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

CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 4 – Feb 2022

Sub:	Web Technology & its Applications				Sub Code:	15CS71 / 17CS71	Branch:	CSE	
Date:	01-02-2022	Duration:	90 min’s	Max Marks:	50	Sem / Sec:	D		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1 (a)	Explain the following HTML5 tags with example. (i) headings (ii)paragraphs (iii)Inline elements (iv)image (v) divisions	[05]	CO1	L2

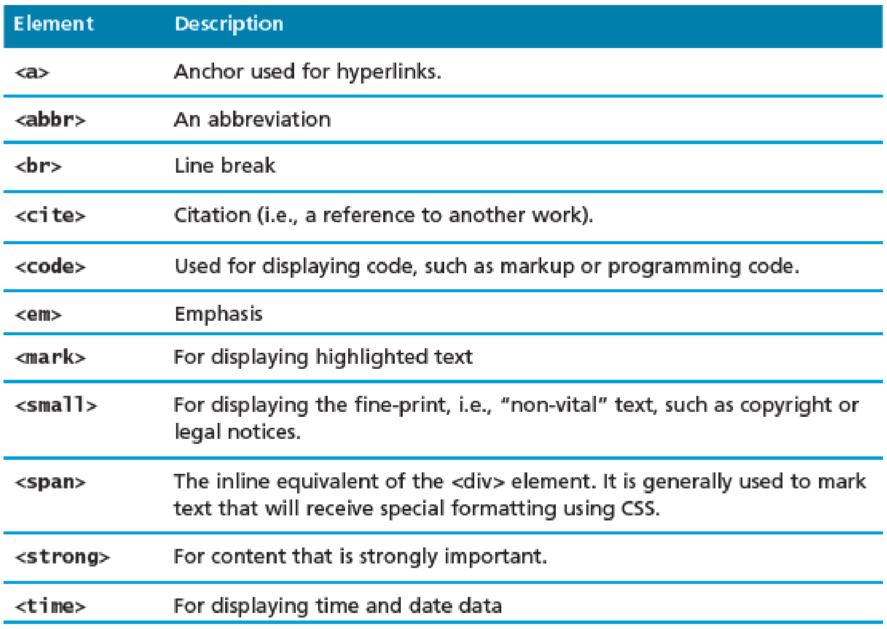
(i) headings: HTML provides six levels of heading (h1 through h6), with the higher heading number indicating a heading of less importance. Headings are also used by the browser to create a document outline for the page.

Example:
<h1>Share Your Travels</h1>
<h2>New York - Central Park</h2>

(ii)paragraphs: It is the most basic unit of text in an HTML document. Notice that the <p> tag is a container and can contain HTML and other inline HTML elements. This term refers to HTML elements that do not cause a paragraph break but are part of the regular “flow” of the text.

Example:
<p>Photo by Randy Connolly</p>
<p>This photo 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>

(iii)Inline elements: They do not disrupt the flow of text (i.e., cause a line break). HTML defines over 30 of these elements. Few are as follows:



Element	Description
<a>	Anchor used for hyperlinks.
<abbr>	An abbreviation
<br>	Line break
<cite>	Citation (i.e., a reference to another work).
<code>	Used for displaying code, such as markup or programming code.
<em>	Emphasis
<mark>	For displaying highlighted text
<small>	For displaying the fine-print, i.e., “non-vital” text, such as copyright or legal notices.
<span>	The inline equivalent of the <div> element. It is generally used to mark text that will receive special formatting using CSS.
<strong>	For content that is strongly important.
<time>	For displaying time and date data

iv)image: defines an image. It defines the following:

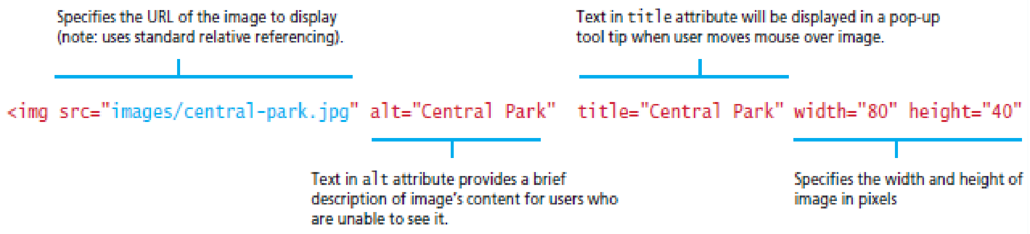

```
<img src="images/central-park.jpg" alt="Central Park" title="Central Park" width="80" height="40"
```
Specifies the URL of the image to display (note: uses standard relative referencing).
Text in title attribute will be displayed in a pop-up tool tip when user moves mouse over image.
Text in alt attribute provides a brief description of image's content for users who are unable to see it.
Specifies the width and height of image in pixels

FIGURE 2.18 The <img> element

(v)time: Used to represent time.

Example:
<p>By Susan on <time>October 1, 2015</time></p>

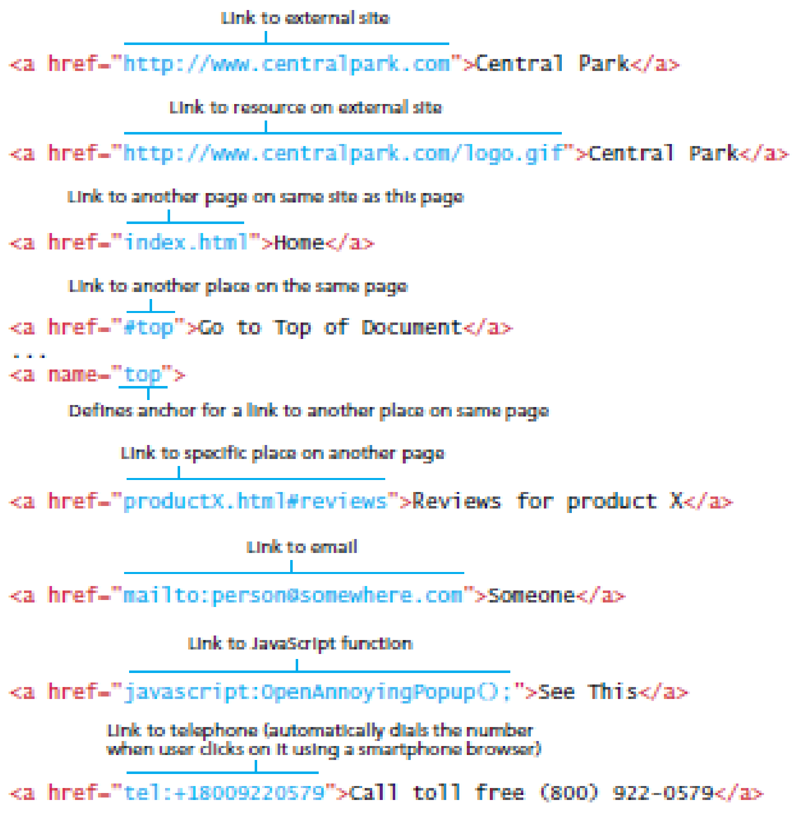
(vi) divisions:
This element is a container element and is used to create a logical grouping of content (text and other HTML elements, including containers such as <p> and other <div> elements). The <div> element has no intrinsic presentation; it is frequently used in contemporary CSS-based layouts to mark out sections.

Example:
<div>
<p>By Ricardo on <time>September 15, 2015</time></p>
<p>Easy on the HDR buddy.</p>
</div>
<div>
<p>By Susan on <time>October 1, 2015</time></p>
<p>I love Central Park.</p>
</div>
<p><small>Copyright &copy; 2015 Share Your Travels</small></p>
</body>

(b) Explain the following in brief with example. [05] CO1 L2
(i) Links to destination (ii) URL referencing

(i) Links to destination:
The use the anchor element to create a wide range of links. These include:

■ Links to external sites (or to individual resources such as images or movies on an external site).
■ Links to other pages or resources within the current site.
■ Links to other places within the current page.
■ Links to particular locations on another page (whether on the same site or on an external site).
■ Links that are instructions to the browser to start the user's email program.
■ Links that are instructions to the browser to execute a JavaScript function.
■ Links that are instructions to the mobile browser to make a phone call.



```
<a href="http://www.centralpark.com">Central Park</a>
<a href="http://www.centralpark.com/logo.gif">Central Park</a>
<a href="index.html">Home</a>
<a href="#top">Go to Top of Document</a>
...
<a name="top">
<a href="productX.html#reviews">Reviews for product X</a>
<a href="mailto:person@somewhere.com">Someone</a>
<a href="javascript:OpenAnnoyingPopup();">See This</a>
<a href="tel:+18009220579">Call toll free (800) 922-0579</a>
```

Link to external site
Link to resource on external site
Link to another page on same site as this page
Link to another place on the same page
Defines anchor for a link to another place on same page
Link to specific place on another page
Link to email
Link to JavaScript function
Link to telephone (automatically dials the number when user clicks on it using a smartphone browser)

FIGURE 2.16 Different link destinations

(i) URL 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.

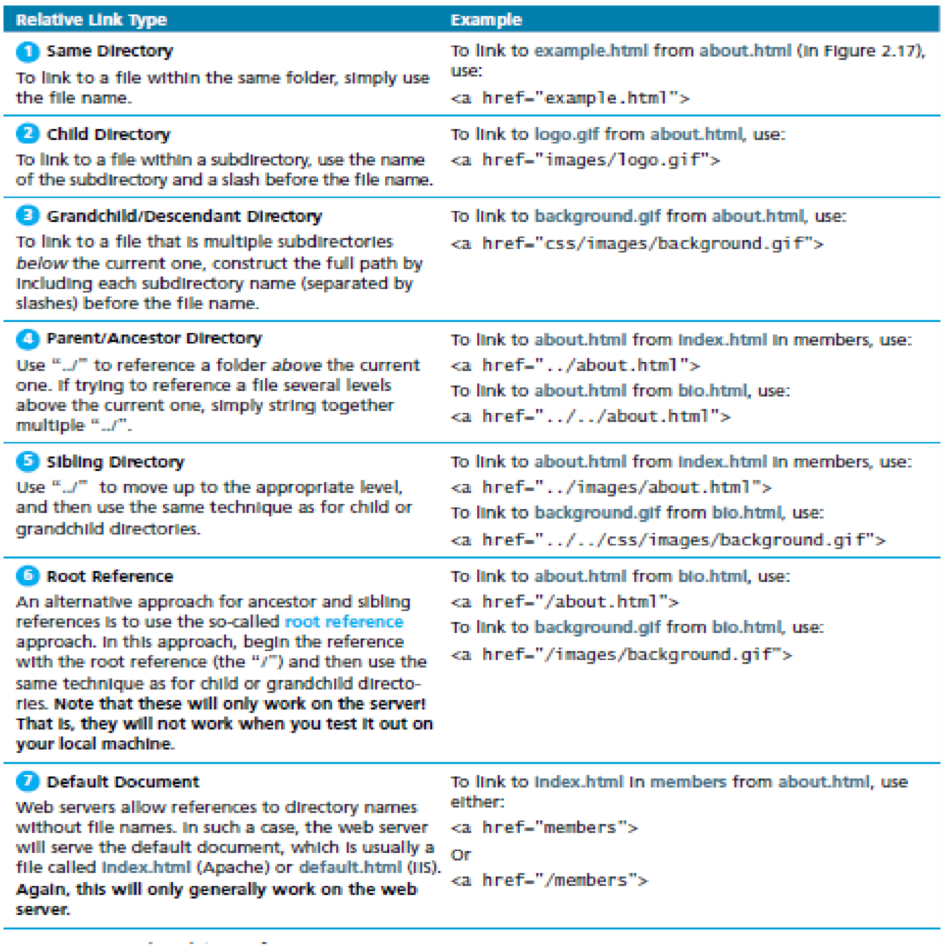
When referencing a page or resource on an external site, a full **absolute reference** is required: that is, a complete URL as described in Chapter 1 with a protocol (typically, http://), the domain name, any paths, and then finally the file name of the desired resource.

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.

The different types of URL referencing are:



Relative Link Type	Example
1 **Same Directory** To link to a file within the same folder, simply use the file name.	To link to example.html from about.html (in Figure 2.17), use: `<a href="example.html">`
2 **Child Directory** To link to a file within a subdirectory, use the name of the subdirectory and a slash before the file name.	To link to logo.gif from about.html, use: `<a href="images/logo.gif">`
3 **Grandchild/Descendant Directory** To link to a file that is multiple subdirectories *below* the current one, construct the full path by including each subdirectory name (separated by slashes) before the file name.	To link to background.gif from about.html, use: `<a href="css/images/background.gif">`
4 **Parent/Ancestor Directory** Use "../" to reference a folder *above* the current one. If trying to reference a file several levels above the current one, simply string together multiple "../".	To link to about.html from index.html in members, use: `<a href="../about.html">` To link to about.html from bio.html, use: `<a href="../../about.html">`
5 **Sibling Directory** Use "../" to move up to the appropriate level, and then use the same technique as for child or grandchild directories.	To link to about.html from index.html in members, use: `<a href="../images/about.html">` To link to background.gif from bio.html, use: `<a href="../../css/images/background.gif">`
6 **Root Reference** An alternative approach for ancestor and sibling references is to use the so-called root reference approach. In this approach, begin the reference with the root reference (the "/") and then use the same technique as for child or grandchild directories. **Note that these will only work on the server! That is, they will not work when you test it out on your local machine.**	To link to about.html from bio.html, use: `<a href="/about.html">` To link to background.gif from bio.html, use: `<a href="/images/background.gif">`
7 **Default Document** Web servers allow references to directory names without file names. In such a case, the web server will serve the default document, which is usually a file called index.html (Apache) or default.html (IIS). **Again, this will only generally work on the web server.**	To link to index.html in members from about.html, use either: `<a href="members">` Or `<a href="/members">`

2 (a) Explain Location of styles with examples. [05] CO3 L2

CSS style rules can be located in three different locations.

1. Inline Styles

Inline styles are style rules placed within an HTML element via the style attribute,An inline style only affects the element it is defined within and overrides any other style definitions for properties used in the inline style. Using inline styles is generally discouraged since they increase bandwidth and decrease maintainability (because presentation and content are intermixed and because it can be difficult to make consistent inline style changes across multiple files.

Internal styles example:

```
<h1>Share Your Travels</h1>
<h2>style="font-size: 24pt"Description</h2>
...
<h2>style="font-size: 24pt; font-weight: bold;">Reviews</h2>
```

2. Embedded Style Sheet

Embedded style sheets (also called **internal styles**) are style rules placed within the <style> element (inside the <head> element of an HTML document), 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.

Just as with inline styles, embedded styles can, however, be helpful when quickly testing out a style that is used in multiple places within a single HTML document.

Example:

```
<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>
...
```

3. External Style Sheet
External style sheets are style rules placed within a external text file with the **.css** extension. 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 as well. To reference an external style sheet, you must use a <link> element (within the <head> element),We can link to several style sheets at a time; each linked style sheet will require its own <link> element.

Example:

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

(b) Explain how CSS styles interact. [05] CO3 L2

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

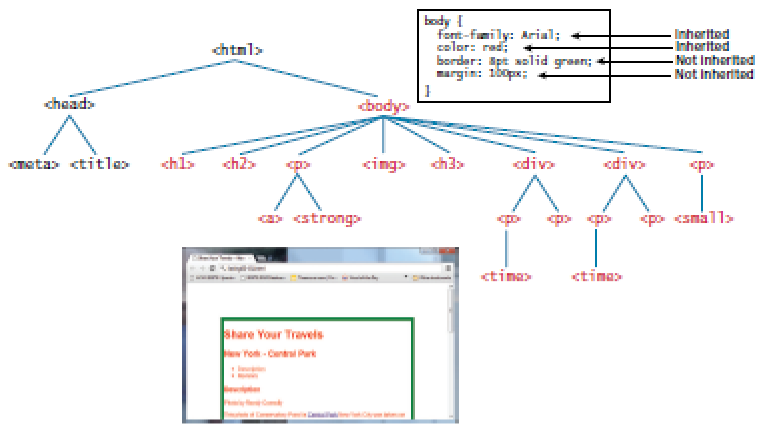
Inheritance is the first of these cascading principles. Many (but not all) CSS properties affect not only themselves but their descendants as well. Font, color, list, and text properties (from Table 3.1) are inheritable; layout, sizing, border, background, and spacing properties are not.
Figures 3.9 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 <div> elements. The <p> and <time> elements within the <div> 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 <div> elements now inherit the border and margins of their parent.

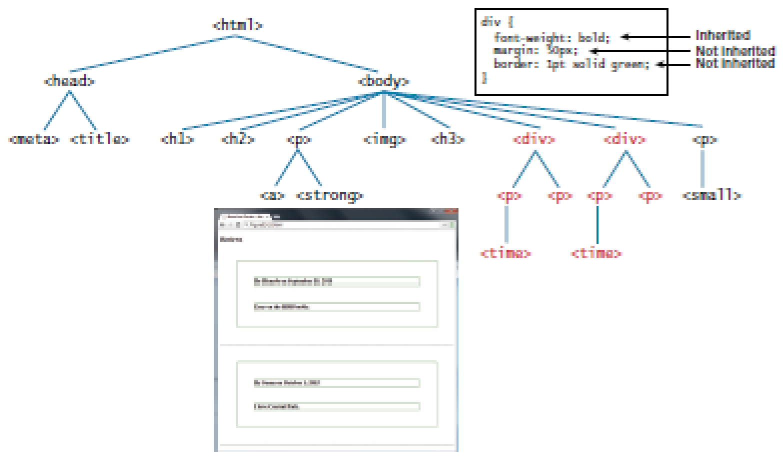
3.5.2 Specificity

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).



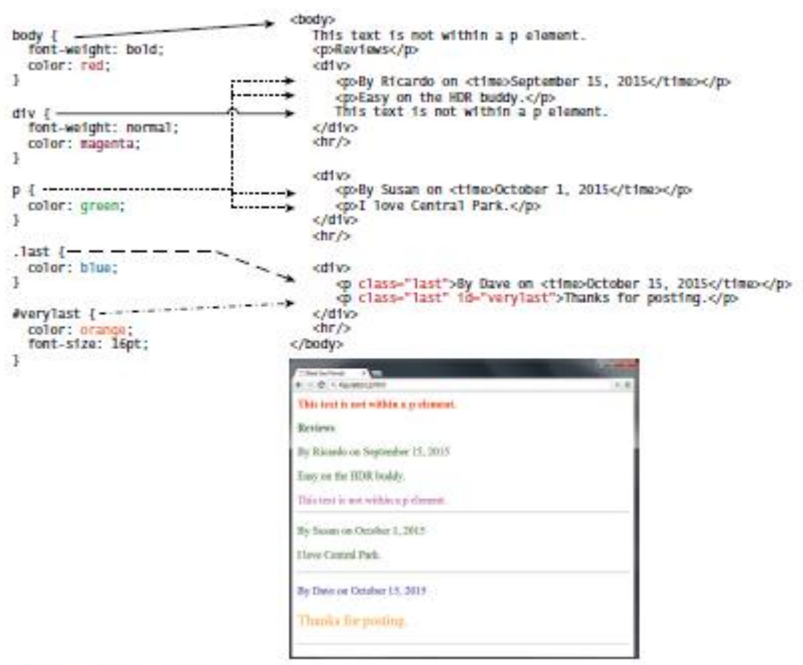
```
body {
   font-family: Arial;
   color: red;
   border: 8pt solid green;
   margin: 100px;
}
```

FIGURE 3.9 Inheritance



```
div {
   font-weight: bold;
   margin: 50px;
   border: 1pt solid green;
}
```

FIGURE 3.10 More Inheritance



```
body {
  font-weight: bold;
  color: red;
}

div {
  font-weight: normal;
  color: magenta;
}

p {
  color: green;
}

.last {
  color: blue;
}

#verylast {
  color: orange;
  font-size: 16pt;
}
```

```
<body>
   This text is not within a p element.
   <p>Reviews</p>
   <div>
      <p>By Ricardo on <time>September 15, 2015</time></p>
      <p>Easy on the HDR buddy.</p>
      This text is not within a p element.
   </div>
   <hr/>

   <div>
      <p>By Susan on <time>October 1, 2015</time></p>
      <p>I love Central Park.</p>
   </div>
   <hr/>

   <div>
      <p class="last">By Dave on <time>October 15, 2015</time></p>
      <p class="last" id="verylast">Thanks for posting.</p>
   </div>
   <hr/>
</body>
```

This text is not within a p element.

Reviews

By Ricardo on September 15, 2015

Easy on the HDR buddy.

This text is not within a p element.

By Susan on October 1, 2015

I love Central Park.

By Dave on October 15, 2015

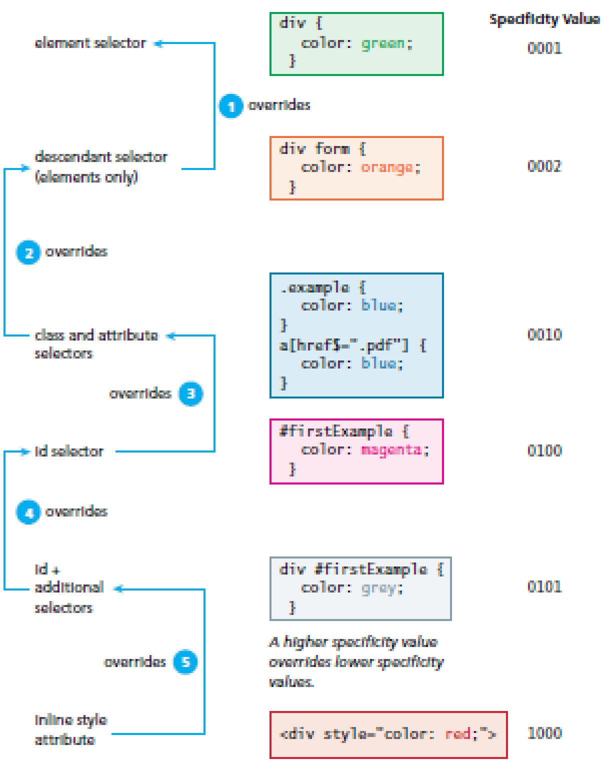
Thanks for posting.

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.


element selector
1 overrides
descendant selector (elements only)
2 overrides
class and attribute selectors
overrides 3
id selector
4 overrides
id + additional selectors
overrides 5
inline style attribute
Specificity Value
A higher specificity value overrides lower specificity values.

```
div {
  color: green;
}
```
0001

```
div form {
  color: orange;
}
```
0002

```
.example {
  color: blue;
}
a[href$=".pdf"] {
  color: blue;
}
```
0010

```
#firstExample {
  color: magenta;
}
```
0100

```
div #firstExample {
  color: grey;
}
```
0101

```
<div style="color: red;">
```
1000

FIGURE 3.13 Specificity algorithm

3 (a) List the different selectors available in CSS and explain in detail [10] CO3 L2

1 Element Selectors

Element selectors select all instances of a given HTML elementYou 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.

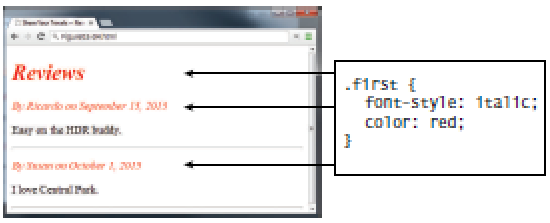
Class Selectors

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.
Listing 3.5 illustrates an example of styling using a class selector. The result in the browser is shown in Figure 3.4.

```
/* commas allow you to group selectors */
p, div, aside {
   margin: 0;
   padding: 0;
}
/* the above single grouped selector is equivalent to the
   following: */
p {
   margin: 0;
   padding: 0;
}
div {
   margin: 0;
   padding: 0;
}
aside {
   margin: 0;
   padding: 0;
}
```

LISTING 3.4 Sample grouped selector

then you can target it for styling by using an id selector, which takes the form: pound/hash (#) followed by the id name. Listing 3.6 illustrates an example of styling using an id selector. The result in the browser is shown in Figure 3.5.



Reviews

By Ricardo on September 15, 2015

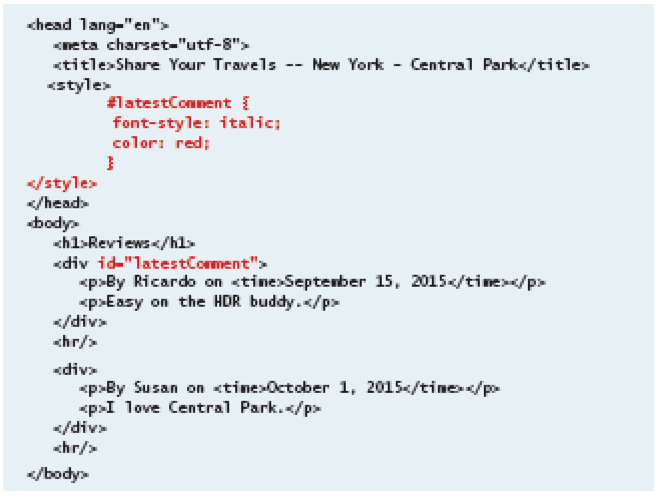
Easy on the HDR buddy.

By Susan on October 1, 2015

I love Central Park.

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

FIGURE 3.4 Class selector example in browser



```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
  <style>
         #latestComment {
          font-style: italic;
          color: red;
         }
</style>
</head>
<body>
   <h1>Reviews</h1>
   <div id="latestComment">
      <p>By Ricardo on <time>September 15, 2015</time></p>
      <p>Easy on the HDR buddy.</p>
   </div>
   <hr/>

   <div>
      <p>By Susan on <time>October 1, 2015</time></p>
      <p>I love Central Park.</p>
   </div>
   <hr/>

</body>
```

LISTING 3.6 Id selector example

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.

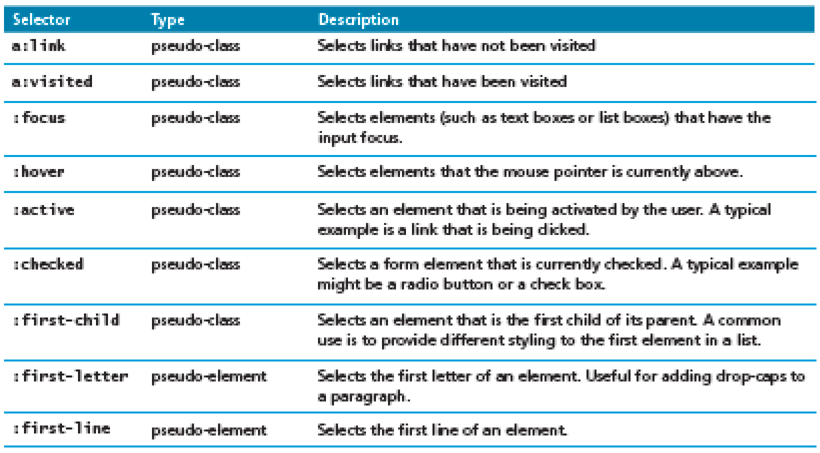
4 Attribute Selectors

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

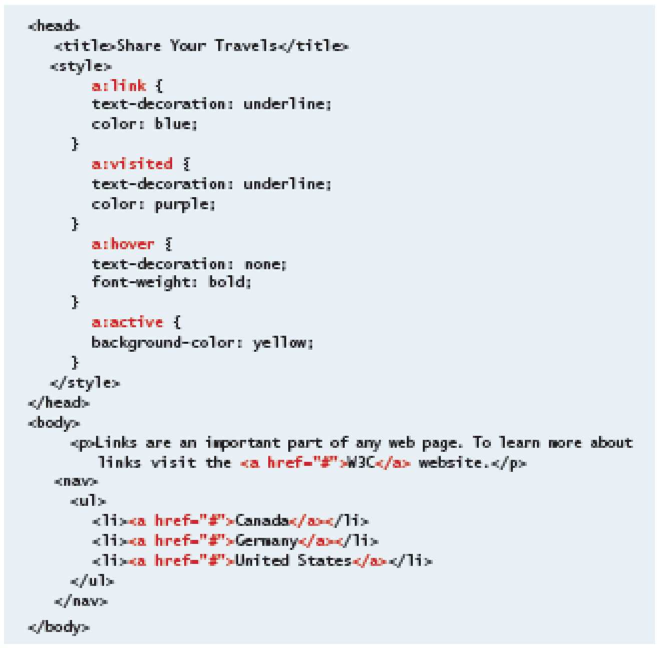
A **pseudo-element selector** is a way to select something that does not exist explicitly as an element in the HTML document tree but which is still a recognizable selectable object. For instance, you can select the first line or first letter of any HTML element using a pseudo-element selector. A **pseudo-class selector** does apply to an HTML element, but targets either a particular state or, in CSS3, a variety of family relationships. Table 3.5 lists some of the more common pseudo-class and pseudoelement selectors. 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. Listing 3.8
illustrates the use of pseudo-class selectors to style not only the visited and unvisited link colors, but also the hover color, which is the color of the link when the mouse is over the link. 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.



Selector	Type	Description
`a:link`	pseudo-class	Selects links that have not been visited
`a:visited`	pseudo-class	Selects links that have been visited
`:focus`	pseudo-class	Selects elements (such as text boxes or list boxes) that have the input focus.
`:hover`	pseudo-class	Selects elements that the mouse pointer is currently above.
`:active`	pseudo-class	Selects an element that is being activated by the user. A typical example is a link that is being clicked.
`:checked`	pseudo-class	Selects a form element that is currently checked. A typical example might be a radio button or a check box.
`:first-child`	pseudo-class	Selects an element that is the first child of its parent. A common use is to provide different styling to the first element in a list.
`:first-letter`	pseudo-element	Selects the first letter of an element. Useful for adding drop-caps to a paragraph.
`:first-line`	pseudo-element	Selects the first line of an element.

TABLE 3.5 Common Pseudo-Class and Pseudo-Element Selectors



```
<head>
   <title>Share Your Travels</title>
   <style>
        a:link {
        text-decoration: underline;
        color: blue;
     }
        a:visited {
        text-decoration: underline;
        color: purple;
     }
        a:hover {
        text-decoration: none;
        font-weight: bold;
     }
        a:active {
        background-color: yellow;
     }
   </style>
</head>
<body>
     <p>Links are an important part of any web page. To learn more about
        links visit the <a href="#">W3C</a> website.</p>
   <nav>
     <ul>
        <li><a href="#">Canada</a></li>
        <li><a href="#">Germany</a></li>
        <li><a href="#">United States</a></li>
     </ul>
   </nav>
</body>
```

LISTING 3.8 Styling a link using pseudo-class selectors

.

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 <div> 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 <div> 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>.

4 (a) Discuss positioning elements with examples [10] CO3 L2

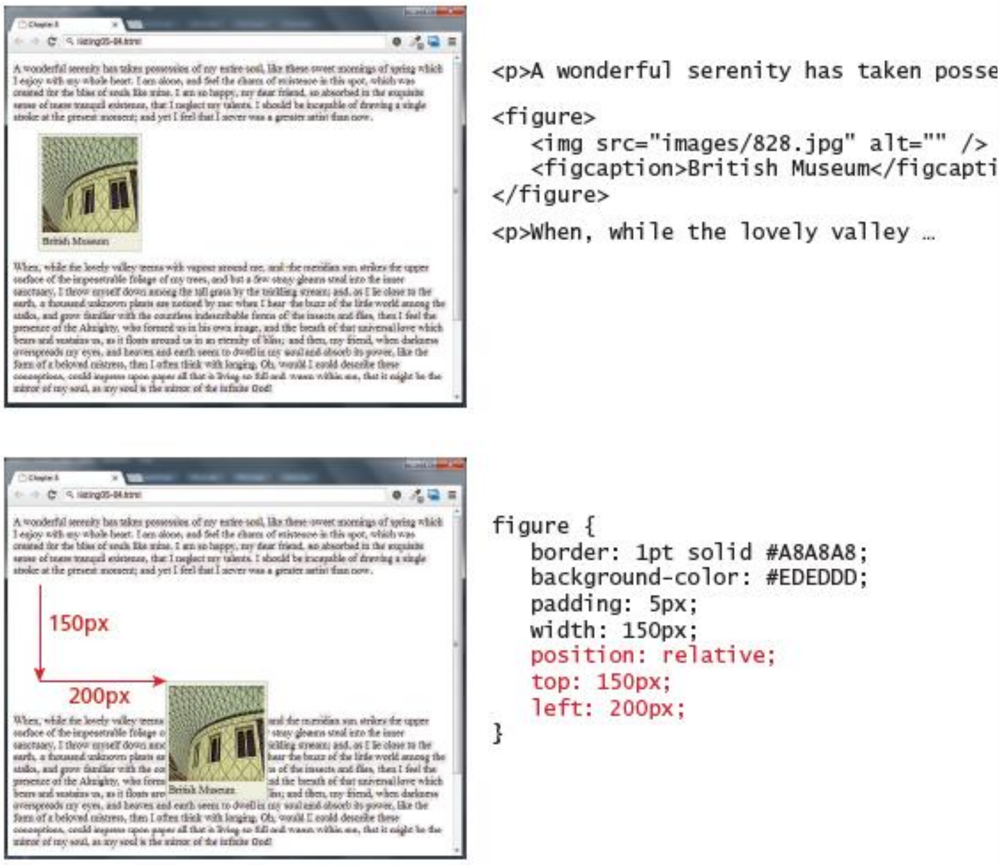
The position property is used to specify the type of positioning, and the possible values are shown in Table 5.1. The left, right, top, and bottom properties are used to indicate the distance the element will move; the effect of these properties varies depending upon the position property.

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 posse

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

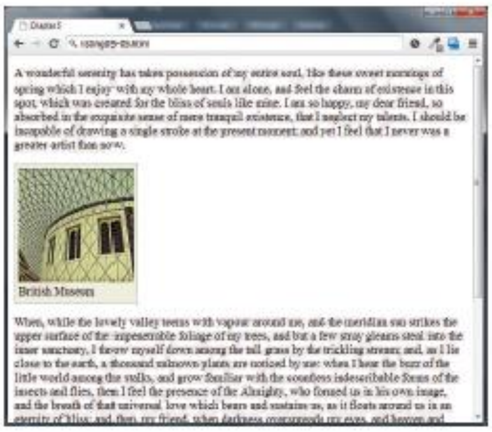
FIGURE 5.4 Relative positioning

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.

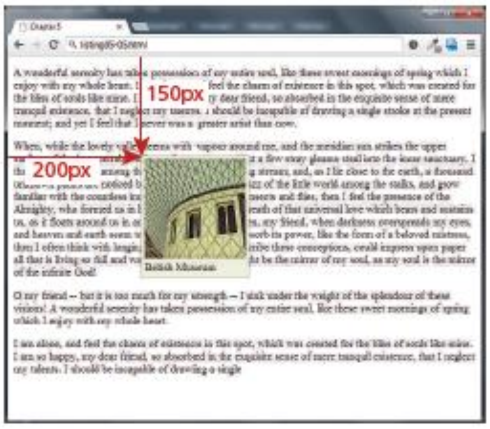
Like with the relative positioning example, the moved block can now overlap content in the underlying normal flow.



```
<p>A wonderful serenity has taken posse

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

5 (a) What does floating an element do in CSS? How do you float an element? [05] CO4 L2

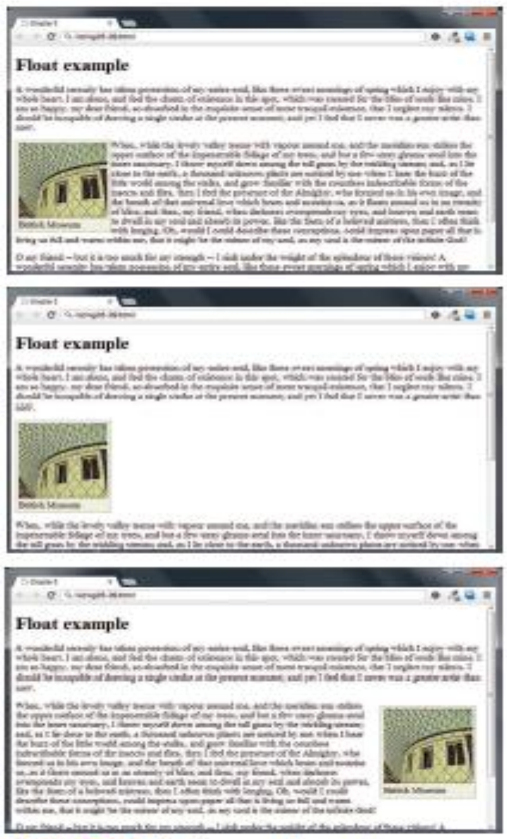
Floating Elements:

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

note in the final example in Figure 5.9 that the margins on the floated element are respected by the content that surrounds the floated element.

Floating within a Container:

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



```
<h1>Float example</h1>
<p>A wonderful serenity has taken ...</p>
<figure>
    <img src="images/828.jpg" alt="" />
    <figcaption>British Museum</figcaption>
</figure>
<p>When, while the lovely valley …</p>

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

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

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

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

FIGURE 5.9 Floating an element

(b) Write short notes on graceful degradation and progressive enhancement. [05] CO3 L2

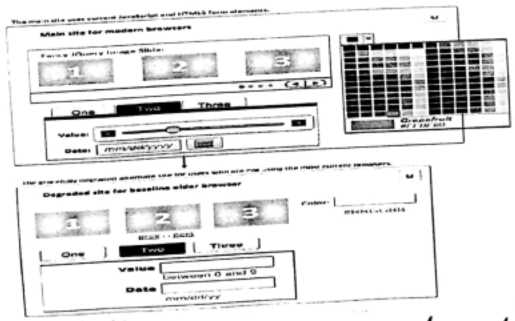
Graceful Degradation and Progressive Enhancement

Graceful Degradation :-

→ with this strategy you develop your site for the abilities of current browsers.

→ For those users who are not using current browsers, we need to provide an alternative site or pages for those using older browsers that lack the JS used on the main site.

→ The site is "degraded" (ie) loses capability "gracefully" (ie) without pop-up Javascript error codes or without condescending messages telling users to upgrade their browsers.



The main site uses current JavaScript and HTML5 form elements.
Main site for modern browsers
Fancy jQuery Image Slide
1
2
3
One
Two
Three
Value:
Date:
mm/dd/yyyy
Grapefruit
The gracefully degraded alternate site for users who are not using the most current browsers.
Degraded site for baseline older browser
Color:
One
Two
Three
Value
between 0 and 9
Date
mm/dd/yy

fig : Example for graceful degradation

progressive Enhancement :-

→ The developer creates the site using CSS, JS and HTML features that are supported by all browsers of a certain age.

→ The developers can now "progressively" "enhance" (ie) add functionality (i.e.) for each browser to their site based on the capabilities of the users browsers.

eg :: current version of chrome and opera might see a fancy HTML5 color input form elements while users of IE7 might see a simple text box.

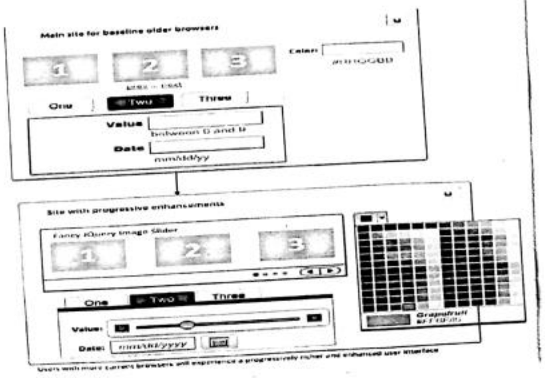
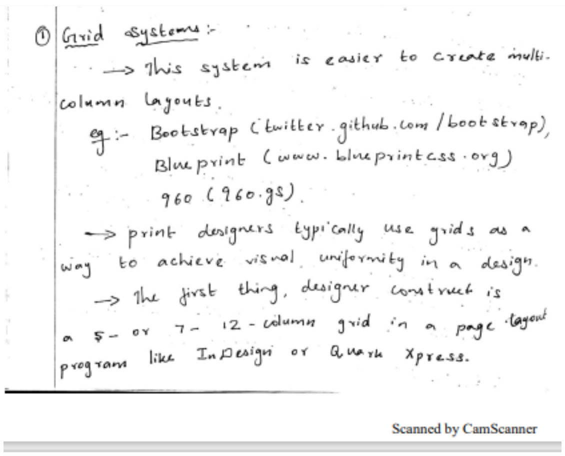

Main site for baseline older browsers
Color:
One
Two
Three
Value
between 0 and 8
Date
mm/dd/yy
Site with progressive enhancements
Fancy jQuery Image Slider
One
Two
Three
Value:
Date:
mm/dd/yyyy
Grapefruit
Users with more current browsers will experience a progressively richer and enhanced user interface

fig: site with progressive enhancements.

6 (a)	Explain Grid systems in CSS with examples.	[05]	CO4	L2



① <u>Grid systems</u> :-

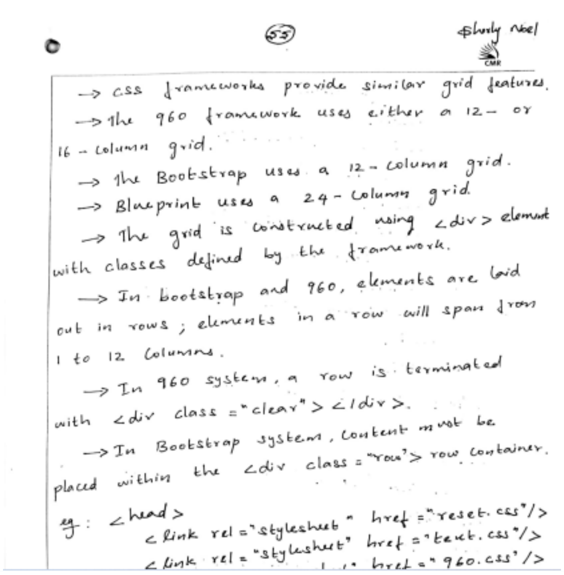
→ This system is easier to create multi-column layouts.

eg :- Bootstrap (twitter.github.com/bootstrap),
Blueprint (www.blueprintcss.org)
960 (960.gs).

→ print designers typically use grids as a way to achieve visual uniformity in a design.

→ The first thing, designer construct is a 5- or 7- 12-column grid in a page layout program like InDesign or Quark Xpress.

Scanned by CamScanner



Shirly Noel

→ css frameworks provide similar grid features.

→ The 960 framework uses either a 12- or 16-column grid.

→ The Bootstrap uses a 12-column grid.

→ Blueprint uses a 24-column grid.

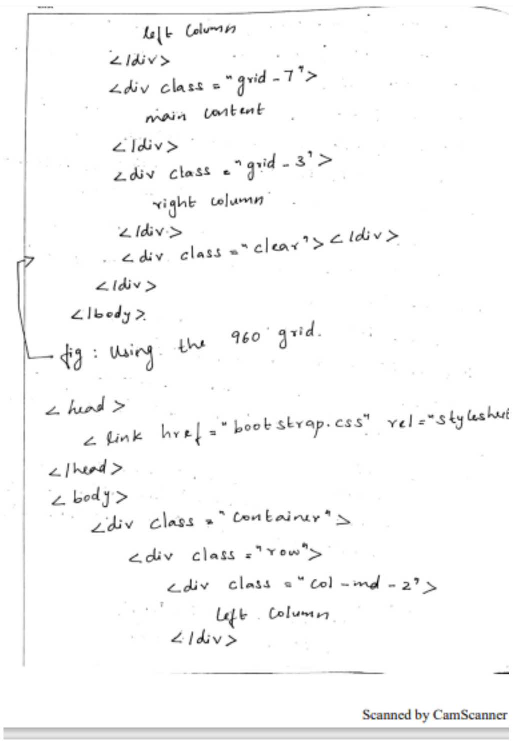
→ The grid is constructed using <div> element with classes defined by the framework.

→ In bootstrap and 960, elements are laid out in rows ; elements in a row will span from 1 to 12 Columns.

→ In 960 system, a row is terminated with <div class="clear"></div>.

→ In Bootstrap system, content must be placed within the <div class="row"> row container.

eg : <head>
<link rel="stylesheet" href="reset.css"/>
<link rel="stylesheet" href="text.css"/>
<link rel="stylesheet" href="960.css'/>

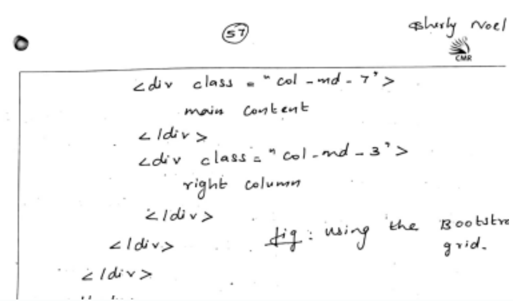


```
     left Column
   </div>
   <div class = "grid - 7">
       main content
   </div>
   <div class = "grid - 3">
       right column
   </div>
   <div class = "clear"></div>
  </div>
</body>
```

fig : Using the 960 grid.

```
<head>
   <link href = "bootstrap.css" rel = "styleshe
</head>
<body>
  <div class = "Container">
     <div class = "row">
        <div class = "col - md - 2">
            left Column
        </div>
```

Scanned by CamScanner



(57) Shirly Noel

```
        <div class = "col - md - 7">
            main Content
        </div>
        <div class = "col - md - 3">
            right column
        </div>
     </div>
  </div>
```

fig : using the Bootstr grid.

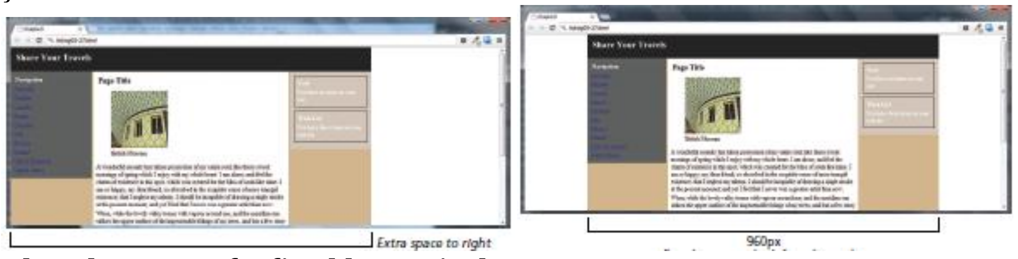
(b)	Write short notes on CSS Layout.	[05]	CO2	L2

- One of the main problems faced by web designers is that the size of the screen used to view the page can vary quite a bit.
- Some users will visit a site on a 21-inch wide screen monitor that can display 1920 × 1080 pixels (px); others will visit it on an older iPhone with a 3.5 screen and a resolution of 320 × 480 px.
- Users with the large monitor might expect a site to take advantage of the extra size; users with the small monitor will expect the site to scale to the smaller size and still be usable.
- Satisfying both users can be difficult; the approach to take for one type of site content might not work as well with another site with different content.
- Basic models:
    A. Fixed Layout:
- 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 <div> 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

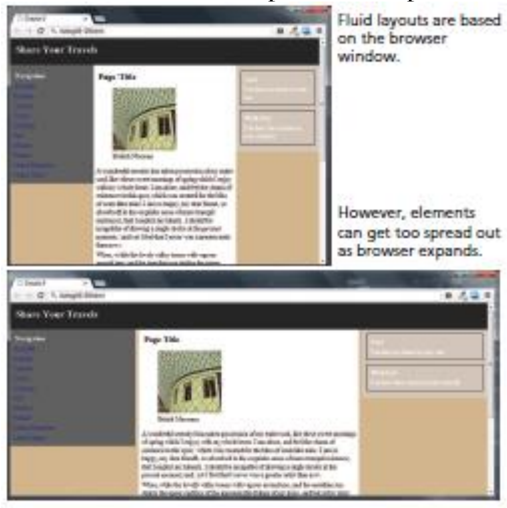
a) It is easier to produce and generally has a predictable visual result.
b) It is also optimized for typical desktop monitors.

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.

B. 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.


Share Your Travels
Page Title
Fluid layouts are based on the browser window.
However, elements can get too spread out as browser expands.

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.