USN | | | | | | | | | | |

Internal Assessment Test 2 – Oct. 2018

Sub:	Web Technologies & its Applications					Sub Code:	15CS71	Branch:	CSE
Date:	15-10-2018	Duration:	90 min's	Max Marks:	50	Sem / Sec:	7 – A, B & C		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1 (a)	Briefly explain positioning elements with examples.	[10]	CO2	L2

It is possible to move an item from its regular position in the normal flow, and even move an item outside of the browser viewport so it is not visible or to position it so it is always visible in a fixed position while the rest of the content scrolls.
The position property is used to specify the type of positioning, and the possible values are:

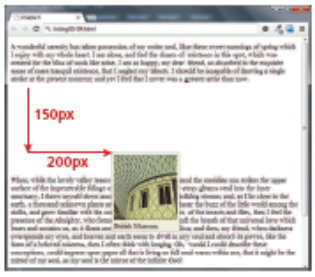
Value	Description
absolute	The element is removed from normal flow and positioned in relation to its nearest positioned ancestor.
fixed	The element is fixed in a specific position in the window even when the document is scrolled.
relative	The element is moved relative to where it would be in the normal flow.
static	The element is positioned according to the normal flow. This is the default.

TABLE 5.1 Position Values

Relative Positioning:

- In **relative positioning** an element is displaced out of its normal flow position and moved relative to where it would have been placed.
- When an element is positioned relatively, it is displaced out of its normal flow position and moved relative to where it would have been placed.
- The other content around the relatively positioned element "remembers" the element's old position in the flow; thus the space the element would have occupied is preserved as shown in Figure 5.4.

```
<p>A wonderful serenity has taken possession of my …
<figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
</figure>
<p>When, while the lovely valley …
```




```
figure {
   border: 1pt solid #A8A8A8;
   background-color: #EDEDDD;
   padding: 5px;
   width: 150px;
   position: relative;
   top: 150px;
   left: 200px;
}
```

FIGURE 5.4 Relative positioning

- As you can see in Figure 5.4, the original space for the positioned

<figure> element is preserved, as is the rest of the document's flow.

- As a consequence, the repositioned element now overlaps other content: that is, the <p> element following the <figure> element does not change to accommodate the moved<figure>.

Absolute Positioning:

- When an element is positioned absolutely, it is removed completely from normal flow. Thus, unlike with relative positioning, space is not left for the moved element, as it is no longer in the normal flow.
- Its position is moved in relation to its container block.
- In the example shown in Figure 5.5, the container block is the <body> element. Like with the relative positioning example, the moved block can now overlap content in the underlying normal flow.

```
<p>A wonderful serenity has taken possession of my …

<figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
</figure>

<p>When, while the lovely valley …
```


```
figure {
   margin: 0;
   border: 1pt solid #A8A8A8;
   background-color: #EDEDDD;
   padding: 5px;
   width: 150px;
   position: absolute;
   top: 150px;
   left: 200px;
}
```

FIGURE 5.5 Absolute positioning

- A moved element via absolute position is actually positioned relative to its nearest **positioned** ancestor container (that is, a block-level element whose position is fixed, relative, or absolute).
- In the example shown in Figure 5.6, the <figcaption> is absolutely positioned; it is moved 150 px down and 200 px to the left of its nearest positioned ancestor, which happens to be its parent (the <figure> element).

Z-Index:

- Each positioned element has a stacking order defined by the z-index property (named for the z-axis).
- Items closest to the viewer (and thus on the top) have a larger **z-index** value.
- First, only positioned elements will make use of their z-index, simply setting the z-index value of elements will not necessarily move them on top or behind other items.

Fixed Position:

- The fixed position value is used relatively infrequently. It is a type of absolute positioning, except that the positioning values are in relation to the viewport (i.e., to the browser window).

- Elements with **fixed positioning** do not move when the user scrolls up or down the page, as can be seen in Figure 5.8.
- The fixed position is most commonly used to ensure that navigation elements or advertisements are always visible.

```
figure {
   ...
   position: fixed;
   top: 0;
   left: 0;
}
```

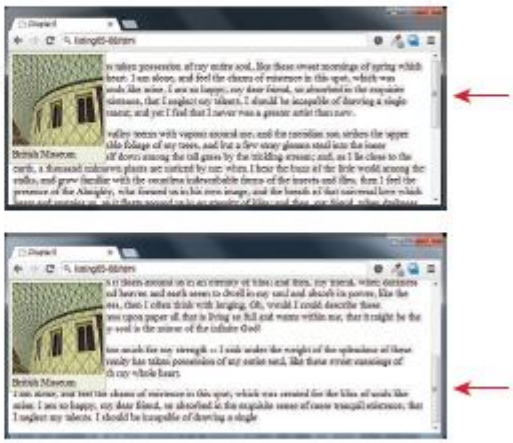


FIGURE 5.8 Fixed position

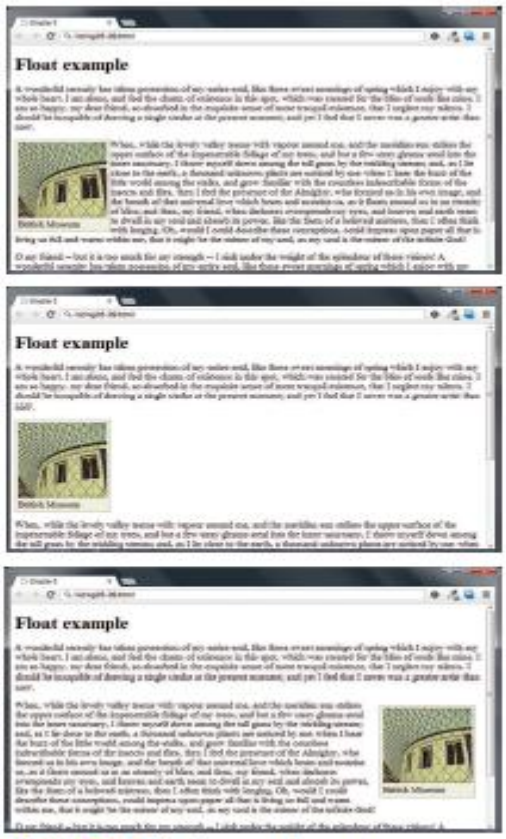
2 (a) What does floating an element do? How to do you float an element? [05] CO2 L2

Floating Elements:

- It is possible to displace an element out of its position in the normal flow via the CSS float **property**.
- An element can be floated to the left or floated to the right.
- When an item is floated, it is moved all the way to the far left or far right of its containing block and the rest of the content is "re-flowed" around the floated element, as can be seen in Figure 5.9.
- Notice that a floated block-level element must have a width specified; if you do not, then the width will be set to auto, which will mean it implicitly fills the entire width of the containing block, and there thus will be no room available to flow content around the floated item. Also note in the final example in Figure 5.9 that the margins on the floated element are respected by the content that surrounds the floated element.

Floating within a Container:

- It should be reiterated that a floated item moves to the left or right of its container (also called its **containing block**). In Figure 5.9, the containing block is the HTML document itself so the figure moves to the left or right of the browser window.




```
<h1>Float example</h1>
<p>A wonderful serenity has taken ...</p>
<figure>
   <img src="images/B28.jpg" alt="" />
   <figcaption>British Museum</figcaption>
</figure>
<p>When, while the lovely valley …</p>

figure {
   border: 1pt solid #A8A8A8;
   background-color: #EDEDDD;
   margin: 0;
   padding: 5px;
   width: 150px;
}
```

Notice that a floated block-level element **must** have a width specified.

```
figure {
   …
   width: 150px;
   float: left;
}
```

```
figure {
   …
   width: 150px;
   float: right;
   margin: 10px;
}
```

FIGURE 5.9 Floating an element

(b) What is viewport? Why is it important? [05] CO2 L2

- The way the webpage works is the mobile browser renders the page on a canvas called the **viewport**.
- On iPhones, for instance, the viewport width is 980 px, and then that viewport is scaled to fit the current width of the device (which can change with orientation and with newer versions that have more physical pixels in the screen), as shown in Figure 5.31.
- The mobile Safari browser introduced the viewport <meta> tag as a way for developers to control the size of that initial viewport.
- If the developer has created a responsive site similar to that shown below. One that will scale to fit a smaller screen, she may not want the mobile browser to render it on the full-size viewport.
- The web page can tell the mobile browser the viewport size to use via the viewport <meta> element, as shown below:

```
<html>
<head>
<meta name="viewport" content="width=device-width" />
```

- By setting the viewport as in this listing, the page is telling the browser

that no scaling is needed, and to make the viewport as many pixels wide as the device screen width.

- This means that if the device has a screen that is 320 px wide, the viewport width will be 320 px; if the screen is 480 px (for instance, in landscape mode), then the viewport width will be 480 px.

3 (a) Explain the role of grid systems in the creation of multicolumn layouts. [06] CO2 L2

- **Grid systems** make it easier to create multicolumn layouts.
- There are many CSS grid systems; some of the most popular are Bootstrap (**twitter.github.com/bootstrap**), Blueprint (**www.blueprintcss.org**), and 960 (**960.gs**).
- The most important of these capabilities is a grid system. Print designers typically use grids as a way to achieve visual uniformity in a design.
- In print design, the very first thing a designer may do is to construct, for instance, a 5- or 7- or 12-column grid in a page layout program like InDesign or Quark Xpress. The rest of the document, whether it be text or graphics, will be aligned and sized according to the grid.
- CSS frameworks provide similar grid features. The 960 framework uses either a 12- or 16-column grid.
- Bootstrap uses a 12-column grid. Blueprint uses a 24-column grid.
- The grid is constructed using elements with classes defined by the framework. The HTML elements for the rest of your site are then placed within these elements
- In the 960 system, a row is terminated with . In Bootstrap, content must be placed within the row container.

```
<head>
  <link rel="stylesheet" href="reset.css" />
  <link rel="stylesheet" href="text.css" />
  <link rel="stylesheet" href="960.css" />
</head>
<body>
  <div class="container_12">
    <div class="grid_2">
      left column
    </div>
    <div class="grid_7">
      main content
    </div>
    <div class="grid_3">
      right column
    </div>
    <div class="clear"></div>
  </div>
</body>
```

LISTING 5.2 Using the 960 grid

- While Listing 5.3 shows the same thing in the Bootstrap framework.

```
<head>
   <link href="bootstrap.css" rel="stylesheet">
</head>
<body>
  <div class="container">
    <div class="row">
      <div class="col-md-2">
        left column
      </div>
      <div class="col-md-7">
         main content
      </div>
      <div class="col-md-3">
        right column
      </div>
    </div>
  </div>
</body>
```

LISTING 5.3 Using the Bootstrap grid

- In both systems, elements are laid out in rows; elements in a row will span from 1 to 12 columns.

(b) What are the advantages and disadvantages of Client and server side scripting? [04] CO2 L2

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.

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

■ There is no guarantee that the client has JavaScript enabled, meaning any required functionality must be housed on the server, despite the possibility that it could be offloaded.

■ 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. JavaScript has often been used through inline HTML hooks that are embedded into the HTML of a web page. Although this technique has been used for years, it has the distinct disadvantage of blending HTML and JavaScript together, which decreases code readability, and increases the difficulty of web development.

4 (a)	Compare graceful degradation with progressive enhancement.	[05]	CO3	L2

- The principle of **graceful degradation** is one possible strategy. 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).
- The alternate strategy is **progressive enhancement**, which takes the opposite approach to the problem. 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. (Eventually, one does have to stop supporting ancient browsers; many developers have, for instance, stopped supporting IE 6.)
- 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. For instance, users using the current version of Opera and Chrome might see the fancy HTML5 color input form elements (since both support it at present), users using current versions of other browsers might see a jQuery plug-in that has similar functionality, while users of IE 7 might just see a simple text box.

(b)	What are the different ways in which JavaScript can be linked to an HTML page?	[05]	CO3	L2

- JavaScript can be linked to an HTML page as **inline**, **embedded**, or **external**.

Inline JavaScript

- Inline JavaScript refers to the practice of including JavaScript code directly within certain HTML attributes, such as that shown below :

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

Embedded JavaScript

- 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>
```

External JavaScript

- JavaScript external files have the extension .js. Modern websites often have links to several, maybe even dozens, of external JavaScript files (also called **libraries**).
- These external files typically contain function definitions, data definitions, and other blocks of JavaScript code.

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

- The link to the external JavaScript file is placed within the <head> element, just as was the case with links to external CSS files. While this is convention, it is in fact possible to place these links anywhere within the <body> element.
- Placing them either in the <head> element or the very bottom of the <body> element.

[10] | CO4 | L2

5. a) Define DOM. Explain DOM property and its methods.[10]

The Document Object Model (DOM) way of programmatically accessing the elements and attributes within the HTML. This is accomplished through a programming interface (API) called the Document Object Model (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.

Method	Description
createAttribute()	Creates an attribute node

Eg:

```
<head>
<style>
.democlass {
   color: red;
}
</style>
</head>
<body>

<h1>Hello World</h1>
<h1>Hello World22</h1>
```

```
<p>Click the button to create a "class" attribute with the value "democlass" and insert it to the H1 element above.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
    var h1 = document.getElementsByTagName("H1")[1];
    var att = document.createAttribute("class");
    att.value = "democlass";
    h1.setAttributeNode(att);
}
</script>

</body>
```

createElement() Creates an element node
createTextNode() Creates a text node

```
<body>

<p>Click the button to make a BUTTON element with text.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
    var btn = document.createElement("BUTTON");
    var t = document.createTextNode("CLICK ME");
    btn.appendChild(t);
    document.body.appendChild(btn);
}
</script>

</body>
```

getElementById(id) Returns the element node whose id attribute
matches the passed id parameter
getElementsByTagName(name) Returns a NodeList of elements whose tag name
matches the passed name parameter

```
<body>

<p>An unordered list:</p>
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>

<p>Click the button to display the innerHTML of the second li element (index 1).</p>
```

```
<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
    var x = document.getElementsByTagName("LI");
    document.getElementById("demo").innerHTML = x[1].innerHTML;
}
</script>

</body>
```

They include getElementByTagName() and the indispensable getElementById(). While the former method returns an array of DOM nodes (called a NodeList) matching the tag, the latter returns a single DOM element (covered below), that matches the id passed as a parameter.

```
<body>
<h1>Reviews</h1>
<div id="latestComment">
<p>By Ricardo on <time>September 15, 2012</time></p>
<p>Easy on the HDR buddy.</p>
</div>
<hr/>
<div>
<p>By Susan on <time>October 1, 2012</time></p>
<p>I love Central Park.</p>
</div>
<hr/>
</body>
var abc = document.getElementById("latestComment");
var list = document.getElementsByTagName("div");
```

Figure 6.19 Relationship between HTML tags and getElementByID() and getElementsByTagName()

```
// specify the doctype, for example html
var a = document.doctype.name;
// specify the page encoding, for example ISO-8859-1
var b = document.inputEncoding;
```

Element Node Object

Property	Description
className	The current value for the class attribute of this HTML element.
id	The current value for the id of this element.
innerHTML	Represents all the things inside of the tags. This can be read or written to and is the primary way in which we update particular elements using JavaScript.
Style	The style attribute of an element. We can read and modify this property.
tagName	The tag name for the element.

```
<body>
<ul class="example">
  <li class="child">Coffee</li>
  <li class="child">Tea</li>
```

```
</ul>

<p>Click the button to change the text of the first list item (index 0).</p>

<button onclick="myFunction()">Try it</button>

<p><strong>Note:</strong> The getElementsByClassName() method is not supported in Internet Explorer 8 and earlier versions.</p>

<script>
function myFunction() {
   var list = document.getElementsByClassName("example")[0];
   list.getElementsByClassName("child")[0].innerHTML = "Milk";
}
</script>

</body>

<body>

<h1 id="myH1">How to change the style of a header</h1>

<p>Click the button to add a color to the H1 element.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
   document.getElementById("myH1").style.color = "red";
}
</script>

</body>
```

https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_element_style

Modifying a DOM Element

Property	Description
href	The href attribute used in a tag to specify a URL to link to a
name	The name property is a bookmark to identify a, input, textarea, this tag. Unlike id, which is available to all tags,name is limited to certain form-related tags.
form	
src	Links to an external URL that should be loaded img, input, iframe,

into the page (as opposed to href, which is a link to follow when clicked) Script
value The value is related to the value attribute of input, textarea,submit
input tags. Often the value of an input field is user defined, and we use value to get that

5. b)What is verbose technique in JavaScript? Explain with an example

A More Verbose Technique
Appendchild()
<body>

```
<ul id="myList">
  <li>Coffee</li>
  <li>Tea</li>
</ul>

<p>Click the button to append an item to the end of the list.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
   var node = document.createElement("LI");
   var textnode = document.createTextNode("Water");
   node.appendChild(textnode);
   document.getElementById("myList").appendChild(node);
}
</script>
```

Removechild()

```
<ul id="myList"><li>Coffee</li><li>Tea</li><li>Milk</li></ul>

<p>Click the button to remove the first item from the list.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
   var list = document.getElementById("myList");
   list.removeChild(list.childNodes[0]);
}
</script>
```

6.a) What are the responsibilities of web server? Explain

A Web Server's Responsibilities
A web server has many responsibilities beyond responding to requests for HTML files.
These include handling

1. HTTP connections,
2. responding to requests for static and dynamic resources,
3. managing permissions and access for certain resources,
4. encrypting and compressing data, managing multiple domains and URLs,
5. managing database connections, cookies, and state, and uploading and managing files.

Apache and Linux

You can 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; both Apache and PHP make use of configuration files that determine exactly how requests are handled,

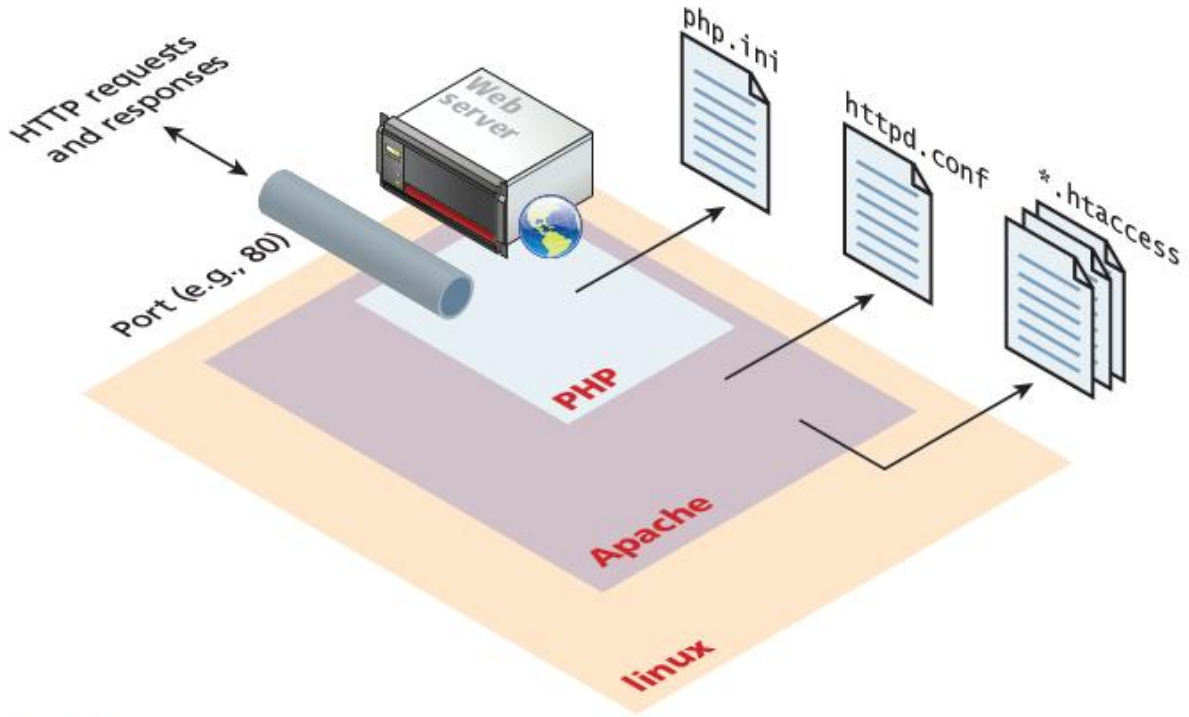


FIGURE 8.5 Linux, Apache, and PHP together

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. As a background process, the Apache daemon waits for incoming HTTP requests. 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. Figure 8.6 illustrates that when a request comes into Apache, each module is given an opportunity to handle some aspect of the request. Some modules handle authorization, others handle URL rewriting, while others handle specific extensions.

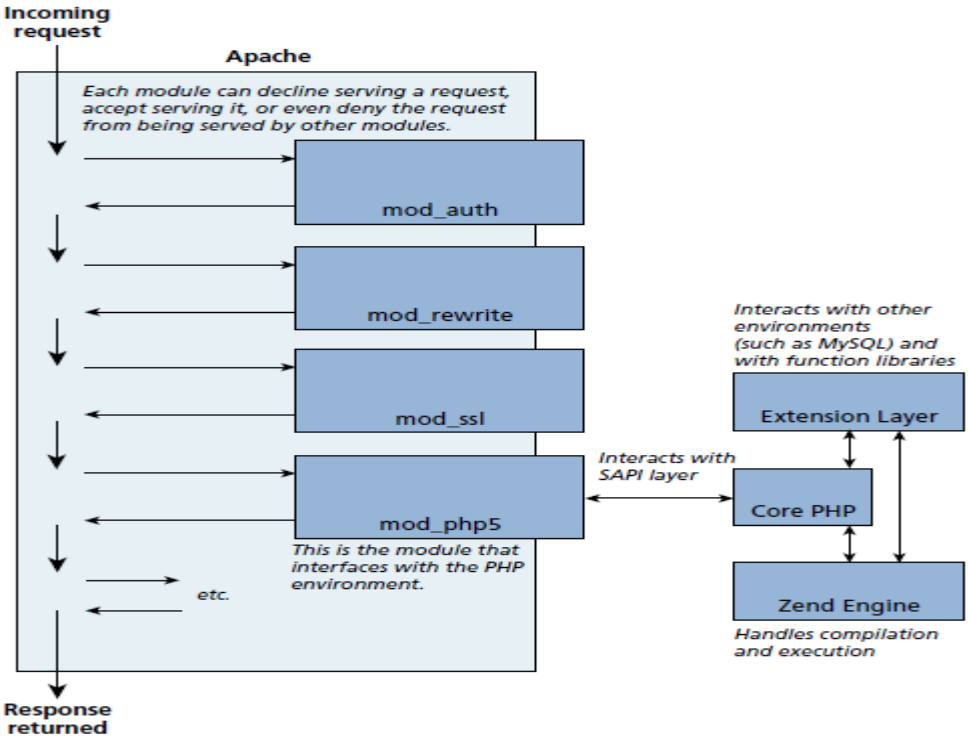


FIGURE 8.6 Apache modules and PHP

APACHE and PHP

PHP is usually installed as an Apache module (though it can alternately be installed as a CGI binary). The PHP module mod_php5 is sometimes referred to as the SAPI (Server Application Programming Interface) layer since it handles the interaction between the PHP environment and the web server environment. Apache runs in two possible modes: multi-process (also called preforked) or multi-threaded (also called worker), which are shown in Figure 8.7. The default installation of Apache runs using the multi-process mode. That is, each request is handled by a separate process of Apache; the term fork refers to the operating system creating a copy of an already running process. Since forking is time intensive, Apache will prefork a set number of additional processes in advance of their being needed. Forking is relatively efficient on Unixbased operating systems, but is slower on Windows-based operating systems. As well, a key advantage of multi-processing mode is that each process is insulated from other processes; that is, problems in one process can't affect other processes.

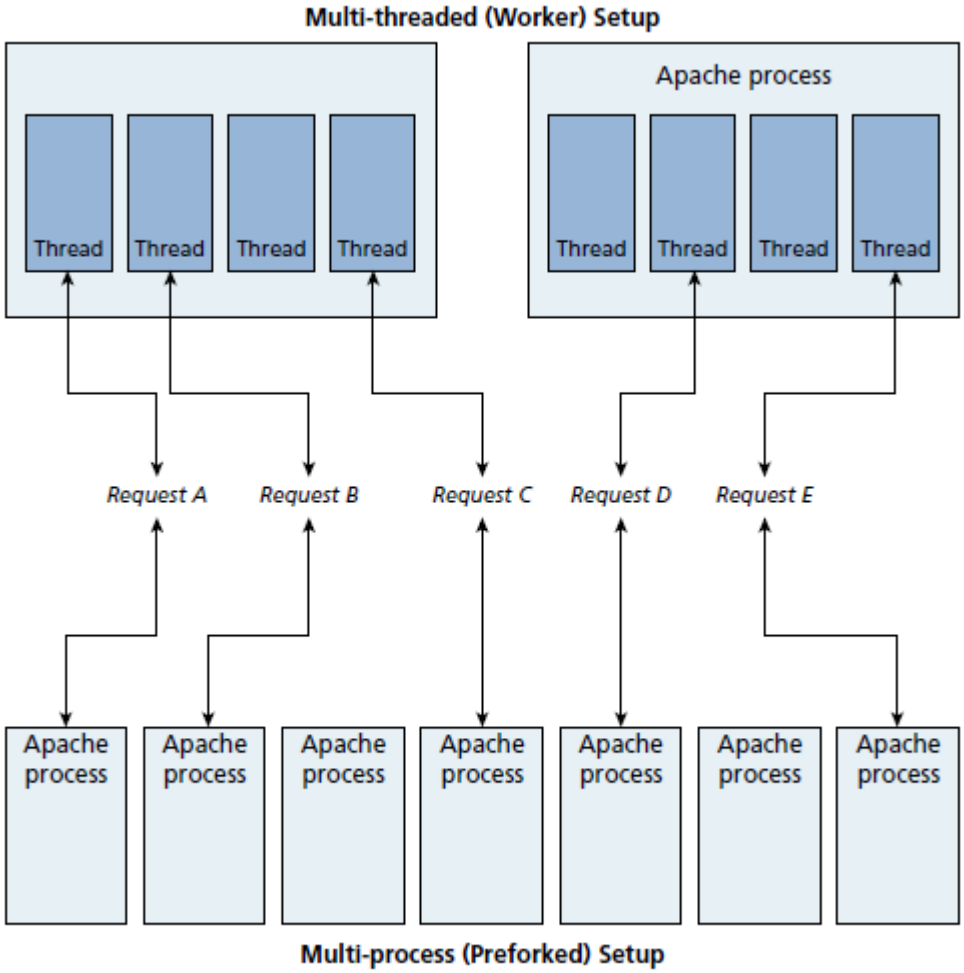


FIGURE 8.7 Multi-threaded versus multi-process

In the multi-threaded mode, a smaller number of Apache processes are forked. Each of the processes runs multiple threads. A thread is like a lightweight process that is contained within an operating system process. A thread uses less memory than a process, and typically threads share memory and code; as a consequence, the multi-threaded mode typically scales better to large loads. When using this mode, all modules running within Apache have to be thread safe. Unfortunately, not every PHP module is thread-safe, and the thread safety of PHP in general is quite disputed.

PHP itself is written in the C programming language and is composed of three main modules:
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. Extension layer. This module defines functions for interacting with services outside of PHP. This includes libraries for MySQL (and other

databases), FTP, SOAP web services, and XML processing, among others. Zend Engine. This module handles the reading in of a requested PHP file, compiling it, and executing it. Figure 8.8 illustrates (somewhat imaginatively) how the Zend Engine operates behind the scenes when a PHP page is requested. The Zend Engine is a virtual machine (VM) analogous to the Java Virtual Machine or the Common Language Runtime in the .NET Framework. A VM is a software program that simulates a physical computer; while a VM can operate on multiple platforms, it has the disadvantage of executing slower than a native binary application.

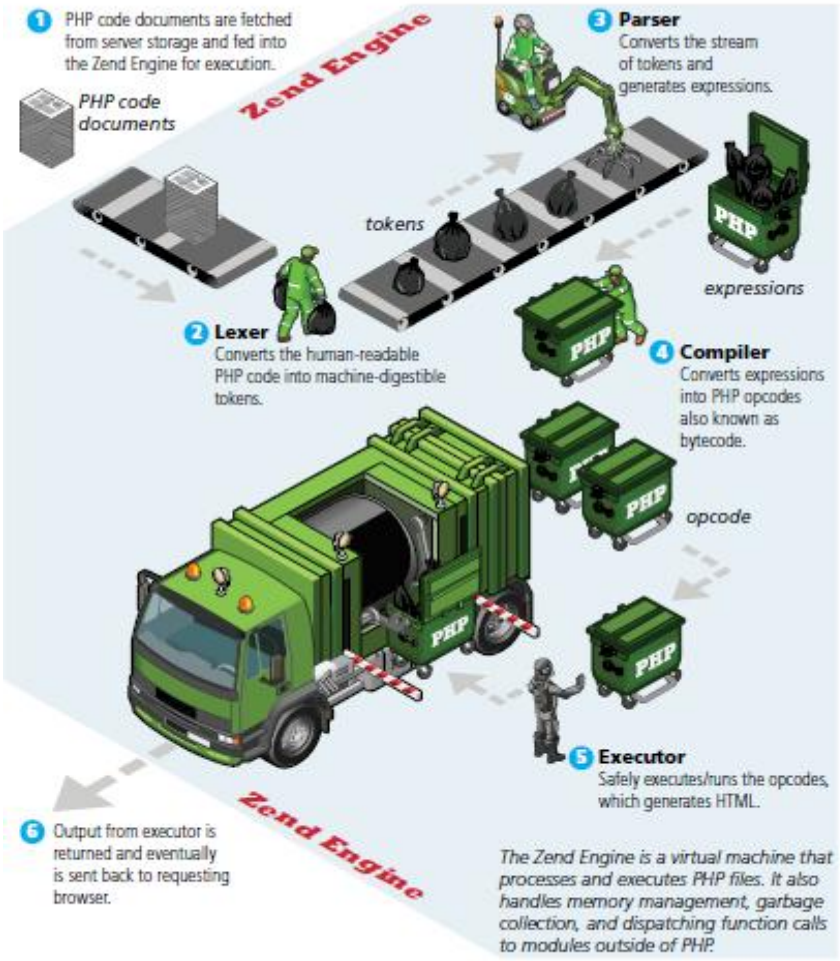


FIGURE 8.8 Zend Engine

7. Briefly explain the JavaScript event classes with examples.

Event Types

The classes are mouse events, keyboard events, form events, and frame events.

Mouse Events

Mouse events are defined to capture a range of interactions driven by the mouse.

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

table 6.7 Mouse Events in JavaScript

Keyboard Events
Keyboard events are often overlooked by novice web developers, but are important tools for power users.

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)

table 6.8 Keyboard Events in JavaScript

Form Events
Forms are the main means by which user input is collected and transmitted to the server.

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 prevalidation when the user submits the form in JavaScript before sending the data on to the server.

table 6.9 Form Events in JavaScript

```
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 (see Table 6.10) 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. In fact, every nontrivial event listener you write requires that the HTML be fully loaded.

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

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

table 6.10 Frame Events in JavaScript

8. Explain functions in PHP elaborately with examples.

Functions

In PHP there are two types of function:

1. User-defined functions

2. Built-in functions.

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

Function Syntax

To create a new function you must think of a name for it, and consider what it willdo. Functions can return values to the caller, or not return a value. They can be set upto take or not take parameters.

```
function getNiceTime() {
return date("H:i:s");
}
```

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

```
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>';
}
```

The definition of a function without a return value

Calling a Function

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 returnvalue directly, as shown below.

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

If the function doesn't return a value, you can just call the function:
outputFooterMenu();

Parameters

It is more common to define functions with parameters, since functions are morepowerful and reusable when their output depends on the input they get. **Parameters**are the mechanism by which values are passed into functions, and there are somecomplexities that allow us to have multiple parameters, default values, and to passobjects by reference instead of value.

To define a function with parameters, you must decide how many parametersyou want to pass in, and in what order they will be passed. Each parameter must benamed.

```
function getNiceTime($showSeconds) {
if ($showSeconds==true)
return date("H:i:s");
else
return date("H:i");
}
```

A function to return the current time as a string with an integer parameterThus 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
```

In fact any nonzero number passed in to the function will be interpreted as truesince the parameter is not type specific.

Parameter Default Values

```
function getNiceTime($showSeconds=1){
if ($showSeconds==true)
return date("H:i:s");
else
return date("H:i");
}
```

A function to return the current time with a parameter that includes a default. If you do include a value in your function call, the default will beoverridden by whatever that value was.

Passing Parameters by Reference

By default, arguments passed to functions are **passed by value** in PHP.

```
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
```

Passing a parameter by value

The mechanism in PHP to specify that a parameter is passed byreference is to add an ampersand (&) symbol next to the parameter name in thefunction 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
```

Passing a parameter by reference

Variable Scope within Functions

It will come as no surprise that all variables defined within a function (such asparameter variables) have **function scope**, meaning that they are only accessiblewithin the function.

```
$count= 56;
function testScope() {
echo $count; // outputs 0 or generates run-time warning/error
}
testScope();
echo $count; // outputs 56
```

While variables defined in the main script are said to have **global scope**,

```
$count= 56;
```

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

Using the global keyword