CMR
INSTITUTE OF
TECHNOLOGY

Internal Assessment Test 2 Answer Key – Sep. 2024									
Sub:	Web Technologies					Sub Code:	22MCA24	Branch:	MCA
Date:	12/09/2024	Duration:	90 min's	Max Marks:	50	Sem	II		OBE

Q1) What is Bootstrap? Explain file structure with a neat diagram. Give an example of basic HTML template using Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation and other interface components.

Bootstrap File Structure bootstrap

```
bootstrap/
  ├── css/
  │   ├── bootstrap.css
  │   ├── bootstrap.min.css
  ├── js/
  │   ├── bootstrap.js
  │   ├── bootstrap.min.js
  └── img/
      ├── glyphicons-halflings.png
      └── glyphicons-halflings-white.png
```

The Bootstrap download includes three folders: css, js, and img. For simplicity, add these to the root of your project. Minified versions of the CSS and JavaScript are also included. It is not necessary to include both the uncompressed and the minified versions.

Basic HTML Template

Normally, a web project looks something like this:

```
<!DOCTYPE html>
<html>
        <head>
                <title>Bootstrap 101 Template</title>
        </head>
        <body>
                <h1>Hello, world!</h1>
        </body>
```

```
</html>
```

With Bootstrap, we include the link to the CSS stylesheet and the JavaScript:

```
<!DOCTYPE html>
<html>
        <head>
                <title>Bootstrap 101 Template</title>
                <link href="css/bootstrap.min.css" rel="stylesheet">
        </head>
        <body>
                <h1>Hello, world! </h1>
                <script src="js/bootstrap.min.js"></script>
        </body>
</html>
```

Q2) How to create a bootstrap table illustrate it with an example. Explain any 3 table classes with 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

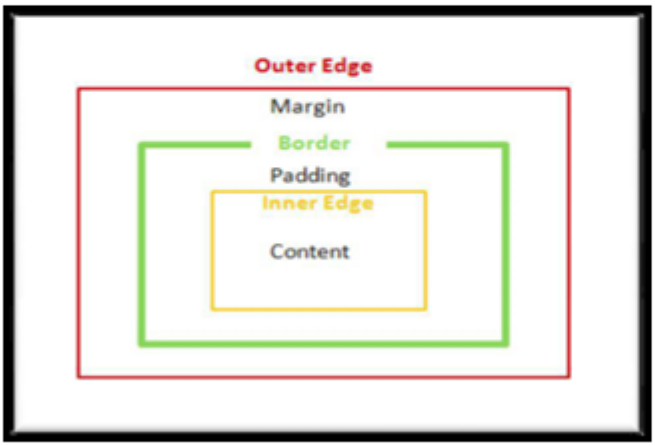
Class	Description	Background color
`.success`	Indicates a successful or positive action.	Green
`.error`	Indicates a dangerous or potentially negative action.	Red
`.warning`	Indicates a warning that might need attention.	Yellow
`.info`	Used as an alternative to the default styles.	Blue

#	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

Q3) Explain CSS Box model with example ?

- On a given web page or a document, all the elements can have borders.
- The borders have various styles, color and width.
- The amount of space between the content of the element and its border is known as *padding.*
- The space between border and adjacent element is known as *margin.*



Borders:

1) **Border-style property** controls whether the elements content has a border, as well as the style of the border
   It can be dotted, dashed, double
   The styles of one of the four sides of an element can be set with
   - Border-top-style
   - Border-bottom-style
   - Border-left-style
   - Border-right-style

2) **Border-width** is used to specify the thickness of a border
   It can be thin, medium, thick or any length value
   The width of each of the four borders of an element specified with:
   - Border-top-width
   - Border-bottom-width
   - Border-left-width
   - Border-right-width

3) **Border-color** control color of a border
   The width of each of the four borders of an element specified with:
   - Border-top-color
   - Border-bottom-color
   - Border-left-color
   - Border-right-color

Margins and Padding:

The margin properties are named margin, which applies to all four sides of an element: margin-left, margin-right, margin-top, and margin-bottom.

The padding properties are named padding, which applies to all four sides: padding-left, padding-right, padding-top, and padding-bottom.

Q4) Discuss Grid systems and Containers of Bootstrap with example.

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

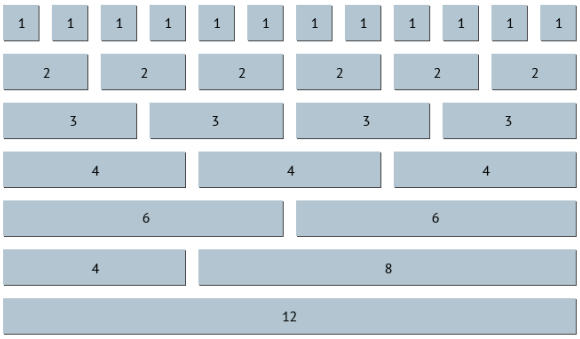


Figure 1-1. Default grid

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>
```

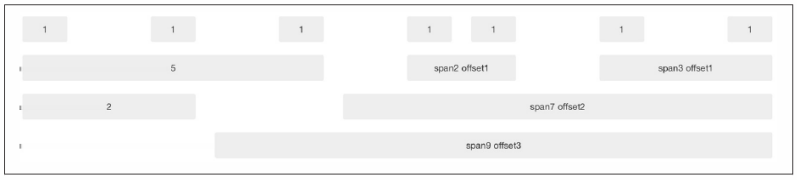


Figure 1-2. Offset grid

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>
```

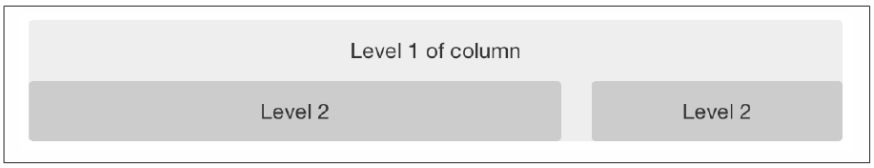


Figure 1-3. Nesting grid

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>
```

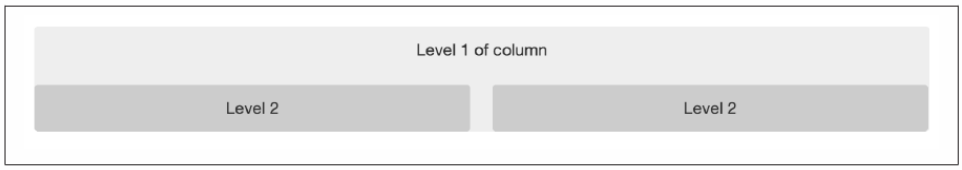


Figure 1-4. Nesting fluid grid

Q5) 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;
  }
}
```

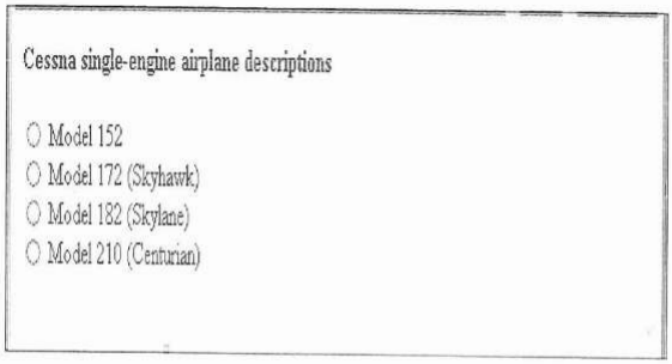


Figure 5.3 Display of `radio_click.html`

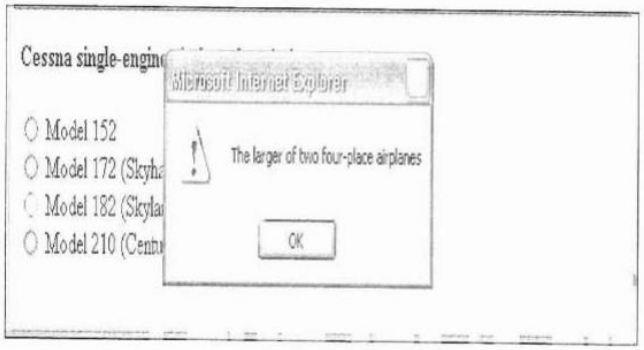


Figure 5.4 The result of pressing the Model 182 button in `radio_click`

Q6) Explain different ways of creating Array with example. Explain any 5 array functions in JavaScript with example.

Array METHODS

Array objects have a collection of useful methods, most of which are described in this section.

- The **join** method converts all of the elements of an array to strings and catenates them into a single string. If no parameter is provided to join, the values in the new string are separated by commas. If a string parameter is provided, it is used as the element separator. Consider the following example:

```
var names = new Array["Mary", "Murray",
                      "Murphy", "Max"];
...
var name_string = names.join(" : ");
```

The value of `name_string` is now `"Mary : Murray : Murphy : Max"`.

- The **reverse** method reverses the order of the elements of the Array object through which it is called.

- The **sort** method coerces the elements of the array to become strings if they are not already strings and sorts them alphabetically
- The **concat** method catenates its actual parameters to the end of the Array object on which it is called.

```
var names = new Array["Mary", "Murray",
                      "Murphy", "Max"];
...
var new_names = names.concat("Moo", "Meow");
```

The new_names array now has length 6, with the elements of names, along with "Moo" and "Meow" as its fifth and sixth elements.

- The **`slice`** method does for arrays what the `substring` method does for strings, returning the part of the `Array` object specified by its parameters, which are used as subscripts. The array returned has the elements of the `Array` object through which it is called, from the first parameter up to, but not including, the second parameter.

```
var list = [2, 4, 6, 8, 10];
...
var list2 = list.slice(1, 3);
```

- The value of `list2` is now `[4, 6]`. If `slice` is given just one parameter, the array that is returned has all of the elements of the object, starting with the specified index.
- When the **`toString`** method is called through an `Array` object, each of the elements of the object is converted (if necessary) to a string. These strings are catenated, separated by commas. So, for `Array` objects, the `toString` method behaves much like `join`.
- The **`push`**, **`pop`**, **`unshift`**, and **`shift`** methods of `Array` allow the easy implementation of stacks and queues in arrays. The `pop` and `push` methods respectively remove and add an element to the high end of an array, as in the following code:

```
var list = ["Dasher", "Dancer", "Donner", "Blitzen"];
var deer = list.pop();    // deer is "Blitzen"
list.push("Blitzen");
  // This puts "Blitzen" back on list
```

The `shift` and `unshift` methods respectively remove and add an element to the beginning of an array.

```
var deer = list.shift(); // deer is now "Dasher"

list.unshift("Dasher"); // This puts "Dasher" back on list
```

```
// nested_arrays.js
//   An example illustrating an array of arrays

// Create an array object with three arrays as its elements
var nested_array = [[2, 4, 6], [1, 3, 5], [10, 20, 30]
                   ];

// Display the elements of nested_list
for (var row = 0; row <= 2; row++) {
  document.write("Row ", row, ":  ");

  for (var col = 0; col <=2; col++)
    document.write(nested_array[row][col], " ");

  document.write("<br />");
}
```

OUTPUT:

```
Row 0: 2 4 6
Row 1: 1 3 5
Row 2: 10 20 30
```

Q7) Develop and demonstrate a XHTML file that includes Javascript script for the following problems: a) Input : A number n obtained using prompt Output : The first n Fibonacci numbers b) Input : A number n obtained using prompt Output : A table of numbers from 1 to n and their squares using alert

```
<!DOCTYPE html>

<html lang="en">

    <head>

        <meta charset="UTF-8">

        <title>Lab3a</title>

    </head>

  <body>

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

```
        //initialize variables
        var fib1=0,fib2=1,fib=0;
        var n=prompt("Enter a number");
        if(n!=null && n>0)
          {
            document.write("<h1>First " + n + " Fibonacci numbers are: </h1><br>");
            //if input is one number
            if(n==1)
              document.write("<p>" + fib1 + "</p><br/>");
            //if input is two numbers
            else
              document.write("<p>" + fib1 + "</p><br/><p>" + fib2 + "</p><br/>");
              //if input is more than two numbers, find the next Fibonacci number
              for(i=3;i<=n;i++)
                {
                  fib=fib1+fib2;
                  document.write("<p>" + fib + "</p><br/>");
                  fib1=fib2;
                  fib2=fib;
                }
          }
        else
          alert("Please enter proper input value");
      </script>
    </body>
  </html>


<!DOCTYPE HTML>
<html lang="en">
        <head>
                <meta charset="UTF-8">
                <title>Lab3b</title>
```

```
</head>

<body>
<h2> Javascript to generate squares of 1 to 'N' numbers</h2>
<script type="text/javascript">

var n=prompt ("Enter the limit 'n' to generate the table of numbers from 1 to n:","");
var msg="";
var res= "0";
for(var x= 1; x<=n;x++)
{
    res = x * x;
    msg = msg + " " + x + " * "+ x + " = " + res + "\n";
}
alert(msg);

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

Q8)Develop and demonstrate a HTML file which includes JavaScript that uses functions for the following problems: a) Input: A string Output: The position in the string of the left-most vowel. b) Input: A number Output: The number with its digits in the reverse order.

```
<!DOCTYPE HTML>
<?xml version="1.0" encoding="UTF-8"?>
<html>
<head>
<script type ="text/javascript">
function str_vowel()
{
 var str=prompt("Enter the string\n", " ");
 var pos=str.search(/[aeiou]/i);
```

```
if(pos>=0){
        document.write("The entered string is:" +str+ "<br/>");
        document.write("The leftmost vowel is :"+str.charAt(pos)+"<br/>");
        document.write("The position of the leftmost vowel " ,str.charAt(pos), " is:" ,(pos+1),"\n");
        exit;
}
else
{
        document.write("The entered string is:" +str+ "<br/>");
        document.write("The entered string has no vowels");
}

}
</script>
</head>
<body onload = "str_vowel();">
</body>
</html>

<!DOCTYPE HTML>
<?xml version="1.0" encoding="UTF-8"?>
<html>
<head>
<script type ="text/javascript">
function rev_num()
{
        var num = prompt("Enter the number to be reveresed :", " ");
        var n= num;
        var rev = 0, rem;
        while (n>0)
        {
                rem = n % 10;
```

```
            rev = rev * 10 + rem ;
            n = Math.floor(n/10);
        }
        document.write("The given number is : " +num+ " <br/> The reversed number is : " +rev+ "\n");
}


</script>
</head>
<body onload = "rev_num();">
</body>
</html>
```

Q9) Explain screen output and keyboard input statements available in javascript.

- JavaScript models the XHTML document with the `Document` object.
- The window in which the browser displays an XHTML document is modelled with the `Window` object.
- The `Window` object includes two properties, `document` and `window`.
- The `document` property refers to the `Document` object.
- The `window` property is self-referential; it refers to the `Window` object.
- `write` is used to create XHTML code, the only useful punctuation in its parameter is in the form of XHTML tags. Therefore, the parameter of `write` often includes `<br />`.
- The `writeln` method implicitly adds `"\n"` to its parameter, but since browsers ignore line breaks when displaying XHTML, it has no effect on the output.
- The parameter of `write` can include any XHTML tags and content.
- The `write` method actually can take any number of parameters.
- Multiple parameters are concatenated and placed in the output.
- Example: **`document.write("The result is: ", result, "<br />");`**

- There are 3 types of pop-up boxes:
  - Alert
  - Confirm
  - Prompt
- The `alert` method opens a dialog window and displays its parameter in that window. It also displays an OK button.
- The string parameter of `alert` is not XHTML code; it is plain text. Therefore, the string parameter of `alert` may include `\n` but never should include `<br />`.

```
alert("The sum is:" + sum + "\n");
```

- The `confirm` method opens a dialog window in which the method displays its string parameter, along with two buttons: OK and Cancel.
- `confirm` returns a Boolean value that indicates the user's button input: `true` for OK and `false` for Cancel. This method is often used to offer the user the choice of continuing some process.

```
var question = confirm("Do you want to continue this
download?");
```

- After the user presses one of the buttons in the `confirm` dialog window, the script can test the variable, `question`, and react accordingly.

- The `prompt` method creates a dialog window that contains a text box used to collect a string of input from the user, which `prompt` returns as its value.

Create an XHTML and JavaScript to compute the real roots of a given quadratic equation

```
<?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">

<!-- roots.html
     A document for roots.js
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title> roots.html </title>
  </head>
  <body>
    <script type = "text/javascript"  src = "roots.js" >
    </script>
  </body>
</html>
```

```
// roots.js
//   Compute the real roots of a given quadratic
//   equation. If the roots are imaginary, this script
//   displays NaN, because that is what results from
//   taking the square root of a negative number

// Get the coefficients of the equation from the user
var a = prompt("What is the value of 'a'? \n", "");
var b = prompt("What is the value of 'b'? \n", "");
var c = prompt("What is the value of 'c'? \n", "");

// Compute the square root and denominator of the result
var root_part = Math.sqrt(b * b - 4.0 * a * c);
var denom = 2.0 * a;

// Compute and display the two roots
var root1 = (-b + root_part) / denom;
var root2 = (-b - root_part) / denom;
document.write("The first root is: ", root1, "<br />");
document.write("The second root is: ", root2, "<br />");
```

Q10) 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.

```
<!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>
```