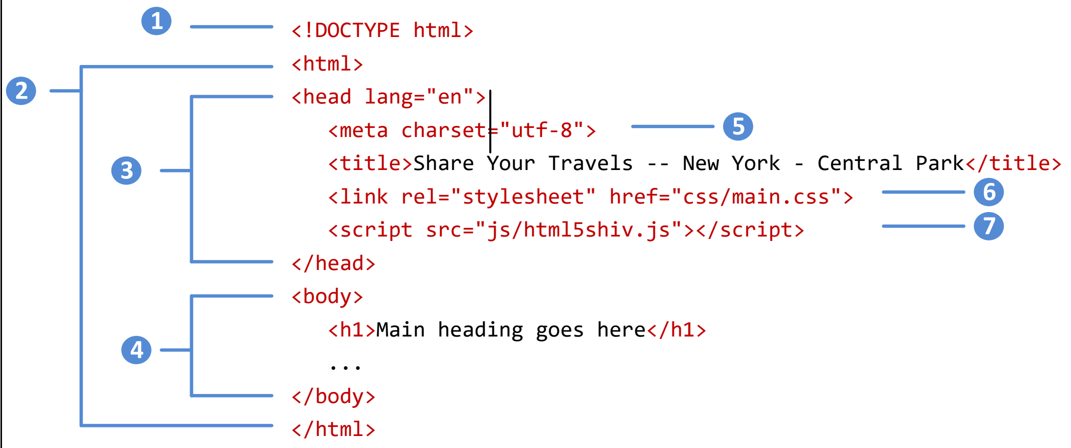
**Semester End Examination – July/August 2022**
**Scheme and Solution**
**Faculty- Prof. Aparna N /Dr. Anand R/Prof. Manjula S**

| Sub: | **Web Technology and Its Applications** | | | | Sub Code: | 18CS63 | Branch: | ISE |
|---|---|---|---|---|---|---|---|---|
| Date: | 02/08/23 | Duration: | 3hrs | Max Marks: | 100 | Sem/Sec: | III A, B & C | |

MARKS

1 Explain the structure of HTML documents with example. — 10(3-Example+explanation)



```
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="css/main.css">
   <script src="js/html5shiv.js"></script>
</head>
<body>
   <h1>Main heading goes here</h1>
   ...
</body>
</html>
```

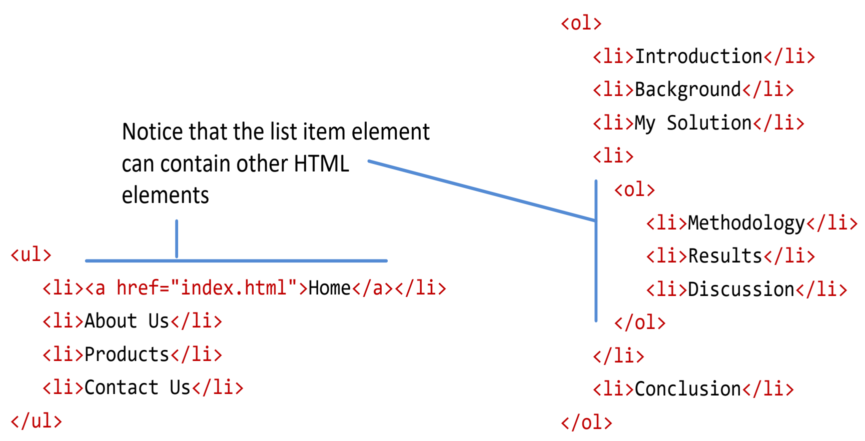
- Doctype: Tells the browser (or any other client software that is reading this HTML document) what type of document it is about to process.
- Notice that it does not indicate what version of HTML is contained within the document: it only specifies that it contains HTML.
- HTML5 does not require the use of the <html>, <head>, and <body>.
- However, in XHTML they were required, and most web authors continue to use them.
- The <html> element is sometimes called the **root element** as it contains all the other HTML elements in the document.
- HTML pages are divided into two sections: the **head** and the **body**, which correspond to the <head> and <body> elements.
- The head contains descriptive elements *about* the document
- The body contains content that will be displayed by the browser.
- You will notice that the <head> element contains a variety of additional elements.
- The first of these is the <meta> element. Our example declares that the character encoding for the document is UTF-8.
- Our example specifies an external CSS style sheet file that is used with this document.
- It also references an external JavaScript file.

1b **Explain ordered and unordered list in HTML5 with code.** — 10(Explnation+Code)

**Unordered lists**. Collections of items in no particular order; these are by default rendered by the browser as a bulleted list.

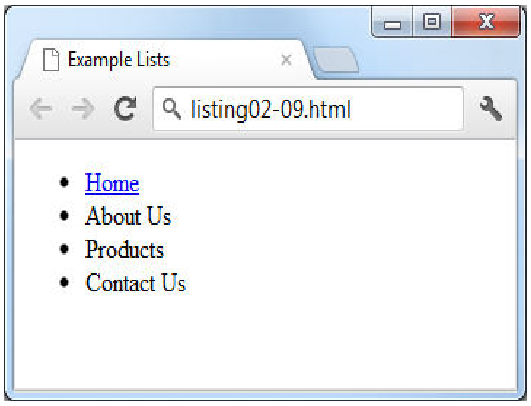
**Ordered lists**. Collections of items that have a set order; these are by default rendered by the browser as a numbered list.

**Definition lists**. Collection of name and definition pairs. These tend to be used infrequently. Perhaps the most common example would be a FAQ list.

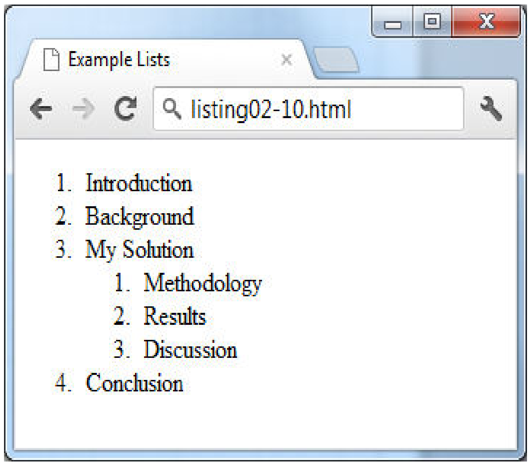

Notice that the list item element can contain other HTML elements

```
<ul>
   <li><a href="index.html">Home</a></li>
   <li>About Us</li>
   <li>Products</li>
   <li>Contact Us</li>
</ul>
```

```
<ol>
   <li>Introduction</li>
   <li>Background</li>
   <li>My Solution</li>
   <li>
     <ol>
        <li>Methodology</li>
        <li>Results</li>
        <li>Discussion</li>
     </ol>
   </li>
   <li>Conclusion</li>
</ol>
```



- Home
- About Us
- Products
- Contact Us



1. Introduction
2. Background
3. My Solution
   1. Methodology
   2. Results
   3. Discussion
4. Conclusion

2a **Explain Emdedded and external style sheet with code** 10(5 M each)

```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <style>
      h1 { font-size: 24pt; }
      h2 {
       font-size: 18pt;
       font-weight: bold;
      }
   </style>
</head>
<body>
   <h1>Share Your Travels</h1>
   <h2>New York - Central Park</h2>
   ...
```

LISTING 3.2 Embedded styles example

While better than inline styles, using embedded styles is also by and large discouraged.

Since each HTML document has its own <style> element, it is more difficult to consistently style multiple documents when using embedded styles.

```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="styles.css" />
</head>
```

**LISTING 3.3** Referencing an external style sheet

This is by far the most common place to locate style rules because it provides the best maintainability.

- When you make a change to an external style sheet, all HTML documents that reference that style sheet will automatically use the updated version.
- The browser is able to cache the external style sheet which can improve the performance of the site

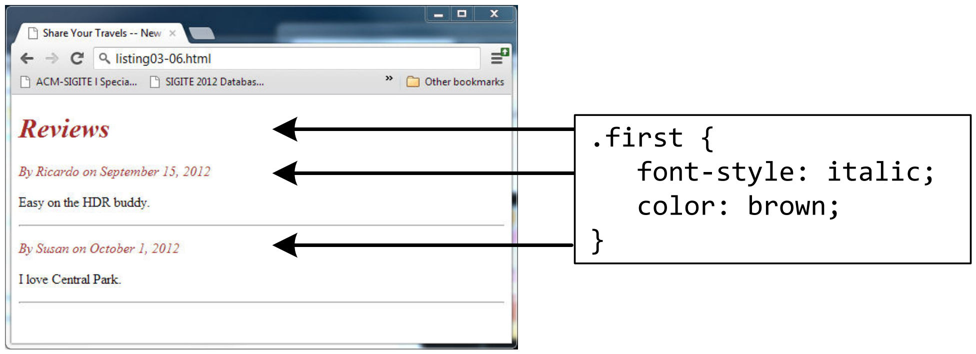
2b **Discuss class & Id selector with code** — 10 (5 M each)

A **class selector** allows you to simultaneously target different HTML elements regardless of their position in the document tree.

If a series of HTML element have been labeled with ***the same class attribute value***, then you can target them for styling by using a class selector, which takes the form: period (.) followed by the class name.

```
<head>
  <title>Share Your Travels </title>
          <style>
                    .first {
                              font-style: italic;
                              color: brown;
                    }
          </style>
</head>
<body>
  <h1 class="first">Reviews</h1>
  <div>
    <p class="first">By Ricardo on <time>September 15, 2012</time></p>
    <p>Easy on the HDR buddy.</p>
  </div>
  <hr/>

  <div>
    <p class="first">By Susan on <time>October 1, 2012</time></p>
    <p>I love Central Park.</p>
  </div>
  <hr/>
</body>
```



Reviews

By Ricardo on September 15, 2012

Easy on the HDR buddy.

By Susan on October 1, 2012

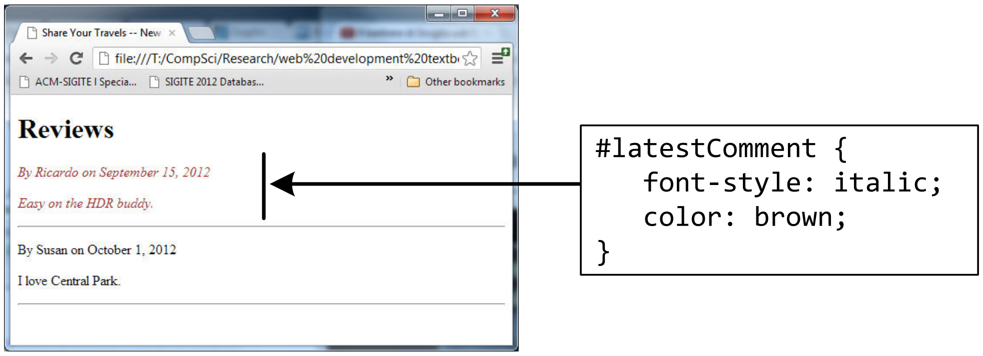
I love Central Park.

```
.first {
   font-style: italic;
   color: brown;
}
```

An **id selector** allows you to target a specific element by its id attribute regardless of its type or position.
If an HTML element has been labeled with an id attribute, then you can target it for styling by using an id selector, which takes the form: pound/hash (#) followed by the id name.

```
<head lang="en">
  <meta charset="utf-8">
  <title>Share Your Travels -- New York - Central Park</title>
        <style>
                #latestComment {
                        font-style: italic;
                        color: brown;
                }
        </style>
</head>
<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>
```


Share Your Travels -- New
file:///T:/CompSci/Research/web%20development%20textb
ACM-SIGITE | Specia...
SIGITE 2012 Databas...
Other bookmarks

**Reviews**

*By Ricardo on September 15, 2012*

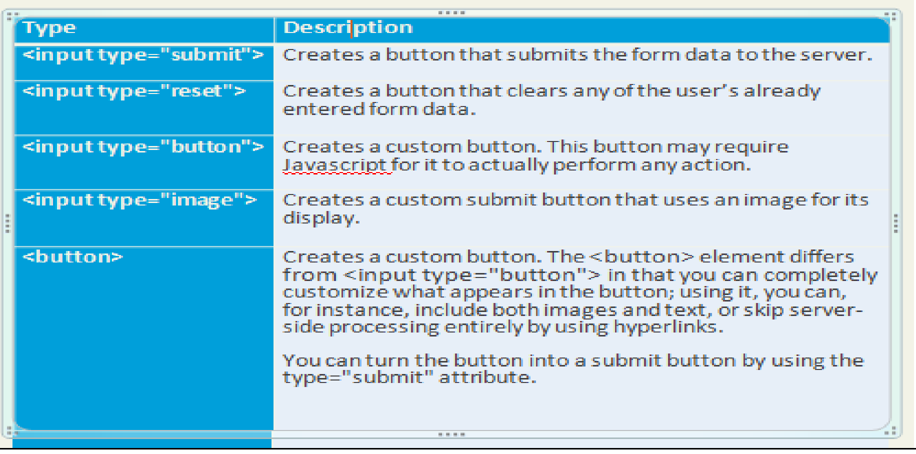
*Easy on the HDR buddy.*

By Susan on October 1, 2012

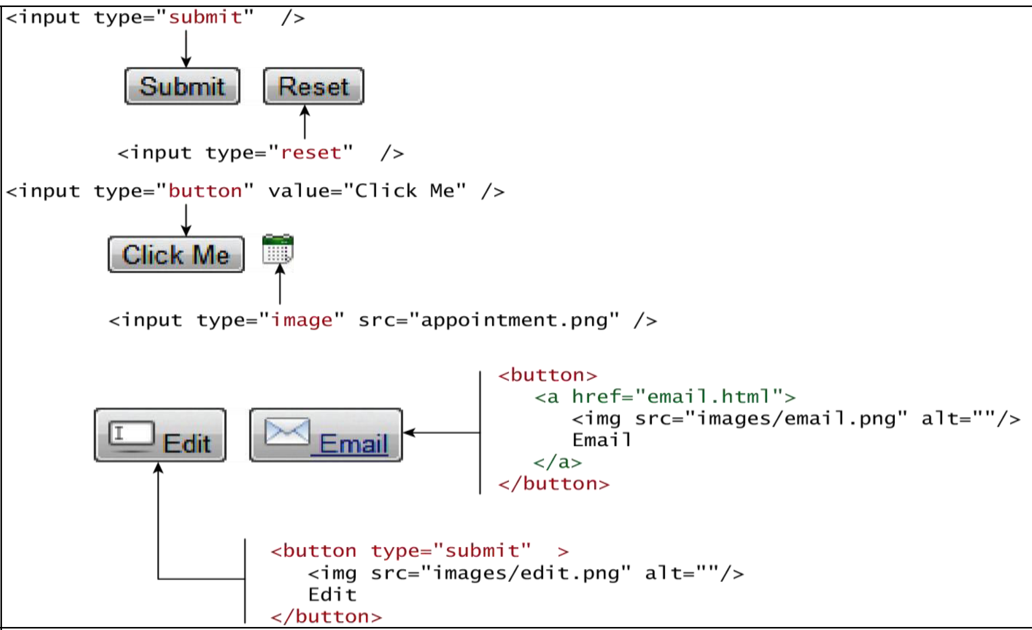
I love Central Park.

```
#latestComment {
   font-style: italic;
   color: brown;
}
```

3a **List & Describe different button controls** 8(list-1 M 7-explanation)



| Type | Description |
|---|---|
| `<input type="submit">` | Creates a button that submits the form data to the server. |
| `<input type="reset">` | Creates a button that clears any of the user's already entered form data. |
| `<input type="button">` | Creates a custom button. This button may require Javascript for it to actually perform any action. |
| `<input type="image">` | Creates a custom submit button that uses an image for its display. |
| `<button>` | Creates a custom button. The `<button>` element differs from `<input type="button">` in that you can completely customize what appears in the button; using it, you can, for instance, include both images and text, or skip server-side processing entirely by using hyperlinks. You can turn the button into a submit button by using the type="submit" attribute. |



```
<input type="submit"  />
<input type="reset"  />
<input type="button" value="Click Me" />
<input type="image" src="appointment.png" />
<button>
   <a href="email.html">
      <img src="images/email.png" alt=""/>
      Email
   </a>
</button>
<button type="submit"  >
   <img src="images/edit.png" alt=""/>
   Edit
</button>
```

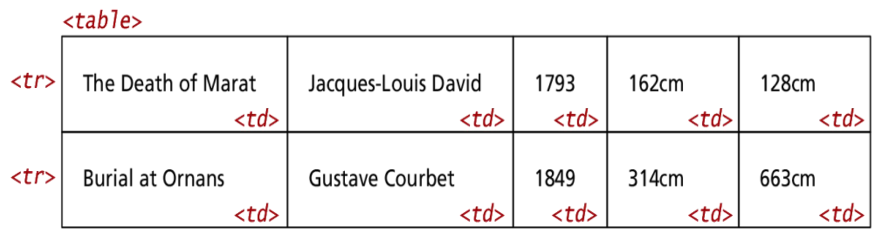
3b

**Explain the elements used to define the structure of an HTML table? Give example for table.**

A table in HTML is created using the **<table>** element
Tables can be used to display:

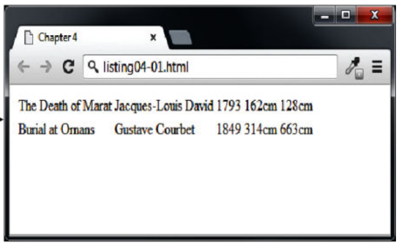
- Many types of content
  - Calendars, financial data, lists, etc…
- Any type of data
  - Images
  - Text
  - Links
  - Other tables
- an HTML **<table>** contains any number of rows (**<tr>**)
- each row contains any number of table data cells (**<td>**)
- Content goes inside of **<td></td>** tags

```
<table>
    <tr>
            <td>The Death of Marat</td>
    </tr>
</table>
```



| | | | | |
|---|---|---|---|---|
| The Death of Marat | Jacques-Louis David | 1793 | 162cm | 128cm |
| Burial at Ornans | Gustave Courbet | 1849 | 314cm | 663cm |

```
<table>
   <tr>
      <td>The Death of Marat</td>
      <td>Jacques-Louis David</td>
      <td>1793</td>
      <td>162cm</td>
      <td>128cm</td>
   </tr>
   <tr>
      <td>Burial at Ornans</td>
      <td>Gustave Courbet</td>
      <td>1849</td>
      <td>314cm</td>
      <td>663cm</td>
   </tr>
</table>
```



```
The Death of Marat Jacques-Louis David 1793 162cm 128cm
Burial at Ornans    Gustave Courbet     1849 314cm 663cm
```

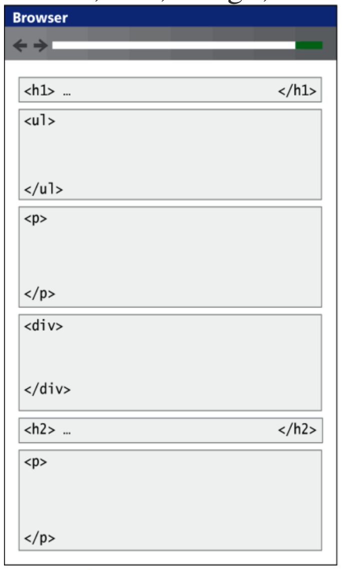
12(structure-6,example-6)

4a **What is normal flow in the context of CSS.** 10

**Normal flow** refers here to how the browser will normally display block-level elements and inline elements from left to right and from top to bottom

- **Block-level elements** are each contained on their own line
  - <p>, <div>, <h2>, <ul>, and <table>
- **Inline elements** do not form their own blocks but instead are displayed within lines

as <em>, <a>, <img>, and <span>

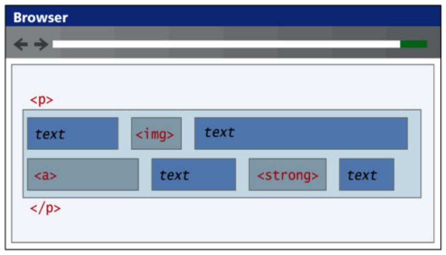


Each block exists on its own line and is displayed in normal flow from the browser window's top to its bottom.

By default each block-level element fills up the entire width of its parent (in this case, it is the <body>, which is equivalent to the width of the browser window).

You can use CSS box model properties to customize, for instance, the width of the box and the margin space between other block-level elements.

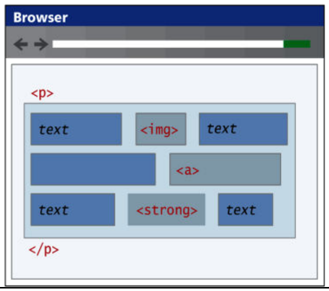
```
<p>
This photo <img src="photo-con.png" alt="" /> of Conservatory Pond in
<a href="http://www.centralpark.com/">Central Park</a> New York City
was taken on October 22, 2015 with a <strong>Canon EOS 30D</strong>
camera.
</p>
```



Inline content is laid out horizontally left to right within its container.

Once a line is filled with content, the next line will receive the remaining content, and so on.

Here the content of this <p> element is displayed on two lines.



If the browser window resizes, then inline content will be "reflowed" based on the new width.

Here the content of this <p> element is now displayed on three lines.

4b **What is responsive design? What are its 4 components, explain them.** 10(2 m each)

.In a **responsive design**, the page "responds" to changes in the browser size that go beyond the width scaling of a liquid layout.

One of the problems of a liquid layout is that images and horizontal navigation elements tend to take up a fixed size, and when the browser window shrinks to the size of a mobile browser, liquid layouts can become unusable. In a responsive layout, images will be scaled down and navigation elements will be replaced as the browser shrinks,

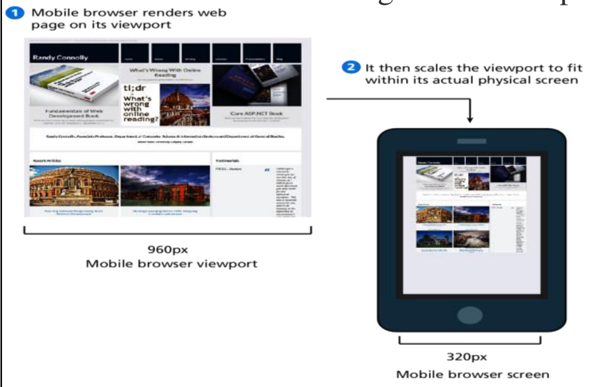
There are four key components that make responsive design work.

1. Liquid layouts
2. Scaling images to the viewport size
3. Setting viewports via the <meta> tag
4. Customizing the CSS for different viewports using media queries

Responsive designs begin with a liquid layout, that is, one in which most elements have their widths specified as percentages. Making images scale in size is actually quite straightforward, in that you simply need to specify the following rule:

```
img {
        max-width: 100%;
}
```

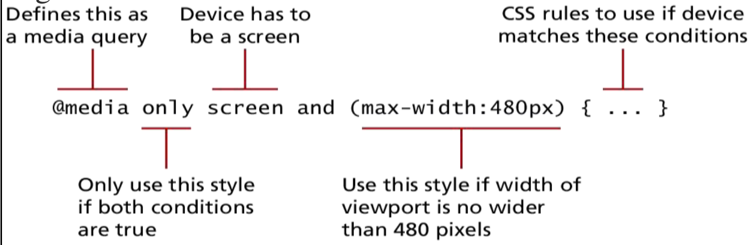
A key technique in creating responsive layouts makes use of the ability of current mobile browsers to shrink or grow the web page to fit the width of the screen.


1 Mobile browser renders web page on its viewport
2 It then scales the viewport to fit within its actual physical screen
960px
Mobile browser viewport
320px
Mobile browser screen

If the developer has created a responsive site 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

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

The other key component of responsive designs is **CSS media queries**. A media query is a way to apply style rules based on the medium that is displaying the file. You can use these queries to look at the capabilities of the device, and then define CSS rules to target that device.

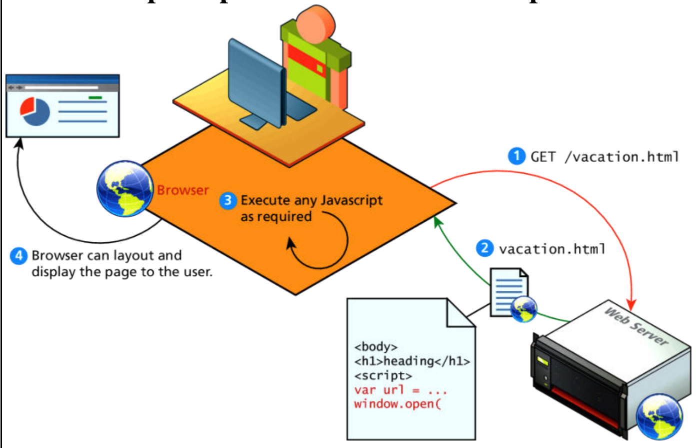


```
@media only screen and (max-width:480px) { ... }
```

Defines this as a media query
Device has to be a screen
CSS rules to use if device matches these conditions
Only use this style if both conditions are true
Use this style if width of viewport is no wider than 480 pixels

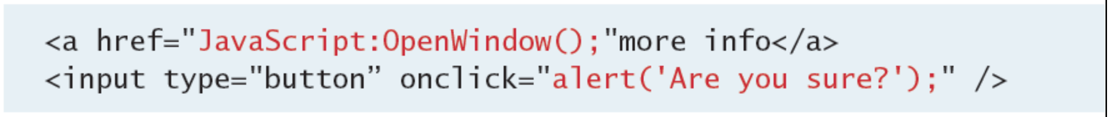
Contemporary responsive sites will typically provide CSS rules for phone displays first, then tablets, then desktop monitors, an approach called **progressive enhancement**, in which a design is adapted to progressively more advanced devices

5a **With example explain client & server script execution.** 10 (5 M each)


Browser
1 GET /vacation.html
2 vacation.html
3 Execute any Javascript as required
4 Browser can layout and display the page to the user.
Web Server

```
<body>
<h1>heading</h1>
<script>
var url = ...
window.open(
```

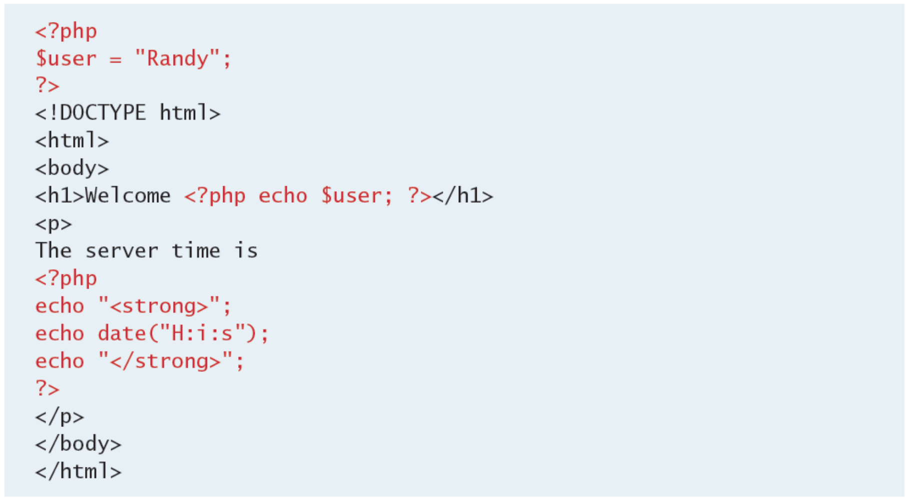
Inline JavaScript refers to the practice of including JavaScript code directly within certain HTML attributes



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

LISTING 6.1 Inline JavaScript example

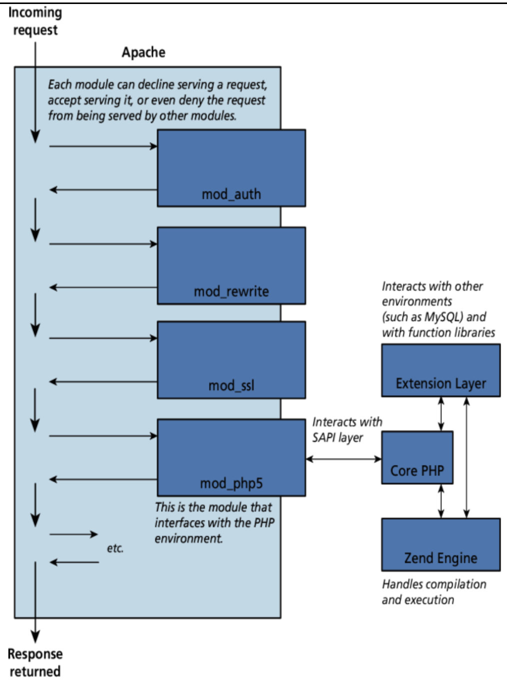
Server-side development is much more than web hosting: it involves the use of a programming technology like PHP or ASP.NET to create scripts that dynamically generate content



```
<?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

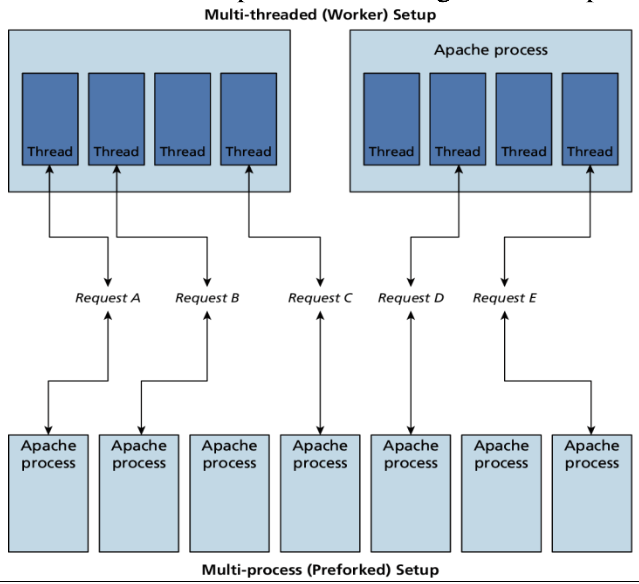
| | | |
|---|---|---|
| 5b | **Explain the Php module in the apache & describe difference between multi threaded & multiprocess setup.** There are 3 main modules 1. PHP core. The Core module defines the main features of the PHP environment, including essential functions for variable handling, arrays, 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. | 10 (5 m each) |


Incoming request
Apache
Each module can decline serving a request, accept serving it, or even deny the request from being served by other modules.
mod_auth
mod_rewrite
mod_ssl
mod_php5
This is the module that interfaces with the PHP environment.
etc.
Response returned
Interacts with other environments (such as MySQL) and with function libraries
Extension Layer
Interacts with SAPI layer
Core PHP
Zend Engine
Handles compilation and execution

Apache runs in two possible modes:

- multi-process (also called preforked). 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.
    - key advantage of multi-processing mode is that each process is insulated from other processes;
- multi-threaded (also called worker) a smaller number of Apache processes are forked. Each of the processes runs multiple threads. A thread is like a light-weight 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.
    - 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.

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



| | | |
|---|---|---|
| 6a | List of Comparison Operators in JavaScript? | 12(4 M each) |

The following is a list of the comparison operators supported by JavaScript:

| Operator | Name | Example | Return |
|---|---|---|---|
| == | Equal to | a==b | True o |
| | | a==3 | True o |
| | | a=="3" | True o |
| === | Equal value and types | a===b | True o |
| | | a===="3" | True o |
| != | Not equal to | a!=b | True o |
| !== | Not equal to and types | a!==b | True o |
| | | a!=="b" | True o |
| > | Greater than | a>b | True o |
| < | Less than | a<b | True o |
| >= | Greater than or equal to | a>=b | True o |
| <= | Less than or equal to | a<=b | True o |

**Logical Operators in JavaScript:**

| Operator | Name | Example |
|---|---|---|
| && | Logical And | ( a< 5 && b>2) |
| \|\| | Logical Or | (a<5 \|\| b>2) |
| ! | Not | (a!=5) |

Ternary (Conditional) Operator

JavaScript also supports a conditional operator, known as the ternary operator, specifying whether the output is true or not based on a condition. It is represented by the **"?"** symbol.

Below is the syntax of the JavaScript ternary operator:

Variable_name = (condition) ? value1:value2

| | | |
|---|---|---|
| | | |
| 6b | What is DOM? Briefly explain the different types of nodes?<br>A6) The Document Object Model (DOM) is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects; that way, programming languages can interact with the page.<br>All items in the DOM are defined as nodes. There are many types of nodes, but there are three main ones that we work with most often:<br>• Element nodes<br>• Text nodes<br>• Comment nodes<br>When an HTML element is an item in the DOM, it is referred to as an element node. Any lone text outside of an element is a text node, and an HTML comment is a comment node. In addition to these three node types, the document itself is a document node, which is the root of all other nodes.<br>The DOM consists of a tree structure of nested nodes, which is often referred to as the DOM tree. You may be familiar with an ancestral family tree, which consists of parents, children, and siblings. The nodes in the DOM are also referred to as parents, children, and siblings, depending on their relation to other nodes. | 8Diagram-4) |
| | | |
| | Module 4 | |
| 7a | Describe polymorphism and data encapsulation in php?<br><br>A7 a) Polymorphism portrays an example in object-oriented programming where methods in various classes that do similar things should have a similar name. Polymorphism is essentially an OOP pattern that enables numerous classes with different functionalities to execute or share a commonInterface. The usefulness of polymorphism is code written in different classes doesn't have any effect which class it belongs because they are used in the same way. In order to ensure that the classes do implement the polymorphism guideline, we can pick between one of the two alternatives of either abstract classes or interfaces. So let's implement the polymorphism principle with the help of the interface.<br><br>Interface An interface is similar to a class except that it cannot contain code. An interface can define method names and arguments, but not the contents of the methods. Any classes executing an interface must execute all methods characterized by the interface. | 10M(4M – Diagram) |

```
<?php
  interface Machine {
    public function calcTask();
  }
  class Circle implements Machine {
    private $radius;
    public function __construct($radius){
      $this -> radius = $radius;
    }
    public function calcTask(){
        return $this -> radius * $this -> radius * pi();
      }
   }
   class Rectangle implements Machine {
      private $width;
      private $height;
      public function __construct($width, $height){
         $this -> width = $width;
         $this -> height = $height;
      }
      public function calcTask(){
         return $this -> width * $this -> height;
      }
   }
   $mycirc = new Circle(3);
   $myrect = new Rectangle(3,4);
   echo $mycirc->calcTask();
   echo $myrect->calcTask();
?>
```

Output:

```
28.274
12
```

Encapsulation

The wrapping up of data and methods into a single unit (called class) is known as encapsulation. Encapsulation is a protection mechanism for the data members and methods present inside the class. In the encapsulation technique, we are restricting the data members from access to outside world end-user.

In PHP, encapsulation utilized to make the code more secure and robust. Using encapsulation, we are hiding the real implementation of data from the user and also does not allow anyone to manipulate data members except by calling the desired operation.

```
<?php
   class ATM {
      private $custid;
      private $atmpin;
      public function PinChange($custid,$atmpin)  {
               ----------perform tasks-----
               }
      public function CheckBalance($custid,$atmpin){
               ----------perform tasks-----
               }
      public function miniStatement($custid)  {
               ----------perform tasks-----
```

|  |  |  |
|---|---|---|
|  | ```<br>            }<br>        }<br>    $obj = new ATM();<br>    $obj ->CheckBalance(10005285637,1**3);<br>?><br>``` |  |
| 7b | Highlight the technique for reading and writing of files in php with example?<br><br>A7b) File handling is needed for any application. For some tasks to be done file needs to be processed. File handling in PHP is similar as file handling is done by using any programming language like C. PHP has many functions to work with normal files. Those functions are:<br><br>1) **fopen()** – PHP fopen() function is used to open a file. First parameter of fopen() contains name of the file which is to be opened and second parameter tells about mode in which file needs to be opened, e.g.,<br>`<?php`<br>`$file = fopen("demo.txt",'w');`<br>`?>`<br>Files can be opened in any of the following modes :<br>• **"w" –** Opens a file for write only. If file not exist then new file is created and if file already exists then contents of file is erased.<br>• **"r" –** File is opened for read only.<br>• **"a" –** File is opened for write only. File pointer points to end of file. Existing data in file is preserved.<br>• **"w+" –** Opens file for read and write. If file not exist then new file is created and if file already exists then contents of file is erased.<br>• **"r+" –** File is opened for read/write.<br>• **"a+" –** File is opened for write/read. File pointer points to end of file. Existing data in file is preserved. If file is not there then new file is created.<br>• **"x" –** New file is created for write only.<br><br>**fread() —**– After file is opened using fopen() the contents of data are read using fread(). It takes two arguments. One is file pointer and another is file size in bytes, e.g.,<br><br>`<?php`<br>`$filename = "demo.txt";`<br>`$file = fopen( $filename, 'r' );`<br>`$size = filesize( $filename );`<br>`$filedata = fread( $file, $size );`<br>`?>`<br><br>**fwrite()** – New file can be created or text can be appended to an existing file using fwrite() function. Arguments for fwrite () function are file pointer and text that is to written to file. It can contain optional third argument where length of text to written is specified, e.g.,<br><br>`<?php`<br>`$file = fopen("demo.txt", 'w');` | 10 M(Diagram-5M) |

|  |  |  |
|---|---|---|
|  | $text = "Hello world\n";<br>fwrite($file, $text);<br>?> |  |
| 8a | What are errors and exceptions?<br>A8a) Errors: Error are the **procedural approach**. The default error handling in PHP is very **simple.** An error message with filename, line number and a message describing the error is sent to the browser. This can be done using **PHP die()** Function. Errors are mostly caused by the environment in which program is running.<br><br>EXCEPTION: Exceptions are an **object-oriented approach** to programming. Exceptions are used to change the normal flow of a script if a specified error occurs. Basic Exception Handling using throw new Exception() in case of advance Exception handling you have to use **try** and **catch** method. Program itself is responsible for causing exceptions. | 10(Diagram-4) |
| 8b | Discuss $ Server array .How is it used?<br><br>**A8b)** PHP $_SERVER<br><br>$_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.<br><br>The example below shows how to use some of the elements in $_SERVER:<br><br>`<!DOCTYPE html>`<br>`<html>`<br>`<body>`<br>`<?php`<br>`echo $_SERVER['PHP_SELF'];`<br>`echo "<br>";`<br>`echo $_SERVER['SERVER_NAME'];`<br>`echo "<br>";`<br>`echo $_SERVER['HTTP_HOST'];`<br>`echo "<br>";`<br>`echo $_SERVER['HTTP_REFERER'];`<br>`echo "<br>";`<br>`echo $_SERVER['HTTP_USER_AGENT'];`<br>`echo "<br>";`<br>`echo $_SERVER['SCRIPT_NAME'];`<br>`?>`<br>`</body>`<br>`</html>` | 10(Diagram-4) |
|  | Module 5 |  |
| 9a | What are cookie? Explain php mechanism for writing and reading cookies with examples?<br><br>A9a) A **cookie** in PHP is a small file with a maximum size of 4KB that the web | 10((Example-4) |

server stores on the client computer. They are typically used to keep track of information such as a username that the site can retrieve to personalize the page when the user visits the website next time. A cookie can only be read from the domain that it has been issued from. Cookies are usually set in an HTTP header but JavaScript can also set a cookie directly on a browser.

**Setting Cookie In PHP**: To set a cookie in PHP, the **setcookie()** function is used. The setcookie() function needs to be called prior to any output generated by the script otherwise the cookie will not be set.

**Syntax:**

setcookie(name, value, expire, path, domain, security);

**Parameters:** The setcookie() function requires six arguments in general which are:

- **Name:** It is used to set the name of the cookie.
- **Value:** It is used to set the value of the cookie.
- **Expire:** It is used to set the expiry timestamp of the cookie after which the cookie can’t be accessed.
- **Path:** It is used to specify the path on the server for which the cookie will be available.
- **Domain:** It is used to specify the domain for which the cookie is available.
- **Security:** It is used to indicate that the cookie should be sent only if a secure HTTPS connection exists.

Reading a Cookie in PHP

Given that you've gone to the trouble of writing a cookie it stands to reason you'll probably want to read it back at some point. This is achieved by accessing the $_COOKIE array. The $_COOKIE array is an associative array whereby the name of the cookie provides the index into the array to extract the corresponding value of the name/value pair For example we can obtain the value of our *userName* cookie as follows:

```
<?php
    echo 'Reading cookie<br>';

    echo 'userName = ' . $_COOKIE['userName'];
?>
```

The above script should generate the following output:

```
Reading cookie
userName = JohnW
```

| | | |
|---|---|---|
| 9b | Discuss strategies to caching web application?<br><br>The five most popular strategies to consider are cache-aside, read-through, write-through, write-back, and write-around. | 10(Diagram-3+example-4) |
| | | |

| | | |
|---|---|---|
| 10a | What is AJAX? Explain how asynchronous request is handled using UML diagram?<br><br>AJAX is a collection of web development techniques that creates asynchronous web applications *(i.e. it allows the program to be executed immediately whereas the synchronous code will block further execution of the remaining code until it finishes the current one)* on the client-side with the help of various web technologies (like HTML, Javascript, etc.). It is not a language or a framework but a set of rules. It lets web applications send and retrieve data from a server asynchronously without interfering with the display and behavior of the current page.<br><br>Sequence Diagram Notations –Asynchronous Messages – An asynchronous message does not wait for a reply from the receiver. The interaction moves forward irrespective of the receiver processing the previous message or not. We use a lined arrow head to represent an asynchronous message. | 10((Diagram-4) |
| 10b | Describe how XML is processed in JavaScript and php?<br><br>***A10b)DOM (Document Object Model) parser***<br>The DOM parser contains all the information of an XML document in the form of the XML document object, the API is very simple and it is used to performs read and write both operations, and it is implemented by a DOM parser and it can create like a tree structure and it is advantageous when we need to access the document in a random manner but it has the disadvantage that it is slower than the other parsers and it consumes more memory because the complete document it needs to load into the memory so that it required more memory. | 10(Diagram-4) |