Module 3

1. **What is JavaScript and listener? Discuss the advantages and disadvantages of client-side scripting.**

**JavaScript** is an interpreted, client-side, event-based, object- oriented scripting language that you can use to add dynamic interactivity to your web pages.

An event **listener** is a procedure in JavaScript that waits for an event to occur .

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 following 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, 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.

1. **Write a JavaScript code that displays text “VTU BELAGAVI” with increasing font size in the interval of 100ms in blue color, when the font size reaches 50 pt it should stop.**

<!DOCTYPE html>

<html>

<head>

<p id="myP1">VTU BELAGAVI</p>

</head>

<body>

<script>

var size = 10;

var myWait1 = setInterval(GrowText1, 100);

function GrowText1() {

if(size<51) {

size = size + 1;

document.getElementById("myP1").style.fontSize = (size+'pt');

document.getElementById("myP1").style.color = "blue";

}

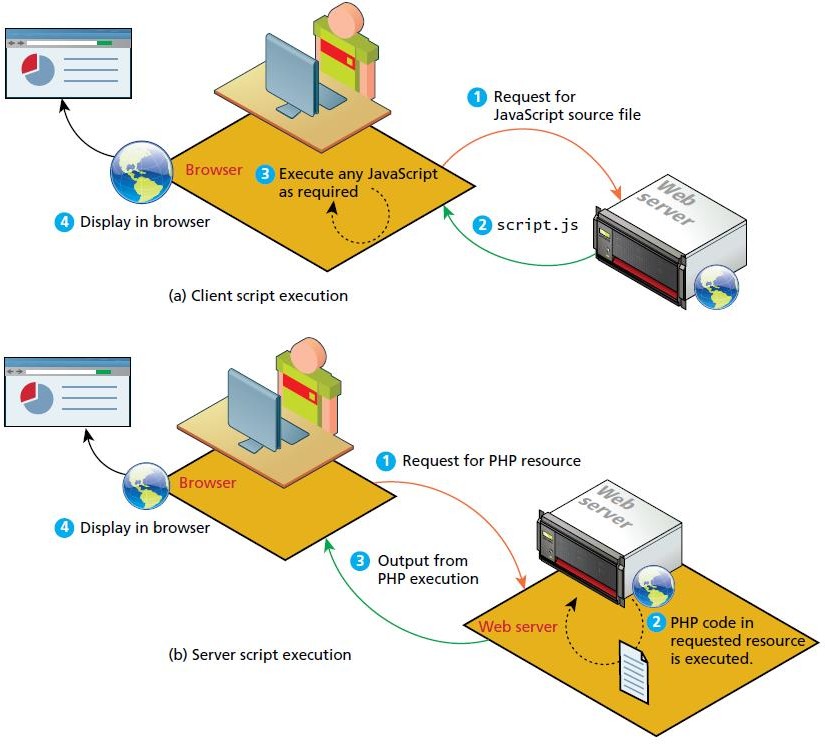
}

</script>

</body>

</html>

1. **With** **a neat diagram, explain client and server script execution.**

****

**Client-side**

* The fundamental difference between client and server scripts is that in a client-side script the code is executed on the client browser.
* Whereas in a server-side script, it is executed on the web server.
* The server sends the JavaScript (that the user could look at), but you have no guarantee that the script will even execute.
* In contrast, server-side source code remains hidden from the client as it is processed on the server. The clients never get to see the code, just the HTML output from the script.

**Server-side**

A server-side script can access any resources made available to it by the server.These resources can be categorized as data storage resources, web services, and software applications, as can be seen in Figure

* The most commonly used resource is data storage, often in the form of a connection to a database management system.
* A database management system (DBMS) is a software system for storing, retrieving, and organizing large amounts of data.
* The term database is often used interchangeably to refer to a DBMS. But it is also used to refer to organized data in general, or even to the files used by DBMS.
* While almost every significant real-world website uses some type of database,many websites also make use of the server’s file system;for example,as a place to store user uploads.

1. **Write a PHP program to greet the user based on time.**

<?php

date\_default\_timezone\_set('Asia/Calcutta');

$Hour = date('G');

if ( $Hour >= 5 && $Hour <= 11 ) {

echo "Good Morning";

} else if ( $Hour >= 12 && $Hour <= 18 ) {

echo "Good Afternoon";

} else if ( $Hour >= 19 || $Hour <= 4 ) {

echo "Good Evening";

}

?>

1. **What are s/w layers? What benefits do they provide? Explain in detail**.

**Layers:**

In object-oriented programming, a software layer is a way of conceptually grouping programming classes that have similar functionality and dependencies. Common software design layer names include:

1. Presentation layer. Classes focused on the user interface.
2. Business layer. Classes that model real-world entities, such as customers, products, and sales.
3. Data layer. Classes that handle the interaction with the data sources.

These layers have different capabilities and responsibilities, but are always considered optional, except in some special circumstances like online games.

Although each layer can perform many tasks, it is helpful to visualize and understand the types of conceptual layers that are common. Figure 3.6 illustrates the idea of JavaScript layers.

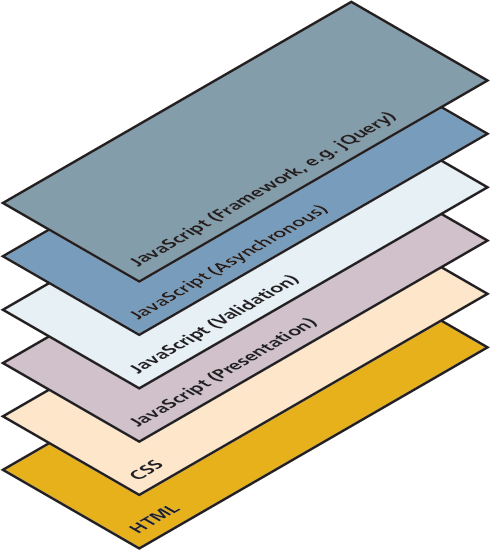


Figure 3.6: JavaScript Layers

**Presentation Layer**

* + This type of programming focuses on the display of information. JavaScript can alter the HTML of a page, which results in a change, visible to the user.
  + These presentation layer applications include common things like creating, hiding, and showing divs, using tabs to show multiple views, or having arrows to page through result sets.
  + This layer is most closely related to the user experience and the most visible to the end user.

**Validation Layer**

* + JavaScript can be also used to validate logical aspects of the user’s experience.
  + This could include, for example, validating a form to make sure the email entered is valid before sending it along.
  + It is often used in conjunction with the presentation layer to create a coherent user experience, where a message to the presentation layer highlights bad fields.
  + Both layers exist on the client machine, although the intention is to prevalidate forms before making transmissions back to the server.

**Asynchronous Layers**

* + During the wait between request and response the browser sits in a loading state and only updates upon receiving the response. In contrast, an asynchronous layer can route requests to the server in the background.
  + In this model, as certain events are triggered, the JavaScript sends the HTTP requests to the server, but while waiting for the response, the rest of the application functions normally, and the browser isn’t in a loading state.
  + When the response arrives JavaScript will (perhaps) update a portion of the page. Asynchronous layers are considered advanced versions of the presentation and validation layers above.

1. **Compare the server-side technologies in detail.**

### Comparing Server-Side Technologies

There are several different server-side technologies for creating web applications. The most common include: Active Server Pages (ASP), ASP.NET, Java Server Pages (JSP), Node.js, Perl, PHP, Python, Ruby on Rails.

All of these technologies share one thing in common: using programming logic, they generate HTML and possibly CSS and JavaScript on the server and send it back to the requesting browser, as shown in Figure 3.13.

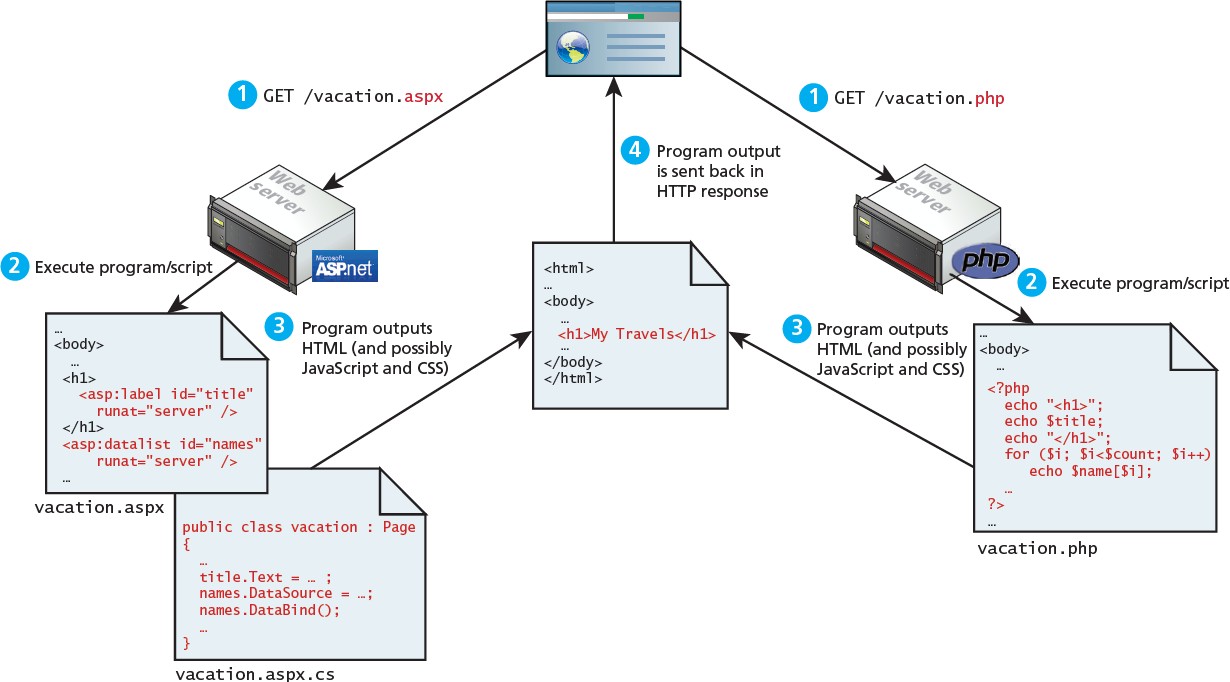


Figure 3.13 Web development technologies

1. **Write a PHP program to demonstrate the session. Program: store page view count on refresh.**

<?php

print"<h2>Refresh page</h2>";

$name="c.txt";

$file=fopen($name,"r");

$hits=fscanf($file,"%d");

fclose($file);

$hits[0]++;

$file=fopen($name,"w");

fprintf($file,"%d",$hits[0]);

fclose($file);

print"total no of views :".$hits[0];

?>

1. **Explain three forms of linking JavaScript to HTML page with suitable code segments.**
   * JavaScript can be linked to an HTML page in a number of ways.
   * Just as CSS styles can be inline, embedded, or external, JavaScript can be included in a number of ways.
   * Just as with CSS these can be combined, but external is the preferred method for cleanliness and ease of maintenance.
   * Running JavaScript scripts in your browser requires downloading the JavaScript code to the browser and then running it.
2. **Inline JavaScript**

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

**<a href="JavaScript:OpenWindow();"more info</a>**

**<input type="button” onclick="alert('Are you sure?');" />**

The same is true with JavaScript. In fact, inline JavaScript is much worse than inline CSS. Inline JavaScript is a real maintenance nightmare, requiring maintainers to scan through almost every line of HTML looking for your inline JavaScript.

1. **Embedded JavaScript**

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

**<script type="text/javascript">**

**alert ("Hello World!"); /\* A JavaScript Comment \*/**

**</script>**

Like its equivalent in CSS, embedded JavaScript is okay for quick testing and for learning scenarios, but is frowned upon for normal real-world pages. Like with inline JavaScript, embedded scripts can be difficult to maintain.

1. **External JavaScript**

Since writing code is a different competency than designing HTML and CSS, it is often advantageous to separate the two into different files. JavaScript supports this separation by allowing links to an external file that contains the JavaScript, as shown

**<head>**

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

**</script>**

**</head>**

This is the recommended way of including JavaScript scripts in your HTML pages. By convention, 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.

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.

1. **With suitable diagrams, explain PHP module in apache. Describe the role of apache threads in web application execution.**

### 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, as shown in Figure 3.14.

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 known by its OS name, httpd) waits for incoming HTTP requests. When a request arrives, Apache then uses modules to determine how to respond to the request.

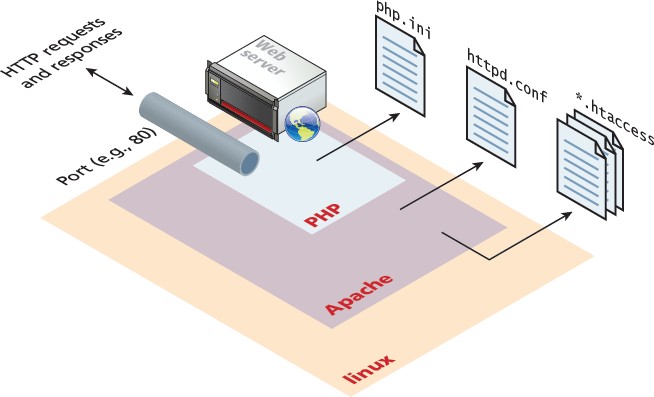


Figure 3.14 Linux, Apache, and PHP together

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 3.15 illustrates that when a request comes into Apache, each module is given an opportunity to handle some aspect of the request.

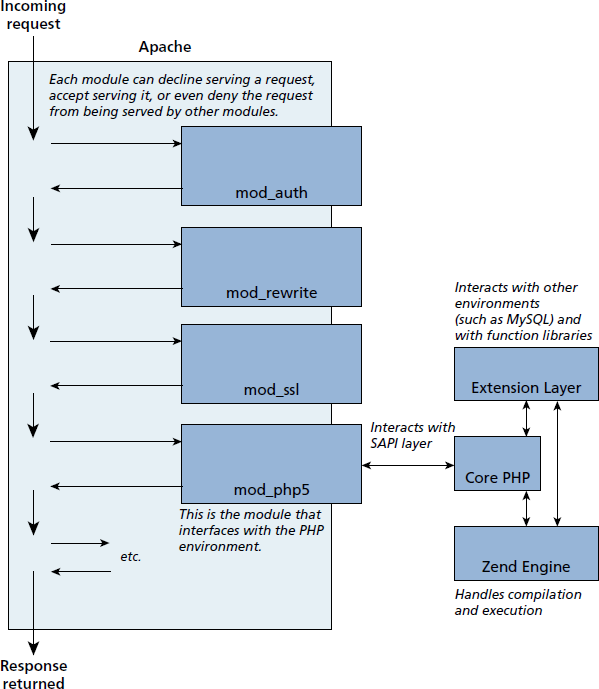


Figure 3.15 Apache modules and PHP

### 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.

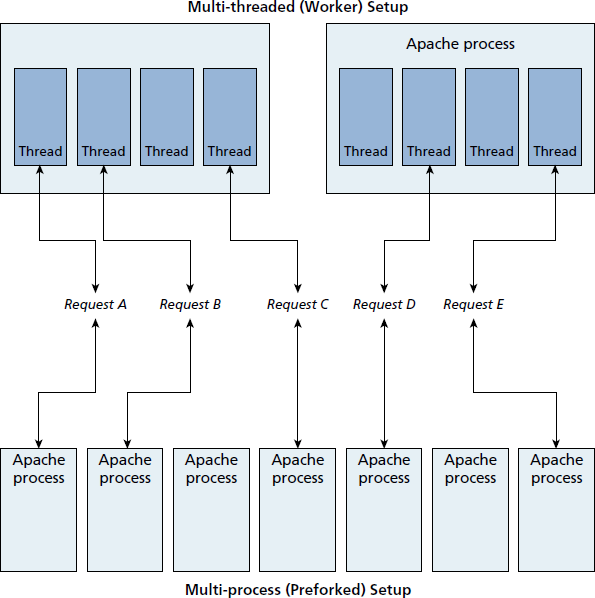
Apache runs in two possible modes: multi-process (also called preforked) or multi-threaded (also called worker), which are shown in Figure 3.16.

Figure 3.16 Multi-threaded versus multi-process

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 Unix based operating systems, but is slower on Windows-based operating systems.

**10. Explain two methods in JavaScript to access DOM nodes with examples.**

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 as illustrated in Figure 3.10.

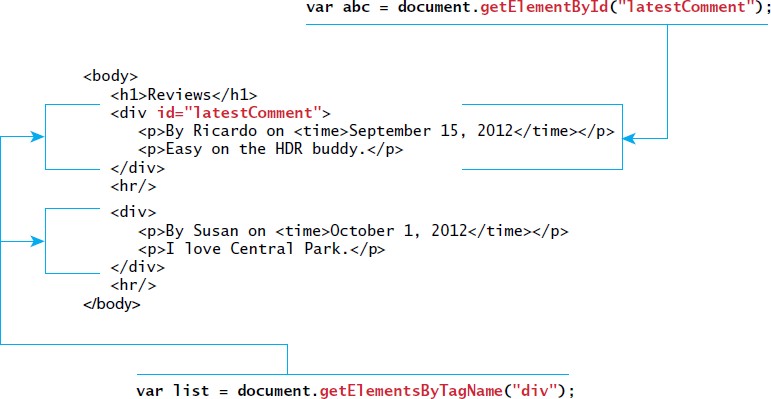


Figure 3.10 Relationship between HTML tags and getElementByID() and getElementsByTagName()

**11.Explain two approaches for event handling in Java Script with suitable code segment.**

Two new approaches for event handling are

1. Inline Event Handler Approach
2. Listener Approach
3. **Inline Event Handler Approach**

JavaScript events allow the programmer to react to user interactions. In early web development, it made sense to weave code and HTML together and to this day, inline JavaScript calls are intuitive. 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>**

In this example the HTML attribute onclick is used to attach a handler to that event. When the user clicks the <div>, the event is triggered and the alert is executed.

1. **Listener Approach**

The problem with the inline handler approach is that it does not make use of layers; that is, it does not separate content from behavior.

For this reason, this book will advocate and use an approach that separates all JavaScript code from the HTML markup.

The inline approach using one of the two approaches shown

1. The “old” style of registering a listener.

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

1. The “new” DOM2 approach to registering listeners.

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

**// IE 8**

**greetingBox.attachEvent('click', alert('Good Morning'))**