COS10024 Web Development

Lecture 10 – jQuery and server Side Scripting

Dr. Gamunu Dassanayake

TP3/2022

Contents

What is jQuery?

- jQuery is a lightweight, "write less, do more", JavaScript library.
- The purpose of jQuery is **to make it much easier to use JavaScript** on your website.
- jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
- jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

About jQuery?

- What You Should Already Know

  Before you start studying jQuery, you should have a basic knowledge of:

  - HTML
  - CSS
  - JavaScript

The jQuery library

The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

Why jQuery?

- There are lots of other JavaScript libraries out there, but jQuery is probably the most popular, and also the most extendable.

- Many of the biggest companies on the Web use jQuery, such as:
    - Google
    - Microsoft
    - IBM
    - Netflix

Will jQuery work in all browsers?

The jQuery team knows all about **cross-browser issues**, and they have written this knowledge into the jQuery library.

jQuery will run exactly the same in all major browsers.

Client Side Scripting

Frameworks

Scripting Frameworks

- The rise of scripting frameworks or libraries is one of the reasons for JavaScript's larger role in today's Web.
- The arguments for using a scripting framework is faster development, better code testing, and cross-browser reliability.
- An argument against using a scripting framework is the initial time required to learn a framework, and the size of the initial download of files to the client.

Scripting Frameworks

- When using a framework, it is important to remember that you are still programming in JavaScript.
- Being conversant with at least one framework is important for a JavaScript programmer.
- Using a framework may not be appropriate for some projects.

http://en.wikipedia.org/wiki/List_of_JavaScript_libraries
http://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks

jQuery
write less, do more.

Bootstrap

YUI LIBRARY

dōjō toolkit

mootools

… and lots more …

Framework: jQuery

- jQuery is a JavaScript library specialised for updating webpage documents on the fly.
- Library can be downloaded from www.jquery.com
- Download the compressed (production) version. This is a “minified” version intended for speed
- The filename format is jQuery-<version>.min.js

  Example: jquery-1.9.1.min.js

Adding jQuery to Your Web Pages

- There are several ways to start using jQuery on your web site. You can:
  - Download the jQuery library from jQuery.com
  - Include jQuery from a CDN, like Google

    (CDN – Content delivery Network or Content Distribution Network)

Downloading jQuery

- There are two versions of jQuery available for downloading:
  - **Production version** - this is for your live website because it has been minified and compressed
  - **Development version** - this is for testing and development (uncompressed and readable code)

- Both versions can be downloaded from jQuery.com.

Framework: jQuery

- The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag (notice that the <script> tag should be inside the <head> section).

- Include the jQuery library

```
<head>
<script src="jquery-1.9.1.min.js"></script>
<script src="myscript.js"></script>
</head>
```

  - Preferably the ***first*** script defined in the HTML

Don’t forget your own JavaScript!

- Selecting element(s) using jQuery
  - `$(<selector>)`

jQuery CDN

- If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).
- **Google** is an example of someone who host jQuery:
- **Google CDN:**

Demo – slide14.html

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
```

jQuery Syntax

- Basic syntax is: **$(*selector*).*action*()**
  - A $ sign to define/access jQuery
  - A (*selector*) to "query (or find)" HTML elements
  - A jQuery *action*() to be performed on the element(s)

Examples:

$(this).hide()	- hides the current element.
$("p").hide()	- hides all <p> elements.
$(".test").hide()	- hides all elements with class="test".
$("#test").hide()	- hides the element with id="test".

The element Selector

The jQuery element selector selects elements based on the element name.

You can select all <p> elements on a page like this:

$("p")

Example

When a user clicks on a button, all <p> elements will be hidden:

```
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
```

Demo – slide16.html

The #id Selector

- **The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.**
- **An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.**
- **To find an element with a specific id, write a hash character, followed by the id of the HTML element:**

  $("#test")

- **Example**

  When a user clicks on a button, the element with id="test" will be hidden:

```
$(document).ready(function(){
  $("button").click(function(){
    $("#test").hide();
  });
});
```

Demo – slide17.html

The .class Selector

- The jQuery .class selector finds elements with a specific class.
- To find elements with a specific class, write a period character, followed by the name of the class:

```
$(".test")
```

Example

When a user clicks on a button, the elements with class="**test**" will be hidden:

```
$(document).ready(function(){
  $("button").click(function(){
    $(".test").hide();
  });
});
```

Demo – slide18.html

More Examples of jQuery Selectors

Syntax	Description	
$("*")	Selects all elements	
$(this)	Selects the current HTML element	
$("p.intro")	Selects all <p> elements with class="intro"	Try it
$("p:first")	Selects the first <p> element	
$("ul li:first")	Selects the first <li> element of the first <ul>	Try it
$("ul li:first-child")	Selects the first <li> element of every <ul>	
$("[href]")	Selects all elements with an href attribute	
$("a[target='_blank']")	Selects all <a> elements with a target attribute value equal to "_blank"	
$("a[target!='_blank']")	Selects all <a> elements with a target attribute value NOT equal to "_blank"	Try
$(":button")	Selects all <button> elements and <input> elements of type="button"	
$("tr:even")	Selects all even <tr> elements	Try
$("tr:odd")	Selects all odd <tr> elements	

jQuery: Variable Assignment

Using JavaScript

```
var regForm =  document.getElementById("regform");
```

```
regForm.onsubmit = validate;
```

Using jQuery

```
$regForm = $("#regform");
```

```
$regForm.submit(validate);
```

jQuery: Property and Method

• **Using JavaScript**	• **Using jQuery**
element	*element*
• .onclick = *function*	• .click(*function*)
• .onsubmit = *function*	• .submit(*function*)
• .value	• .val()
• *.checked*	• .attr("checked")
• .style.*color*	• .css("color")
• .style.display="none"	• .css("display", "none")

JavaScript to jQuery: Events

Mouse Event	The event occurs when the user
click	clicks on an element
dblclick	double-clicks on an element
hover	Moves in and/or out an element
mouseup	releases a mouse button over an element
mousedown	presses a mouse button over an element
mousemove	moves while it is over an element
mouseover	moves into an element
mouseout	moves out of an element
mouseenter	moves into a bound not descendant element
mouseleave	moves out of bound not descendant element

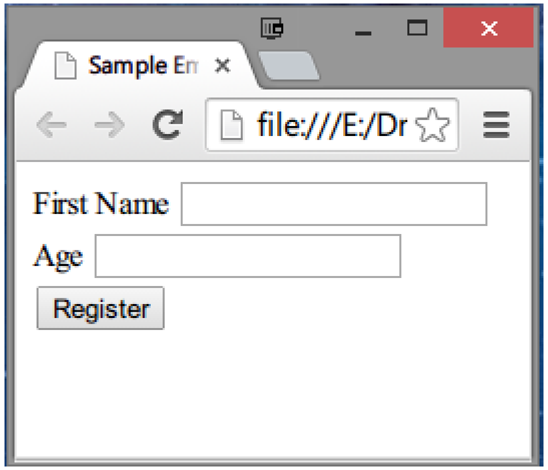
JavaScript to jQuery: Events

keydown	is pressing a key
keypress	presses a key
keyup	releases a key
blur	a form element loses focus
focus	an element gets focus
focusin	an element gets focus
focusout	a form element loses focus
change	the content of a form element, the selection, or the checked state have changed
select	a user selects some text
submit	a form is submitted

jQuery: Form Data Validation

Given the following HTML form:

```
<form id="regForm" method="post" action="process.php">
    <div class="textinput">
        <label for="firstname">First Name</label>
        <input type="text" name="firstname"
                                       id="firstName" />
    </div>
    <div class="textinput">
        <label for="age">Age</label>
        <input type="text" name="age" id="age" />
    </div>
    <div class="buttoninput">
        <input type="submit" value="Register" >
    </div>
</form>
```



jQuery: Form Data Validation [2]

```
function validateForm () {
/* validation code here */
   return true/false;
}
```

Write the validation code, and return **`true`** if valid, otherwise **`false`.**

```
function init() {
 $("#regForm").submit(validateForm);
}
```

Form ID

JavaScript: **var** regForm = document.getElementById("**regForm**");
regForm.onsubmit=validateForm;

```
$(document).ready(init);
```

Don't forget this!

↓

JavaScript: window.onload(init);

jQuery: Form Data Validation [3]

- Validation Function – Part 1

```
function validateForm() {
 var firstName = $("#firstName").val();
 var age = $("#age").val();
 var errMsg = "";
 var result = true;
```

Get the `value` property of an input element, e.g., text box or select.

jQuery: Form Data Validation [4]

- Validation Function– Part 2

```
  if (firstName == "") {
   errMsg += "First Name cannot be empty.\n";
 }
 if (age == "") {
    errMsg += "Age cannot be empty.\n";
 }
 if (isNaN(age)) {
    errMsg += "Age is not a valid number.\n";
 }
 if (errMsg != "") {
    alert (errMsg);
    result = false;
 }
 return result;
}
```

If **`false`** is returned here, the form would not be submitted.

JQuery

jQuery Effects

jQuery Effects

Effect	Description
.hide()	Hide the selected elements DEMO!
.show()	Display the selected elements
.toggle()	Display or hide the selected element
.fadeIn()	Display the selected elements by fading in DEMO!
.fadeOut()	Hide the selected elements by fading out
.slideDown()	Display in sliding motion the selected elementsDEMO!
.slideUp()	Hide in sliding motion the selected elements
.slideToggle()	Display or hide in sliding motion selected element
.animate()	Display with customs css property value

jQuery Effects [1]

Example

- **$("sidTip").show();**
- **$("sidTip").hide();**
- **$("sidTip").fadeIn();**
- **$("sidTip").fadeOut();**
- **$("sidTip").slideUp();**
- **$("sidTip").slideDown();**
- **$("sidTip").animate({height:"300px"});**

Duration parameters on all effects such as `"slow"`, `"fast"`, or a number can be specified.

DEMO!

jQuery Effects [2]

- JavaScript

```
function showTip () {
  sidTip.style.display
     = "inline";
}

function hideTip () {
  sidTip.style.display
     = "none";
}
```

- jQuery 1

```
function showTip () {
  $("#sidTip").show();
}

function hideTip () {
  $("#sidTip").hide();
}
```

jQuery Effects [3]

- **jQuery 2**

```
function showTip (){

$("#sidTip").fadeIn();
}
function hideTip () {

$("#sidTip").fadeOut()
;
}
```

- **jQuery 3**

```
function showTip () {
  $("#sidTip").slideUp();
}


function hideTip () {
$("#sidTip").slideDown();
}
```

Handling Arrays

Array Objects

- **Selecting Multiple Elements**

  $h1s = $("h1");

  $elements =$(".red");

DEMO!

- **jQuery provides methods that access elements in an array:**

  $element.**first()** – get first element

  $element.**eq(<#>)** – get element by index #

  $element.**last()** – get last element

Array Objects: HTML Code

Given the following

```
        <article>
ps[0]      <p>1</p>
           <section>
ps[1]         <p>1.1</p>
ps[2]         <p>1.2</p>
           </section>
ps[3]      <p>2</p>
           <section>
ps[4]         <p>2.1</p>
ps[5]         <p>2.2</p>
           </section>
        </article>
```

index	ps[index]
0	`<p>1</p>`
1	`<p>1.1</p>`
2	`<p>1.2</p>`
3	`<p>2</p>`
4	`<p>2.1</p>`
5	`<p>2.2</p>`

- Retrieve all `<p>` elements will retrieve all paragraph elements **in the order they appear in the HTML**

```
var ps = $("p");
```

Array Objects: JavaScript

- Retrieve all <p> elements using

```
var p1s = document.getElementsByTagName("p");
```

i	ps[i].innerHTML
0	1
1	1.1
2	1.2
3	2
4	2.1
5	2.2

- Sample loop code

```
var p1s =
document.getElementsByTagName
("p");
var i;

for(i = 0;i < p1s.length;i++)
{
   alert(p1s[i].innerHTML);
}
```

SWINBURNE UNIVERSITY OF TECHNOLOGY

Array Objects: jQuery

- Retrieve all <p> elements using

```
var $p3s = $("p");
```

i	$ps.eq(i).html()
0	1
1	1.1
2	1.2
3	2
4	2.1
5	2.2

- Sample loop code

```
$p3s = $("p");
var i;
for(i = 0;i<$p3s.length;i++)
{
 alert($p3s.eq(i).html());
}
```

JavaScript:
`p2s[i].innerHTML`

`$p3s.first()`

`$p3s.last()`

SWINBURNE UNIVERSITY OF TECHNOLOGY

jQuery

Handling objects

Handling Objects

- **jQuery provides methods that allows one to add or remove HTML elements.**
- **First, get access to the target element: `$e = $(…);`**
- **Methods used to dynamically insert elements**
  - `$e.before("<p>Paragraph</p>");`
  - `$e.after("<hr />");`
- **Methods used to select previous and next sibling elements**
  - `$e.prev():` get the previous sibling element
  - `$e.next():` get the next sibling element

DEMO!

Handling Objects

- For example to interactively create a tooltip on focus, we have

```
function showAgeTip() {
   $(this).after("<span class='tooltip'>
         must be above 18yo</span>");
}
function hideAgeTip() {
   $(this).next().remove();
}
function init () {
   $("#tbAge").focusin(showAgeTip);
   $("#tbAge").focusout(hideAgeTip);
}
```

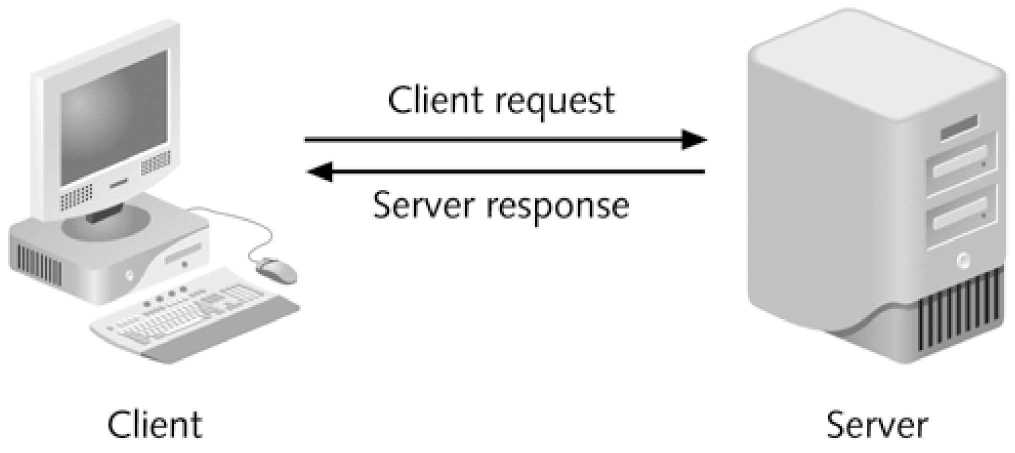
DEMO!

Server-Side Web development

Introduction

Client/Server Architecture

- A system consisting of a Client and a Server is known as a **two-tier system**



The design of a two-tier client/server system

Client/Server Architecture [2]

- **Client ("front end"):**
  - Presents an interface to the user.
  - Gathers information from the user, submits it to a server, then receives, formats, and presents the results returned from the server.
- **Server ("back end"):**
  - Fulfills a request for information by managing the request or serving the requested information to the client.
  - A computer from which a client requests information.
  - Responsible for data storage and management.

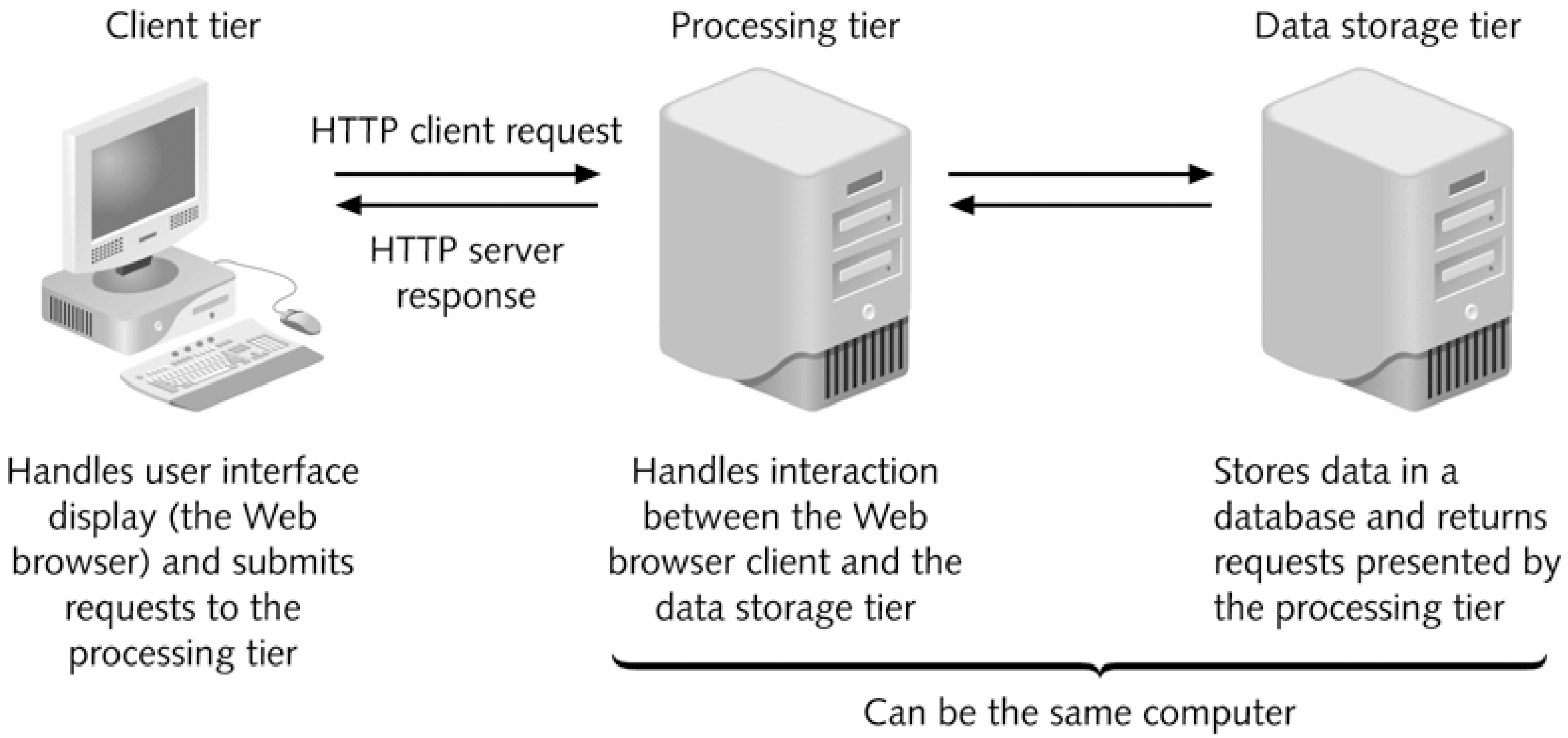
Client/Server Architecture [3]

- **A three-tier, or multi-tier, client/server system consists of three distinct pieces:**
  - **Client tier**, or **user interface tier**,
    is the **Web browser.**
  - **Processing tier**, or **middle tier**, handles the interaction between the Web browser client and the **data storage tier.**

- Performs necessary processing or calculations based on the request from the client tier.

- Handles the return of any information to the client tier.

Client/Server Architecture [4]


Client tier
Processing tier
Data storage tier
HTTP client request
HTTP server response
Handles user interface display (the Web browser) and submits requests to the processing tier
Handles interaction between the Web browser client and the data storage tier
Stores data in a database and returns requests presented by the processing tier
Can be the same computer

The design of a three-tier client/server system

Server-Side Web development

Server-side includes

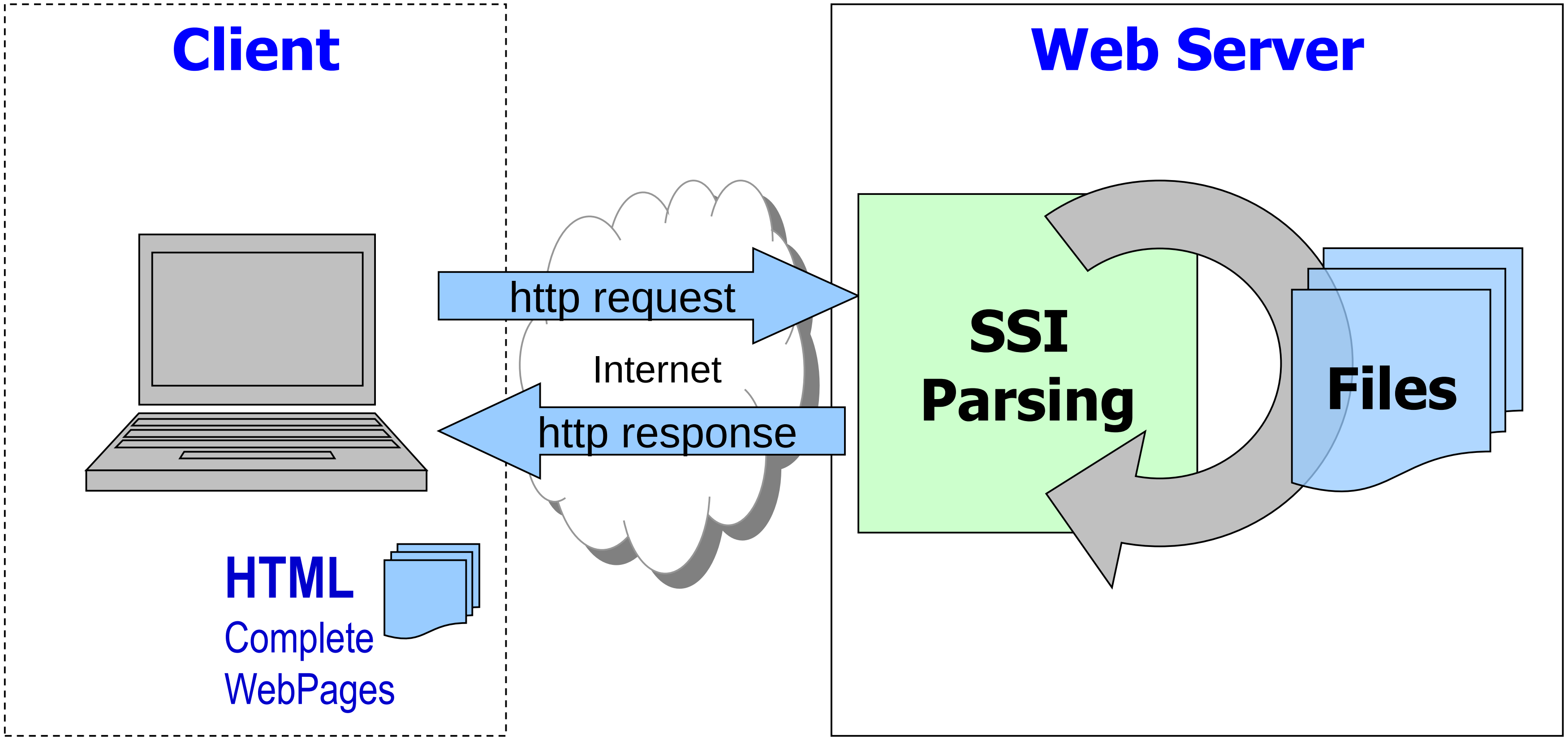
What are Server Side Includes(SSI)?

- SSI is a simple web server based method of dynamically *including* (adding) information into HTML pages *before* the normal http server response back to the client.
- SSI provides an easy way to include *common static content,* into many pages, such as a standard "header", "menu" or "footer".
- SSI also allows a small amount of *dynamic content,* to be included in the HTML pages in the response, such as file location and file last modified date etc.,

SSI … is a kind of "Copy-n-Paste" by the web server!

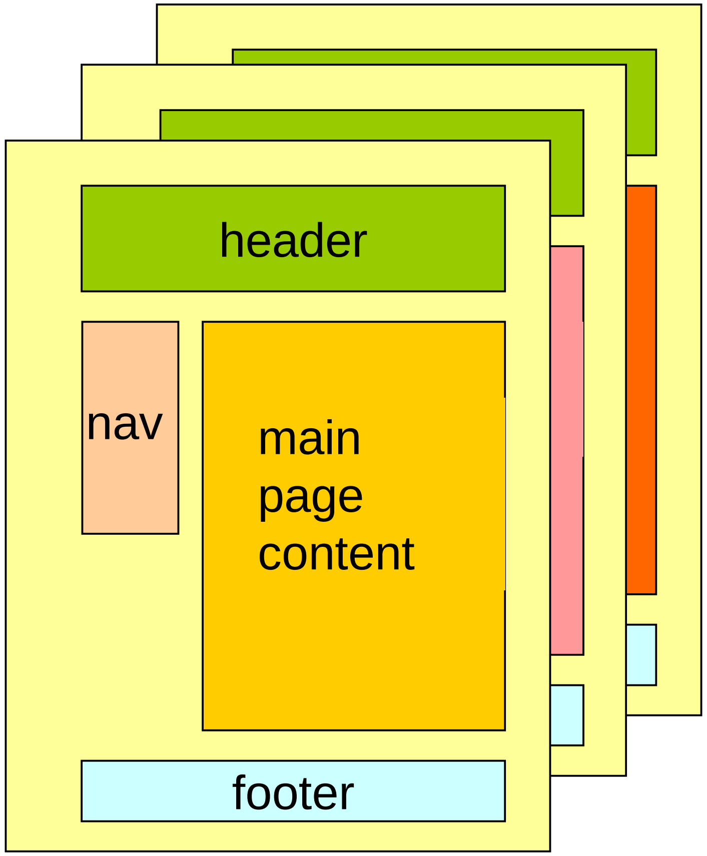
SSI - Server Side Includes

- ***Web server does the work of a script "parser"…***



SSI Example – DEMO!

Typical simple website of ***static pages***.

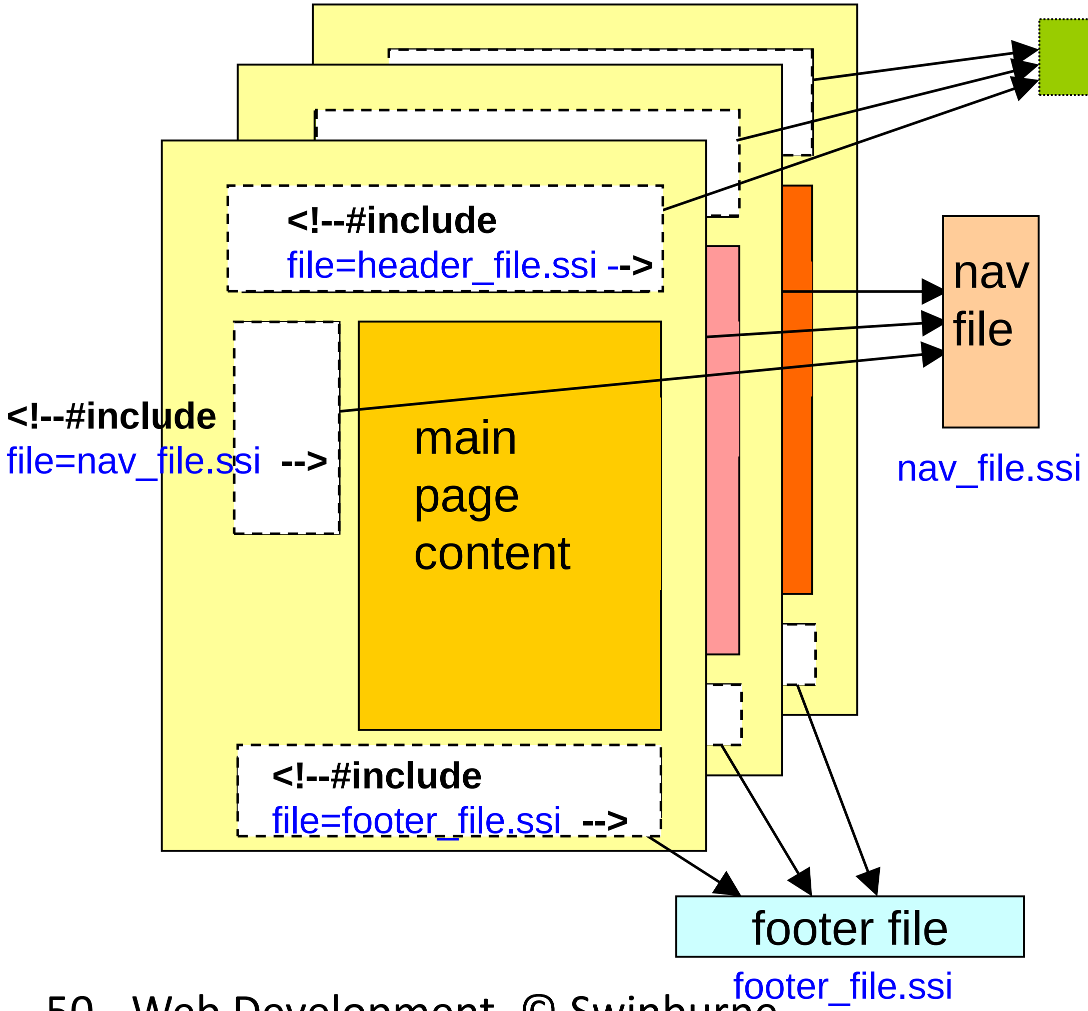


Each page usually contains the ***same duplicated code*** for header, nav, and footer
…. *with different 'main' content*

Any change in the content of *header*, *nav* and *footer*, means changing this same content in ***every*** page. ☹

SSI Example [1]

Typical simple website with SSI



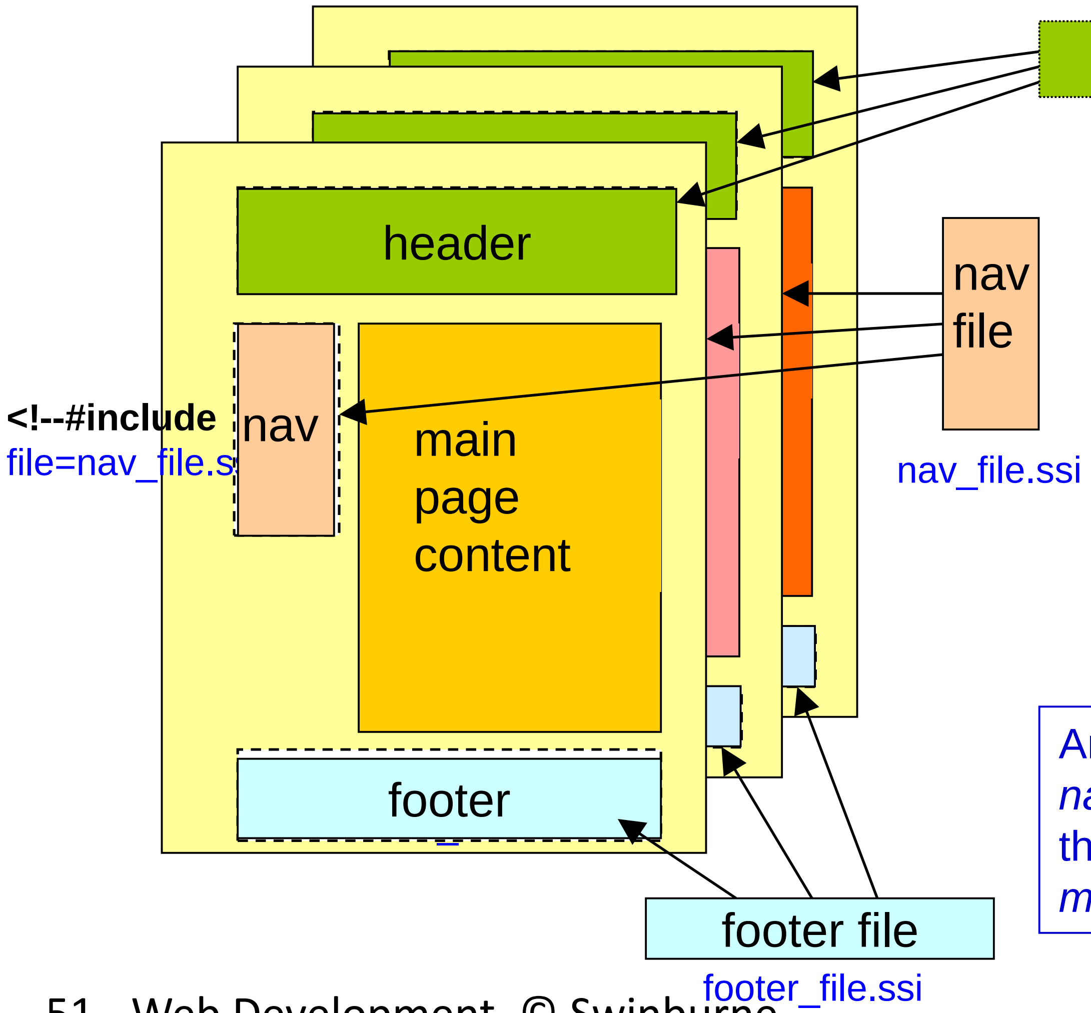
File Separation:

The common code for header, nav and footer is removed and placed in ***separate files.***

#include statements are substituted in each webpage.

SSI Example [2]

Typical simple website with SSI



Server Parsing:

The server ***parses*** the pages and adds the ***#include*** information ***prior*** to sending the response back to the client.

The client receives the ***complete*** webpage - with the *content of the* ***#include*** *added*.

Any change in the content of *header*, *nav* and *footer*, means changing ***only*** the include files.☺ Simplifies *Code maintenance.*

SSI == Web Server Work

How does it work ?

- HTML pages are *"parsed"* by the server which looks for ***special tags***
  eg <!--#include ... -->
- The server ***replaces the tags*** with the requested information.
  Delivers the completed combined page.
- SSI depends on server ability and configuration.
  Pages may need to have a special file extension to indicate that they should be parsed before being sent to clients, ***such as .shtml***

  - The web server does the work.
    So there is no need for other software tools, or performance pass-through issues. ☺

However, as pages are no longer static, each page requires server-side processing!

SSI - Tags Syntax

- **All SSI tags have the following syntax**

```
<!--#element  attribute="value"  attribute="value" ... -->
```

Note: If the tag is *not* processed by the server, the html comment will hide the unprocessed tag from being displayed by a normal html client.

- **The "#element" can be a number of things. For example:**

#include	insert the contents of a file (or script result)
#echo	print the value of variables
#flastmod	show the file last modified date
#exec	used to execute commands

Work through the examples in the SSI Lab, for a better understanding.

SSI - Examples

- **Include a standard footer in the page:**

```
<!--#include file="footer.html" -->
```

or

```
<!--#include virtual="../footer.html" -->
```

Note that there is *no space* after <!--
"<!-- #include …"
will not work

- **Include the result of a "hit counter" from a Perl script:**

```
<!--#include virtual="/cgi-bin/counter.pl" -->
```

- **Execute and display a CGI program result:**

```
<!--#exec cgi="/cgi-bin/example.cgi" -->
```

- **Show today's date from the server:**

```
<!--#echo var="DATE_LOCAL" -->
```

- **Show the last modified date for the file "index.html":**

```
<!--#flastmod file="index.html" -->
```

SSI - Features

- SSI #include tags can also be nested for added flexibility.

- Some SSI implementations allow the setting of variables which can be used dynamically:

```
<!--#set var="name" value="Fred" -->
```

- Some web servers **(ie. Apache)** also allow “conditional” statements:

```
<!--#if expr="test_condition" -->
<!--#elif expr="test_condition" -->
<!--#else -->
<!--#endif -->
```

Note:
Different servers are configured differently, so you always need to test your SSI on the server that will be used to host your website.

SSI – Apache Tutorial

- **Apache Tutorial: Introduction to Server Side Includes** **http://httpd.apache.org/docs/current/howto/ssi.html**
  - *"SSI (Server Side Includes) are directives that are placed in HTML pages, and evaluated on the server while the pages are being served.*
    They let you add dynamically generated content to an existing HTML page, without having to serve the entire page via a CGI program, or other dynamic technology."
  - *"SSI is a great way to add small pieces of information, such as the current time. But if a majority of your page is being generated at the time that it is served, you need to look for some other solution."*

SSI - When to Use?

- **SSI is an easy, simple and effective technique for adding some dynamic content to HTML pages.**
- **If other more capable techniques are being used to dynamically generate content, such as embedded scripting (ASP, PHP, etc) or CGI scripts, it is better to *only use one tool* - rather than use many different tools to handle "includes".**

eg. PHP: `<?php include("myheader.ssi"); ?>`

ASP: `<!--#include file="wisdom.inc"-->`

Note: If the "include" file *also* needs be parsed on the server, then the file type will need to be set appropriately. eg .shtml, .php, .asp etc, and you will need to check how deeply embedded the server will parse.

Server-Side Web development

a quick look at PHP

What is Embedded Scripting ?

- *Embedded Scripts* are scripts that are *embedded* or linked into HTML documents, and stored on the server.

- In response to client requests, the *called pages are "parsed"* by the *server software*, the *embedded scripts are "processed"* and the requested information or content is *returned as formatted html*.

- Client requests usually include parameters (key=value pairs) that are passed to the server, so the embedded scripts might query databases, or retrieve other dynamic information.

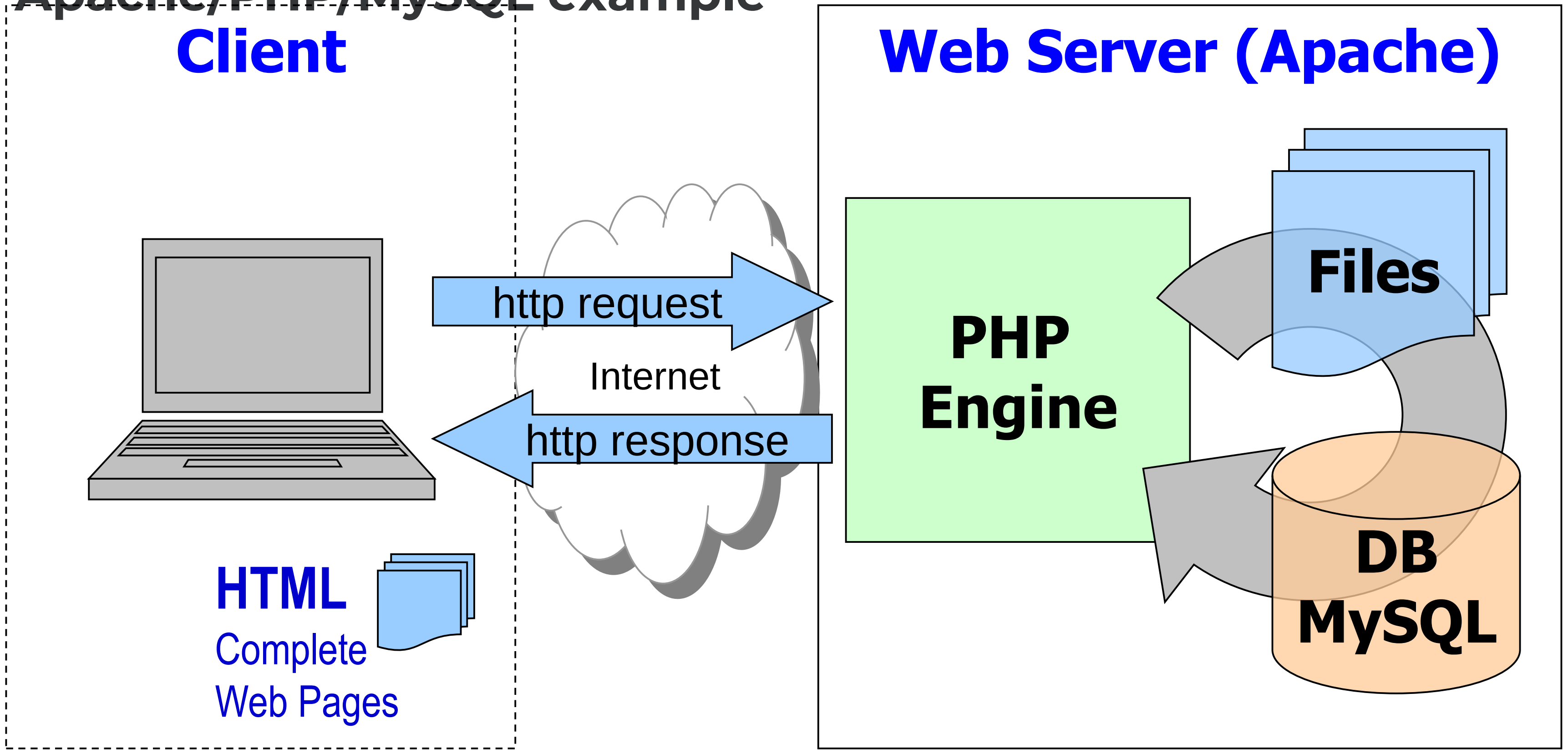
- The client response is *(usually) browser independent,* as it returns *"plain html".*

- The embedded script *is not visible to the client*

  - the client only sees the completed html page.

Embedded Scripting

Apache/PHP/MySQL example



Embedded Scripting and PHP

- **Server-side scripting refers to a scripting language that is executed from a Web server.**

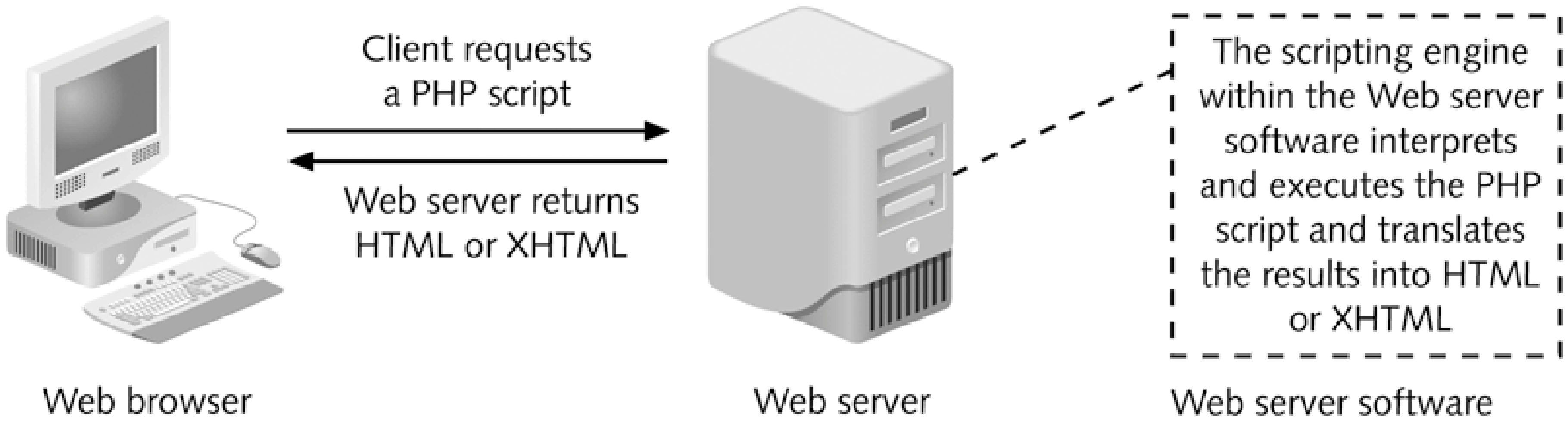
- **PHP is a server-side *embedded* scripting language that is used to develop interactive Web sites.**
  - Is easy to learn
  - Includes object-oriented programming capabilities
  - Supports many types of databases
    (MySQL, Oracle, Sybase, ODBC-compliant)

Embedded Scripting and PHP [2]

- **PHP (continued):**
  - PHP is an open source programming language.

- Open source refers to software where source code can be freely used and modified.
  - Can't access or manipulate a Web browser, like JavaScript.
  - Exists and executes solely on a Web server, where it performs various types of processing or accesses databases.

Embedded Scripting and PHP (continued)


Client requests a PHP script
Web server returns HTML or XHTML
The scripting engine within the Web server software interprets and executes the PHP script and translates the results into HTML or XHTML
Web browser
Web server
Web server software

- **General rule:**
  Use *client-side scripting* to handle user interface processing and light

- **processing, such as data validation; use *server-side scripting* for intensive calculations and data storage.**

How a Web server processes a PHP script

Quick PHP Intro

What is PHP? http://www.php.net

- PHP stands for PHP: Hypertext Preprocessor.
- PHP is a server-side scripting language, like ASP.
- PHP scripts are executed on the server.
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.).
- PHP is an open source software (OSS).
- PHP is free to download and use.
- PHP filename .php

Source: **w3schools**

PHP Example...

```
<html>
...
<body>
<h1>Hello World!</h1>
<?php
 echo "<p>";
 $i=1;
 while($i<=5) {
   echo "The number is " . $i . "<br />";
   $i++;
 }
 echo "</p>";
?>
</body>
</html>
```

You are not expected to be able to create any PHP scripts.
This example is just to help you understand Embedded Scripting Concepts.

Embedded PHP Script is processed on the Server, before it is sent to the client

SWINBURNE UNIVERSITY OF TECHNOLOGY

Next Lecture

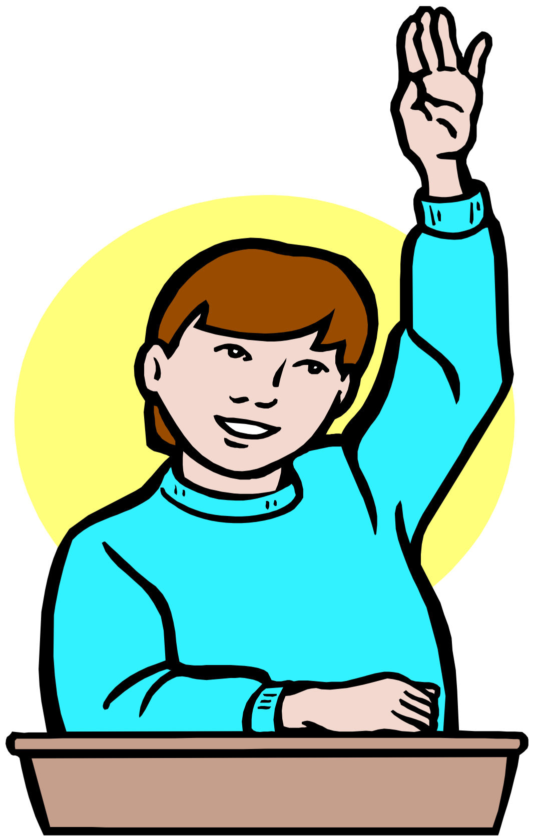
Introduction to XML

Reminder

- Week – 10 Lab Submission
- Online Quiz 9 (Week 10)
- Assignment 2 (Due Week 12)

Question?

- A good question deserve a good grade...



Thanks Lot!!!