CMR
INSTITUTE OF
TECHNOLOGY

Internal Assessment Test 2 Answer Key – Sep. 2022									
Sub:	Web Technologies					Sub Code:	20MCA23	Branch:	MCA
Date:	30/8/2022	Duration:	90 min's	Max Marks:	50	Sem	II		OBE

Q1) Discuss general syntactic characteristics and uses of JavaScript

General Syntactic Characteristics

- Scripts can appear directly as the content of a <script> tag.
- The type attribute of <script> must be set to "text/javascript".
- The JavaScript script can be indirectly embedded in an XHTML document with the src attribute of a <script> tag, whose value is the name of a file that contains the script—for example,
  <script type = "text/javascript" src = "tst_number.js" >

  </script>

- Notice that the script element requires the closing tag, even though it has no content when the src attribute is included.
- In JavaScript, identifiers, or names, must begin with a letter, an underscore (_), or a dollar sign ($). Subsequent characters may be letters, underscores, dollar signs, or digits. There is no length limitation for identifiers.
- JavaScript has 25 reserved words

break	delete	function	return	typeof
case	do	if	switch	var
catch	else	in	this	void
continue	finally	instanceof	throw	while
default	for	new	try	with

- In addition, JavaScript has a large collection of predefined words, including alert, open, java, and self.
- JavaScript has two forms of comments, both of which are used in other languages. First, whenever two adjacent slashes (//) appear on a line, the rest of the line is considered a comment. Second, /* may be used to introduce a comment, and */ to terminate it, in both single- and multiple-line comments.
- The XHTML comment used to hide JavaScript uses the normal beginning syntax,
  <!--.
- The following XHTML comment form hides the enclosed script from browsers that do not have JavaScript interpreters, but makes it visible to browsers that do support JavaScript:

```
<!--
-- JavaScript script --
// -->
```

- The use of semicolons in JavaScript is unusual. The JavaScript interpreter tries to make semicolons unnecessary, but it does not always work.
- When the end of a line coincides with what could be the end of a statement, the interpreter effectively inserts a semicolon there. But this can lead to problems.
- For example, return x;
- The interpreter will insert a semicolon after return, making x an invalid orphan.
- The safest way to organize JavaScript statements is to put each on its own line whenever possible and terminate each statement with a semicolon. If a statement does not fit on a line, be careful to break the statement at a place that will ensure that the first line does not have the form of a complete statement.

Uses of JavaScript

- The JavaScript was initially introduced to provide programming capability at both the server and client ends of web connection
- JavaScript therefore is implemented at 2 ends:
    - Client end
    - Server end
- The client side JavaScript is embedded in XHTML documents and is interpreted by the browser
- It also provides some means of computation, which serves as an alternative for some tasks done at the server side
- This transfer of task ensures that the server is not overloaded and performs only required task
- Interactions with users through form elements, such as buttons and menus, can be conveniently described in JavaScript. Because button clicks and mouse movements are easily detected with JavaScript, they can be used to trigger computations and provide feedback to the user.
- For example, when a user moves the mouse cursor from a text box, JavaScript can detect that movement and check the appropriateness of the text box's value (which presumably was just filled by the user).
- Even without forms, user interactions are both possible and simple to program in JavaScript. These interactions, which take place in dialog windows, include getting input from the user and allowing the user to make choices through buttons. It is also easy to generate new content in the browser display dynamically.
- But client side JavaScript cannot replace serves side JavaScript; because server side software supports file operations, database access, security, networking etc.
- JavaScript is also used as an alternative to java applets.
- Programming in JavaScript is much simpler than compared to java
- JavaScript support DOM [Document Object Model] which enables JavaScript to access and modify CSS properties and content of any element of a displayed XHTML document

Q2) Explain different image shapes used in Bootstrap with examples. Explain usage of icons with any three examples

Images have three classes that can be used to apply some simple styles: .img-rounded adds border-radius:6px to give the image rounded corners, .img-circle makes the entire image round by adding border-radius: 500px, and .img-polaroid adds a bit of padding and a gray border:

```
<img src="..." class="img-rounded">
<img src="..." class="img-circle">
<img src="..." class="img-polaroid">
```

Icons

Bootstrap bundles 140 icons into one sprite that can be used with buttons, links, navi-gation, and form fields. The icons are provided by GLYPHICONS

GLYPHICONS Attribution

Users of Bootstrap are fortunate to use the GLYPHICONS for free on Bootstrap projects. The developers have asked that you link back to GLYPHICONS when practical. GLYPHICONS Halflings are normally not available for free, but an arrangement between Bootstrap and the GLYPHICONS creators have made this possible at no cost to you as developers. As a thank you, we ask you to include an optional link back to GLYPHICONS whenever practical.

Usage

To use the icons, simply use an tag with the namespaced .icon- class. For example, if you want to use the edit icon, you add the .icon-edit class to the tag:

If you want to use the white icon, simply add the .icon-white class to the tag:

Button groups
By using button groups combined with icons, you can create nice interface elements with minimal markup

```
<div class="btn-toolbar">
 <div class="btn-group">
 <a class="btn" href="#"><i class="icon-align-left"></i></a>
 <a class="btn" href="#"><i class="icon-align-center"></i></a>
 <a class="btn" href="#"><i class="icon-align-right"></i></a>
 <a class="btn" href="#"><i class="icon-align-justify"></i></a>
 </div>
</div>
```

Navigation When you are using icons next to a string of text, make sure to add a space to provide the proper alignment of the image. Navigation code will be covered further in the next chapter.

```
<ul class="nav nav-list">
 <li class="active"><a href="#"><i class="icon-home icon-white"></i>
 Home</a></li>
 <li><a href="#"><i class="icon-book"></i> Library</a></li>
 <li><a href="#"><i class="icon-pencil"></i> Applications</a></li>
 <li><a href="#"><i class="i"></i> Misc</a></li>
</ul>
```

Q3) Discuss the purpose of the following javascript functions with examples:
a)Split() b)Match() c)Replace() d)Search()

a) Split()

The split method of String splits its object string into substrings on the basis of a given string or pattern. The substrings are returned in an array. For example, consider the following code:

```
var str = “grapes:apples:oranges”;
var fruit = str.split(“:”);
```

In this example, fruit is set to [grapes, apples, oranges]

b) match()

- The match method is the most general of the String pattern-matching methods.
- The match method takes a single parameter: a pattern. It returns an array of the results of the pattern-matching operation.
- If the pattern has the g modifier, the returned array has all of the substrings of the string that matched.
- If the pattern does not include the g modifier, the returned array has the match as its first element, and the remainder of the array has the matches of parenthesized parts of the pattern if there are any:

```
var str =
  "Having 4 apples is better than having 3 oranges";
var matches = str.match(/\d/g);
```

- In this example, matches is set to [4, 3].

c) replace()

The replace method is used to replace substrings of the String object that match the given pattern.

- The replace method takes two parameters: the pattern and the replacement string.
- The g modifier can be attached to the pattern if the replacement is to be global in the string, in which case the replacement is done for every match in the string.
- The matched substrings of the string are made available through the predefined variables $1, $2, and so on. For example, consider the following statements:
  var str = "Fred, Freddie, and Frederica were siblings";
  str.replace(/Fre/g, "Boy");
- In this example, str is set to "Boyd, Boyddie, and Boyderica were siblings", and $1, $2, and $3 are all set to "Fre".

d) search()

The simplest pattern-matching method is search, which takes a pattern as a parameter.

- The search method returns the position in the String object (through which it is called) at which the pattern matched.
- If there is no match, search returns –1.
- Most characters are normal, which means that, in a pattern, they match themselves.
- The position of the first character in the string is 0.
- As an example, the following statements

```
var str = "Rabbits are furry";
var position = str.search(/bits/);
if (position >= 0)
    document.write("'bits' appears in position", position,
                   "<br />");
else
    document.write("'bits' does not appear in str <br />");
```

produce the following output:

```
'bits' appears in position 3
```

Q4) What is an event and event handler? Explain any two event handler with examples.

An event is a notification that something specific has occurred, either with the browser, such as the completion of the loading of a document, or because of a browser user action, such as a mouse click on a form button.

An event handler is a script that is implicitly executed in response to the appearance of an event. Event handlers enable a Web document to be responsive to browser and user activities.

HANDLING EVENTS FROM BODY ELEMENTS

The events most often created by body elements are load and unload. As our first example of event handling, we consider the simple case of producing an alert message when the body of the document has been loaded. In this case, we use the onload attribute of <body> to specify the event handler

```
<?xml version = "1.0"  encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- load.html
     A document for load.js
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title> load.html </title>
    <script type = "text/javascript"  src = "load.js" >
    </script>
  </head>
  <body onload="load_greeting();">
    <p />
  </body>
</html>
```

```
// load.js
//   An example to illustrate the load event

// The onload event handler
function load_greeting () {
  alert("You are visiting the home page of \n" +
        "Pete's Pickled Peppers \n" + "WELCOME!!!");
}
```

OUTPUT

The unload event is probably more useful than the load event. It is used to do some cleanup before a document is unloaded, as when the browser user goes on to some new document. For example, if the document opened a second browser window, that window could be closed by an unload event handle

HANDLING EVENTS FROM BUTTON ELEMENTS

Buttons in a Web document provide an effective way to collect simple input from the browser user.
Example:

```
<?xml version = "1.0" encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- radio_click.hmtl
     A document for radio_click.js
     Creates four radio buttons that call the planeChoice
     event handler to display descriptions
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title> radio_click.html </title>
  <script type = "text/javascript"  src = "radio_click.js" >
  </script>
  </head>
  <body>
    <h4> Cessna single-engine airplane descriptions </h4>
    <form id = "myForm"  action = "">
      <p>
        <label> <input type = "radio"  name = "planeButton"
                       value = "152"
                       onclick = "planeChoice(152)" />
        Model 152 </label>
        <br />
        <label> <input type = "radio"  name = "planeButton"
                       value = "172"
                       onclick = "planeChoice(172)" />
        Model 172 (Skyhawk) </label>
        <br />
        <label> <input type = "radio"  name = "planeButton"
                       value = "182"
                       onclick = "planeChoice(182)" />
        Model 182 (Skylane) </label>
        <br />
        <label> <input type = "radio"  name = "planeButton"
                       value = "210"
                       onclick = "planeChoice(210)" />
        Model 210 (Centurian) </label>
      </p>
    </form>
  </body>
</html>
```

```
// radio_click.js
//   An example of the use of the click event with radio buttons,
//   registering the event handler by assignment to the button
//   attributes

// The event handler for a radio button collection
function planeChoice (plane) {

// Produce an alert message about the chosen airplane
  switch (plane) {
    case 152:
      alert("A small two-place airplane for flight training");
      break;
    case 172:
      alert("The smaller of two four-place airplanes");
      break;
    case 182:
      alert("The larger of two four-place airplanes");
      break;
    case 210:
      alert("A six-place high-performance airplane");
      break;
    default:
      alert("Error in JavaScript function planeChoice");
      break;
  }
}
```

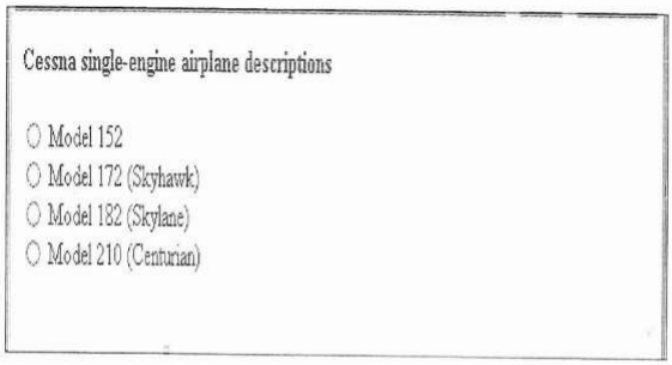


Figure 5.3 Display of `radio_click.html`

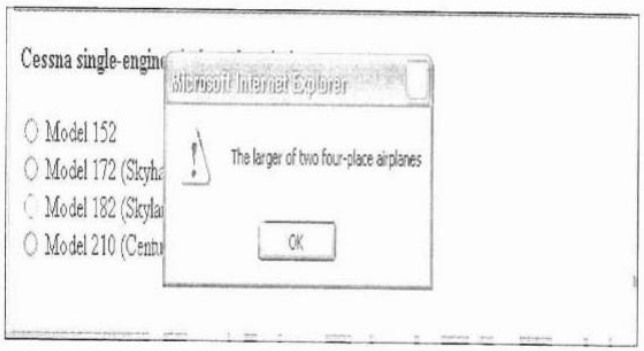


Figure 5.4 The result of pressing the Model 182 button in `radio_click`

Q5) Write in detail about bootstrap grid system with examples.

Default Grid System

The default Bootstrap grid (see Figure 1-1) system utilizes 12 columns, making for a 940px-wide container without responsive features enabled. With the responsive CSS file added, the grid adapts to be 724px or 1170px wide, depending on your viewport. Below 767px viewports, such as the ones on tablets and smaller devices, the columns become fluid and stack vertically. At the default width, each column is 60 pixels wide and offset 20 pixels to the left. An example of the 12 possible columns is in Figure 1-1

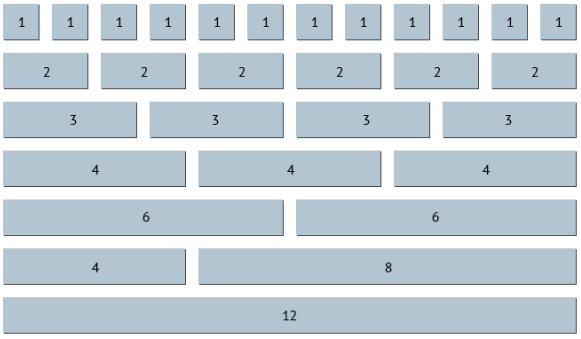


Figure 1-1. Default grid

Basic Grid HTML

To create a simple layout, create a container with a that has a class of .row and add the appropriate amount of .span* columns. Since we have a 12-column grid, we just need the amount of .span* columns to equal 12. We could use a 3-6-3 layout, 4-8, 3-5-4, 2-8-2… we could go on and on, but I think you get the gist.
The following code shows .span8 and .span4, which adds up to 12:

```
<div class="row">
<div class="span8">...</div>
<div class="span4">...</div>
</div>
```

Offsetting Columns

You can move columns to the right using the .offset* class. Each class moves the span over that width. So an .offset2 would move a .span7 over two columns (see Figure 1-2):

```
<div class="row">
<div class="span2">...</div>
<div class="span7 offset2">...</div>
</div>
```

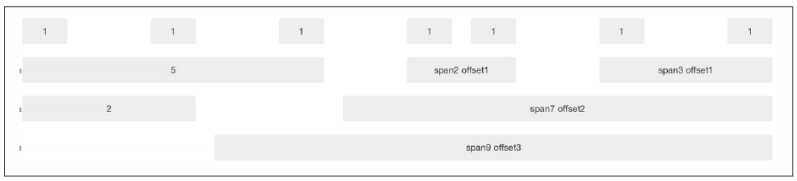


Figure 1-2. Offset grid

Nesting Columns

To nest your content with the default grid, inside of a .span*, simply add a new .row with enough .span* that it equals the number of spans of the parent container (see Figure 1-3):

```
<div class="row">
  <div class="span9">
    Level 1 of column
    <div class="row">
      <div class="span6">Level 2</div>
      <div class="span3">Level 2</div>
    </div>
  </div>
</div>
```

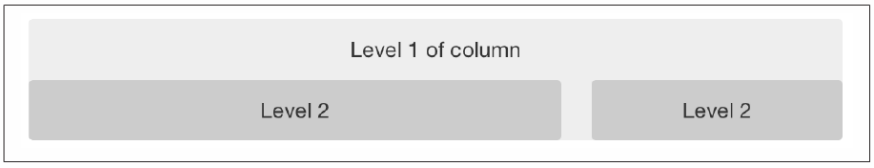


Figure 1-3. Nesting grid

Fluid Grid System

The fluid grid system uses percentages instead of pixels for column widths. It has the same responsive capabilities as our fixed grid system, ensuring proper proportions for key screen resolutions and devices. You can make any row “fluid” by changing .row to .row-fluid. The column classes stay exactly the same, making it easy to flip between fixed and fluid grids. To offset, you operate in the same way as the fixed grid system— add .offset* to any column to shift by your desired number of columns:

```
<div class="row-fluid">
  <div class="span4">...</div>
  <div class="span8">...</div>
</div>

<div class="row-fluid">
  <div class="span4">...</div>
  <div class="span4 offset2">...</div>
</div>
```

Nesting a fluid grid is a little different. Since we are using percentages, each .row resets the column count to 12. For example, if you were inside a .span8, instead of two .span4 elements to divide the content in half, you would use two .span6 divs (see Figure 1-4). This is the case for responsive content, as we want the content to fill 100% of the container:

```
<div class="row-fluid">
  <div class="span8">
        <div class="row">
              <div class="span6">...</div>
              <div class="span6">...</div>
        </div>
  </div>
</div>
```

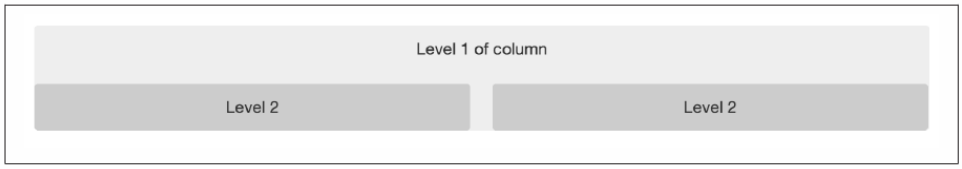


Figure 1-4. Nesting fluid grid

Q6) Explain the typography and emphasis classes in Bootstrap.

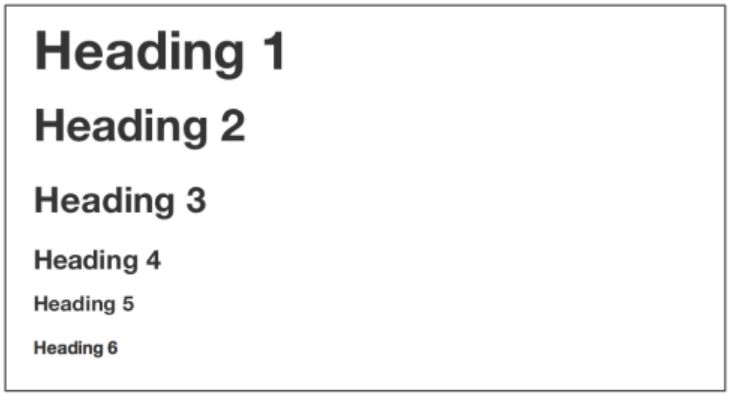
Starting with typography, Bootstrap uses Helvetica Neue, Helvetica, Arial, and sansserif in its default font stack. These are all standard fonts and are included as defaults on all major computers. If by chance these fonts don't exist, they fall back to sansserif (the catchall) to tell the browser to use the default font for the browser. All body copy has the font-size set at 14 pixels, with the line-height set at 20 pixels. The <p> tag has a margin-bottom of 10 pixels, or half the line-height.

Headings

All six standard heading levels have been styled in Bootstrap (see Figure 2-1), with the <h1> at 36 pixels tall, and the <h6> down to 12 pixels (for reference, default body text is 14 pixels tall). In addition, to add an inline subheading to any of the headings, simply add <small> around any of the elements and you will get smaller text in a lighter color.In the case of the <h1>, the small text is 24 pixels tall, normal font weight (i.e., not bold), and gray instead of black:

```
h1 small {

 font-size:24px;

 font-weight:normal;

 line-height:1;

 color:#999;
```

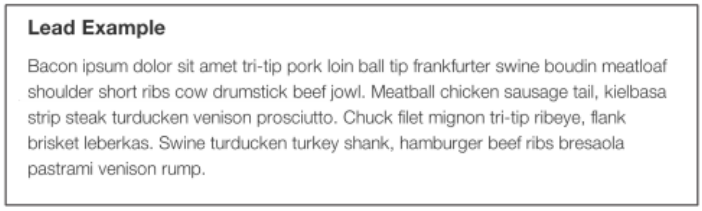
}



Lead Body Copy

To add some emphasis to a paragraph, add class="lead" (see Figure 2-2). This will give you larger font size, lighter weight, and a taller line height. This is generally used for the first few paragraphs in a section, but it can really be used anywhere:

Bacon ipsum dolor sit amet tri-tip pork loin ball tip frankfurter swine boudin meatloaf shoulder short ribs cow drumstick beef jowl. Meatball chicken sausage tail, kielbasa strip steak turducken venison prosciutto. Chuck filet mignon tri-tip ribeye, flank brisket leberkas. Swine turducken turkey shank, hamburger beef ribs bresaola pastrami venison rump.



Emphasis

In addition to using the <small> tag within headings, as discussed above, you can also use it with body copy. When <small> is applied to body text, the font shrinks to 85% of its original size.

Bold

To add emphasis to text, simply wrap it in a tag. This will add fontweight:bold; to the selected text.

Italics

For italics, wrap your content in the tag. The term “em” derives from the word “emphasis” and is meant to add stress to your text

Emphasis Classes

Along with and , Bootstrap offers a few other classes that can be used to provide emphasis (see Figure 2-3). These could be applied to paragraphs or spans:

This content is muted

```
<p class="text-warning">This content carries a warning class</p>

<p class="text-error">This content carries an error class</p>

<p class="text-info">This content carries an info class</p>

<p class="text-success">This content carries a success class</p>

<p>This content has <em>emphasis</em>, and can be <strong>bold</strong></p>
```

Q7) Develop a simple calculator to perform arithmetic (addition, subtraction, multiplication and division) operations on given two numbers. Use Bootstrap for styling

```
<!DOCTYPE html>
<html>
	<head>
		<title>Program 9</title>
		<script type="text/javascript">
			function add(){
				var n1=document.getElementById("num1").value;
				var n2=document.getElementById("num2").value;
				if((n1=="")||(n2=="")){
					alert("Please enter both the numbers");
					return false;
				}
				var result= parseInt(n1)+parseInt(n2);
				document.getElementById("res").value=result;
			}
```

```
function sub(){
	var n1=document.getElementById("num1").value;
	var n2=document.getElementById("num2").value;
	if((n1=="")||(n2="")){
		alert("Please enter both the numbers");
		return false;
	}
	var result= parseInt(n1)-parseInt(n2);
	document.getElementById("res").value=result;
}
function mul(){
	var n1=document.getElementById("num1").value;
	var n2=document.getElementById("num2").value;
	if((n1=="")||(n2="")){
		alert("Please enter both the numbers");
		return false;
	}
	var result= parseInt(n1)*parseInt(n2);
	document.getElementById("res").value=result;
}
function div(){
	var n1=document.getElementById("num1").value;
	var n2=document.getElementById("num2").value;
	if((n1=="")||(n2="")){
		alert("Please enter both the numbers");
		return false;
	}
	if(parseInt(n2)>0)
	{
		var result= parseInt(n1)/parseInt(n2);
		document.getElementById("res").value=result;
```

```
                }
                else
                {
                        document.getElementById("res").value="";
                        alert("Divide By Zero");
                        return false;
                }
        }
        function clearAll(){
                document.getElementById("num1").value="";
                document.getElementById("num2").value="";
                document.getElementById("res").value="";
        }

    </script>
</head>
<body style="text-align: center;">
    <h1> A SIMPLE CALCULATOR </h1>
    <label>Number 1 = </label><input type="text" id="num1"/><br/><br/>
    <label>Number 2 = </label><input type="text" id="num2"/><br/><br/>
    <label>Result = </label><input type="text" id="res"/><br/><br/>
    <input type="button" value="ADD" onclick="add();"/>
    <input type="button" value="SUB" onclick="sub();"/>
    <input type="button" value="MUL" onclick="mul();"/>
    <input type="button" value="DIV" onclick="div();"/>
    <input type="button" value="CLEAR" onclick="clearAll();"/>
</body>
</html>
```

Q8) How to create a bootstrap table Illustrate it with an example.

One of my favorite parts of Bootstrap is the nice way that tables are handled. I do a lot of work looking at and building tables, and the clean layout is a great feature that's included in Bootstrap right off the bat. Table 2-1 lists the various elements supported by Bootstrap.

Tag	Description
`<table>`	Wrapping element for displaying data in a tabular format
`<thead>`	Container element for table header rows (`<tr>`) to label table columns
`<tbody>`	Container element for table rows (`<tr>`) in the body of the table
`<tr>`	Container element for a set of table cells (`<td>` or `<th>`) that appears on a single row
`<td>`	Default table cell
`<th>`	Special table cell for column (or row, depending on scope and placement) labels. Must be used within a `<thead>`
`<caption>`	Description or summary of what the table holds, especially useful for screen readers

If you want a nice, basic table style with just some light padding and horizontal dividers, add the base class of .table to any table (see Figure 2-13). The basic layout has a top border on all of the <td> elements:

```
<table class="table">
  <caption>...</caption>
  <thead>
    <tr>
      <th>...</th>
      <th>...</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>...</td>
      <td>...</td>
    </tr>
  </tbody>
</table>
```

Name	Phone Number	Rank
Kyle West	707-827-7001	Eagle
Davey Preston	707-827-7003	Eagle
Taylor Lemmon	707-827-7005	Eagle

Optional Table Classes

Along with the base table markup and the .table class, there are a few additional classes that you can use to style the markup. These four classes are: .table-striped, .tablebordered, .table-hover, and .table-condensed. Striped table By adding the .table-striped class, you will get stripes on rows within the <tbody> (see Figure 2-14). This is done via the CSS :nth-child selector, which is not available on Internet Explorer 7–8

Name	Phone Number	Rank
Kyle West	707-827-7001	Eagle
Davey Preston	707-827-7003	Eagle
Taylor Lemmon	707-827-7005	Eagle

Figure 2-14. Striped table class

Bordered table

If you add the .table-bordered class, you will get borders surrounding every element and rounded corners around the entire table, as shown in Figure 2-15

Name	Phone Number	Rank
Kyle West	707-827-7001	Eagle
Davey Preston	707-827-7003	Eagle
Taylor Lemmon	707-827-7005	Eagle

Figure 2-15. Bordered table class

Hover table

Figure 2-16 shows the `.table-hover` class. A light gray background will be added to rows while the cursor hovers over them.

Name	Phone Number	Rank
Kyle West	707-827-7001	Eagle
Davey Preston	707-827-7003	Eagle
Taylor Lemmon	707-827-7005	Eagle

Figure 2-16. Hover table class

Condensed table

If you add the `.table-condensed` class, as shown in Figure 2-17, row padding is cut in half to condense the table. This is useful if you want denser information.

Name	Phone Number	Rank
Kyle West	707-827-7001	Eagle
Davey Preston	707-827-7003	Eagle
Taylor Lemmon	707-827-7005	Eagle

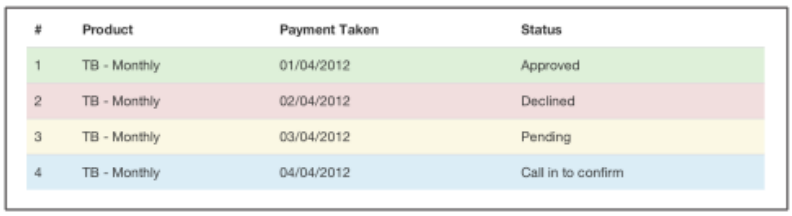
Figure 2-17. Condensed table class

Table Row Classes

The classes shown in Table 2-2 will allow you to change the background color of your rows (see Figure 2-18).

Table 2-2. Optional table row classes

Class	Description	Background color
`.success`	Indicates a successful or positive action.	Green
`.error`	Indicates a dangerous or potentially negative action.	Red
`.warning`	Indicates a warning that might need attention.	Yellow
`.info`	Used as an alternative to the default styles.	Blue



#	Product	Payment Taken	Status
1	TB - Monthly	01/04/2012	Approved
2	TB - Monthly	02/04/2012	Declined
3	TB - Monthly	03/04/2012	Pending
4	TB - Monthly	04/04/2012	Call in to confirm

Figure 2-18. Table row classes

Q9) Develop and demonstrate, using JavaScript script, a XHTML document that contains three short paragraphs of text, stacked on top of each other, with only enough of each showing so that the mouse cursor can be placed over some part of them. When the cursor is placed over the exposed part of any paragraph, it should rise to the top to become completely visible.

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
	<head>
		<title>Program 8a</title>
		<script type="text/javascript">
			var oldid="p3";
			function toTop(newid)
			{
				domTop=document.getElementById(oldid).style;
				domNew=document.getElementById(newid).style;
				domTop.zIndex="0";
				domNew.zIndex="10";
				oldid=newid;

			}
		</script>
		<style type="text/css">
			.para1{
				position:absolute;
				top:0px;
				left:0px;
				z-index:0;
				width:200px;
				height:200px;
				background-color:red;
			}
			.para2{
				position:absolute;
				top:100px;
				left:100px;
				z-index:0;
				width:200px;
				height:200px;
				background-color:blue;
			}
			.para3{
				position:absolute;
				top:200px;
				left:200px;
				z-index:0;
				width:200px;
				height:200px;
				background-color:green;
			}
		</style>
	</head>
	<body>
		<p class="para1" id="p1" onmouseover="toTop('p1')"> Frame One </p>
		<p class="para2" id="p2" onmouseover="toTop('p2')"> Frame Three </p>
		<p class="para3" id="p3" onmouseover="toTop('p3')"> Frame Two </p>


	</body>
```

</html>

Q10) a) Develop and demonstrate, a HTML document that collects the USN (the valid format is : A digit from 1 to 4 followed by two upper-case characters followed by two digits followed by three upper-case characters followed by two digits; (no embedded spaces are allowed) from the user. Use JavaScript that validate the content of the document. Suitable messages should be display in the alert if errors are detected in the input data. Use CSS and event handlers to make your document appealing.
b) Modify the above program to get the current semester also(restricted to be a number from 1 to 6)

```
<!DOCTYPE html>
<html>
	<head>
		<title>Program 5a</title>
		<script type="text/javascript">
			function funValidate(){
				var usn=document.getElementById('usn').value;
				var pattern=/^[1-4]{1}[A-Z]{2}[0-9]{2}[A-Z]{3}[0-9]{2}$/;

				if(usn.match(pattern))
				{
					alert("Valid Format");
				}
				else
				{
					alert("Invalid Format");
				}
			}

		</script>
	</head>
	<body>
		<label>Enter your USN: <input type="text" name="usn" id="usn" /></label>
```

```
		<input type="submit" name="validate" onClick="funValidate()" />
	</body>
</html>
<!DOCTYPE html>
<html>
	<head>
		<title>Program 5b</title>
		<script type="text/javascript">
			function funValidate(){
				var usn=document.getElementById('usn').value;
				var sem=document.getElementById('sem').value;
				var usn_pattern=/^[1-4]{1}[A-Z]{2}[0-9]{2}[A-Z]{3}[0-9]{2}$/;
				var sem_pattern=/^[1-6]{1}$/;
				if((usn.match(usn_pattern))&&(sem.match(sem_pattern)))
				{
					alert("Valid Format");
				}
				else if((!usn.match(usn_pattern))&&(sem.match(sem_pattern)))
				{
					alert("Invalid USN and Valid Sem");
				}
				else if((usn.match(usn_pattern))&&(!sem.match(sem_pattern)))
				{
					alert("Valid USN and Invalid Sem");
				}
				else{
					alert("Both are invalid");
				}
			}
```

```
            </script>
        </head>
        <body>
            <label>Enter your USN: <input type="text" name="usn" id="usn" /></label>
            <label>Enter your Sem: <input type="text" name="sem" id="sem" /></label>
            <input type="submit" name="validate" onClick="funValidate()" />
        </body>
</html>
```