CMR INSTITUTE OF TECHNOLOGY

Affiliated to VTU, Approved by AICTE, Accredited by NBA and NAAC with "A+" Grade

ITPL MAIN ROAD, BROOKFIELD, BENGALURU-560037, KARNATAKA, INDIA

Department of CSE

17CS71 – WEB TECHNOLOGY & ITS APPLICATIONS

VTU SOLUTION - 2021

MODULE - 1

1. a) What are the 3 aims of HTML5? (04 Marks)

   There are three main aims to HTML5:

   1. Specify unambiguously how browsers should deal with invalid markup.

   2. Provide an open, nonproprietary programming framework (via JavaScript) for creating rich web applications.

   3. Be backwards compatible with the existing web.

   While parts of the HTML5 are still being finalized, all of the major browser manufacturers have at least partially embraced HTML5. Certainly not all browsers and all versions support every feature of HTML5. This is in fact by design. HTML in HTML5 is now a living language: that is, it is a language that evolves and devel ops over time. As such, every browser will support a gradually increasing subset of HTML5 capabilities. In late September 2012, the W3C announced that they planned to have the main elements of the HTML5 specification moved to Recommendation status (i.e., the specification would be finalized in terms of features) by late 2014, and the less stable parts of HTML5 moved to HTML5.1 (with a tentative completion date of 2016).

b) Explain the need of cascade in CSS. Explain the 3 principles of cascade with suitable CSS script segments. (08 Marks)

The "Cascade" in CSS refers to how conflicting rules are handled. The visual metaphor behind the term **cascade** is that of a mountain stream progressing downstream over rocks (and not that of a popular dishwashing detergent). The downward movement of water down a cascade is meant to be analogous to how a given style rule will continue to take precedence with child elements (i.e., elements "below" in a document outline as shown in Figure 3.3).
CSS uses the following cascade principles to help it deal with conflicts: inheritance, specificity, and location.

3.5.1 Inheritance

Inheritance is the first of these cascading principles. Many (but not all) CSS properties affect not only themselves but their descendants as well. Font, color, list, and text properties (from Table 3.1) are inheritable; layout, sizing, border, background, and spacing properties are not.

Figures 3.9 and 3.10 illustrate CSS inheritance. In the first example, only some of the property rules are inherited for the <body> element. That is, only the body element (thankfully!) will have a thick green border and the 100-px margin; however, all the text in the other elements in the document will be in the Arial font and colored red.

In the second example in Figure 3.10, you can assume there is no longer the body styling but instead we have a single style rule that styles *all* the elements. The <p> and <time> elements within the inherit the bold font-weight property but not the margin or border styles. However, it is possible to tell elements to inherit properties that are normally not inheritable, as shown in Figure 3.11. In comparison to Figure 3.10, notice how the <p> elements nested within the elements now inherit the border and margins of their parent.

3.5.2 Specificity

Specificity is how the browser determines which style rule takes precedence when more than one style rule could be applied to the same element. In CSS, the more specific the selector, the more it takes precedence (i.e., overrides the previous definition).

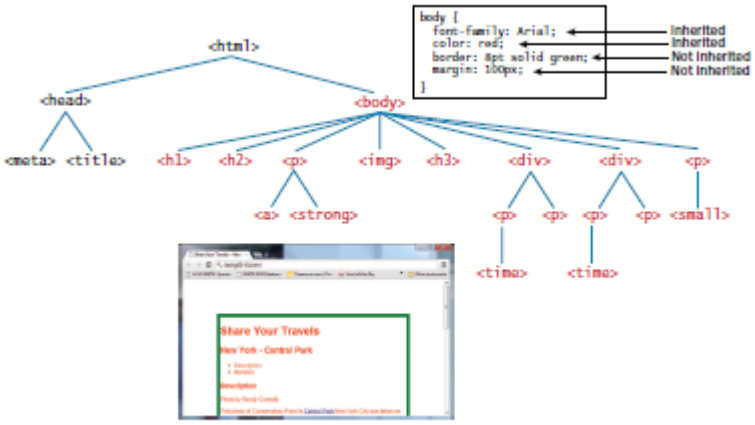


FIGURE 3.9 Inheritance

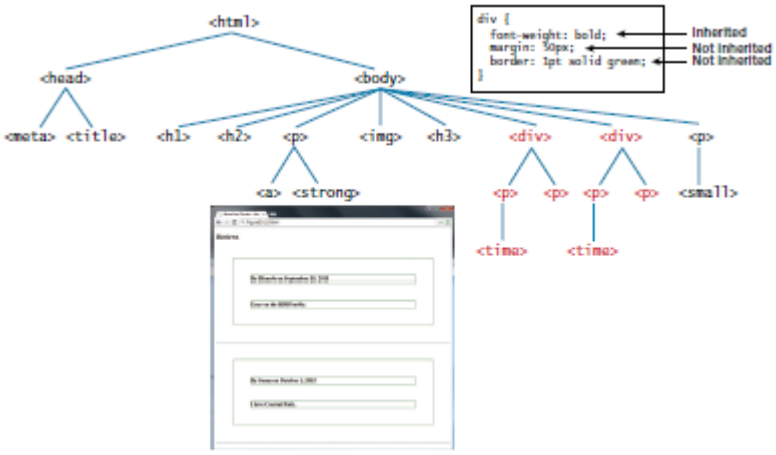


FIGURE 3.10 More Inheritance

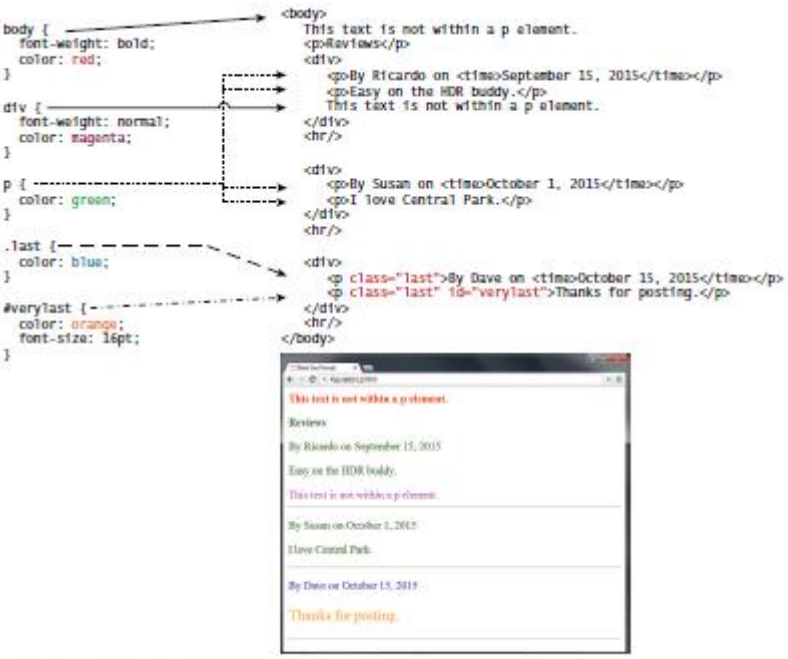


FIGURE 3.12 Specificity

As you can see in Figure 3.12, class selectors take precedence over element selectors, and id selectors take precedence over class selectors. The precise algorithm the browser is supposed to use to determine specificity is quite complex.6 A simplified version is shown in Figure 3.13.

3.5.3 Location

Finally, when inheritance and specificity cannot determine style precedence, the principle of **location** will be used. The principle of location is that when rules have the same specificity, then the latest are given more weight. For instance, an inline style will override one defined in an external author style sheet or an embedded style sheet.

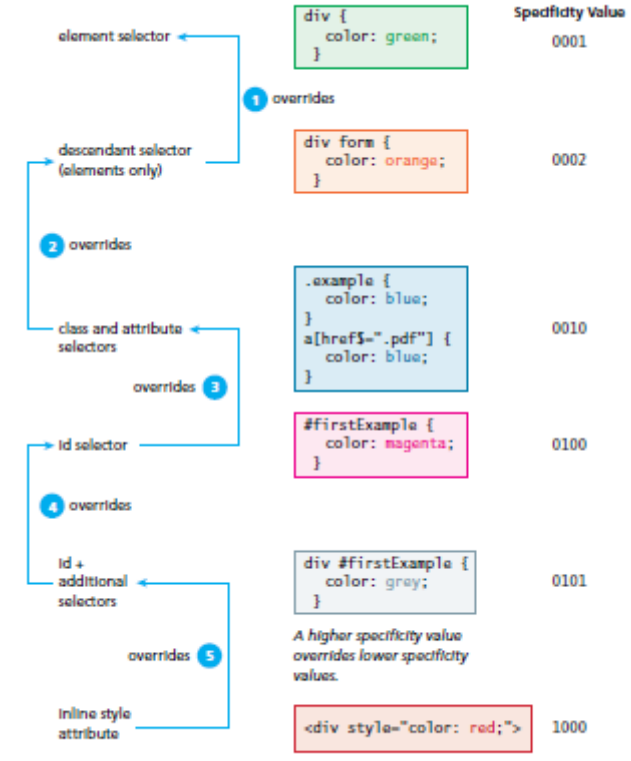


FIGURE 3.13 Specificity algorithm

c) Explain two types of URL referencing techniques with suitable scripts in HTML5. (08 Marks)

URL Relative Referencing

Whether we are constructing links with the <a> element, referencing images with the <img> element, or including external JavaScript or CSS files, we need to be able to successfully reference files within our site. This requires learning the syntax for so-called **relative referencing**. As you can see from Figure 2.16, when referencing a page or resource on an external site, a full **absolute reference** is required: that is, a complete URL as described in Chapter 1 with a protocol (typically, http://), the domain name, any paths, and then finally the file name of the desired resource.

However, when referencing a resource that is on the same server as your HTML document, you can use briefer relative referencing. If the URL does not include the "**http://**" then the browser will request the current server for the file. If all the resources for the site reside within the same **directory** (also referred to as a **folder**), then you can reference those other resources simply via their file name. However, most real-world sites contain too many files to put them all within a single directory. For these situations, a relative pathname is required along with the file name. The **pathname** tells the browser where to locate the file on the server. Pathnames on the web follow Unix conventions. Forward slashes ("/") are used to separate directory names from each other and from file names.Double periods ("**..**") are used to reference a directory "above" the current one in the directory tree.

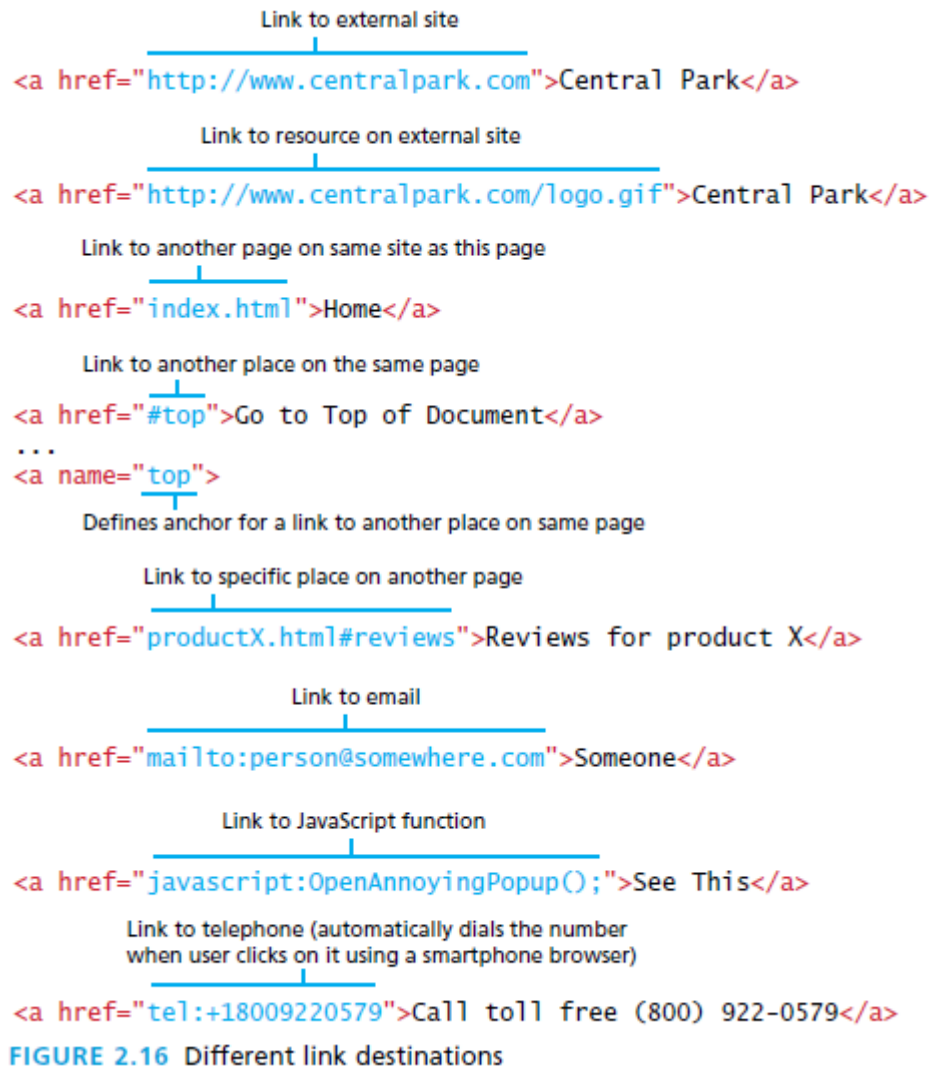


FIGURE 2.16 Different link destinations

OR

2. a) List and explain the different selectors available in CSS. (08 Marks)

1 Element Selectors

Element selectors select all instances of a given HTML element. You can select all elements by using the **universal element selector**, which is the * (asterisk) character. You can select a group of elements by separating the different element names with commas. This is a sensible way to reduce the size and complexity of your CSS files, by combining multiple identical rules into a single rule.

2 Class Selectors

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

3 Id Selectors

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.

4 Attribute Selectors

An **attribute selector** provides a way to select HTML elements either by the presence of an element attribute or by the value of an attribute. This can be a very powerful technique, but because of uneven support by some of the browsers, not all web authors have used them. Attribute selectors can be a very helpful technique in the styling of hyperlinks and images. For instance, perhaps we want to make it more obvious to the user when a pop-up tooltip is available for a link or image. We can do this by using the following attribute selector: [title] { … } This will match any element in the document that has a title attribute.

5 Pseudo-Element and Pseudo-Class Selectors

A **pseudo-element selector** is a way to select something that does not exist explicitly as an element in the HTML document tree but which is still a recognizable selectable object. For instance, you can select the first line or first letter of any HTML element using a pseudo-element selector. A **pseudo-class selector** does apply to an
HTML element, but targets either a particular state or, in CSS3, a variety of family relationships. The most common use of this type of selectors is for targeting link states. By default, the browser displays link text blue and visited text links purple. Do be aware that this state does not occur on touch screen devices. Note the syntax of pseudo-class selectors: the colon (:) followed by the pseudo-class selector name. Do be aware that a space is *not* allowed after the colon. Believe it or not, the order of these pseudo-class elements is important. The :link and :visited pseudo-classes should appear before the others. Some developers use a mnemonic to help them remember the order. My favorite is “Lord Vader, Former Handle Anakin” for Link, Visited, Focus, Hover, Active.

6 Contextual Selectors

A **contextual selector** (in CSS3 also called **combinators**) allows you to select elements based on their *ancestors*, *descendants*, or *siblings*. That is, it selects elements based on their context or their relation to other elements in the document tree. While some of these contextual selectors are used relatively infrequently, almost all
web authors find themselves using descendant selectors.A **descendant selector** matches all elements that are contained within another element. The character used to indicate descendant selection is the space character.

Descendant A specified element that is contained somewhere within another specified element. div p Selects a <p> element that is contained somewhere within a

 element. That is, the <p> can be any descendant, not just a child. **Child** A specified element that is a direct child of the specified element.

div>h2

Selects an <h2> element that is a child of a element.
Adjacent sibling A specified element that is the next sibling (i.e., comes directly after) of the specified element.

h3+p

Selects the first <p> after any <h3>. **General sibling** A specified element that shares the same parent as the specified element.

h3~p

Selects all the <p> elements that share the same parent as the <h3>.

b) Discuss the HTML5 semantic structure elements. (08 Marks)

header and Footer Most website pages have a recognizable header and footer section. Typically the header contains the site logo and title (and perhaps additional subtitles or taglines), horizontal navigation links, and perhaps one or two horizontal banners. The typical footer contains less important material, such as smaller text versions of the navigation, copyright notices, information about the site's privacy policy, and perhaps twitter feeds or links to other social sites.

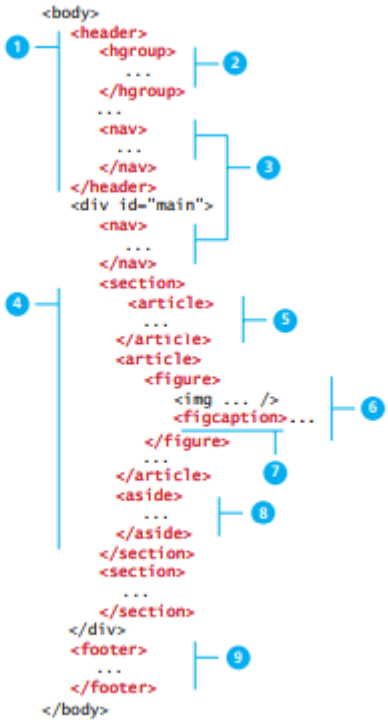


FIGURE 2.21 Sample layout using new HTML5 semantic structure elements

Heading Groups As mentioned in the previous section, it is not that unusual for a header to contain multiple headings in close proximity. The element can be used in such a circumstance to group them together within one container. The element can be used in contexts other than a header. For instance, one could also use an within an or a element as well.

c) List the different text properties with a description. (04 Marks)

Property	Description
`letter-spacing`	Adjusts the space between letters. Can be the value `normal` or a length unit.
`line-height`	Specifies the space between baselines (equivalent to leading in a desktop publishing program). The default value is `normal`, but can be set to any length unit. Can also be set via the shorthand `font` property.
`list-style-image`	Specifies the URL of an image to use as the marker for unordered lists.
`list-style-type`	Selects the marker type to use for ordered and unordered lists. Often set to none to remove markers when the list is a navigational menu or a input form.
`text-align`	Aligns the text horizontally in a container element in a similar way as a word processor. Possible values are `left`, `right`, `center`, and `justify`.
`text-decoration`	Specifies whether the text will have lines below, through, or over it. Possible values are: `none`, `underline`, `overline`, `line-through`, and `blink`. Hyperlinks by default have this property set to `underline`.
`text-direction`	Specifies the direction of the text, left-to-right (`ltr`) or right-to-left (`rtl`).
`text-indent`	Indents the first line of a paragraph by a specific amount.
`text-shadow`	A new CSS3 property that can be used to add a drop shadow to a text. Not yet supported in IE9.
`text-transform`	Changes the capitalization of text. Possible values are `none`, `capitalize`, `lowercase`, and `uppercase`.
`vertical-align`	Aligns the text vertically in a container element. Most common values are: `top`, `bottom`, and `middle`.
`word-spacing`	Adjusts the space between words. Can be the value `normal` or a length unit.

TABLE 3.10 Text Properties

MODULE - 2

3. a) Explain the different form widgets created with the <input> tag. (08 Marks)

text Creates a single-line text entry box.
<input type="text" name="title" />
textarea Creates a multiline text entry box. You can add content text or if using an HTML5 browser, placeholder text (hint text that disappears once user begins typing into the field).
<textarea rows="3" ... />
password Creates a single-line text entry box for a password (which masks the user entry as bullets or some other character)
A <form> element, which is a container for other elements that represent the various input elements within the form as well as plain text and almost any other HTML element.

Most forms need to gather text information from the user. Whether it is a search box, or a login form, or a user registration form, some type of text input is usually necessary. Table 4.3 lists the different text input controls.
While some of the HTML5 text elements are not uniformly supported by all browsers, they still work as regular text boxes in older browsers.

Type	Description
text	Creates a single-line text entry box. `<input type="text" name="title" />`
textarea	Creates a multiline text entry box. You can add content text or if using an HTML5 browser, placeholder text (hint text that disappears once user begins typing into the field). `<textarea rows="3" ... />`
password	Creates a single-line text entry box for a password (which masks the user entry as bullets or some other character) `<input type="password" ... />`
search	Creates a single-line text entry box suitable for a search string. This is an HTML5 element. Some browsers on some platforms will style search elements differently or will provide a clear field icon within the text box. `<input type="search" ... />`
email	Creates a single-line text entry box suitable for entering an email address. This is an HTML5 element. Some devices (such as the iPhone) will provide a specialized keyboard for this element. Some browsers will perform validation when form is submitted. `<input type="email" ... />`
tel	Creates a single-line text entry box suitable for entering a telephone. This is an HTML5 element. Since telephone numbers have different formats in different parts of the world, current browsers do not perform any special formatting or validation. Some devices may, however, provide a specialized keyboard for this element. `<input type="tel" ... />`
url	Creates a single-line text entry box suitable for entering a URL. This is an HTML5 element. Some devices may provide a specialized keyboard for this element. Some browsers also perform validation on submission. `<input type="url" ... />`

TABLE 4.3 Text Input Controls

FIGURE 4.16 Text input controls

b. Write HTML code for the table. (12 Marks)

Day	SEMINAR		
	SCHEDULE		TOPIC
	BEGIN	END	
MONDAY	8.00 am	5.00 pm	Introduction to XML
			Validity: DTD & NG
TUESDAY	11.00 am	2.00 pm	XPAT4
	11.00 am	2.00 pm	
	2.00 pm	5.00 pm	XSL Transformations
WEDNESDAY	8.00 am	5.00 pm	XSL Formatting Objects

```
<!DOCTYPE html>
<html>
<head>
    <title>Table Program</title>
```

```
</head>
<body>
    <table border="2">
        <tr>
            <th rowspan="3">DAY</th><th colspan="3">SEMINAR</th>
        </tr>
        <tr>
            <th colspan="2">SCHEDULE</th><th rowspan="2">TOPIC</th>
        </tr>
        <tr><th>BEGIN</th><th>END</th></tr>
        <tr>
            <td rowspan="2">Monday</td>
            <td rowspan="2">8.00 am</td>
            <td rowspan="2">5.00 pm</td>
            <td>Introduction to XML</td>
        </tr>
        <tr><td>Validity: DTD & NG</td></tr>
        <tr><td rowspan="3">TUESDAY</td><td>11.00 am</td><td>2.00 pm</td><td
rowspan="2">XPAT4</td></tr>
        <tr><td>11.00 pm</td><td>2.00 pm</td></tr>
        <tr><td>2.00 pm</td><td>5.00 pm</td><td>XSL Transformations</td></tr>
        <tr><td>WEDNESDAY</td><td>8.00 am</td><td>5.00 pm</td><td>XSL Formatting
Objects</td></tr>
    </table>
</body>
</html>
```

OUTPUT:

DAY	SEMINAR		
	SCHEDULE		TOPIC
	BEGIN	END	
Monday	8.00 am	5.00 pm	Introduction to XML
			Validity: DTD & NG
TUESDAY	11.00 am	2.00 pm	XPAT4
	11.00 pm	2.00 pm	
	2.00 pm	5.00 pm	XSL Transformations
WEDNESDAY	8.00 am	5.00 pm	XSL Formatting Objects

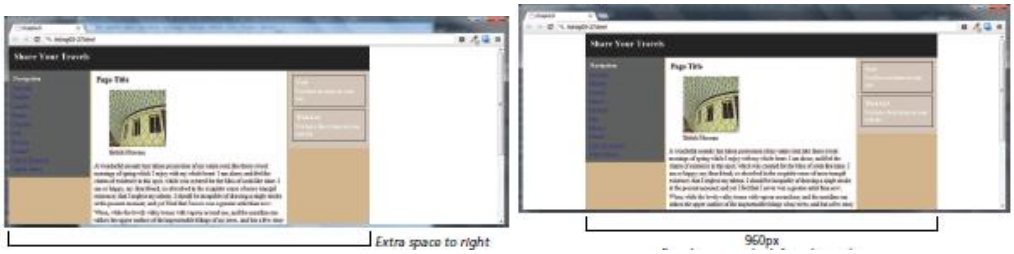
OR

4 a) Explain liquid layout design for websites with an example. List the fluid layout benefits and limitations. (08 Marks)

- In a **fixed layout**, the basic width of the design is set by the designer.
- A common width used is something in the 960 to 1000 pixel range, which fits nicely in the common desktop monitor resolution (1024 × 768).
- This content can then be positioned on the left or the center of the monitor.
- Fixed layouts are created using pixel units, typically with the entire content within a container whose width property set to some width.

```
<body>
<div id="wrapper">
<header>
...
</header>
<div id="main">
...
</div>
<footer>
...
</footer>
</div>
</body>
```

```
div#wrapper {
width: 960px;
background_color: tan;
}
```



The advantage of a fixed layout is that

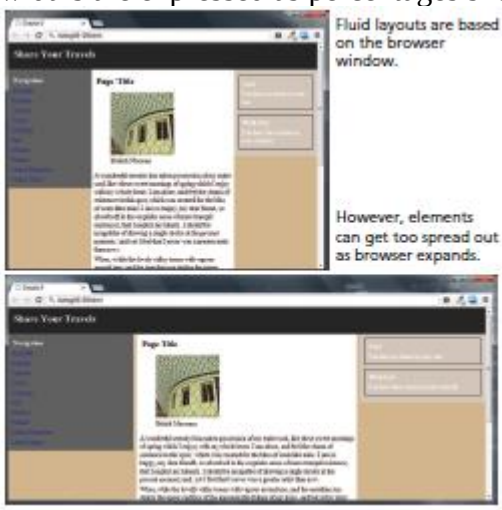
a) It is easier to produce and generally has a predictable visual result.
b) It is also optimized for typical desktop monitors.

Fixed layouts have drawbacks:

c) For larger screens, there may be an excessive amount of blank space to the left and/or right of the content.
d) Much worse is when the browser window shrinks below the fixed width; the user will have to horizontally scroll to see all the content.

A. Liquid layout:

- In this approach, widths are not specified using pixels, but percentage values.
- Percentage values in CSS are a percentage of the current browser width, so a layout in which all widths are expressed as percentages should adapt to any browser size.



Advantage of a liquid layout

- It adapts to different browser sizes, so there is neither wasted white space nor any need for horizontal scrolling.

Disadvantage of a liquid layout

- Liquid layouts can be more difficult to create because some elements, such as images, have fixed pixel sizes.
- Another problem will be noticeable as the screen grows or shrinks dramatically, in that the line length (which is an important contributing factor to readability) may become too long or too short.

b) Explain different ways of-positioning elements in CSS layout techniques. (08 Marks)

The position property is used to specify the type of positioning, and the possible values are shown in Table 5.1. The left, right, top, and bottom properties are used to indicate the distance the element will move; the effect of these properties varies depending upon the position property.

Relative Positioning

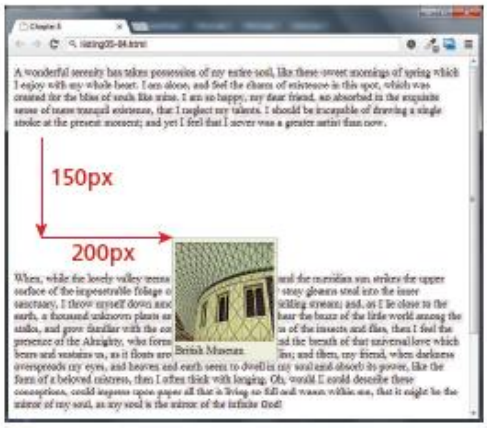
In **relative positioning** an element is displaced out of its normal flow position and moved relative to where it would have been placed. When an element is positioned relatively, it is displaced out of its normal flow position and moved relative to where it would have been placed. The other content 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.

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

```
<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

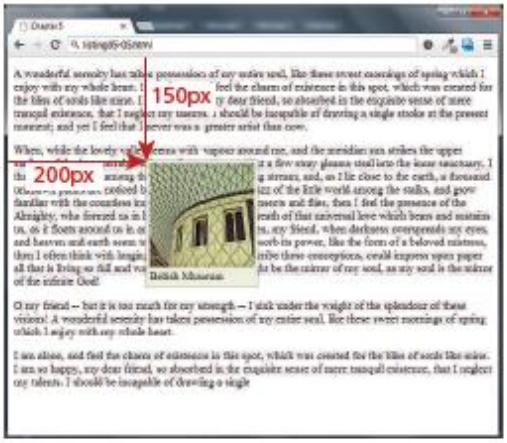
When an element is positioned absolutely, it is removed completely from normal flow. Thus, unlike with relative positioning, space is not left for the moved element, as it is no longer in the normal flow. Its position is moved in relation to its container block. In the example shown in Figure 5.5, the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

While this example is fairly clear, **absolute positioning** can get confusing. 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).

c) What are the importances of responsive design? Explain briefly. (04 Marks)

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 now several books devoted to responsive design, so this chapter can only provide a very brief overview of how it works.

There are four key components that make responsive design work.
They are:
1. Liquid layouts
2. Scaling images to the viewport size

3. Setting viewports via the 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%; }

Of course this does not change the downloaded size of the image; it only shrinks or expands its visual display to fit the size of the browser window, never expanding beyond its actual dimensions.
More sophisticated responsive designs will serve different sized images based on the viewport size.

MODULE – 3

5 a) Write a Javascript code that displays text "CORONA VIRUS" with increasing font size in the interval of 100 ms in blue color, when font size reaches 50 pt in teal color and should stop. (08 Marks)

```
<!DOCTYPE HTML>
<html>
<head>
<title>program</title>
</head>
<body>
<p id="demo"></p>
<script>
var var1 = setInterval(inTimer, 1000);
var fs = 5;
var ids = document.getElementById("demo");
function inTimer() {
ids.innerHTML = 'CORONA VIRUS; ids.setAttribute('style', "font-size: " + fs + "px;
color: red"); fs += 5;
if(fs >= 50 ){
clearInterval(var1);
var2 = setInterval(deTimer, 1000);
}
```

```
}
function deTimer() {
fs -= 5;
ids.innerHTML = 'TEXT SHRINKING'; ids.setAttribute('style', "font-size: " + fs +
"px; color: blue"); if(fs === 5 ){
clearInterval(var2);

}
}
</script>
</body>
</html>
```

b) Explain the advantages and disadvantages of client side scripting. (06 Marks)

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.

c) With suitable diagram, explain APACHE modules in PHP. (06 Marks)

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.

Some modules handle authorization, others handle URL rewriting, while others handle specific extensions.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).

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. 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. 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

OR

6 a) With suitable code segment, explain 2 approaches for event handling in Java Script (08 Marks)

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
you might program:
Click for pop-up
Hands-On Exercises Lab 6 Exercise Handling JavaScript Events6.7 JavaScript Events 269 In this example the HTML attribute onclick is used to attach a handler to that event. When the user clicks the
, the event is triggered and the alert is executed. The problem with this type of programming is that the HTML markup and the cor responding JavaScript logic are woven together. This reduces the ability of designers to work separately from programmers, and generally complicates maintenance of applications. The better way to program this is to remove the JavaScript from the HTML. Note Formally, we use an event handler to react to an event. Event handlers are simply methods that are designed explicitly for responding to particular events. If no response to an event is defined, the event might be passed up to another object for handling.

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. Although the book and its labs may occa sionally illustrate a quick concept with the old-style inline handler approach.

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'));

The approach shown in Listing 6.10 is widely supported by all browsers.
The first line in the listing creates a temporary variable for the HTML element that will trigger the event. The next line attaches the element’s onclick event to the event handler, which invokes the JavaScript alert() method (and thus annoys the user with a pop-up hello message).

The main advantage of this approach is that this code can be written anywhere, including an external file that helps uncouple the HTML from the JavaScript. However, the one limitation with this approach (and the inline approach) is that only one handler can respond to any given element event. listing 6.10 The “old” style of registering a listener.

var greetingBox = document.getElementById('example1');

greetingBox.onclick = alert('Good Morning');

The use of addEventListener() shown in Listing 6.11 was introduced in DOM Version 2, and as such is unfortunately not supported by IE 8 or earlier. This approach has all the other advantages of the approach shown in Listing 6.10, and has the additional advantage that multiple handlers can be assigned to a single object’s event.6.7 JavaScript Events 271 The examples in Listing 6.10 and Listing 6.11 simply used the built-in JavaScript alert() function.

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. listing 6.12

Listening to an event with a function

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);

b) Write PHP program to greet the user based on time (08 Marks)

<html>

<head>
<title>program</title>

</head>

<body action=”call.php” method=”POST”>

Enter the user name:

<input type=”text” name=”na”>

</body>

</html>

Call.php

<?php

$name=$_POST[‘name’];

$time=date(‘H’);

If($time>0 && &time<10)

Echo “Good morning”.$name;

Else if($time>10 && &time<13)

Echo “Good afternoon”.$name;

Else if($time>13 && &time<18)

Echo “Good evening”.$name;

Else

Echo “Good night”.$name;

?>

c) Explain 2 methods in Java Script to access DOM nodes with examples. (04 Marks)

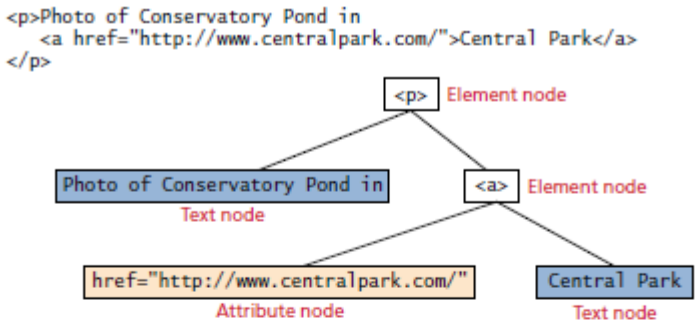


FIGURE 6.18 DOM nodes

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

TABLE 6.3 Some Essential Node Object Properties

MODULE - 4

7 a) List and Explain different super global arrays. (08 Marks)

PHP uses special predefined associative arrays called superglobal variables that allow the programmer to easily access HTTP headers, query string parameters, and other commonly needed information. They are called superglobal because these arrays are always in scope and always exist, ready for the programmer to access or modify them without having to use the global keyword. Name Description $GLOBALS Array for storing data that needs superglobal scope $_COOKIES Array of cookie data passed to page via HTTP request $_ENV Array of server environment data $_FILES Array of file items uploaded to the server $_GET Array of query string data passed to the server via the URL $_POST Array of query string data passed to the server via the HTTP header $_REQUEST Array containing the contents of $_GET, $_POST, and $_COOKIES $_SESSION Array that contains session data $_SERVER Array containing information about the request and the server 4.2.1 $_GET and $_POST Super global Arrays The $_GET and $_POST arrays are the most important super global variables in PHP since they allow the programmer to access data sent by the client in a query string.An HTML form (or an HTML link) allows a client to send data to the server. That data is formatted such that each value is associated with a name defined in the form. If the form was submitted using an HTTP GET request, then the resulting URL will contain the data in the query string. PHP will populate the superglobal $_GET array using the contents of this query

string in the URL. If the form was sent using HTTP POST, then the values would not be visible in the URL, but will be sent through HTTP POST request body. From the PHP programmer's perspective, almost nothing changes from a GET data post except that those values and keys are now stored in the $_POST array. Determining If Any Data Sent PHP that you will use the same file to handle both the display of a form as well as the form input. For example, a single file is often used to display a login form to the user, and that same file also handles the processing of the submitted form data, as shown in Figure 9.8. In such cases you may want to know whether any form data was submitted at all using either POST or GET. In PHP, there are several techniques to accomplish this task. First, you can determine if you are responding to a POST or GET by checking the $_SERVER['REQUEST_METHOD'] variable. To check if any of the fields are set. To do this you can use the isset() function in PHP to see if there is anything set for a particular query string parameter.

Some page that has a login form

Name
Pass
Submit

Accessing Form Array Data Sometimes in HTML forms you might have multiple values associated with a single Name.
Please select days of the week you are free.
Monday
Tuesday
Wednesday
Thursday
Friday
Submit

Unfortunately, if the user selects more than one day and submits the form, the $_GET['day'] value in the superglobal array will only contain the last value from the list that was selected. To overcome this limitation, you must change the HTML in the form. In particular, you will have to change the name attribute for each checkbox from day to day[]. Monday Tuesday After making this change in the HTML, the corresponding variable $_GET['day'] will now have a value that is of type array. PHP code to display an array of checkbox variables Using Query Strings in Hyperlinks It is possible to combine query strings with anchor tags . . . the answer is YES! Anchor tags (i.e., hyperlinks) also use the HTTP GET method.it is extraordinarily common in web development to programmatically construct the URLs for a series of links from, for instance, database data.Imagine a web page in which we are displaying a list of book links. One approach would be to have a separate page for each book (as shown in Figure 9.9). This is not a very sensible approach. Our database may have hundreds or thousands of books in it: surely it would be too much work to create a separate page for each book! It would make a lot more sense to have a single Display Book page that receives as input a query string that specifies which book to display, as shown in Figure 9.10. Notice that we typically pass some type of unique identifier in the query string. Sanitizing Query Strings The process of checking user input for incorrect or missing information is sometimes referred to as the process of sanitizing user inputs. // This uses a database API . . . we will learn about it in Chapter 11 $pid = mysqli_real_escape_string($link, $_GET['id']); if (is_int($pid)) { //

Continue processing as normal } else { // Error detected. Possibly a malicious user }Program must be able to handle the following cases for every query string or form value ■ If query string parameter doesn't exist. ■ If query string parameter doesn't contain a value. ■ If query string parameter value isn't the correct type. ■ If value is required for a database lookup, but provided value doesn't exist in the database table. What should we do when an error occurs in Listing 9.9? There are a variety of possibilities; Chapter 12 will examine the issue of exception and error handling in more detail. For now, we might simply redirect to a generic error handling page using the header directive, for instance: header("Location: error.php"); exit(); 4.3 $_SERVER Arrays The $_SERVER associative array contains a variety of information. It contains some of the information contained within HTTP request headers sent by the client. echo $_SERVER["SERVER_NAME"] . "
"; echo $_SERVER["SERVER_SOFTWARE"] . "
"; echo $_SERVER["REMOTE_ADDR"] . "
"; Server Information Keys λ SERVER_NAME is a key in the $_SERVER array that contains the name of the site that was requested. If you are running multiple hosts on the same code base, this can be a useful piece of information. λ SERVER_ADDR is a complementary key telling us the IP of the server. Either of these keys can be used in a conditional to output extra HTML to identify a development server. λ DOCUMENT_ROOT tells us the file location from which you are currently running your script. λ SCRIPT_NAME key that identifies the actual script being executed. Request Header Information Keys These keys provide programmatic access to the data in the request header. The REQUEST_METHOD key returns the request method that was used to access the page: that is, GET, HEAD, POST, PUT. REMOTE_ADDR key returns the IP address of the requestor, which can be a useful value to use in your web applications. HTTP_USER_AGENT contains the operating system and browser that the client is using. HTTP_REFERER contains the address of the page that referred us to this one (if any) through a link. $previousPage = $_SERVER['HTTP_REFERER']; // Check to see if referer was our search page if (strpos("search.php",$previousPage) != 0) { echo "Back to search"; }

b) Explain the different error handling methods with suitable code segments. (08 Marks)

Reading/Writing Files There are two basic techniques for read/writing files in PHP:

■ Stream access. In this technique, our code will read just a small portion of the file at a time. While this does require more careful programming, it is the most memory-efficient approach when reading very large files.

■ All-In-Memory access. In this technique, we can read the entire file into memory (i.e., into a PHP variable). While not appropriate for large files, it does make processing of the file extremely easy. Stream Access functions like fopen(), fclose(), and fgets() from the C programming language, The function fopen() takes a file location or URL and access mode as parameters.

The returned value is a stream resource, which you can then read sequentially. Some of the common modes are "r" for read, "rw" for read and write, and "c," which creates a new file for writing. Once the file is opened, you can read from it in several ways. To read a single line, use the fgets() function, which will return false if there is no more data, and if it reads a line it will advance the stream forward to the next one so you can use the === check to see if you have reached the end of the file. To read an arbitrary amount of data (typically for binary files), use fread() and for reading a single character use fgetsc().

Finally, when finished processing the file you must close it using fclose(). Listing 9.19 illustrates a script using fopen(), fgets(), and fclose() to read a file and echo it out (replacing new lines with
tags).

```
$f = fopen("sample.txt", "r");

$ln = 0;

while ($line = fgets($f))

{

$ln++;

printf("%2d: ", $ln);

echo $line . "";

}

fclose($f);
```

To write data to a file, you can employ the fwrite() function in much the same way as fgets(), passing the file handle and the string to write.

In-Memory File Access While the previous approach to reading/writing files gives you complete control, the programming requires more care in dealing with the streams, file handles, and other low-level issues. The alternative simpler approach is much easier to use, at the cost of relinquishing fine-grained control. Function Description file() Reads the entire file into an array, with each array element corresponding to one line in the file file_get_contents Reads the entire file into a string variable file_put_contents Writes the contents of a string variable out to a file.

To read an entire file into a variable you can simply use:

```
$fileAsString = file_get_contents(FILENAME);
```

To write the contents of a string $writeme to a file, you use file_put_contents(FILENAME, $writeme);

let us imagine we have a comma-delimited text file that contains information about paintings, where each line in the file corresponds to a different painting: 01070,Picasso,The Actor,1904 01080,Picasso,Family of Saltimbanques,1905 02070,Matisse,The Red Madras Headdress,1907 05010,David,The Oath of the Horatii,1784 // read the file into memory; if there is an error then stop processing $paintings = file($filename) or die('ERROR: Cannot find file'); // our data is comma-delimited $delimiter = ','; // loop through each line of the file foreach ($paintings as $painting) { // returns an array of strings where each element in the array // corresponds to each substring between the delimiters

```
$paintingFields = explode($delimiter, $painting);

$id= $paintingFields[0];

$artist = $paintingFields[1]; $title = $paintingFields[2];

$year = $paintingFields[3]; // do something with this data . . . }
```

c) How do you read or write file on server from PHP? Give examples. (04 Marks)

Reading/Writing Files There are two basic techniques for read/writing files in PHP:

■ Stream access. In this technique, our code will read just a small portion of the file at a time. While this does require more careful programming, it is the most memory-efficient approach when reading very large files.

■ All-In-Memory access. In this technique, we can read the entire file into memory (i.e., into a PHP variable). While not appropriate for large files, it does make processing of the file extremely easy. Stream Access functions like fopen(), fclose(), and fgets() from the C programming language, The function fopen() takes a file location or URL and access mode as parameters. The returned value is a stream resource, which you can then read sequentially. Some of the common modes are “r” for read, “rw” for read and write, and “c,” which creates a new file for writing. Once the file is opened, you can read from it in several ways. To read a single line, use the fgets() function, which will return false if there is no more data, and if it reads a line it will advance the stream forward to the next one so you can use the === check to see if you have reached the end of the file. To read an arbitrary amount of data (typically for binary files), use fread() and for reading a single character use fgetsc(). Finally, when finished processing the file you must close it using fclose().

Listing 9.19 illustrates a script using fopen(), fgets(), and fclose() to read a file and echo it out (replacing new lines with tags). $f = fopen("sample.txt", "r"); $ln = 0; while ($line = fgets($f)) { $ln++; printf("%2d: ", $ln); echo $line . ""; } fclose($f); To write data to a file, you can employ the fwrite() function in much the same way as fgets(), passing the file handle and the string to write.In-Memory File Access While the previous approach to reading/writing files gives you complete control, the programming requires more care in dealing with the streams, file handles, and other low-level issues. The alternative simpler approach is much easier to use, at the cost of relinquishing fine-grained control. Function Description file() Reads the entire file into an array, with each array element corresponding to one line in the file file_get_contents Reads the entire file into a string variable file_put_contents Writes the contents of a string variable out to a file To read an entire file into a variable you can simply use: $fileAsString = file_get_contents(FILENAME); To write the contents of a string $writeme to a file, you use file_put_contents(FILENAME, $writeme); let us imagine we have a comma-delimited text file that contains information about paintings, where each line in the file corresponds to a different painting: 01070,Picasso,The Actor,1904 01080,Picasso,Family of Saltimbanques,1905 02070,Matisse,The Red Madras Headdress,1907 05010,David,The Oath of the Horatii,1784 // read the file into memory; if there is an error then stop processing $paintings = file($filename) or die('ERROR: Cannot find file'); // our data is comma-delimited $delimiter = ','; // loop through each line of the file foreach ($paintings as $painting) { // returns an array of strings where each element in the array // corresponds to each substring between the delimiters $paintingFields = explode($delimiter, $painting); $id= $paintingFields[0]; $artist = $paintingFields[1]; $title = $paintingFields[2]; $year = $paintingFields[3]; // do something with this data . . . }

OR

8 a) Write a php program to create a class Employee with the following specifications: (08 Marks)

Data members: Name, ID, Payment.

Member function: Read(getters) and write (setters) Use the above specification to read and print the information of 10 students.

Data members : Name, Roll number, Average marks

Member function : Read(getters) and write (setters)

Use the above specification to read and print the information of 2 students.

```
Class STUDENT{

Public $Name;

Public $Roll-number

Public $Average-marks

}

$student1=new STUDENT()

$student2=new STUDENT()
```

b) Explain the support for inheritance in PHP with UML class diagram. (06 Marks)

Inheritance

Along with encapsulation, inheritance is one of the three key concepts in object oriented design and programming Inheritance enables you to create new PHP classes that reuse, extend, and modify the behavior that is defined in another PHP class. Although some languages allow it, PHP only allows you to inherit from one class at a time.

A class that is inheriting from another class is said to be a subclass or a derived class. The class that is being inherited from is typically called a superclass or a base class. When a class inherits from another class, it inherits all of its public and protected methods and properties. Figure 10.9 illustrates how inheritance is shown in a UML class diagram. Just as in Java, a PHP class is defined as a subclass by using the extends keyword.

```
class Painting extends Art { . . . }
```

Referencing Base Class Members

As mentioned above, a subclass inherits the public and protected members of the base class. Thus in the following code based on Figure 10.9, both of the references will work because it is as if the base class public members are defined within the subclass.

```
$p = new Painting();
. . .
// these references are ok
echo $p->getName(); // defined in base class
echo $p->getMedium(); // defined in subclass
```

Unlike in languages like Java or C#, in PHP any reference to a member in the base class requires the addition of the parent:: prefix instead of the $this-> prefix. So within the Painting class, a reference to the getName() method would be:

```
parent::getName()
```

It is important to note that private members in the base class are not available to its subclasses. Thus, within the Painting class, a reference like the following would not work.

```
$abc = parent::name; // would not work within the Painting class
```

If you want a member to be available to subclasses but not anywhere else, you can use the protected access modifier, which is shown in Figure 10.10.

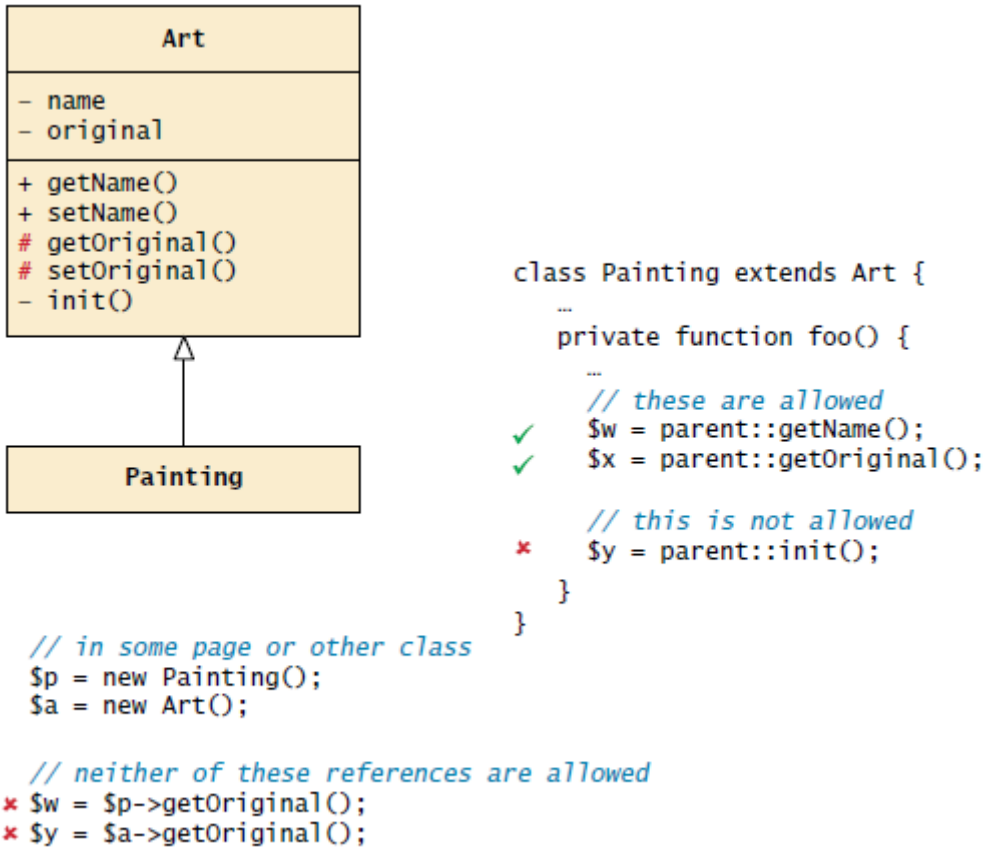


FIGURE 10.10 Protected access modifier

Inheriting Methods

Every method defined in the base/parent class can be overridden when extending a class, by declaring a function with the same name. A simple example of overriding can be found in Listing 10.8 in which each subclass overrides the __toString() method.

To access a public or protected method or property defined within a base class from within a subclass, you do so by refixing the member name with parent::. So to access the parent's __toString() method you would simply use parent::__toString().

c) Explain 3 approaches to restrict file size in file upload with suitable code segments. (06 Marks)

File Size Restrictions

Some scripts limit the file size of each upload. There are three main mechanisms for maintaining uploaded file size restrictions:

- HTML in the input form
- JavaScript in the input form
- PHP coding

HTML in the input form

This mechanism is to add a hidden input field before any other input fields in your HTML form with a name of MAX_FILE_SIZE. This technique allows your php.ini maximum file size to be large, while letting some forms override that largelimit with a smaller one.

```
<form enctype='multipart/form-data' method='post'>
<input type="hidden" name="MAX_FILE_SIZE" value="1000000" />
<input type='file' name='file1' />
<input type='submit' />
```

</form>

JavaScript in the input form

The more complete client-side mechanism to prevent a file from uploading if it is too big is to pre validate the form using JavaScript.

```
<script>
var file = document.getElementById('file1');
var max_size = document.getElementById("max_file_size").value;
if (file.files && file.files.length ==1){
if (file.files[0].size > max_size) {
alert("The file must be less than " + (max_size/1024) + "KB");
e.preventDefault();
}
}
</script>
```

PHP coding

This mechanism is to add a simple check on the server side.

```
$max_file_size = 10000000;
foreach($_FILES as $fileKey => $fileArray) {
if ($fileArray["size"] > $max_file_size) {
echo "Error: " . $fileKey . " is too big";
}
printf("%s is %.2f KB", $fileKey, $fileArray["size"]/1024);
}
```

MODULE - 5

9 a) Explain different types of caching used to improve performance of web application(08 Marks)

There are a number of different ways of managing session state in such a webfarm situation, some of which can be purchased from third parties. There are effectively two categories of solution to this problem. 1. Configure the load balancer to be “session aware” and relate all requests using a session to the same server. 2. Use a shared location to store sessions, either in a database, memcache (coveredin the next section), or some other shared session state mechanism as seen in Figure 13.12. Using a database to store sessions is something that can be done programmatically, but requires a rethinking of how sessions are used. Code that was written to work on a single server will have to be changed to work with sessions in a shared database, and therefore is cumbersome. The other alternative is to configure PHP to use memcache on a shared server (covered in Section 13.8). To do this you must have PHP compiled with memcache enabled; if not, you may need to install themodule. Once installed, you must change the php.ini on all servers to utilize a shared location, rather than local files as shown in Listing 13.7.listing 13.7 Configuration in php.ini to use a shared location for sessions [Session] ; Handler used to store/retrieve data. session.save_handler = memcache session.save_path = "tcp://sessionServer:11211"

b) With suitable PHP script, explain loading and processing an XML document in Java script(08 Marks)

XML Processing in JavaScript

All modern browsers have a built-in XML parser and their JavaScript implementations support an in-memory XML DOM API, which loads the entire document into memory where it is transformed into a hierarchical tree data structure. You can then use the already familiar DOM functions such as

getElementById(), getElementsByTagName(), and createElement() to access and manipulate the data. For instance, Listing 17.5 shows the code necessary for loading an XML document into an XML DOM object, and it displays the id attributes of the <painting> elements as well as the content of each painting's <title> element.

```
<script>
if (window.XMLHttpRequest) {
// code for IE7+, Firefox, Chrome, Opera, Safari
xmlhttp=new XMLHttpRequest();
}
else {
// code for old versions of IE (optional you might just decide to
// ignore these)
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}

// load the external XML file
xmlhttp.open("GET","art.xml",false);
xmlhttp.send();
xmlDoc=xmlhttp.responseXML;
// now extract a node list of all <painting> elements
paintings = xmlDoc.getElementsByTagName("painting");
if (paintings) {
// loop through each painting element
for (var i = 0; i < paintings.length; i++)
{
// display its id attribute
alert("id="+paintings[i].getAttribute("id"));
// find its <title> element
title = paintings[i].getElementsByTagName("title");
if (title) {
// display the text content of the <title> element
alert("title="+title[0].textContent);
}
}
}
</script>
```

c) Explain creating and reading cookies with suitable PHP scripts. (04 Marks)

Using Cookies Like any other web development technology, PHP provides mechanisms for writing and reading cookies. Cookies in PHP are created using the setcookie() function and are retrieved using the $_COOKIES superglobal associative array. Below example illustrates the writing of a persistent cookie in PHP The setcookie() function also supports several more parameters, which further customize the new cookie. You can examine the online official PHP documentation for more information. The below example illustrates the reading of cookie values. Notice that when we read a cookie, we must also check to ensure that the cookie exists. In PHP, if the cookie has expired (or never existed in the first place), then the client's browser would not send anything, and so the $_COOKIE array would be blank. Persistent Cookie Best Practices Many sites provide a "Remember Me" checkbox on login forms, which relies on the use of a persistent cookie. This login cookie would contain the user's username but not the password. Instead, the

login cookie would contain a random token; this random token would be stored along with the username in the site's back-end database. Every time the user logs in, a new token would be generated and stored in the database and cookie. Another common, nonessential use of cookies would be to use them to store user preferences. For instance, some sites allow the user to choose their preferred site color scheme or their country of origin or site language. In these cases, saving the user's preferences in a cookie will make for a more contented user, but if the user's browser does not accept cookies, then the site will still work just fine; at worst the user will simply have to reselect his or her preferences again. Another common use of cookies is to track a user's browsing behavior on a site. Some sites will store a pointer to the last requested page in a cookie; this information can be used by the site administrator as an analytic tool to help understand how users navigate through the site.

OR

10 a) Define AJAX. Explain AJAX request by writing UML diagram. (08 Marks)

Making a request to vote for option C in a poll could easily be encoded as a URL request GET /vote.php?option=C. However, rather than submit the whole page just to vote in the poll, jQuery's $.get() method sends that GET request asynchronously as follows:

```
$.get("/vote.php?option=C");
```

Note that the $ symbol is followed by a dot. Recall that since $ is actually shorthand for jQuery(), the above method call is equivalent tojQuery().("/vote.php?option=C");

Attaching that function call to the form's submit event allows the form's default behavior to be replaced with an asynchronous GET request.get() method can request a resource very easily, handling the response from the request requires that we revisit the notion of the handler and listener. The event handlers used in jQuery are no different than those we've seen in JavaScript, except that they are attached to the event triggered by a request completing

rather than a mouse move or key press. The formal definition of the get()method lists one required parameter url and three optional ones: data, a callbackto a success() method, and a dataType.

```
jQuery.get ( url [, data ] [, success(data, textStatus, jqXHR) ]
[, dataType ] )
```

■ url is a string that holds the location to send the request.

■ data is an optional parameter that is a query string or a Plain Object.

■ success(data,textStatus,jqXHR) is an optional callback function that executes when the response is received. Callbacks are the programming term given to placeholders for functions so that a function can be passed into another function and then called from there (called back). This callback function can take three optional parameters

° data holding the body of the response as a string.

° textStatus holding the status of the request (i.e., "success").

° jqXHR holding a jqXHR object, described shortly.

■ dataType is an optional parameter to hold the type of data expected from the server. By default jQuery makes an intelligent guess between xml, json, script,or html.

```
$.get("/vote.php?option=C", function(data,textStatus,jsXHR) {
if (textStatus=="success") {
console.log("success! response is:" + data);
}
else {
console.log("There was an error code"+jsXHR.status);
}
console.log("all done");
```

});
the callback function is passed as the second parameter to the get() method and uses the text Status parameter to distinguish between a successful post and an error. The data parameter contains plain text and is echoed out to the user in an alert. Passing a function as a parameter can be an odd syntax for newcomers to jQuery. Unfortunately, if the page requested (vote.php, in this case) does not exist on the server, then the callback function does not execute at all, so the code announcing an error will never be reached. To address this we can make use of the jqXHR object to build a more complete solution.

POST Requests

POST requests are often preferred to GET requests because one can post an unlimitedamount of data, and because they do not generate viewable URLs for each action.GET requests are typically not used when we have forms because of the messy URLsand that limitation on how much data we can transmit. Finally, with POST it ispossible to transmit files, something which is not possible with GET.Although the differences between a GET and POST request are relatively minor,the HTTP 1.1 definition describes GET as a safe method meaning that they shouldnot change anything, and should only read data. POSTs on the other hand are notsafe, and should be used whenever we are changing the state of our system.

jQuery handles POST almost as easily as GET, with the need for an added field tohold our data. The formal definition of a jQuery post() request is identical to theget() request, aside from the method name.

jQuery.post (url [, data] [, success(data, textStatus, jqXHR)][, dataType])

The main difference between a POST and a GET http request is where the data istransmitted. The data parameter, if present in the function call, will be put into thebody of the request. Interestingly, it can be passed as a string (with each name=valuepair separated with a “&” character) like a GET request or as a Plain Object, as withthe get() method.

If we were to convert our vote casting code from Listing 15.14 to a POST request,it would simply change the first line from

var jqxhr = $.get("/vote.php?option=C");

to

var jqxhr = $.post("/vote.php", "option=C");

Since jQuery can be used to submit a form, you may be interested in the shortcutmethod serialize(), which can be called on any form object to return its currentkey-value pairing as an & separated string, suitable for use with post().Consider our simple vote-casting example. Since the poll’s form has a singlefield, it’s easy to understand the ease of creating a short query string on the fly.However, as forms increase in size this becomes more difficult, which is why jQueryincludes a helper function to serialize an entire form in one step. The serialize()method can be called on a DOM form element as follows:

var postData = $("#voteForm").serialize();

With the form’s data now encoded into a query string (in the postData variable),you can transmit that data through an asynchronous POST using the $.post()method as follows:

$.post("vote.php", postData);

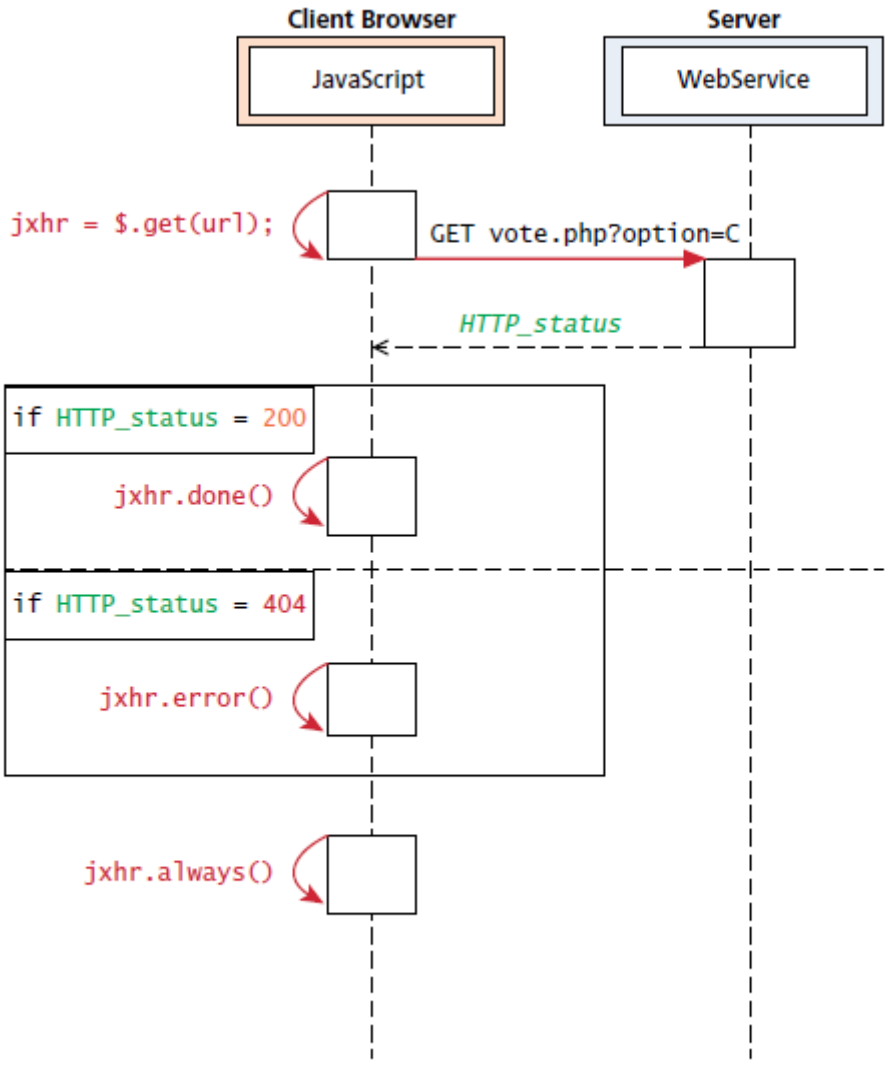


FIGURE 15.12 Sequence diagram depicting how the jqXHR object reacts to different response codes

b) Explain JavaScript pseudo-classes with examples. (08 Marks)

JavaScript Pseudo-Classes

Using Object Literals an array in JavaScript can be instantiated with elements in the following way:

var daysofWeek = ["sun","mon","tue","wed","thu","fri","sat"];

An object can be instantiated using the similar concept of object literals: that is,an object represented by the list of key-value pairs with colons between the key and value with commas separating key-value pairs.

A dice object, with a string to hold the color and an array containing the values representing each side (face), could be defined all at once using object literals as follows:

var oneDie = { color : "FF0000", faces : [1,2,3,4,5,6] };

Once defined, these elements can be accessed using dot notation.

For instance,one could change the color to blue by writing:

oneDie.color="0000FF";

Emulate Classes through Functions Although a formal class mechanism is not available to us in JavaScript, it is possible to get close by using functions to encapsulate variables and methods together,

function Die(col) { this.color=col; this.faces=[1,2,3,4,5,6]; }

The this keyword inside of a function refers to the instance, so that every reference to internal properties or methods manages its own variables, as is the case withPHP.

One can create an instance of the object as follows, very similar to PHP.

var oneDie = new Die("0000FF");

Adding Methods to the Object To define a method in an object's function one can either define it internally, or use a reference to a function defined outside the class. External definitions can quickly cause namespace conflict issues, since all method names must remain conflict free with all other methods for other classes.

For this reason, one technique for adding a method inside of a class definition is by assigning an anonymous function to a variable.

function Die(col) { this.color=col;this.faces=[1,2,3,4,5,6]; // define method randomRoll as an anonymous function

this.randomRoll = function() { var randNum = Math.floor((Math.random() * this.faces.length)+ 1);

return faces[randNum-1]; }; }

var oneDie = new Die("0000FF");

console.log(oneDie.randomRoll() + " was rolled");

c) Explain converting a JSON string to JSON object in JavaScript with suitable code segments. (04 Marks)

PHP comes with a JSON extension and as of version 5.2 of PHP, the JSON extension is bundled and compiled into PHP by default.Converting a JSON string intoa PHP object is quite straightforward:

```
<?php
// convert JSON string into PHP object
$text = '{"artist": {"name":"Manet","nationality":"France"}}';
$anObject = json_decode($text);
echo $anObject->artist->nationality;
// convert JSON string into PHP associative array
$anArray = json_decode($text, true);
echo $anArray['artist']['nationality'];
?>
```

json_decode() function can return either a PHP object or anassociative array. Since JSON data is often coming from an external source, oneshould always check for parse errors before using it, which can be done via thejson_last_error() function:

```
<?php
// convert JSON string into PHP object
$text = '{"artist": {"name":"Manet","nationality":"France"}}';
$anObject = json_decode($text);
// check for parse errors
if (json_last_error() == JSON_ERROR_NONE) {
```

```
echo $anObject->artist->nationality;
}
?>
```

To go the other direction (i.e., to convert a PHP object into a JSON string), youcan use the json_encode() function.

```
// convert PHP object into a JSON string
$text = json_encode($anObject);
```