatusId, TrimSize, Description) VALUES (?,?,?,?,?,?,?)";
$statement = $pdo->prepare($sql);
$statement->bindValue(1, $_POST['isbn']);
$statement->bindValue(2, $_POST['title']);
$statement->bindValue(3, $_POST['year']);
$statement->bindValue(4, $_POST['imprint']);
$statement->bindValue(5, $_POST['status']);
$statement->bindValue(6, $_POST['size']);
$statement->bindValue(7, $_POST['desc']);
$statement->execute();
```

Accessing MySQL in PHP

Working with Parameters – Technique 1 ? Placeholders with Array

```
$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']);
```

Accessing MySQL in PHP

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) ";
$statement = $pdo->prepare($sql);
$statement->bindValue(':isbn', $_POST['isbn']);
$statement->bindValue(':title', $_POST['title']);
$statement->bindValue(':year', $_POST['year']);
$statement->bindValue(':imprint', $_POST['imprint']);
$statement->bindValue(':status', $_POST['status']);
$statement->bindValue(':size', $_POST['size']);
$statement->bindValue(':desc', $_POST['desc']);
$statement->execute();
```

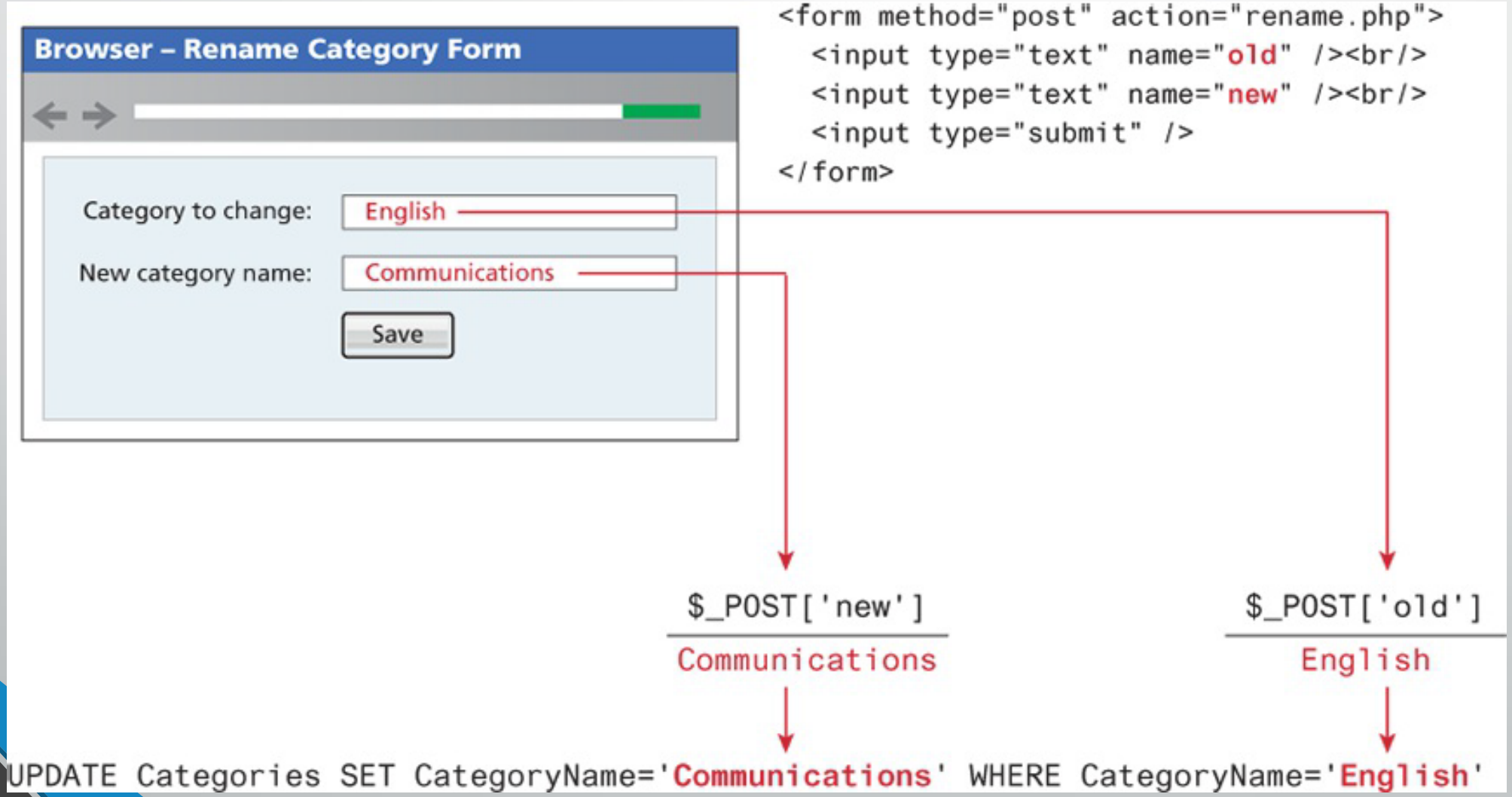
Accessing MySQL in PHP

Working with Parameters – Technique 2 ? Named Parameters with Array

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

Accessing MySQL in PHP

Getting User Input into a Query



Web Systems and Technologies

Accessing MySQL in PHP
Performing Transactions

Accessing MySQL in PHP

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();
}
```

Topic 97 to 99 are examples

WEEK 12

TOPIC 100 TO 108

Web Systems and Technologies

- 100 Example: Database Update
- 101 Error Handling in PHP; Types of Error
- 102 PHP Error Reporting
- 103 PHP Error and Exception Handling
- 104 Regular Expressions
- 105 Regular Expressions - Examples
- 106 PHP Filters
- 107 Validating User Input
- 108 Where to Perform Validation

Example - database

Types of Errors

Types of Errors

- Expected errors
- Warning errors
- Fatal errors

Types of Errors

Notice that this parameter has no value.

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

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

`isset($_GET['id'])`	returns	**true**	
`isset($_GET['name1'])`	returns	**true**	Notice that a missing value for a parameter is still considered to be `isset`.
`isset($_GET['name2'])`	returns	**true**	
`isset($_GET['name3'])`	returns	**true**	
`isset($_GET['name4'])`	returns	**false**	Notice that only a missing parameter name is considered to be not `isset`.
`empty($_GET['id'])`	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.
`empty($_GET['name1'])`	returns	**true**	
`empty($_GET['name2'])`	returns	**false**	
`empty($_GET['name3'])`	returns	**false**	Notice that a value of space is considered to be **not** empty.
`empty($_GET['name4'])`	returns	**true**	

PHP Error Reporting

The error_reporting Setting

- error_reporting specifies which type of errors are to be reported
- Can be configured using

  ini_set(...);

- Or within the php.ini file

The display errors Setting

- The display_error setting specifies whether error messages should or should not be displayed in the browser
- ini_set('display_errors','0');
- It can also be set within the php.ini file:
- display_errors = Off

The log_errors Setting

- The log_errors setting specifies whether error messages should or should not be sent to the server error

- ini_set('log_errors','1');
- It can also be set within the php.ini file:
- log errors = On

PHP Error and Exception Handling

Exceptions

- An exception refers to objects that are of type Exception and which are used in conjunction with the object- oriented try ... Catch language construct for dealing with runtime errors.

Procedural Error Handling

```
$connection = mysqli_connect(DBHOST,
DBUSER, DBPASS, DBNAME);
$error = mysqli_connect_error();
if ($error != null) {
  // handle the error
  ...
}
```

Object-Oriented Exception Handling

```
function throwException($message = null,$code = null) {
  throw new Exception($message,$code);
}
try {
  $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 {
  ...
}
```

Regular Expressions

Introduction

- A regular expression is a set of special characters that define a pattern
- Regular expressions are a concise way to eliminate the conditional logic that would be necessary to ensure that input data follows a specific format
- PHP, JavaScript, Java, the .NET environment, and most other modern languages support regular expressions (each slightly different)

Regular Expression Syntax

- A literal is just a character you wish to match in the target
- A metacharacter is a special symbol that acts as a command to the regular expression parser
  - . [ ] ( ) ^ $ | * ? { } +
- Regular Expression Patterns can be combined to form complex expressions

Regular Expression Syntax Patterns

- ^qwerty$ 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.

Regular Expression Syntax Patterns

- [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.

Regular Expression Syntax Patterns

- \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.

Regular Expression Syntax Patterns

- {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.

Regular Expressions - Examples

Examples

- CNIC
- Email Addresses
- Mobile numbers in Pakistan

PHP Filters

Introduction

- Filters in PHP validate input against predefined patterns
  - Validate filters
  - Sanitize filters
- Use the function
  - filter_var($value, FILTER)

Validate Filters

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 filteres
  - FILTER_VALIDATE_BOOL, FILTER_VALIDATE_FLOAT, FILTER_VALIDATE_INT
  - FILTER_VALIDATE_MAC
  - FILTER_VALIDATE_IP
  - FILTER_VALIDATE_URL
  - FILTER_VALIDATE_REGEXP
  - ...

Validate Filters

- Validate against a regular expression (FILTER_VALIDATE_REGEXP)

Validate Filters

- Validate against a regular expression (FILTER_VALIDATE_REGEXP)

```
$cnic = filter_var('12345-1234567-8', FILTER_VALIDATE_REGEXP,
        array("options" => array("regexp" => "/^\\d{5}-?\\d{7}-?\\d$/")));
if($cnic == null) {
  echo "Invalid CNIC";
} else {
  echo "Valid CNIC $cnic";
}
```

Validate Filters

- Validate against a regular expression (FILTER_VALIDATE_REGEXP)

```
$cnic = filter_var('12345-1234567-8', FILTER_VALIDATE_REGEXP,
        array("options" => array("regexp" => "/^\\d{5}-?\\d{7}-?\\d$/")));
if($cnic == null) {
  echo "Invalid CNIC";
} else {
  echo "Valid CNIC $cnic";
}
```

Sanitize Filters

- Remove or replace specific characters

- FILTER_SANITIZE_EMAIL
- FILTER_SANITIZE_STRING
- FILTER_SANITIZE_FULL_SPECIAL_CHARS

Sanitize Filters

- Remove or replace specific characters
- FILTER_SANITIZE_EMAIL
  - Removes all characters except letters, digits and !#$%&'*+-=?^_`{|}~@.[].
  - 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" ' B '
- FILTER_SANITIZE_FULL_SPECIAL_CHARS
  - filter_var('<h1>A" \' B</h1>',FILTER_SANITIZE_FULL_SPECIAL_CHARS) == '<h1>A" ' B</h1>'

Sanitize Filters

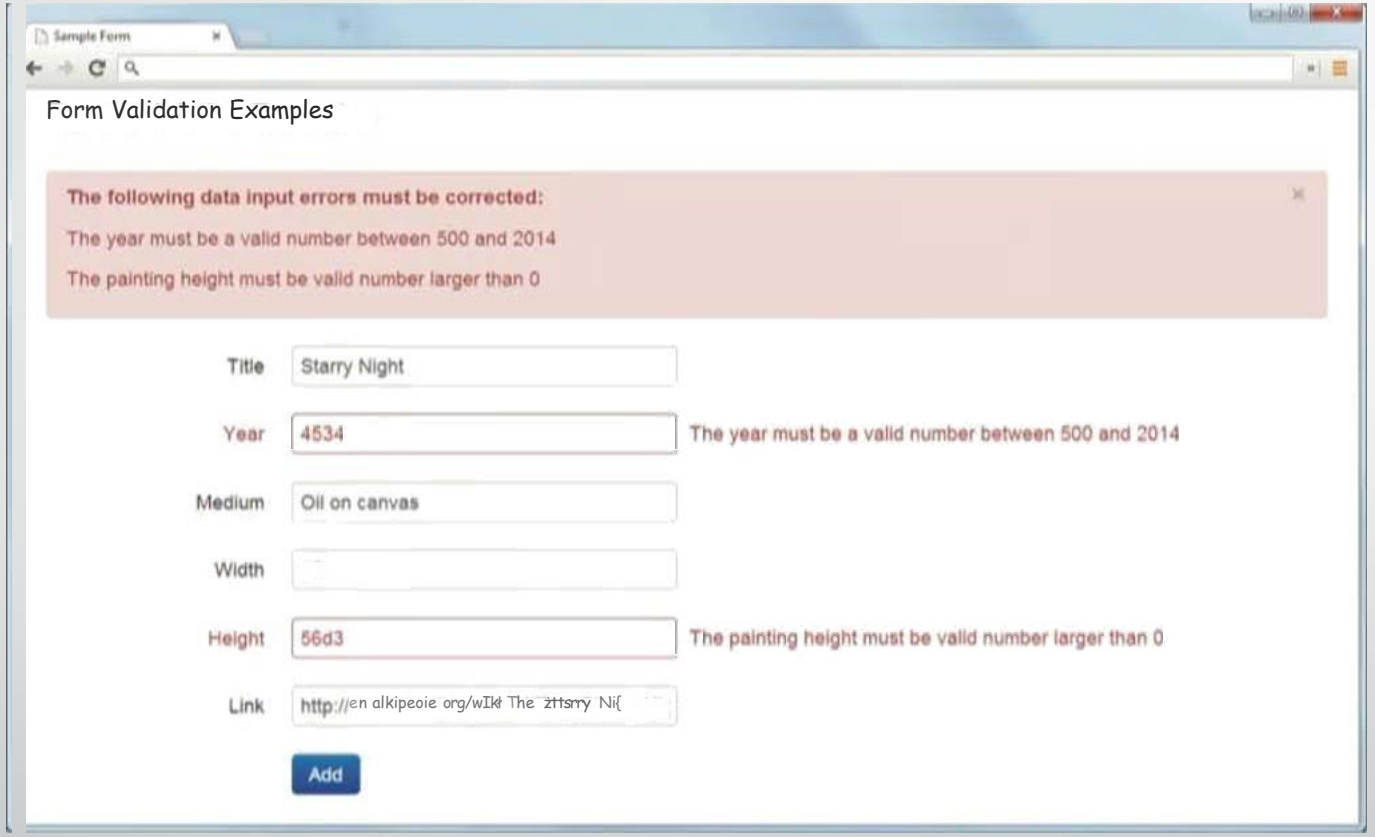
- FILTER_SANITIZE_ADD_SLASHES
- FILTER_SANITIZE_ ENCODED
- FILTER_SANITIZE_ NUMBER_FLOAT
- FILTER_SANITIZE_ NUMBER_INT
- FILTER_SANITIZE_ SPECIAL_CHARS
- FILTER_SANITIZE_ URL

Validating User Input

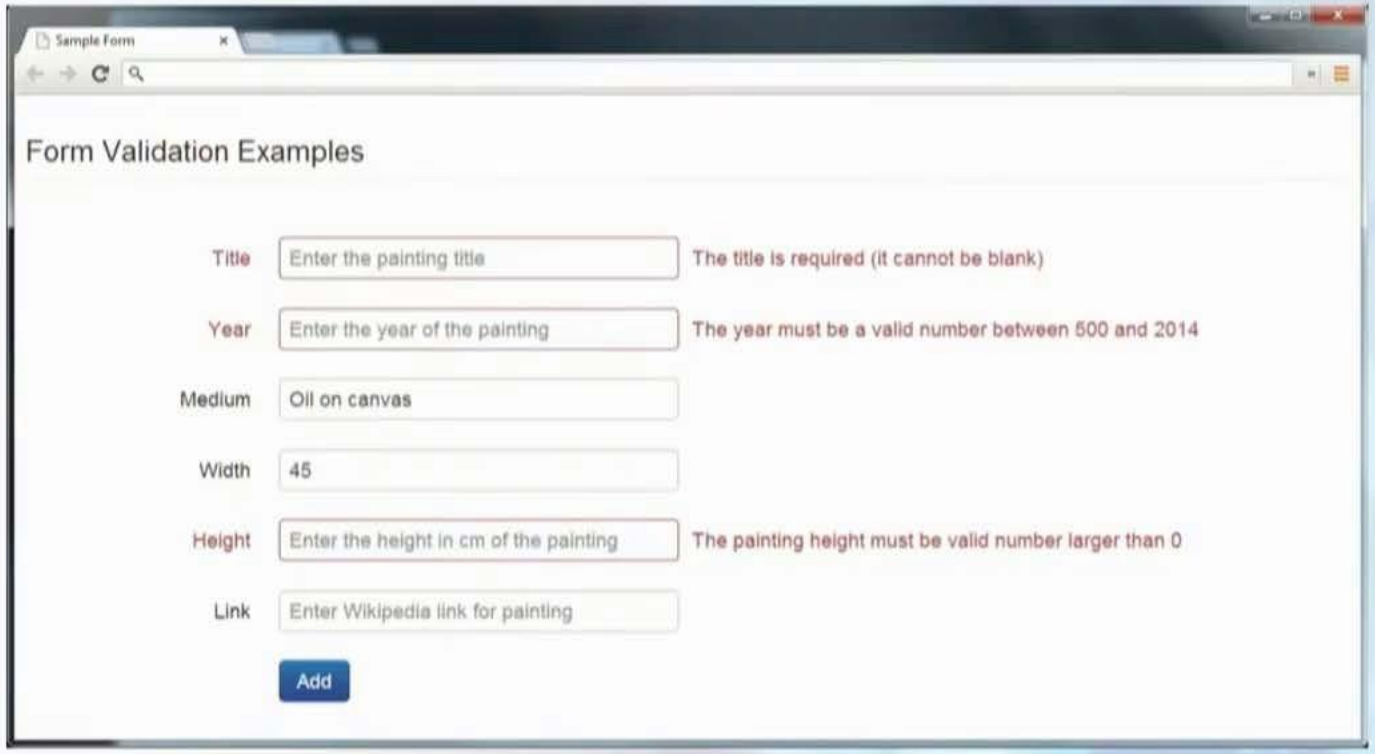
Types of Input Validation

- Required information
- Correct data type
- Correct format
- Comparison
- Range check
- Custom

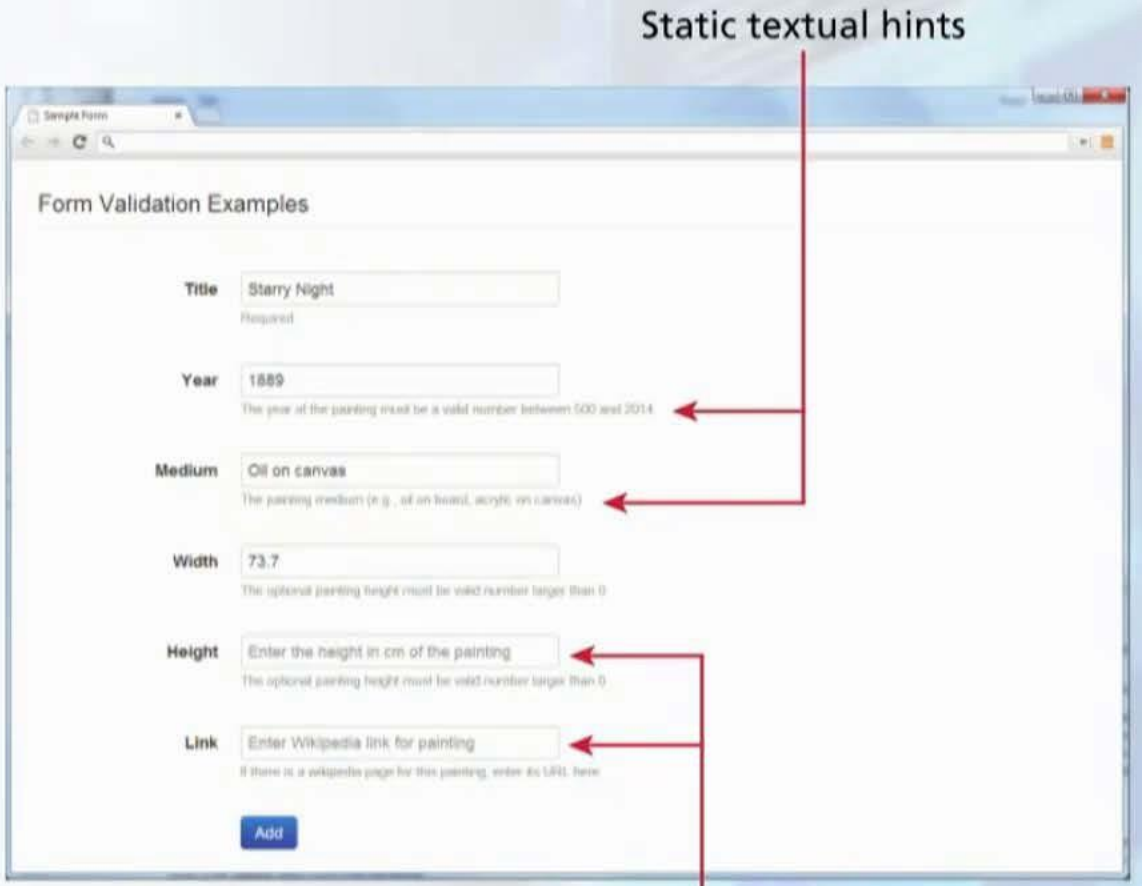
Notifying the user



How to Reduce Validation Errors – Show Where Error Located



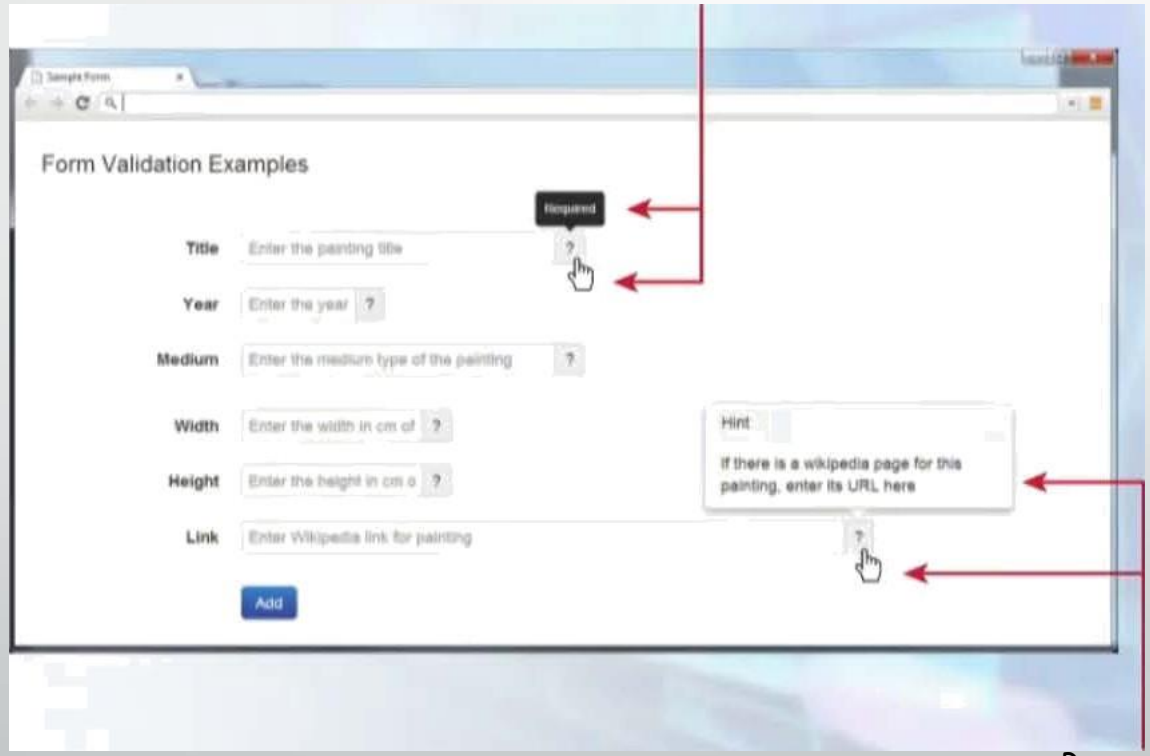
How to Reduce Validation Errors – Providing Textual Hints




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

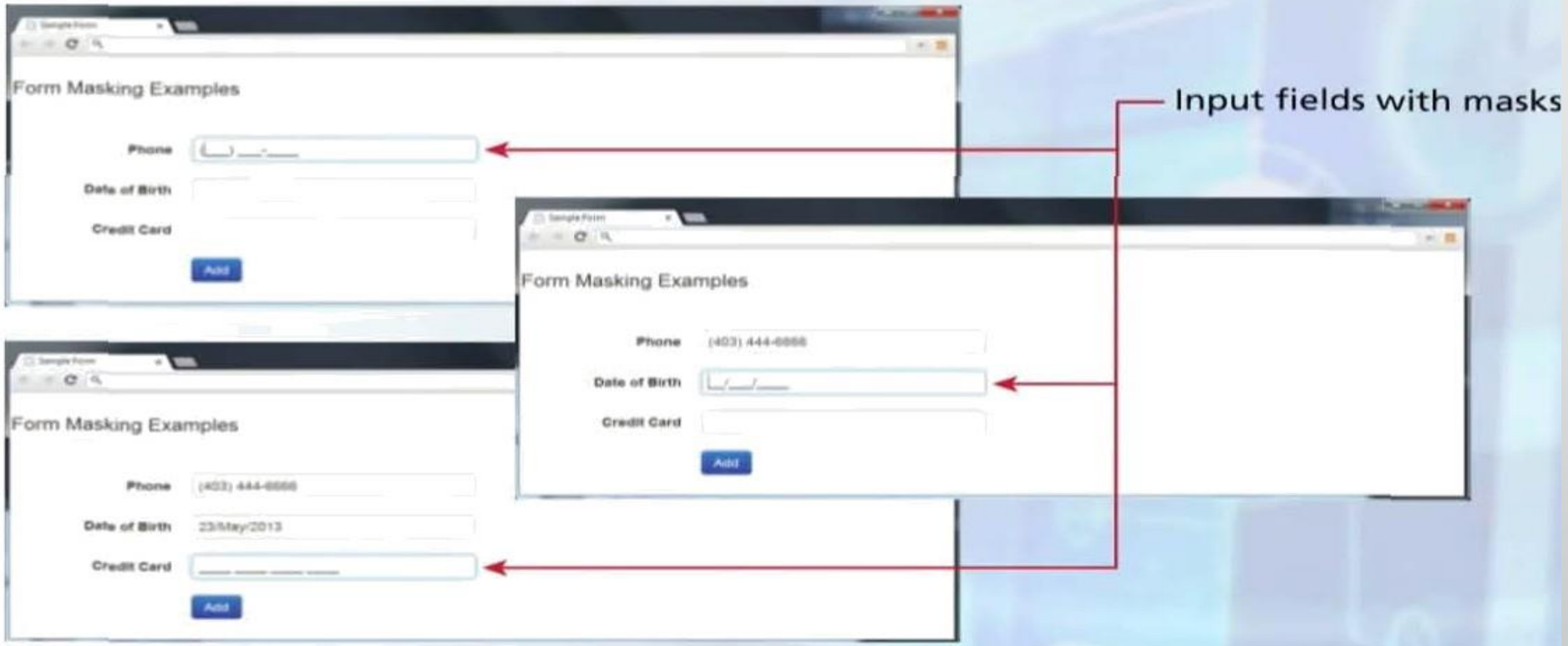
HOW TO REDUCE VALIDATION ERRORS- USE TOOLS TIPS

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



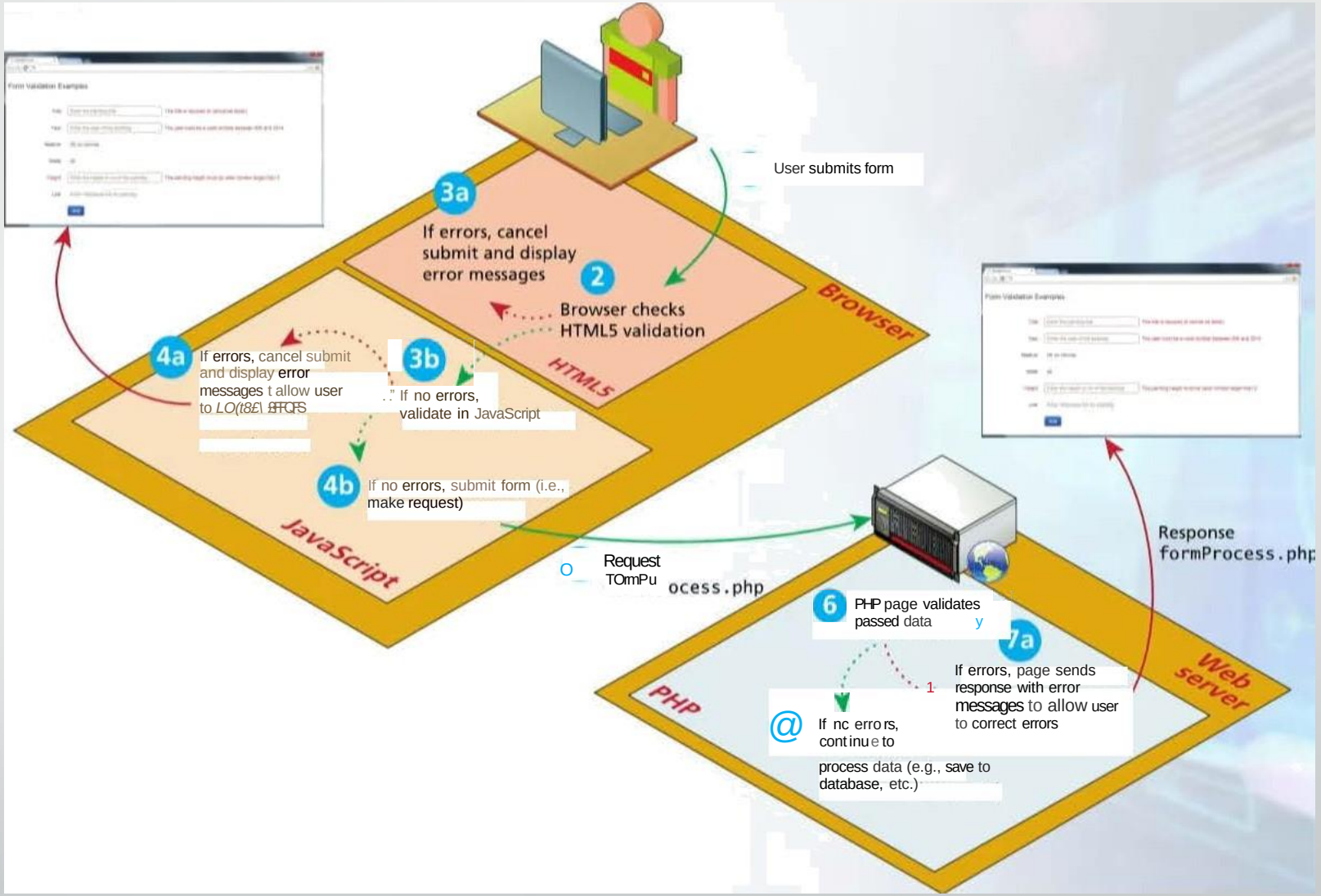
Pop-over

How to Reduce Validation Errors Use Input Masks



Where to Perform Validation?

Bigger Picture



Validation at the JavaScript Level

- Client process
- Can reduce server load
- Can be bypassed

Validation at the PHP Level

- Validation on the server side using PHP is the most important form of validation and the only one that is absolutely essential

WEEK 13

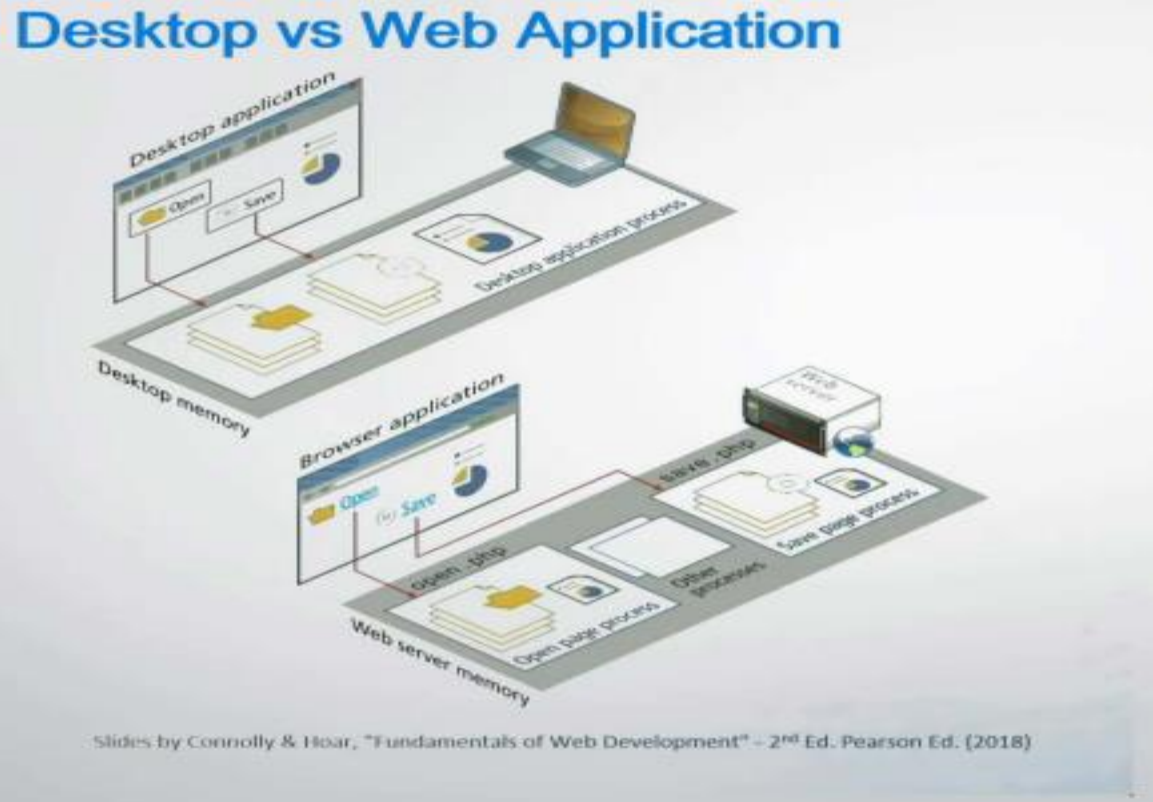
TOPIC 109 TO 117

Web Systems and Technologies

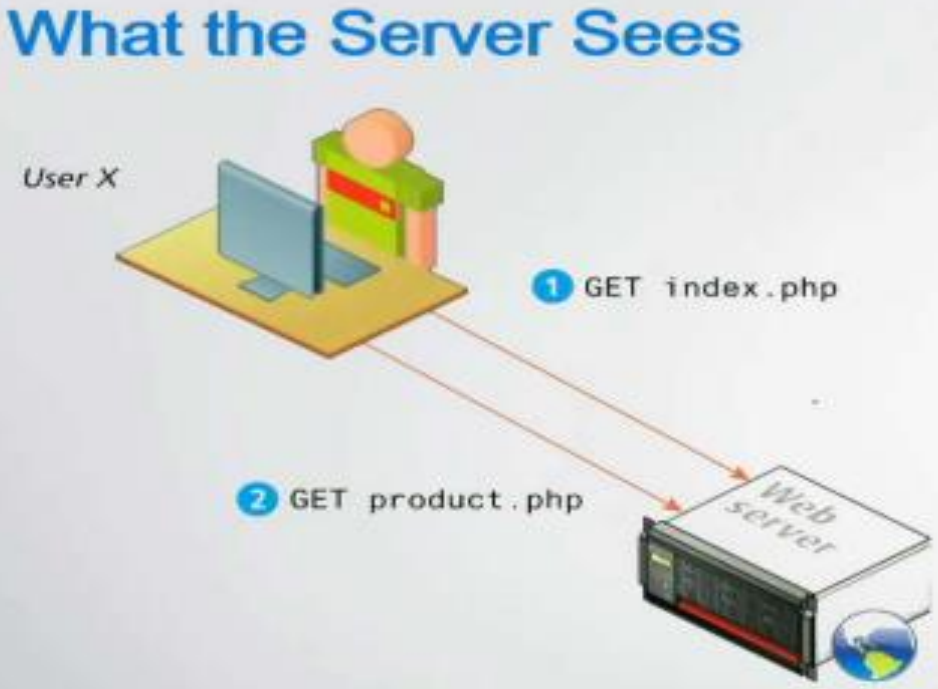
- 109 State Management on Server Side
- 110 State Management Using Query String
- 111 Cookies
- 112 Cookies - Example
- 113 Serialization
- 114 Session State
- 115 Security Principles
- 116 Security by Design
- 117 Authentication

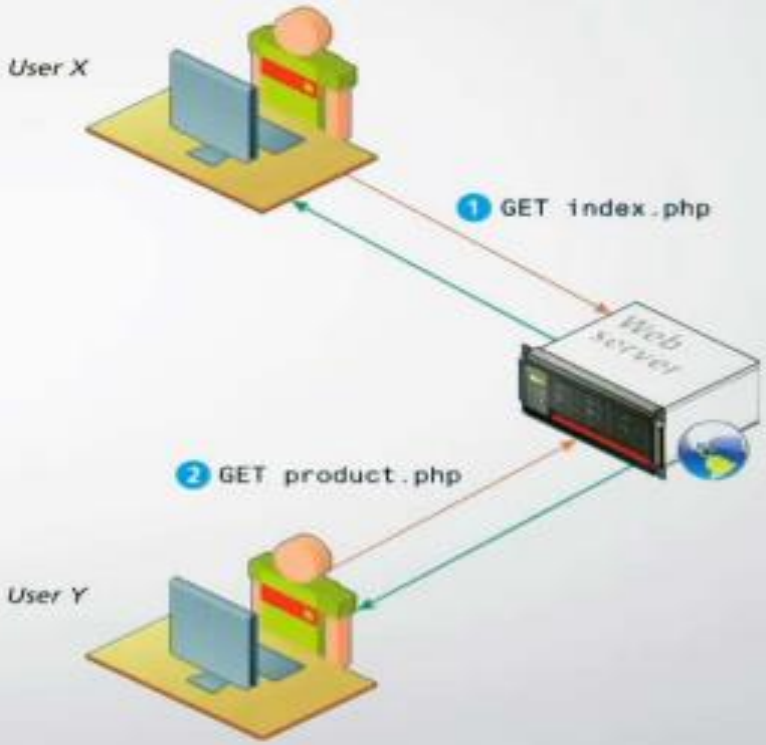
The Problem of state Management

Desktop vs Web Application



What the Sewer Sees





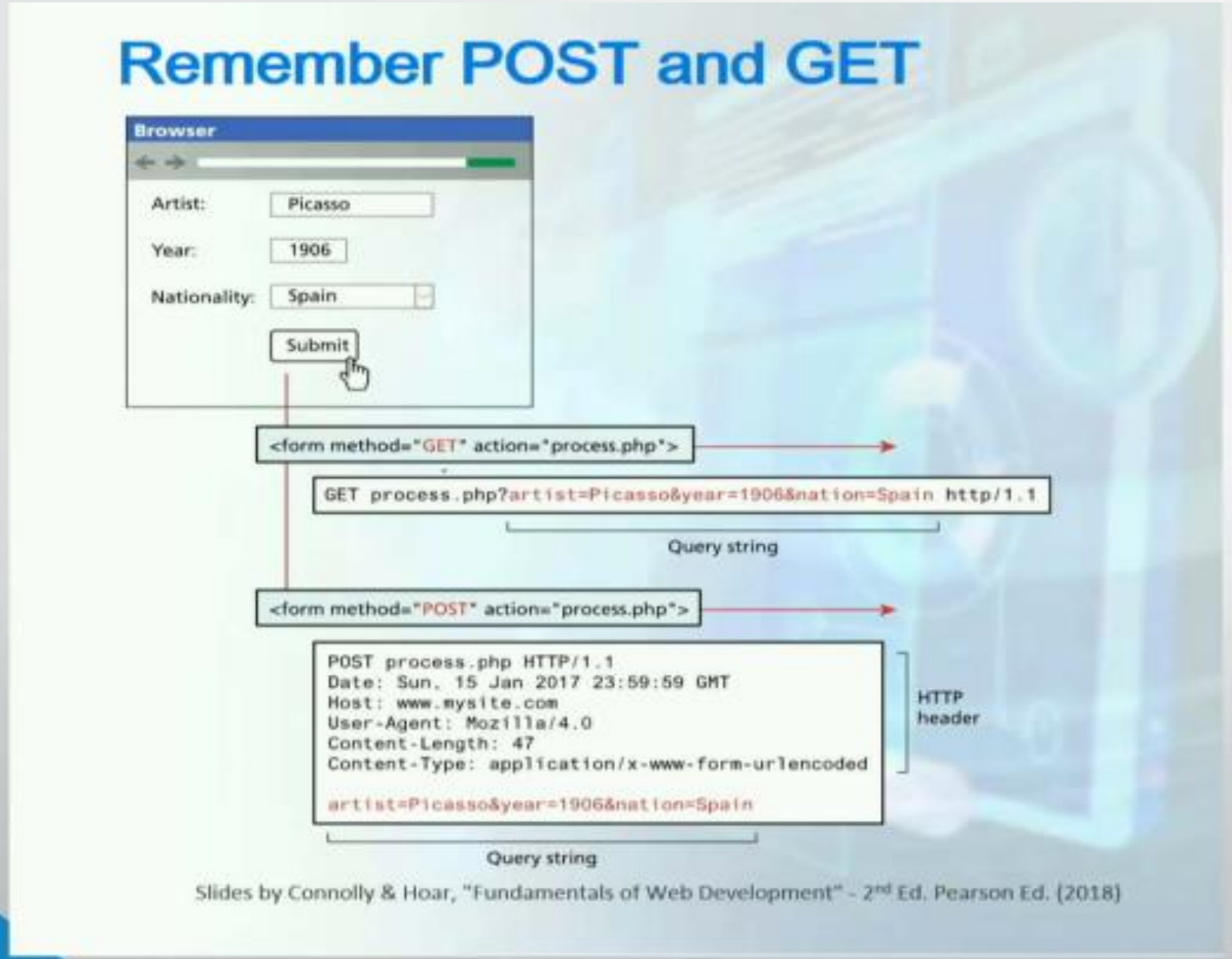
What the User Wants

Solution

- Query Strings
- Cookies
- Sessions

Passing Information via Query Strings

Remember POST and GET



Dynamic vs S atic 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 pösition.
  - Another benefit to static URLs is that users tend to prefer them.
- Can use URL rewriting to make static URLs be dynamic

Passing Information via Query Strings

URL Rewriting in Apache

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.html

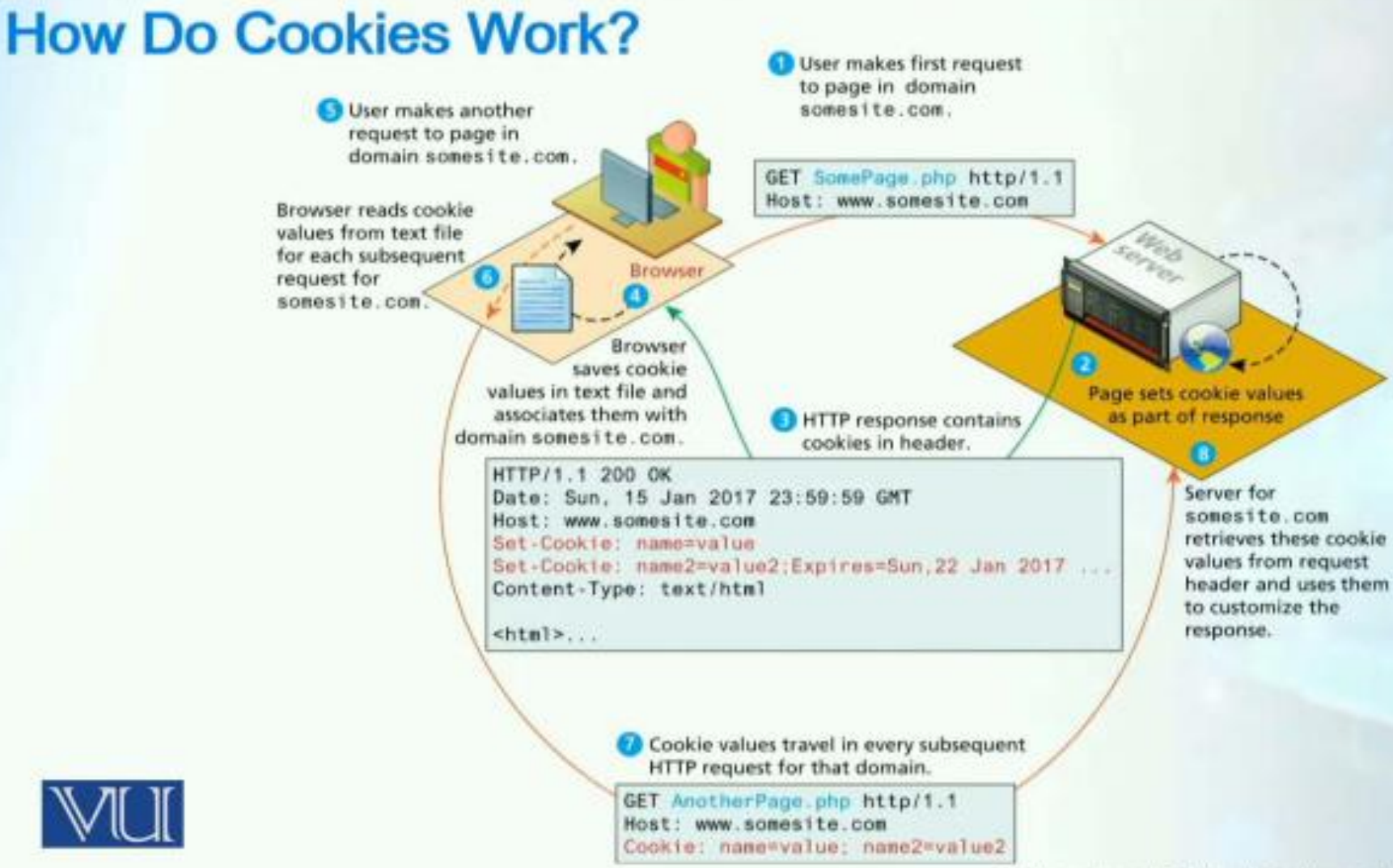
http://www.artsheaven.com/raphael-la-donna-velata.html

http://www.paintingswholesaler.com/detail.asp?vcode=6umd7krr1yqi161c&title=La+Donna+Velata

Cookies

Cookies

How Do Cookies Work?



Persistent Cookie Best Practices

- Use cookies for improving user experience
  - User preferences
  - User behavior
- The website functionality should not depend upon cookies
- If using cookies for login information, back it up with a token on each login, saved on the server as well

Setting a Cookie

```
<?php
// add 1 day to the current time
expiry time
$expiryiime   time()+60*60*24;
   create a persistent cookie
$name    "Username";
$value = "Ricardo";
setcookie($name, $value, $expiryTime);
?>
```

Reading a Cookies

```
<?php
If (!isset($ COOKIE['Username'] valid cookie found


else
  echo ”The username retrieved from the
cookie is:".$ COOKIE['Username' ;

}
?>
```

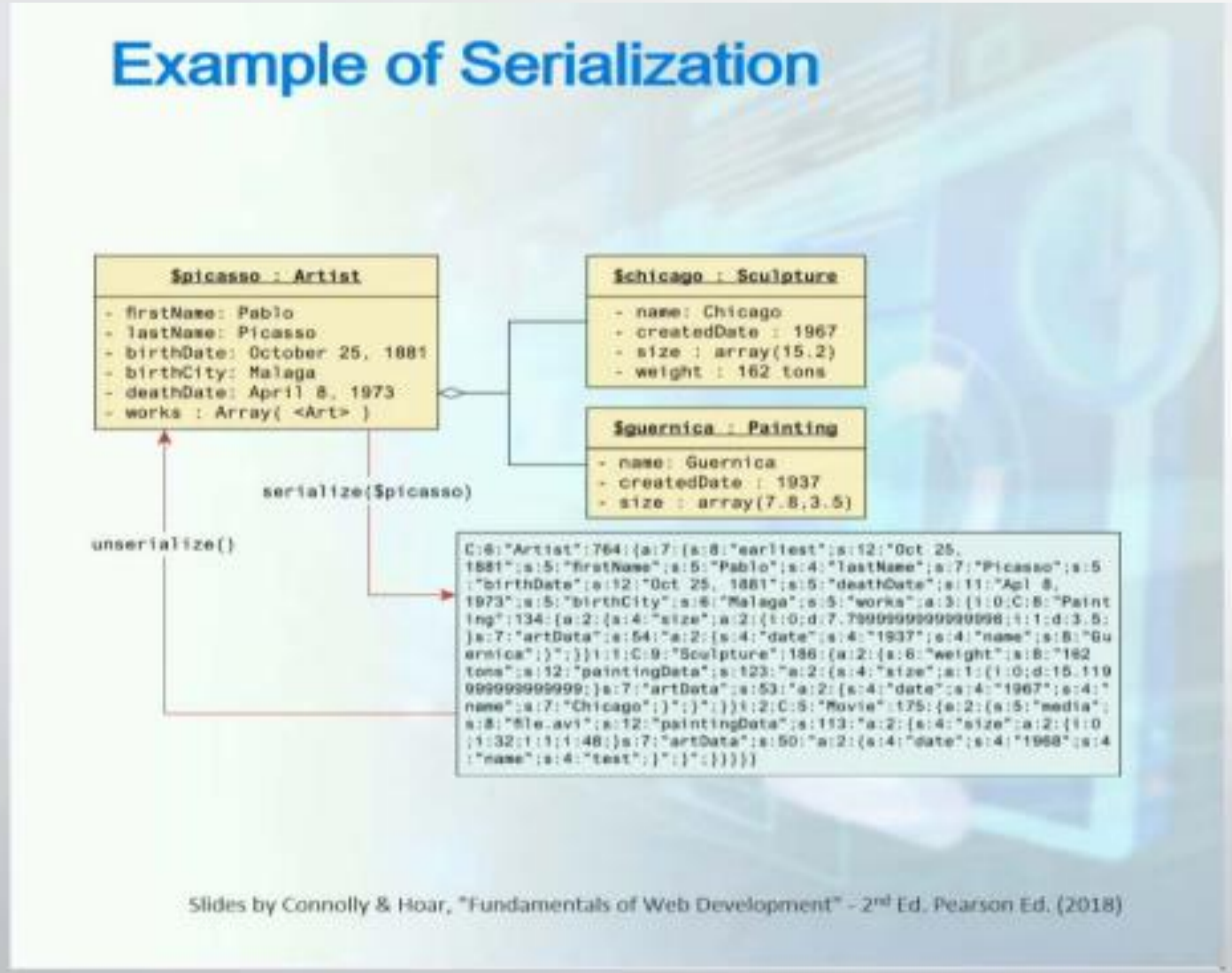
Cookies – Example

Serialization

Introduction

- Serialization converts an object into binary form
- Used for storage and transmission
- Unserialization converts the serialized data back into an object
- Can be used to manage states
  - Serialize data at the end of the request
  - Unserialize at the start of the next request

Example of Serialization



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

Serialization in PHP

- PHP built –in data types and object are serializable
- Custom objects need to implement the serializable interface

Example

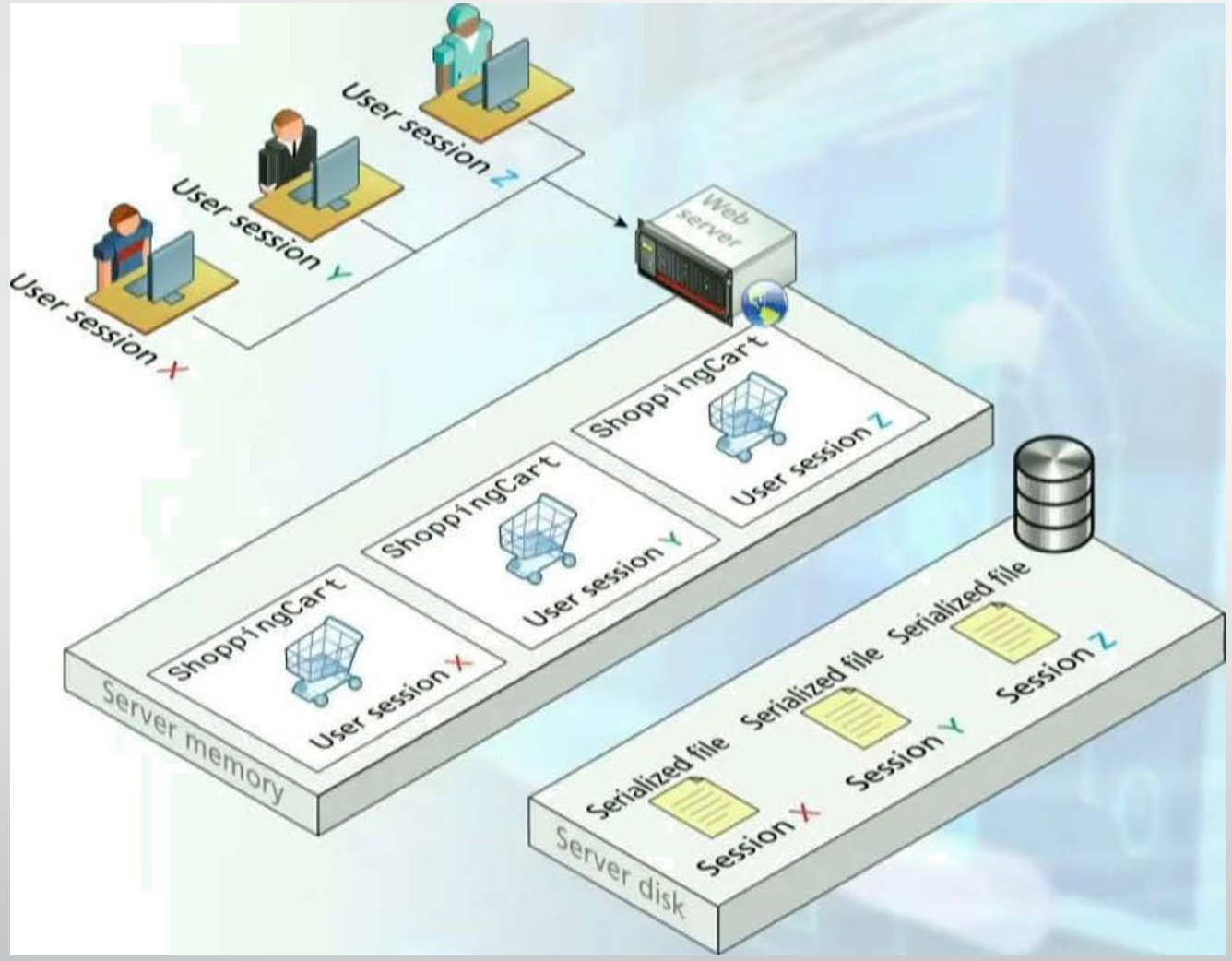
Example

```
class Artist implements Serializable {
  public function serialize() {
  return serialize(
  array("earliest" =>self::$earliestDate,
              "first" => $this->firstName,
              "last" => $this->lastName,
              "bdate" => $this->birthDate,
              "ddate" => $this->deathDate,
              "bcity" => $this->birthCity,
              "works" => $this->artworks
        )
    );
}
    ...
```

```
class Artist implements Serializable {
...
public function unserialize($data) {
  $data = unserialize($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'];
}
...
}
//for serialization
$serialized = serialize($object);

//for unserialization
$objectClone = unserialize($serialized);
```

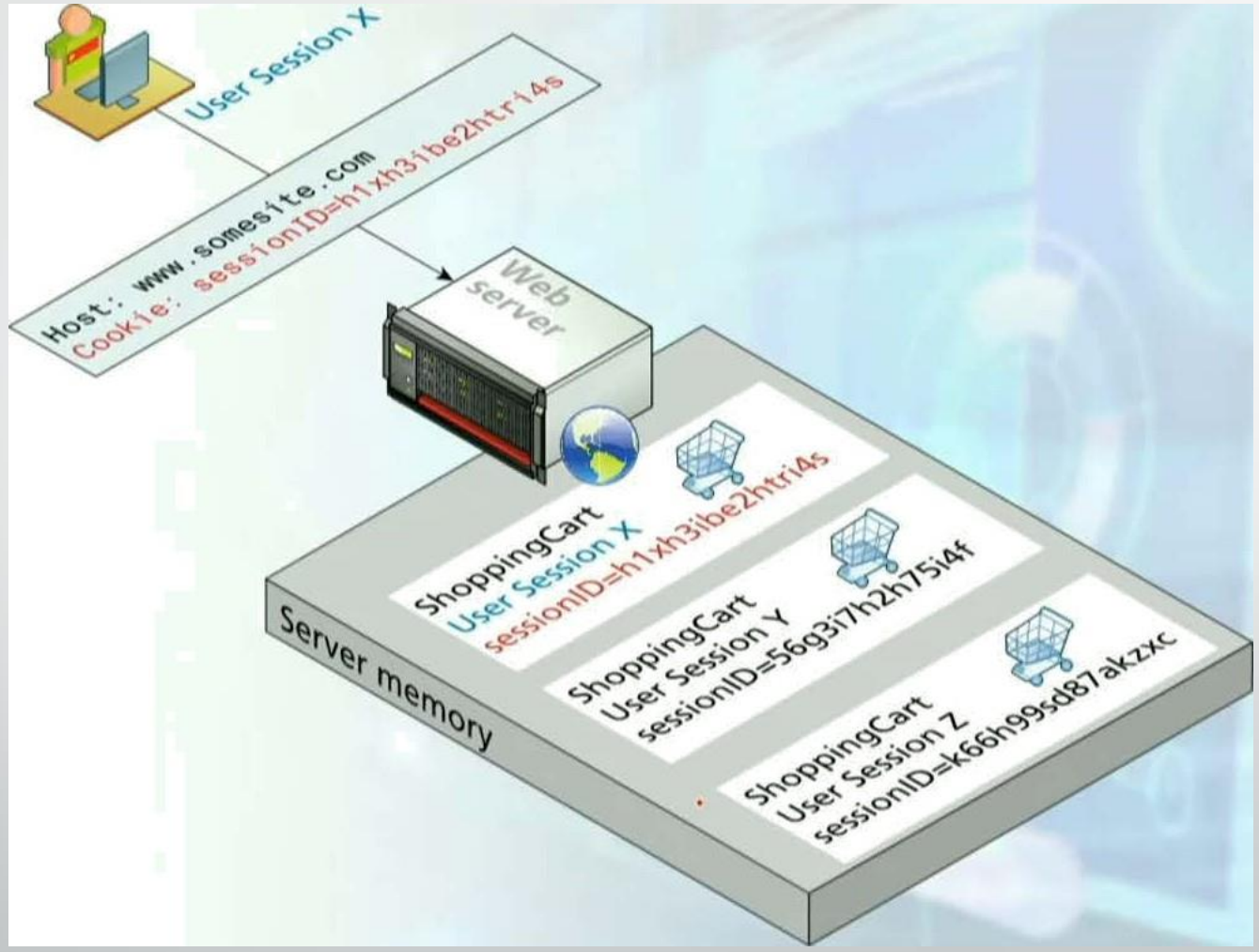
Session State

How Does Session state work?

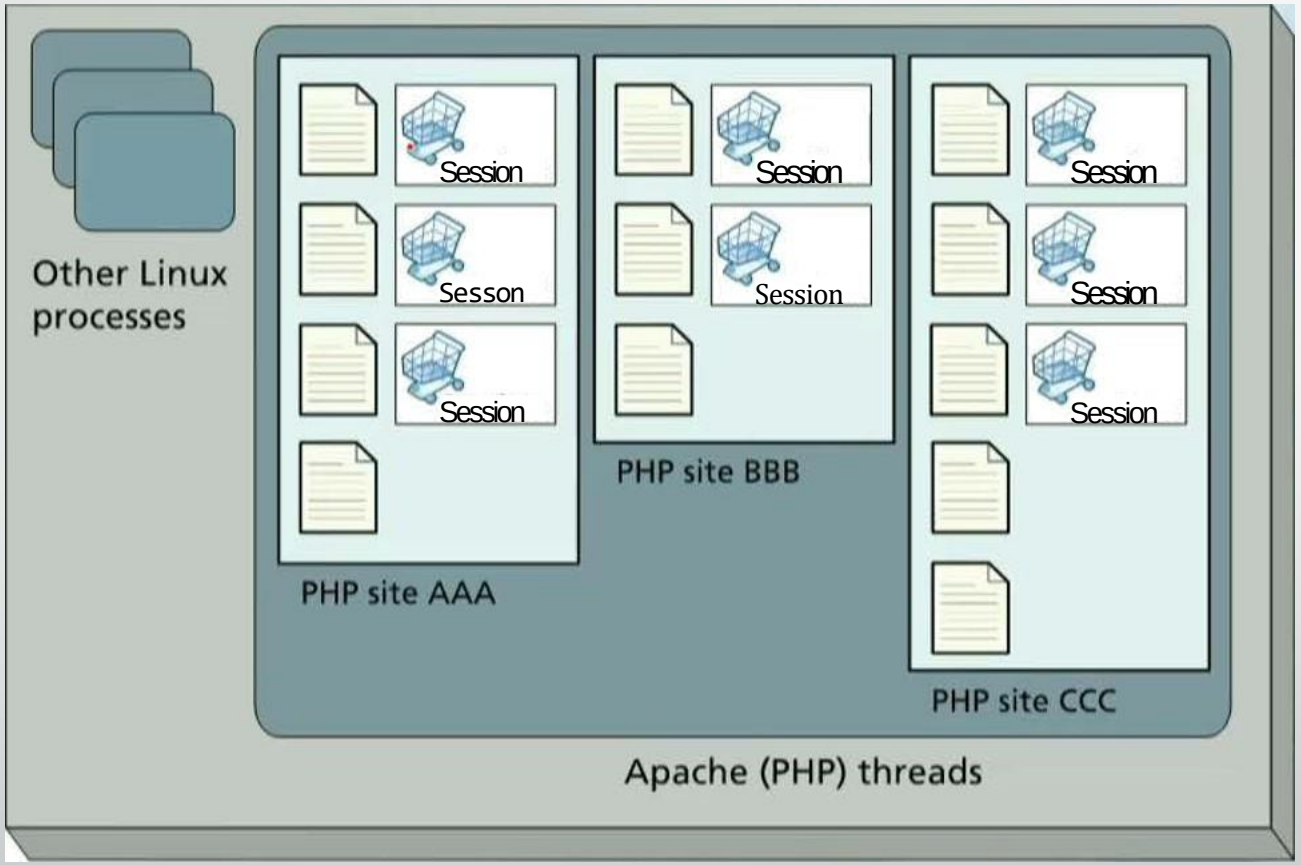


SlfTes by Com›ully & I-luar, "Fundamentals of Web Development" - 7 * Ed. Pearson Ed, (sofa)

How does it all work



Session Storage and configuaration



Server memory

Accessing session state

Accessing session state

```
<?php
session_start();
if ( isset($_SESSION['user']) ) {
  // User is logged in
}
else {
  // No one is logged in (guest)
}
?>
```

Checking session existence

Checking session existence

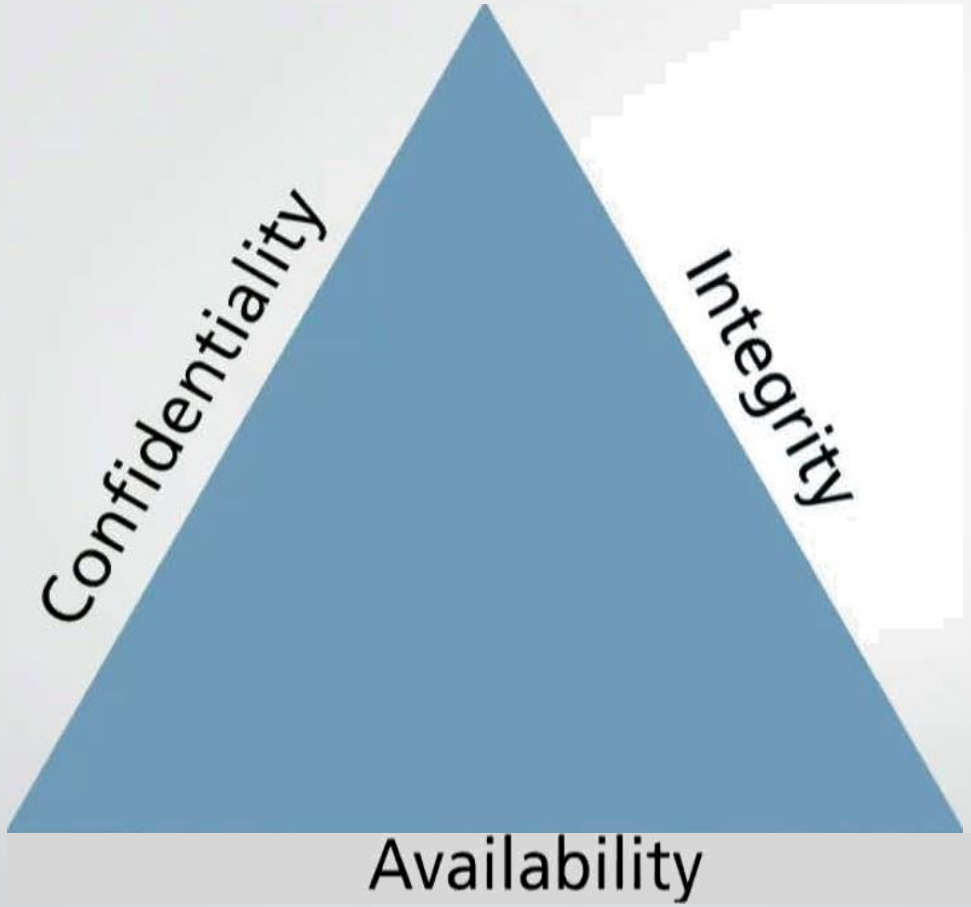
```php
<?php
include_once("ShoppingCart.class.php");
session_start();
// always check for existence of
//session object before accessing it
if ( !isset($_SESSION["Cart"]) ) {
  // session variables can be strings,
  //arrays, or objects, but smaller is better
  $_SESSION["Cart"] = new ShoppingCart();
}
$cart = $_SESSION["Cart"];
?>
```

Terminating a session

- Removing session variables
  - Session_unset()
- Destroying the session
  - session_destroy();

Security Principles

Information security



Risk Assessment and Management

- Actors, Impact, Threats, and Vulnerabilities

Actors

- Internal actors are the people who work for the organization. They can be anywhere in the organization from the cashier through the IT staff, all the way to the CEO.
- External actors are the people outside of the organization. They have a wide range of intent and skill, and they are the most common source of attacks.
- Partner actors are affiliated with an organization that you partner or work with. If your partner is somehow compromised, there is a chance your data is at risk as well because quite often partners are granted some access to each other's systems (to place orders, for example).

Impact

- A loss of availability prevents users from accessing some or all of the systems.
- A loss of confidentiality includes the disclosure of confidential information to a (often malicious) third party
- A loss of integrity changes your data or prevents you from having correct data. This might manifest as an attacker hijacking a user session, perhaps placing fake orders or changing a user's home address.

Security Principles

Threats

Broadly, threats can be categorized using the STRIDE mnemonic

- **Spoofing—**Theattacker uses someone else's information to access the system.
- **Tampering—**The attacker modifies some data in nonauthorized 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 are the security holes in your system. The top five classes of vulnerability from the Open Web Application Security Project are:
    1. Injection
    2. Broken authentication and session management
    3. Cross-site scripting
    4. Insecure deserialization
    5. Security misconfiguration

Security Policy

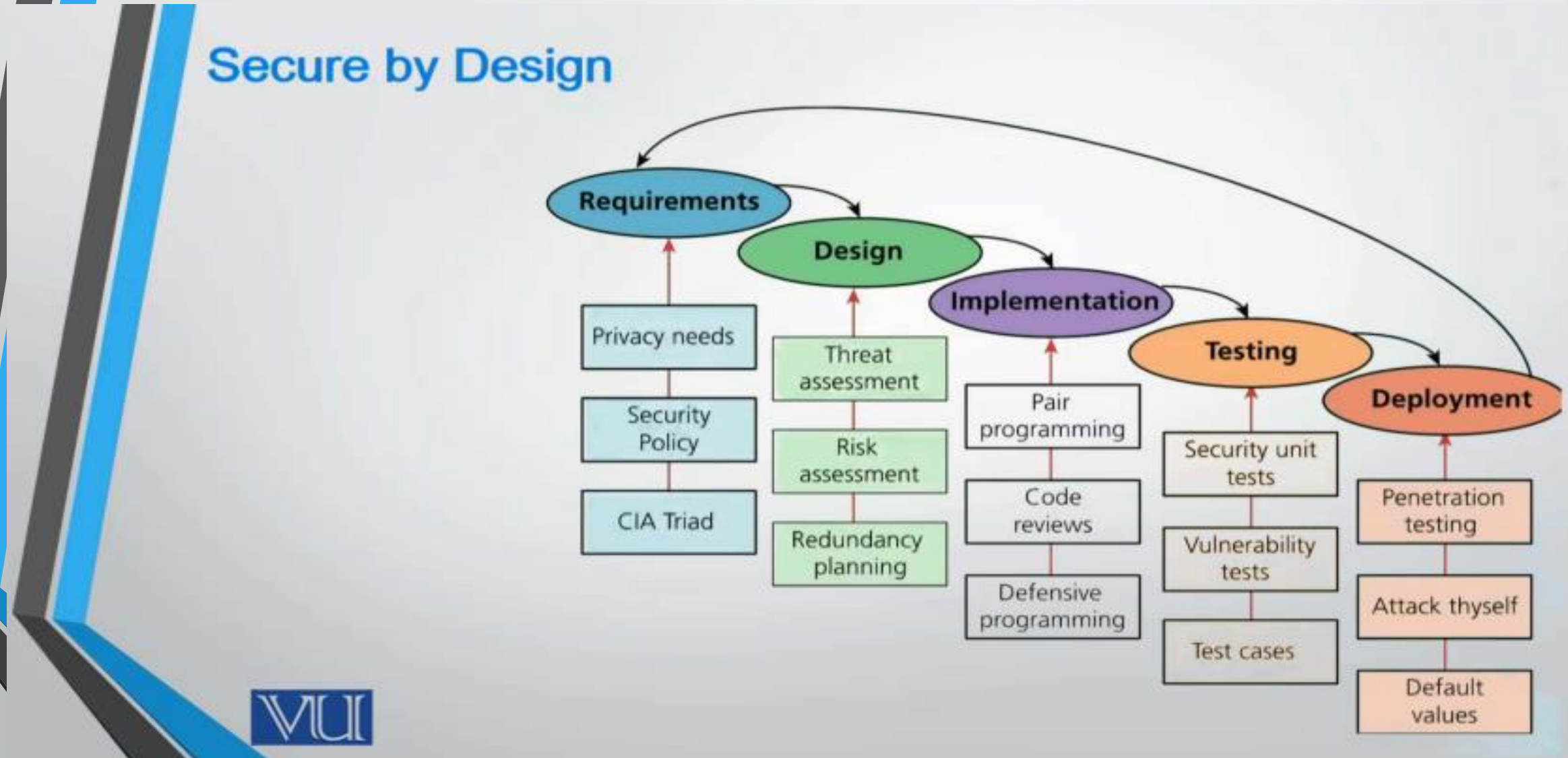
- Usage policy defines what systems users are permitted to use, and under what situations.
- Authentication policy controls how users are grantecl access to the systems.
- Legal policies define a wide range of things including data retention and backup policies as well as accessibility requirements (like having all public communication well organized for the blind).

Business Continuity

- Admin Password Management
- Backups and Redundancy
- Geographic Redundancy
- Stage Mock Events
- Auditing

Security by Design

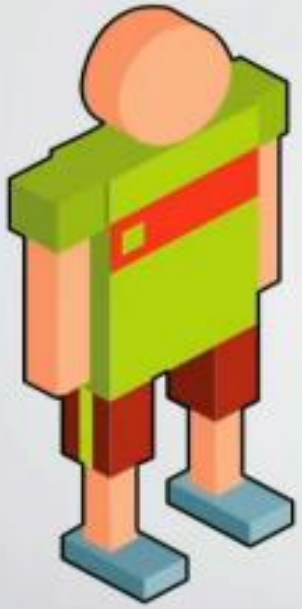
Secure by Design



Authentication

Authentication Factors

Authentication Factors



What you know (Knowledge)

Passwords, PIN, security questions, ...

What you have (Ownership)

Access card, cell phone, cryptographic FOB, ...

What you are (Inherence)

Retinas, fingerprints, DNA, walking gait, ...

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

Authentication Factors

- Single-factor authentication is the weakest and most common category of authentication system where you ask for only one of the three factors.
- Multifactor authentication is where two distinct factors of authentication must pass before you are granted access.

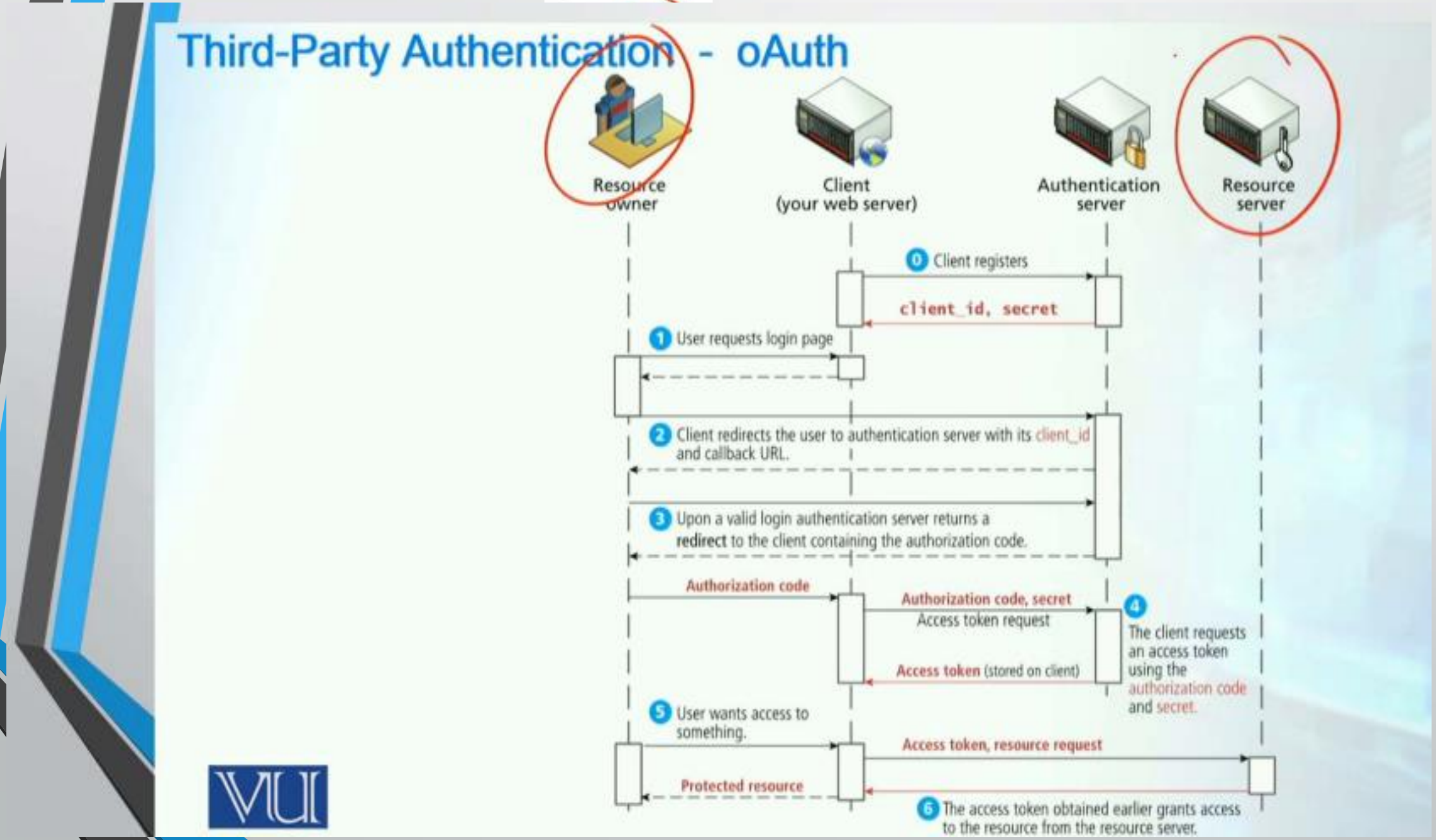
HMP Authentication

- HTTP supports several diXerent forms of authentication via the www- authenticate response header.
- HTTP Basic Authentication
- HTTP Digest Authentication
- Form-Based Authentication

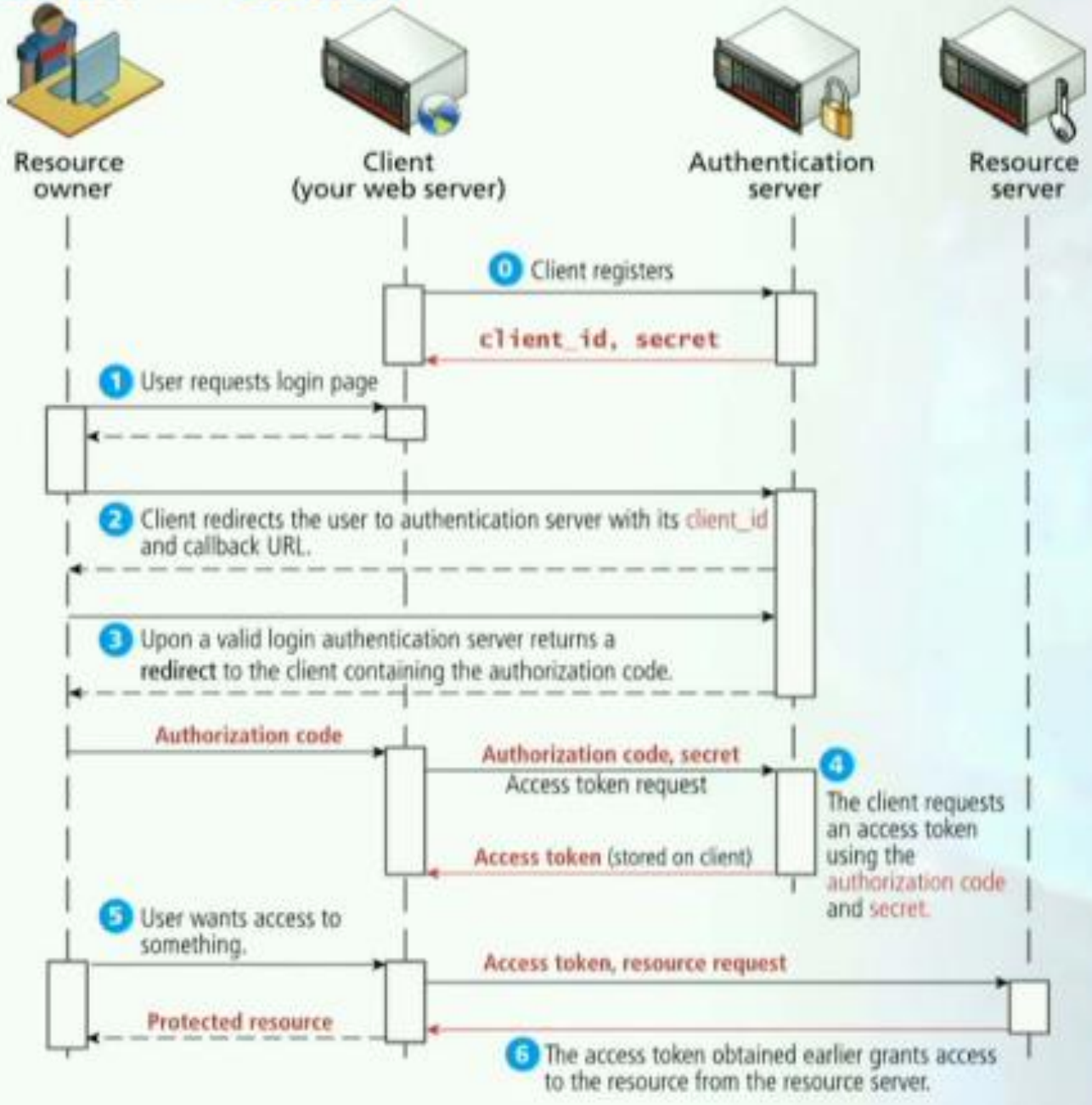
Third-Party Authentication

- Many popular services allow you to use their system to authenticate the user and provide you with enough data to manage your applicatiön

Third-Party Authenticaton - oAuth



Third-Party Authentication - oAuth



Authorization

Authorization defines what rights and privileges a user has once they are authenticated. Some principles:

- Using a separate database user for read and write privileges on a database.
- Providing each user an account where they can access their own files securely.
- Setting permissions correctly so as to not expose files to unauthorized users.
- Using Unix groups to grant users permission to access certain functionality rather than grant users admin access.
- Ensuring Apache is not running as the root account (i.e., the account that can access everythino).

WEEK 14

TOPIC 118 TO 126

Web Systems and Technologies

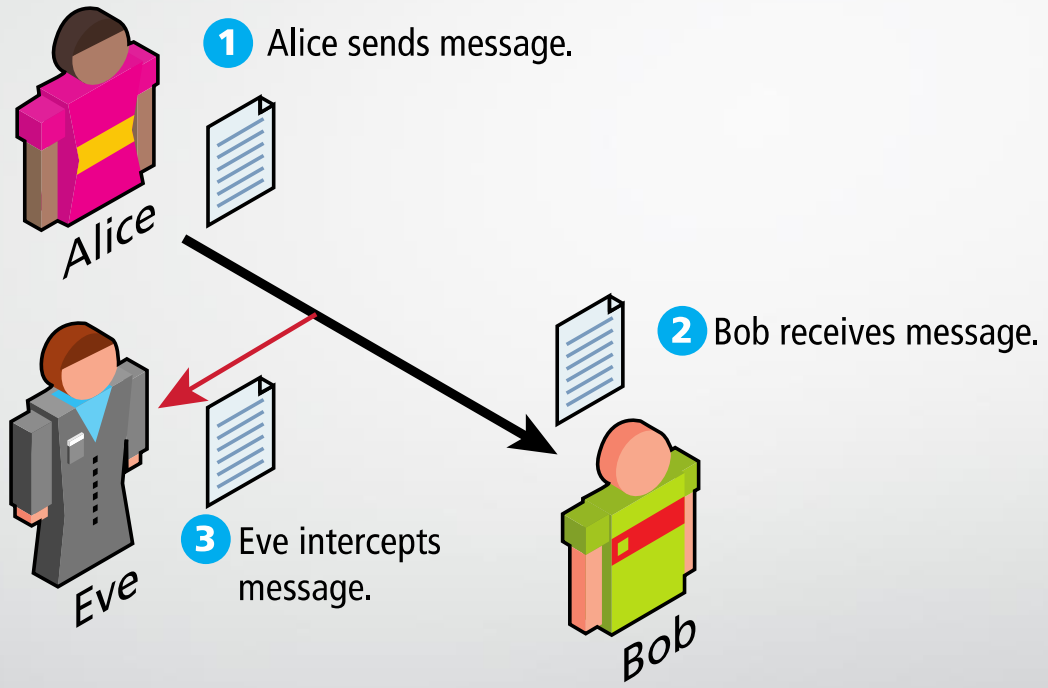
- 118 Cryptography, Cyphers, Digital Signatures
- 119 HTTPS
- 120 XML overview, XML Parser; XML schema
- 121 XSLT and XPath
- 122 XML Processing; XML in JavaScript; XML in PHP
- 123 JSON
- 124 Overview of Web Services
- 125 Consuming Web Services in PHP
- 126 Creating a Web Service in PHP

Web Systems and Technologies

Cryptography

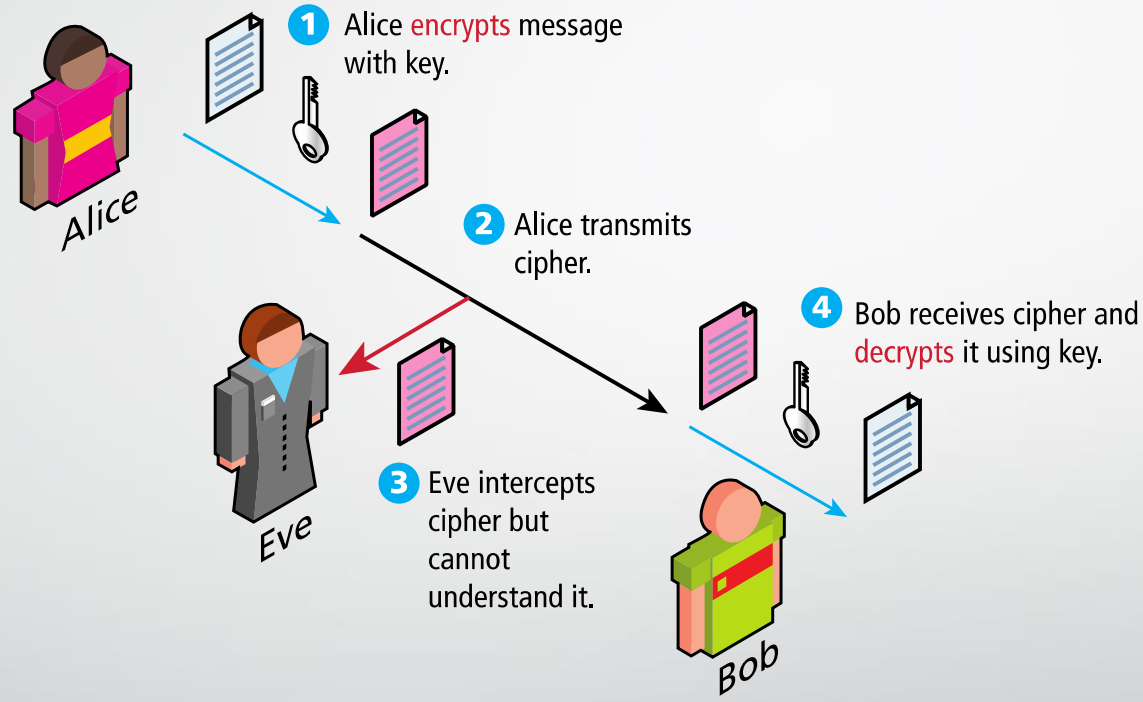
Cryptography

An Overview



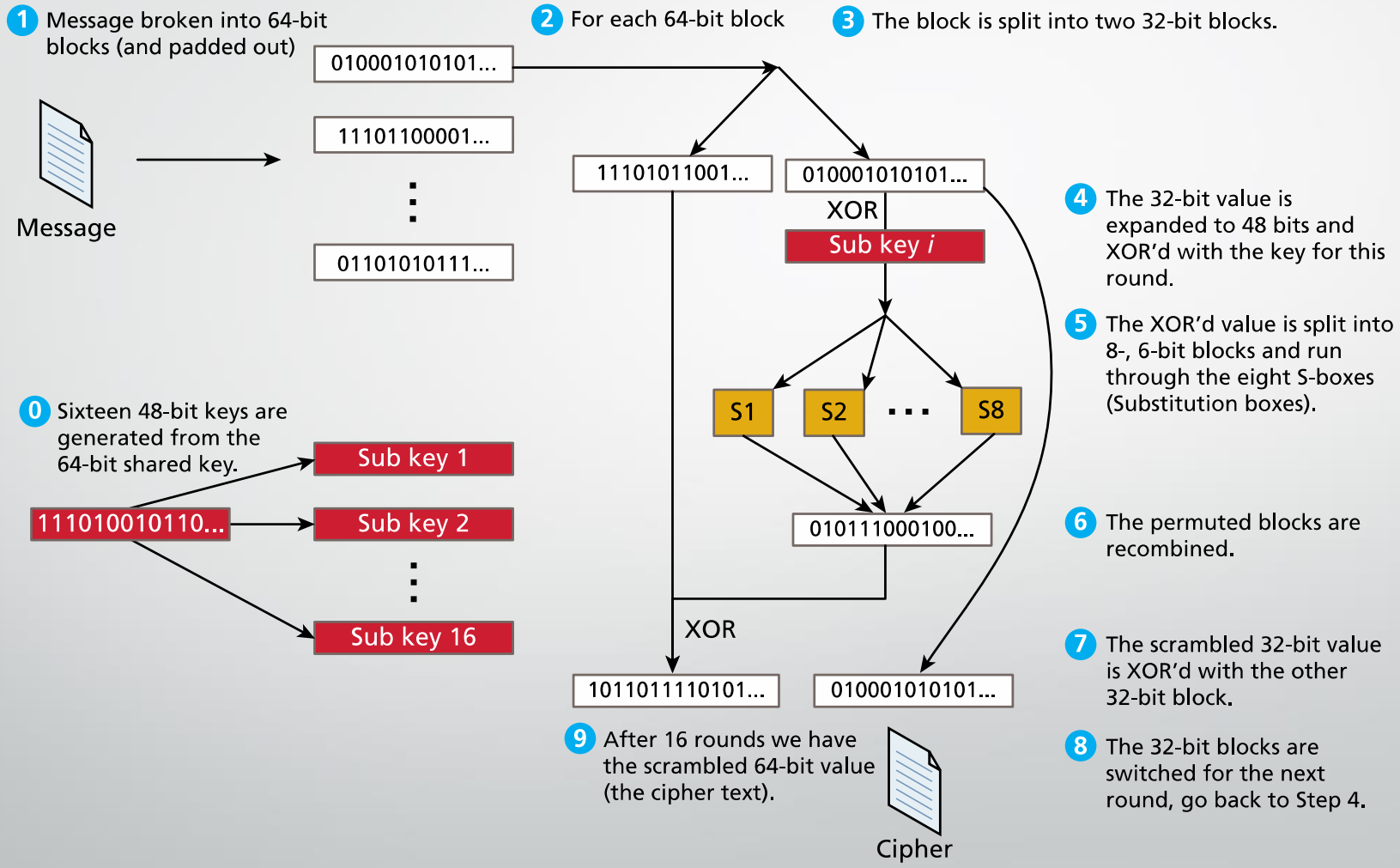
Cryptography

Symmetric Encryption



Cryptography

Modern Block Ciphers



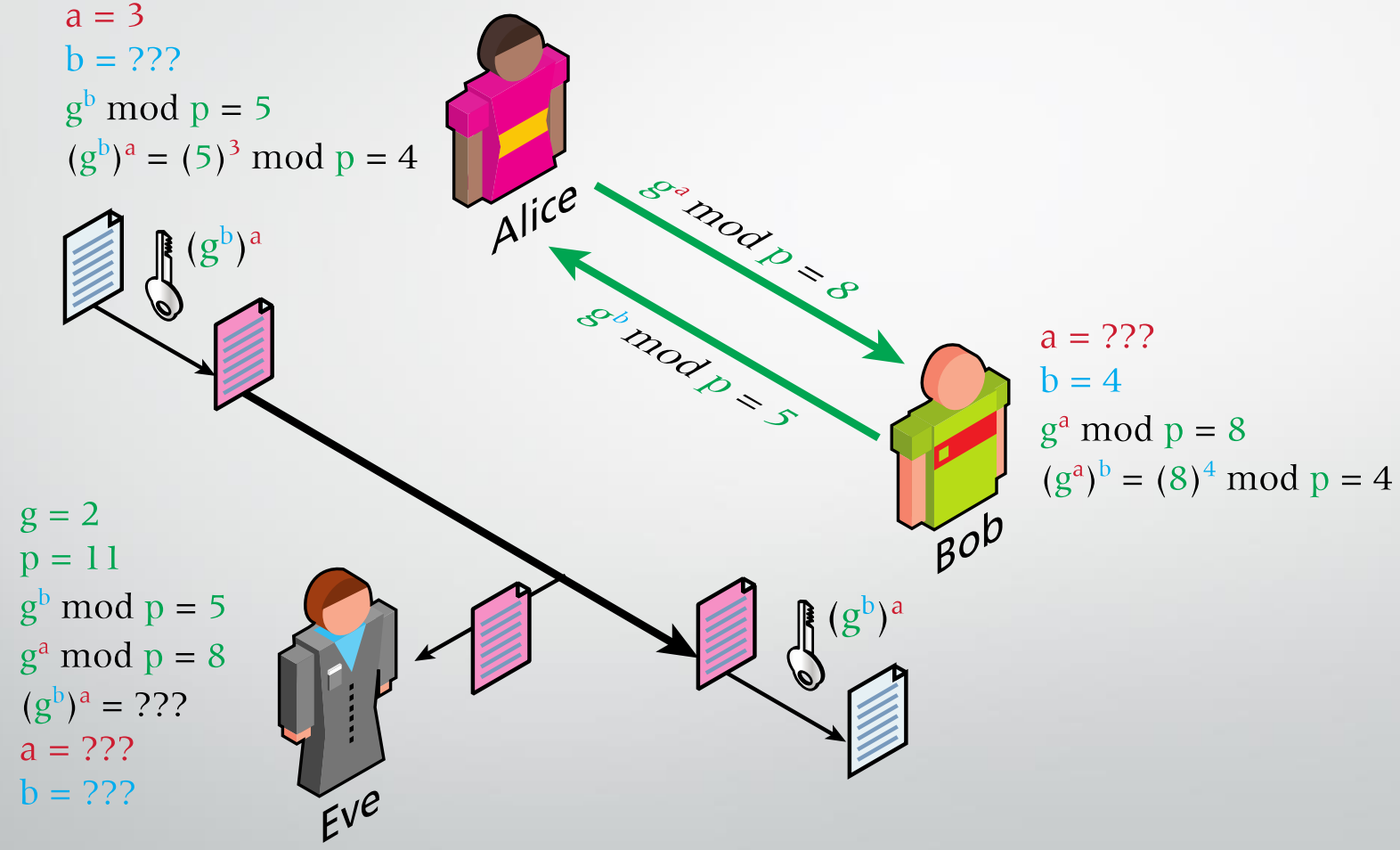
Cryptography

Public Key Cryptography

- Public key cryptography (or asymmetric cryptography) solves the problem of the secret key by using two distinct keys: a public one, widely distributed and another one, kept private.

Cryptography

Public Key Cryptography – Diffie Hellman



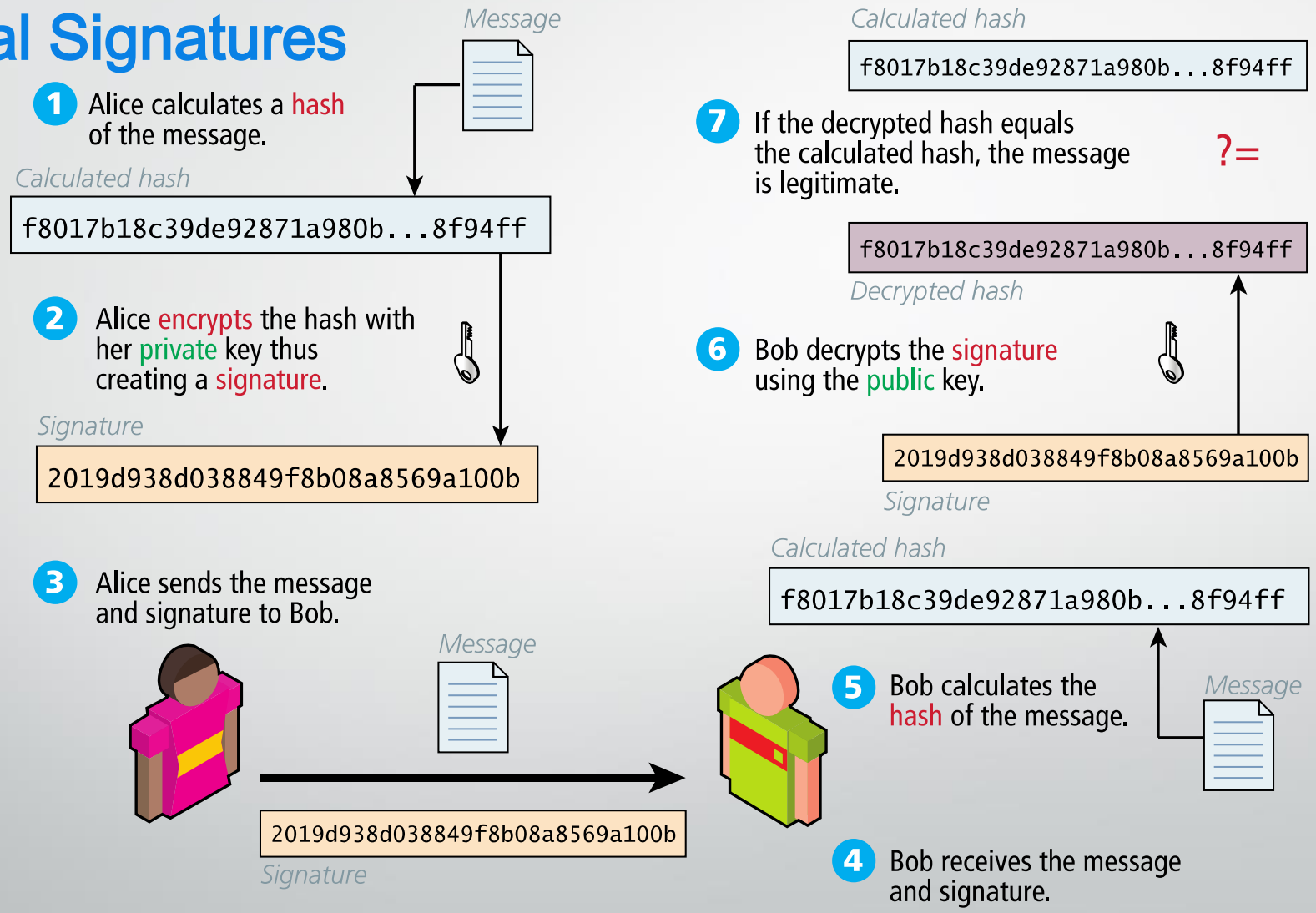
Cryptography

Digital Signatures

- A **digital signature** is a mathematically secure way of validating that a particular digital document was
  - created by the person claiming to create it (authenticity),
  - was not modified in transit (integrity), and
  - cannot be denied (nonrepudiation).

Cryptography

Digital Signatures



Web Systems and Technologies

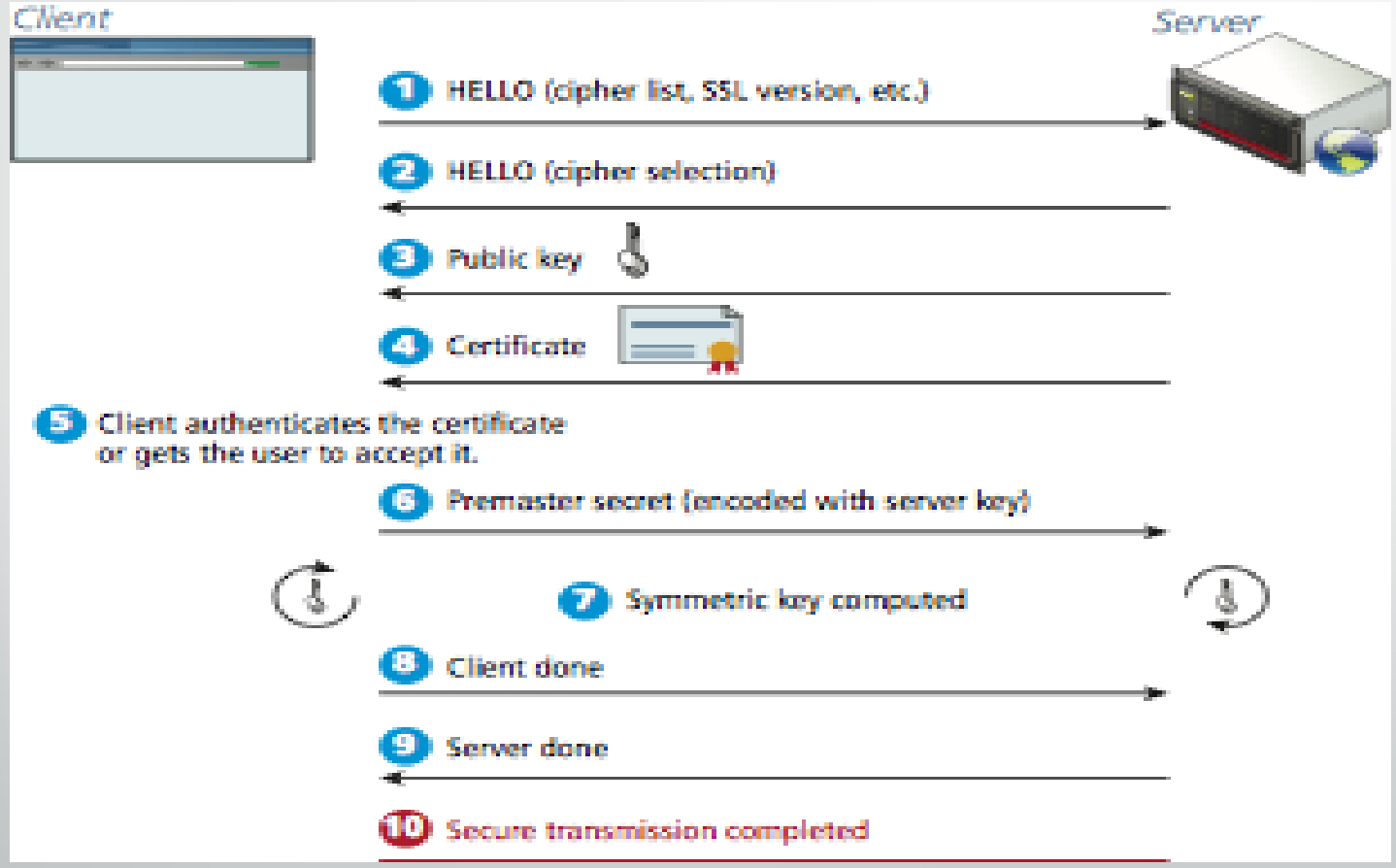
HTTPS

Hypertext Transfer Protocol Secure (HTTPS)

What the Browsers Say

Hypertext Transfer Protocol Secure (HTTPS)

Secure Handshakes



Hypertext Transfer Protocol Secure (HTTPS)

Certificates and Authorities

Plain text content

Common Name: funwebdev.com
Organization: funwebdev.com
Locality: Calgary
State: Alberta
Country: CA
Valid From: July 23, 2013
Valid To: July 23, 2014
Issuer: funwebdev.com, funwebdev.com
Key Size: 1024 bit
Serial Number: 9f6da4acd62500a0

Actual transmitted certificate

-----BEGIN CERTIFICATE-----
MIICfTCCAeYCCQCfbaSs1iUAoDANBgkqhkiG9w0BAQUFADCBgjEL
MAkGA1UEBhMCQ0ExEDAOBgNVBAgTB0FsYmVydGExEDAOBgN
VBAcTB0NhbGdhcnkxFjAUBgNVBAoTDWZ1bndlYmRldi5jb20xFjAU
BgNVBAMTDWZ1bndlYmRldi5jb20xHzAdBgkqhkiG9w0BCQEWEH
Job2FyQG10cm95YWwuY2EwHhcNMTMwNzIzMjI0NjU2WhcNMT
QwNzIzMjI0NjU2WjCBgjELMAkGA1UEBhMCQ0ExEDAOBgNVBAg
TB0FsYmVydGExEDAOBgNVBAcTB0NhbGdhcnkxFjAUBgNVBAoTD
WZ1bndlYmRldi5jb20xFjAUBgNVBAMTDWZ1bndlYmRldi5jb20xHz
AdBgkqhkiG9w0BCQEWEHJob2FyQG10cm95YWwuY2EwgZ8w
DQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMSS8uQ6ZXVW6yV
6MUcdZxdQTPfUlpXXW6DYmQMVmOEE7mjrhmj3jLDQn+FU8Qsv
IS8+GrDoyZ/5hhGBLYQLIhIcRQBULS9yNRIB7+mWOT45QycqJH/9xC
VcTwI4D//qVvAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAAzOsxgr
ltLw/DZXmqcV/W8C859m43D3gbc66jaaNYu5cA+Fn2FpS7z8oYeV
m0wWXcrmIj4bIWvpp3IbhPT12+XcVfJMda4nLSb/SPyjv4yvz9jeL
Ya/c0Z1IA7v6bk1ixwZSB9E=
-----END CERTIFICATE-----

Hypertext Transfer Protocol Secure (HTTPS)

Certificates and Authorities

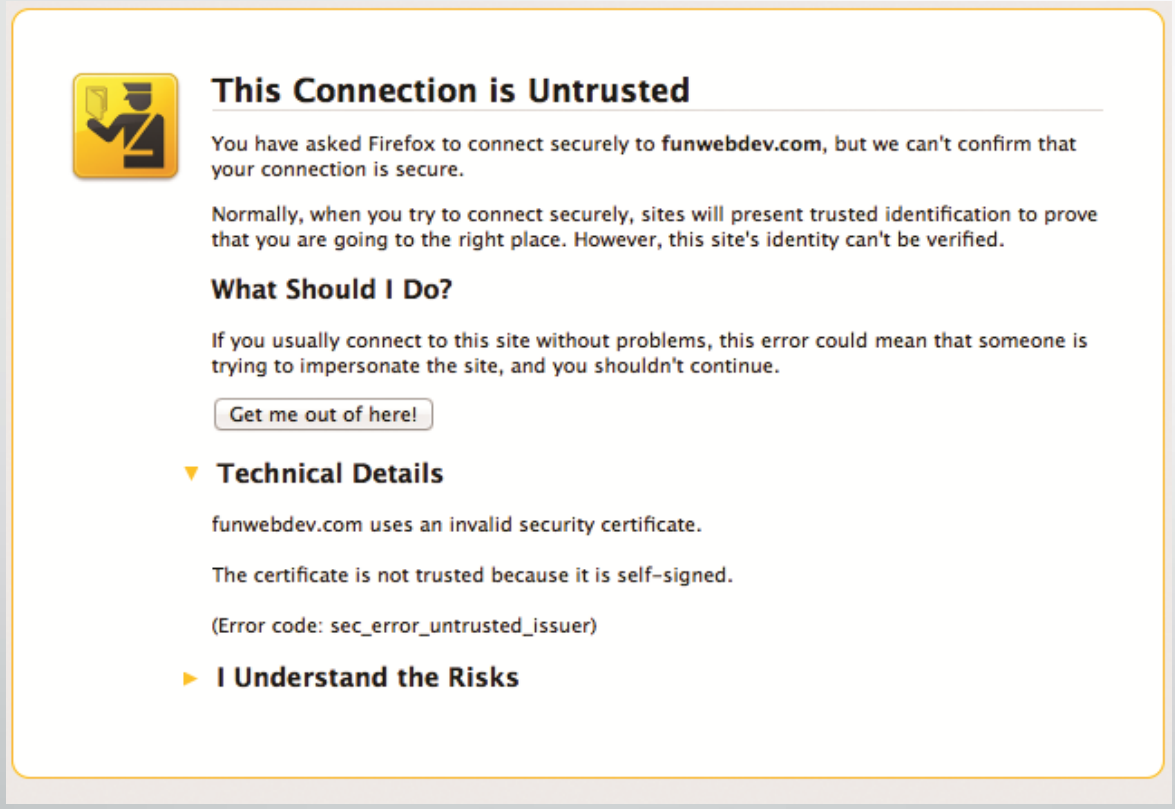
- **Certificate Authority (CA)** allows users to place their trust in the certificate since a trusted, independent third party signs it.
- The CA's primary role is to validate that the requestor of the certificate is who they claim to be, and issue and sign the certificate containing the public keys so that anyone seeing them can trust they are genuine.
- In browsers, there are many dozens of CAs trusted by default
- You can also self-sign certificates (generates warnings)

Hypertext Transfer Protocol Secure (HTTPS)

Certificates and Authorities

Hypertext Transfer Protocol Secure (HTTPS)

Certificates and Authorities



Hypertext Transfer Protocol Secure (HTTPS)

Migrating to HTTPS from HTTP

- Coordinating the migration of a website can be a complex endeavor
- Mixed Content
  - Internal links within the site.
  - External links to frameworks delivered through a CDN.
  - Any links or references generated by PHP code that might include a hardcoded http.
  - References to http within any HTML markup outside of PHP blocks.
- Redirects from old site

Web Systems and Technologies

XML Overview

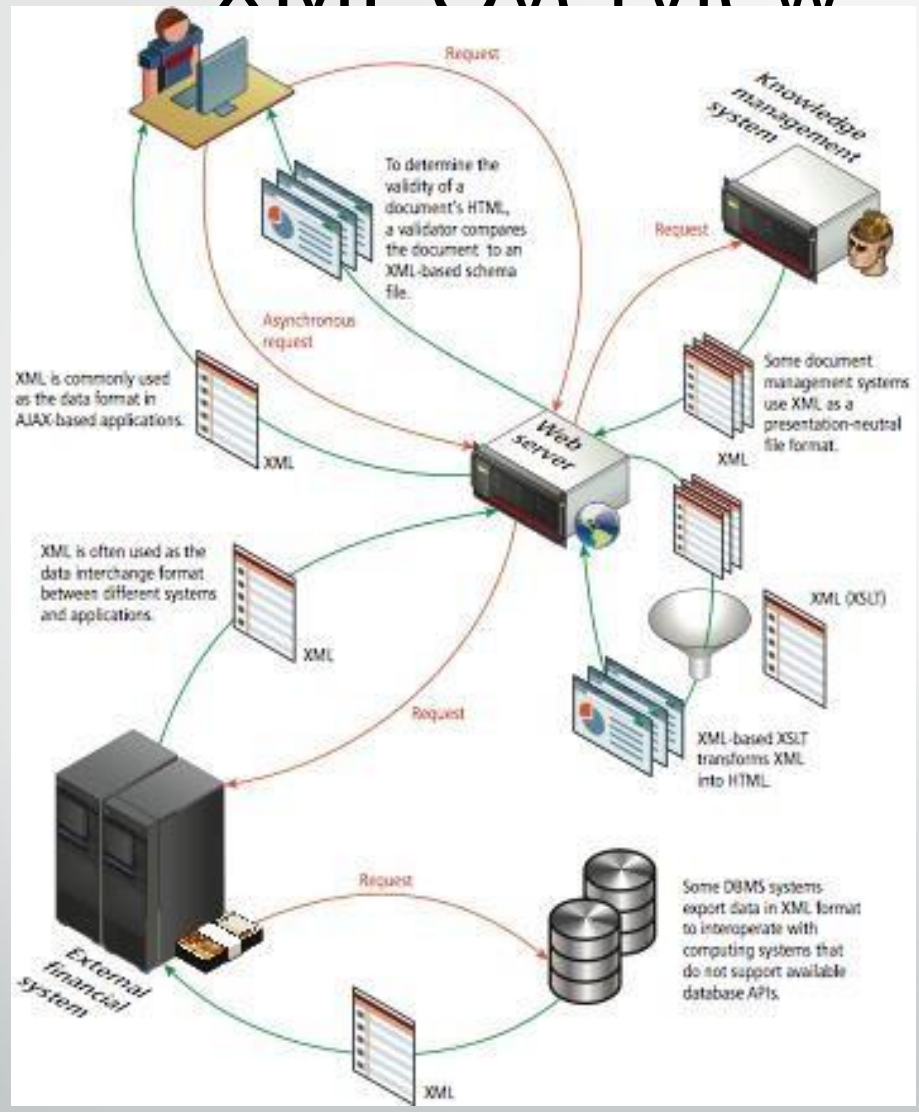
XML Overview

Introduction

- XML 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

XML Overview

- XML Usage



XML Overview

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

XML Overview

Well-Formed XML Simplified 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>
```

XML Overview

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

XML Overview

Example Document Type Definition (DTD)

```
<?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 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)>
]>
```

XML Overview

- Example 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>
```

```
<?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 Overview

- Example XML Schema

```
            <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:schema>
```

```
<?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>
```

Web Systems and Technologies

XPath and XSLT

XPath and XSLT

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 and XSLT

XPath – Expressions

Expression	Description	
nodename	All nodes with the name "nodename"	painting
/	From root	/art/painting
//	Anywhere in the document	
.	Current node	
..	Parent node	
@	Attribute	
[]	Predicate	

XPath and XSLT

```
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>
```

XPath – Examples

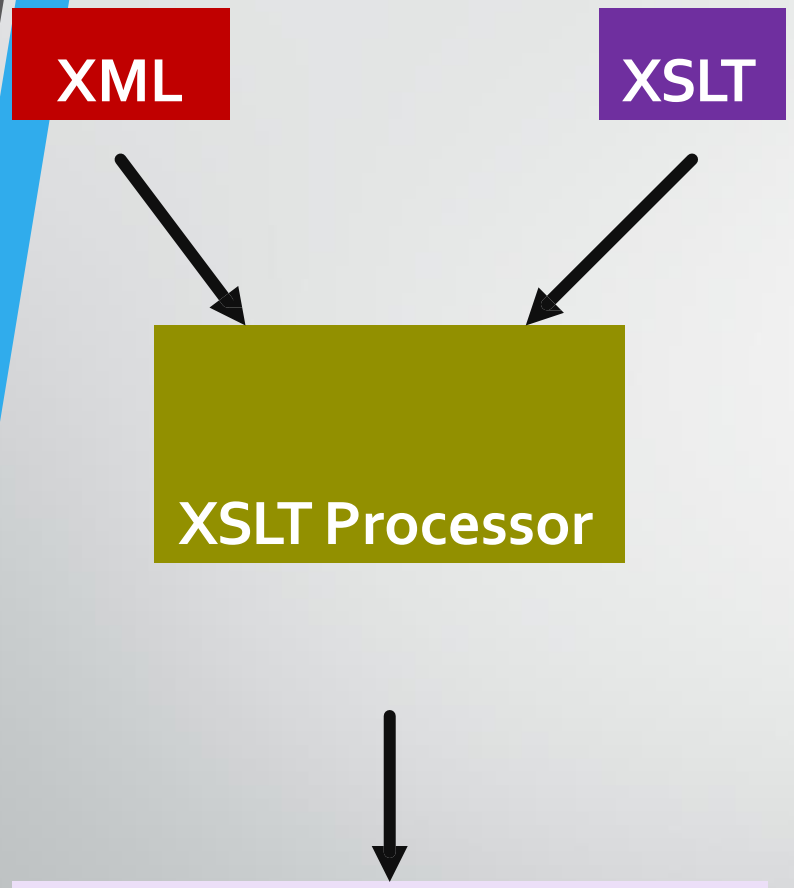
- /art
  - Selects element "art"
- //year
  - Selects element "year"
- //name/..
  - Selects element "artist"
- //painting/@id
  - Selects attribute "id" in painting
- //painting[year>1800]
  - Selects "painting" elements with year > 1800

XPath and XSLT

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

XPath and XSLT



XSLT – Example

```
<?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>
```

Web Systems and Technologies

XML Processing

XML Processing

XML Processing in JavaScript

- The in-memory approach, which involves reading the entire XML file into memory
- The event or pull approach, which lets you pull in just a few elements or lines at a time

XML Processing

XML Processing in JavaScript

```
var xmlhttp = new XMLHttpRequest()
xmlhttp.open("GET","art.xml",false);
xmlhttp.send();
var xmlDoc = xmlhttp.responseXML;
var paintings = xmlDoc.getElementsByTagName("painting");
```

XML Processing

XML Processing in jQuery

```
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

XML Processing in PHP

- Can process XML using
  - SimpleXML
  - XMLReader

XML Processing

SimpleXML

```
<?php
$filename = 'art.xml';
if (file_exists($filename)) {
  $art = simplexml_load_file($filename);
  // access a single element
  $painting = $art->painting[0];
  echo '<h2>'.$painting->title.'</h2>';
}
```

XML Processing

XPath with SimpleXML

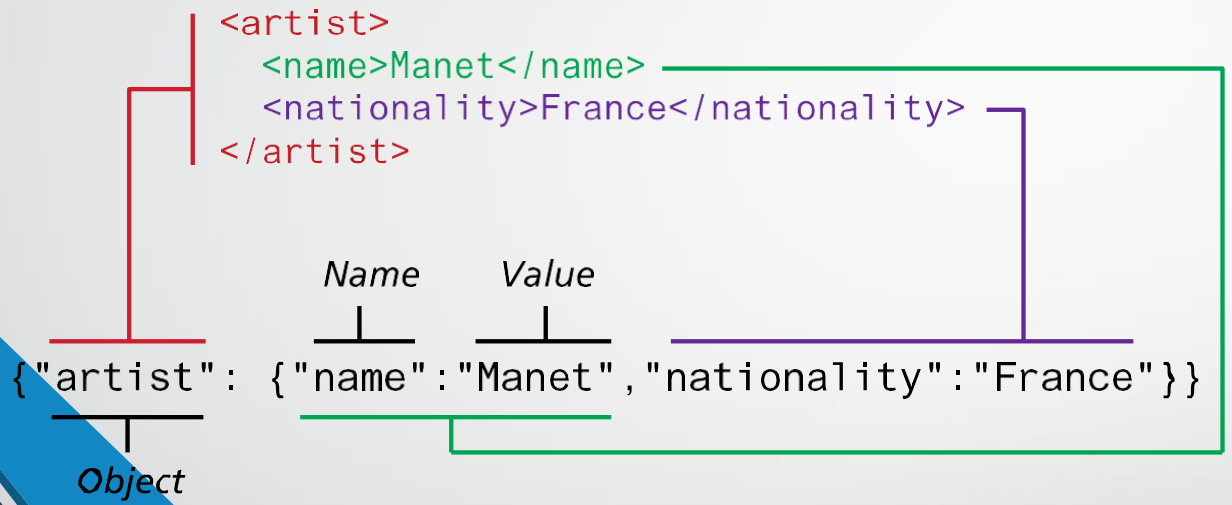
```
$art = simplexml_load_file($filename);
$titles = $art->xpath('/art/painting/title');
foreach ($titles as $t) {
  echo $t . '<br/>';
}
$names = $art->xpath('/art/painting[year>1800]/artist/name');
foreach ($names as $n) {
  echo $n . '<br/>';
}
```

Web Systems and Technologies

JSON

JSON

Sample JSON



JSON

Using JSON in JavaScript

```
var text =  '{"artist": {
             "name":"Manet",
             "nationality":"France"
                    }
                  }';
var a = JSON.parse(text);  alert(a.artist.nationality);
//or a['artist']['nationality']
```

JSON

Using JSON in PHP

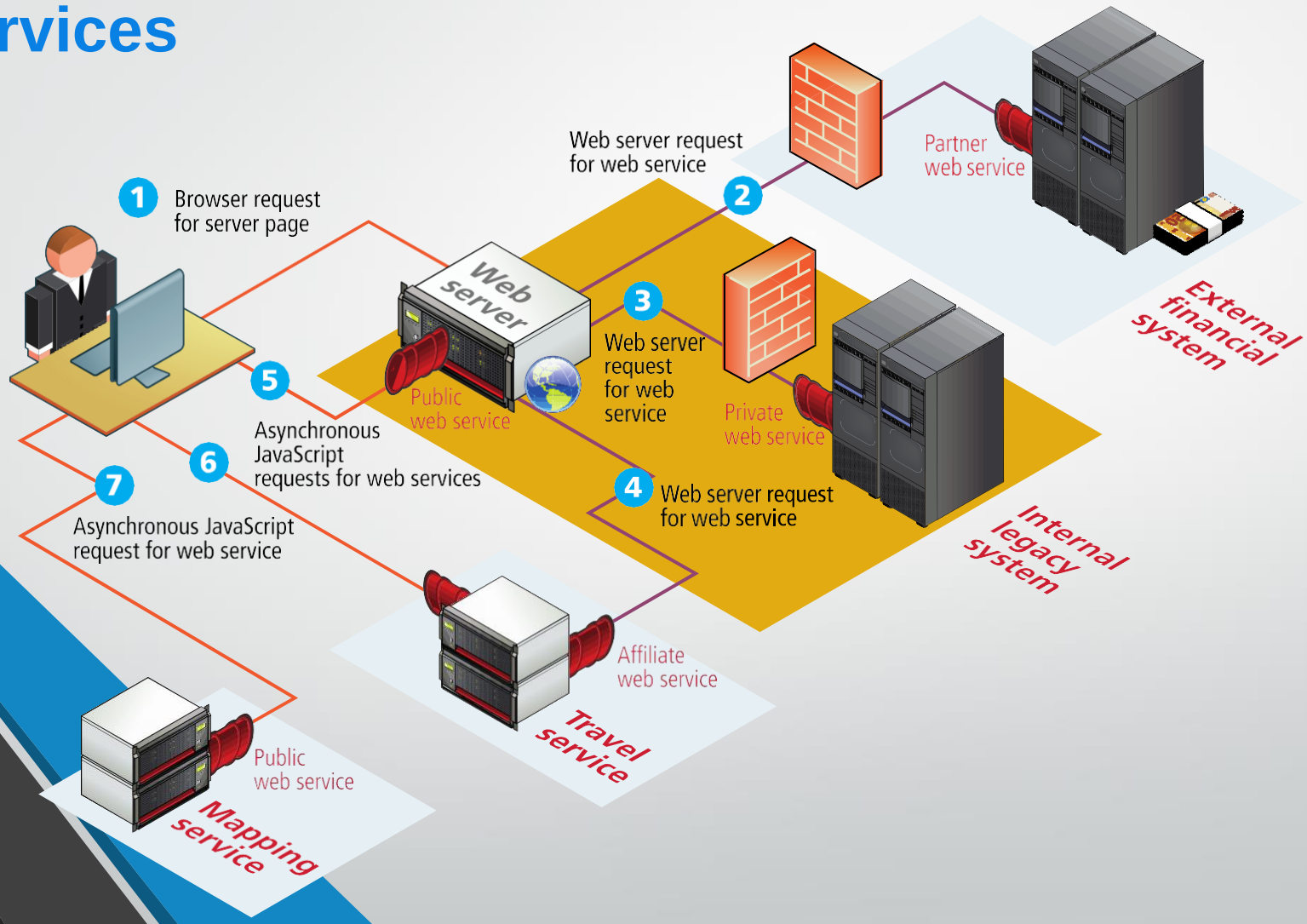
```
<?php
// convert JSON string into PHP object
$text =  '{"artist": {
              "name":"Manet",  "nationality":"France"
                  }
                }';
$anObject = json_decode($text);
// check for parse errors
if (json_last_error() == JSON_ERROR_NONE)
{
  echo $anObject->artist->nationality;
}

?>
```

Web Systems and Technologies

Overview of Web Services

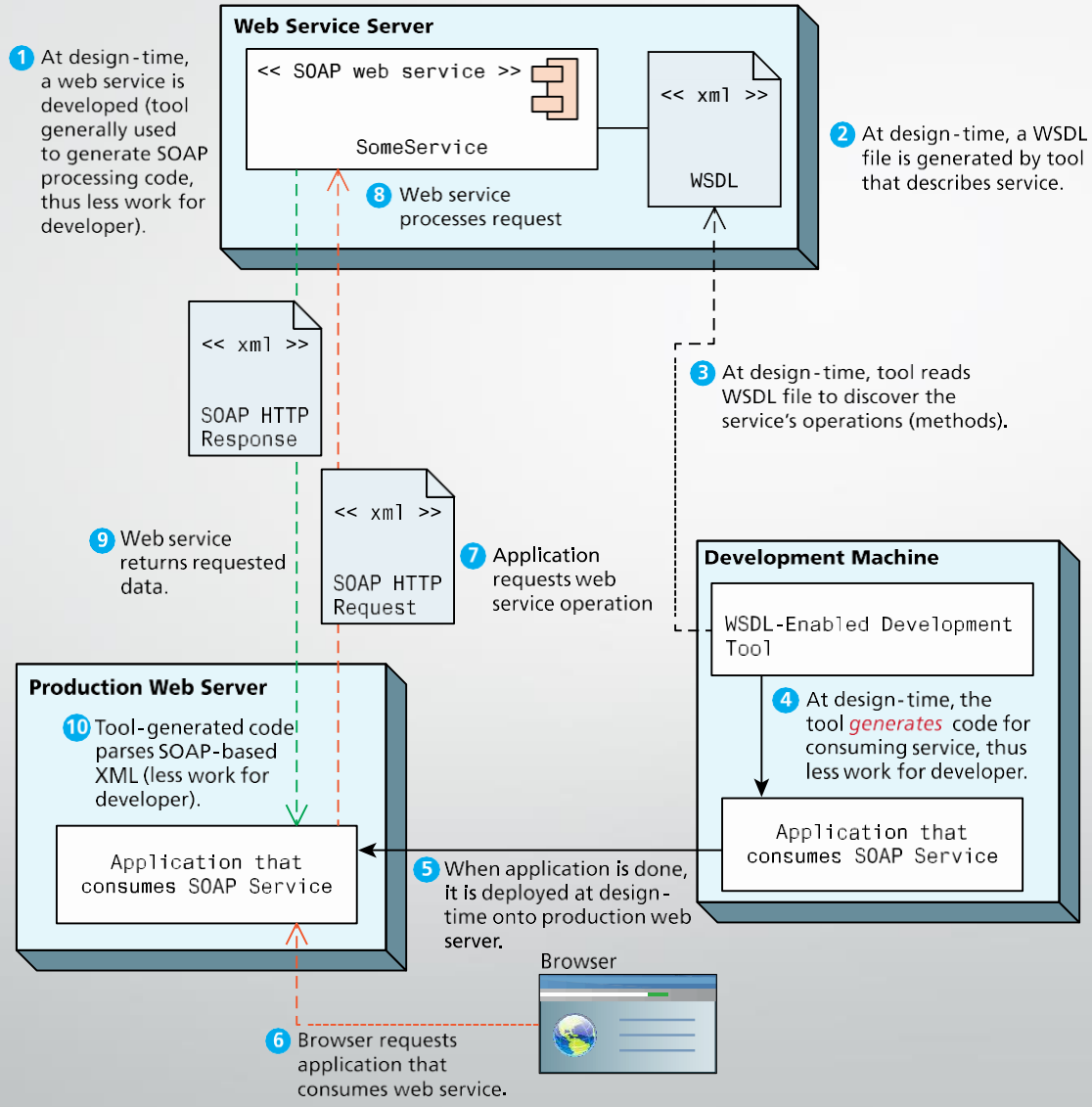
Overview of Web Services

Web Services



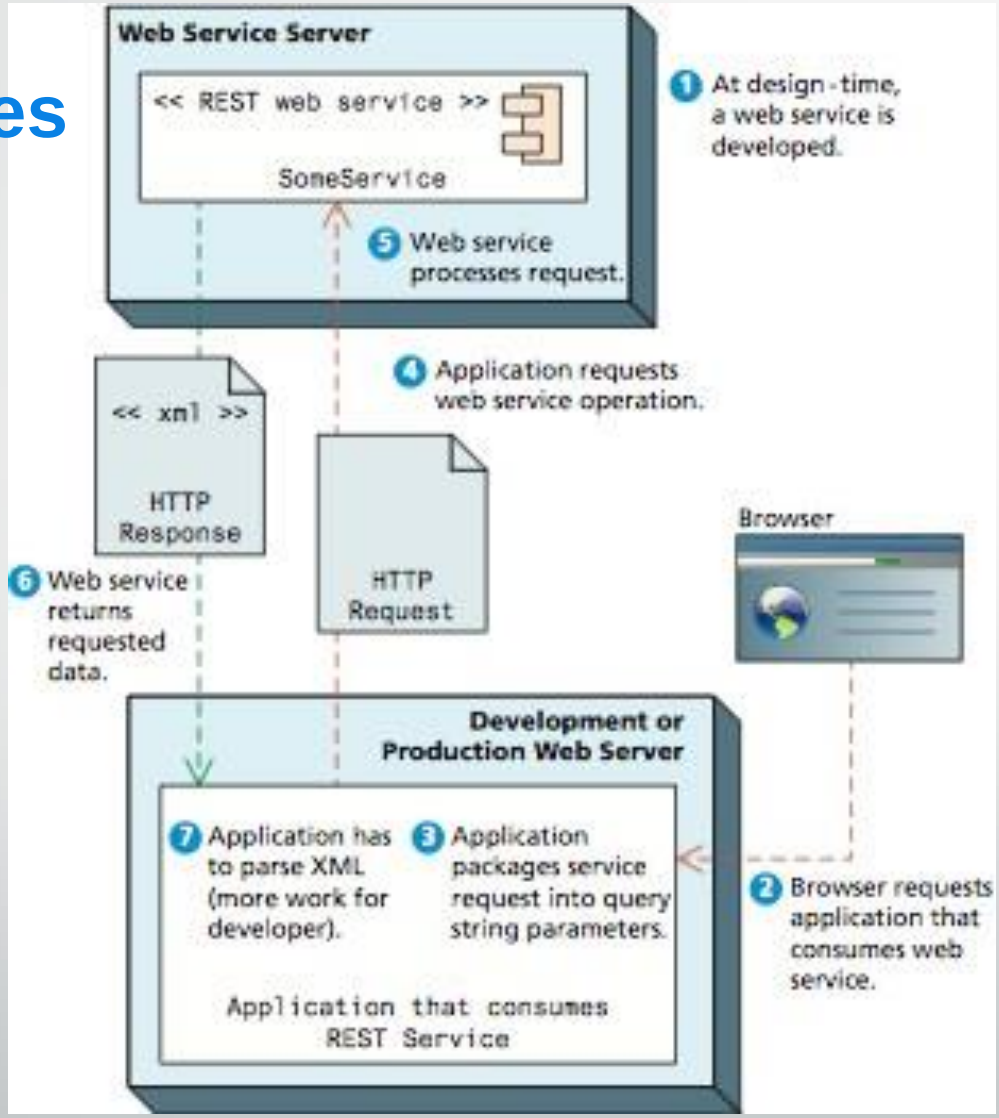
Overview of Web Services

SOAP Services



Overview of Web Services

REST Services



/authors/1

Get Post
Put Delete

Overview of Web Services

Identifying and Authenticating Service Requests

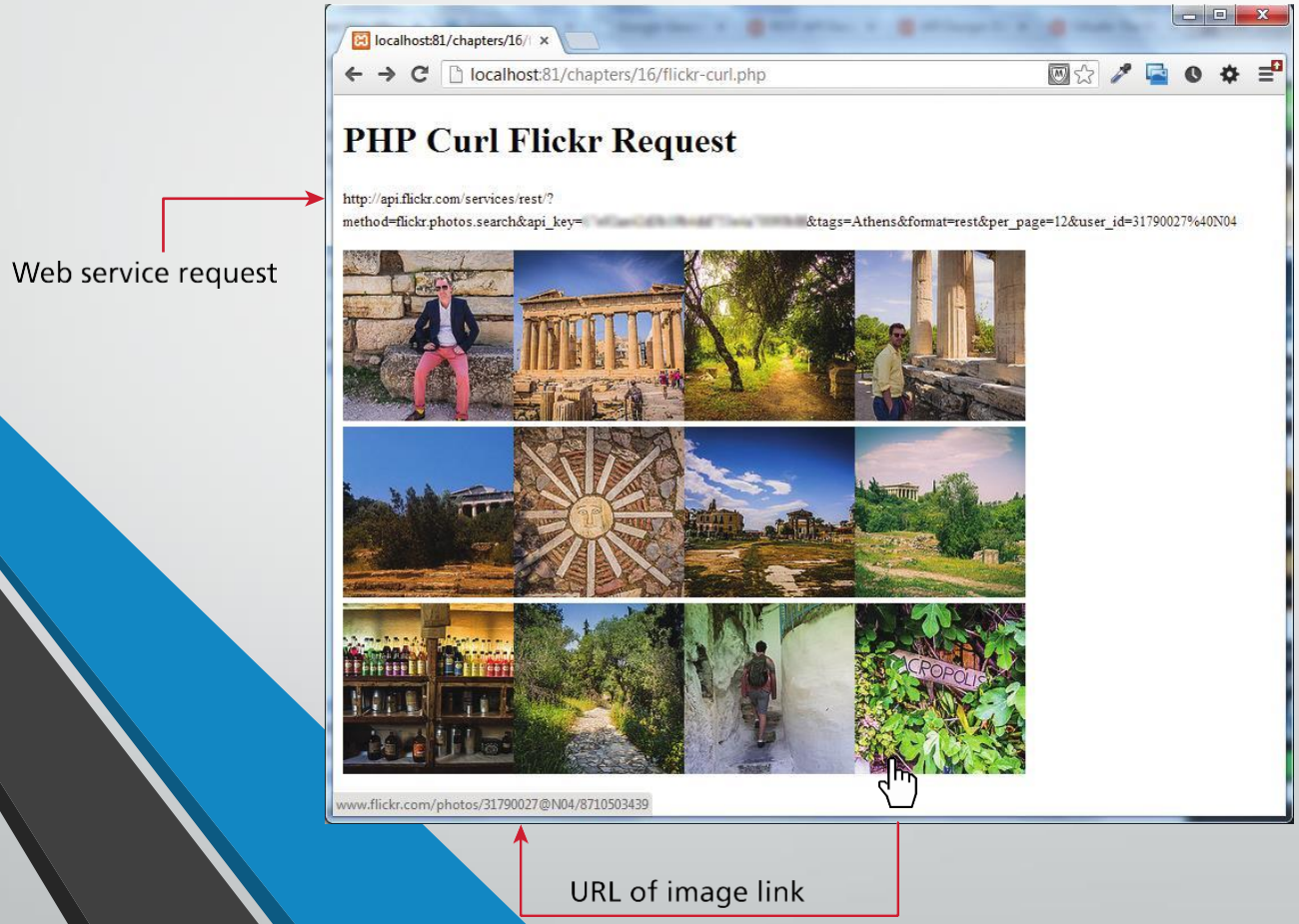
- Identity. Each web service request must identify who is making the request.
- Authentication. Each web service request must provide additional evidence that they are who they say they are.
- API Keys
- https://dev.virtualearth.net/REST/v1/Locations?o=json&query=British%20Museum,+Great+Russell+Street,+London,+WC1B+3DG,+UK&key=**[BING API KEY HERE]**

Web Systems and Technologies

Consuming Web Services in PHP

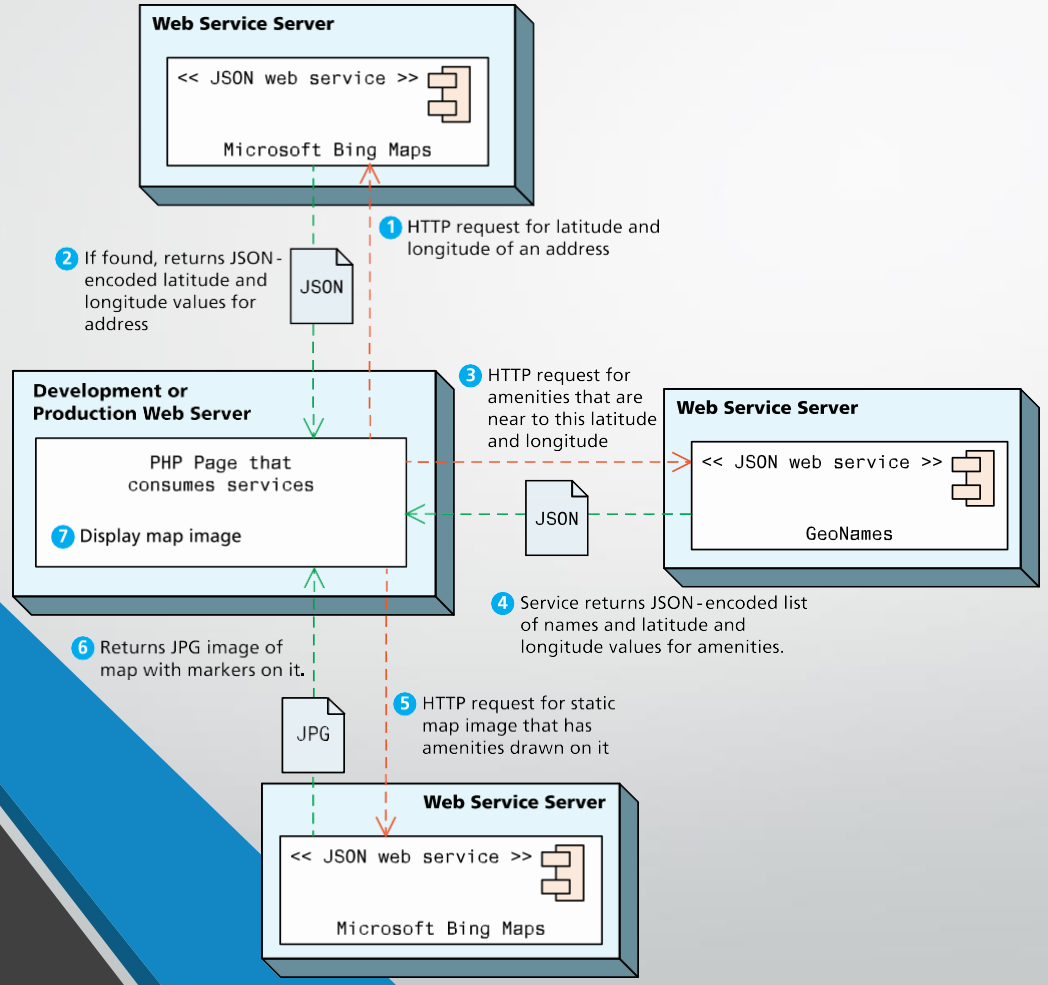
Consuming Web Services in PHP

Consuming an XML Web Service



Consuming Web Services in PHP

Consuming a JSON Web Service



Consuming Web Services in PHP

Consuming a JSON Web Service

URL of service request for static road map image

Zoom level (between 1 and 21)

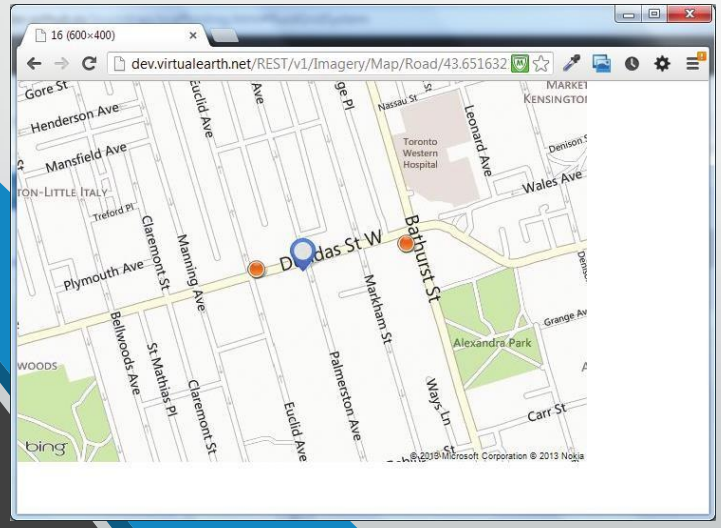
```
http://dev.virtualearth.net/REST/v1/Imagery/Map/Road/43.65163,-79.40853/16?
key=[your api key]
&mapSize=600,400
&pp=43.65163,-79.40853;66;
&pp=43.65208,-79.40618;34;
&pp=43.65166,-79.40958;34;
```

Width and height of map in pixels

Location of marker (marker 66 = blue circle)

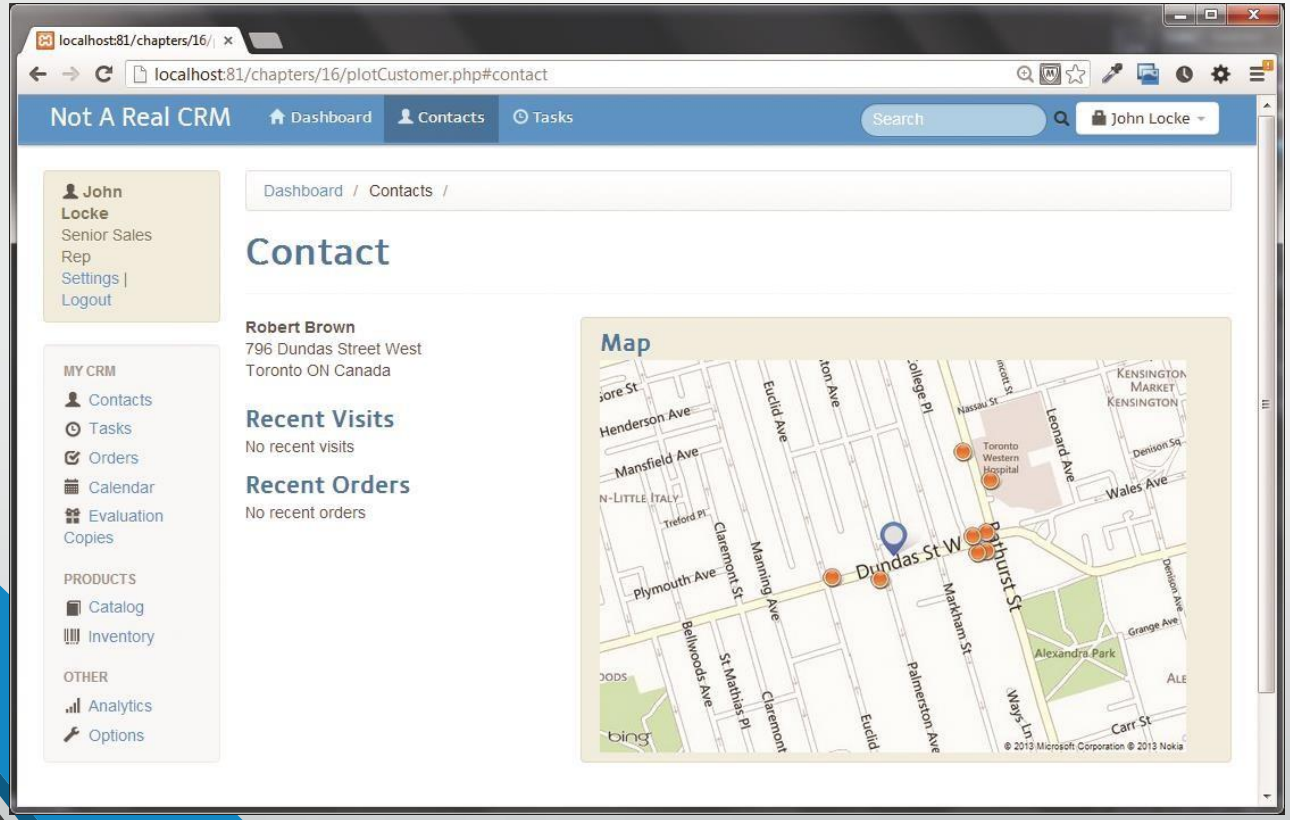
Location of other markers (amenities) with marker 34 = orange circle

Location (latitude and longitude) of center of map



Consuming Web Services in PHP

Consuming a JSON Web Service



Web Systems and Technologies

Creating Web Services

Creating Web Services

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*

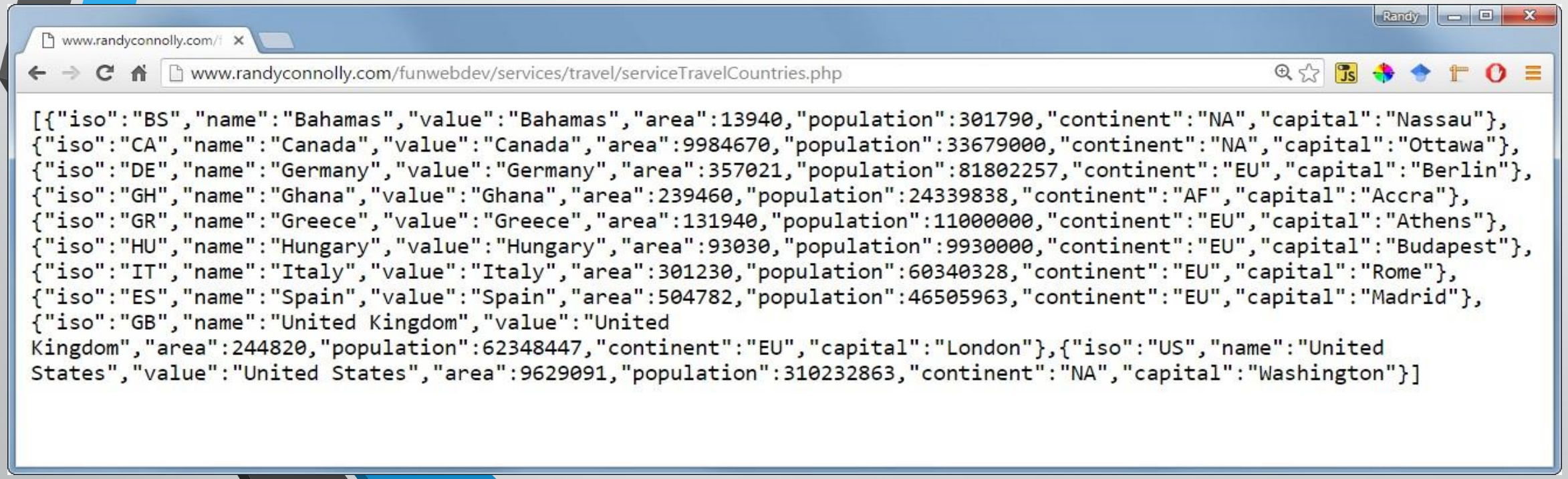
Creating Web Services

Creating a JSON Web Service

```
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

]; } }
```

Creating Web Services

Creating a JSON Web Service



WEEK 15

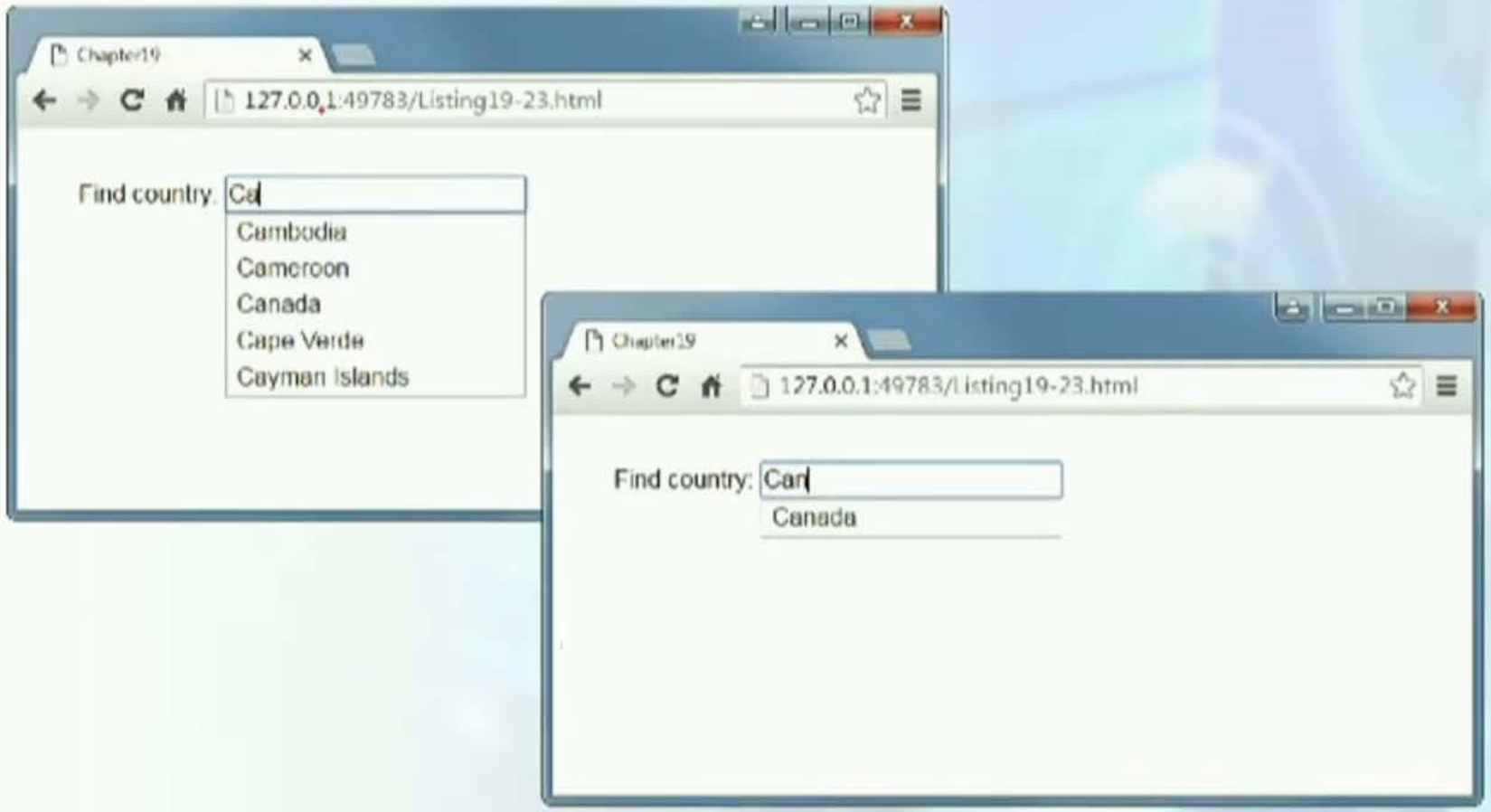
TOPIC 127 TO 135

Web Systems and Technologies

- 127-Web Services Using AJAX
- 128-JavaScript Frameworks
- 129-Node.JS
- 130-Simple Node.JS Program
- 131-Express.js, Routing in Express
- 132-Node.JS MYSQL
- 133-Introduction to Angular
- 134-Introduction to Web Sockets
- 135-Web Server Hosting

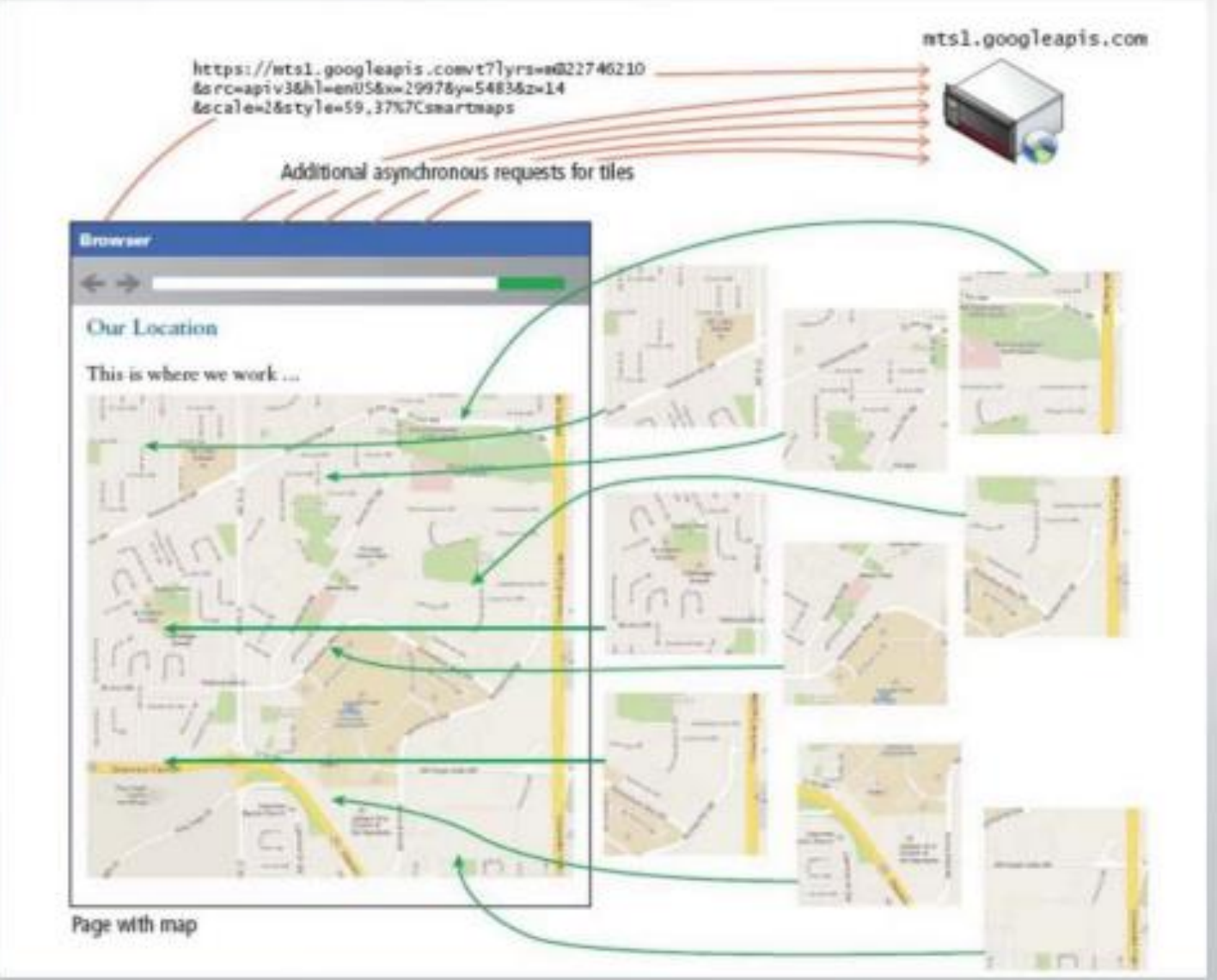
Interacting with Web Services using AJAX

Consuming Your Own Service



Using Google maps

Using Google Maps



Using Google Maps

```
<style>
#map {
  height: 500px;
  width: 600px
}
</style>

<script
src="https://maps.googleapis.com/maps/a
pi/js?key=YOUR_API_KEY" async></script>
```

Using Google Maps

```
<script>
  $(function() {
  // hard-coded latitude and longitude for demonstration purposes
  var ourLatLong = {lat: 51.011179 , lng: -114.132866 };
  var ourMap = new google.maps.Map(document.getElementById('map'), {
    center: ourLatLong,
    scrollwheel: false,
    zoom: 14
    });});
</script>
</head>
<body>
  <h2>Our Location</h2>
  <h3>This is where we work ... </h3>
  <div id="map"></div>
</body>
</html>
```

JavaScript Frameworks

JavaScript Frame work

JavaScript Frame work

JavaScript framework provide structure for developing JavaScript applications

Popular frameworks

- Angular
- Ember
- Vue
- React
- Express

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

Node.js

Introduction

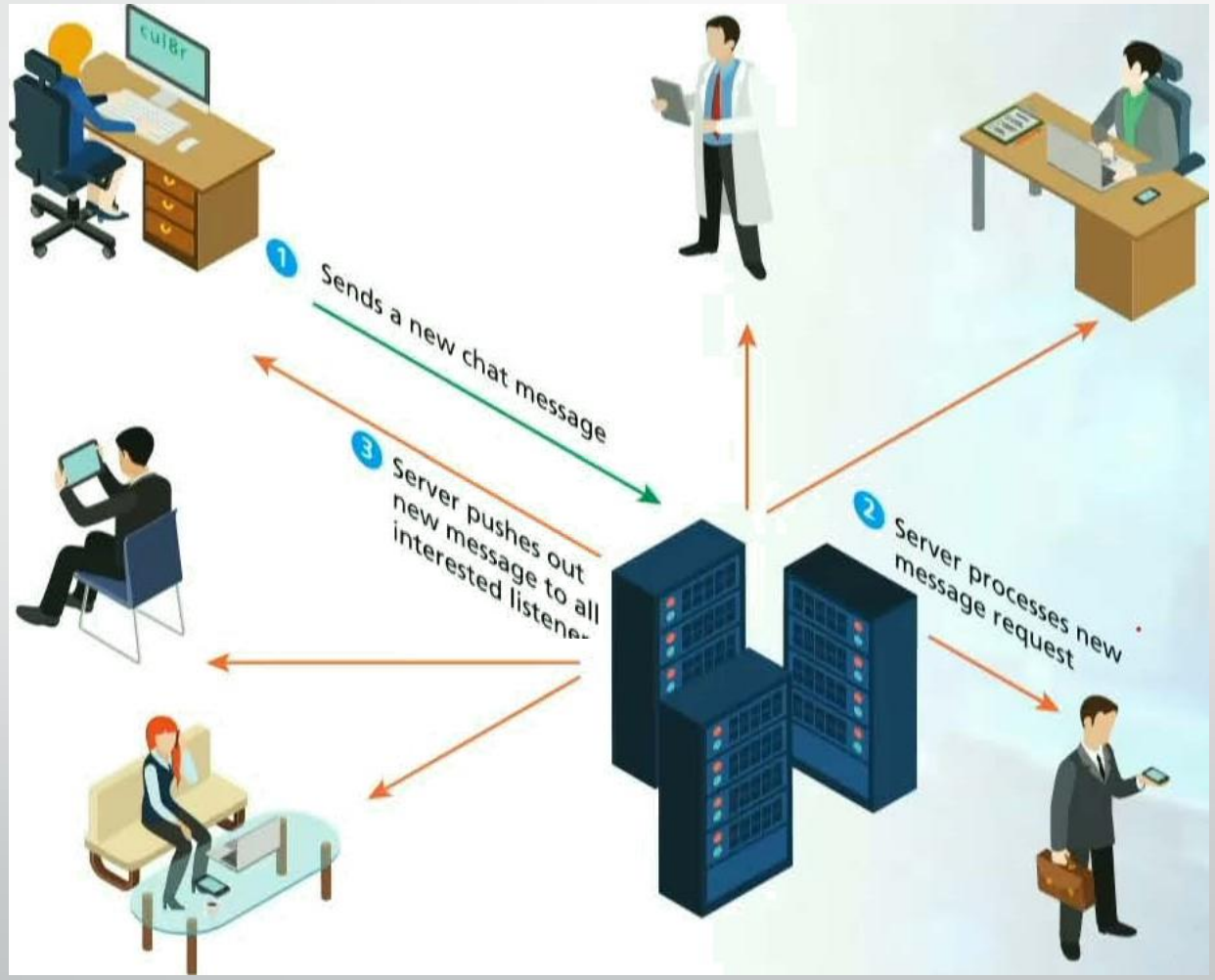
- Started in 2009
- Runs Javascript outside the browser
- Uses Chrome V8 Javascript engine
- Javascript used to generate HTML code
- Uses non-blocking I/O
- Often used for server-side web Development
    - Also used for non-web development,

Introduction

- 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

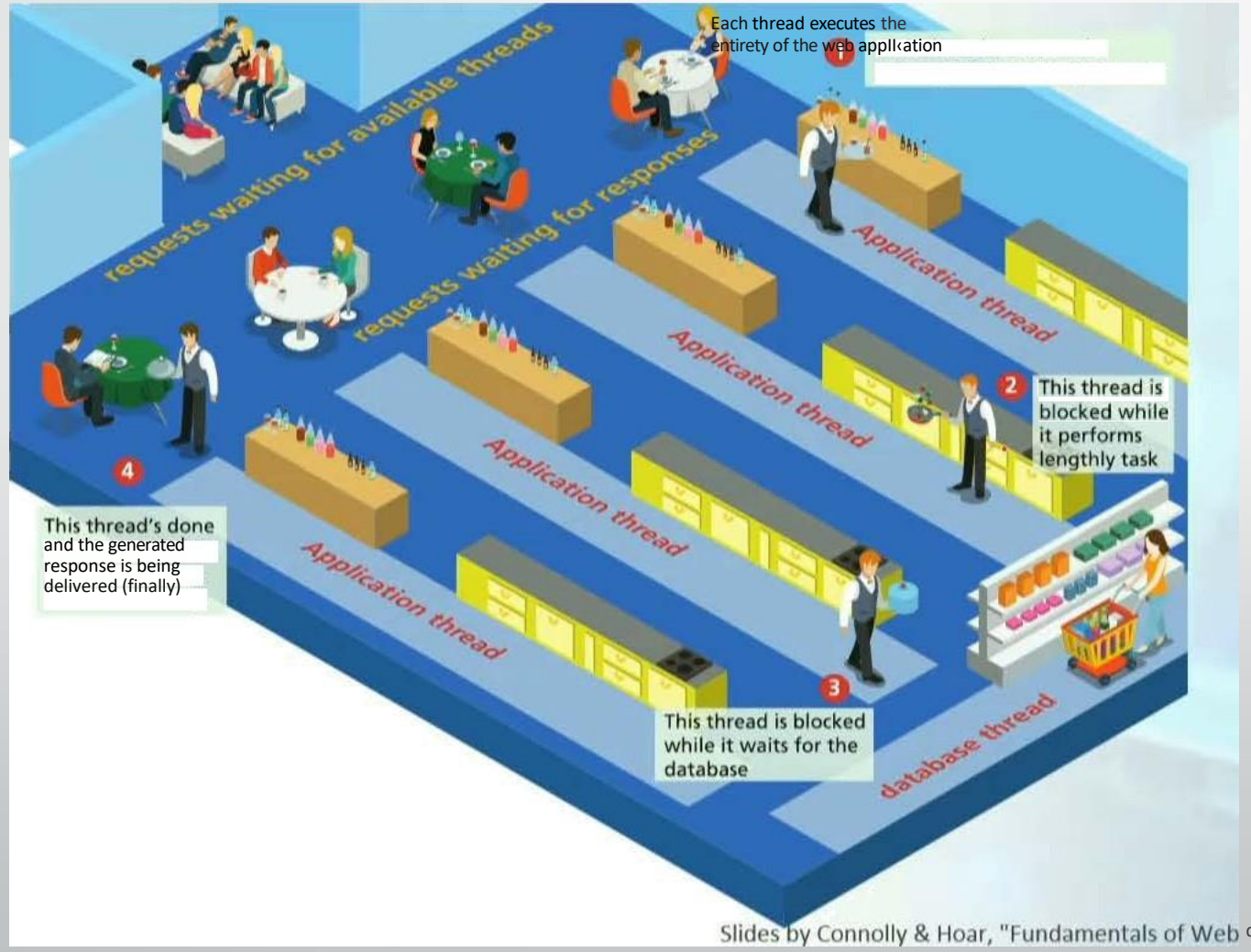
Node.js

Push based web application

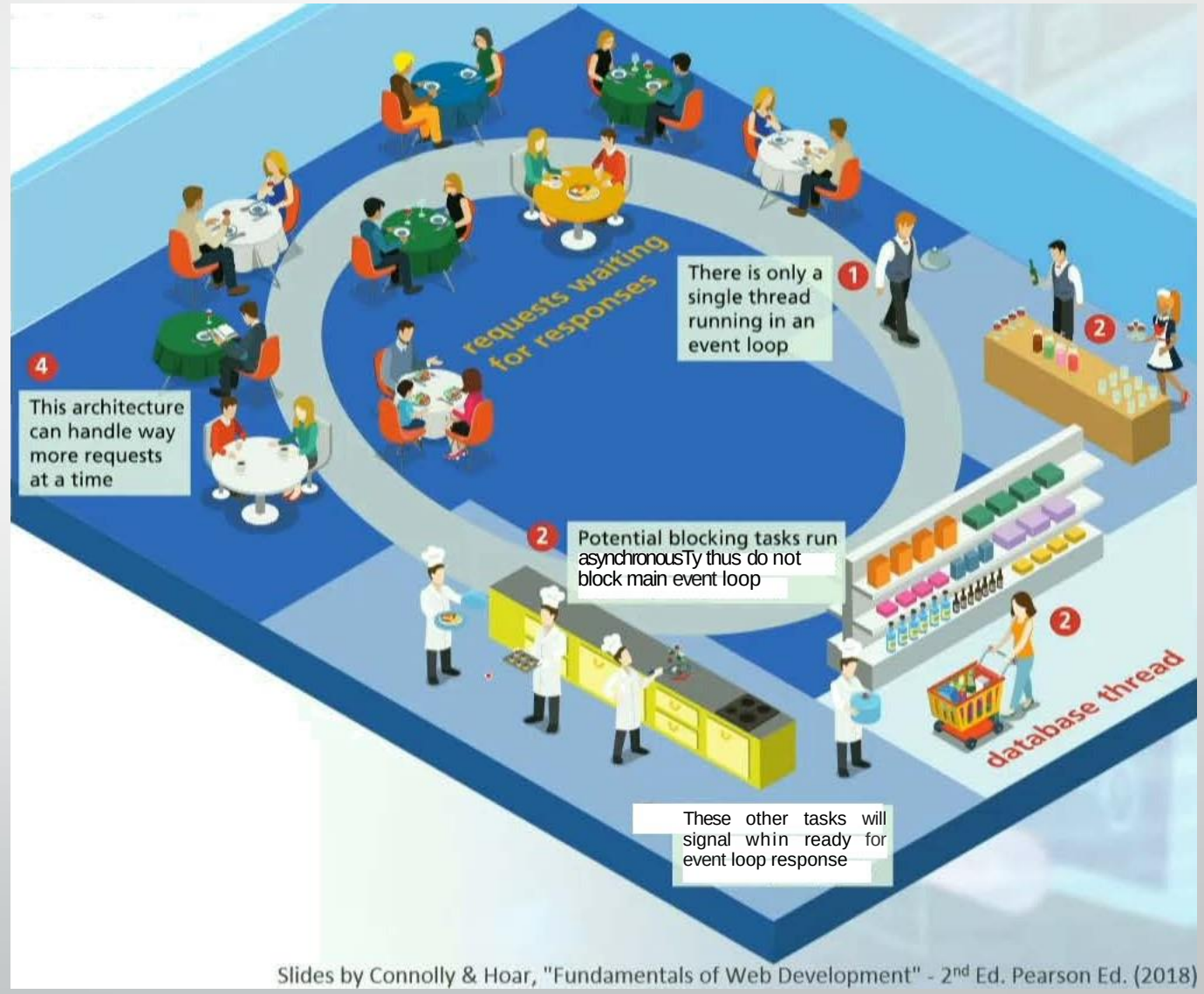


Slit1es by CDnnDIly & I3oar, "Fundamental.s of Web Development" - 2'* Ed. Pearson Ed. (2018)

Blocking Thread-based Architecture (Traditional)



Node.js Single-Thread Architecture



Node Package Manager (NPM)

- Package manager for JavaScript
- Used for installing JavaScript libraries and frameworks
- Can handle dependencies
- Usage

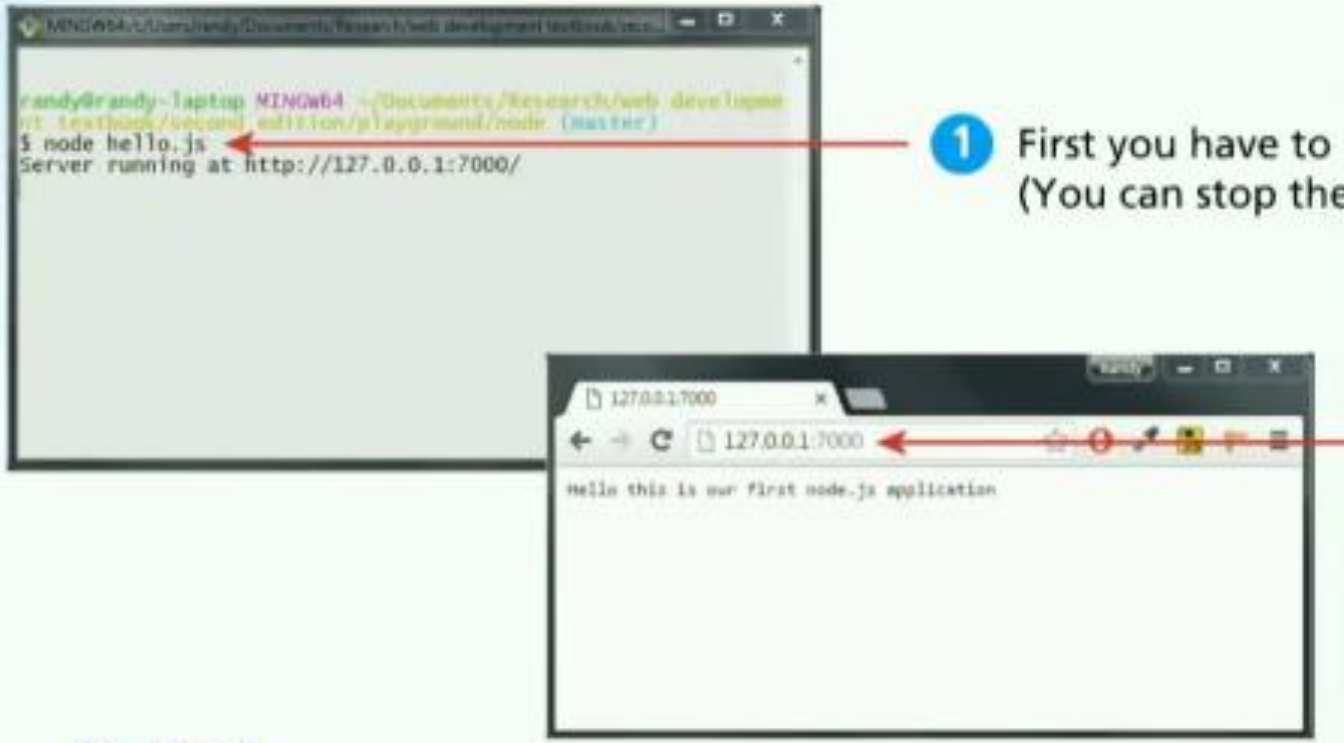
```
npm install
npm install packagename
npm install packagename@version
npm update
npm list
```

Node.js – Example

Working with Node.js

```
// Load the http module to create an HTTP server
var http = require('http');

// Configure HTTP server to respond with Hello World to all requests 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/”);
```

Working with Node.js



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.

Example

- Writing to a file

  data is written in the file 'message.txt'

  When writing is complete, anonymous function is "called back"

```
const fs = require('fs')

fs.writeFile('message.txt',
  'Hello World!', function (err) {
  if (err) {
    console.error(err);
    process.exit(1);
  } else {
    console.log('Writing is done.');
  }
})
```

Express.js

Introduction

Minimalist web application framework for Node.js

- Handles
  - Routes
  - Requests/Responses
  - Views

Example

```
const express = require('express')
const app = express()
const port = 7000

app.get('/:name', function(req, res) {
  res.send('Hello '.req.params.name)
})

app.listen(port, function (){
 console.log("http://localhost:".${port})
})
```

MySQL with Node.js

Introduction

- Mostly Node.js is used with NoSQL databases like MongoDB
- Can be used with traditional DBMS like mysql

Example

```
Var require('mysql');
var sql ='SELECT * FROM books';



Var con = mysql.createConnection({
Host :"localhost",
user:"root",
password:"",

});


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);

       });

       });
```

Angular

Introduction

- Angular is a popular browser-based, open-source JavaScript MVC framework (Google driven)
- It is the A' in the MEAN stack, independent of the other components of the stack, and can be used without any of them
- Angular allows developers to write in TypeScript, JavaScript, or Dart
- AngularJS uses JavaScript
- Well suited for single page applications

Angular

Creating Simple AngularJS Application

Angular

Creating a Simple AngularJS Application

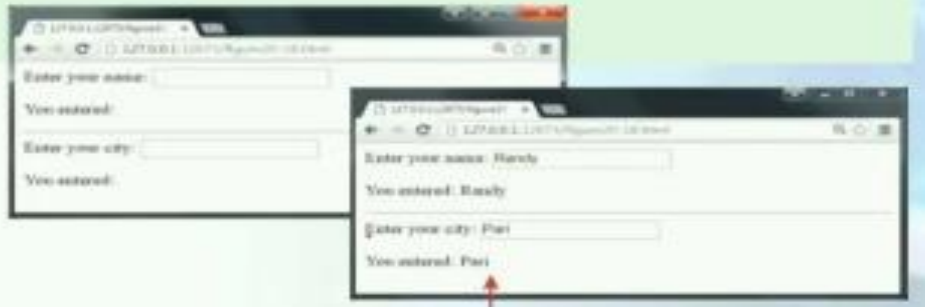
A directive for designating the root AngularJS element

```
<html ng-app>
<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" />
  <p>You entered: {{ name }}  </p>
  <hr>
  Enter your city: <input type="text" ng-model="city" />
  <p>You entered: {{ city }}  </p>
</body>
</html>
```

A directive for saving the field value in the Model

A data binding expression

A template



Appears as user types into textbox

A Controller

Angular

A Controller

Now this directive is specifying the *module* used in the application

```
<html ng-app="demo">
...   This element is going to use a controller to get its data
<body ng-controller="myController">
    <div id="search">                    Save the user's input in a model property named search
      City Search: <input type="text" ng-model="search" />
    </div>
    <table>  A directive to loop through a collection named cities (which is defined in the controller)
       <tr ng-repeat="city in cities | filter:search | orderBy: 'name'">
          <td>{{city.name }}</td>
           <td>{{city.country}}</td>
       </tr>
    </table>
</body>
</html>
```

Data bind to values in the collection

Uses *filters* to alter how this element works. In this example, the `filter` filter and the `orderBy` filter are used to modify how the `ng-repeat` works. Here the search refers to data item in the model.

A module is an AngularJS container for the different components used in the application

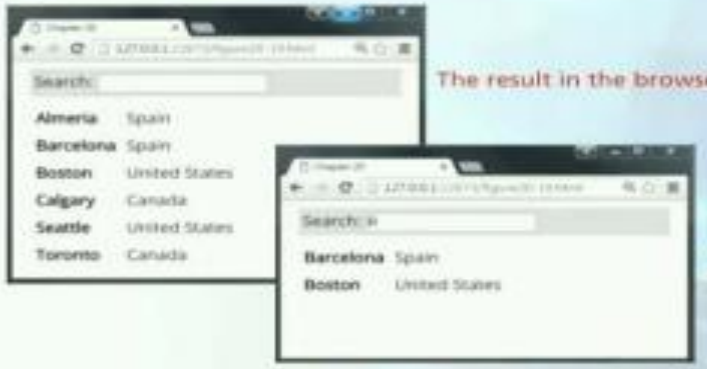
```
var myapp = angular.module('demo',[]);
```

Add a controller to the module named myController

The $scope variable is passed (injected into) the controller by AngularJS

```
myapp.controller('myController', function ($scope) {
    $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'}];
});
```

The $scope variable is used to store the model (data). Here we are defining an array of object literals named `cities`



The result in the browser (notice the sort order)

The `filter` filter alters the displayed cities based on the current value of the Search text field.

Getting Data from a Web Service

```
var myapp = angular.module('demo',[ ]),
  //notice that our callback function is passed the $http parameter.
  //parameters are injected by the AngularJS system "behind-the-scenes"
myapp.controller('myController',function (6scope, $http)

  //retrieve country data from web service
  Var url  =' http://randyoonnolly.com/funwebdev/services/travel/countries.php†continent=EU;'

$http.get(url).then(function (response)
$scope.countries= response.data
});
});
```

Web sockets

Introduction

- Web socket is a full duplex communication protocol
- Enables interaction between web browser and web server

Server Side Example (Node.js)

Server Side Example (Node.js)

```
const express = require('express');
const app = express();
const http = require('http');
const server = http.createServer(app);
const { Server } = require("socket.io");
const io = new Server(server);

io.on('connection', (socket) => {
  console.log('A user connected');});

io.on('message', function(message) {
  console.log('Message from client: '+message);
  io.emit('Message received '+message});

server.listen(7000, () => {
console.log('localhost:7000');});
```

Client Side Example (JavaScript)

Client Side Example (JavaScript)

```
var client = new WebSocket("http://localhost:7000");

//sending message on socket open
client.onopen = function() {
  client.send("Message sent to the server");
}

//receiving message
client.onmessage = function (event) {
  console.log(event.data);
}

//close connection
client.close();
```

Web server Hosting

Hosting Types

- Shared Hosting
  - Simple
  - Virtualized
- Dedicated
- Collocated
- In-house
- Cloud

WEEK 16

TOPIC 136 TO 144

Web Systems and Technologies

- 136 Domain and Name Server Administration
- 137Search Engines
- 138Page Rank
- 139Search Engine Optimization
- 140Understanding Bad SEO
- 141Introduction to Online Social Networks
- 142Monetization with Ads
- 143Web Analytics
- 144 Google Analytics

Web Systems and Technologies

Domain and Name Server Administration

Domain and Name Server Administration

Introduction

- Domain Name System (DNS) is used to resolve an IP address of a website
- Domain name registrars register website
  - GoDaddy
  - TuCows
  - Network Solutions

Domain and Name Server Administration

DNS

- Registration
  - WHOIS
    - Use whois command
  - Private registration
- Updating name servers
  - Takes up to 48 hours
  - Check status using dig command
  - https://toolbox.googleapps.com/apps/dig
- Reverse DNS

Domain and Name Server Administration

DNS Record Types

- Mapping (A/AAA)
- Mail Exchange (MX)
- Authoritative (NS/SOA)
- Validation (TXT/SPF)

Web Systems and Technologies

Search Engines

Search Engines

Introduction

- Before the search engines
  - Emails, message boards, websites
  - Web directories (vlib, yahoo, dmoz)
- Web Crawlers
- Google

Search Engines

Search Engine Overview

- Components of a search engine
  - Web Crawler, URL submission, …
  - Indexes
  - Query Server

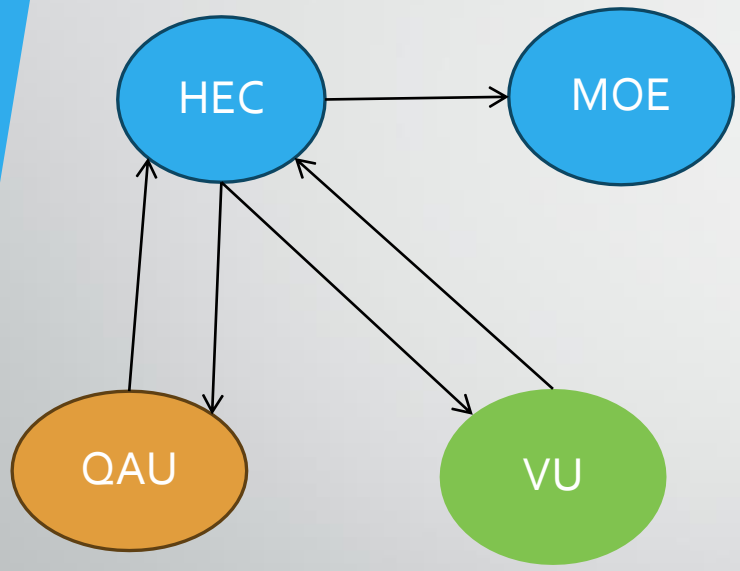
Web Systems and Technologies

PageRank

PageRank

- PageRank

- An algorithm for ranking websites
- Models WWW as a graph, example:



- On VU website

  HEC

- On QAU website

  HEC

- On HEC's website

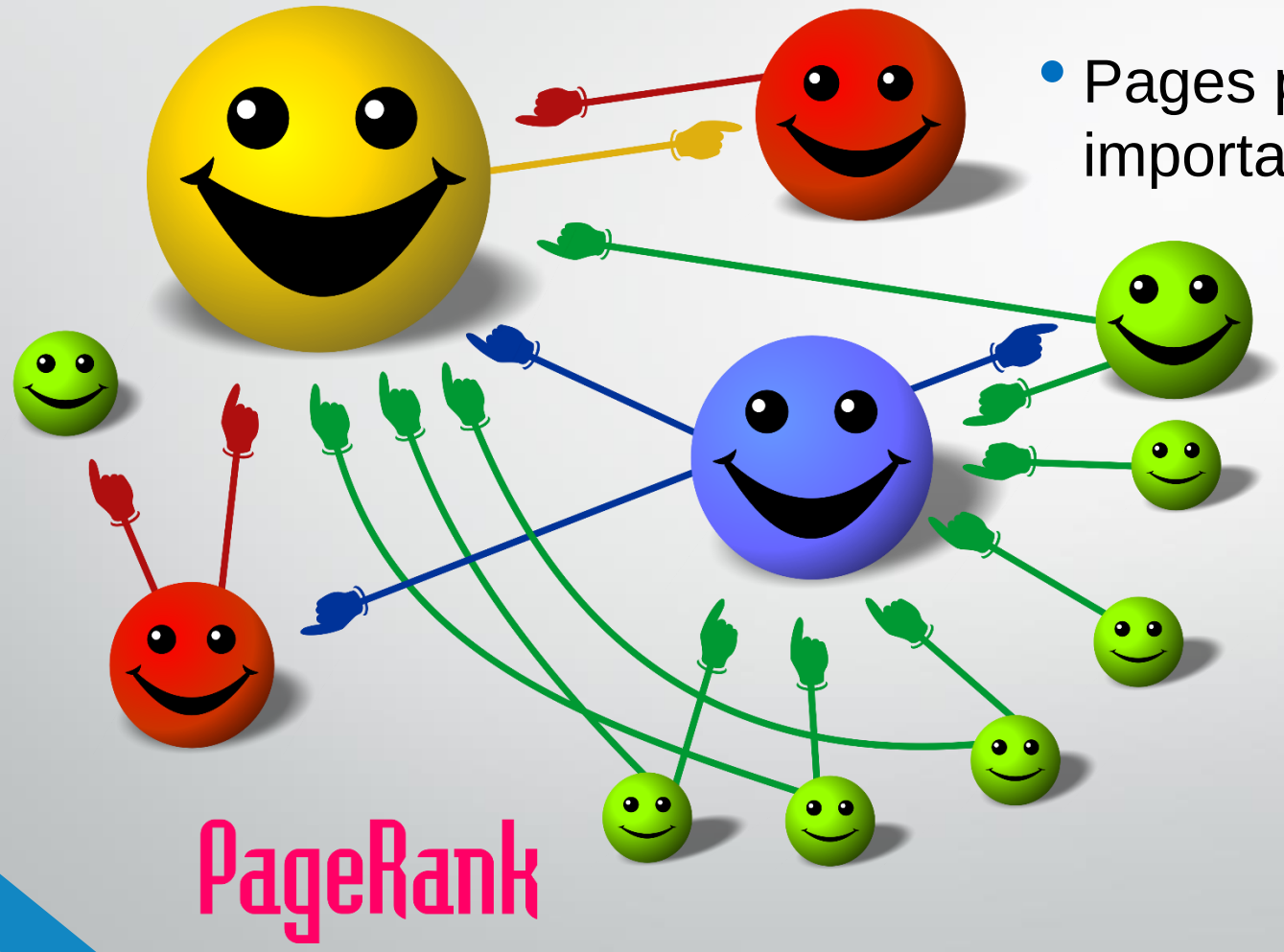
  VU

  QAU

  MOE

PageRank

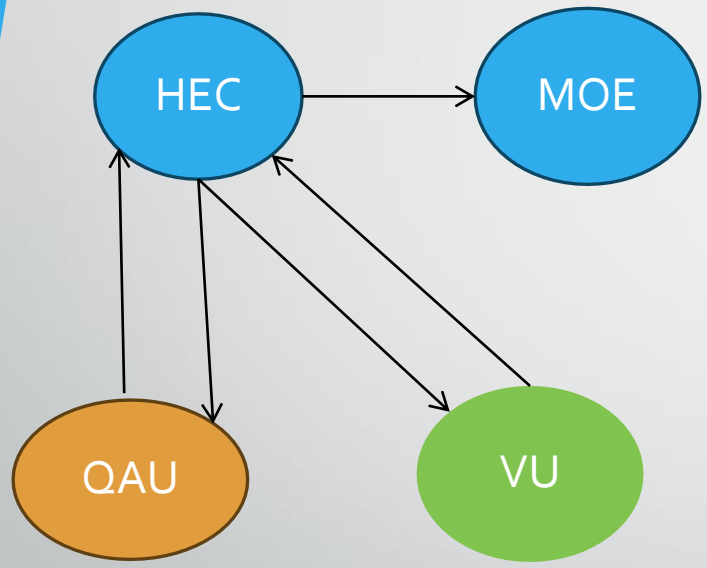
PageRank



- Pages pointed by important pages are important

PageRank

How PageRank Works



	HEC	VU	QAU	MOE
Iter 0	¼	¼	¼	1/4
Iter 1	¼+ ¼=1/2	1/12	1/12	1/12
Iter 2				
…				

Web Systems and Technologies

Search Engine Optimization

Search Engine Optimization

Introduction

- Search Engine Optimization(SEO) is a way to increase the rank of a website
  - White hat SEO
  - Black hat SEO

Search Engine Optimization

White-Hat SEO Techniques

- Title and Meta-data
  - Title, description, robot

/users/id/123

- Designing URLs
  - Descriptive paths and file names
- Website Design
  - Layout, readability, JavaScript, structure
- Sitemaps
- Anchors and Images
  - Anchor text, image name, alt
- Contents
  - Original, fresh, UGCs

Web Systems and Technologies

Black Hat SEO

Search Engines

Introduction

- Try to exploit search engine rankings
- Should not be used

Search Engines

Black-Hat SEO Techniques

- Content spamming
  - Keyword stuffing, hidden content, paid links, doorway pages
- Link spam
  - Hidden links, comment spam, link farms, and pyramids, google bombing
- Google bowling
- Cloaking
- Duplicate Contents

Web Systems and Technologies

Online Social Networks

Online social Networks

Introduction

- Social networks
- Free usage
- User profile
- Monetization
- Extensive API support

Online social Networks

Social Media Presence

- Home pages on social media platforms
- Link and logos
- Using URL shorteners

Online social Networks

Facebook Plugins

- Register as a developer
- Facebook Markup Language(FBML)
- Like and recommend buttons

```
<div class="fb-like"
    data-href="https://example.com/page.html"
    data-width="200"
    data-layout="standard"  data-action="like"
    data-size="small"
    data-share="true">
</div>
```

- More plugins

  https://devolopers.facebook.com/docs/plugins

Online social Networks

Twitter Widgets

- Twitter
- Widgets
  - Tweet this button

<a class="twitter-share-button"

href=https://twitter.com/intent/tweet?text=Hello World!">Tweet

<script async

src=https://platform.twitter.com/widgets.js charset="utf-8"></script>

- More widgets

  https://publish.twitter.com

Web Systems and Technologies

Monetization with Ads

Monetization with Ads

Introduction

- Ads on a website can generate revenue
- How web ads work?
- Ad Networks

Monetization with Ads

Ad Types

- Ad types
  - Graphics
  - Text
  - Dynamic

Monetization with Ads

Creating Ads

- Register for a Google AdSense (or other account)
- Get code through AdSense account
- Place it on your website

Monetization with Ads

Web Ads Economy

- Ad Commodities
  - Views
  - Clicks
  - Actions
- Web Commodity markets
  - Cost Per Click(CPC)
  - Cost Per Mile (CPM)
  - Cost Per Action(CPA)
  - ClickThrough Rate (CTR)
- Auctions

Web Systems and Technologies

Web Analytics

Web Analytics

Introduction

- Web analytics provide us insights about website usage
  - Bandwidth usage
  - Referrers
  - Popular domains/page
  - Search terms used for landing
  - User behavior
  - Demographics

Web Analytics

Web Metrics

- Page Views
- Unique page views
- Average visit duration
- Bounce rate

Web Analytics

Internal Analytics

- IP address
- User agent analysis
- Frequency analysis
- Can be provided by hosting company

Web Analytics

Third Party Analytics

- Google, Bing, …
- Dashboards
- Flow analysis
- In-page analytics

Web Systems and Technologies

Google Analytics-Demo