

WEB TECHNOLOGY AND ITS APPLICATIONS

WEB TECHNOLOGY AND ITS APPLICATIONS
[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2017 - 2018)

SEMESTER – VII

Subject Code	17CS71	IA Marks	40
Number of Lecture Hours/Week	04	Exam Marks	60
Total Number of Lecture Hours	50	Exam Hours	03

CREDITS – 04

Module – 1	Teaching Hours
Introduction to HTML, What is HTML and Where did it come from?, HTML Syntax, Semantic Markup, Structure of HTML Documents, Quick Tour of HTML Elements, HTML5 Semantic Structure Elements, Introduction to CSS, What is CSS, CSS Syntax, Location of Styles, Selectors, The Cascade: How Styles Interact, The Box Model, CSS Text Styling.	10 Hours
Module – 2	
HTML Tables and Forms, Introducing Tables, Styling Tables, Introducing Forms, Form Control Elements, Table and Form Accessibility, Microformats, Advanced CSS: Layout, Normal Flow, Positioning Elements, Floating Elements, Constructing Multicolumn Layouts, Approaches to CSS Layout, Responsive Design, CSS Frameworks.	10 Hours
Module – 3	
JavaScript: Client-Side Scripting, What is JavaScript and What can it do?, JavaScript Design Principles, Where does JavaScript Go?, Syntax, JavaScript Objects, The Document Object Model (DOM), JavaScript Events, Forms, Introduction to Server-Side Development with PHP, What is Server-Side Development, A Web Server's Responsibilities, Quick Tour of PHP, Program Control, Functions	10 Hours
Module – 4	
PHP Arrays and Superglobals, Arrays, \$_GET and \$_POST Superglobal Arrays, \$_SERVER Array, \$_Files Array, Reading/Writing Files, PHP Classes and Objects, Object-Oriented Overview, Classes and Objects in PHP, Object Oriented Design, Error Handling and Validation, What are Errors and Exceptions?, PHP Error Reporting, PHP Error and Exception Handling	10 Hours
Module – 5	
Managing State, The Problem of State in Web Applications, Passing Information via Query Strings, Passing Information via the URL Path, Cookies, Serialization, Session State, HTML5 Web Storage, Caching, Advanced JavaScript and jQuery, JavaScript Pseudo-Classes, jQuery Foundations, AJAX, Asynchronous File Transmission, Animation, Backbone MVC Frameworks, XML Processing and Web Services, XML Processing, JSON, Overview of Web Services.	10 Hours

MODULE 3 - SYLABUS

- JavaScript: Client-Side Scripting, What is JavaScript and What can it do?, JavaScript Design Principles, Where does JavaScript Go?, Syntax, JavaScript Objects, The Document Object Model (DOM), JavaScript Events, Forms, Introduction to Server-Side Development with PHP, What is Server-Side Development, A Web Server's Responsibilities, Quick Tour of PHP, Program Control, Functions

JavaScript: Client-Side Scripting

Chapter 6

Section 1 of 8

WHAT IS JAVASCRIPT

What is JavaScript

- JavaScript runs right inside the browser
- JavaScript is dynamically typed (Weakly Typed)
- JavaScript is object oriented in that almost everything in the language is an object
- The objects in JavaScript are prototype-based rather than class-based, which means that while JavaScript shares some syntactic features of PHP, Java or C#, it is also quite different from those languages

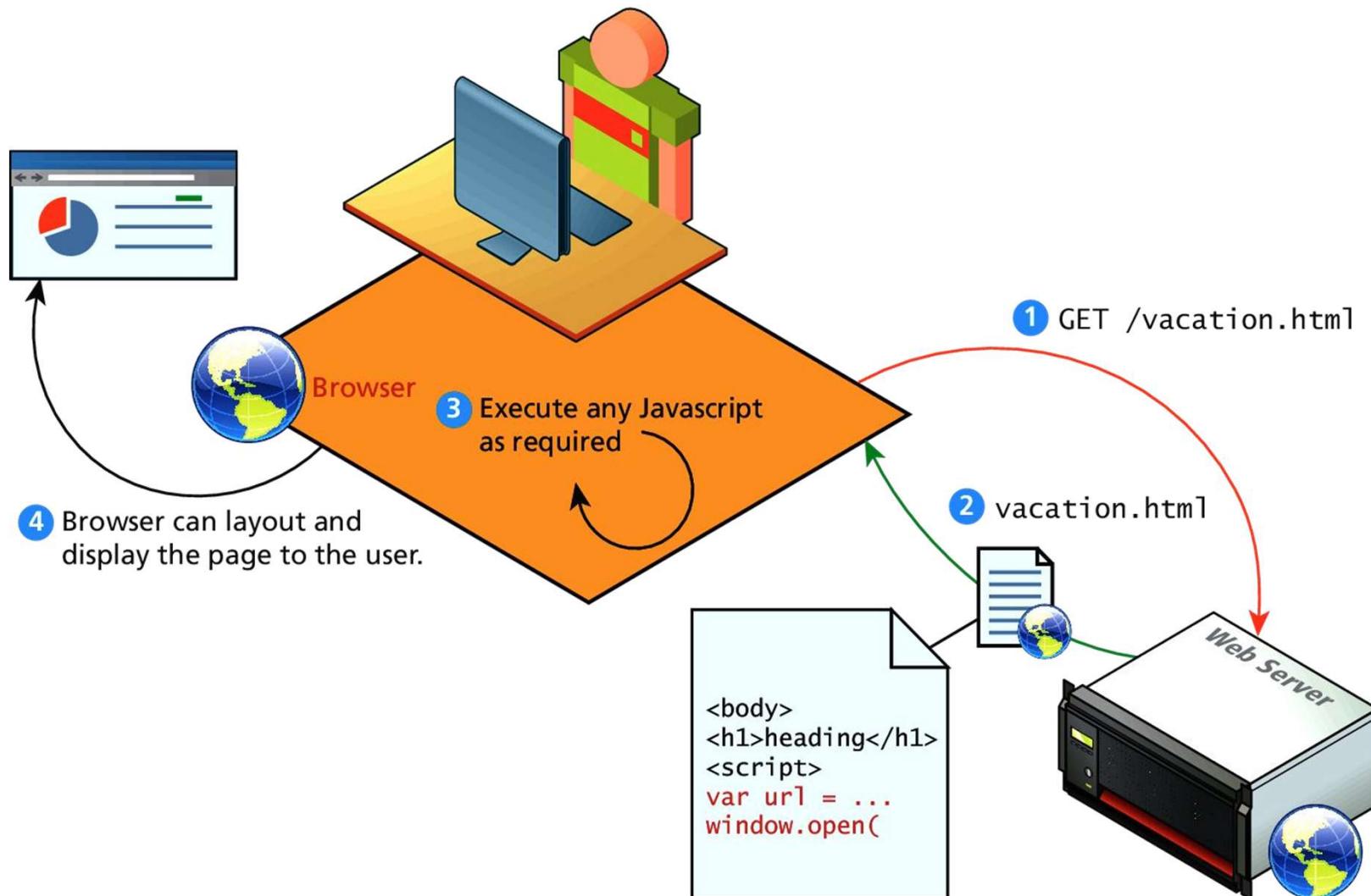
What isn't JavaScript

It's not Java

- ❖ Although it contains the word *Java*, JavaScript and Java are vastly different programming languages with different uses.
- ❖ Java is a full-fledged compiled, object-oriented language, popular for its ability to run on any platform with a JVM installed.
- ❖ Conversely, JavaScript is one of the world's most popular languages, with fewer of the object-oriented features of Java, and runs directly inside the browser, without the need for the JVM.

Client-Side Scripting

Let the client compute



Client-Side Scripting

It's good

There are many **advantages** of client-side scripting:

- Processing can be offloaded from the server to client machines, thereby reducing the load on the server.
- The browser can respond more rapidly to user events than a request to a remote server ever could, which improves the user experience.
- JavaScript can interact with the downloaded HTML in a way that the server cannot, creating a user experience more like desktop software than simple HTML ever could.

Client-Side Scripting

There are challenges

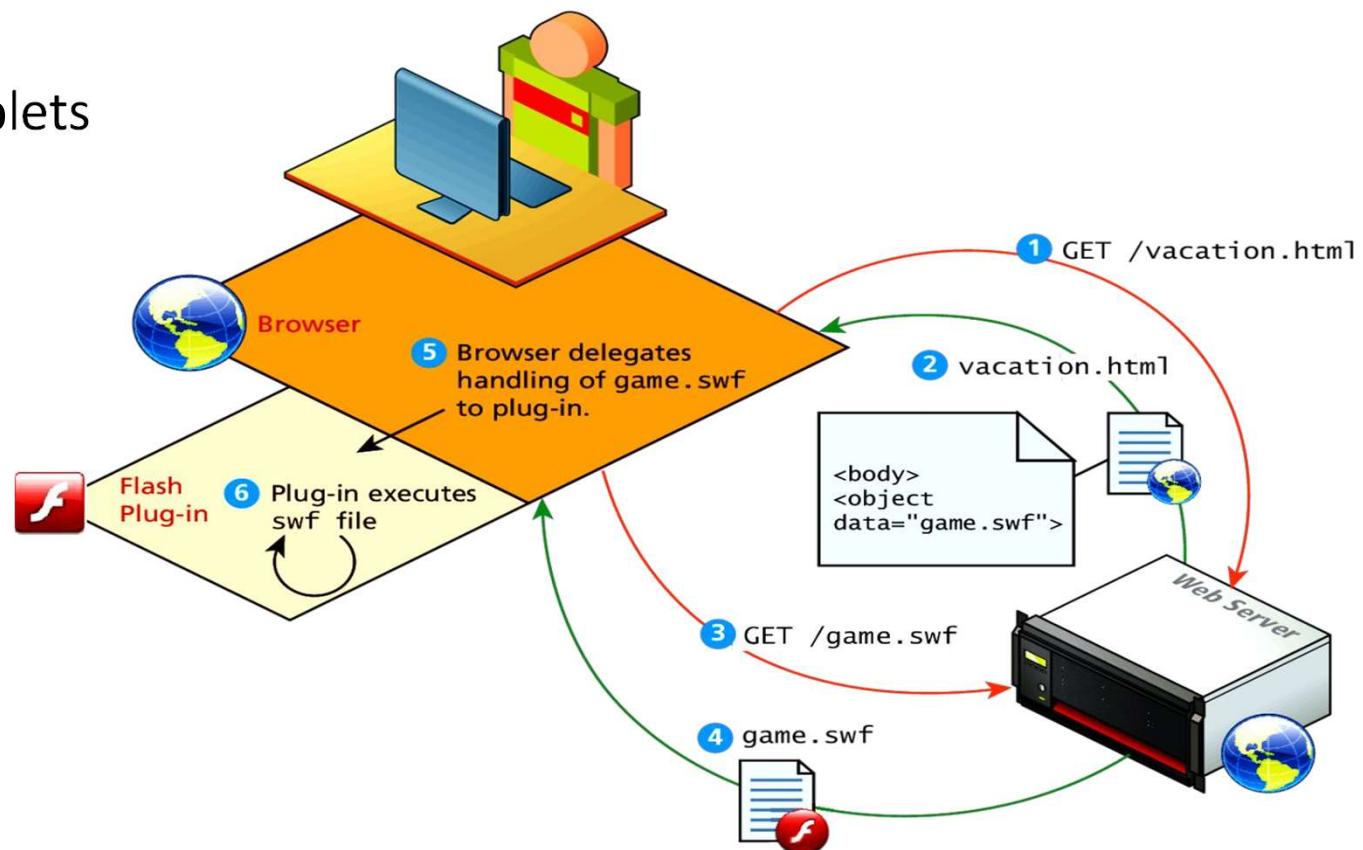
The disadvantages of client-side scripting are mostly related to how programmers use JavaScript in their applications.

- There is no guarantee that the client has JavaScript enabled
- The idiosyncrasies between various browsers and operating systems make it difficult to test for all potential client configurations. What works in one browser, may generate an error in another.
- JavaScript-heavy web applications can be complicated to debug and maintain.

Client-Side Flash

JavaScript is not the only type of client-side scripting.

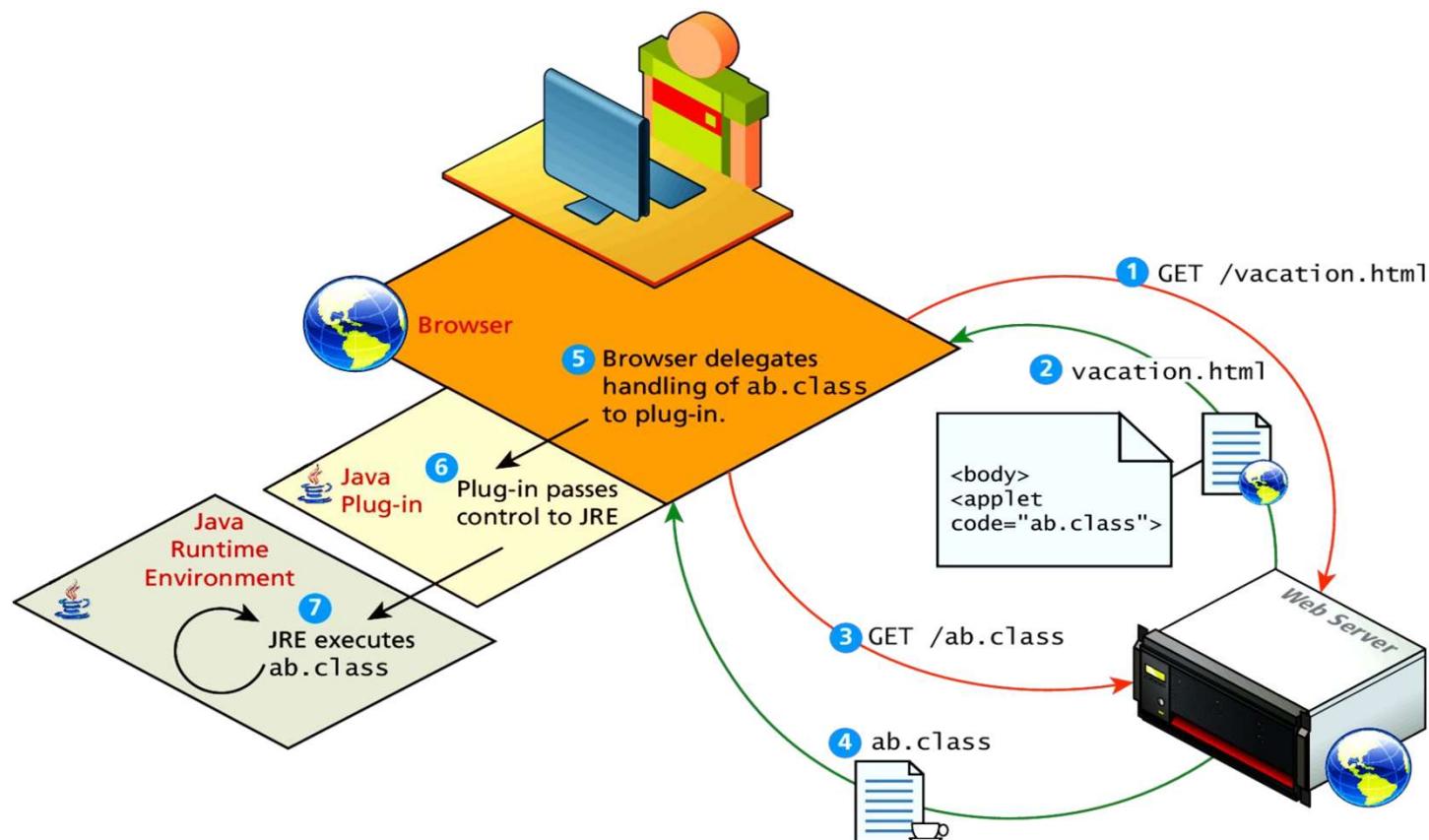
- Flash
- Java Applets



Client-Side Applets

Java Applets

Java applets are written in and are separate objects included within an HTML document via the <applet> tag

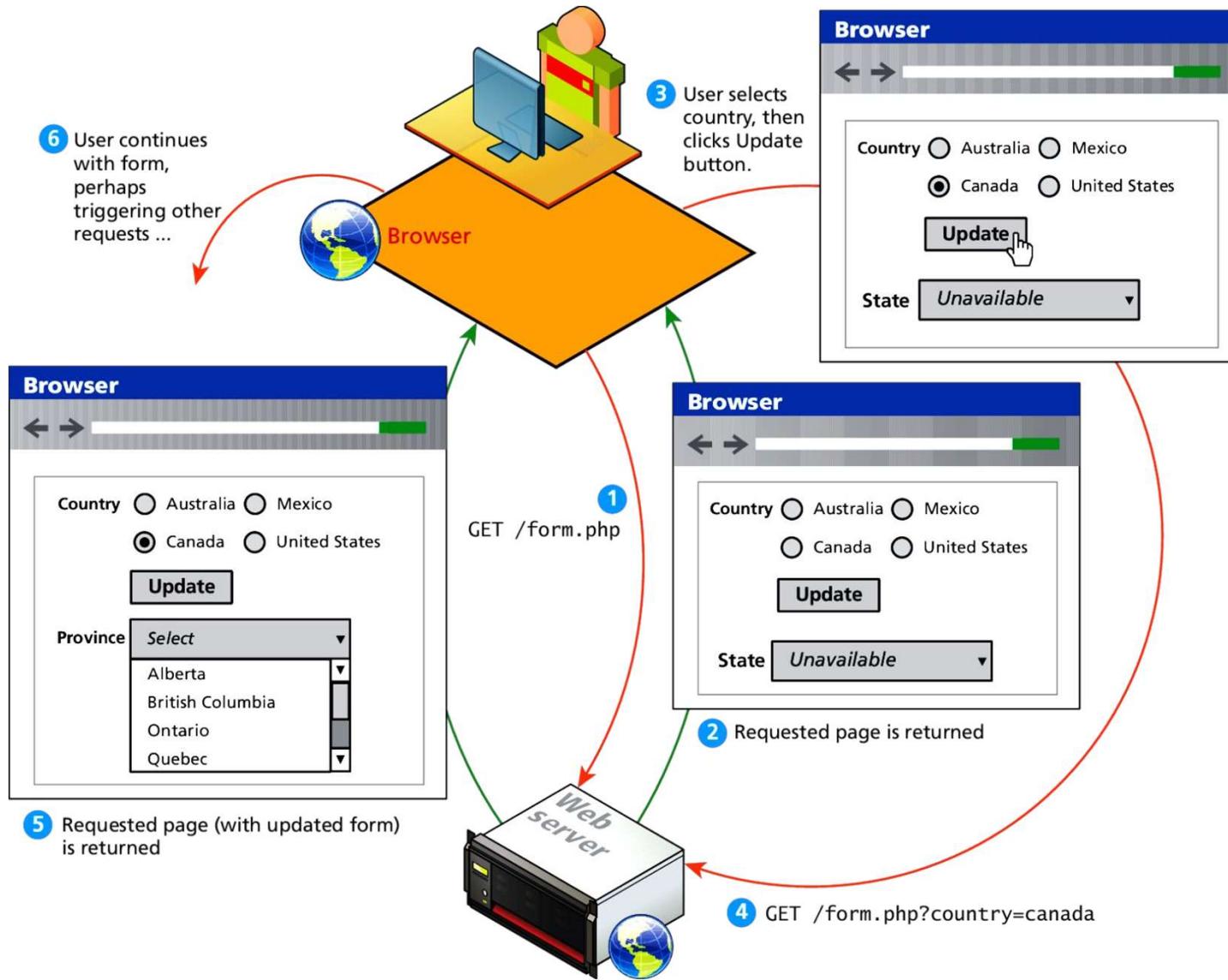


JavaScript History

- JavaScript was introduced by Netscape in their Navigator browser back in 1996.
- It was originally called LiveScript
- JavaScript is in fact an implementation of a standardized scripting language called **ECMAScript**
- JavaScript was only slightly useful, and quite often, very annoying to many users

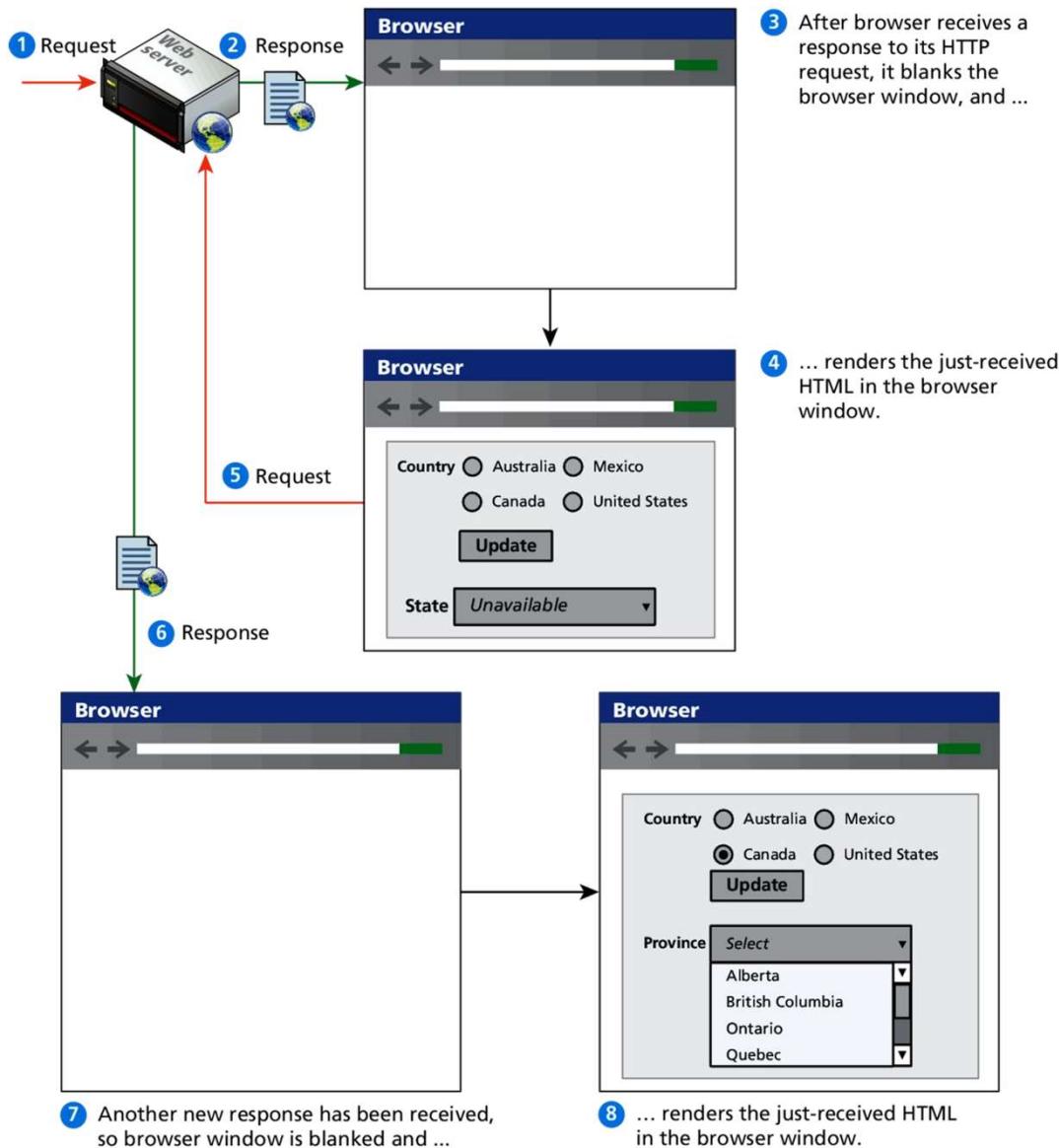
HTTP request-response loop

Without JavaScript



HTTP request-response loop

Detail



JavaScript in Modern Times

AJAX

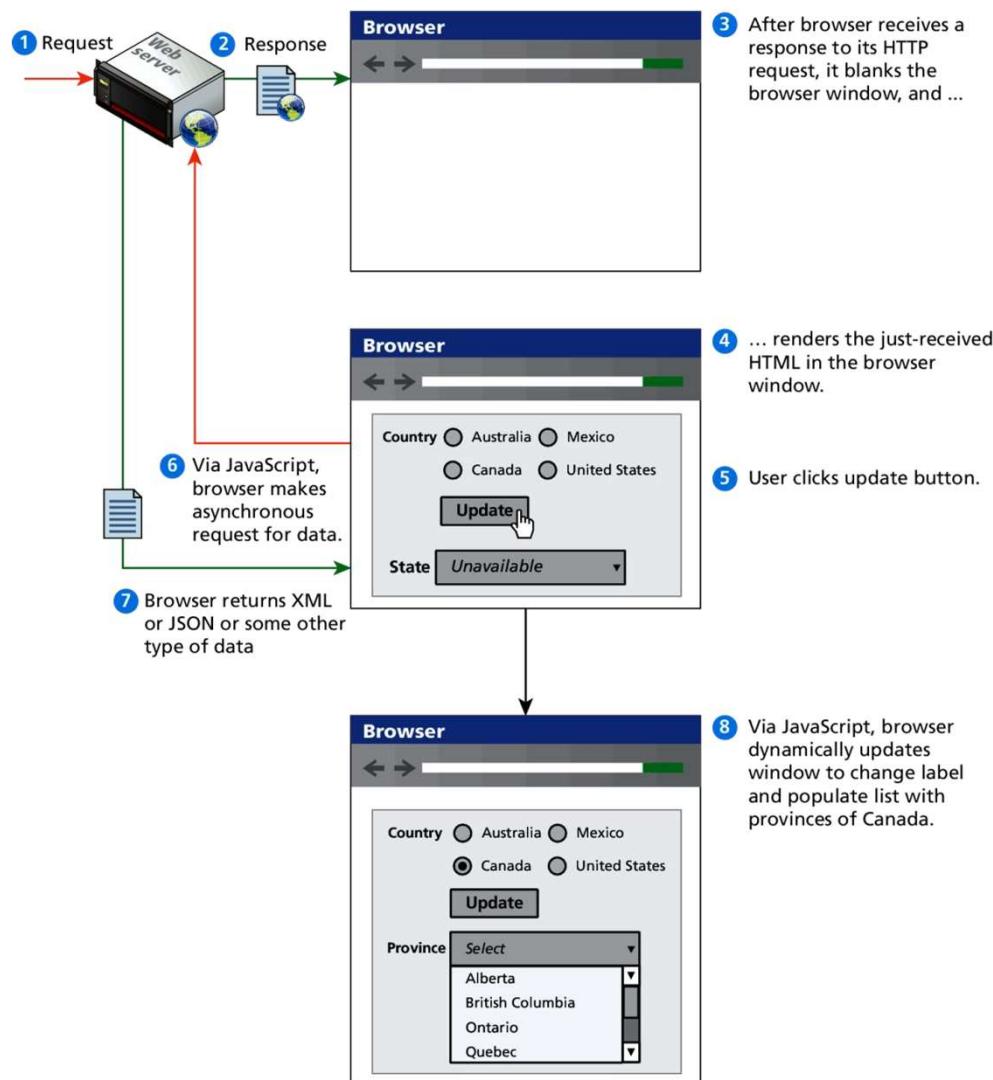
JavaScript became a much more important part of web development in the mid 2000s with **AJAX**.

AJAX is both an acronym as well as a general term.

- As an acronym it means **Asynchronous JavaScript And XML**.
- The most important feature of AJAX sites is the asynchronous data requests.

Asynchronous data requests

The better AJAX way



Frameworks

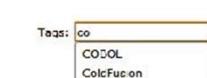
Lots of this is done for you, once you get the basics



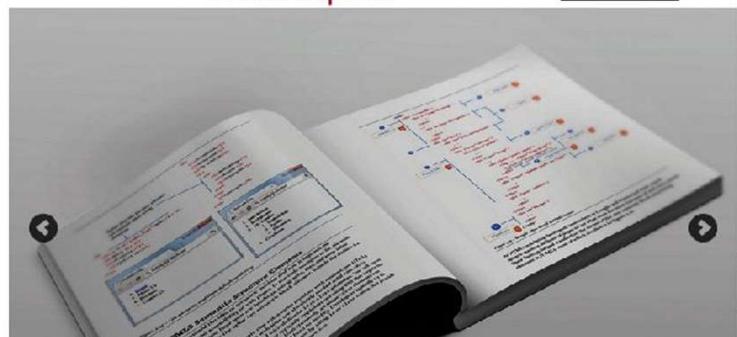
Date Picker

Sed non urna. Donec et ante. Phasellus ac ligula. Vestibulum sit amet purus. Vivamus hendrerit, dolor et aliquet lacruct, moris turpis porttitor velit, feugibus interdum tellus libero ac justo. vivamus non quam, in suscipit faucibus urna.

Accordion



AutoComplete



CONTENT PRESENTATION SUITABLE FOR VISUALLY ORIENTED LEARNERS

As long-time instructors, the authors are well aware that today's students are often extremely reluctant to read long blocks of text. Our approach is to prioritize diagrams over textual explanation. That is, we have tried to make the chapters visually appealing and to explain complicated ideas not only through text but also through visual aids.

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

— — — — —

Section 2 of 8

JAVASCRIPT DESIGN PRINCIPLES

Layers

They help organize

- When designing software to solve a problem, it is often helpful to abstract the solution a little bit to help build a cognitive model in your mind that you can then implement.
- Perhaps the most common way of articulating such a cognitive model is via the term **layer**.
- In object-oriented programming, a software **layer** is a way of conceptually grouping programming classes that have similar functionality and dependencies.

Layers

Common Layers

- **Presentation layer.**

Classes focused on the user interface.

- **Business layer.**

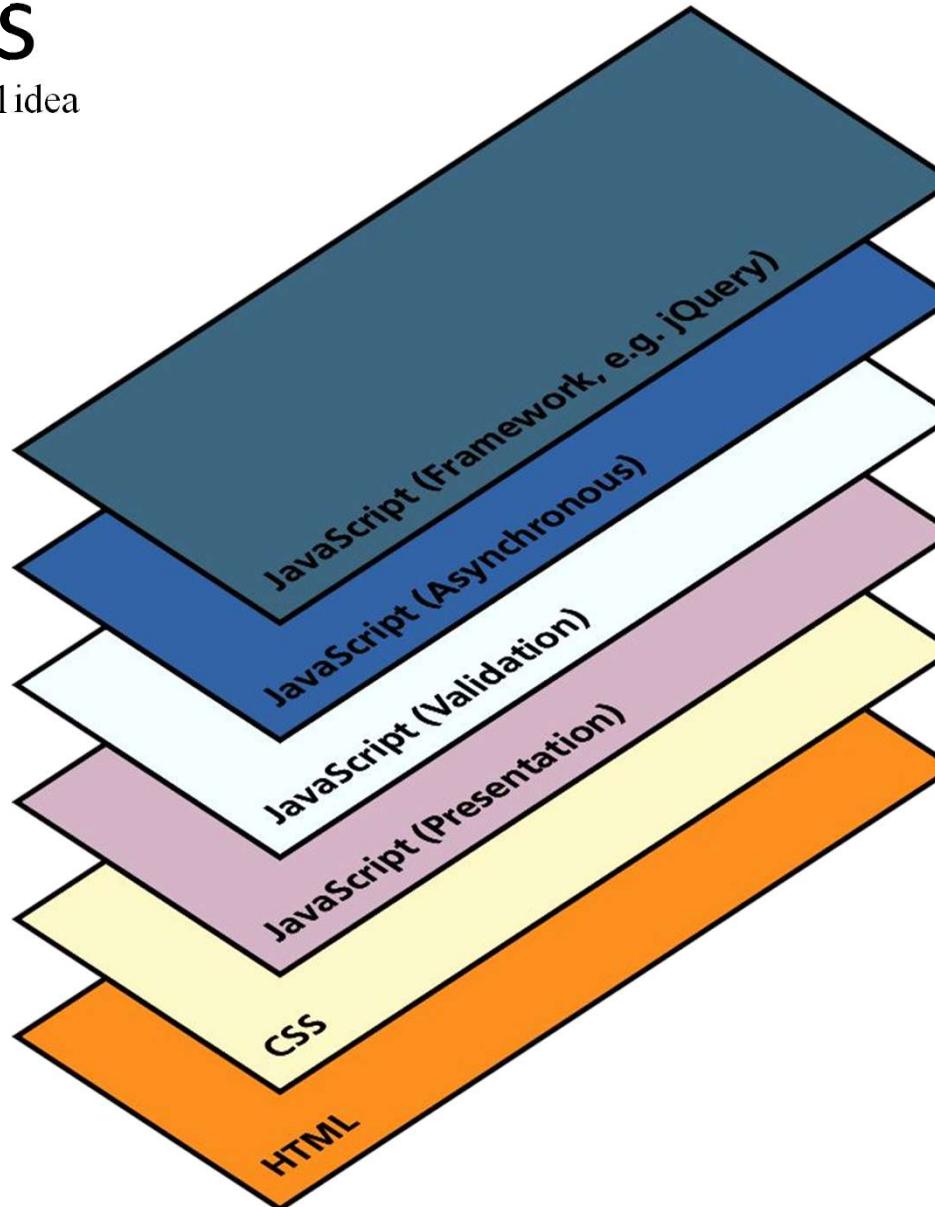
Classes that model real-world entities, such as customers, products, and sales.

- **Data layer.**

Classes that handle the interaction with the data sources.

Layers

Just a conceptual idea



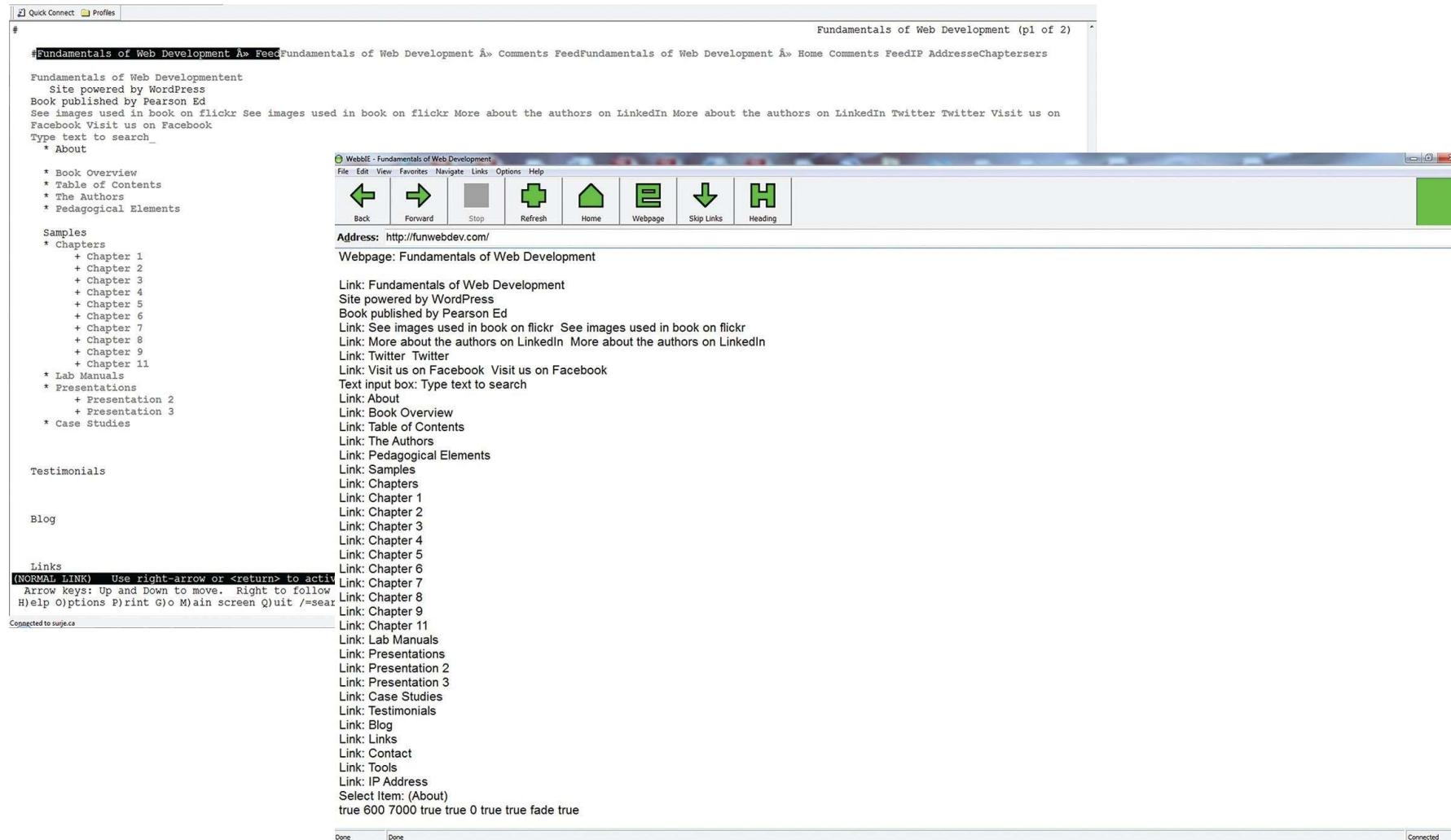
Users Without Javascript

They do exist

- **Web crawler.** A web crawler is a client running on behalf of a search engine to download your site, so that it can eventually be featured in their search results.
- **Browser plug-in.** A browser plug-in is a piece of software that works within the browser, that might interfere with JavaScript.
- **Text-based client.** Some clients are using a text-based browser.
- **Visually disabled client.** A visually disabled client will use special web browsing software to read the contents of a web page out loud to them.

Users Without Javascript

Lynx, and WebIE



Graceful Degradation and Progressive Enhancement

Over the years, browser support for different JavaScript objects has varied. Something that works in the current version of Chrome might not work in IE version 8; something that works in a desktop browser might not work in a mobile browser.

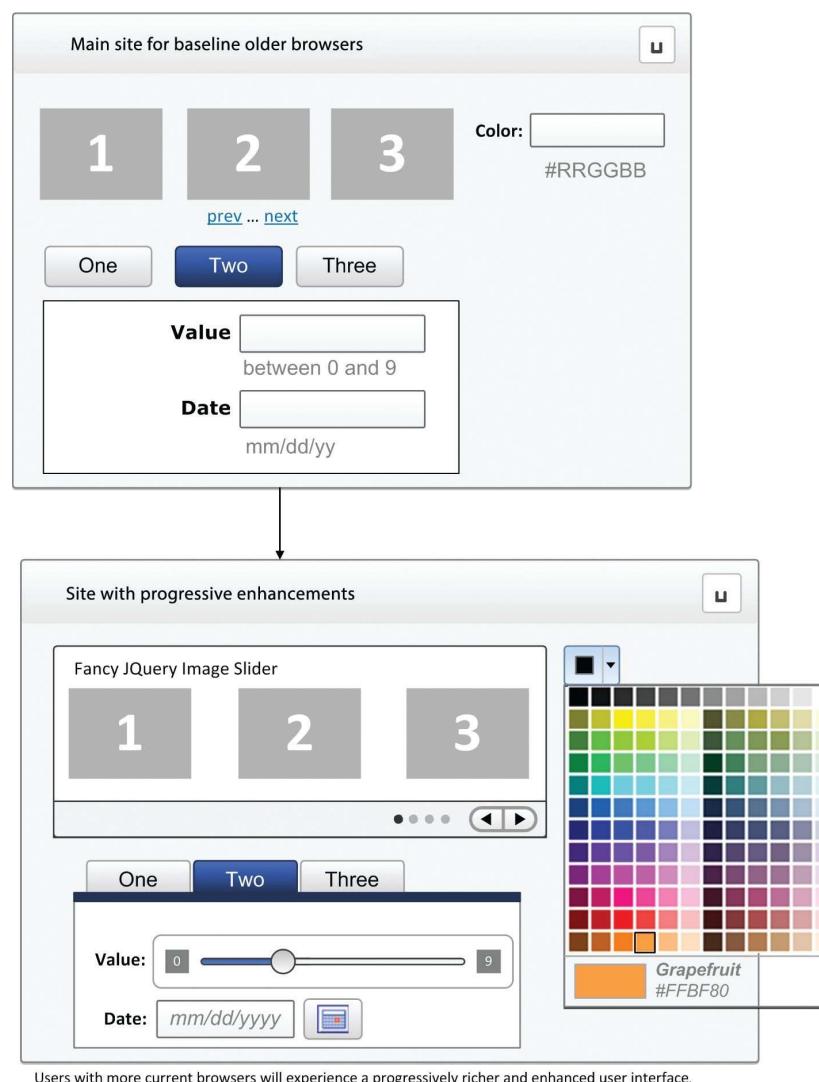
There are two strategies:

- **graceful degradation**
- **progressive enhancement**

Progressive Enhancement

- In this case, the developer creates the site using CSS, JavaScript, and HTML features that are supported by all browsers of a certain age or newer.
- To that baseline site, the developers can now “progressively” (i.e., for each browser) “enhance” (i.e., add functionality) to their site based on the capabilities of the users’ browsers.

Progressive Enhancement

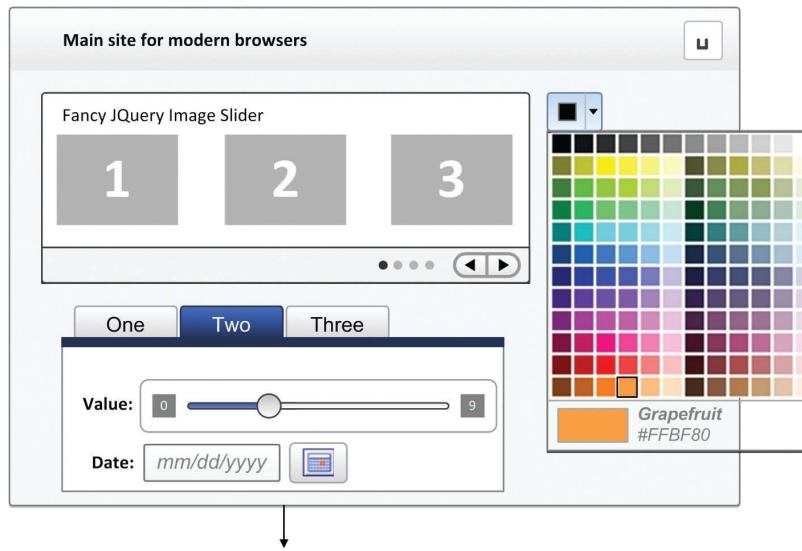


Graceful Degradation

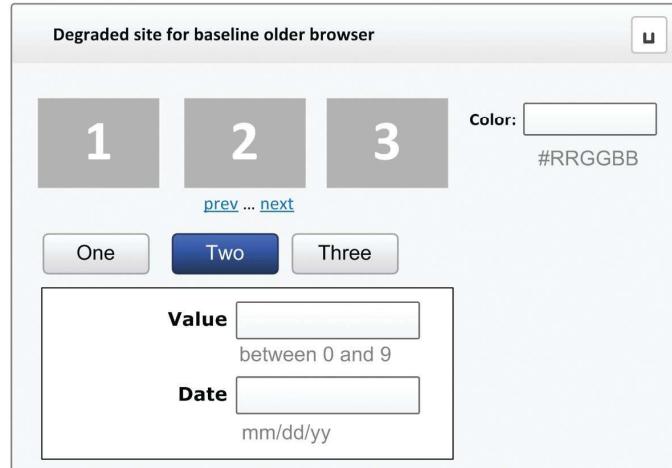
- ❖ With this strategy you develop your site for the abilities of current browsers.
- ❖ For those users who are not using current browsers, you might provide an alternate site or pages for those using older browsers that lack the JavaScript (or CSS or HTML5) used on the main site.
- ❖ The idea here is that the site is “degraded” (i.e., loses capability) “gracefully” (i.e., without pop-up JavaScript error codes or without condescending messages telling users to upgrade their browsers)

Graceful Degradation

The main site uses current JavaScript and HTML5 form elements.



The gracefully degraded alternate site for users who are not using the most current browsers.



Section 3 of 8

WHERE DOES JAVASCRIPT GO?

Where does JavaScript go?

JavaScript can be linked to an HTML page in a number of ways.

- Inline
- Embedded
- External

Inline JavaScript

Mash it in

Inline JavaScript refers to the practice of including JavaScript code directly within certain HTML attributes

Inline JavaScript is a real maintenance nightmare

```
<a href="JavaScript:OpenWindow();" more info</a>
<input type="button" onclick="alert('Are you sure?');" />
```

LISTING 6.1 Inline JavaScript example

Embedded JavaScript

Better

Embedded JavaScript refers to the practice of placing JavaScript code within a <script> element

```
<script type="text/javascript">
/* A JavaScript Comment */
alert ("Hello World!");
</script>
```

LISTING 6.2 Embedded JavaScript example

External JavaScript

Better

JavaScript supports this separation by allowing links to an external file that contains the JavaScript.

By convention, JavaScript external files have the extension .js.

```
<head>
  <script type="text/JavaScript" src="greeting.js">
    </script>
</head>
```

LISTING 6.3 External JavaScript example

Section 4 of 8

SYNTAX

JavaScript Syntax

We will briefly cover the fundamental syntax for the most common programming constructs including

- **variables**,
- **assignment**,
- **conditionals**,
- **loops**, and
- **arrays**

before moving on to advanced topics such as **events** and **classes**.

JavaScript's Reputation

Precedes it?

JavaScript's reputation for being quirky not only stems from its strange way of implementing object-oriented principles but also from some odd syntactic *gotchas*:

- Everything is type sensitive, including function, class, and variable names.
- The scope of variables in blocks is not supported. This means variables declared inside a loop may be accessible outside of the loop, counter to what one would expect.
- There is a `==` operator, which tests not only for equality but type equivalence.
- **Null** and **undefined** are two distinctly different states for a variable.
- Semicolons are not required, but are permitted (and encouraged).
- There is no integer type, only number, which means floating-point rounding errors are prevalent even with values intended to be integers.

Variables

`var`

- ❖ **Variables** in JavaScript are **dynamically typed**, meaning a variable can be an integer, and then later a string, then later an object, if so desired.
- ❖ This simplifies variable declarations, so that we do not require the familiar type fields like *int*, *char*, and *String*. Instead we use `var`
- ❖ **Assignment** can happen at declaration-time by appending the value to the declaration, or at run time with a simple right-to-left assignment

Variables

Assignment

var x; ← a variable **x** is defined

var y = 0; ← **y** is defined and initialized to 0

y = 4; ← **y** is assigned the value of 4

/* x conditional assignment */
x = (y==4) ? "y is 4" : "y is not 4";

Condition

Value
if true

Value
if false

Comparison Operators

True or not True

Operator	Description	Matches ($x=9$)
<code>==</code>	Equals	$(x==9)$ is true $(x=="9")$ is true
<code>===</code>	Exactly equals, including type	$(x==="9")$ is false $(x === 9)$ is true
<code>< , ></code>	Less than, Greater Than	$(x<5)$ is false
<code><= , >=</code>	Less than or equal, greater than or equal	$(x<=9)$ is true
<code>!=</code>	Not equal	$(4!=x)$ is true
<code>!==</code>	Not equal in either value or type	$(x!=="9")$ is true $(x!==9)$ is false

Logical Operators

The Boolean operators and, or, and not and their truth tables are listed in Table 6.2. Syntactically they are represented with `&&` (and), `||` (or), and `!` (not).

A	B	$A \&\& B$
T	T	T
T	F	F
F	T	F
F	F	F

AND Truth Table

A	B	$A \parallel B$
T	T	T
T	F	T
F	T	T
F	F	F

OR Truth Table

A	$\neg A$
T	F
F	T

NOT Truth Table

TABLE 6.2 AND, OR, and NOT Truth Tables

Conditionals

If, else if, ..., else

JavaScript's syntax is almost identical to that of PHP, Java, or C when it comes to conditional structures such as if and if else statements. In this syntax the condition to test is contained within () brackets with the body contained in { } blocks.

```
var hourOfDay;    // var to hold hour of day, set it later...
var greeting;    // var to hold the greeting message.
if (hourOfDay > 4 && hourOfDay < 12){
    // if statement with condition
    greeting = "Good Morning";
}
else if (hourOfDay >= 12 && hourOfDay < 20){
    // optional else if
    greeting = "Good Afternoon";
}
else{ // optional else branch
    greeting = "Good Evening";
}
```

LISTING 6.4 Conditional statement setting a variable based on the hour of the day

Loops

Round and round we go

- Like conditionals, loops use the () and { } blocks to define the condition and the body of the loop.
- You will encounter the **while** and **for** loops
- While loops normally initialize a **loop control variable** before the loop, use it in the condition, and modify it within the loop.

`var i=0; // initialise the Loop Control Variable`

`while(i < 10){ //test the loop control variable`

`i++; //increment the loop control variable`

`}`

For Loops

Counted loops

A **for loop** combines the common components of a loop: initialization, condition, and post-loop operation into one statement.

This statement begins with the **for** keyword and has the components placed between () brackets, semicolon (;) separated as shown

```
for (var i = 0; i < 10; i++) {  
    //do something with i  
}
```

Functions

Functions are the building block for modular code in JavaScript, and are even used to build **pseudo-classes**, which you will learn about later.

They are defined by using the reserved word **function** and then the function name and (optional) parameters.

Since JavaScript is dynamically typed, functions do not require a return type, nor do the parameters require type.

Functions

Example

Therefore a function to raise x to the yth power might be defined as:

```
function power(x,y){
    var pow=1;
    for (var i=0;i<y;i++){
        pow = pow*x;
    }
    return pow;
}
```

And called as

```
power(2,10);
```

Alert

Not really used anymore, console instead

- ✓ The alert() function makes the browser show a pop-up to the user, with whatever is passed being the message displayed. The following JavaScript code displays a simple hello world message in a pop-up:

```
alert ( "Good Morning" );
```

- ✓ Using alerts can get tedious fast. When using debugger tools in your browser you can write output to a log with:

```
console.log("Put Messages Here");
```

And then use the debugger to access those logs.

Errors using try and catch

- When the browser's JavaScript engine encounters an error, it will *throw* an **exception**. These exceptions interrupt the regular, sequential execution of the program and can stop the JavaScript engine altogether.
- However, you can optionally catch these errors preventing disruption of the program using the **try–catch block**

```
try {  
    nonexistantfunction("hello");  
}  
catch(err) {  
    alert("An exception was caught:" + err);  
}
```

LISTING 6.5 Try-catch statement

Throw your own

Exceptions that is.

- Although try-catch can be used exclusively to catch built-in JavaScript errors, it can also be used by your programs, to throw your own messages.
- The throw keyword stops normal sequential execution, just like the built-in exceptions

```
try {  
    var x = -1;  
    if (x<0)  
        throw "smallerthan0Error";  
}  
catch(err){  
    alert (err + "was thrown");  
}
```

LISTING 6.6 Throwing a user-defined exception

Tips

With Exceptions

- ❖ Try-catch and throw statements should be used for *abnormal* or *exceptional* cases in your program.
- ❖ Throwing an exception disrupts the sequential execution of a program. When the exception is thrown all subsequent code is not executed until the catch statement is reached.
- ❖ This reinforces why try-catch is for exceptional cases.

Section 5 of 8

JAVASCRIPT OBJECTS

JavaScript Objects

Objects not Classes

- ❖ JavaScript is not a full-fledged object-oriented programming language.
- ❖ It does not support many of the patterns you'd expect from an object-oriented language like inheritance and polymorphism.

The language does, however, support objects.

JavaScript Objects

Not full-fledged O.O.

- Objects can have **constructors**, **properties**, and **methods** associated with them.
- There are objects that are included in the JavaScript language; you can also define your own kind of objects.

Constructors

- Normally to create a new object we use the new keyword, the class name, and () brackets with n optional parameters inside, comma delimited as follows:

```
var someObject = new ObjectName(p1,p2,..., pn);
```

- For some classes, shortcut constructors are defined

```
var greeting = "Good Morning";
```

```
var greeting = new String("Good Morning");
```

Properties

Use the dot

- Each object might have properties that can be accessed, depending on its definition.
- When a property exists, it can be accessed using **dot notation** where a dot between the instance name and the property references that property.

```
//show someObject.property to the user  
alert(someObject.property);
```

Methods

Use the dot, with brackets

- Objects can also have methods, which are **functions** associated with an instance of an object. These methods are called using the same dot notation as for properties, but instead of accessing a variable, we are calling a method.

`someObject.doSomething();`

- Methods may produce different output depending on the object they are associated with because *they can utilize the internal properties of the object.*

Objects Included in JavaScript

A number of useful objects are included with JavaScript including:

- Array
- Boolean
- Date
- Math
- String
- Dom objects

Arrays

- ❑ Arrays are one of the most used data structures. In practice, this class is defined to behave more like a linked list in that it can be resized dynamically, but the implementation is browser specific, meaning the efficiency of insert and delete operations is unknown.
- ❑ The following code creates a new, empty array named greetings:

```
var greetings = new Array();
```

Arrays

Initialize with values

- ✓ To initialize the array with values, the variable declaration would look like the following:

```
var greetings = new Array("Good Morning", "Good Afternoon");
```

or, using the square bracket notation:

```
var greetings = ["Good Morning", "Good Afternoon"];
```

Arrays

Access and Traverse

- ✓ To access an element in the array you use the familiar square bracket notation from Java and C-style languages, with the index you wish to access inside the brackets.

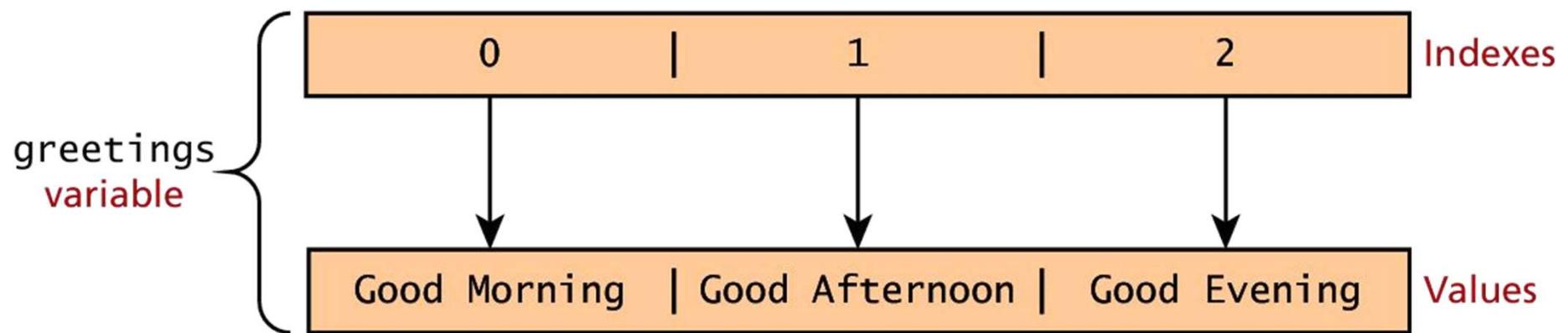
```
alert ( greetings[0] );
```

- ✓ One of the most common actions on an array is to traverse through the items sequentially. Using the Array object's **length** property to determine the maximum valid index. We have:

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

Arrays

Index and Value



Arrays

Modifying an array

- ✓ To add an item to an existing array, you can use the **push** method.

```
greetings.push("Good Evening");
```

- ✓ The **pop** method can be used to remove an item from the back of an array.
- ✓ Additional methods: concat(), slice(), join(), reverse(), shift(), and sort()

Math

- The **Math class** allows one to access common mathematic functions and common values quickly in one place.
- This static class contains methods such as max(), min(), pow(), sqrt(), and exp(), and trigonometric functions such as sin(), cos(), and arctan().
- Many mathematical constants are defined such as PI, E, SQRT2, and some others

Math.PI; // 3.141592657

Math.sqrt(4); // square root of 4 is 2.

Math.random(); // random number between 0 and 1

String

❖ The **String class** has already been used without us even knowing it.

❖ **Constructor usage**

```
var greet = new String("Good"); // long form constructor
```

```
var greet = "Good"; // shortcut constructor
```

❖ **Length of a string**

```
alert (greet.length); // will display "4"
```

String

Concatenation and so much more

```
var str = greet.concat("Morning"); // Long form concatenation
```

```
var str = greet + "Morning"; // + operator concatenation
```

Many other useful methods exist within the String class, such as

- accessing a single character using charAt()
- searching for one using indexOf().
- Strings allow splitting a string into an array, searching and matching with split(), search(), and match() methods.

Date

Not that kind

- The Date class is yet another helpful included object you should be aware of.
- It allows you to quickly calculate the current date or create date objects for particular dates.
- To display today's date as a string, we would simply create a new object and use the `toString()` method.

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

// This outputs Today is Mon Nov 12 2012 15:40:19 GMT-0700

```
alert ("Today is "+ d.toString());
```

Window

- ❖ The window object in JavaScript corresponds to the browser itself. Through it, you can access the current page's URL, the browser's history, and what's being displayed in the status bar, as well as opening new browser windows.
- ❖ In fact, the alert(), prompt(), confirm() function mentioned earlier is actually a method of the window object.

Section 6 of 8

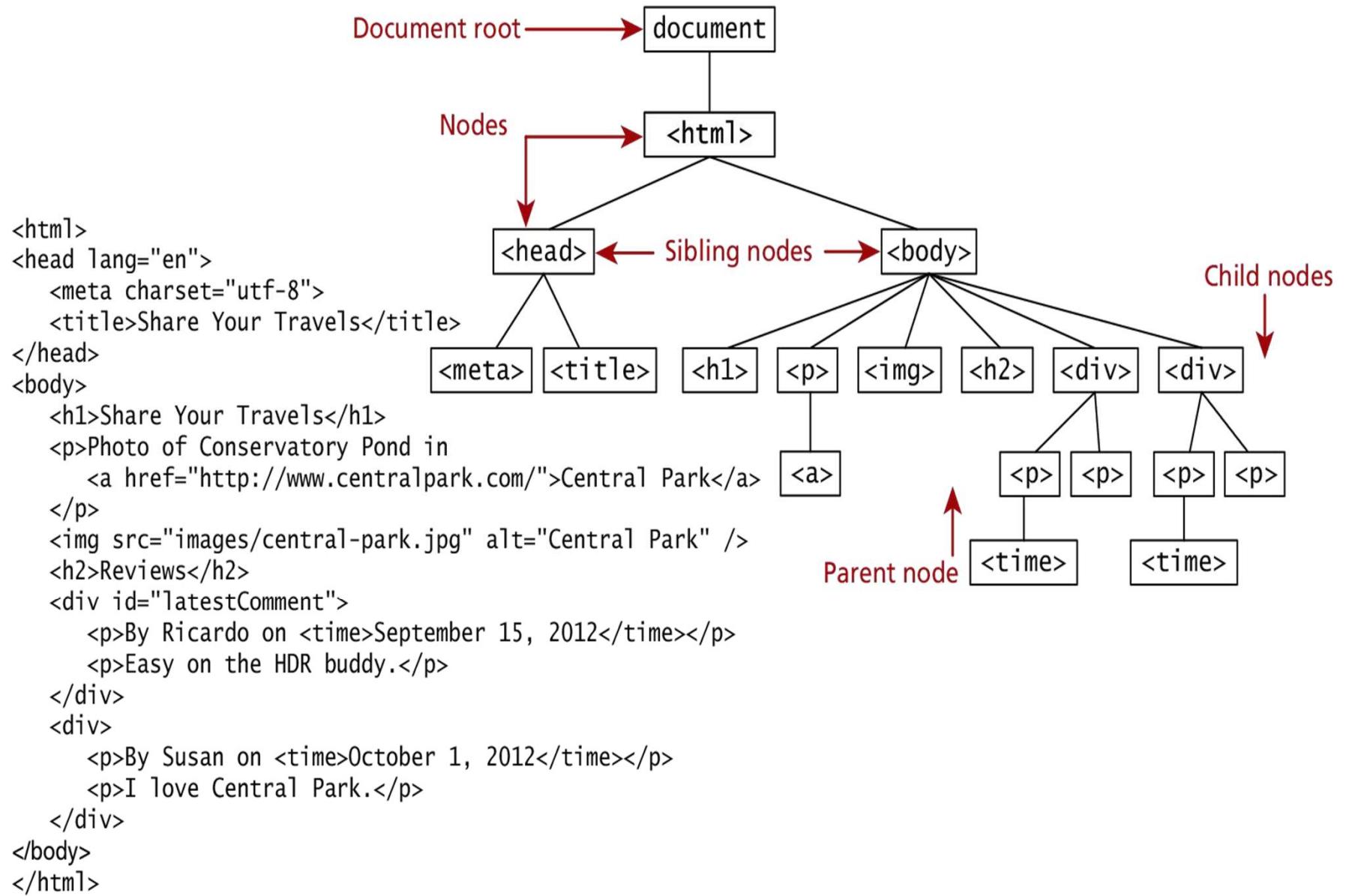
THE DOCUMENT OBJECT MODEL (DOM)

The DOM

Document Object Model

- JavaScript is almost always used to interact with the HTML document in which it is contained.
- This is accomplished through a programming interface (API) called the **Document Object Model**.

- According to the W3C, the DOM is a:
Platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.



DOM Nodes

- In the DOM, each element within the HTML document is called a **node**. If the DOM is a tree, then each node is an individual branch.

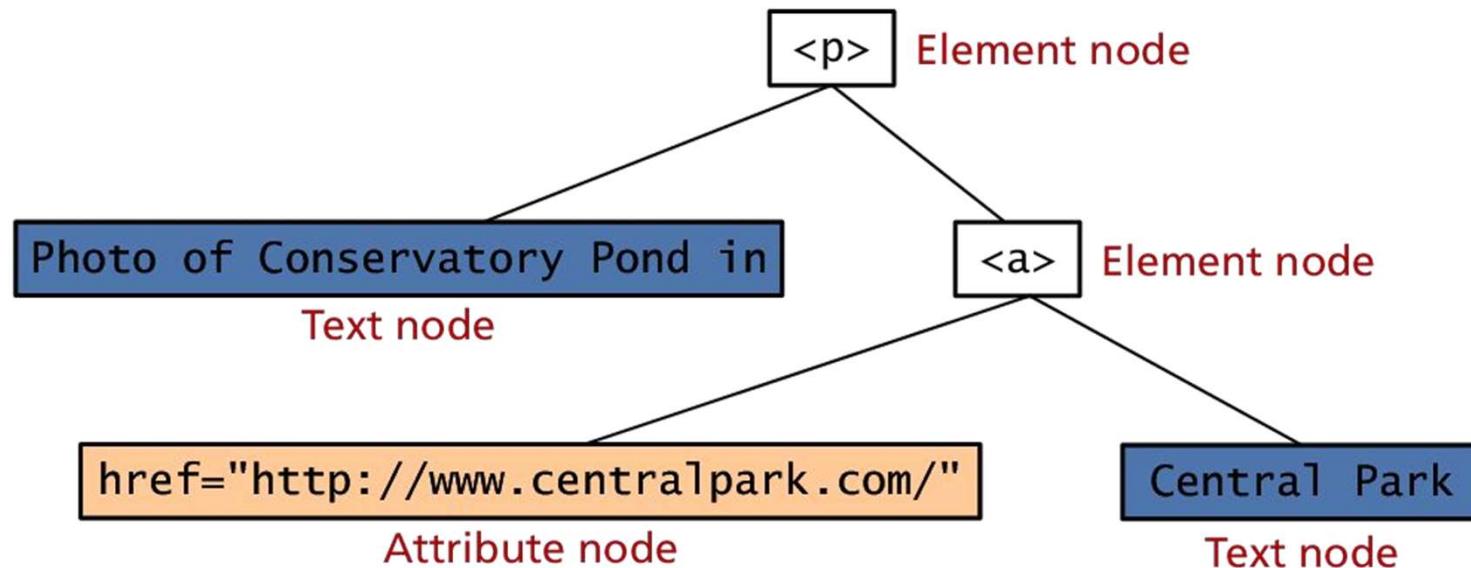
There are:

- element nodes,
 - text nodes, and
 - attribute nodes
- All nodes in the DOM share a common set of properties and methods.

DOM Nodes

Element, text and attribute nodes

```
<p>Photo of Conservatory Pond in  
  <a href="http://www.centralpark.com/">Central Park</a>  
</p>
```



DOM Nodes

Essential Node Object properties

Property	Description
<code>attributes</code>	Collection of node attributes
<code>childNodes</code>	A <code>NodeList</code> of child nodes for this node
<code>firstChild</code>	First child node of this node.
<code>lastChild</code>	Last child of this node.
<code>nextSibling</code>	Next sibling node for this node.
<code>nodeName</code>	Name of the node
<code>nodeType</code>	Type of the node
<code>nodeValue</code>	Value of the node
<code>parentNode</code>	Parent node for this node.
<code>previousSibling</code>	Previous sibling node for this node.

Document Object

One root to ground them all

- The **DOM document object** is the root JavaScript object representing the entire HTML document.
- It contains some properties and methods that we will use extensively in our development and is globally accessible as **document**.

// specify the doctype, for example html

```
var a = document.doctype.name;
```

// specify the page encoding, for example ISO-8859-1

```
var b = document.inputEncoding;
```

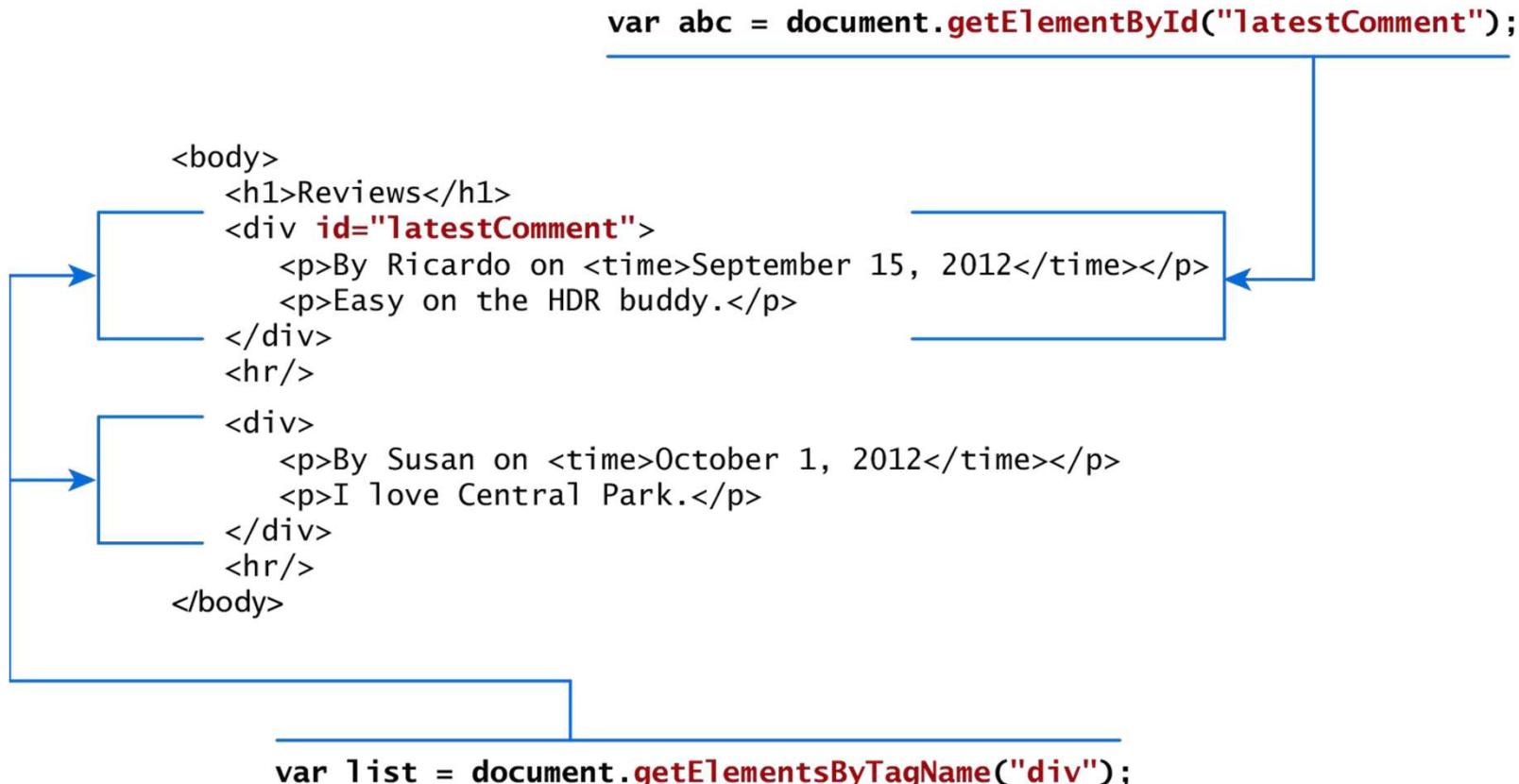
Document Object

Document Object Methods

Method	Description
<code>createAttribute()</code>	Creates an attribute node
<code>createElement()</code>	Creates an element node
<code>createTextNode()</code>	Create a text node
<code>getElementById(id)</code>	Returns the element node whose id attribute matches the passed id parameter.
<code>getElementsByName(name)</code>	Returns a nodeList of elements whose tag name matches the passed name parameter.

Accessing nodes

getElementById(), getElementsByTagName()



Element node Object

The type of object returned by the method `document.getElementById()` described in the previous section is an **element node** object.

This represents an HTML element in the hierarchy, contained between the opening `<>` and closing `</>` tags for this element.

- can itself contain more elements

Element node Object

Essential Element Node Properties

Property	Description
<code>className</code>	The current value for the class attribute of this HTML element.
<code>id</code>	The current value for the id of this element.
<code>innerHTML</code>	Represents all the things inside of the tags. This can be read or written to and is the primary way which we update particular div's using JS.
<code>style</code>	The style attribute of an element. We can read and modify this property.
<code>tagName</code>	The tag name for the element.

Modifying a DOM element

- ❖ The `document.write()` method is used to create output to the HTML page from JavaScript. The modern JavaScript programmer will want to write to the HTML page, but in a particular location, not always at the bottom
- ❖ Using the DOM document and HTML DOM element objects, we can do exactly that using the **innerHTML** property

```
var latest = document.getElementById("latestComment");
var oldMessage = latest.innerHTML;
latest.innerHTML = oldMessage + "<p>Updated this div with JS</p>";
```

LISTING 6.8 Changing the HTML using `innerHTML`

Modifying a DOM element

More verbosely, and validated

- Although the innerHTML technique works well (and is very fast), there is a more verbose technique available to us that builds output using the DOM.
- DOM functions `createTextNode()`, `removeChild()`, and `appendChild()` allow us to modify an element in a more rigorous way

```
var latest = document.getElementById("latestComment");
var oldMessage = latest.innerHTML;
var newMessage = oldMessage + "<p>Updated this div with JS</p>";
latest.removeChild(latest.firstChild);
latest.appendChild(document.createTextNode(newMessage));
```

LISTING 6.9 Changing the HTML using `createTextNode()` and `appendChild()`

Changing an element's style

- ✓ We can add or remove any style using the **style** or **className** property of the Element node.
- ✓ Its usage is shown below to change a node's background color and add a three-pixel border.

```
var commentTag = document.getElementById("specificTag");

commentTag.style.backgroundColor = "#FFFF00";

commentTag.style.borderWidth="3px";
```

Changing an element's style

With class

- The `className` property is normally a better choice, because it allows the styles to be created outside the code, and thus be better accessible to designers.

```
var commentTag = document.getElementById("specificTag");
```

```
commentTag.className = "someClassName";
```

- HTML5 introduces the `classList` element, which allows you to add, remove, or toggle a CSS class on an element.

```
label.classList.addClass("someClassName");
```

More Properties

Some Specific HTML DOM Element Properties for Certain Tag Types

Property	Description	Tags
href	The href attribute used in a tags to specify a URL to link to.	a
name	The name property is a bookmark to identify this tag. Unlike id which is available to all tags, name is limited to certain form related tags.	a, input, textarea, form
src	Links to an external URL that should be loaded into the page (as opposed to href which is a link to follow when clicked)	img, input, iframe, script
value	The value is related tot he value attribute of input tags. Often the value of an input field is user defined, and we use value to get that user input.	Input, textarea, submit

Section 7 of 8

JAVASCRIPT EVENTS

JavaScript Events

- A JavaScript **event** is an action that can be detected by JavaScript.
- We say then that an event is *triggered* and then it can be *caught* by JavaScript functions, which then do something in response.

JavaScript Events

A brave new world

- In the original JavaScript world, events could be specified right in the HTML markup with *hooks* to the JavaScript code (and still can).
- As more powerful frameworks were developed, and website design and best practices were refined, this original mechanism was supplanted by the **listener** approach.

JavaScript Events

Two approaches

Old, Inline technique

```
...  
  <script type="text/javascript" src="inline.js"></script>  
...  
  
<form name='mainForm' onsubmit="validate(this);">  
  <input name="name" type="text" onhover="hover(this); onfocus="focus(this);>  
  <input name="email" type="text" onhover="hover(this); onfocus="focus(this);>  
  <input type="submit" onclick="validate(this);>  
...  
...
```



New, Layered Listener technique

```
...  
  <script type="text/javascript" src="listener.js"></script>  
...  
  
<form name='mainForm'>  
  <input name="name" type="text">  
  <input name="email" type="text">  
  <input type="submit">  
...  
...
```



Inline Event Handler Approach

- ❖ For example, if you wanted an alert to pop-up when clicking a <div> you might program:

```
<div id="example1" onclick="alert('hello')>Click for pop-up</div>
```

- ❖ The problem with this type of programming is that the HTML markup and the corresponding JavaScript logic are woven together. It does not make use of layers;
- ❖ that is, it does not separate content from behavior.

Listener Approach

Two ways to set up listeners

```
var greetingBox = document.getElementById('example1');
greetingBox.onclick = alert('Good Morning');
```

LISTING 6.10 The "old" style of registering a listener.

```
var greetingBox = document.getElementById('example1');
greetingBox.addEventListener('click', alert('Good Morning'));
greetingBox.addEventListener('mouseout', alert('Goodbye'));

// IE 8
greetingBox.attachEvent('click', alert('Good Morning'));
```

LISTING 6.11 The "new" DOM2 approach to registering listeners.

Listener Approach

Using functions

- What if we wanted to do something more elaborate when an event is triggered? In such a case, the behavior would have to be encapsulated within a function, as shown in Listing 6.12.

```
function displayTheDate() {  
    var d = new Date();  
    alert ("You clicked this on " + d.toString());  
}  
var element = document.getElementById('example1');  
element.onclick = displayTheDate;  
  
// or using the other approach  
element.addEventListener('click',displayTheDate);
```

LISTING 6.12 Listening to an event with a function

Listener Approach

Anonymous functions

An alternative to that shown in Listing 6.12 is to use an anonymous function (that is, one without a name), as shown in Listing 6.13.

```
var element = document.getElementById('example1');
element.onclick = function() {
    var d = new Date();
    alert ("You clicked this on " + d.toString());
};
```

LISTING 6.13 Listening to an event with an anonymous function

Event Object

- ✓ No matter which type of event we encounter, they are all **DOM event objects** and the event handlers associated with them can access and manipulate them.
- ✓ Typically we see the events passed to the function handler as a parameter named *e*.

```
function someHandler(e) {  
    // e is the event that triggered this handler.  
}
```

Event Object

Several Options

- **Bubbles.** If an event's bubbles property is set to true then there must be an event handler in place to handle the event or it will bubble up to its parent and trigger an event handler there.
- **Cancelable.** The Cancelable property is also a Boolean value that indicates whether or not the event can be cancelled.
- **preventDefault.** A cancelable default action for an event can be stopped using the preventDefault() method in the next slide

Event Object

Prevent the default behaviour

```
function submitButtonClicked(e) {  
    if(e.cancelable){  
        e.preventDefault();  
    }  
}
```

LISTING 6.14 A sample event handler function that prevents the default event

Event Types

There are several classes of event, with several types of event within each class specified by the W3C:

- mouse events
- keyboard events
- form events
- frame events

Mouse events

Event	Description
onclick	The mouse was clicked on an element
ondblclick	The mouse was double clicked on an element
onmousedown	The mouse was pressed down over an element
onmouseup	The mouse was released over an element
onmouseover	The mouse was moved (not clicked) over an element
onmouseout	The mouse was moved off of an element
onmousemove	The mouse was moved while over an element

Keyboard events

Event	Description
onkeydown	The user is pressing a key (this happens first)
onkeypress	The user presses a key (this happens after onkeydown)
onkeyup	The user releases a key that was down (this happens last)

Keyboard events

Example

- <input type="text" id="keyExample">
- The input box above, for example, could be listened to and each key pressed echoed back to the user as an alert as shown in Listing 6.15.

```
document.getElementById("keyExample").onkeydown = function  
myFunction(e){  
    var keyPressed=e.keyCode;      //get the raw key code  
    var character=String.fromCharCode(keyPressed); //convert to string  
    alert("Key " + character + " was pressed");  
}
```

LISTING 6.15 Listener that hears and alerts keypresses

Form Events

Event	Description
onblur	A form element has lost focus (that is, control has moved to a different element, perhaps due to a click or Tab key press.
onchange	Some <input>, <textarea> or <select> field had their value change. This could mean the user typed something, or selected a new choice.
onfocus	Complementing the onblur event, this is triggered when an element gets focus (the user clicks in the field or tabs to it)
onreset	HTML forms have the ability to be reset. This event is triggered when that happens.
onselect	When the users selects some text. This is often used to try and prevent copy/paste.
onsubmit	When the form is submitted this event is triggered. We can do some pre-validation when the user submits the form in JavaScript before sending the data on to the server.

Form Events

Example

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

LISTING 6.16 Catching the onsubmit event and validating a password to not be blank

Frame Events

- **Frame events** are the events related to the browser frame that contains your web page.
- The most important event is the **onload** event, which tells us an object is loaded and therefore ready to work with. If the code attempts to set up a listener on this not-yet-loaded <div>, then an error will be triggered.

```
window.onload= function(){  
    //all JavaScript initialization here.  
}
```

Frame Events

Table of frame events

Event	Description
onabort	An object was stopped from loading
onerror	An object or image did not properly load
onload	When a document or object has been loaded
onresize	The document view was resized
onscroll	The document view was scrolled
onunload	The document has unloaded

Section 8 of 8

FORMS

Validating Forms

You mean pre-validating right?

- Writing code to prevalidate forms on the client side will reduce the number of incorrect submissions, thereby reducing server load.
- There are a number of common validation activities including email validation, number validation, and data validation.

Validating Forms

Empty field

```
document.getElementById("loginForm").onsubmit = function(e){  
    var fieldValue=document.getElementById("username").value;  
    if(fieldValue==null || fieldValue==""){  
        // the field was empty. Stop form submission  
        e.preventDefault();  
        // Now tell the user something went wrong  
        alert("you must enter a username");  
    }  
}
```

LISTING 6.18 A simple validation script to check for empty fields

Validating Forms

Empty field

- ❖ If you want to ensure a checkbox is ticked, use code like that below.

```
var inputField=document.getElementById("license");
```

```
if (inputField.type=="checkbox")
```

```
{
```

```
if (inputField.checked)
```

//Now we know the box is checked

```
}
```

Validating Forms

Number Validation

```
function isNumeric(n) {  
    return !isNaN(parseFloat(n)) && isFinite(n);  
}
```

LISTING 6.19 A function to test for a numeric value

Submitting Forms

- ❖ Submitting a form using JavaScript requires having a node variable for the form element. Once the variable, say, `formExample` is acquired, one can simply call the `submit()` method:

```
var formExample = document.getElementById("loginForm");  
formExample.submit();
```

- ✓ This is often done in conjunction with calling `preventDefault()` on the `onsubmit` event.

Introduction to Server-Side Development with PHP

Chapter 8

Section 1 of 5

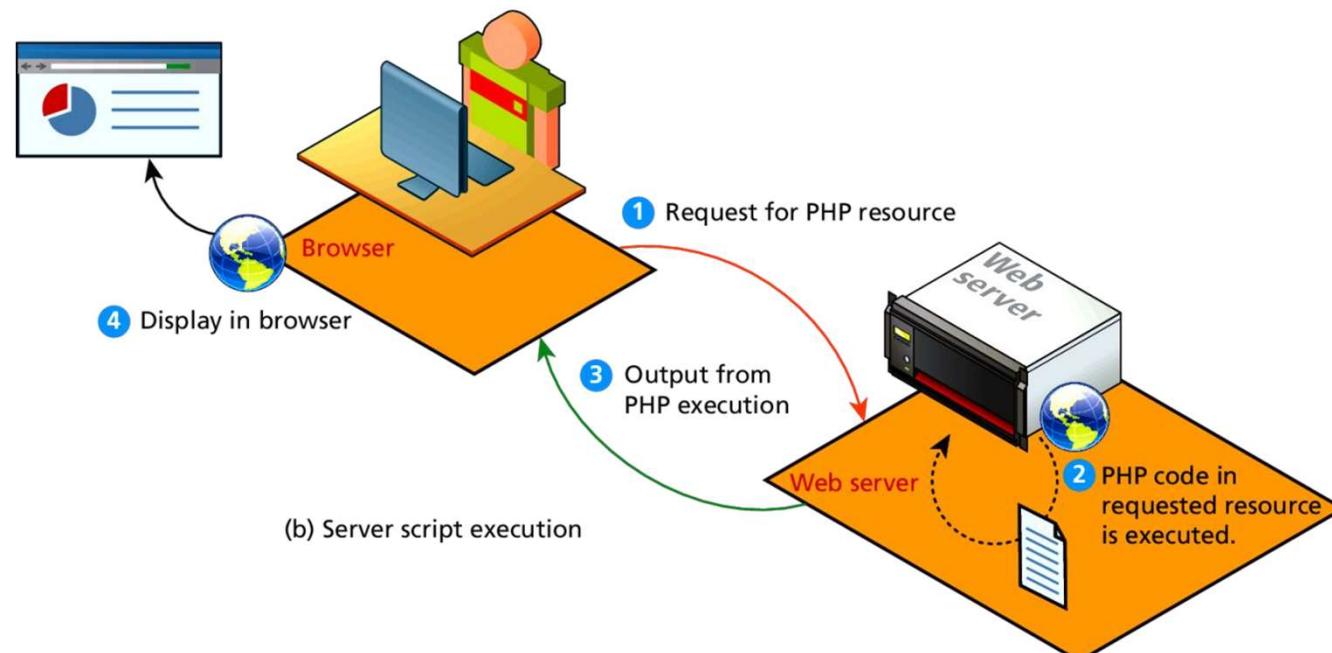
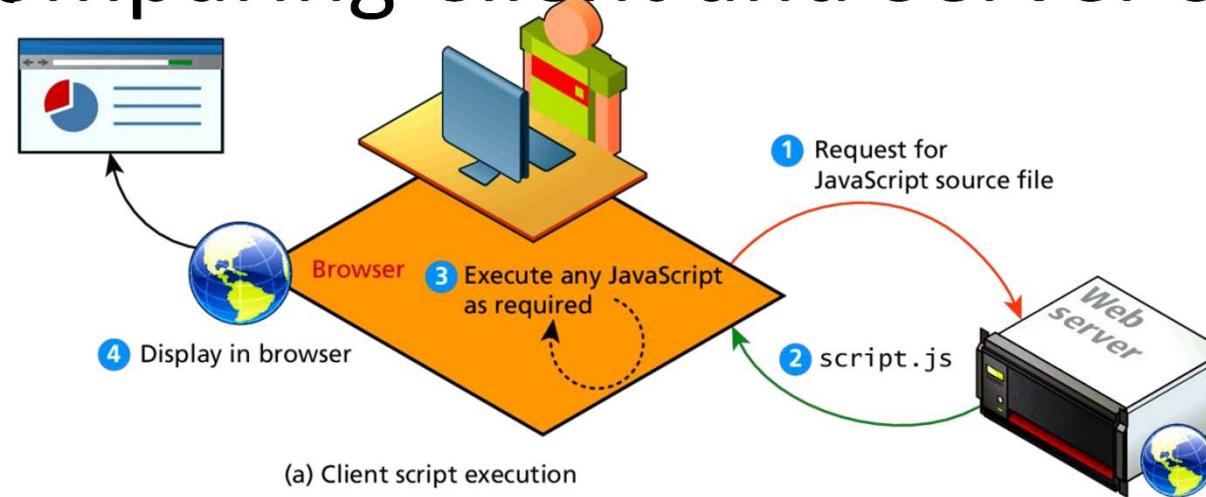
WHAT IS SERVER-SIDE DEVELOPMENT

What is Server-Side Development

- The basic hosting of your files is achieved through a web server.
- Server-side development is much more than web hosting: it involves the use of a programming technology like PHP or ASP.NET to create scripts that dynamically generate content

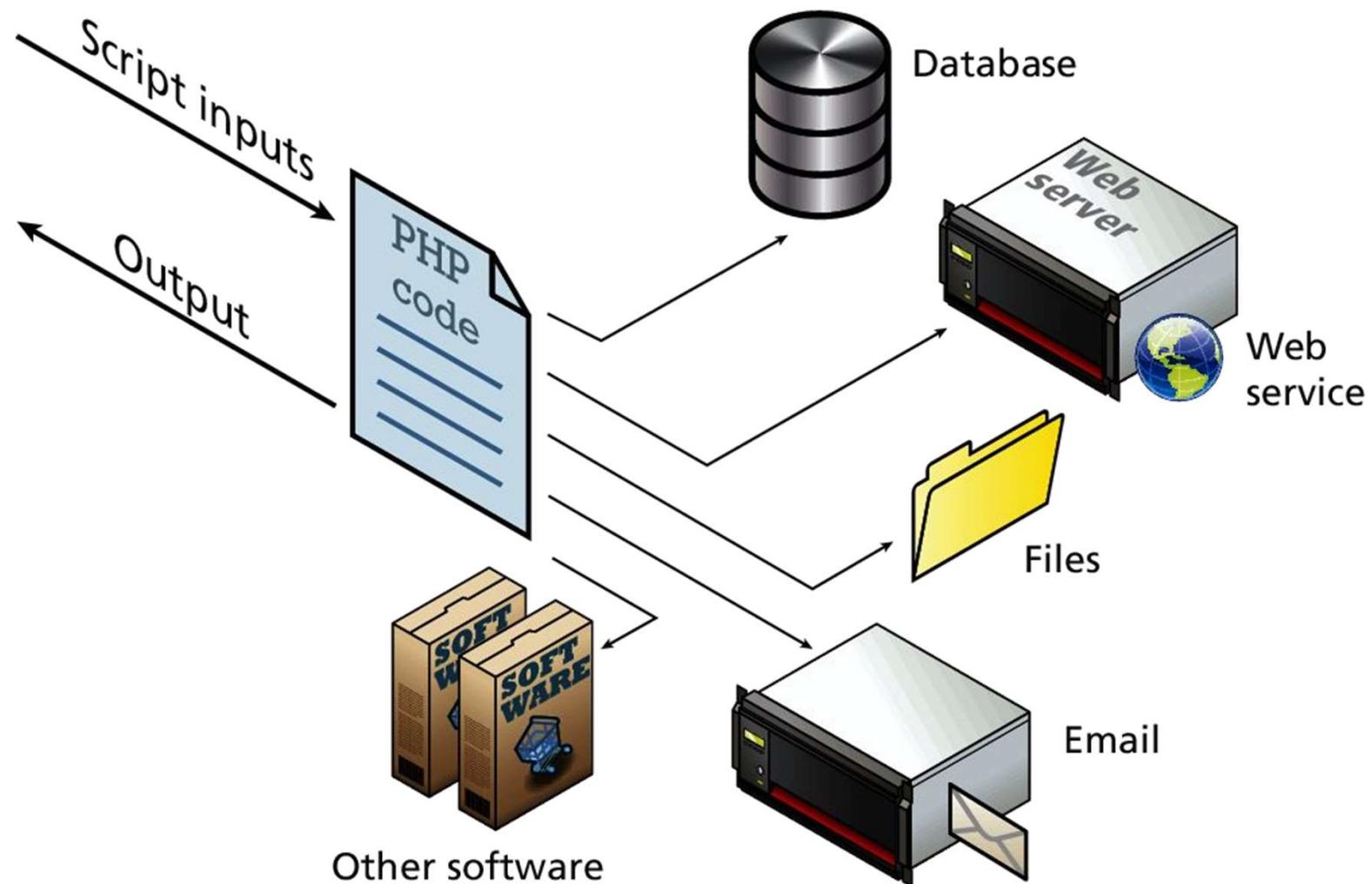
Consider distinction between client side and server side...

Comparing Client and Server Scripts

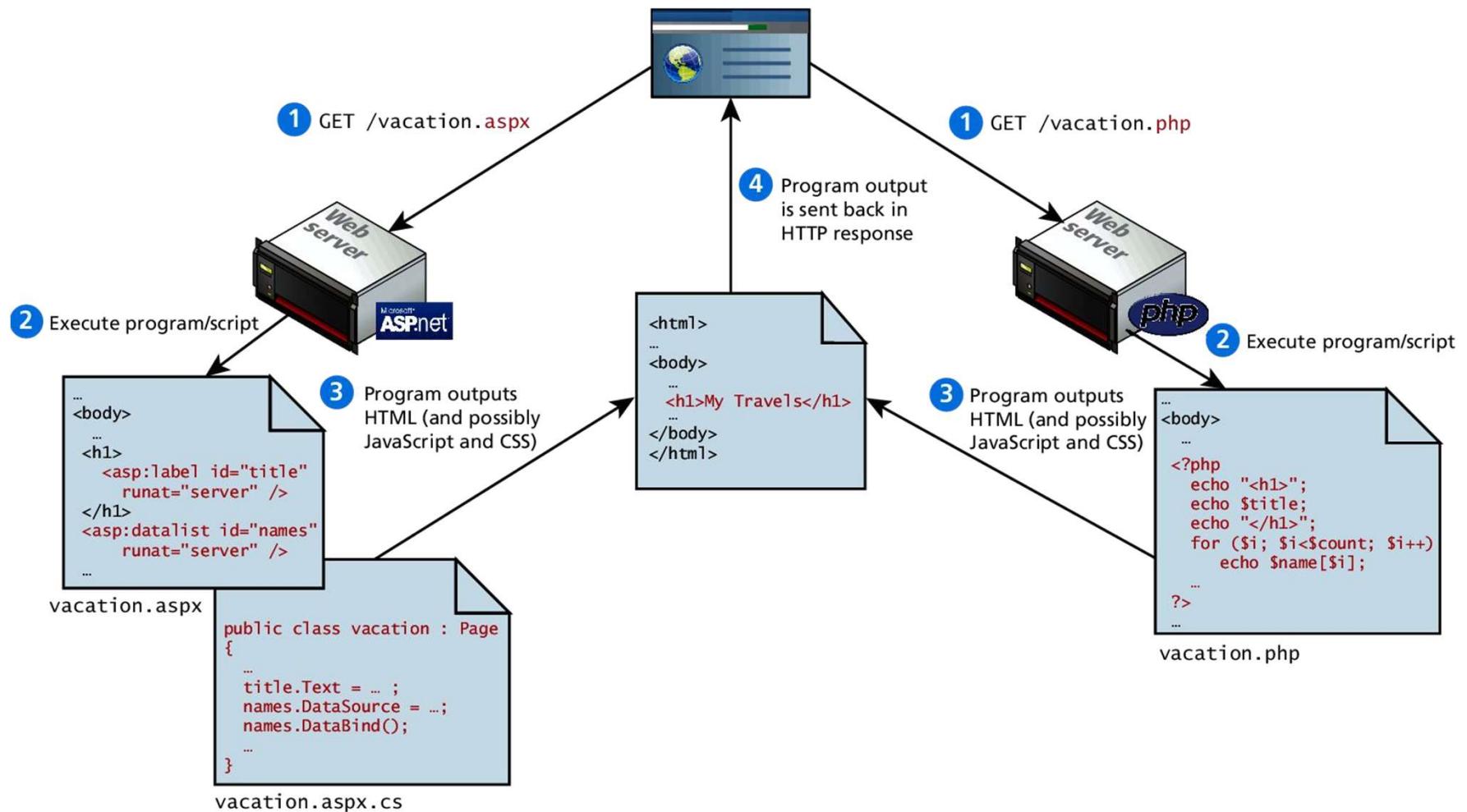


Server-Side Script Resources

So many tools in your kit



Web Development Technologies



Comparing Server-Side Technologies

- **ASP (Active Server Pages).** Like PHP, ASP code (using the VBScript programming language) can be embedded within the HTML. ASP programming code is interpreted at run time, hence it can be slow in comparison to other technologies.
- **ASP.NET.** ASP.NET is part of Microsoft's .NET Framework and can use any .NET programming language (though C# is the most commonly used). ASP.NET uses an explicitly object-oriented approach. It also uses special markup called web server controls that encapsulate common web functionality such as database-driven lists, form validation, and user registration wizards. ASP.NET pages are compiled into an intermediary file format called MSIL that is analogous to Java's byte-code. ASP.NET then uses a Just-In-Time compiler to compile the MSIL into machine executable code so its performance can be excellent. However, ASP.NET is essentially limited to Windows servers.

Comparing Server-Side Technologies

- **JSP (Java Server Pages).** JSP uses Java as its programming language and like ASP.NET it uses an explicit object-oriented approach and is used in large enterprise web systems and is integrated into the J2EE environment. Since JSP uses the Java Runtime Engine, it also uses a JIT compiler for fast execution time and is cross-platform. While JSP's usage in the web as a whole is small, it has a substantial market share in the intranet environment, as well as with very large and busy sites.
- **Node.js.** This is a more recent server environment that uses JavaScript on the server side, thus allowing developers already familiar with JavaScript to use just a single language for both client-side and server-side development. Unlike the other development technologies listed here, node.js also is its own web server software, thus eliminating the need for Apache, IIS, or some other web server software.

Comparing Server-Side Technologies

- **Perl.** Until the development and popularization of ASP, PHP, and JSP, Perl was the language typically used for early server-side web development. As a language, it excels in the manipulation of text. It was commonly used in conjunction with the **Common Gateway Interface (CGI)**, an early standard API for communication between applications and web server software.
- **PHP.** Like ASP, PHP is a dynamically typed language that can be embedded directly within the HTML, though it now supports most common object-oriented features, such as classes and inheritance. By default, PHP pages are compiled into an intermediary representation called **opcodes** that are analogous to Java's byte-code or the .NET Framework's MSIL. Originally, PHP stood for *personal home pages*, although it now is a recursive acronym that means *PHP: Hypertext Processor*.

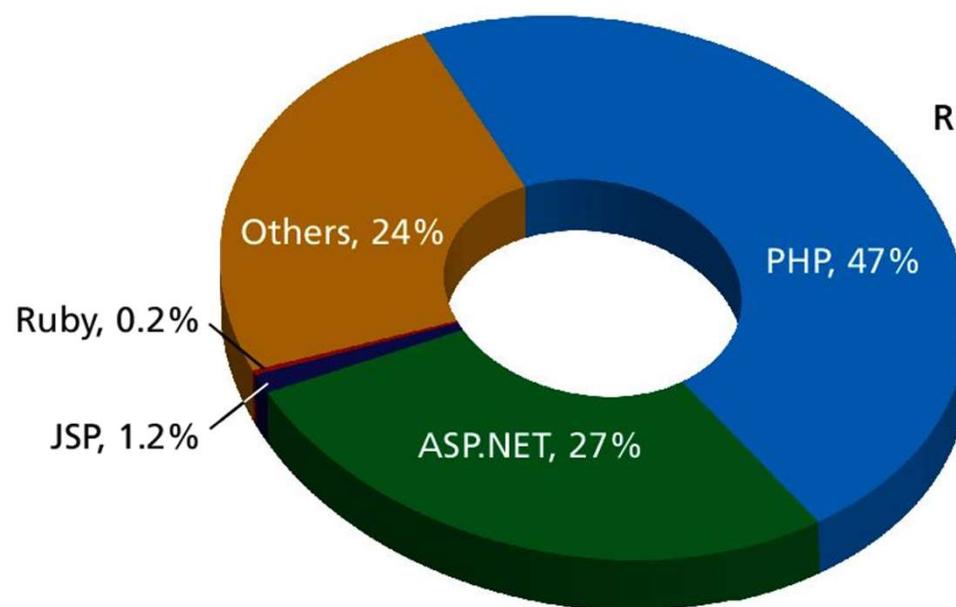
Comparing Server-Side Technologies

- **Python.** This terse, object-oriented programming language has many uses, including being used to create web applications. It is also used in a variety of web development frameworks such as Django and Pyramid.
- **Ruby on Rails.** This is a web development framework that uses the Ruby programming language. Like ASP.NET and JSP, Ruby on Rails emphasizes the use of common software development approaches, in particular the MVC design pattern. It integrates features such as templates and engines that aim to reduce the amount of development work required in the creation of a new site.

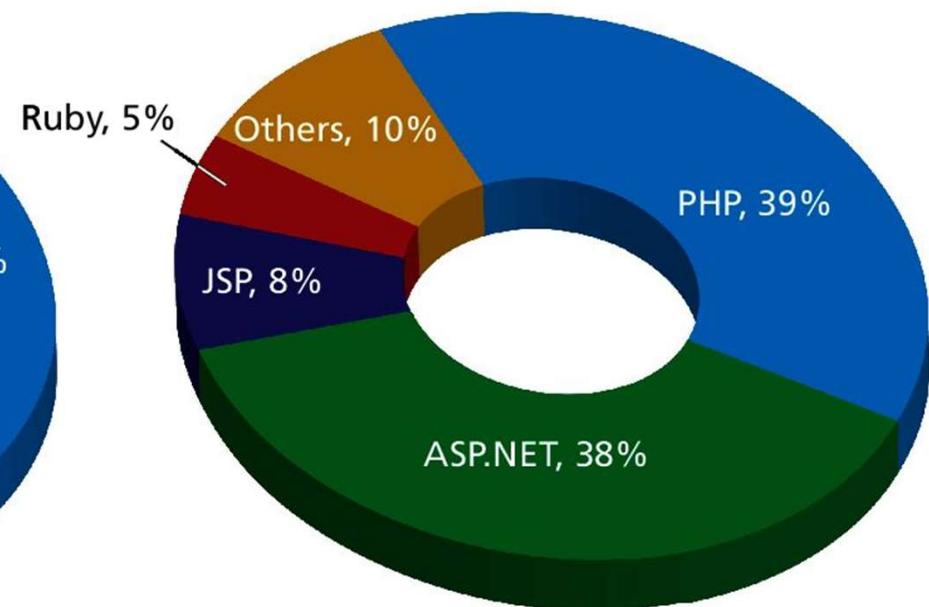
Market Share

Of web development environments

Top 50 Million Sites



Top 10,000 Sites



Section 2 of 5

WEB SERVER'S RESPONSABILITIES

A Web Server's Responsibilities

A web server has many responsibilities:

- handling HTTP connections
- responding to requests for static and dynamic resources
- managing permissions and access for certain resources
- encrypting and compressing data
- managing multiple domains and URLs
- managing database connections
- managing cookies and state
- uploading and managing files

LAMP stack

WAMP, MAMP, ...

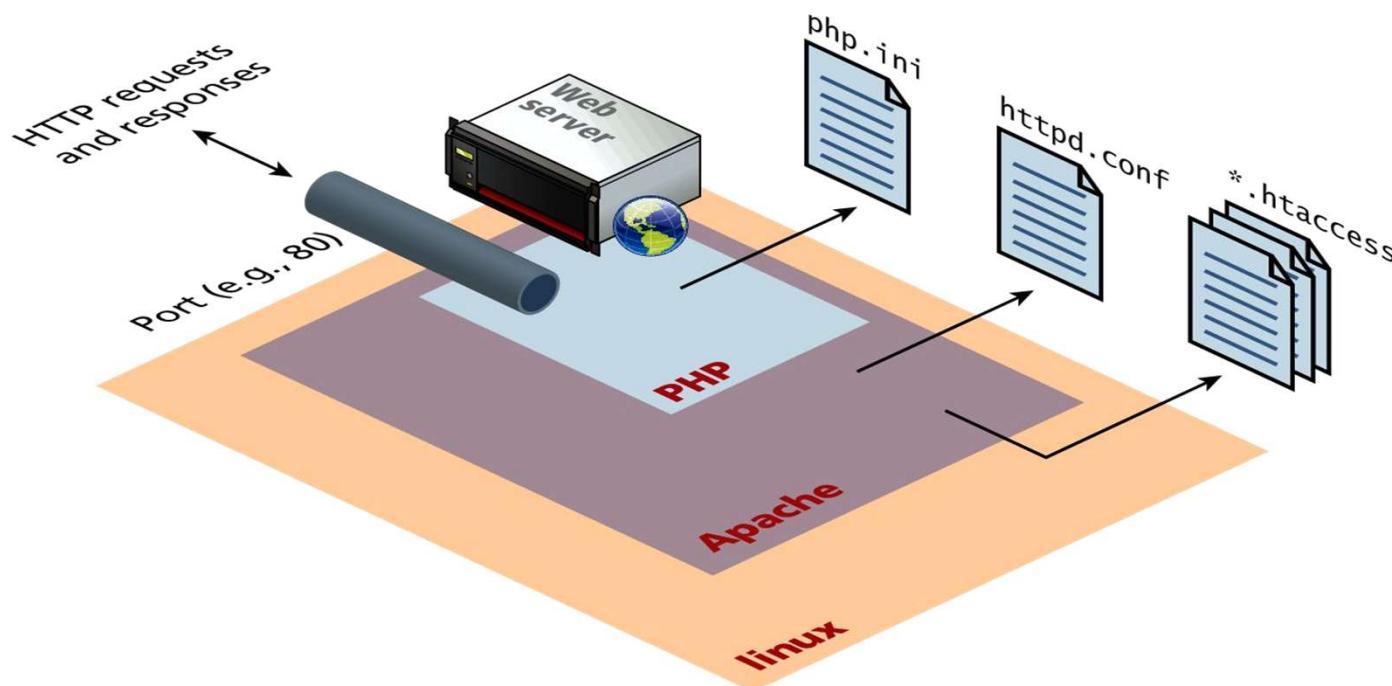
You will be using the LAMP software stack

- Linux operating system
- Apache web server
- MySQL DBMS
- PHP scripting language

Apache and Linux

LA

Consider the **Apache** web server as the intermediary that interprets HTTP requests that arrive through a network port and decides how to handle the request, which often requires working in conjunction with PHP.



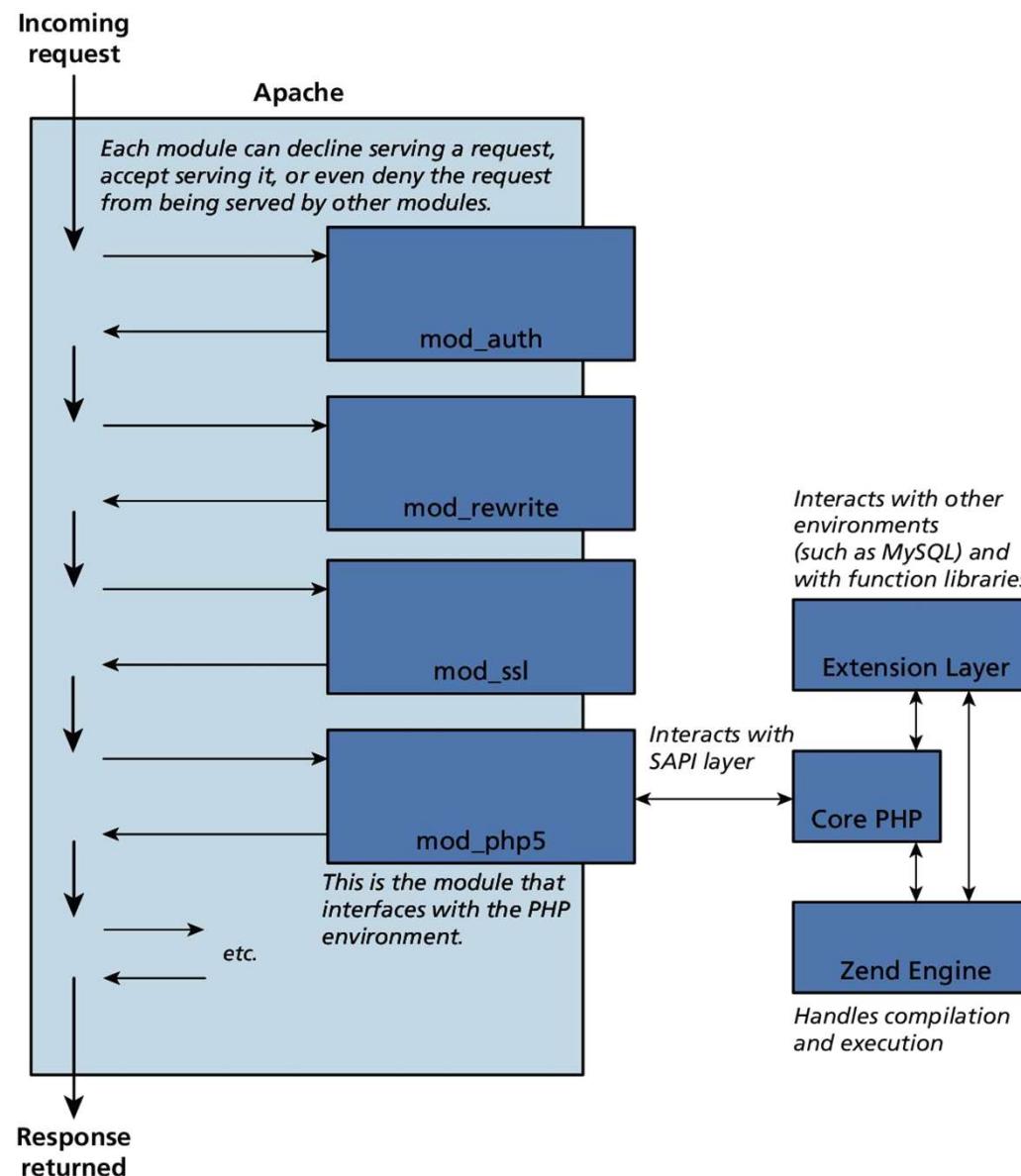
Apache

Continued

- Apache runs as a daemon on the server. A **daemon** is an executing instance of a program (also called a **process**) that runs in the background, waiting for a specific event that will activate it.
- When a request arrives, Apache then uses modules to determine how to respond to the request.
- In Apache, a **module** is a compiled extension (usually written in the C programming language) to Apache that helps it *handle* requests. For this reason, these modules are also sometimes referred to as **handlers**.

Apache and PHP

PHP Module in Apache



Apache Threads

Multi-thread and multi-process

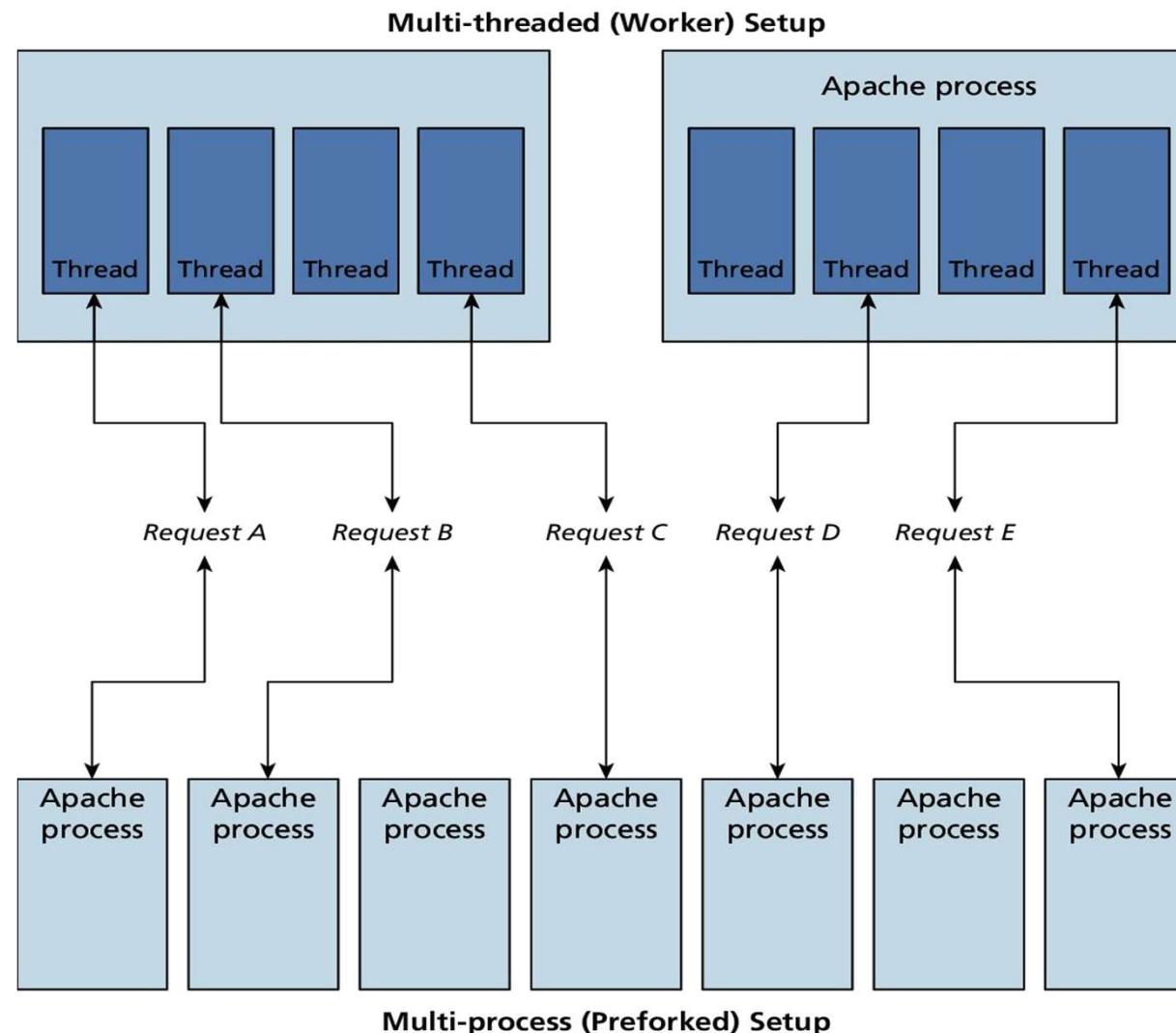
Apache runs in two possible modes:

- **multi-process** (also called **preforked**)
- **multi-threaded** (also called **worker**)

The default installation of Apache runs using the multi-process mode.

Apache Threads

Multi-thread and multi-process



PHP Internals

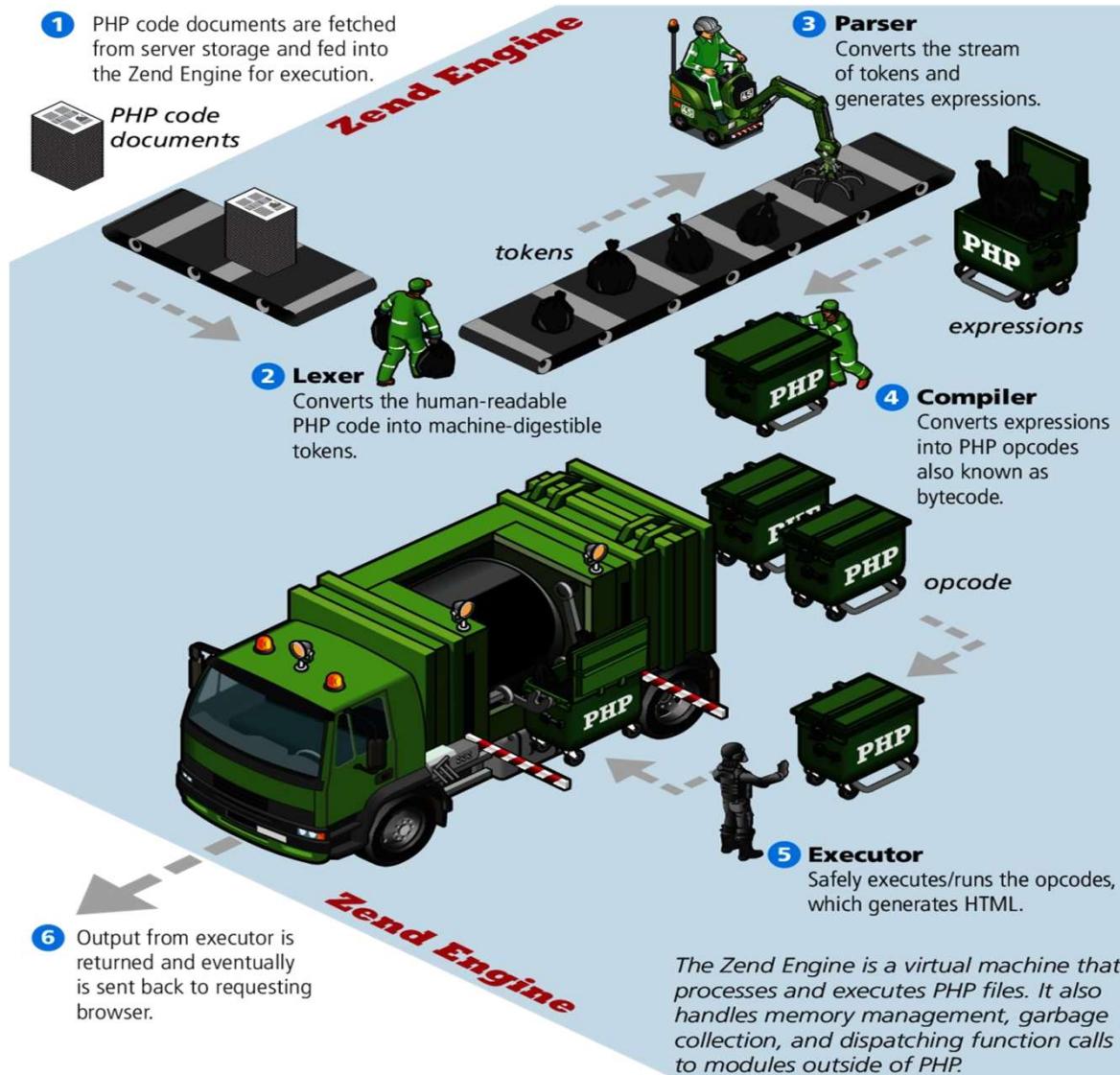
PHP itself is written in C

There are 3 main modules

1. **PHP core.** The Core module defines the main features of the PHP environment, including essential functions for variable handling, arrays, strings, classes, math, and other core features.
2. **Extension layer.** This module defines functions for interacting with services outside of PHP. This includes libraries for MySQL, FTP, SOAP web services, and XML processing, among others.
3. **Zend Engine.** This module handles the reading in of a requested PHP file, compiling it, and executing it.

Zend Engine

No, your code is not garbage.



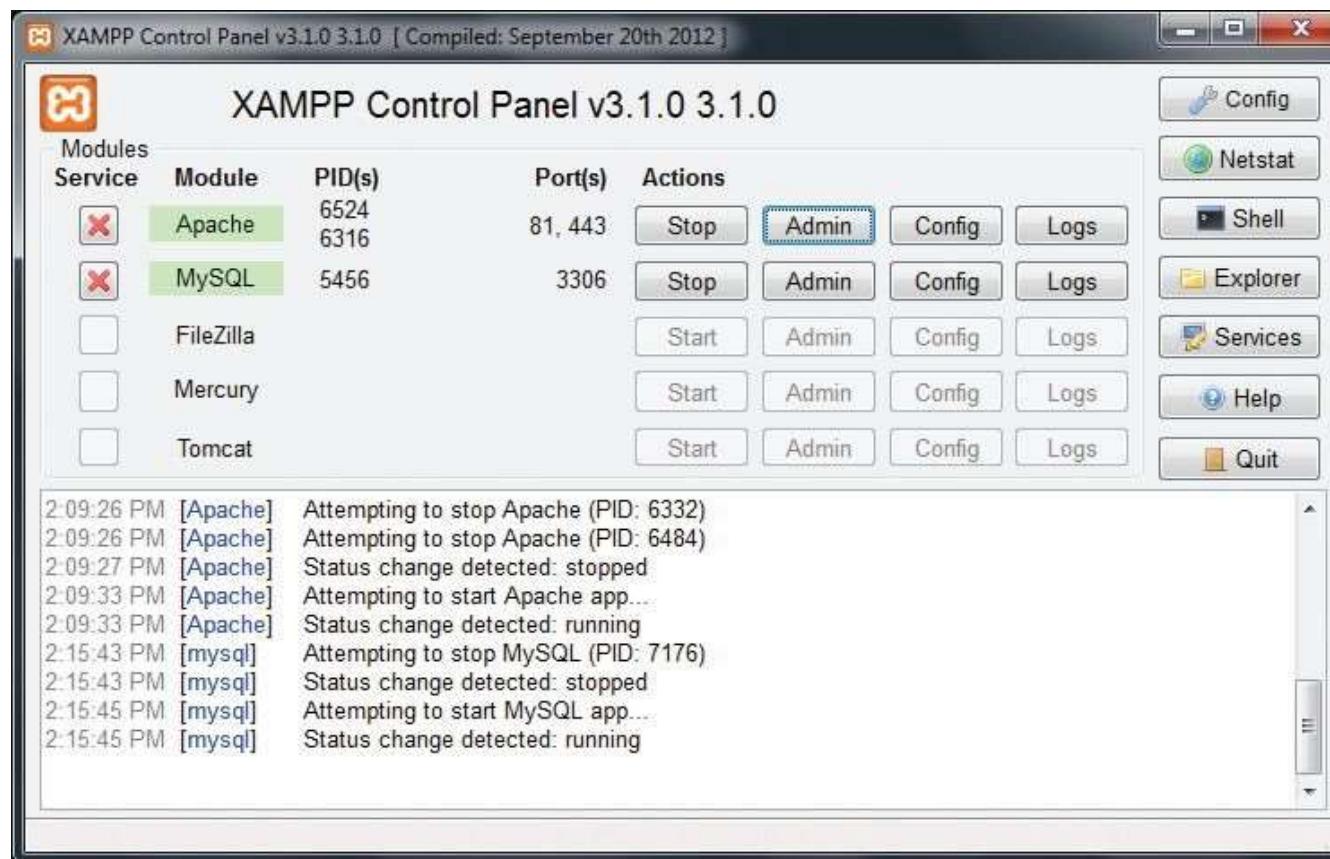
Installing LAMP locally

Turn this key

- ❖ The easiest and quickest way to do so is to use the
 - **XAMPP** For [Windows installation package](#)
 - **MAMP** for [Mac installation package](#)
- ❖ Both of these installation packages install and configure Apache, PHP, and MySQL.
- ❖ Later we can come back and configure these systems in more detail.

XAMPP Control Panel

Turn this key



XAMPP Settings

Defaults are

- PHP requests in your browser will need to use the **localhost** domain (127.0.0.1)
- PHP files will have to be saved somewhere within the **C:\xampp\htdocs** folder



Section 3 of 5

QUICK TOUR OF PHP

Quick Tour

- PHP, like JavaScript, is a dynamically typed language.
- it uses classes and functions in a way consistent with other object-oriented languages such as C++, C#, and Java
- The syntax for loops, conditionals, and assignment is identical to JavaScript
- Differs when you get to functions, classes, and in how you define variables

PHP Tags

The most important fact about PHP is that the programming code can be embedded directly within an HTML file.

- A PHP file will usually have the extension **.php**
- programming code must be contained within an opening **<?php** tag and a matching closing **?>** tag
- any code outside the tags is echoed directly out to the client

PHP Tags

```
<?php
$user = "Randy";
?>
<!DOCTYPE html>
<html>
<body>
<h1>Welcome <?php echo $user; ?></h1>
<p>
The server time is
<?php
echo "<strong>";
echo date("H:i:s");
echo "</strong>";
?>
</p>
</body>
</html>
```

LISTING 8.1 PHP tags

```
<!DOCTYPE html>
<html>
<body>
<h1>Welcome Randy</h1>
<p>
The server time is <strong>02:59:09</strong>
</p>
</body>
</html>
```

LISTING 8.2 Listing 8.1 in the browser

HTML and PHP

Two approaches

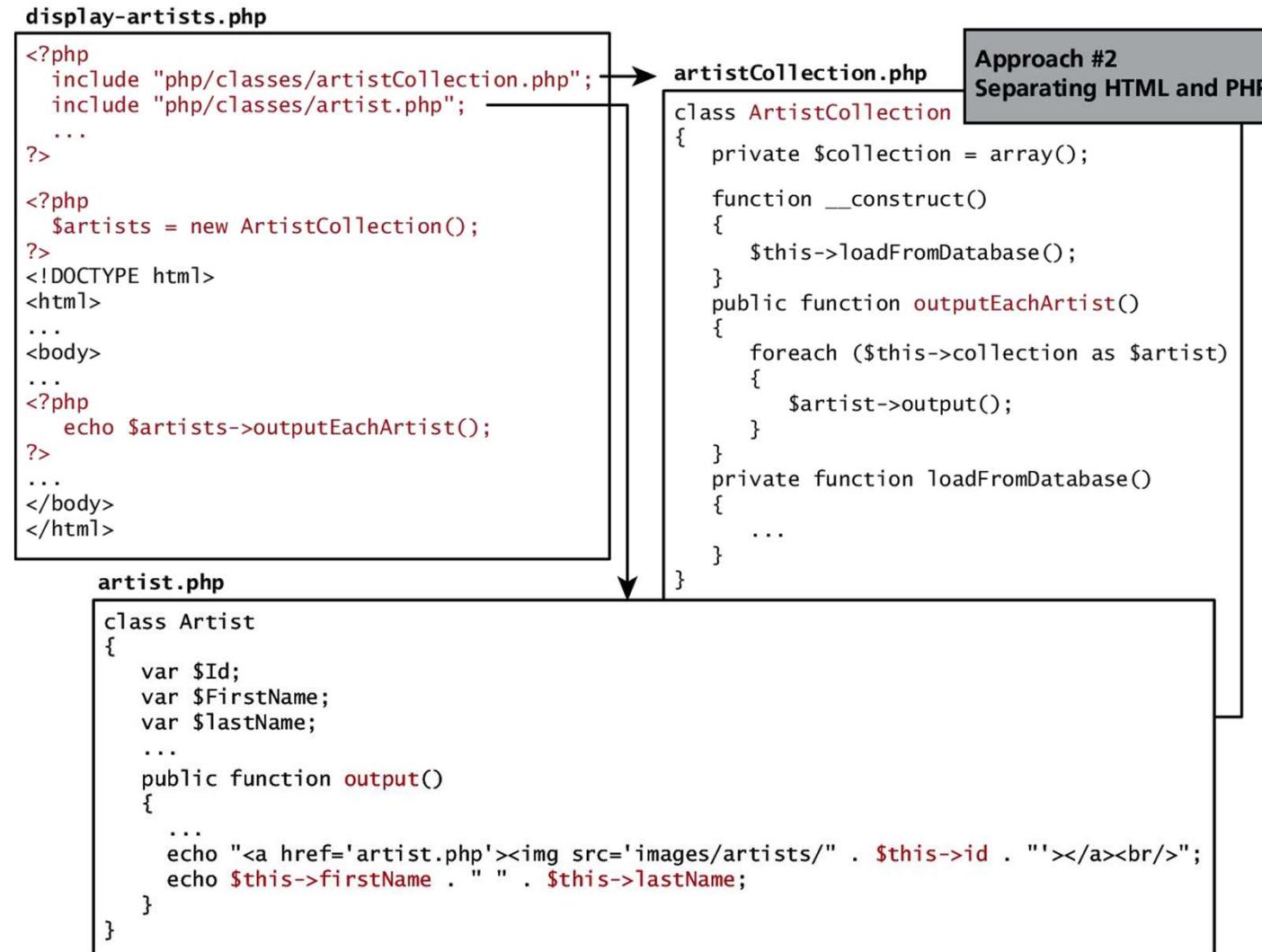
`display-artists.php`

```
<?php
    $db = new mysqli('localhost', 'dbuser', 'dbpassword', 'dbname');
    $sql = "SELECT * FROM Artists ORDER BY lastName";
    $result = $db->query($sql);
?>
...
<body>
...
<ul>
<?php
while( $row = $result->fetch_assoc() ) {
    echo "<li>";
?>
 
<?php
    echo "<a href='artist.php'><img src='images/artists/" . $row['id'] . "'></a><br/>";
    echo $row['firstName'] . " " . $row['lastName'];
    echo "</li>";
}
?>
</ul>
...
<?php
$result->close();
$db->close ();
?>
</body>
</html>
```

Approach #1
Mixing HTML and PHP

HTML and PHP

Two approaches



PHP Comments

3 kinds

The types of comment styles in PHP are:

- **Single-line comments.** Lines that begin with a # are comment lines and will not be executed.
- **Multiline (block) comments.** These comments begin with a /* and encompass everything that is encountered until a closing */ tag is found.
- **End-of-line comments.** Whenever // is encountered in code, everything up to the end of the line is considered a comment.

PHP Comments

3 kinds

```
<?php
```

single-line comment

```
/*
```

This is a multiline comment.

They are a good way to document functions or complicated blocks of code

```
*/
```

```
$artist = readDatabase(); // end-of-line comment
```

```
?>
```

Variables

- Variables in PHP are **dynamically typed**.
- Variables are also **loosely typed** in that a variable can be assigned different data types over time
- To declare a variable you must preface the variable name with the dollar (\$) symbol.

```
$count = 42;
```

Data Types

Data Type	Description
Boolean	A logical true or false value
Integer	Whole numbers
Float	Decimal numbers
String	Letters
Array	A collection of data of any type (covered in the next chapter)
Object	Instances of classes

Constants

A **constant** is somewhat similar to a variable, except a constant's value never changes . . . in other words it stays constant.

- Typically defined near the top of a PHP file via the **define()** function
- once it is defined, it can be referenced without using the \$ symbol

Constants

```
<?php

# Uppercase for constants is a programming convention
define("DATABASE_LOCAL", "localhost");
define("DATABASE_NAME", "ArtStore");
define("DATABASE_USER", "Fred");
define("DATABASE_PASSWD", "F5^7%ad");
...
# notice that no $ prefaces constant names
$db = new mysqli(DATABASE_LOCAL, DATABASE_NAME, DATABASE_USER,
DATABASE_NAME);

?>
```

LISTING 8.4 PHP constants

Writing to Output

Hello World

To output something that will be seen by the browser,
you can use the echo() function.

echo ("hello"); //long form

echo "hello"; //shortcut

String Concatenation

Easy

Strings can easily be appended together using the concatenate operator, which is the period (.) symbol.

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

Will Output **Hello World**

String Concatenation

Example

```
$firstName = "Pablo";  
$lastName = "Picasso";  
/*
```

Example one:

These two lines are equivalent. Notice that you can reference PHP variables within a string literal defined with double quotes. The resulting output for both lines is: Pablo Picasso

```
*/
```

```
echo "<em>" . $firstName . " ". $lastName. "</em>";
```

```
echo "<em> $firstName $lastName </em>";
```

String Concatenation

Example

```
/*
```

Example two:

These two lines are also equivalent. Notice that you can use either the single quote symbol or double quote symbol for string literals.

```
*/
```

```
echo "<h1>";
```

```
echo '<h1>';
```

String Concatenation

Example

```
/*
```

Example three:

These two lines are also equivalent. In the second example, the escape character (the backslash) is used to embed a double quote within a string literal defined within double quotes.

```
*/
```

```
echo '';
```

```
echo "<img src=\"23.jpg\" >";
```

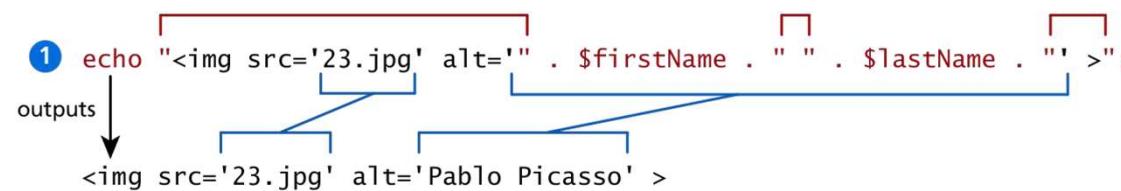
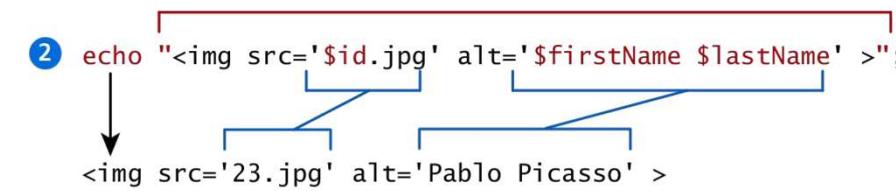
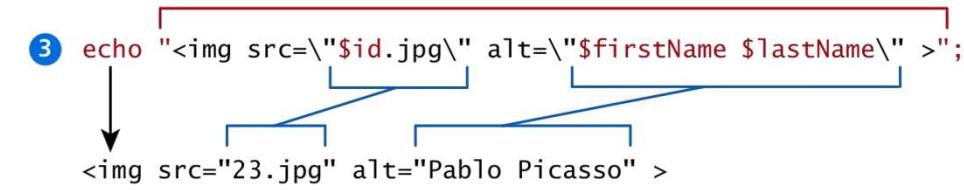
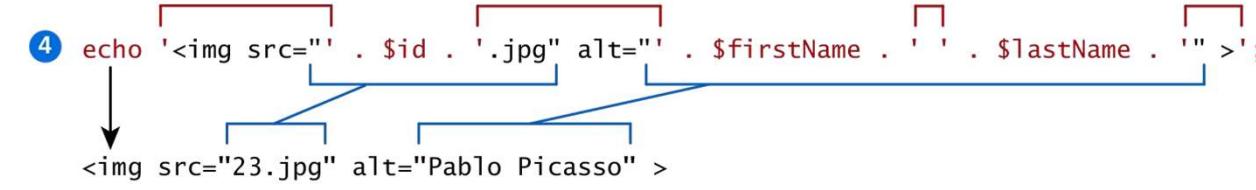
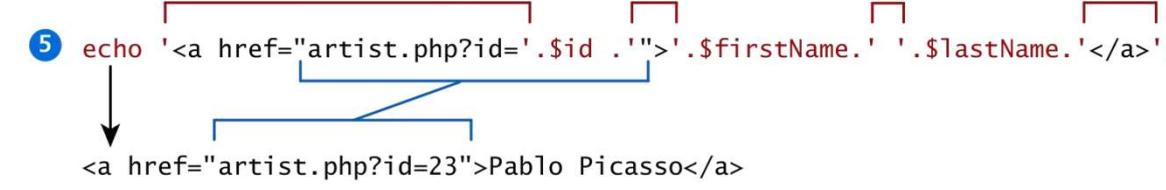
String escape Sequences

Sequence	Description
\n	Line feed
\t	Horizontal tab
\\\	Backslash
\\$	Dollar sign
\"	Double quote

Complicated Concatenation

```
echo "<img src='23.jpg' alt=\"". $firstName . " ". $lastName . "'>";  
echo "<img src='".$id.jpg' alt='".$firstName $lastName'>";  
echo "<img src=\"$id.jpg\" alt=\"$firstName $lastName\">";  
echo '';  
echo '<a href="artist.php?id=' . $id . "'>' . $firstName . ' ' . $lastName . '</a>';
```

Illustrated Example

- ① `echo "";`
outputs

``
- ② `echo "";`

``
- ③ `echo "";`

``
- ④ `echo '' ;`

``
- ⑤ `echo ''.$firstName.' '.$lastName.'';`

`Pablo Picasso`

PrintF

Good ol' printf

As an alternative, you can use the **printf()** function.

- derived from the same-named function in the C programming language
- includes variations to print to string and files (sprintf, fprintf)
- takes at least one parameter, which is a string, and that string optionally references parameters, which are then integrated into the first string by placeholder substitution
- Can also apply special formatting, for instance, specific date/time formats or number of decimal places

Printf

Illustrated example

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

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

outputs ↓

Placeholders Precision specifier

The box is 1.57 pounds.

Printf

Type specifiers

Each placeholder requires the percent (%) symbol in the first parameter string followed by a type specifier.

- b for binary
- d for signed integer
- f for float
- o for octal
- x for hexadecimal

Printf

Precision

- Precision allows for control over how many decimal places are shown. Important for displaying calculated numbers to the user in a “pretty” way.
- Precision is achieved in the string with a period (.) followed by a number specifying how many digits should be displayed for floating-point numbers.

Section 4 of 5

PROGRAM CONTROL

If...else

The syntax for conditionals in PHP is almost identical to that of JavaScript

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

LISTING 8.7 Conditional statement using if . . . else

If...else

Alternate syntax

```
<?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 } ?>

<?php
    // equivalent to the above conditional
    if ($userStatus == "loggedin") {
        echo '<a href="account.php">Account</a> ';
        echo '<a href="logout.php">Logout</a>';
    }
    else {
        echo '<a href="login.php">Login</a> ';
        echo '<a href="register.php">Register</a>';
    }
?
?>
```

LISTING 8.8 Combining PHP and HTML in the same script

Switch...case

Nearly identical

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

LISTING 8.9 Conditional statement using switch

While and Do..while

Identical to other languages

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

LISTING 8.10 while loops

For

Identical to other languages

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

LISTING 8.11 for loops

Alternate syntax for Control Structures

PHP has an alternative syntax for most of its control structures.
In this alternate syntax

- the colon (:) replaces the opening curly bracket,
- while the closing brace is replaced with endif;, endwhile;, endfor;, endforeach;, or endswitch;

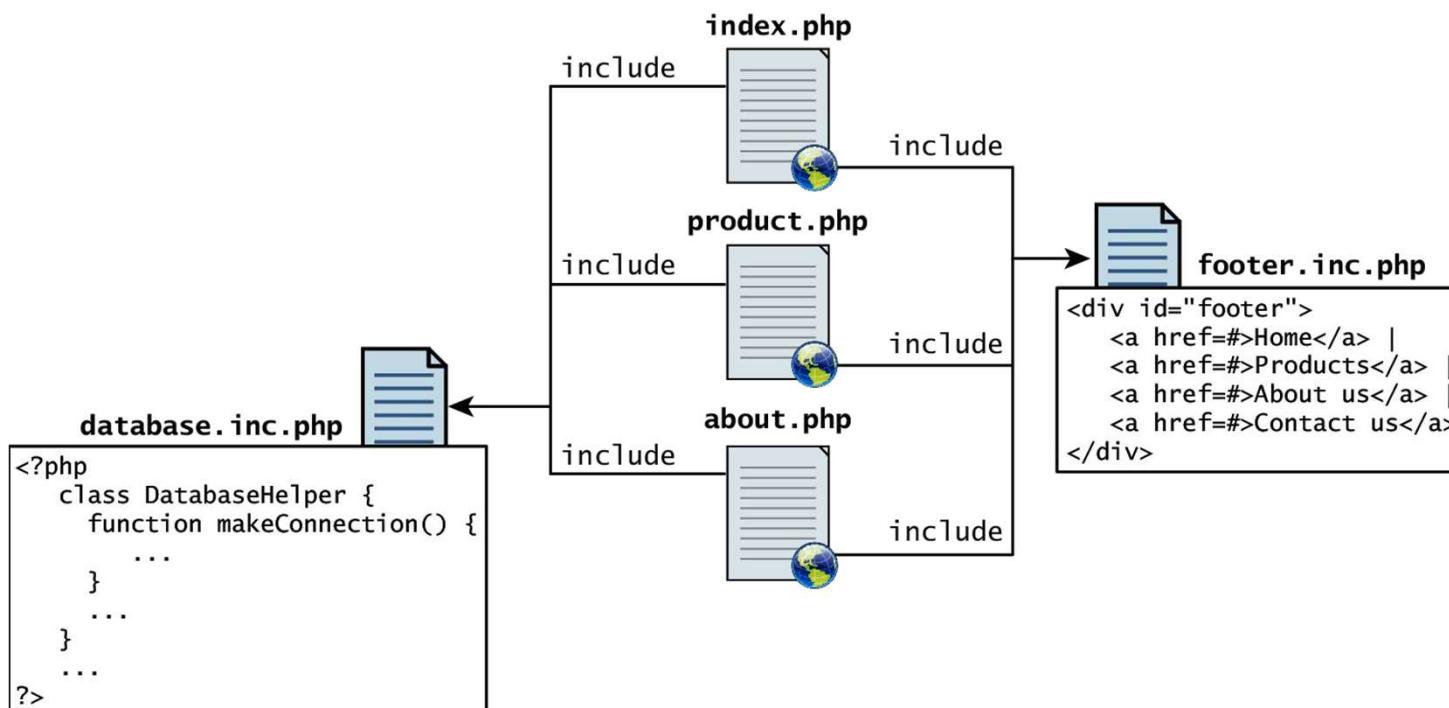
```
<?php if ($userStatus == "loggedin") : ?>
    <a href="account.php">Account</a>
    <a href="logout.php">Logout</a>
<?php else : ?>
    <a href="login.php">Login</a>
    <a href="register.php">Register</a>
<?php endif; ?>
```

LISTING 8.12 Alternate syntax for control structures

Include Files

Organize your code

PHP does have one important facility that is generally unlike other nonweb programming languages, namely the ability to include or insert content from one file into another.



Include Files

Organize your code

PHP provides four different statements for including files, as shown below.

include "somefile.php";

include_once "somefile.php";

require "somefile.php";

require_once "somefile.php";

With include, a warning is displayed and then execution continues.
With require, an error is displayed and execution stops.

Include Files

Scope

Include files are the equivalent of copying and pasting.

- Variables defined within an include file will have the scope of the line on which the include occurs
- Any variables available at that line in the calling file will be available within the called file
- If the include occurs inside a function, then all of the code contained in the called file will behave as though it had been defined inside that function

Section 5 of 5

FUNCTIONS

Functions

You mean we don't write everything in main?

Just as with any language, writing code in the main function (which in PHP is equivalent to coding in the markup between <?php and ?> tags) is not a good habit to get into.

A **function** in PHP contains a small bit of code that accomplishes one thing. In PHP there are two types of function: user-defined functions and built-in functions.

1. A **user-defined function** is one that you the programmer define.
2. A **built-in function** is one of the functions that come with the PHP environment

Functions

syntax

```
/**  
 * This function returns a nicely formatted string using the current  
 * system time.  
 */  
function getNiceTime() {  
    return date("H:i:s");  
}
```

LISTING 8.13 The definition of a function to return the current time as a string

While the example function in Listing 8.13 returns a value, there is no requirement for this to be the case.

Functions

No return – no big deal.

```
/**  
 * This function outputs the footer menu  
 */  
function outputFooterMenu() {  
    echo '<div id="footer">';  
    echo '<a href="#">Home</a> | <a href="#">Products</a> | ';  
    echo '<a href="#">About us</a> | <a href="#">Contact us</a>';  
    echo '</div>';  
}
```

LISTING 8.14 The definition of a function without a return value

Call a function

- ❑ Now that you have defined a function, you are able to use it whenever you want to. To call a function you must use its name with the () brackets.
- ❑ Since `getNiceTime()` returns a string, you can assign that return value to a variable, or echo that return value directly, as shown below.

```
$output = getNiceTime();
```

```
echo getNiceTime();
```

If the function doesn't return a value, you can just call the function:

```
outputFooterMenu();
```

Parameters

Parameters are the mechanism by which values are passed into functions.

To define a function with parameters, you must decide

- how many parameters you want to pass in,
- and in what order they will be passed
- Each parameter must be named

Parameters

```
/**  
 * This function returns a nicely formatted string using the current  
 * system time. The showSeconds parameter controls whether or not to  
 * include the seconds in the returned string.  
 */  
function getNiceTime($showSeconds) {  
    if ($showSeconds==true)  
        return date("H:i:s");  
    else  
        return date("H:i");  
}
```

LISTING 8.15 A function to return the current time as a string with an integer parameter

Thus to call our function, you can now do it in two ways:

```
echo getNiceTime(1); //this will print seconds  
echo getNiceTime(0); //will not print seconds
```

Parameter Default Values

```
/**  
 * This function returns a nicely formatted string using the current  
 * system time. The showSeconds parameter controls whether or not  
 * to show the seconds.  
 */  
function getNiceTime($showSeconds=1){  
    if ($showSeconds==true)  
        return date("H:i:s");  
    else  
        return date("H:i");  
}
```

LISTING 8.16 A function to return the current time with a parameter that includes a default

Now if you were to call the function with no values, the \$showSeconds parameter would take on the default value, which we have set to 1, and return the string with seconds.

Pass Parameters by Value

By default, arguments passed to functions are **passed by value** in PHP. This means that PHP passes a copy of the variable so if the parameter is modified within the function, it does not change the original.

```
function changeParameter($arg) {
    $arg += 300;
    echo "<br/>arg=" . $arg;
}

$initial = 15;
echo "<br/>initial=" . $initial;      // output: initial=15
changeParameter($initial);            // output: arg=315
echo "<br/>initial=" . $initial;      // output: initial=15
```

LISTING 8.17 Passing a parameter by value

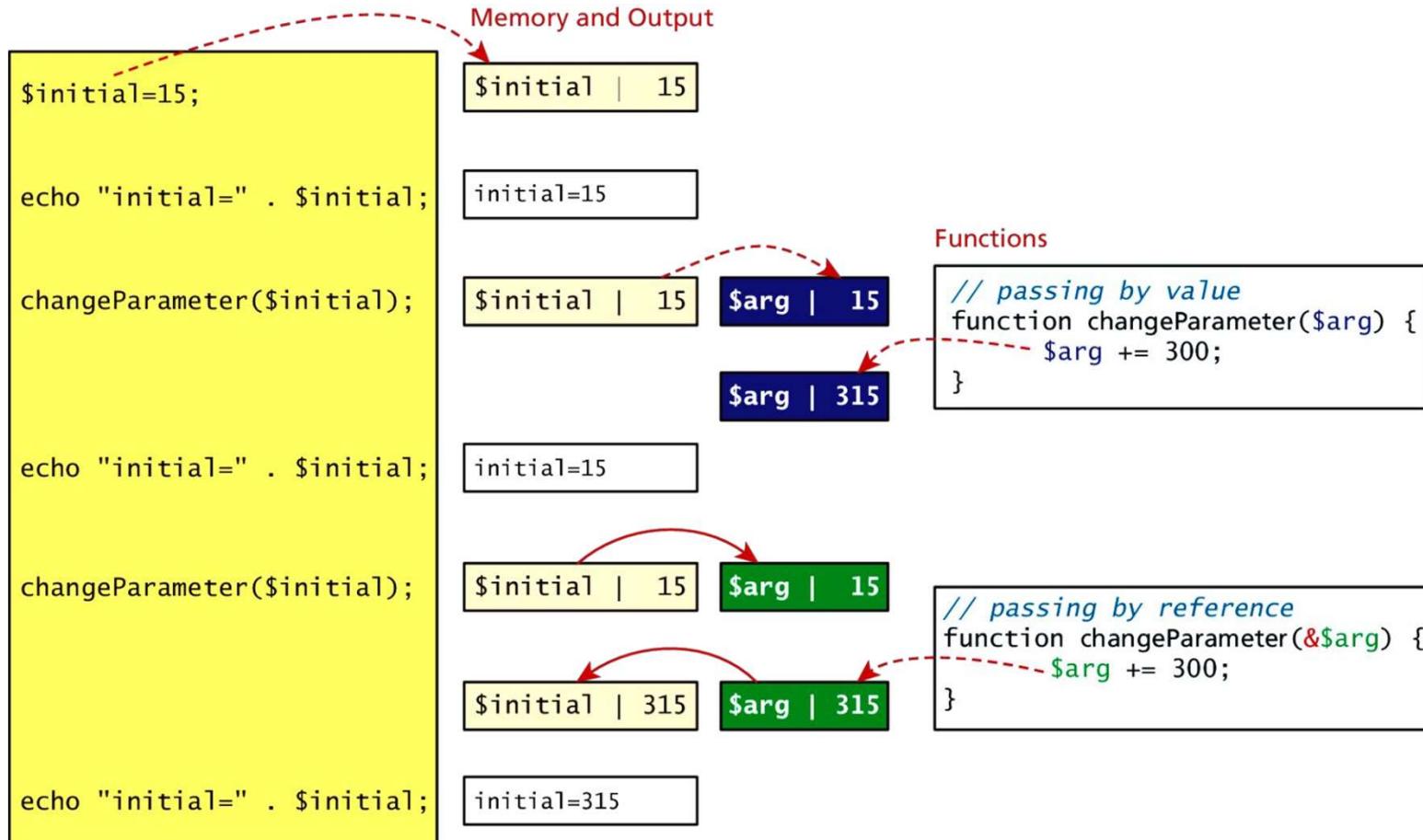
Pass Parameters by Reference

- 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

```
function changeParameter(&$arg) {  
    $arg += 300;  
    echo "<br/>arg=". $arg;  
}  
  
$initial = 15;  
echo "<br/>initial=" . $initial;    // output: initial=15  
changeParameter($initial);        // output: arg=315  
echo "<br/>initial=" . $initial;    // output: initial=315
```

LISTING 8.18 Passing a parameter by reference

Value vs Reference



Variable Scope in functions

- All variables defined within a function (such as parameter variables) have **function scope**, meaning that they are only accessible within the function.
- Any variables created outside of the function in the main script are unavailable within a function.

```
$count= 56;

function testScope()
    echo $count; // outputs 0 or generates run-time
                  //warning/error

}

testScope();
echo $count; //outputs 56
```

Global variables

Sometimes unavoidable

- ❖ Variables defined in the main script are said to have **global scope**.
- ❖ Unlike in other programming languages, a global variable is not, by default, available within functions.
- ❖ PHP does allow variables with global scope to be accessed within a function using the **global** keyword

```
$count= 56;

function testScope() {
    global $count;
    echo $count;    // outputs 56
}

testScope();
echo $count;      // outputs 56
```

LISTING 8.19 Using the global keyword

What You've Learned

1 Server-Side Development

2 Web Server's Responsibilities

3 Quick Tour of PHP

4 Program Control

5 Functions