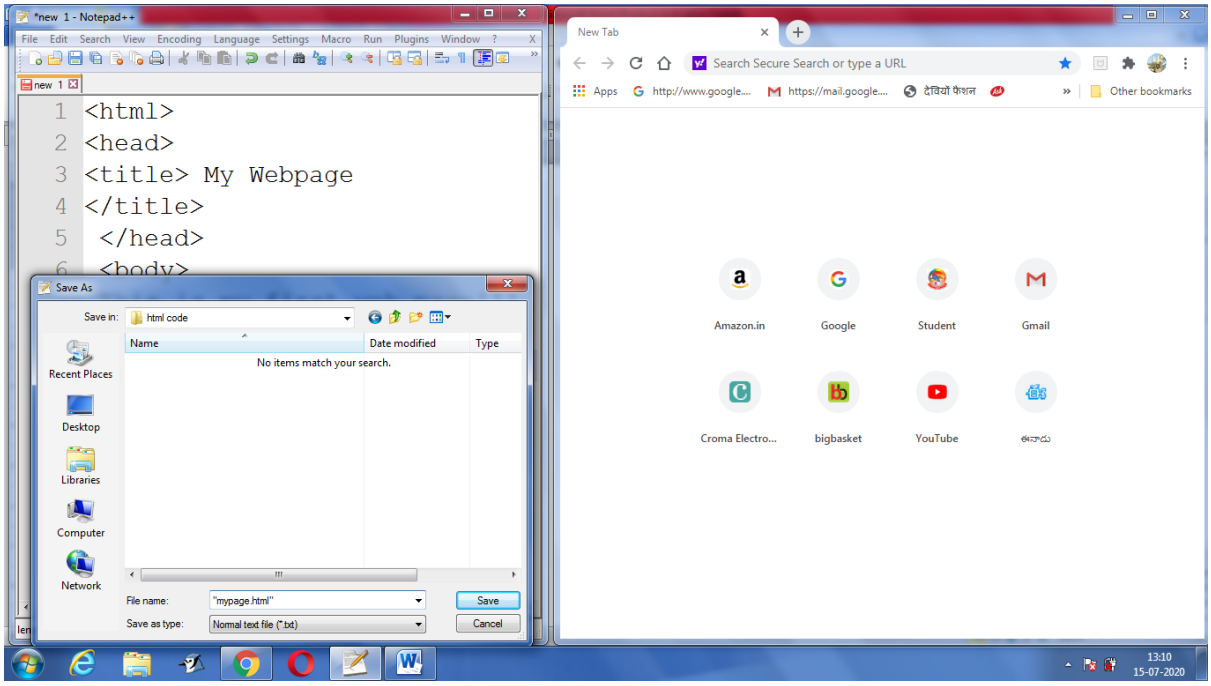
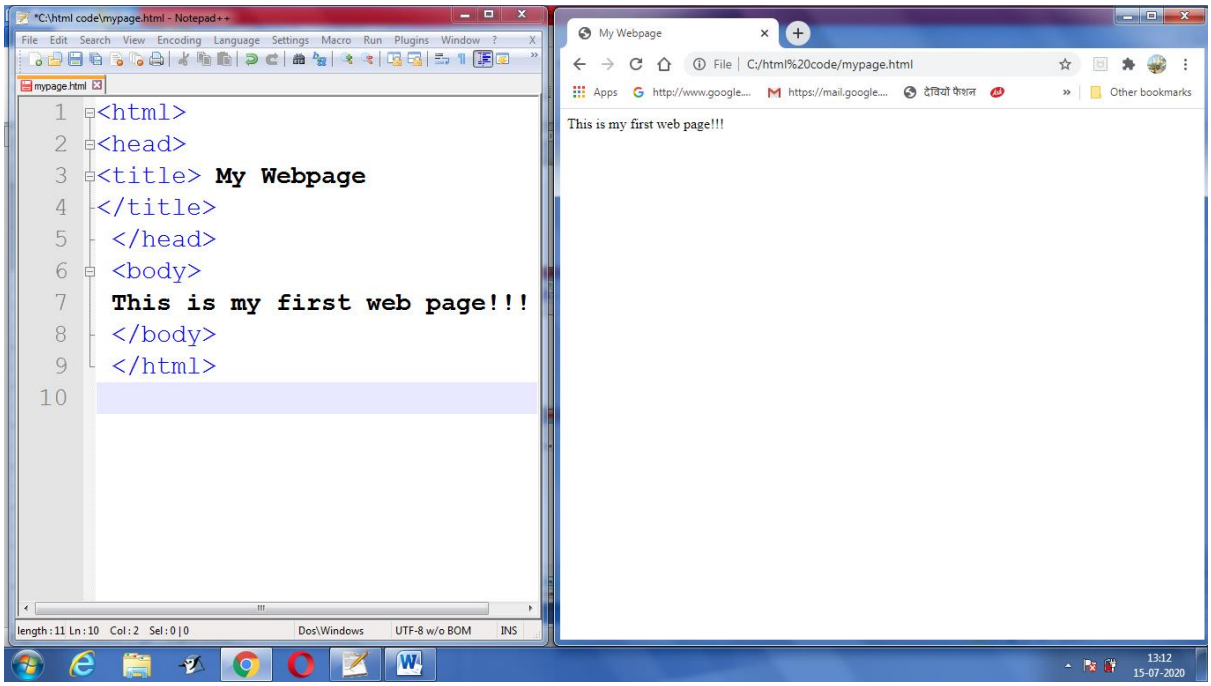
HTML_CSS_JAVASCRIPT-SAMPLE CODES FOR PRACTICE






```
head>
<title> My Webpage </title>
 </head>
```

The body part of the HTML program will help us to create a look and feel of the web page. The above given HTML program is the simplest one in which the web page contains the text *" This is my first web page!!!".*

```
<body>
 This is my first web page!!!
 </body>
```

Comments in HTML

The comment in HTML can be denotes as follows:

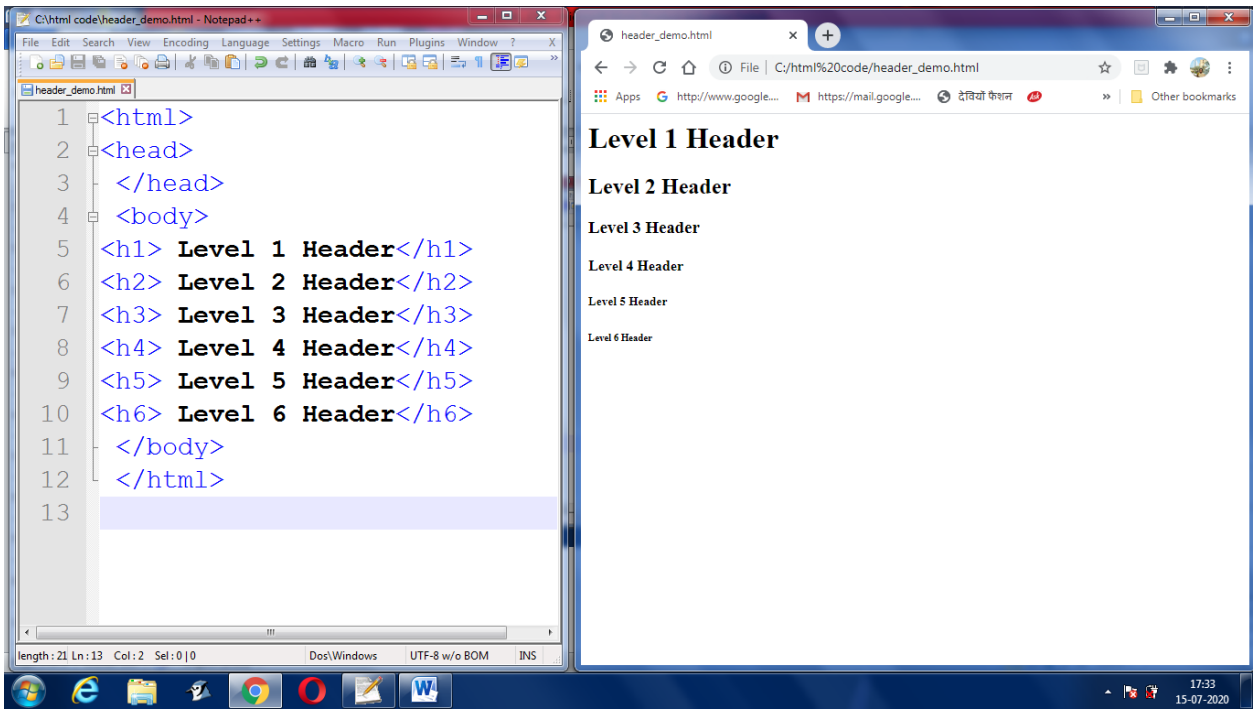
```
<!.. This is a comment statement ..!>
```

There should not be a space between angular bracket and exclamation mark. The comment is beginning with <!.. and ending with >>!>

TEXT

Displaying Header

There are header which helps to display the text as some header. The header tag is denoted by h1, h2 ,----- to h6. The following HTML page shows the demonstration of header tags.



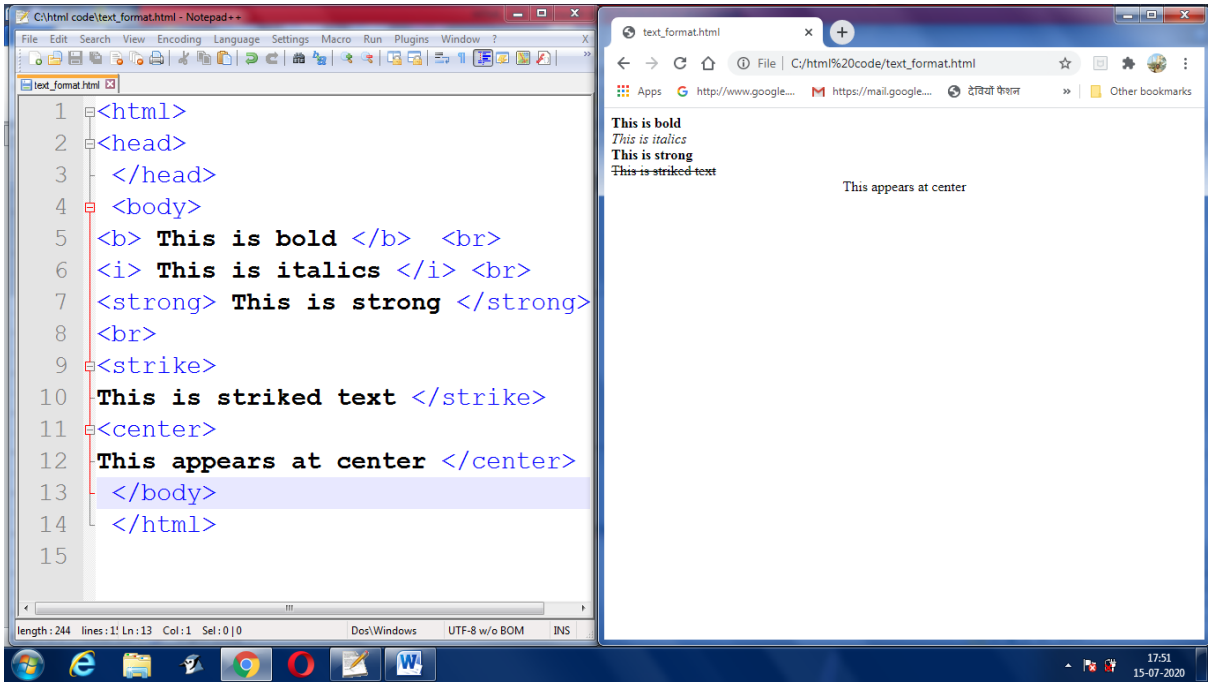
The folowing are some commanly used tags for text formatting-

Tag	Meaning
<p>	The <p> tag is used to indicate paragraphs. The p element automatically creates some space before and after itself.

 	The tfag inserts a line break but does not add extra space before the new line. The tag is empty, which means that the closing tag isn't required.

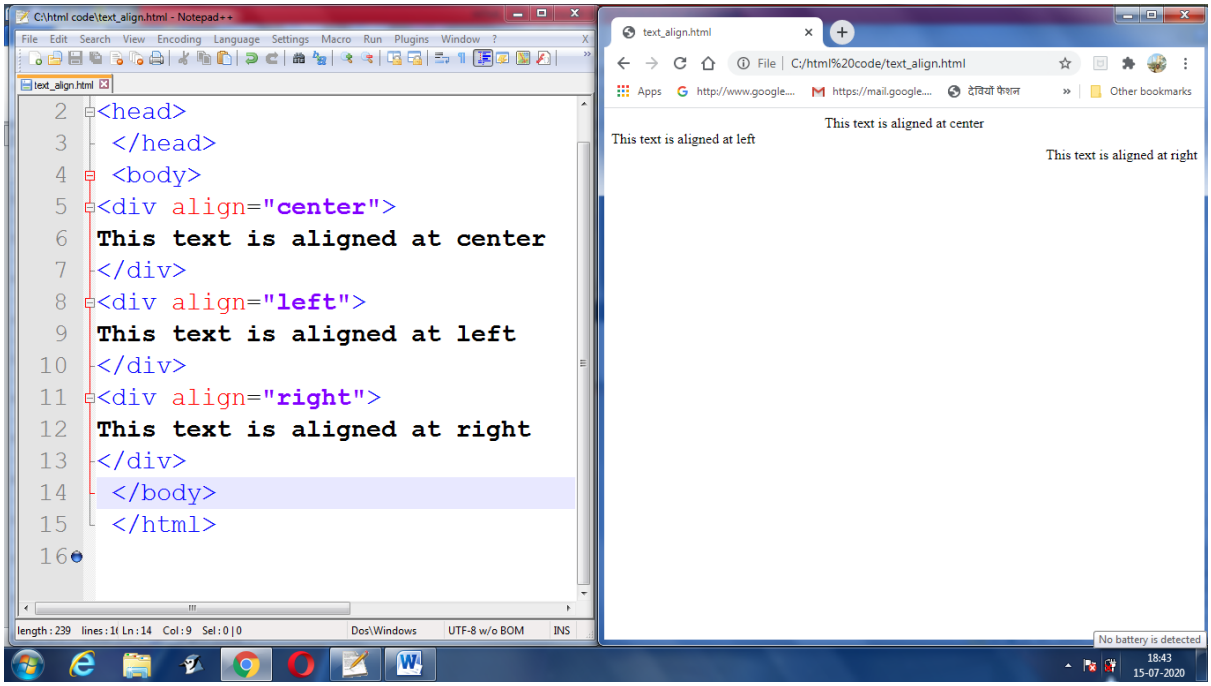
Setting font Style

We can set the text in bold face or Italics or a striked through text. The following example illustrates the same :



Text Alignment

We can align the text at left, right or at the center using a tag. Here is the example to illustrate this.



We can set bullets and numbers to the text.

HTML Unordered List | HTML Bulleted List

HTML Unordered List or Bulleted List displays elements in bulleted format . We can use unordered list where we do not need to display items in any particular order. The HTML ul tag is used for the unordered list. There can be 3 types of bulleted list:

- disc
- circle
- square

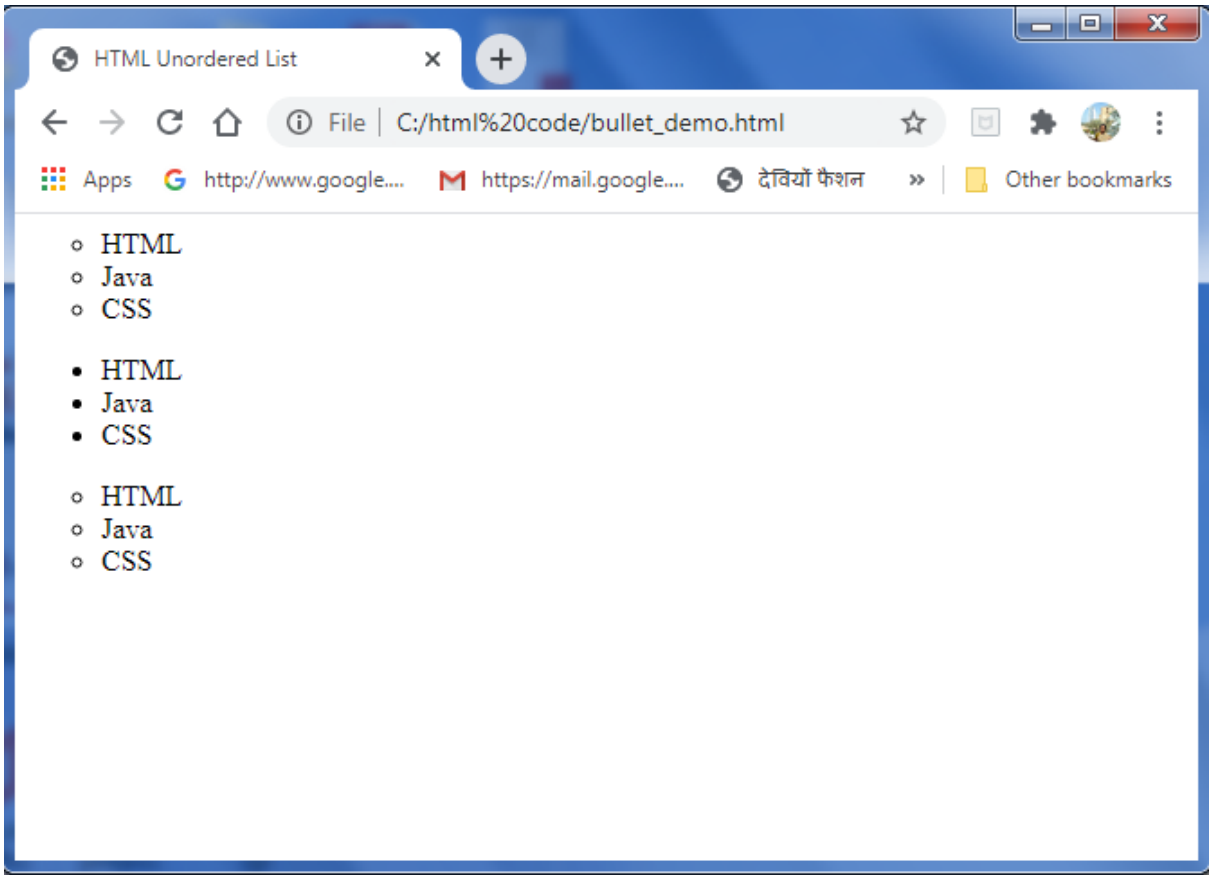
Following table shows various tags for this purpose –

Tags	Meaning
and	Beginning and end of the bulleted list.
and	Displays the bulleted list on separate line.
<ul Type = "circle">	Displays circular bullets
<ul Type = "disc">	Displays the solid round bullets
<ul Type = "square">	Displays the square bullets

Here is the example -

```
<html>
<head>
<title>HTML Unordered List</title>
 </head>
```

```
<body>
    <ul type = "circle">
      <li>HTML </li>
      <li>Java </li>
      <li>CSS</li>
    </ul>
<ul type = "disc">
      <li>HTML </li>
      <li>Java </li>
      <li>CSS</li>
    </ul>
          <ul type = "circle">
      <li>HTML </li>
      <li>Java </li>
      <li>CSS</li>
    </ul>
</body>
</html>
```



HTML Ordered List | HTML Numbered List

HTML Ordered List or Numbered List displays elements in numbered format using tag. We can use <ol> tag to represent items either in numerical order format or alphabetical order format, or any format . There can be different types of numbered list:

- Numeric Number (1, 2, 3)
- Capital Roman Number (I II III)
- Small Romal Number (i ii iii)
- Capital Alphabet (A B C)
- Small Alphabet (a b c)

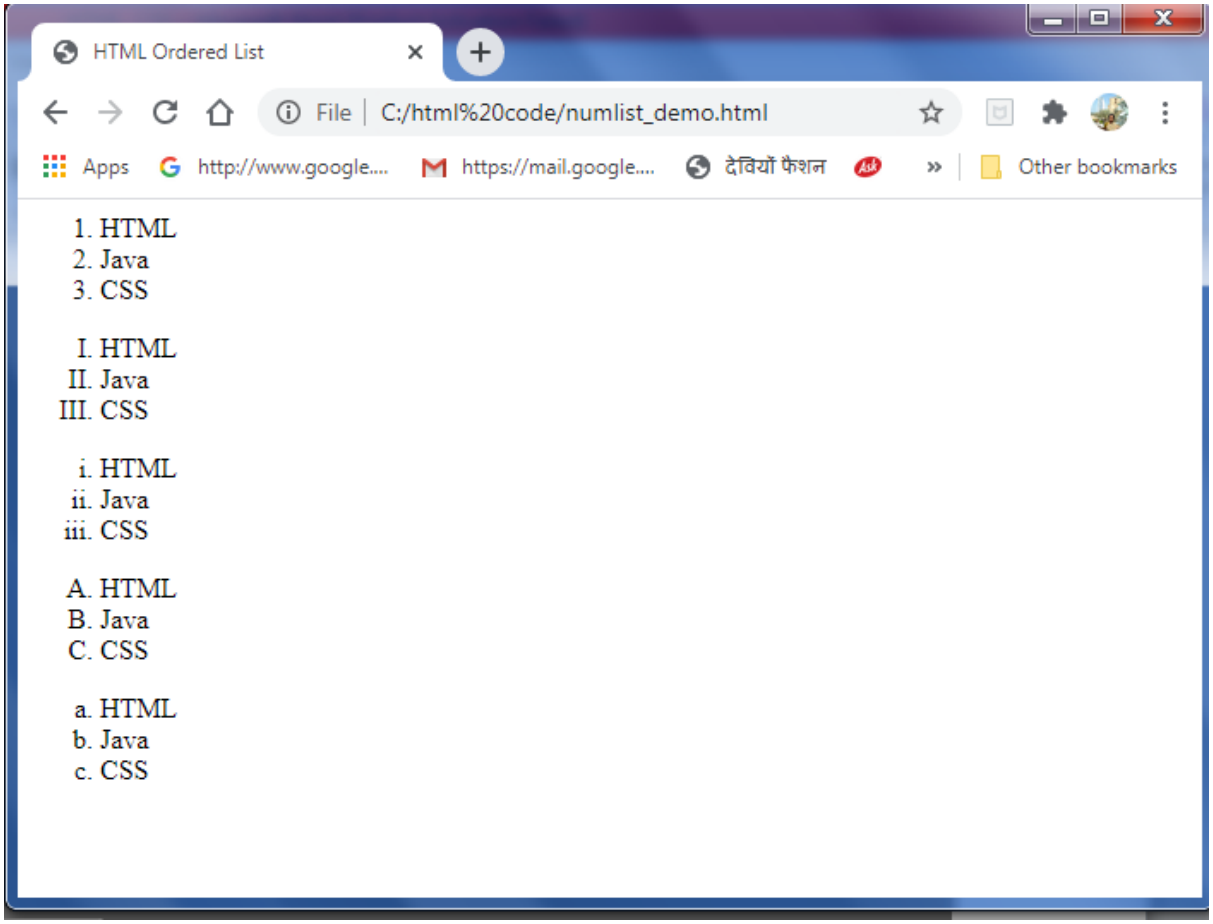
To represent different ordered lists, there are 5 types of attributes in <ol> tag.

Tags	**Meaning**
** and **	**Beginning and end of the numberd list.**
** and **	**Displays the numbered list on separate line.**
<ol type= "1"	**Displays the list in the following manner** **1.** **2.** **3.** **….**
<ol type= "I"	**Displays the list in the following manner** **I.** **II.** **III.** **….**
<ol type= "i"	**Displays the list in the following manner** **i.** **ii.** **iii.** **….**
<ol type= "A"	**Displays the list in the following manner** **A.** **B.** **C.** **….**
<ol type= "a"	**Displays the list in the following manner** **a.** **b.** **c.** **….**

Here is the Example –

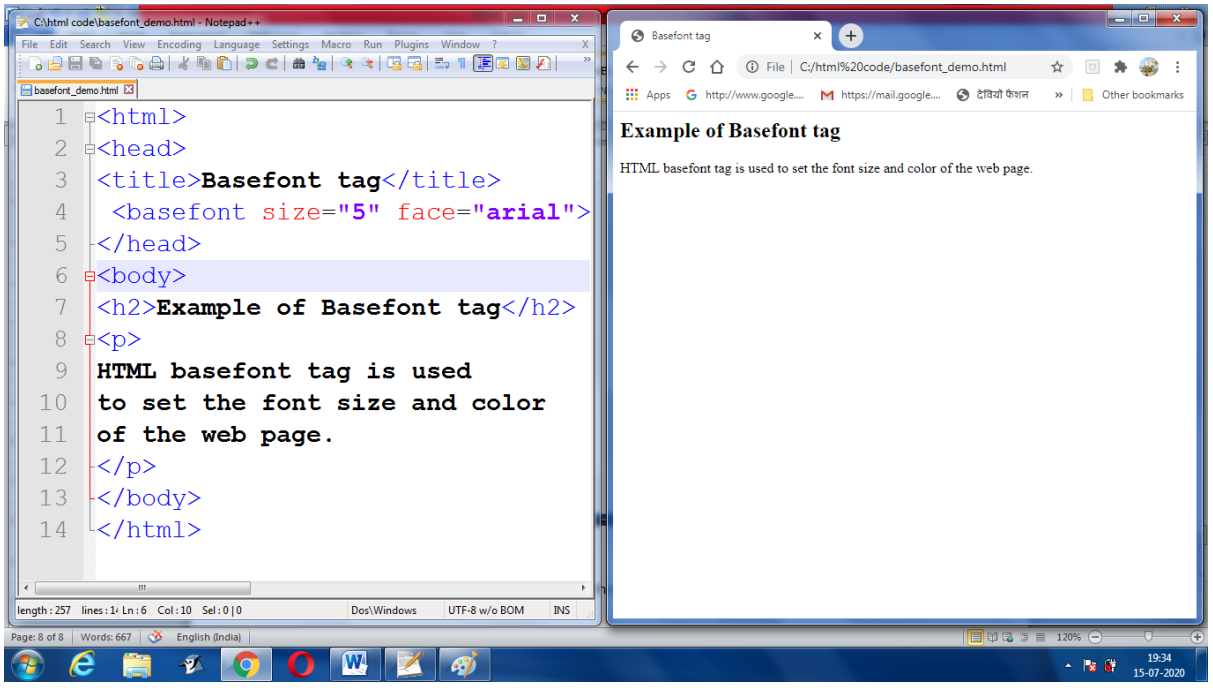
```
<html>
<head>
<title>HTML Ordered List</title>
 </head>
 <body>
 <ol>
 <li>HTML</li>
```

```
 <li>Java</li>
 <li>CSS</li>
</ol>

<ol type="I">
 <li>HTML</li>
 <li>Java</li>
 <li>CSS</li>
</ol>
            <ol type="i">
 <li>HTML</li>
 <li>Java</li>
 <li>CSS</li>
</ol>
<ol type="A">
 <li>HTML</li>
 <li>Java</li>
 <li>CSS</li>
</ol>
<ol type="a">
 <li>HTML</li>
 <li>Java</li>
 <li>CSS</li>
</ol>

 </body>
 </html>
```



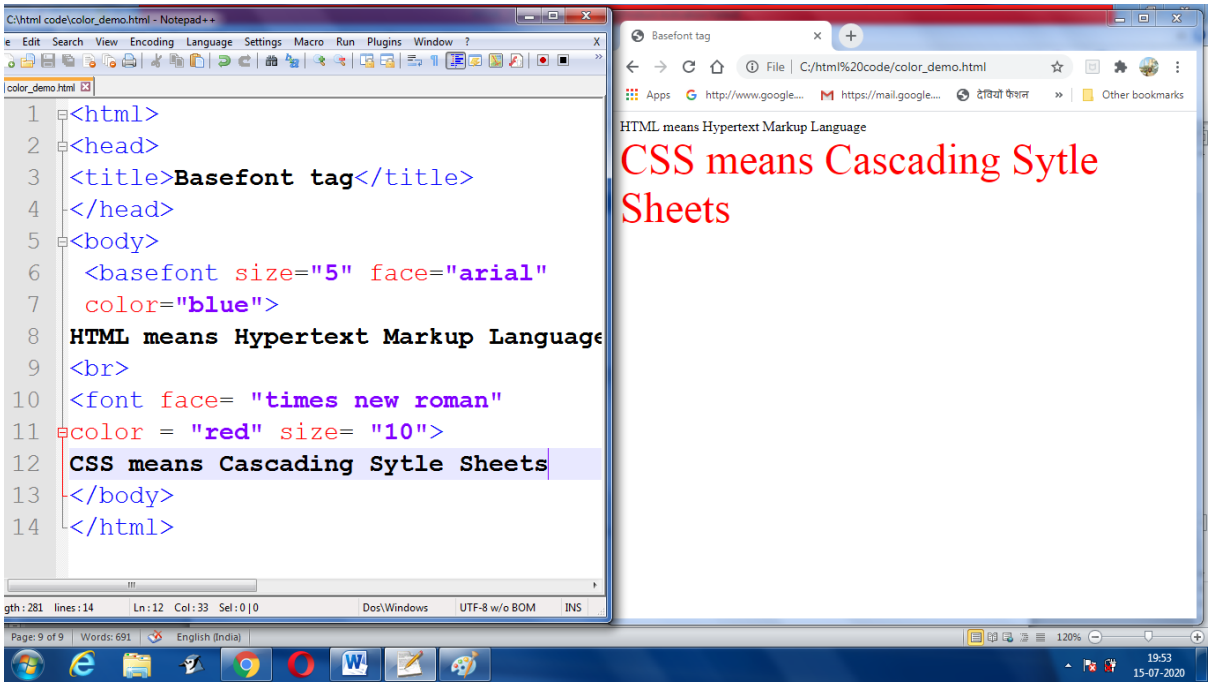
Setting the font

We can set the font size and color of the text in the web page by using <base font> tag.



Colors

Some times the web designers need to create colourful web pages. Using <basefont> tag we can display colourful text on the web pages.



In the above program we hve used <font face> tag to set the font. Using the attributes color and size we can specify the color and size in double quotes.

We can set the back ground color of the web page using the attribute **bgcolor**.

Following code sets the back ground color of the web page to red.

```
<html>
<body bgcolor= “#FF0000”>
</body>
</html>
```

Note that the background color is specified by the hexadecimal values. The following table shows the hex and corresponding decimal values –

Hex	Decimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

In HTML , the colors are specified by beginning with # and FF denotes 255. In this color code, first two digts specify the amount of red color then next two digits specify green and the last two digit specify blue color.

FF 00 00 → R G B

Hence is used to set the background color red. Each pair of digits specifies 0 to 255 color values. Thus Hexadecimal color coding helps to specify

256 * 256 * 256 = 16777216 colors.

The color can aslo be specified by its name as follows –

```
< html>
<body bgcolor = yellow>
</html>
```

LINKS

There is a common practice to specify the web link in the web page. Links are specified in HTML using the “<a>” tag. The following is the syntax :

Step 1: the beginning of the web link can be specified by the tag <a href = “ “ . The URL of the desired link is specified in double quotes.

```
<a href="url">link text</a>
```

Step 2: end the web link with

The most important attribute of the `<a>` element is the `href` attribute, which indicates the link's destination.

The following example illustrates this:

Example :

```
<html>
 <h3>Example Of Adding a link</h3>
   <body>
      <p>Click on the following link</p>
      <a href = "https://www.google.com">google</a>
   </body>

</html>
```

```
In the above program the word google will appear as a web link. If you
click on the  hyperlink then the home page of www.google.com will be
displayed.
```

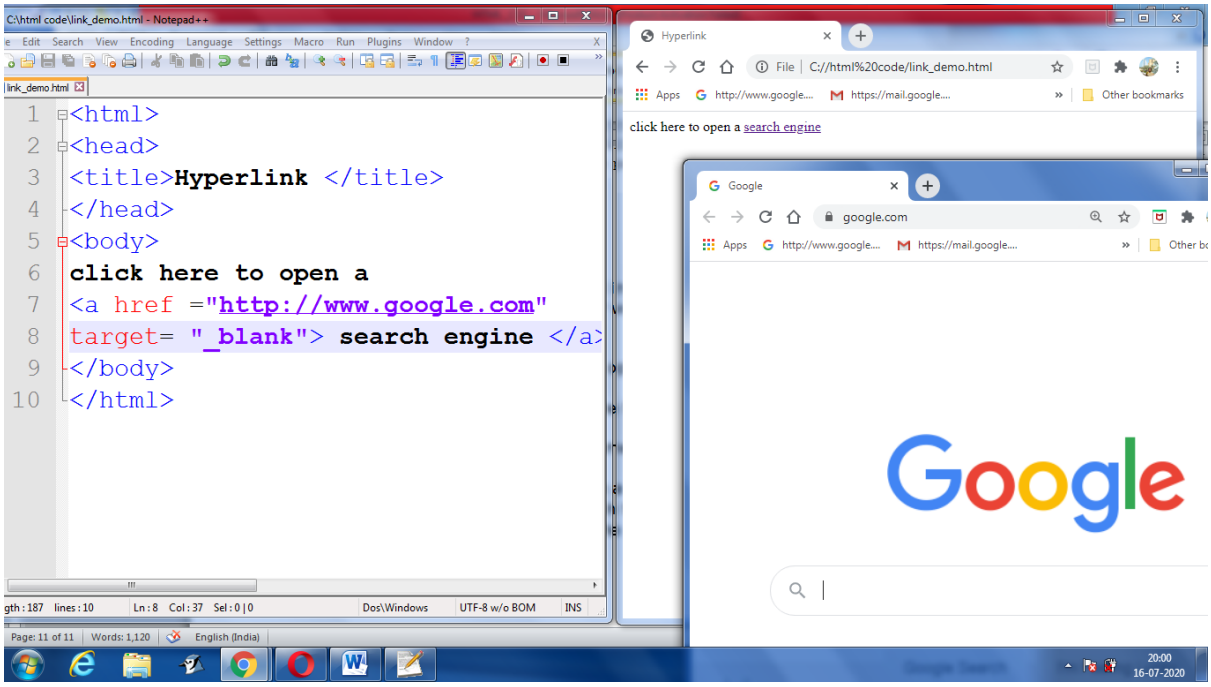
The target Attribute

By default, this web page gets opened in the current browser window. If you want to get the link opened in another window, you must specify another target for the link.

The `target` attribute specifies where to open the linked document.

The `target` attribute can have one of the following values:

- `_self` - Default. Opens the document in the same window/tab as it was clicked
- `_blank` - Opens the document in a new window or tab
- `_parent` - Opens the document in the parent frame
- `_top` - Opens the document in the full body of the window



Images

We can display images as a link. We have to use the <img src = " " > tag for this purpose. The location of image file should be mentioned in double quotes.

For example –

```
<html>
<head>
<title>image link </title>
</head>
<body>
<img src ="C:\Users\Public\Pictures\Sample Pictures\yellow.jpg"
alt="Oh! what a beautiful flower">
</body>
</html>
```

In the above program “Yellow.jpg” file is opened. Note that using “alt” we can display an alternative text on the image.

We can leave some specific amount of space around the image in the web browser by using Hspace and Vspace.

Example:

```
<html>
<head>
<title>image link </title>
</head>
<body>
<img src ="C:\Users\Public\Pictures\Sample Pictures\yellow.jpg" Hspace = “20” vspace= “20”
```

```
alt="Oh! what a beautiful flower">c
</body>
</html>
```

One can specify some text in the web browser along with the image, it can be done with the help of the following example

```
<html>
<head>
<title>image link </title>
</head>
<body>
<img src ="C:\Users\Public\Pictures\Sample Pictures\yellow.jpg" align= “right”>
This is such a beautiful flower
</body>
</html>
```

FORMS

HTML Forms are required to collect user input. The user input can then be sent to a server for processing.

For example, during user registration user need to enter information such as name, email address, credit card, etc.

The HTML `<form>` element defines a form that is used to collect user input:

Form elements are different types of input elements, like: text fields, checkboxes, radio buttons, submit buttons, and more.

The <input> Element

The `<input>` element is the most important form element. It is used to create form fields, to take input from user. We can apply different input filed to gather different information form user.

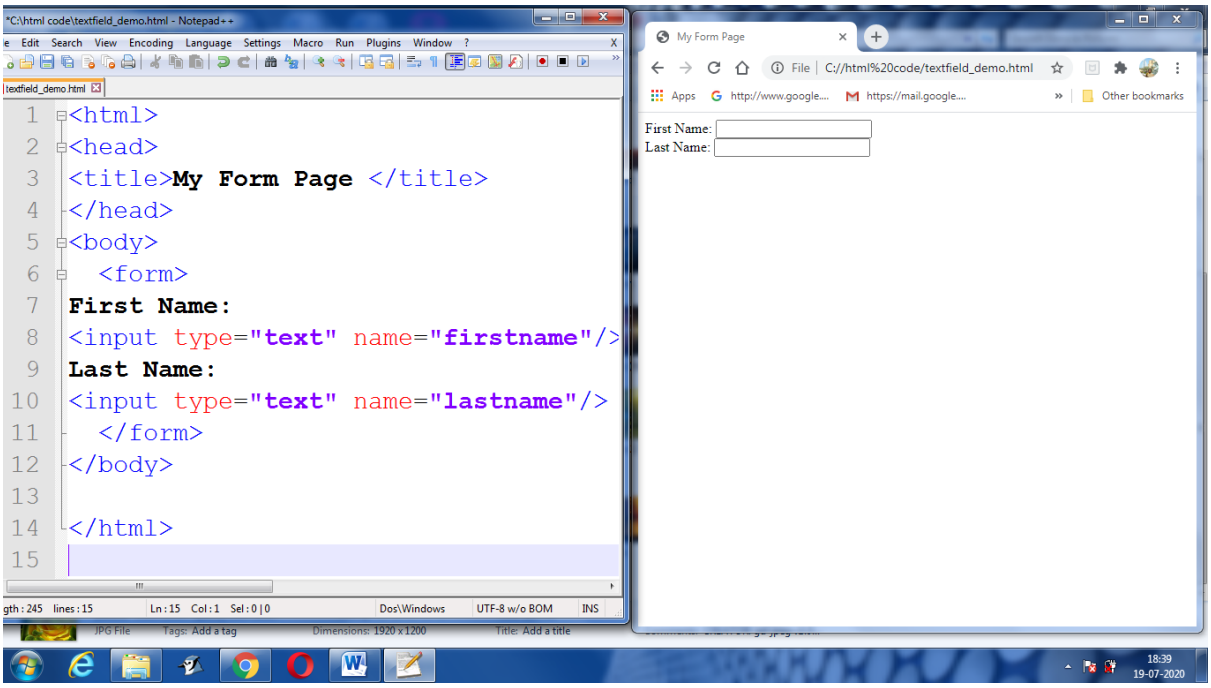
The `<input>` element is displayed in several ways, depending on the **type** attribute.

Text Fields

A text field typically Defines a single-line text input field. The type="text" attribute of input tag creates textfield control. The text field can be set using

```
<input type="text" size = 30, value= “ “ >
```

The input type is text and the value of the text field is “ “, that means the blank text field is displayed initially and we can enter the text of our choice into it. The size parameter restricts the length of the text to be entered in it.

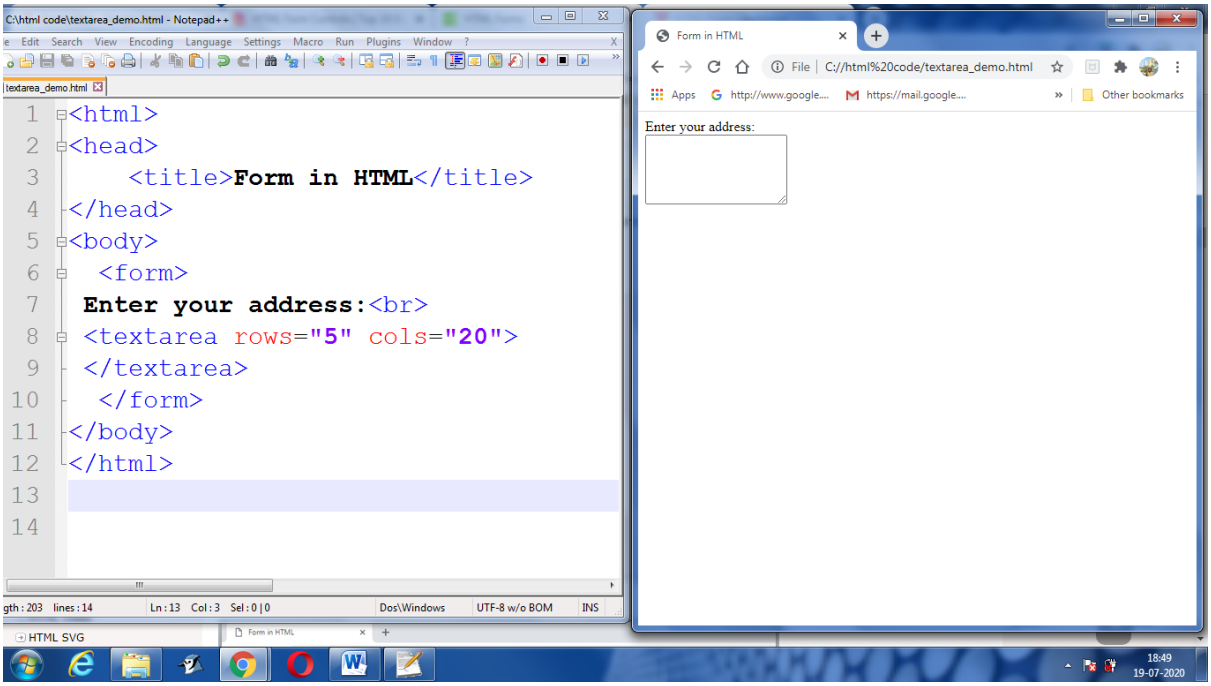


Some other parameters can be used with text field are:

- **maxlength** attribute that allows us to enter the text of some maximum length
- **name** attribute indicates name of the text field
- **align** attribute denotes the alignment of the text in the text field. The alignment can be right, left, top and bottom.

Text Area

The <textarea> tag in HTML is used to insert multiple-line text in a form. The size of <textarea> can be specify either using "rows" or "cols" attribute.



Various parameters that cab used for text area can be,

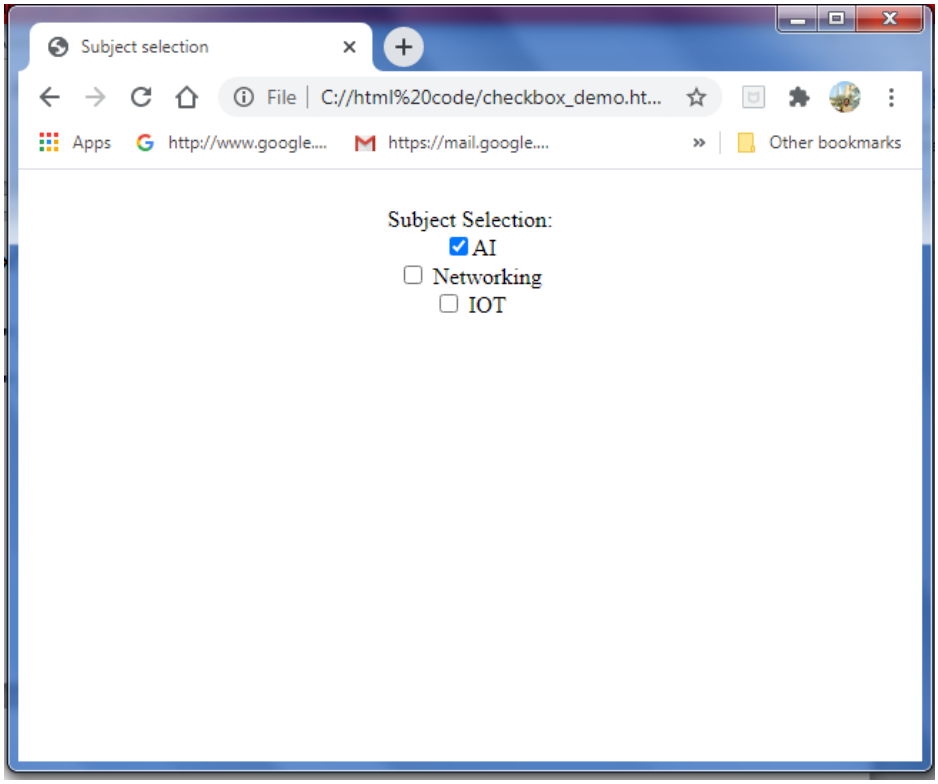
- **row** denotes total number of rows in the text area.
- **column** denotes total number of rows in the text area.
- **name** indicates the name of the text area
- **wrap** can be virtual or physical. If the wrap is virtual the line breaks gets disappeared when the text is actually submitted to the server. But if the wrap is physical the line breaks appears as it is in the text.

Check Box

The checkbox control is used to check multiple options from given checkboxes.

```
<html>
<html>
<head>
   <title>Subject selection </title>
</head>
<body>
 <form name="selectionForm">
<div align="center"> <br>
Subject Selection:<br>
<input type="checkbox" name="AI" value="Artificial Intelligence"checked/>AI <br>
<input type="checkbox" name="Net" value="Networking"/> Networking <br>
<input type="checkbox" name="IOT" value="IOT"/> IOT <br>
</div>
```

```
</form>
 </body>
</html>
```



Note: To set some check box in checked state, we need to set the parameter checked.

Radio Button

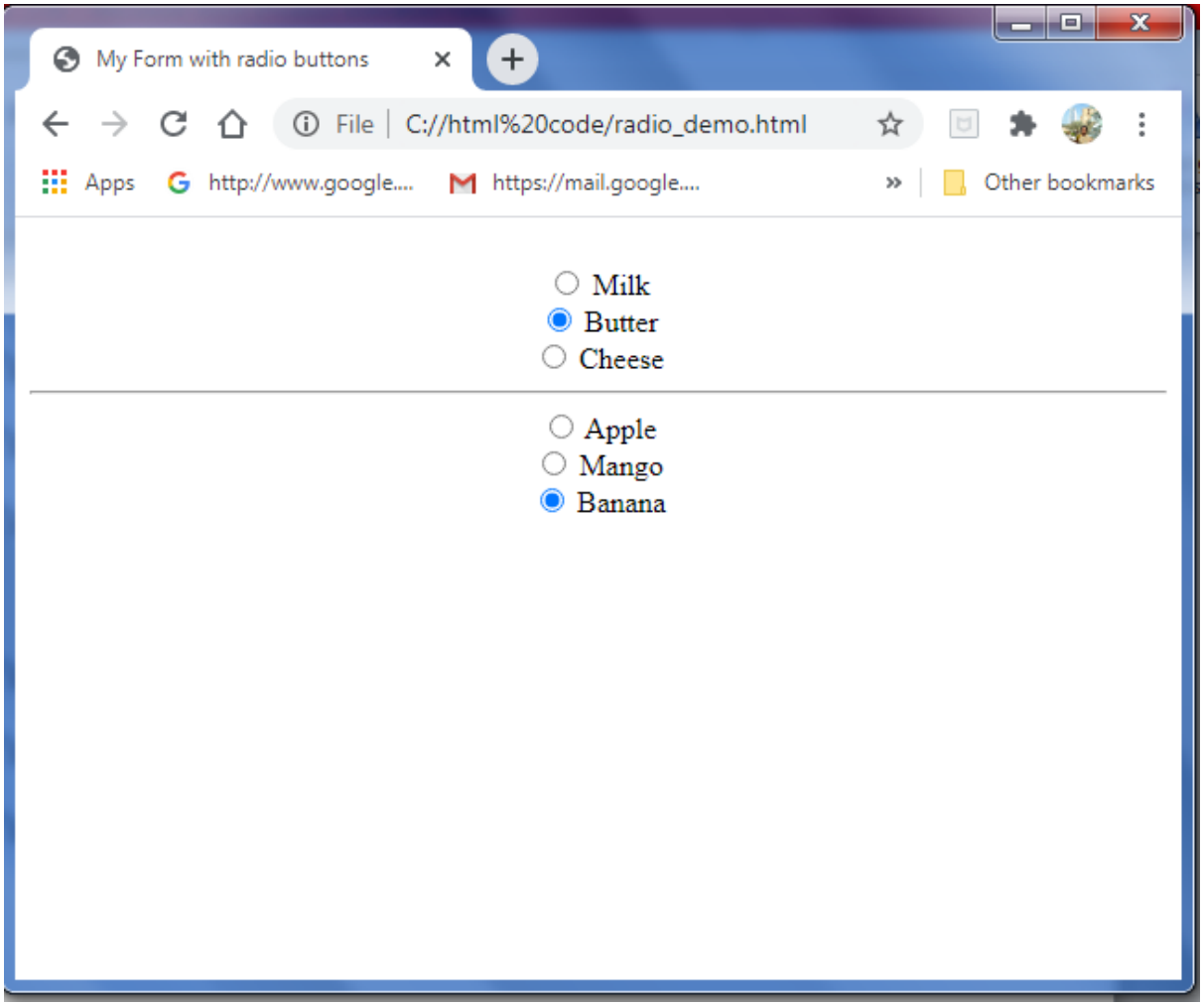
The radio button is used to select one option from multiple options. It is used for selection of gender, quiz questions etc.

If you use one name for all the radio buttons, only one radio button can be selected at a time. we can create a group of some radio button component by using

<input type = "radio>. Following program displays the radio buttons for two different groups.

```
<html>

<head>

<title>My Form with radio buttons</title>

</head>

<body>
```

```
<form name="myform" action="http://www.mydomain.com/myformhandler.cgi"
method="POST">

<div align="center"><br>

<input type="radio" name="group1" value="Milk"> Milk<br>

<input type="radio" name="group1" value="Butter" checked> Butter<br>

<input type="radio" name="group1" value="Cheese"> Cheese

<hr>

<input type="radio" name="group2" value="Water"> Apple<br>

<input type="radio" name="group2" value="Beer"> Mango<br>

<input type="radio" name="group2" value="Wine" checked> Banana<br>

</div>

</form>

</body>

</html>
```



Button

There are two types of buttons that can be created in HTML. Thety are

- Submit Button
- Reset Button

Submit Button

When a visitor clicks a submit button, the form is sent to the address specified in the action setting of the <form> tag.

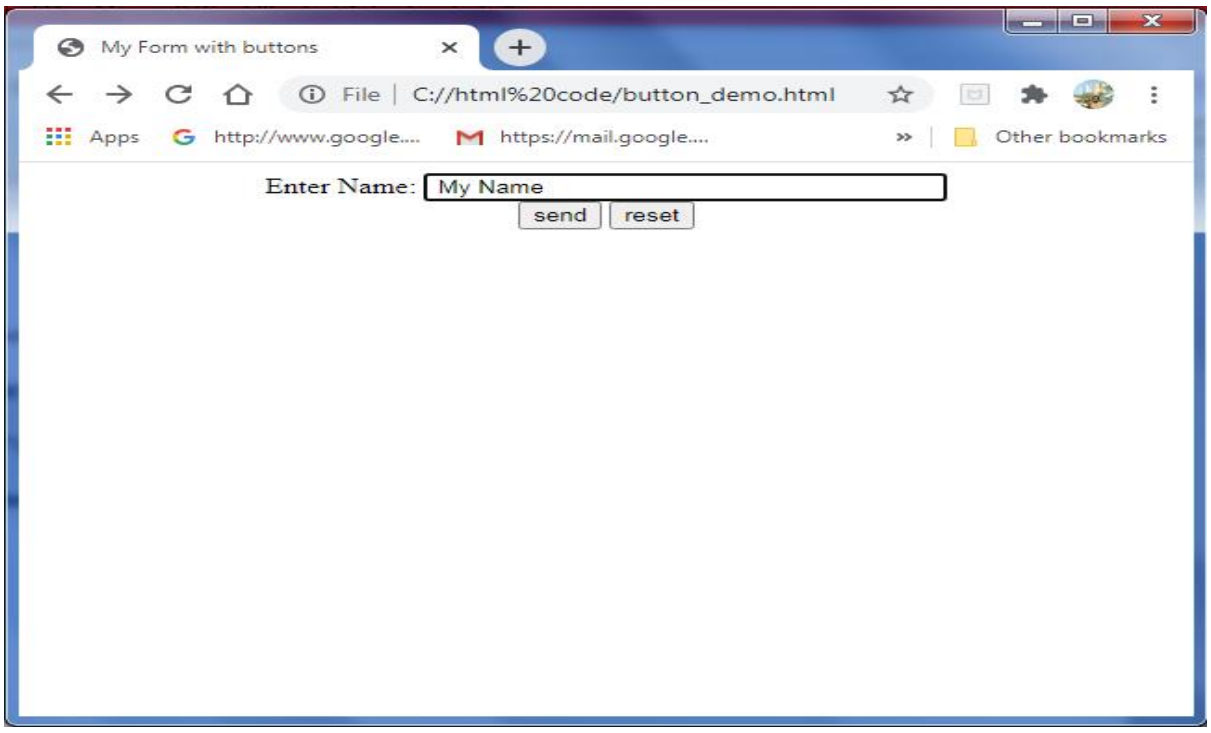
```
<input type="submit" value="submit" align =”center”>
```

Various parameters of submit button are:

- **Name** denotes the name of the submit button
- **Value** is for writing some text on the button
- **Align** specifies alignment of the button

<html>

```
<head>
<title>My Form with buttons</title>
</head>
<body>
<div align="center">
<form name = "myform" action=http://www.mydomain.com/test-button.cgi method="POST">
  Enter Name: <input type="text" size = 35 name= "name" ><br>
   <input type="submit" value="send">
   <input type="reset" value="reset">

          </form>
</div>
</body>
</html>
```



In the above program we have written

```
<form name = "myform" action=http://www.mydomain.com/test-button.cgi method="POST">
```

This line is specifying sending the contents to some specific location. There are two parameters: action and method. The action parameter indicates the address and the cgi script where the contents should go and the methods are the methods for submitting the data. The methods can be get or post.

Menus

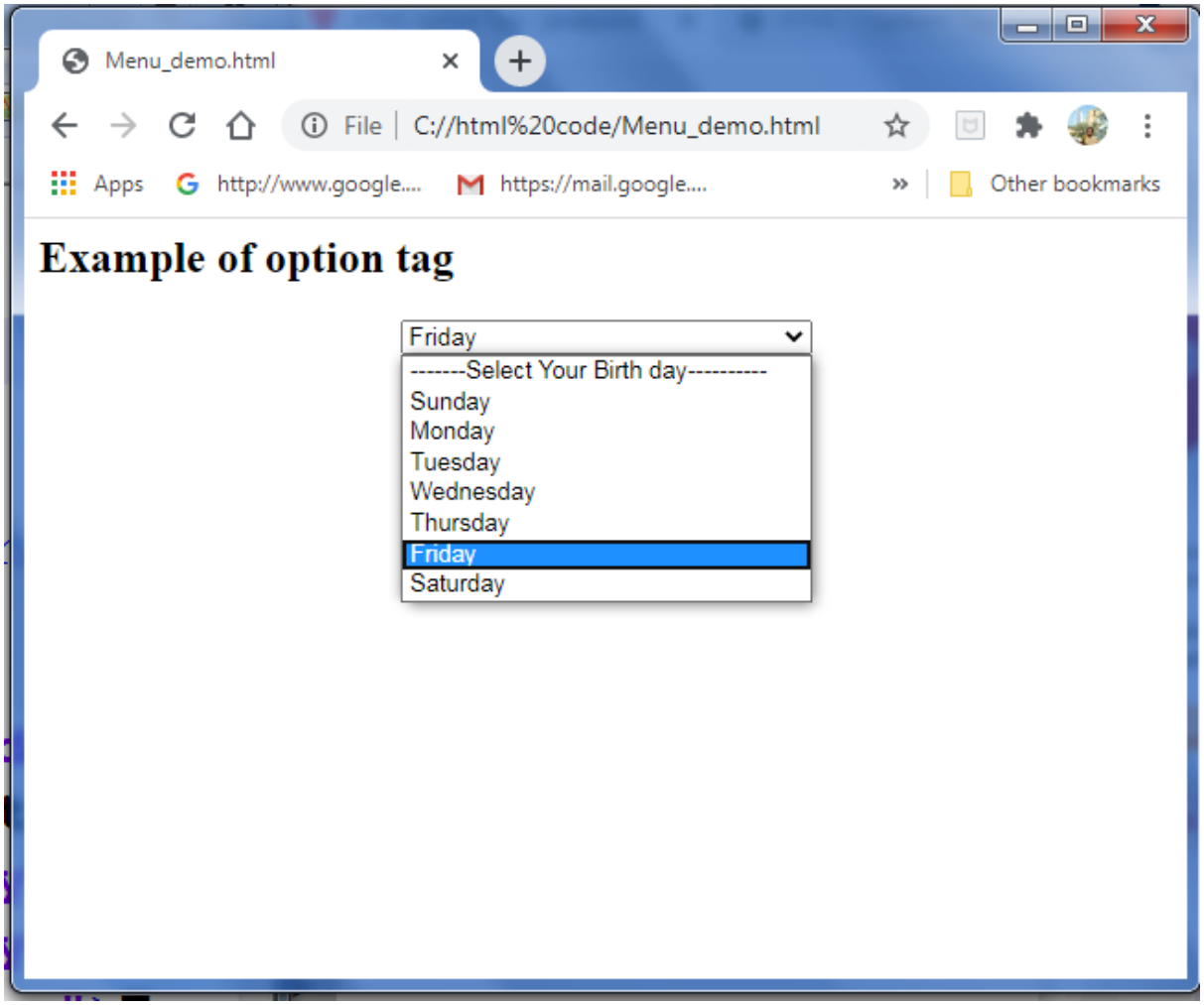
HTML allows to have pop down menu on the web page to select the desired option. This can be done by using HTML <option> tag is used to define options in a dropdown list within <select> or <datalist> element. A dropdown list must have at least one <option> element.

The related <option> of a dropdown list can be grouped using <optgroup> element which helps to understand a large list.

```
<html>

<head>

   <title></title>

   </head>

<body>

 <h2>Example of option tag</h2>

 <form Name= "My Form">

<div align = "center">

   <select name = "My Birthday">

      <option>-------Select Your Birth day----------</option>

      <option value="Sunday" >Sunday</option>

      <option value="monday">Monday</option>

      <option value="Tuesday">Tuesday</option>

      <option value="Wednesday">Wednesday</option>

      <option value="Thursday">Thursday</option>

                  <option value="Friday">Friday</option>

      <option value="Saturday">Saturday</option>

          </select>

</div>

 </form>
```

</body>

</html>



Note: We can make some specific value selected by **selected value=** . In the above program we can create a drop down menu by specifying the default day is **Sunday**. This can be done in the following way:

<option selected value="Sunday" >Sunday</option>

Sending Form

As we know that by specifying the form action and method we can send the data at the desired location. Here, we will see how to send an email to the form. For that, we will create a form on which the username , subject and message fields are placed. There should be some send button to send the data as email.

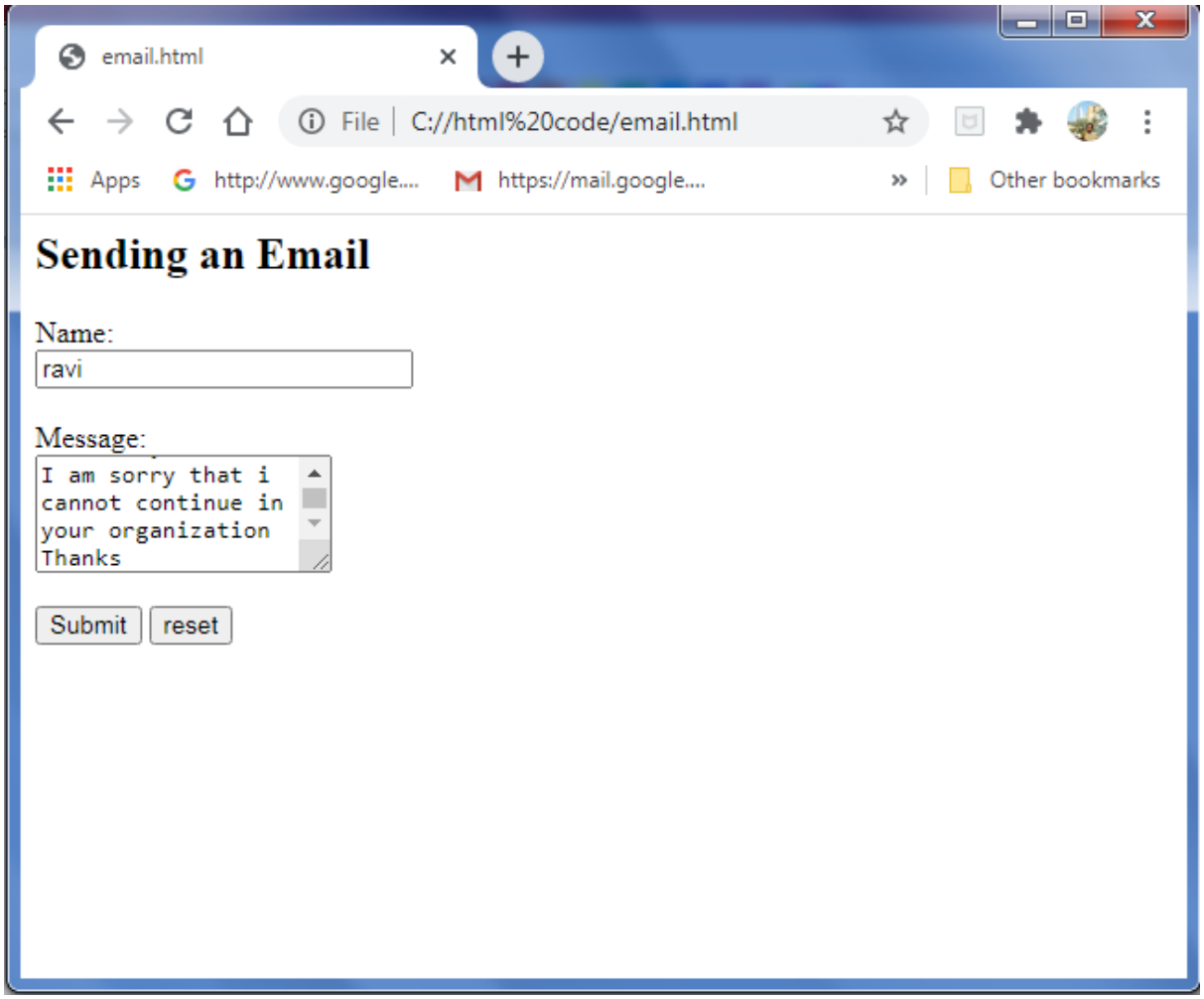
Note: some email client like outlook must be configured in your system

Here is the example HTML program :

<html>

<head>

```
   <title></title>
   </head>
<body>
 <h2>Sending an Email</h2>
<form action="mailto:contact@yourdomain.com"
method="POST"
enctype="text/plain"
name="EmailTestForm">
Name:<br>
<input type="text" size="24" name="VisitorName"><br><br>
Message:<br> <textarea name="VisitorComment" rows="4" cols="20">
</textarea><br><br>
<input type="submit" value="Submit">
<input type="reset" value="reset">
</form>
 </body>
</html>
```



In the above program we have specified action as

mailto:contact@yourdomain.com

and method as post. That means this form can be sent as an email to the email address mentioned by an action parameter. On the form we have placed on text field for storing name and a text area for storing message, a submit button to send the information as e-mail and the reset button to reset the previous data.

TABLES

HTML table allows you to arrange data into rows and columns. They are commonly used to display tabular data like product listings, customer's details, financial reports, and so on.

To create a table on the web page the `<table>` tag is used.

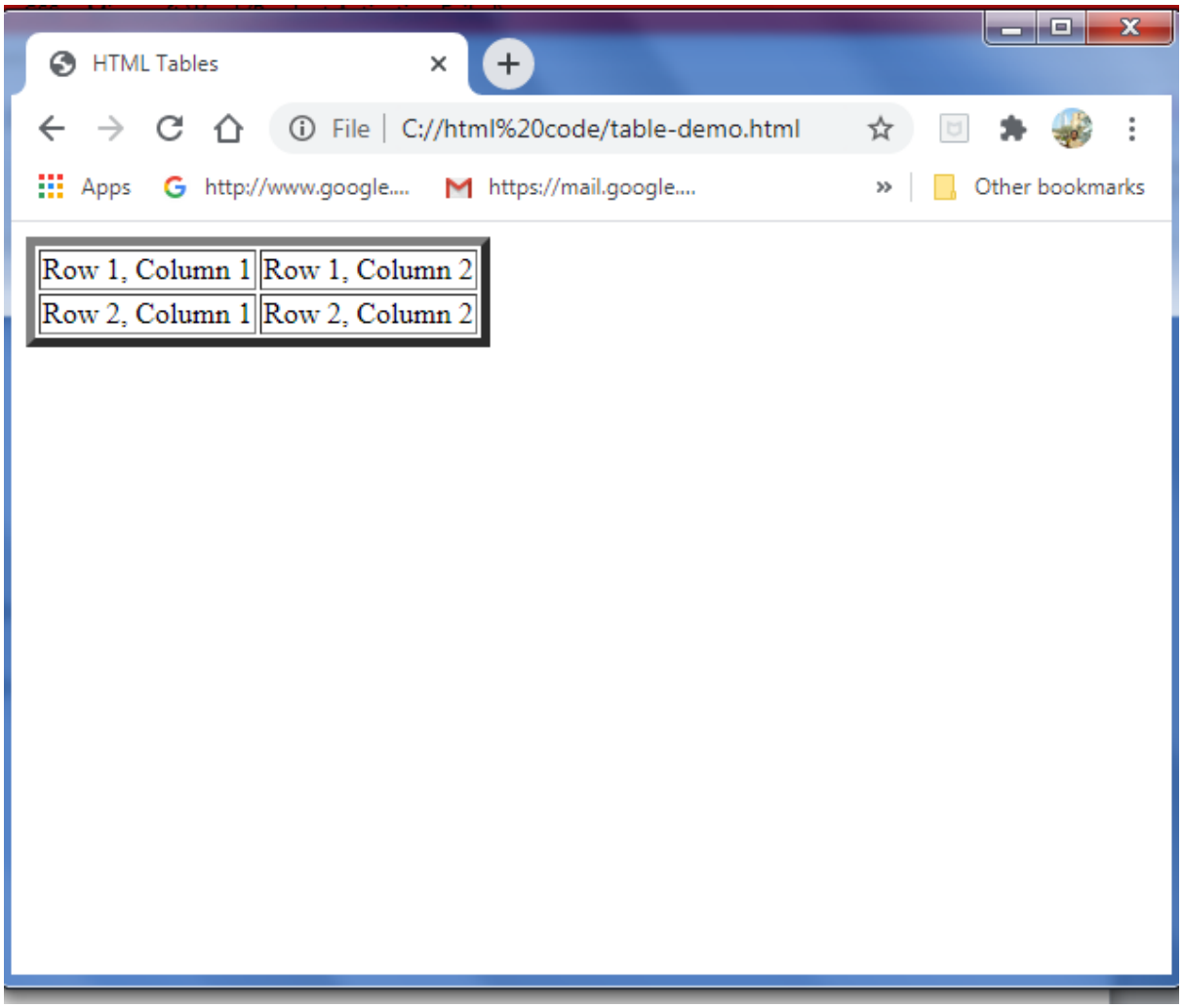
Each table row is defined with a `<tr>` tag. Each table header is defined with a `<th>` tag. Each table data/cell is defined with a `<td>` tag.

By default, the text in `<th>` elements are bold and centered.

By default, the text in `<td>` elements are regular and left-aligned.

The following example demonstrates the table tag:

```
<html>
  <head>
    <title>HTML Tables</title>
  </head>

  <body>
    <table border = "5">
      <tr>
        <td>Row 1, Column 1</td>
        <td>Row 1, Column 2</td>
      </tr>

      <tr>
        <td>Row 2, Column 1</td>
        <td>Row 2, Column 2</td>
      </tr>
    </table>
      </body>
</html>
```



In the above program the parameters are explained below

- **border= "5 "** is used to set the table border. You can give any value to set the desired border.
- **Caption** parameter is used to set the caption to the table.
- **Align** parameter is used to set the caption at the top or at the bottom

We can set the header to each column of the table by <th> tag. Here is the simple HTML program that adds the header to each column.

```
<html>
  <head>
    <title>HTML Tables</title>
  </head>
<center>
    <table border="3">
<caption align="bottom">
```

 Table Demo

<tr><th>First_Name</th><th>Last_Name</th><th>Marks</th></tr>

<tr><td>Ravi</td><td>Shankar</td><td>60</td></tr>

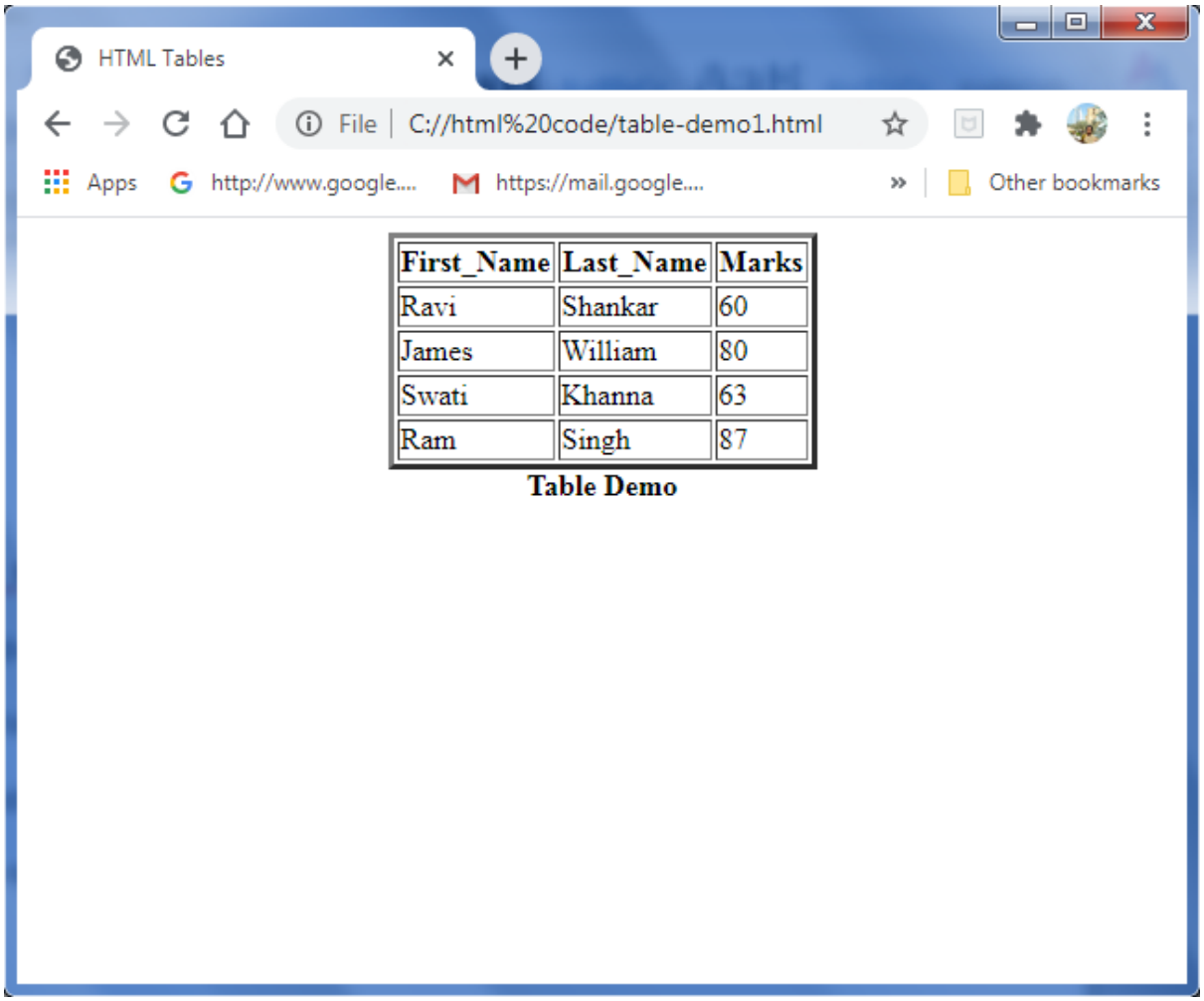
<tr><td>James</td><td>William</td><td>80</td></tr>

<tr><td>Swati</td><td>Khanna</td><td>63</td></tr>

<tr><td>Ram</td><td>Singh</td><td>87</td></tr>

</table>

</html>



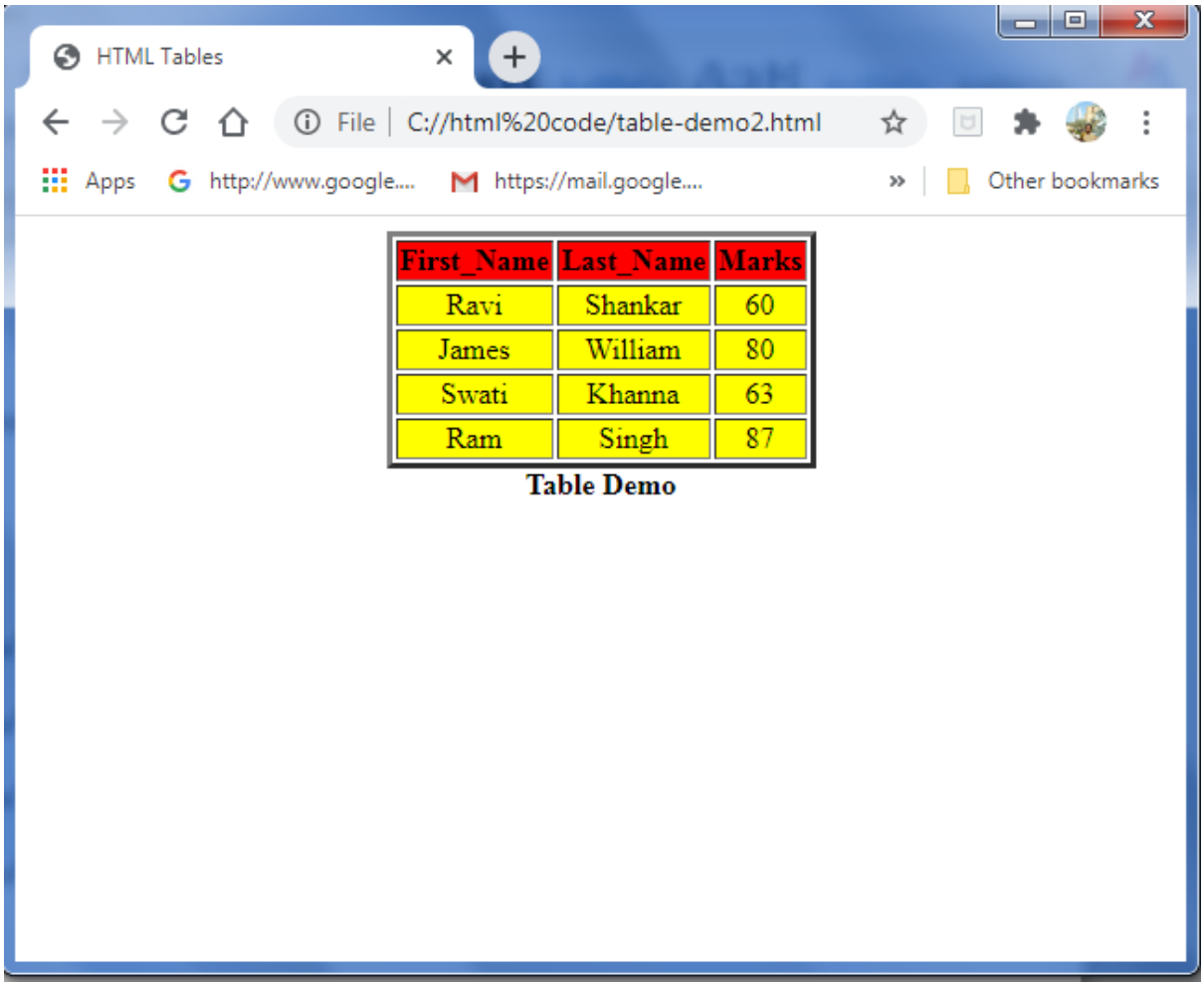
We can further decorate our table by setting the back ground of the table. the back ground can be colour or can be an image.

<html>

<head>

<title>HTML Tables</title>

```
  </head>
<center>
    <table border="3">
<caption align="bottom">
<b> Table Demo </b>
<tr align = "center"><th bgcolor="red">First_Name</th>
                        <th bgcolor="red">Last_Name</th>
                        <th bgcolor="red">Marks</th></tr>
<tr align = "center" bgcolor="yellow" ><td >Ravi</td><td>Shankar</td><td>60</td></tr>
<tr align = "center" bgcolor="yellow"><td>James</td><td>William</td><td>80</td></tr>
<tr align = "center" bgcolor="yellow"><td>Swati</td><td>Khanna</td><td>63</td></tr>
<tr align = "center" bgcolor="yellow"><td>Ram</td><td>Singh</td><td>87</td></tr>
</table>
</center>
</html>
```



In the above program

<tr align = "center" bgcolor="yellow" > specifies that the data in each row is aligned center by using **align=”center”** parameter. In the same way we can align the data **right** or **left** also. Y The background color of each row is set with yellow by using **bgcolor** tag.

Colspan and rowspan of tables

Some times we may require to add sub-columns and sub-rows to categorise the information properly. To do this colspan or rowspan can be used. Colspan will divide one cell/row into multiple columns, and the number of columns depend on the value of colspan attribute. rowspan will divide a cell into multiple rows. The number of divided rows will depend on rowspan values.

For example when colspan=2 then the column can be extended horizontally by to cells.

<td> First <td> Second

<td colspan =2> Third

First	Second
Third	

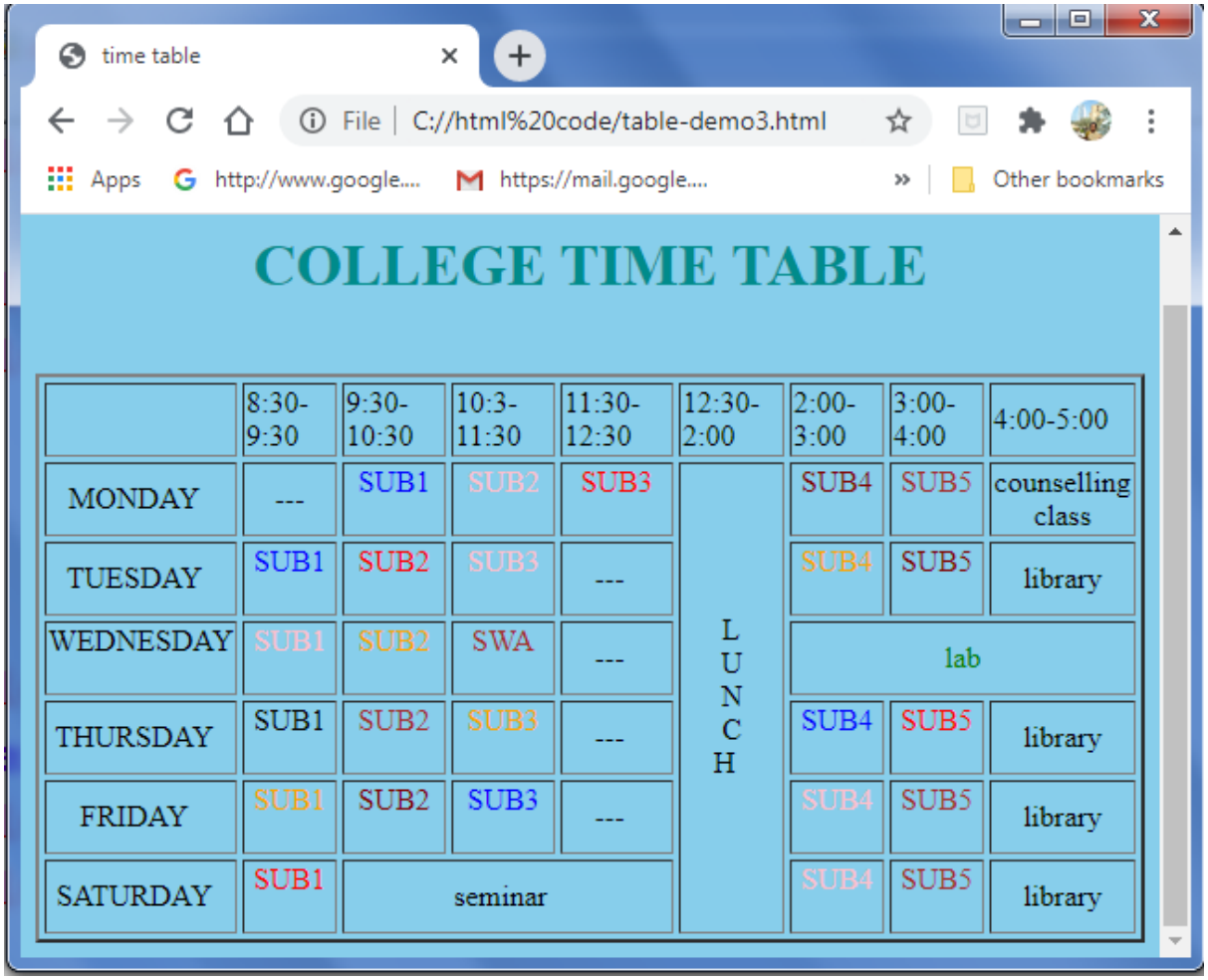
In the similar way rowspan is also used.

Here is simple HTML program(display timetable) to demonstrate rowspan and colspan.

```
<html>
<head>
  <title>time table</title>
</head>
<body bgcolor="skyblue">
<H1><FONT COLOR="DARKCYAN"><CENTER>COLLEGE TIME TABLE</FONT></H1>
<table border="2" cellspacing="3" align="center">
<tr>
 <td align="center">
 <td>8:30-9:30
 <td>9:30-10:30
 <td>10:3-11:30
 <td>11:30-12:30
 <td>12:30-2:00
 <td>2:00-3:00
 <td>3:00-4:00
 <td>4:00-5:00
</tr>
<tr>
 <td align="center">MONDAY
 <td align="center">---<td align="center"><font color="blue">SUB1<br>
 <td align="center"><font color="pink">SUB2<br>
 <td align="center"><font color="red">SUB3<br>
 <td rowspan="6"align="center">L<br>U<br>N<br>C<br>H
 <td align="center"><font color="maroon">SUB4<br>
```

```
 <td align="center"><font color="brown">SUB5<br>
 <td align="center">counselling class
</tr>
<tr>
 <td align="center">TUESDAY
 <td align="center"><font color="blue">SUB1<br>
 <td align="center"><font color="red">SUB2<br>
 <td align="center"><font color="pink">SUB3<br>
 <td align="center">---
 <td align="center"><font color="orange">SUB4<BR>
 <td align="center"><font color="maroon">SUB5<br>
 <td align="center">library
</tr>
<tr>
 <td align="center">WEDNESDAY
 <td align="center"><font color="pink">SUB1<br>
 <td align="center"><font color="orange">SUB2<BR>
 <td align="center"><font color="brown">SWA<br>
 <td align="center">---
 <td colspan="3" align="center"><font color="green"> lab
</tr>
<tr>
 <td align="center">THURSDAY
 <td align="center">SUB1<br>
 <td align="center"><font color="brown">SUB2<br>
 <td align="center"><font color="orange">SUB3<BR>
 <td align="center">---
 <td align="center"><font color="blue">SUB4<br>
```

```
 <td align="center"><font color="red">SUB5<br>
 <td align="center">library
</tr>
<tr>
 <td align="center">FRIDAY
 <td align="center"><font color="orange">SUB1<BR>
 <td align="center"><font color="maroon">SUB2<br>
 <td align="center"><font color="blue">SUB3<br>
 <td align="center">---
 <td align="center"><font color="pink">SUB4<br>
 <td align="center"><font color="brown">SUB5<br>
 <td align="center">library
</tr>
<tr>
 <td align="center">SATURDAY
 <td align="center"><font color="red">SUB1<br>
 <td colspan="3" align="center">seminar
 <td align="center"><font color="pink">SUB4<br>
 <td align="center"><font color="brown">SUB5<br>
 <td align="center">library
</tr>
</body>
</html>
```



JAVA SCRIPT

Basics of Java Script

HTML itself have some limitations, one of it is the interactivity. HTML can not create interactive web pages. Hence Java Script is designed to add the interactivity in HTML pages.

We will discuss how to create first Java Script program.

Let us write some javascript by which some message is displayed on the web page.

<HTML>

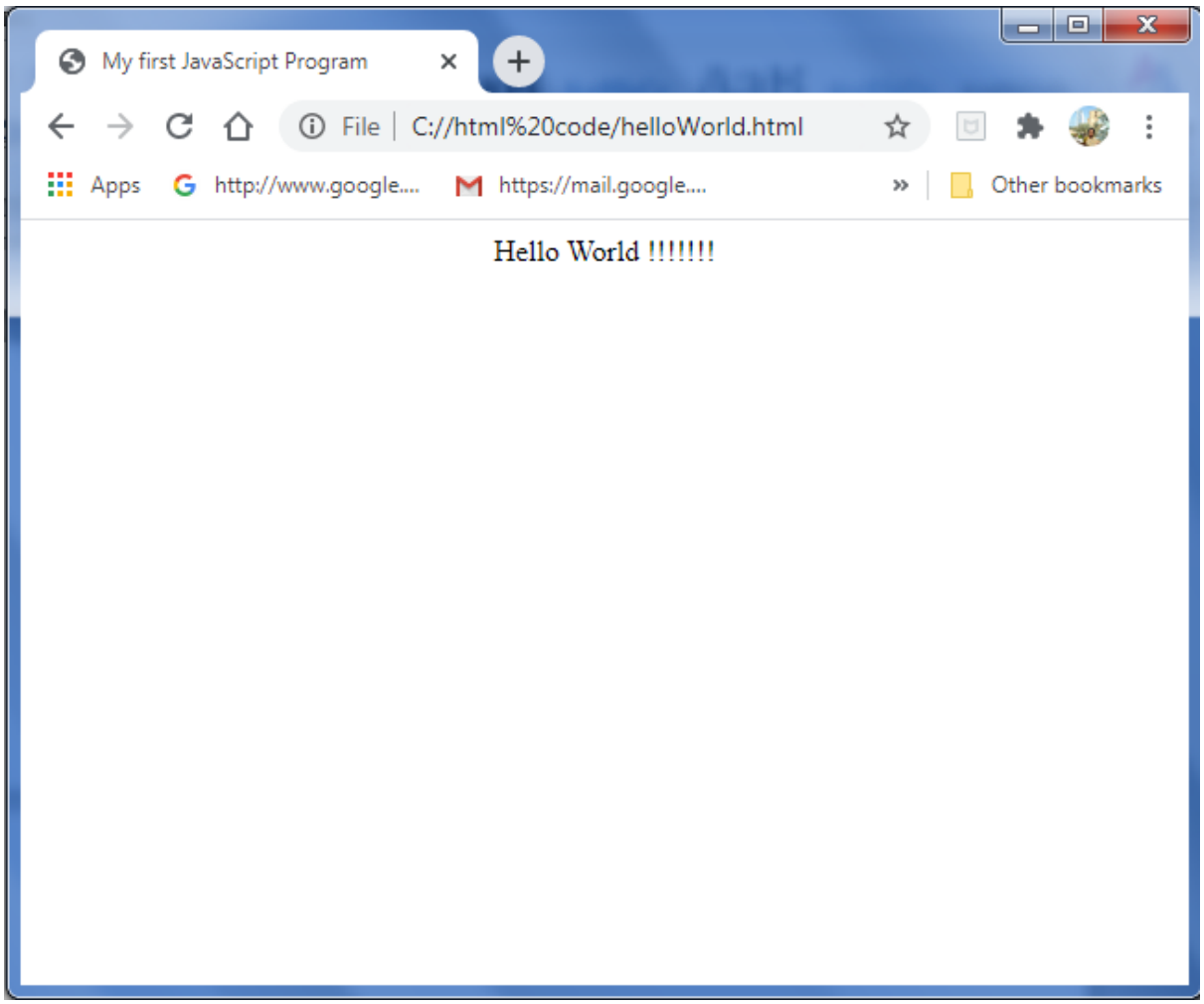
<head>

<TITLE>My first JavaScript Program</TITLE>

</head>

<Body>

```
<SCRIPT type = "text/JavaScript">
   document.write("Hello World !!!!!!!");
</SCRIPT>
</center>
</Body>
</HTML>
```



Similar to HTML , Javascript program also have two sections head and body. All the tags which are used in HTML can be used along with Javascript.

Here comes and important part -

<SCRIPT type = "text/JavaScript">

This line tells the web browser that is Javascript

   document.write("Hello World !!!!!!!");

The document.write is used to display some text on the web page.

Note : all the Javascript code must be enclosed with in <script> and </script> tags.

Comments

Comments in JavaScript are used to **explain** the code and make the program more **readable** for developers.

The Java Script allows two types of comments

- **single-line** comments (which comment out one line or a part of one line)

It is represented by double forward slashes (//). It can be used before and after the statement.

```
<script>
// It is single line comment
document.write("hello javascript");
</script>
```

- **multi-line** comments (which comment out a block of code).

It can be used to add single as well as multi line comments. So, it is more convenient.

It is represented by forward slash with asterisk then asterisk with forward slash. For example:

```
<script>
/* It is multi line comment.
It will not be displayed */
</script>
```

Variables

Now let us understand how to handle variables in java script.

Variables are named values and can store any type of JavaScript value.

Here's how to declare a variable:

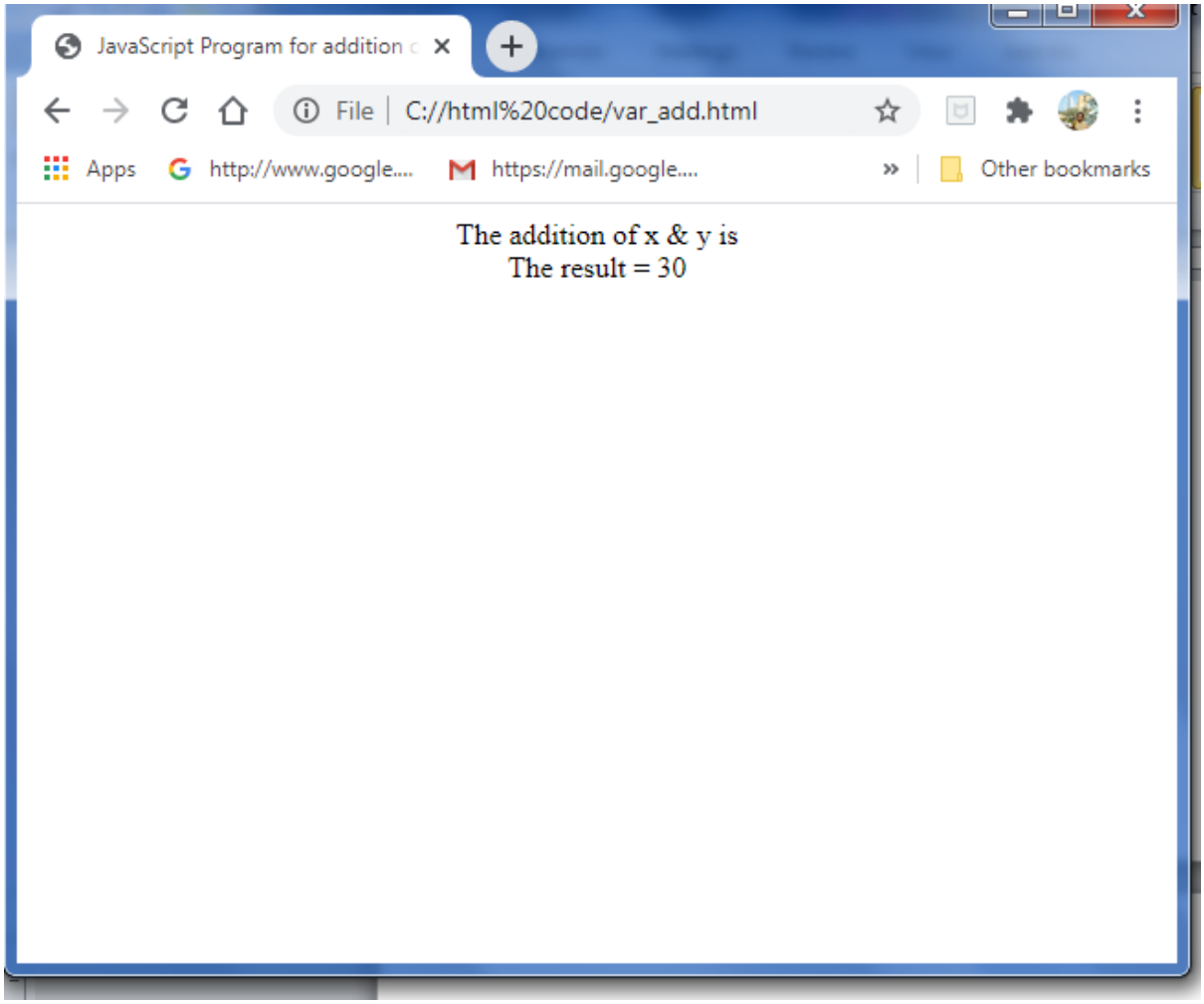
EXAMPLE

```
var a = 10;
```

Here's the explanation of above example :

- var is the keyword that tells JavaScript you're declaring a variable.
- a is the name of that variable.
- = is the operator that tells JavaScript a value is coming up next.
- 10 is the value for the variable to store.

```
<HTML>
<head>
<TITLE>JavaScript Program for addition of varibles</TITLE>
</head>
<Body>
<center>
<SCRIPT>
var x,y,z;
var string ;
x=10;
y=20;
z=x+y;
string = " The result = ";
  document.write("The addition of x & y is "+"<br>");
document.write(string);
document.write(z);
</SCRIPT>
</center>
</Body>
</HTML>
```



In the above program we have used the
 tag in document.write to add new line. Hence on the output result will be displayed on the new line.

Note that javascript is case sensitive language. If you declare one variable as sum and another variable as SUM , both treated as different variables.

JavaScript Operators

An operator performs some operation on single or multiple operands and produces a result.

JavaScript operators are classified as:

*** Arithmetic**

*** Logical**

*** Comparison**

*** String**

The following table shows the list of operators used in JavaScript:

Category	Operator	Name/Description	Example	Result
Arithmetic	+	Addition	3+2	5
	-	Subtraction	3-2	1
	*	Multiplication	3*2	6
	/	Division	10/5	2
	%	Modulus	10%5	0
	++	Increment and then return value	X=3; ++X	4
		Return value and then increment	X=3; X++	3
	--	Decrement and then return value	X=3; --X	2
		Return value and then decrement	X=3; X--	3
Logical	&&	Logical "and" evaluates to true when both operands are true	3>2 && 5>3	False
	||	Logical "or" evaluates to true when either operand is true	3>1 || 2>5	True
	!	Logical "not" evaluates to true if the operand is false	3!=2	True
Comparison	==	Equal	5==9	False
	!=	Not equal	6!=4	True
	<	Less than	3<2	False
	<=	Less than or equal	5<=2	False
	>	Greater than	4>3	True
	>=	Greater than or equal	4>=4	True
String	+	Concatenation(join two strings together)	"A"+"BC"	ABC

Control Structures

A **control structure**, refers to the flow of execution of the program.

Commonly used control structures in java script are –

1. conditional statements (also called conditional logic)
2. looping structures
3. Labels

1. Conditional Statements

Conditional statements are used to decide the flow of execution based on different conditions. If a condition is true, you can perform one action and if the condition is false, you can perform another action.

In Java Script there are two types of conditional statements are used –

- **If Statement**
- **Switch Statement**

If Statement

There are mainly three types of conditional statements in JavaScript.

1. If statement
2. If…Else statement
3. If…Else If…Else statement

1. If statement

Syntax:

```
if (condition)

{

Statements if the condition is true

}
```

You can use If statement if you want to check only a specific condition.

2. If…Else statement

Syntax:

```
if (condition)

{

Statements if the condition is true

}

else

{

Statements if the condition is false

}
```

You can use If….Else statement if you have to check two conditions and execute a different set of codes.

3. If…Else If…Else statement

Syntax:

```
if (condition1)
```

```
{

Statements if the condition1 is true

}

else if(condition2)

{

Statements if the condition2 is true

}

else

{

Statements if both the conditions are false

}
```

You can use If….Else If….Else statement if you want to check more than two conditions.

Here. Is the example to demonstrate if-else-if

<HTML>

<head>

<TITLE>JavaScript Program to display grades of student</TITLE>

</head>

<Body>

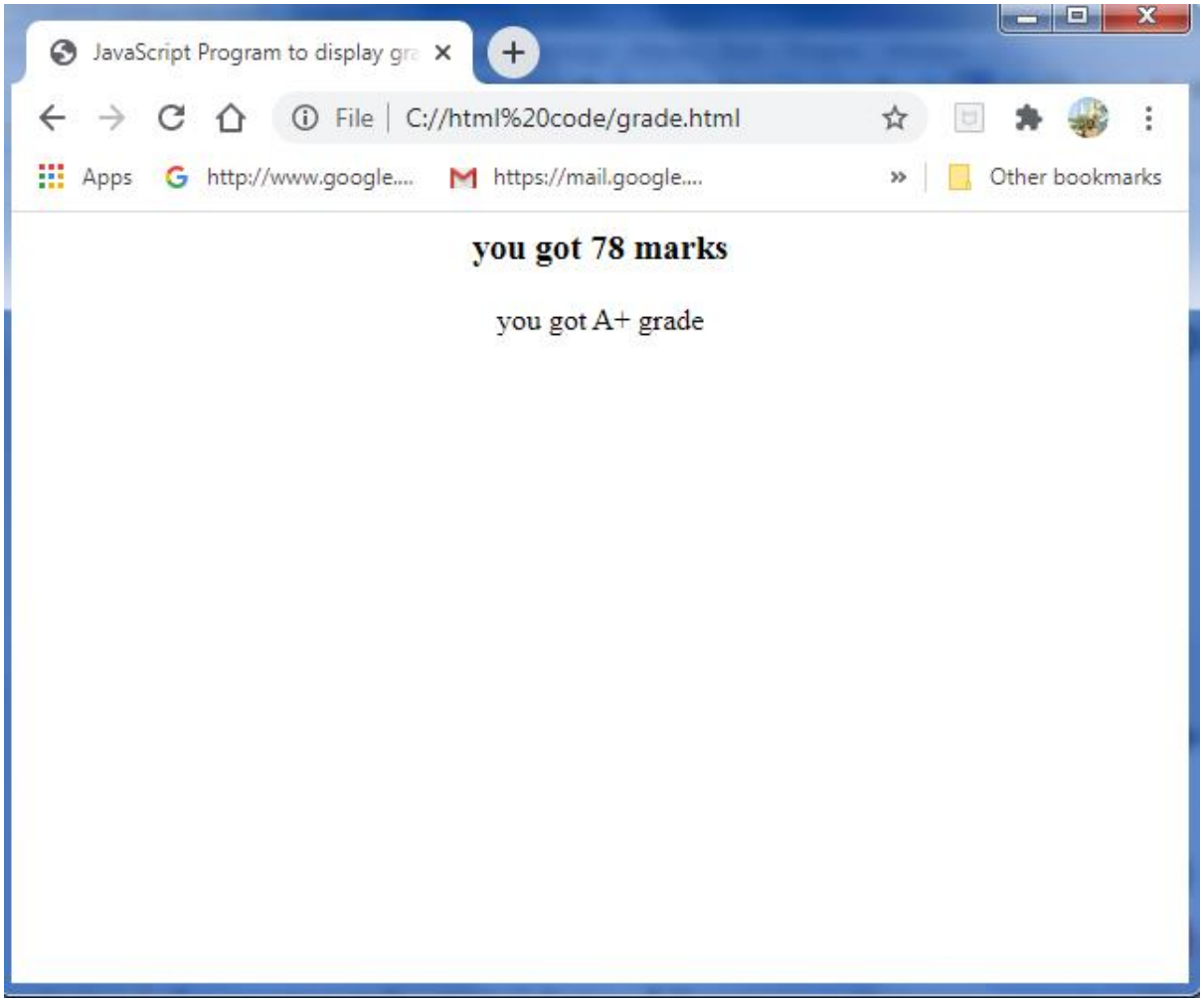
<SCRIPT>

var marks = 78;

document.write ("<h3> you got " + marks + "marks"+ "
</h3>");

if (marks < 40)

```
document.write ("you are failed");
else if (marks >= 40 && marks<50)
document.write ("you  have got C grade");
else if (marks >= 50 && marks<60)
document.write ("you  have got B grade ");
else if (marks >= 60 && marks<70)
document.write ("you  have got A grade");
else
document.write ("you got A+ grade ");
</SCRIPT>
</center>
</Body>
</HTML>
```



4. Nested If statement

Syntax

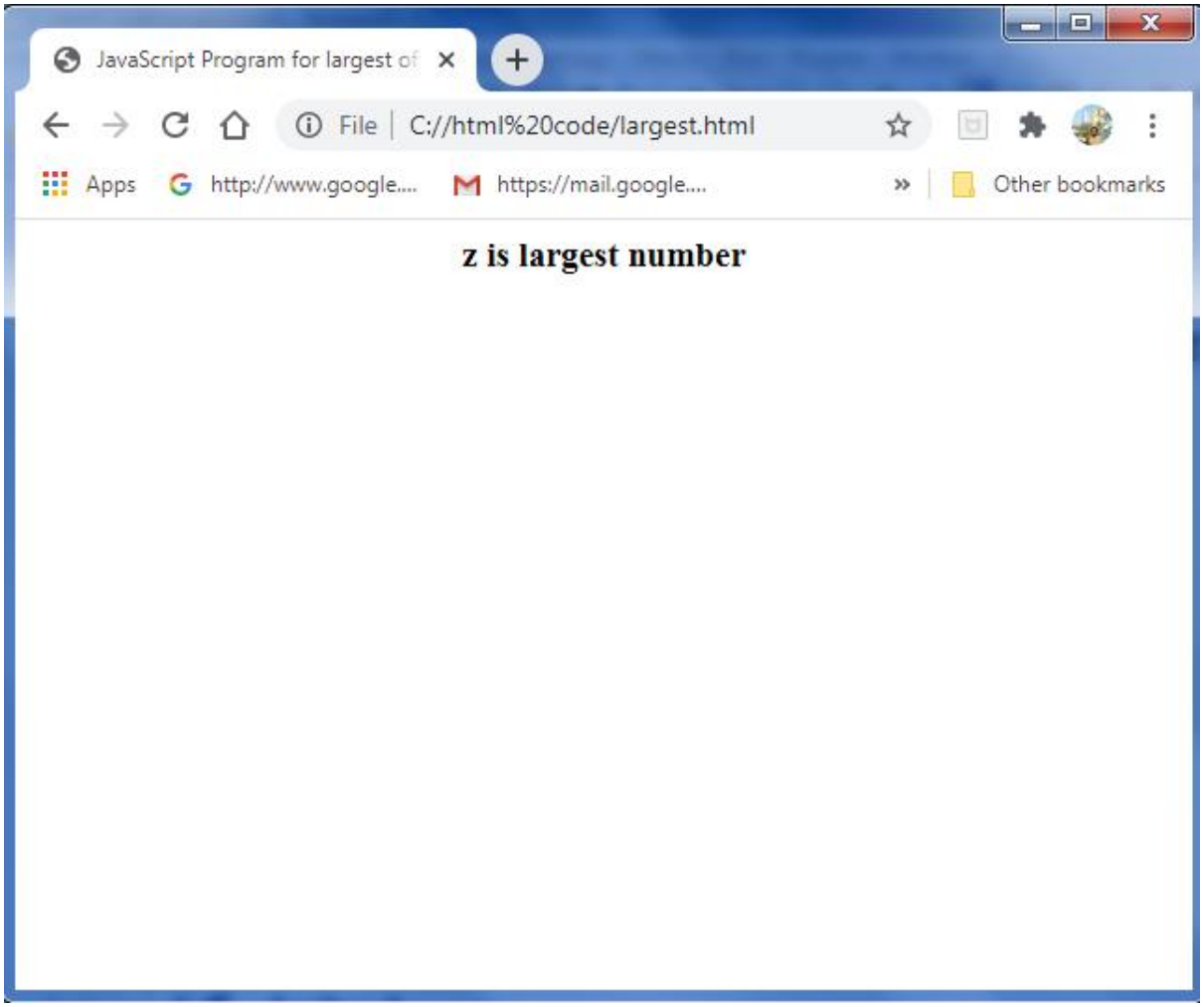
```
if (condition 1)
{

  //If the condition 1 is TRUE then these it will check for test condition 2
  if (condition 2)
  {
   //If test condition 2 is TRUE then these statements will be executed
   condition 2 True statements;
  }

  else
  {
   //If the condition 2 is FALSE then these statements will be executed
   condition 2 False statements;
  }

else
{
 //If the condition 1 is FALSE then these statements will be executed
 condition 1 False statements;
}
```

Here is the simple java script program to demonstrate Nested If

```
<HTML>
<head>
<TITLE>JavaScript Program for largest  of three numbers</TITLE>
</head>
<Body>
<center>
<SCRIPT>
var x,y,z;
var string ;
x=10;
y=20;
z=30;
if (x>y)
{
if (x>z)
document.write("<h3> x is largest number </h3>");
else
document.write("<h3> z is largest number </h3>");
}
else
{
if (y>z)
document.write("<h3> y is largest number </h3>");
else
document.write("<h3> z is largest number </h3>");
}
</SCRIPT>
</center>
```

</Body>

</HTML>



Conditional Operator

In Java Script the conditional operator is ?. The syntax of conditional operator is

Condition ? expression1:expression2

Where expression1 denotes the true condition and expression 2 denotes the false condition.

For Example:

a>b true : false

This means that if a is greater than b then the expression will return the value true otherwise it will return false.

Switch Statement

A `switch` statement is used *to execute one code from multiple expressions*. It is similar to else- if statement.

The syntax of java switch statement is –

```
switch(x) {
  case 'value1':  // if (x === 'value1')
    ...
    [break]

  case 'value2':  // if (x === 'value2')
    ...
    [break]

  default:
    ...
    [break]
}
```

Let us rewrite the above Student grade program using switch case –

<HTML>

<head>

<TITLE>JavaScript Program to display grades of student</TITLE>

</head>

<Body>

<SCRIPT>

var marks = 62;

document.write ("<h3> you got " + marks + " marks "+ "
</h3>");

switch(marks){

case 'marks < 40':

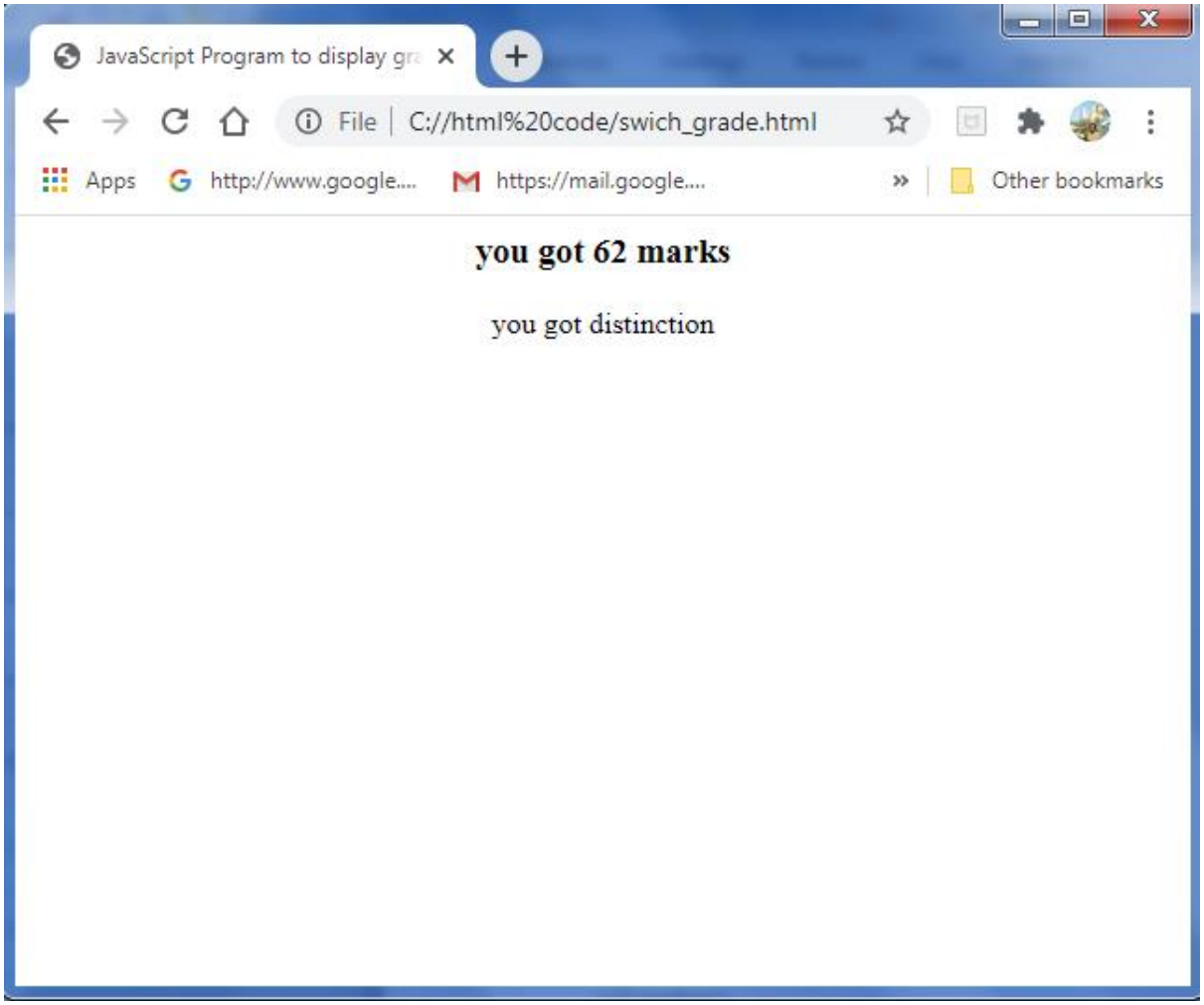
document.write ("you are failed");

break;

case 'marks >= 40 && marks<50':

document.write ("you have got C grade");

```
break;
case 'marks >= 50 && marks<60':
document.write ("you  have got B grade ");
break;
case 'marks >= 60 && marks<70':
document.write ("you  have got A grade ");
break;
default:
document.write ("you got distinction ");
}
</SCRIPT>
</center>
</Body>
</HTML>
```



2. Looping Structures

Loops are used to execute the same block of code again and again, as long as a certain condition is met.

JavaScript supports five different types of loops:

- **while loop**
- **do...while loop**
- **for loop**
- **for...in loop**
- **for...of loop**

The While loop

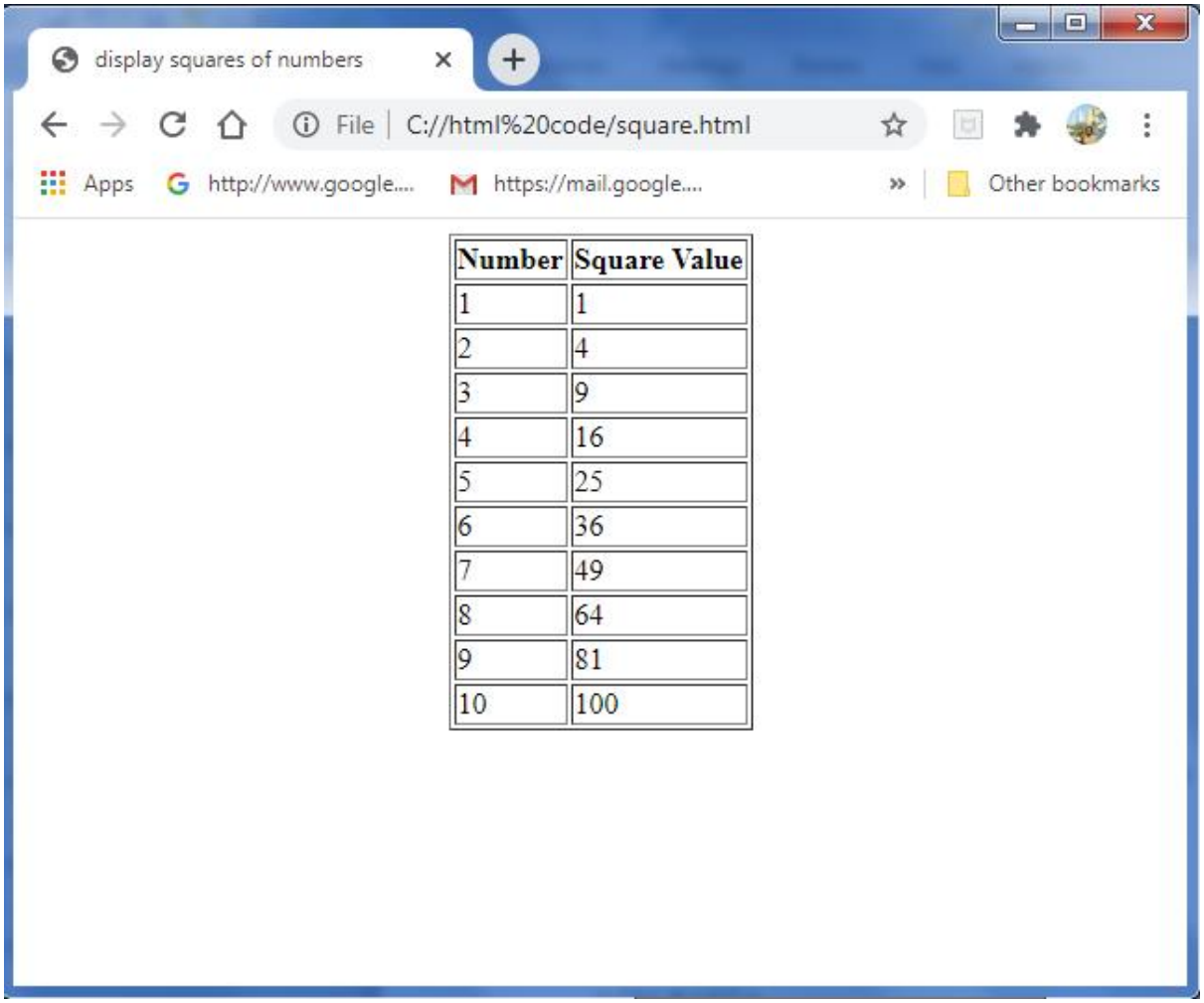
The `while` loop executes a block of code as long as a specified condition is true.

Syntax

```
while (terminating_condition)                                    {
      /* code to be executed
    till the specified condition is true */

}
```

The following is the example of javascript to demonstrate while loop

```
<HTML>

<head>

<TITLE> display squares of numbers</TITLE>

</head>

<Body>

<center>

<table border=1 align="center">

<th> Number </th>

<th> Square Value </th>

<SCRIPT>

i = 1;

while (i<=10)

{

document.write("<tr> <td>" + i + "</td> <td>"+ i*i + "</td> </tr>");

i++;

}

</SCRIPT>

</center>

</Body>

</HTML>
```



Number	Square Value
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

The Do- while loop

The do-while loop is similar to while loop the only difference is that the do-while loop will execute atleast once.

The syntax of do-while is –

do

{

} while (condition);

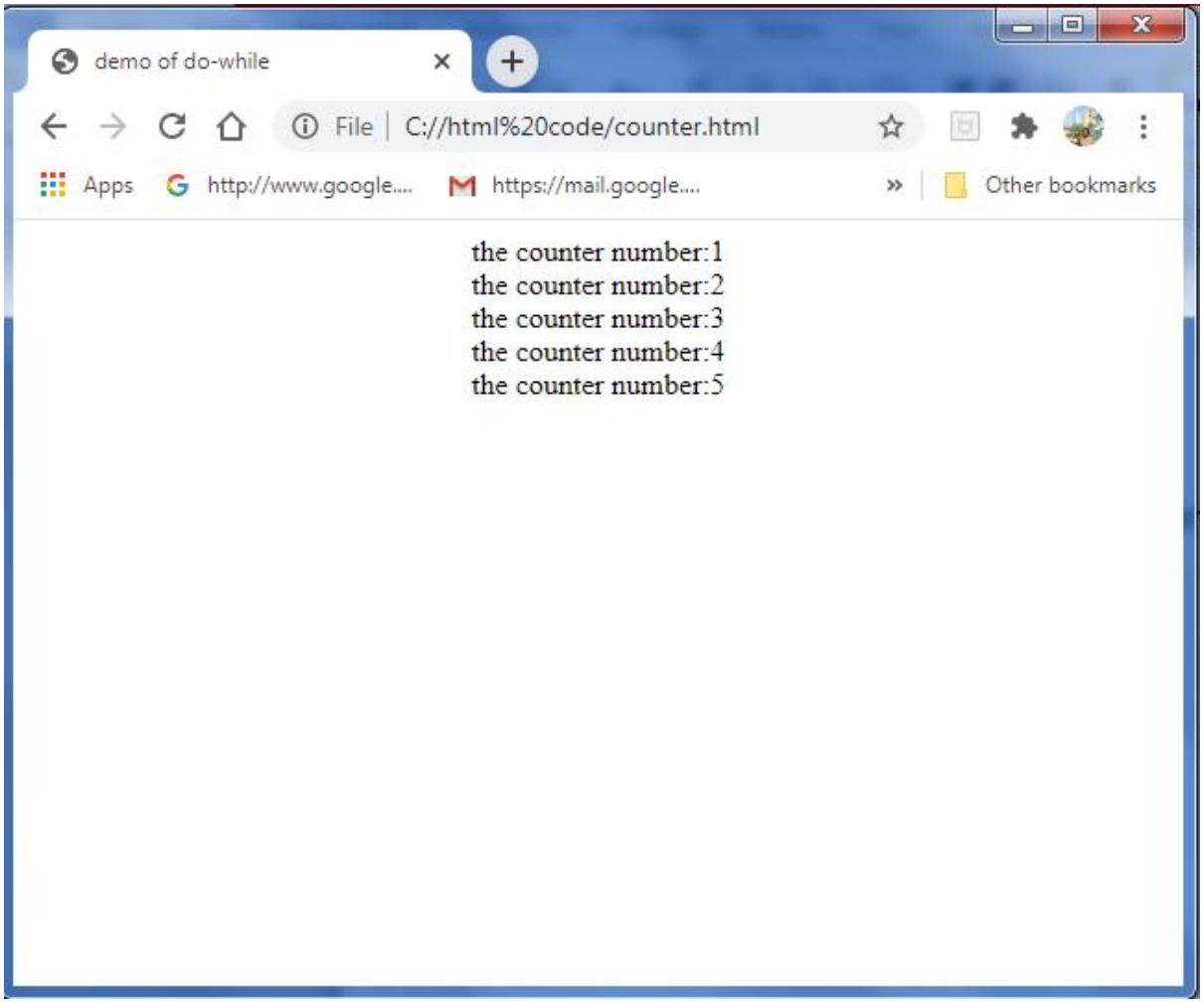
The following program illustrate do-while—

<HTML>

<head>

<TITLE> demo of do-while </TITLE>

```
</head>
<Body>
<center>
<SCRIPT>
counter = 1;
do{
document.write(" the counter number:" + counter + "<br>");
counter++;
} while(counter<=5);

</SCRIPT>
</center>
</Body>
</HTML>
```



The for Loop

The `for` loop repeats a block of code as long as a certain condition is met.

the syntax of for loop is:

```
for(initialization; condition; increment) {
    // Code to be executed
}
```

Here is the program which gives the demonstration of for loop

<HTML>

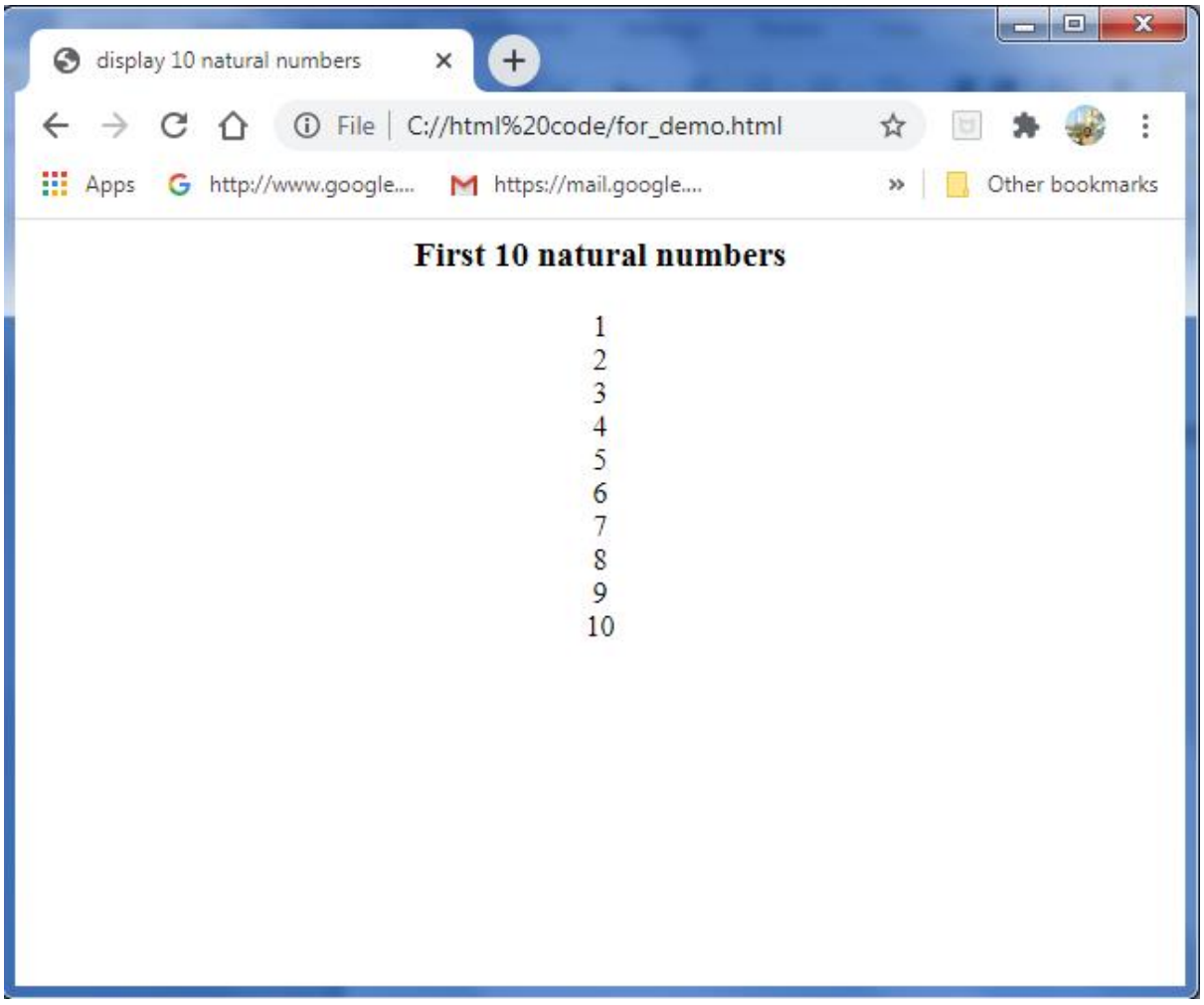
<head>

<TITLE> display 10 natural numbers </TITLE>

</head>

<Body>

```
<center>
<SCRIPT>
var i;
document.write("<h3> First 10 natural numbers </h3>");
for (i=1; i<=10; i++)
{
document.write(i + "<br/>")
}
</SCRIPT>
</center>
</Body>
</HTML>
```



The for – in loop

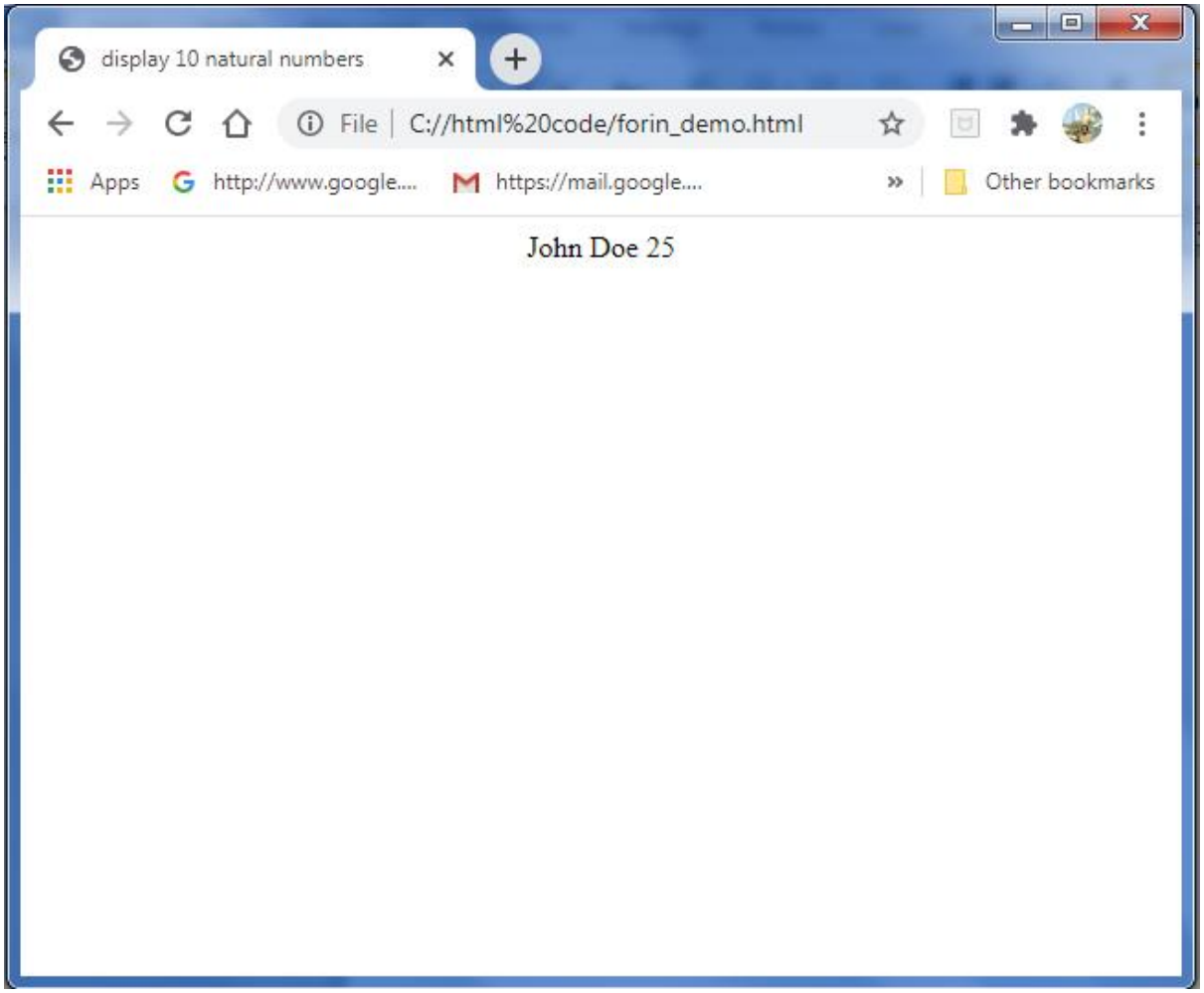
The `for-in` loop is a special type of a loop that iterates over the properties of an object, or the elements of an array.

The syntax of for-in loop is –

```
for(variable in object) {
    // Code to be executed
}
```

The following is the example to demonstrate the for – in loop –

```
<HTML>

<head>

<TITLE> for in demo </TITLE>

</head>

<Body>

<center>

<SCRIPT>

var person = {fname:"Ravi", lname:"Kumar", age:25};


var text = "";

var x;

for (x in person) {

  text += person[x] + " ";

}

document.write(text);

</SCRIPT>

</center>

</Body>

</HTML>
```



3. Lables

The javascript label statements are used to jump out of the code block

There are two types of label statements are used in java script

-
- Break
- Continue

Break Statement

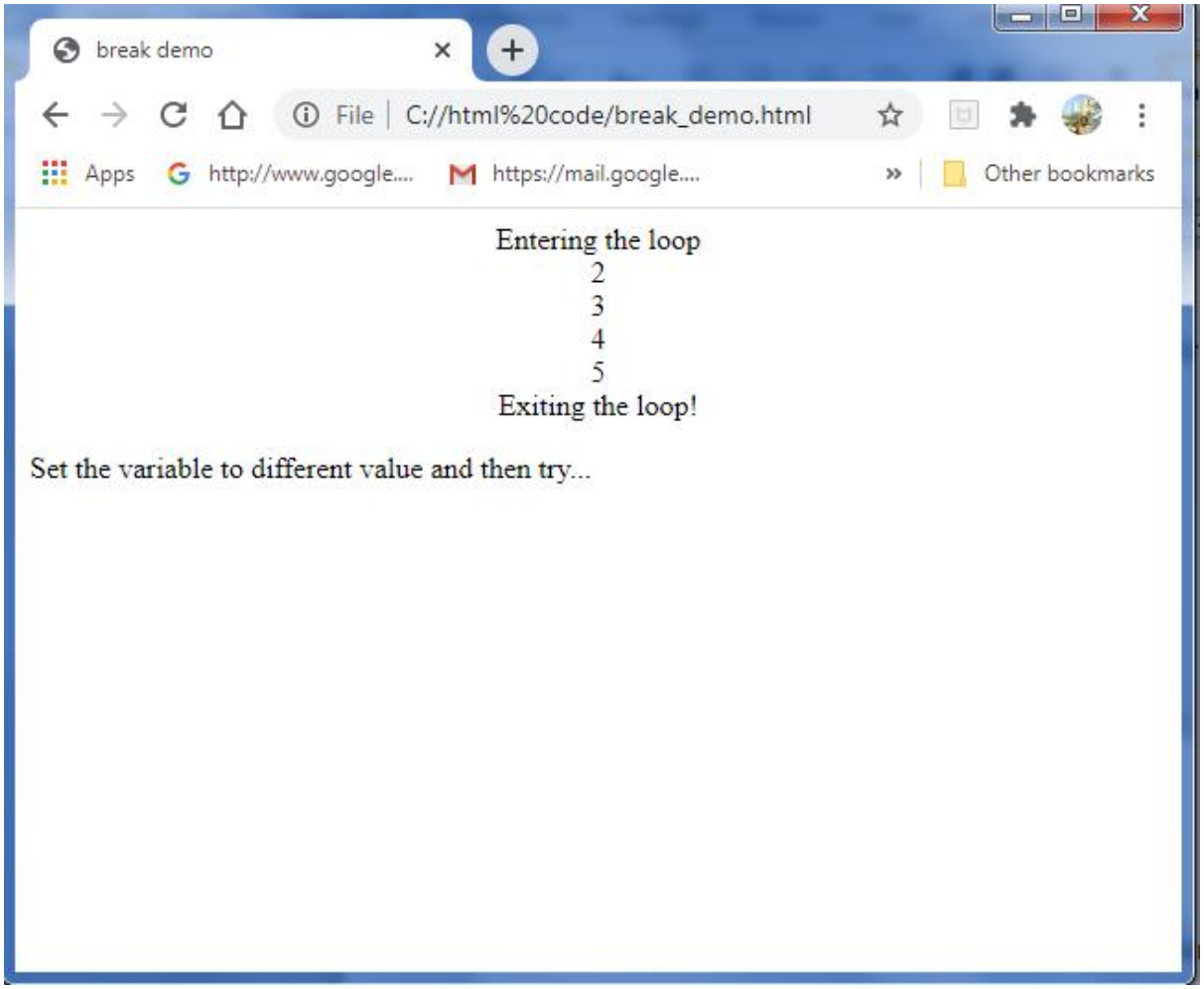
The break statement is used to break the loop. This can be used with the keyword **break**.

The following program demonstrates the example of break –

The above example illustrates the use of a **break** statement with a while loop. Notice how the loop breaks out early once **x** reaches 5 and reaches to **document.write (..)** statement just below to the closing curly brace −

<HTML>

```
<head>
<TITLE>  break demo </TITLE>
</head>
<Body>
<center>
<SCRIPT>
var x = 1;
       document.write("Entering the loop<br /> ");

       while (x < 20) {
         if (x == 5) {
           break;   // breaks out of loop completely
         }
         x = x + 1;
         document.write( x + "<br />");
       }
       document.write("Exiting the loop!<br /> ");
</SCRIPT>
</center>
    <p>Set the variable to different value and then try...</p>

</Body>
</HTML>
```



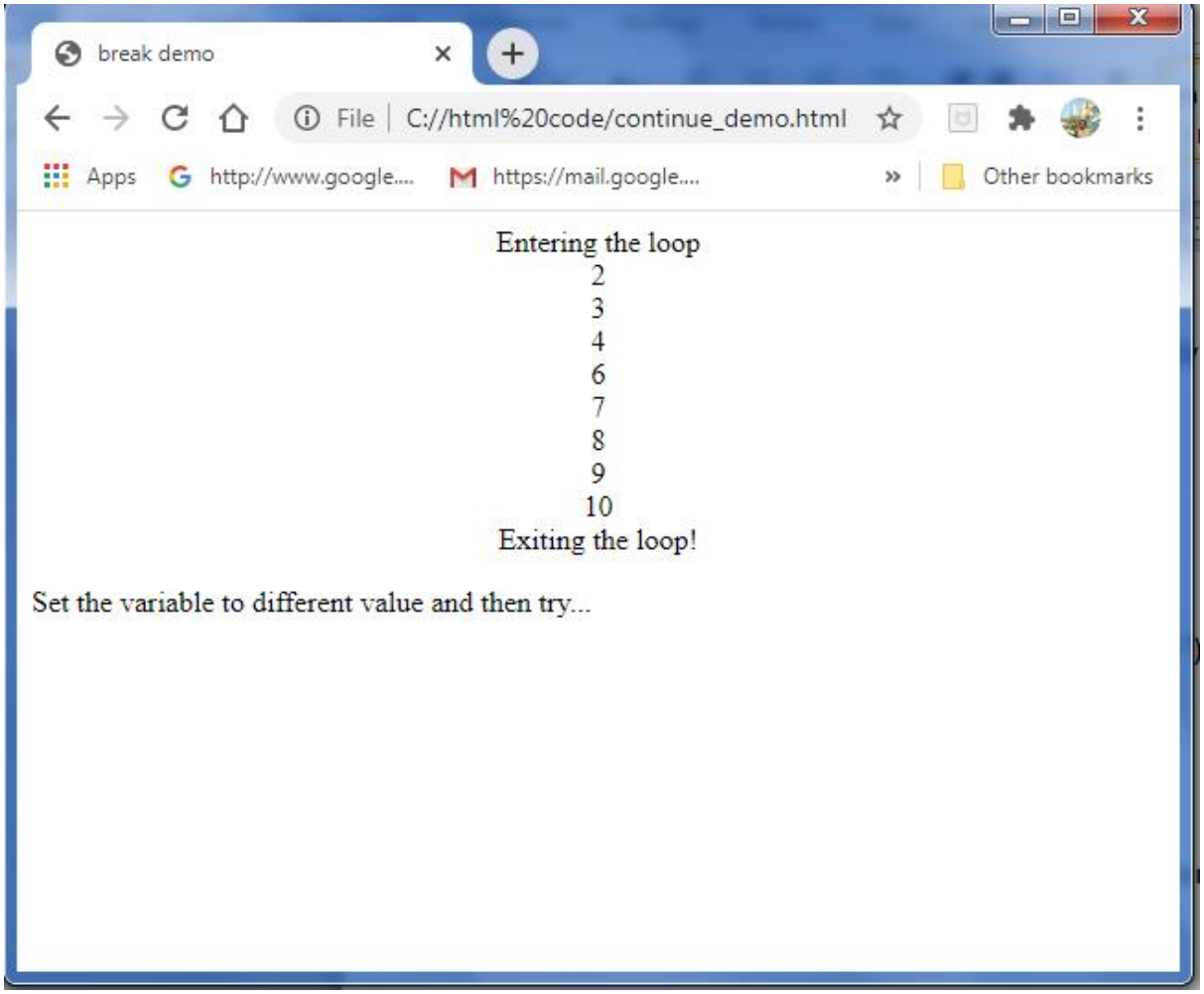
Continue Statement

The continue statement is used in a loop in order to continue(skip). The key word **continue** is used to make use of continue statement in a loop.

The following program demonstrate this.

```
<HTML>

<head>

<TITLE>  break demo </TITLE>

</head>

<Body>

<center>

<SCRIPT>

var x = 1;

        document.write("Entering the loop<br /> ");
```

```
          while (x < 10) {
            x = x + 1;

            if (x == 5) {
              continue;   // skip rest of the loop body
            }
            document.write( x + "<br />");
          }
          document.write("Exiting the loop!<br /> ");
      </SCRIPT>
</center>
    <p>Set the variable to different value and then try...</p>

</Body>
</HTML>
```



POPUP BOXES

One of the important feature in java script is interactivity with the user. There are three types of popup boxes are used in java script to interact the user with browser.

Alert Box

An alert box is often used if you want to make sure information comes through to the user.

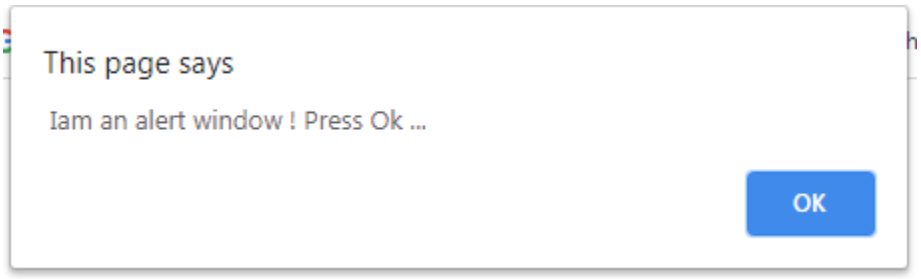
When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

Alert (“ Write something here”);

Example

```
<SCRIPT>

alert( " Iam an alert window ! Press Ok ...");

        </SCRIPT>
```



Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.

Syntax

If (confirm (“ write some thing here”))

{

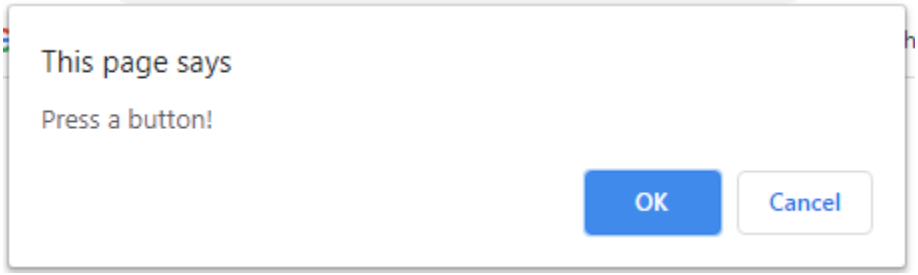
Message; // if press ok button

Else

Message; // if press cancel button

Example

```
<SCRIPT>
if (confirm("Press a button!")) {
  txt = "You pressed OK!";
} else {
  txt = "You pressed Cancel!";
}
        </SCRIPT>
```



Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

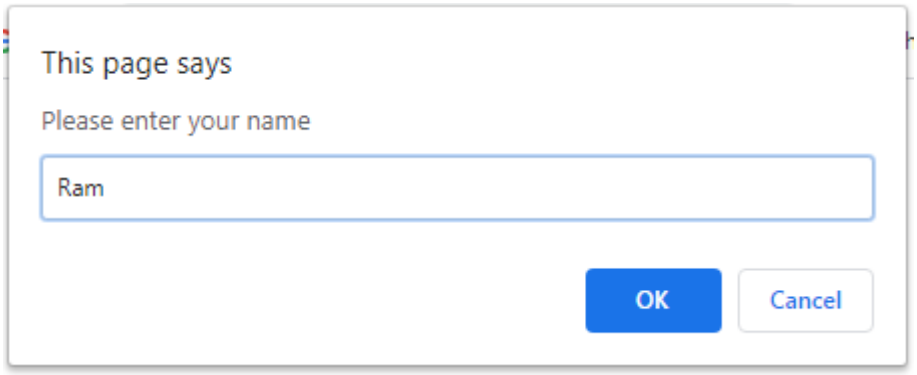
Syntax

```
Var var_name= promt (“ write some thing”);
```

Example

```
<SCRIPT>
var person = prompt("Please enter your name", "Harry Potter");


if (person == null || person == "") {
```

```
  txt = "User cancelled the prompt.";
} else {
  txt = "Hello " + person + "! How are you today?";
}
        </SCRIPT>
```



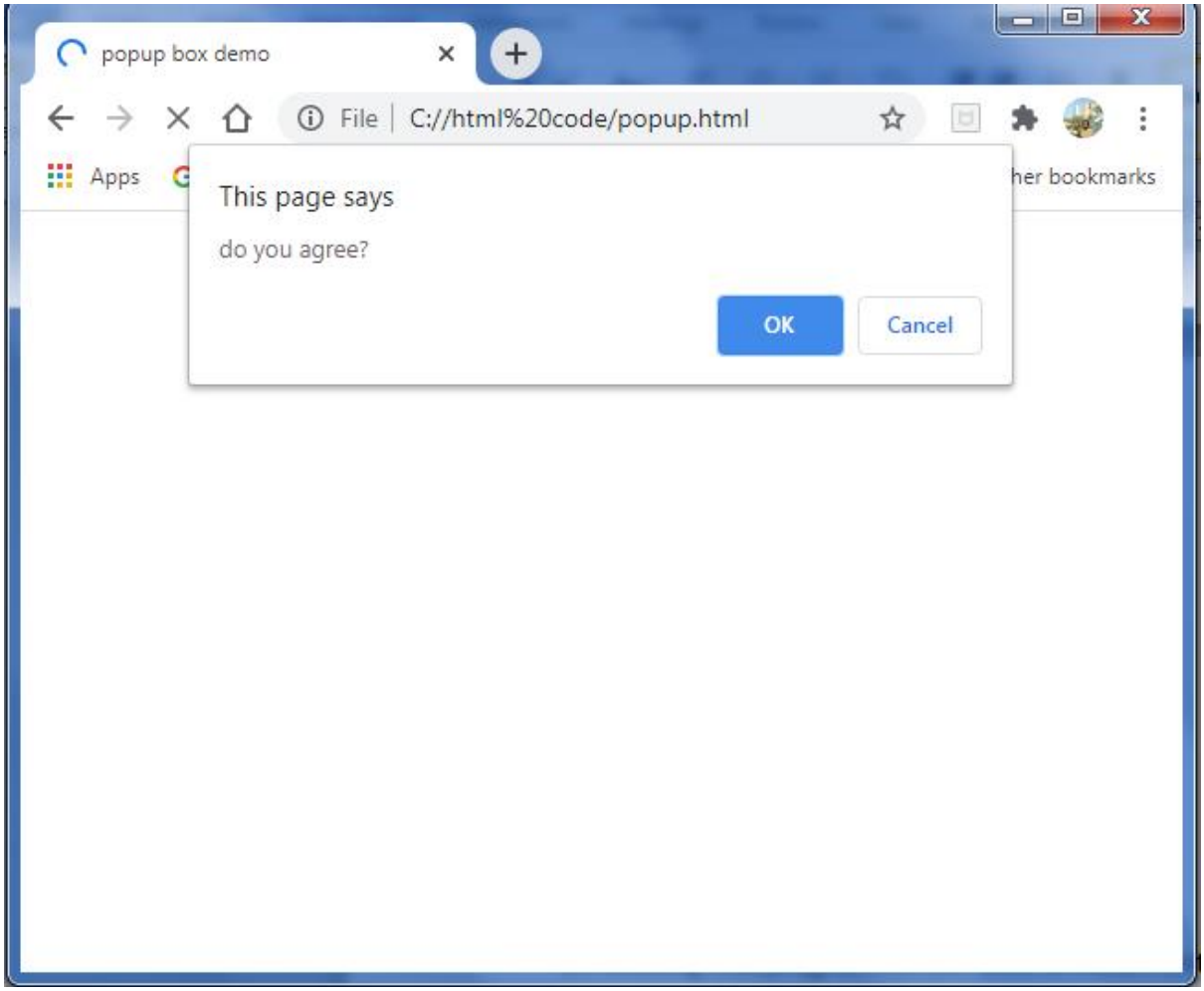
Following example demonstrates the use of these pop up boxes

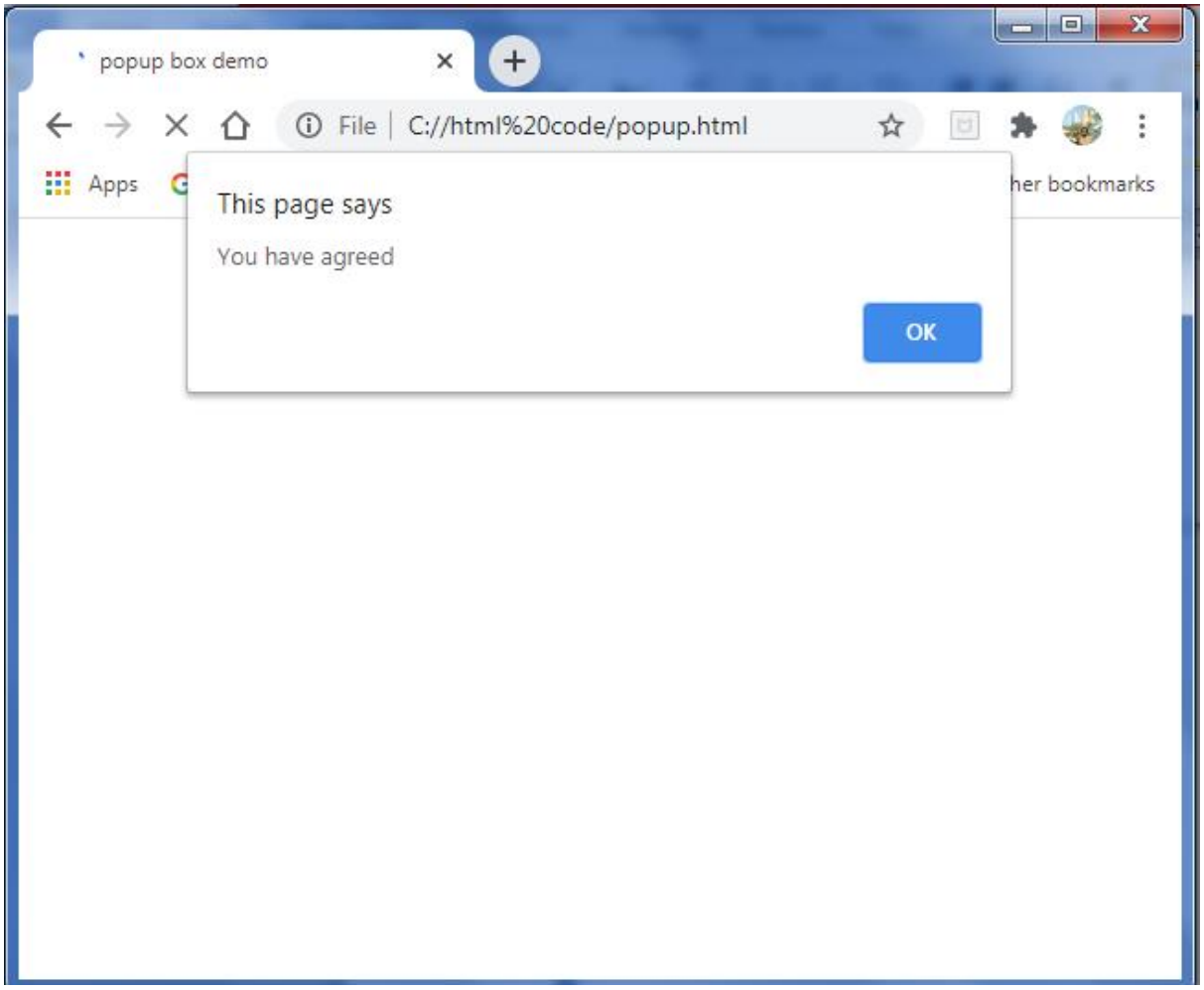
```
<HTML>
<head>
<TITLE>  popup box demo </TITLE>
</head>
<Body>
<center>
<SCRIPT>
if (confirm("do you agree?"))
alert("You have agreed");
else
input_text=prompt("enter your condition here...");
/* the value entered in the prompt box is returned and
stored in the variable text */
alert ("Hi " + input_text);

    </SCRIPT>
```

</Body>

</HTML>

On loading this program on the web browser first of all **confirm box** will be displayed. If we click on Ok button an alert box will appear. Otherwise a prompt box will appear . again if you click on the Ok button of prompt box agan an alert box will appear. If you run the above program you will get all these effects.





Functions

Functions are the main "building blocks" of the program. They allow the code to be called many times without repetition. We can define the function anywhere in the script, either head or body section or in both. But it is a standard practice to define the function in head section and call that function from body section.

The keyword function is used while defining the function. The syntax for defining the function is

```
//defining a function
function <function-name>(parameters)
{
    // code to be executed
};

//calling a function
<function-name>(arguments);
```

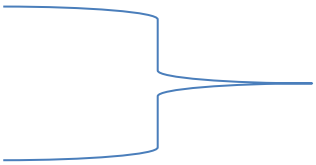
Function **parameters** are listed inside the parentheses () in the function definition.

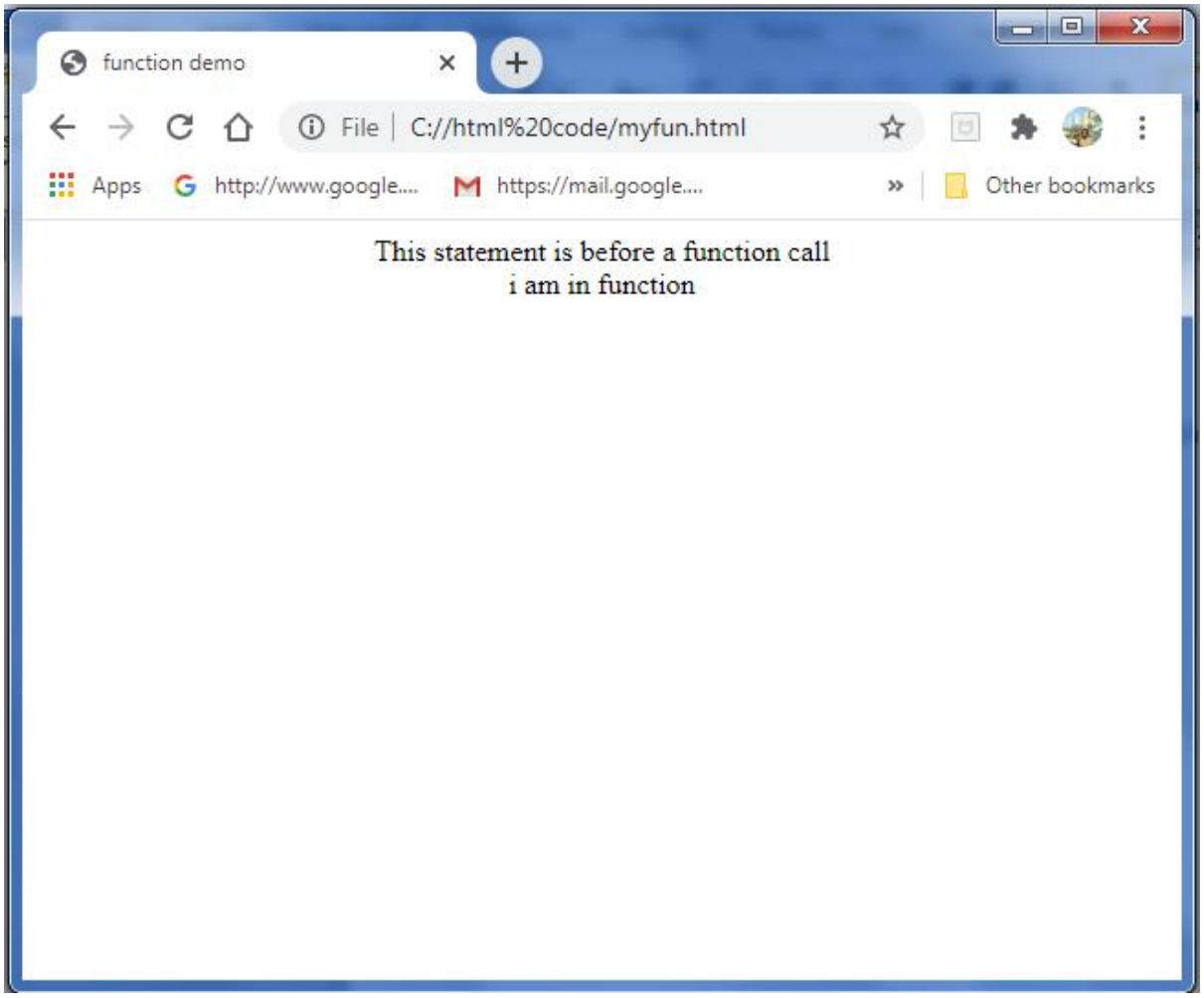
Function **arguments** are the **values** received by the function when it is invoked.

Inside the function, the arguments (the parameters) behave as local variables.

Here, is the simple illustration in which we have written a function my_fun()

```
<HTML>
<head>
<TITLE>  function demo </TITLE>
<SCRIPT>
function my_func()
{                                                    // definition of function
document.write(" i am in function");
}
      </SCRIPT>
</head>
<Body>
<center>
<SCRIPT>
document.write(" This statement is before a function call<br>");
my_func(); // call of the function
      </SCRIPT>
</center>
</Body>
</HTML>
```





Function Return

When JavaScript reaches a `return` statement, the function will stop executing.

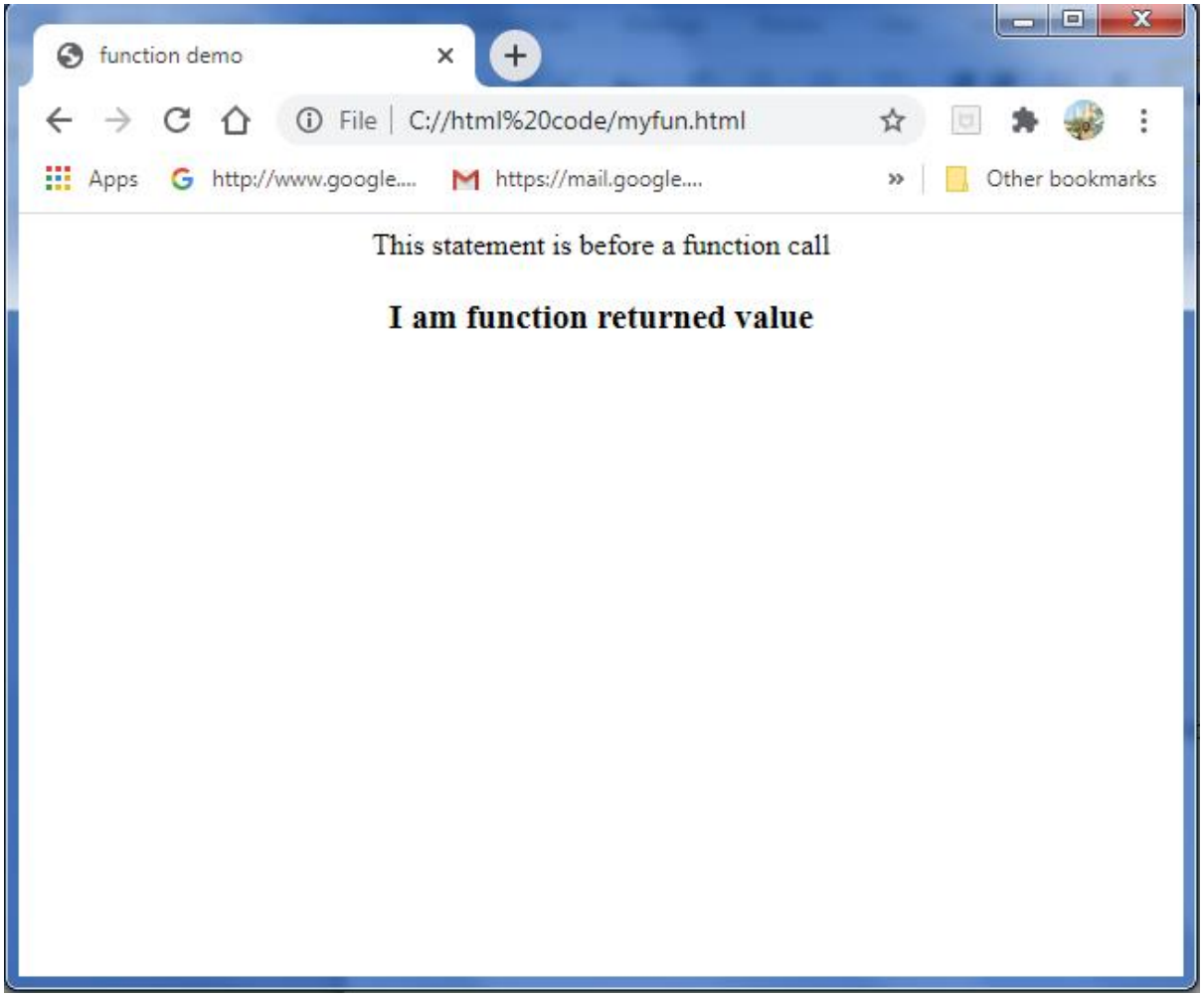
If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

Functions often compute a **return value**. The return value is "returned" back to the "caller":

The above program is slightly modified in which function will return something.

```
<HTML>

<head>

<TITLE>  function demo </TITLE>

<SCRIPT>

function my_func()

{
```

```
str= " I am function returned value";
return str;
}

    </SCRIPT>
</head>
<Body>
<center>
<SCRIPT>
document.write(" This statement is before a function call<br>");
document.write("<h3>"+my_func()+"<h3>");
    </SCRIPT>
</center>
</Body>
</HTML>
```



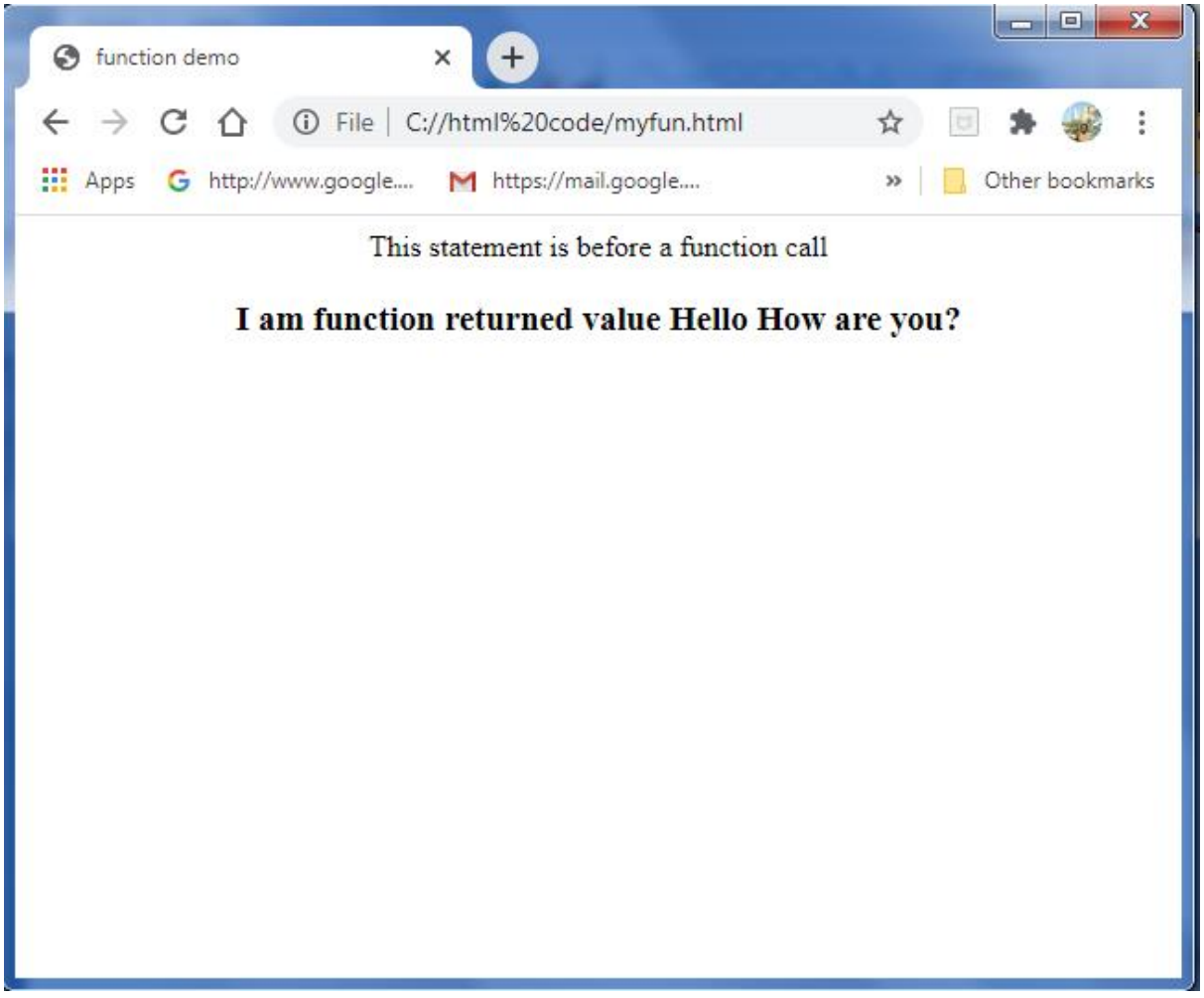
Passing arguments to the function

Similarly we can pass some arguments to the function.

The following is the modification of above program , have passed two arguments to the function and returning some value form the function.

```
<HTML>

<head>

<TITLE>  function demo </TITLE>

<SCRIPT>

function my_func(s1, s2)

{

str= " I am function returned value" + " " + s1+ " " + s2;

return str;

}
```

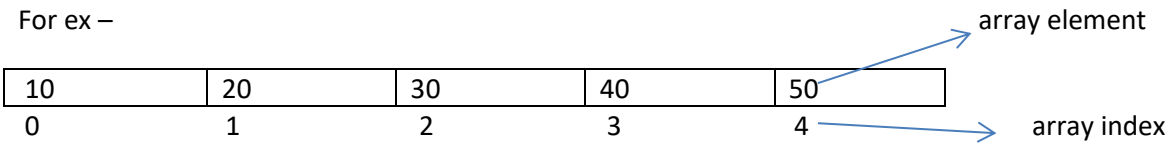
```
    </SCRIPT>
</head>
<Body>
<center>
<SCRIPT>
document.write(" This statement is before a function call<br>");
document.write("<h3>"+my_func("Hello", " How are you?")+"<h3>");
    </SCRIPT>
</center>
</Body>
</HTML>
```



Arrays

Arrays is collection of similar type of elements which can be referred by common name. an element in an array is referred BY an array name followed by [position of the element]. The particylar position of the element in an array is called array index or subscript.

For ex –



Array a[5]

Fig : arrays

We can start storing the elements in an array from any position, but usually it starts form 0^{th} position.

Array declaration

In javascript the array can be created using array object. Let us see now how to declare and create an array.

Syntax:

```
var array_name = new array [item1, item2, ...];
```

Suppose, if we want to create an array of 10 elements then we can write,

Var ar = new Array[10];

Using new operator we can allocate the memory dynamically for an array. Var ar denotes the name of the array and the size of an array is mentioned in brackets[].

The above statement can also be written like this:

Var ar

ar = new Array[10];

Array initialization

Let us see how toe store some elements in the array in the following program

<HTML>

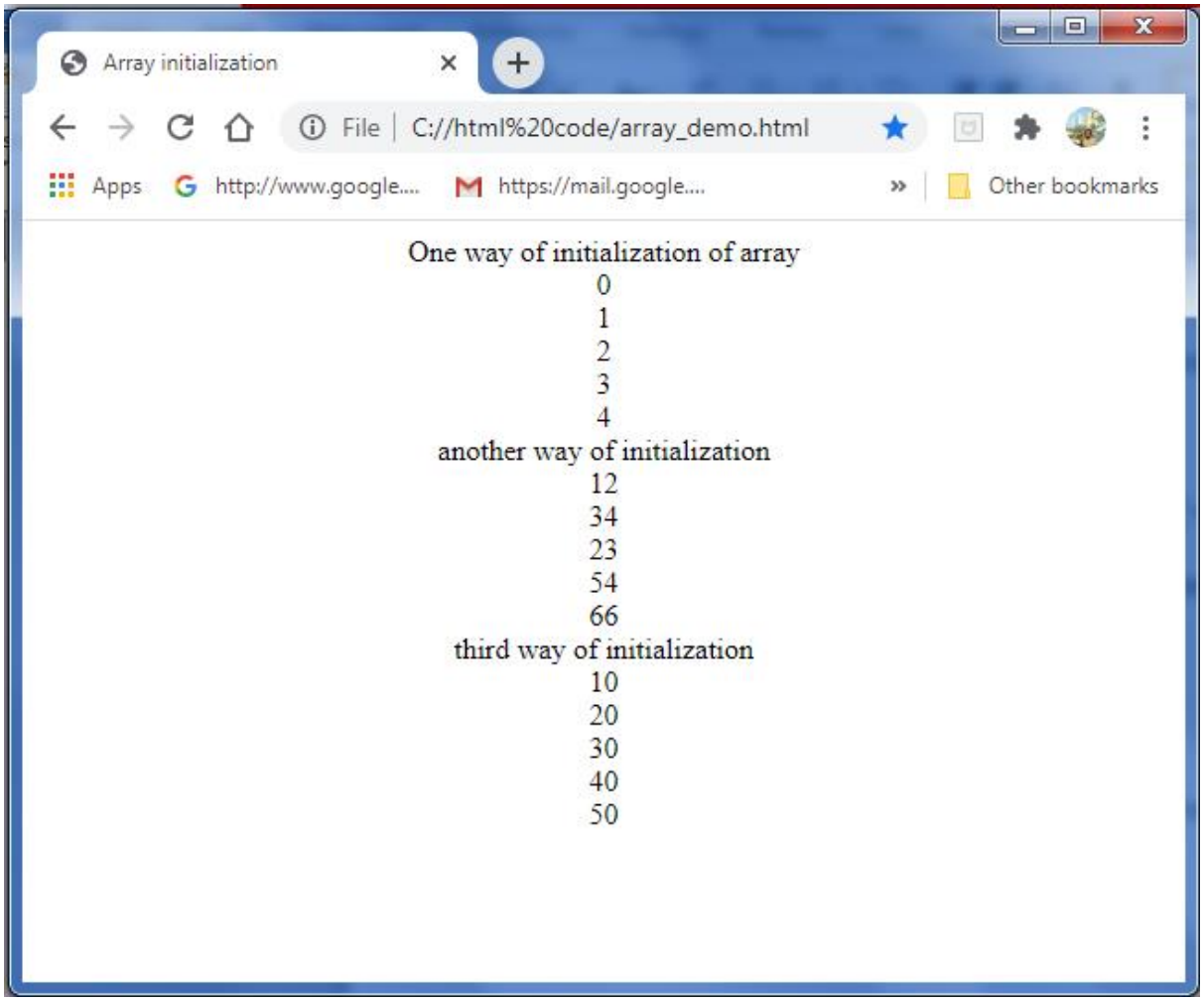
<head>

<TITLE> Array initialization </TITLE>

</head>

<Body>

```
<SCRIPT>
document.write(" One way of initialization of array <br>");
a = new Array(5); // creation of an array
for (var i=0;i<5; i++)
{
a[i]= i;
document.write(a[i] + "<br>"); // displaying array
}
document.write ("another way of initialization" + " <br>");
b = new Array(12,34,23,54,66); // creation of array
for (i=0;i<5; i++)
{
document.write (b[i] + " <br>"); // displaying array
}
document.write ("third way of initialization" + " <br>");
var c = Array(10,20,30,40,50); // creation of array
for ( i=0;i<5; i++)
{
document.write (c[i] + " <br>"); // displaying array
}
      </SCRIPT>
</center>
</Body>
</HTML>
```



Using for –in control structure

The for .. in control structure in javascript is closely associated with array elements . let us see how to use for .. in control structure in array.

<HTML>

<head>

<TITLE> using for.. in in Array </TITLE>

</head>

<Body>

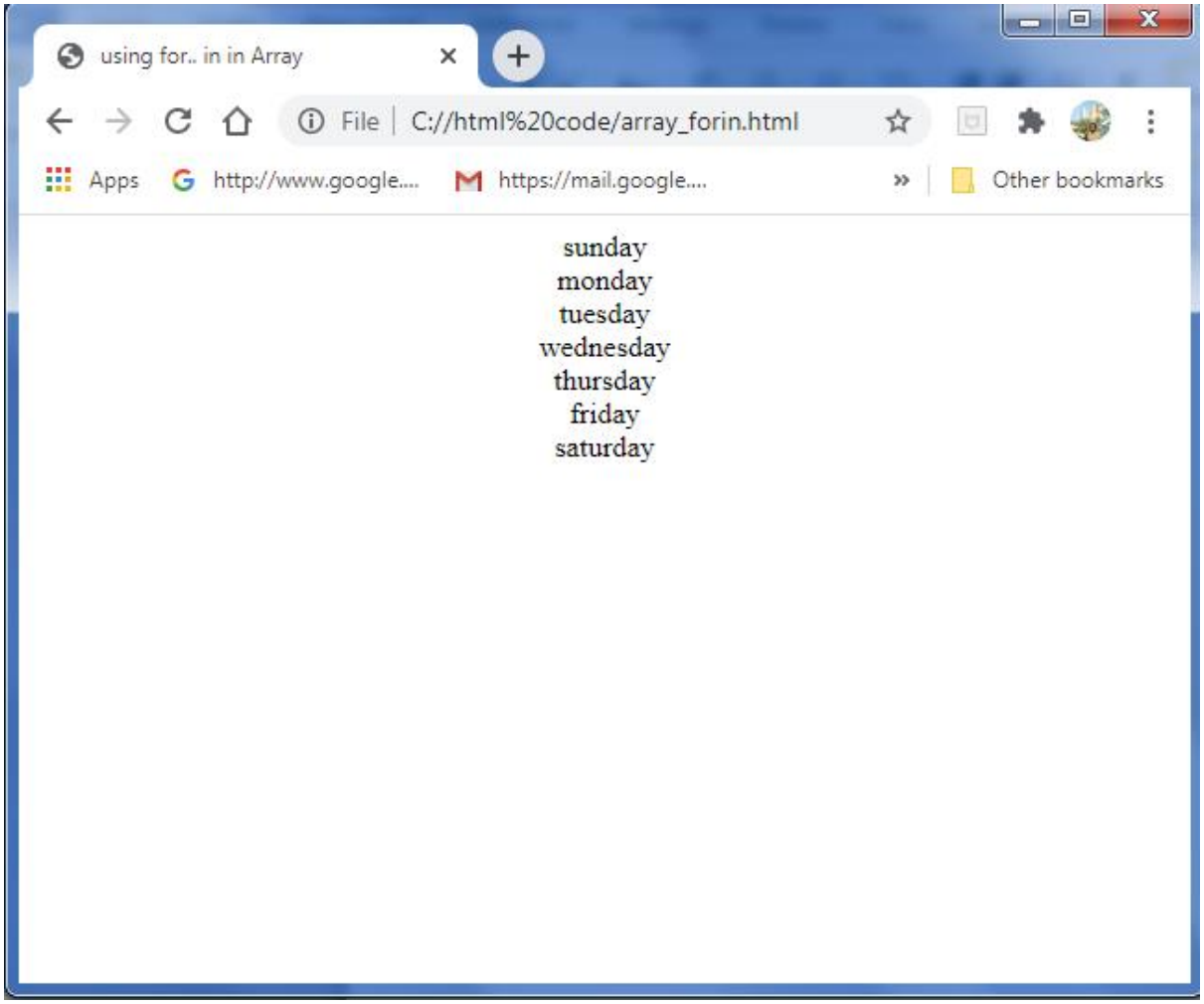
<SCRIPT>

days = new Array();

days[0]= "sunday"

days[1]= "monday"

```
days[2]= "tuesday"

days[3]= "wednesday"

days[4]= "thursday"

days[5]= "friday"

days[6]= "saturday"

for (i in days)

{

document.write(days[i]+ "<br>");

}

      </SCRIPT>

</center>

</Body>

</HTML>
```



Passing an array to the function

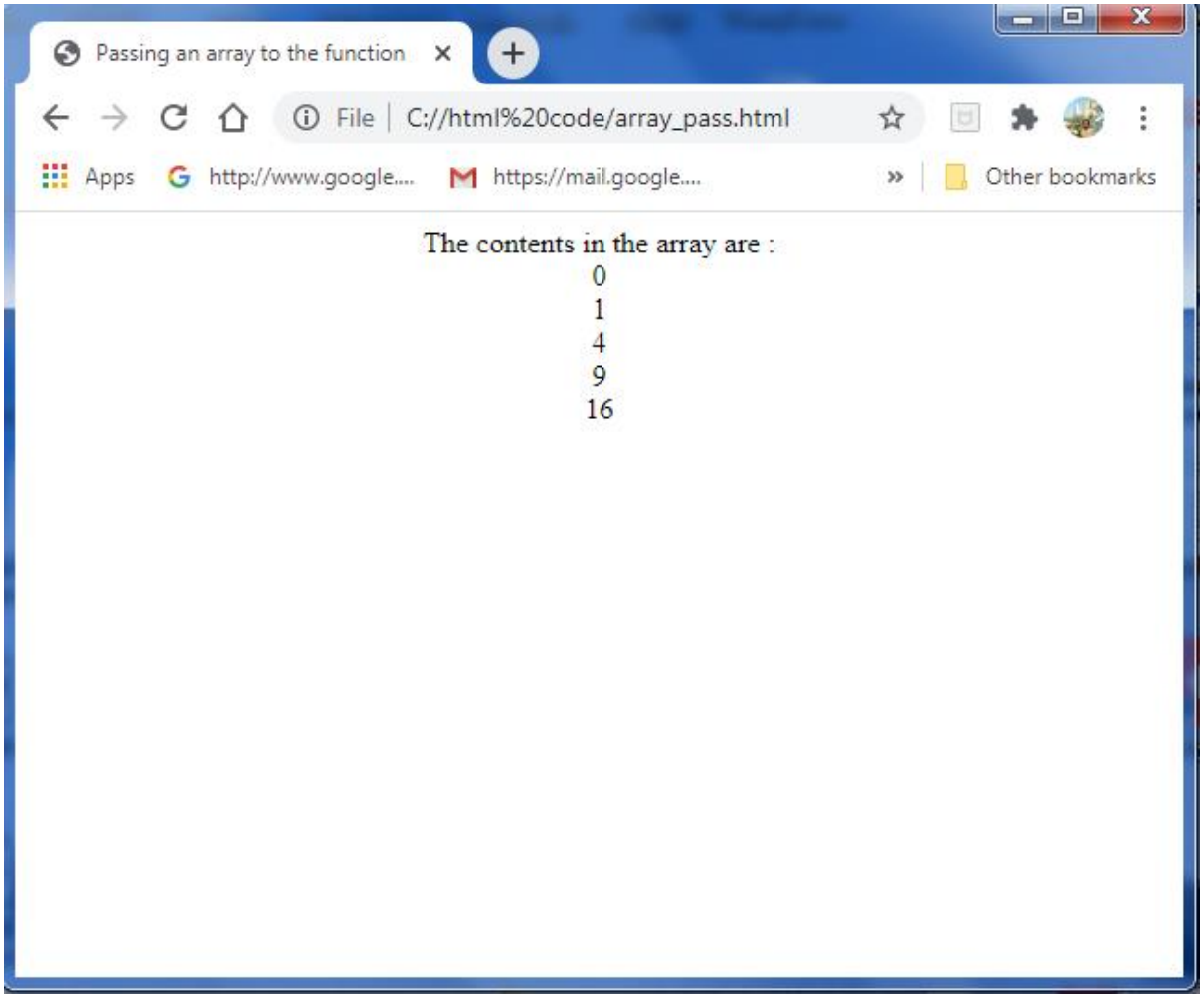
J:ust like C we can pass an entire array as a parameter to the function. This method of an array passing is called as call by reference. When an array is passed to the function the array name is simply passed.

The following javascript program illustrates this example

```
<HTML>

<head>

<TITLE> Passing an array to the function </TITLE>

<script>

function display_square(s)

{

document.write(" The contents in the array are : <br>");

i=0;

for(i in s)

{

document.write(s[i]+"<br>");

i++;

}

}

</script>

</head>

<Body>

<center>

<SCRIPT>

 var sq= new Array(5);

 for(i=0;i<5;i++)

 {

 sq[i]=i*i;

 }
```

```
 display_square(sq);

 </SCRIPT>

</center>

</Body>

</HTML>
```

In the above program we have initialized an array in the body part and to display the contents of the array we used square_display() function. And to this function an array is passed as an argument.



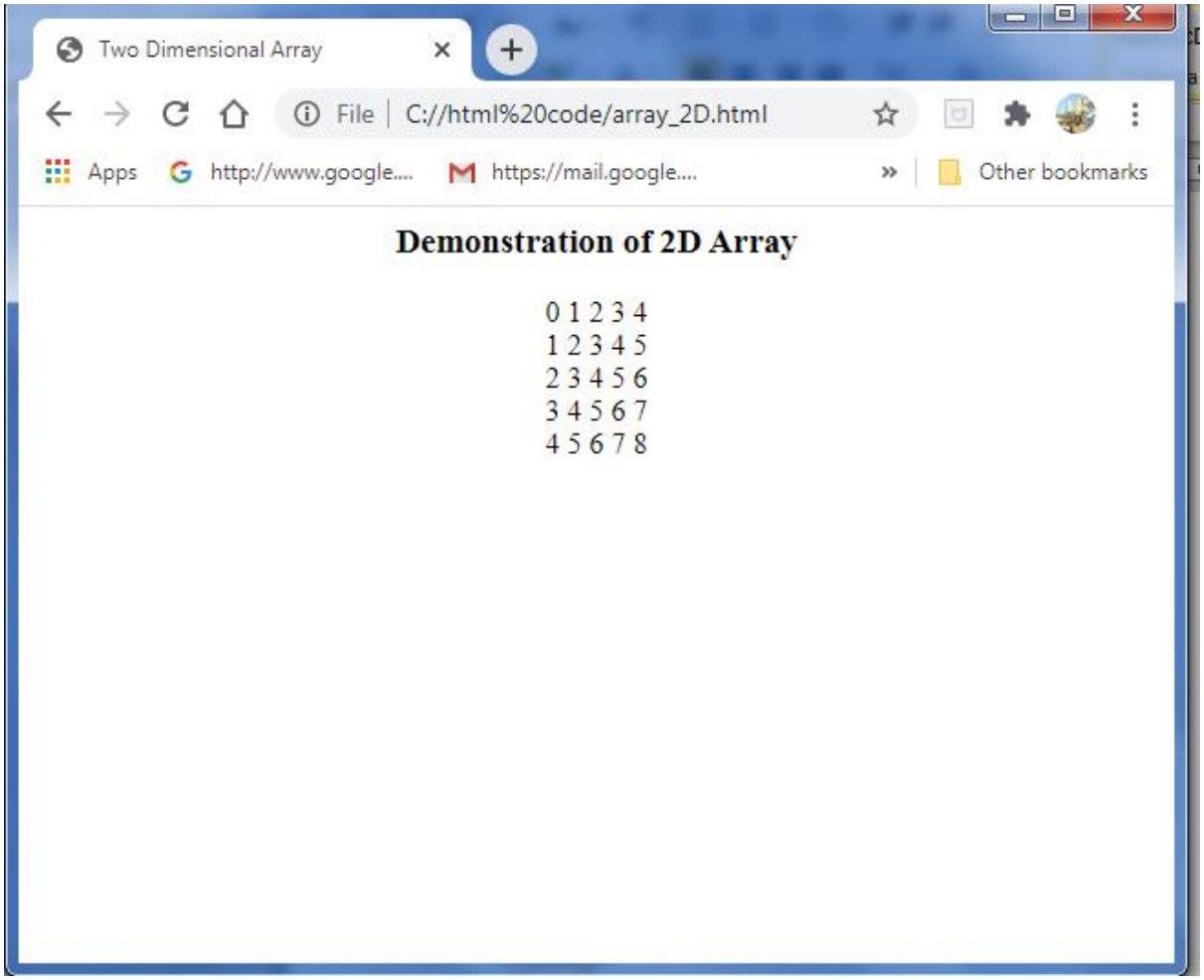
Two dimensional array

Actually Javascript does not support multi dimensional arrays. Hence for defining two dimensional arrays we make use of single dimensional arrays.

Here it is :

```
<HTML>

<head>

<TITLE> Two Dimensional Array </TITLE>
```

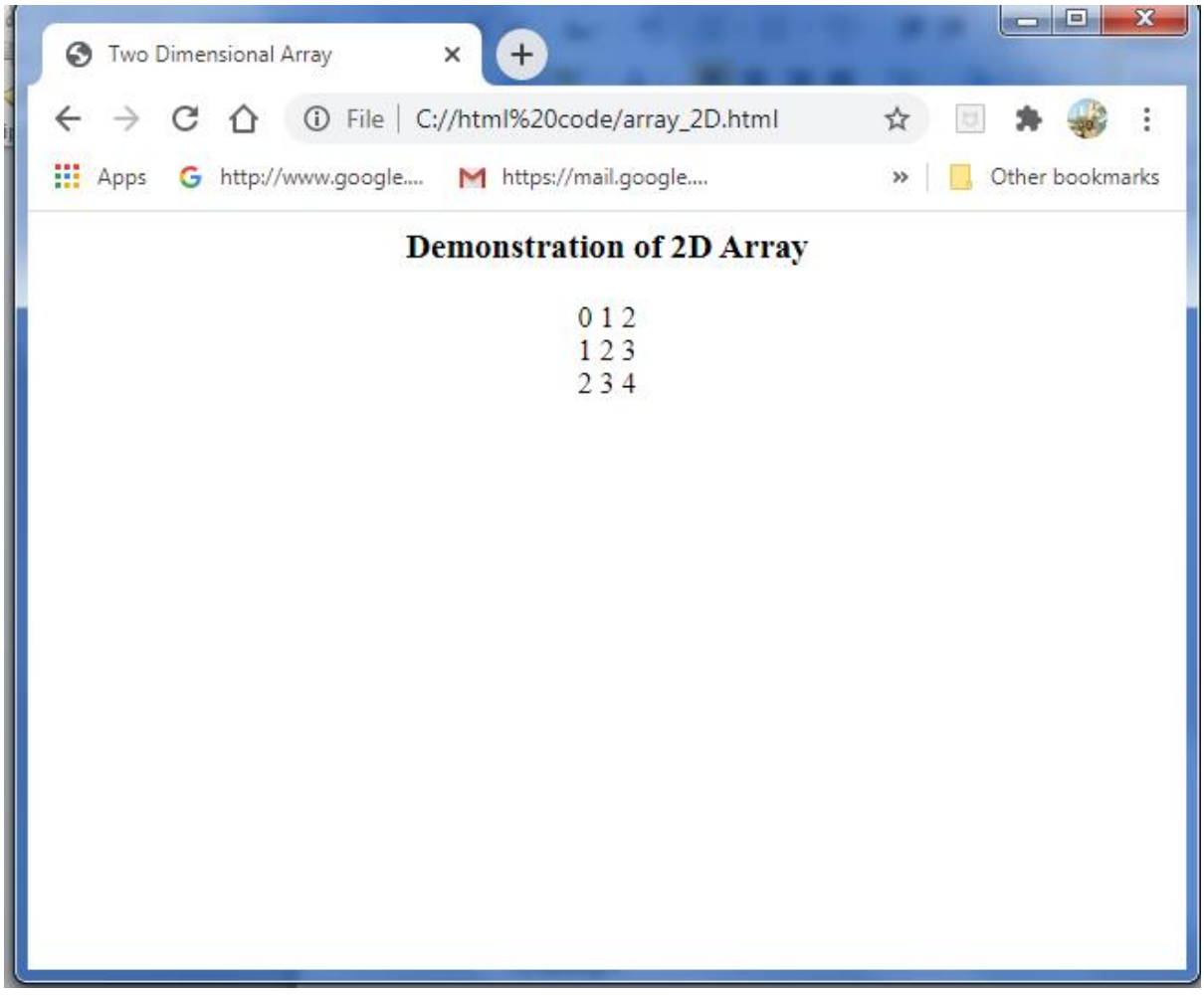
```
</head>
<Body>
<center>
<h3> Demonstration of 2D Array </h3>
<SCRIPT>
 a=new Array(); // creation of rows of array
for(i=0;i<5;i++)
a[i]=new Array(); // creation of columns of array
for(i=0;i<5;i++)
{
for(j=0;j<5;j++)
{
a[i][j]= i+j;
document.write(a[i][j] + "  ");
}
document.write("<br>");
}
</SCRIPT>
</center>
</Body>
</HTML>
```



There is another way by which a 2D array is initialized. This is represented in the following program.

```
<head>

<TITLE> Two Dimensional Array </TITLE>

</head>

<Body>

<center>

<h3> Demonstration of 2D Array </h3>

<SCRIPT>

 var a=[ [1,2,3],[4,5,6],[7,8,9]]; // creation of array

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{
```

```
a[i][j]= i+j;

document.write(a[i][j] + "  ");

}

document.write("<br>");

}

</SCRIPT>

</center>

</Body>

</HTML>
```



Events

There are mainly 3 different kind of events in javascript. They are

Onsubmit

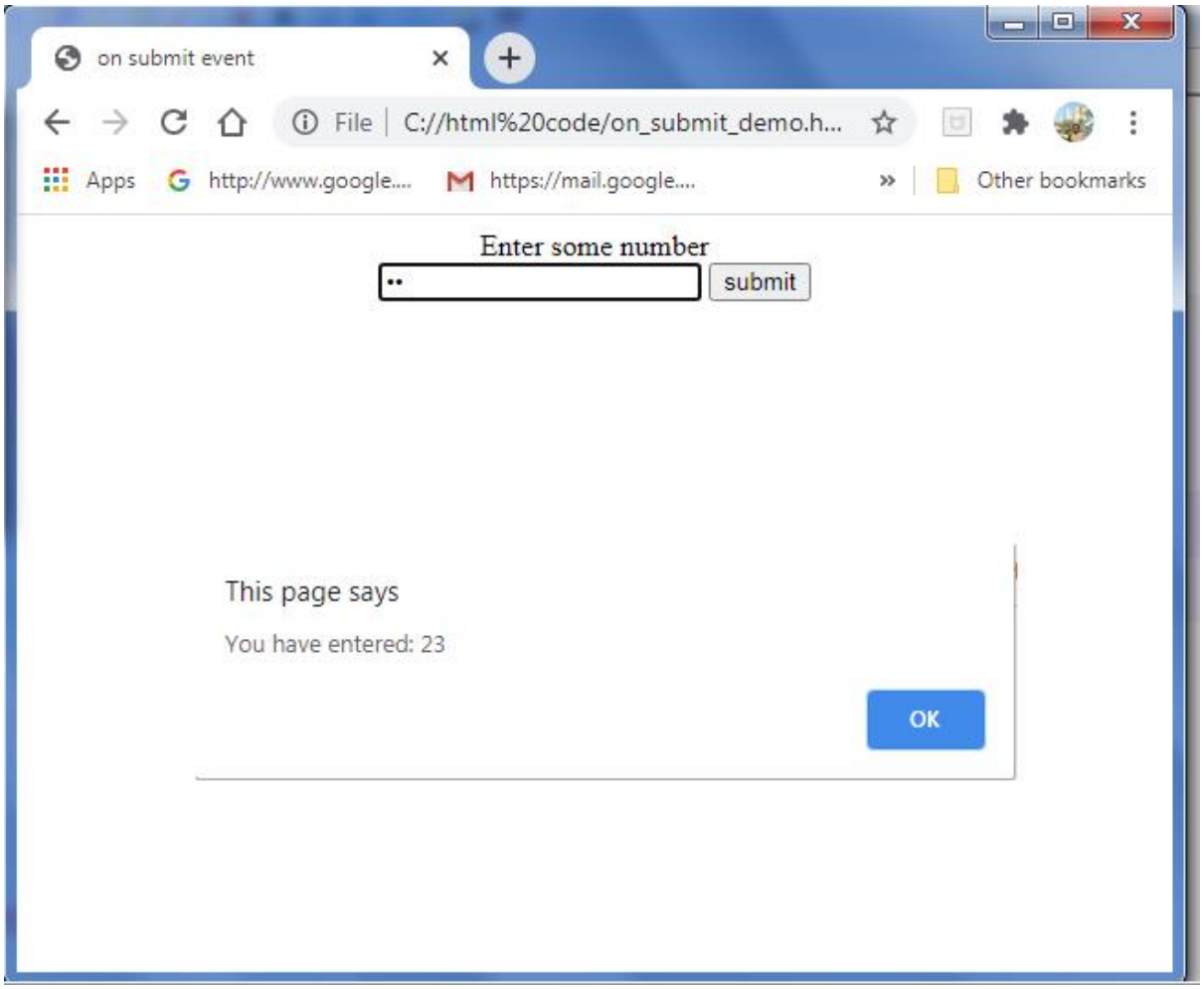
Onclick

Onload

To make use of these events we need to put some form components on the javascript. And when we handle these components there are some possible events that could occur.

Onsubmit : it is one such event which occurs when we click submit button.

Let us see some example to represent this:



In the above program we have used onsubmit() event in the form to call the function guess().

Onclick: This event is associated with the button click.

Here the example javascript program to call clik() function as soon as the button is clicked.

<HTML>

<head>

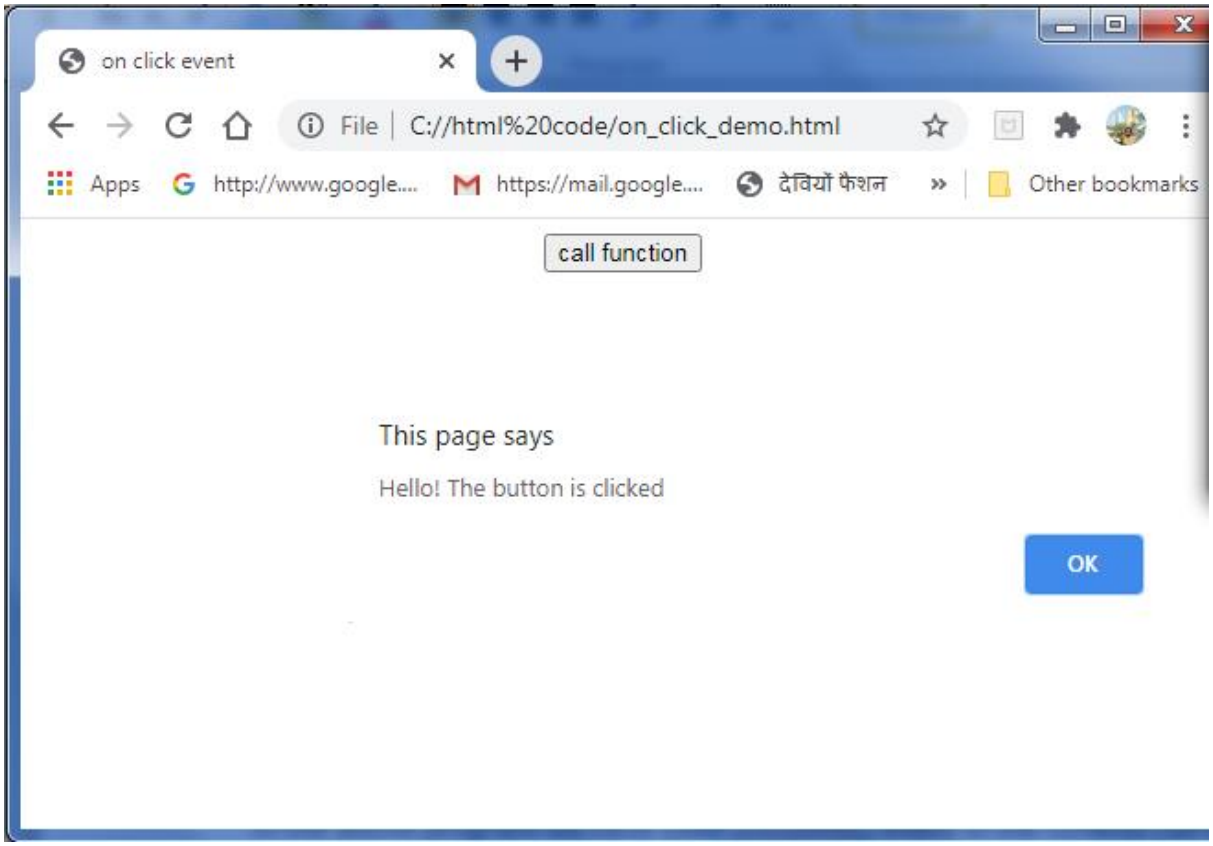
<TITLE> on click event </TITLE>

<SCRIPT>

function clik()

{

```
alert("Hello! The button is clicked");

}

</SCRIPT>

</head>

<Body>

<center>

<form>

<input type="button" value="call function" onclick="clik()">

</form>

</center>

</Body>

</HTML>
```

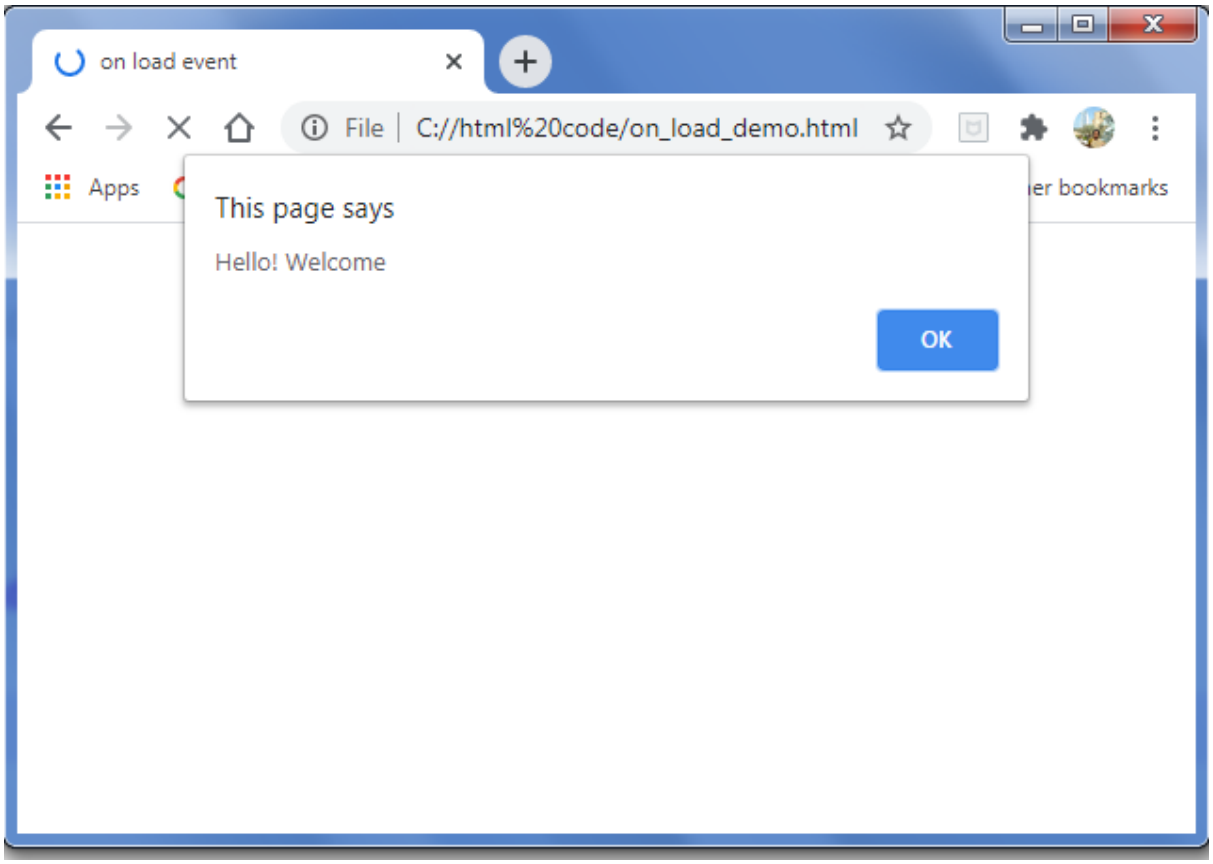


Onload: this event gets activated as soon as the web page gets loaded in the browser's window.

Here is the example:

```
<HTML>
```

```
<head>
<TITLE> on load event </TITLE>
<SCRIPT>
function load1()
{
// this message wiil be displayed non page loading
alert("Hello! Welcome");
}
</SCRIPT>
</head>
<Body onload="load1()">
<center>
</center>
</Body>
</HTML>
```



Objects

Javascript has many built in objects which posses the capability of performing many tasks. Every objects consists of attributes and behaviours.

Let us discuss some commonly used objects of javascript along with their attributers and behaviours.

Date: This object is used for obtaining the date and time. By default, JavaScript will use the browser's time zone and display a date as a full text string:

The following are the commonly used methods of date object:

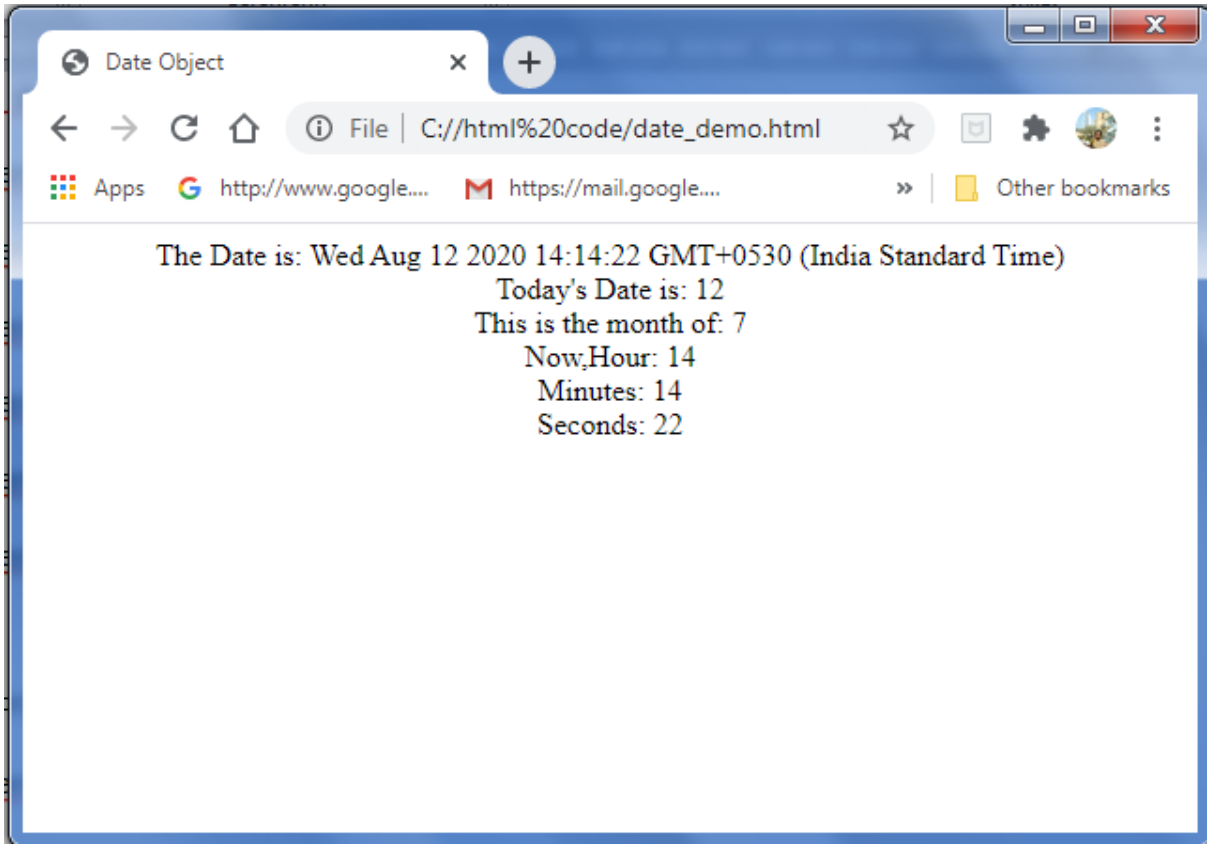
Method	**Description**
getDate()	Get the **day** as a number (1-31)
getHours()	Get the **hour** (0-23)
getMinutes()	Get the **minute** (0-59)
getSeconds()	Get the **second** (0-59)
getMilliseconds()	Get the **millisecond** (0-999)
getTime()	Get the time (milliseconds since January 1, 1970)
getDay()	Get the weekday as a number (0-6)
getUTCDate()	Same as getDate(), but returns the UTC date
getUTCDay()	Same as getDay(), but returns the UTC day

getUTCHours()	Same as getHours(), but returns the UTC hour
getUTCMilliseconds()	Same as getMilliseconds(), but returns the UTC milliseconds
getUTCMinutes()	Same as getMinutes(), but returns the UTC minutes
getUTCSeconds()	Same as getSeconds(), but returns the UTC seconds
setDate()	Set the day as a number (1-31)
setHours()	Set the hour (0-23)

The following program demonstrates Date() objects with some methods.

```
<HTML>
<head>
<TITLE> Date Object </TITLE>
</head>
<Body>
<center>
<SCRIPT>
var my_date= new Date();
document.write("The Date is: "+ my_date.toString()+"<br>");
document.write("Today's Date is: "+ my_date.getDate()+"<br>");
document.write("This is the month of: "+ my_date.getMonth()+"<br>");
document.write("Now,Hour: " +my_date.getHours()+"<br>");
document.write("Minutes: " +my_date.getMinutes()+"<br>");
document.write("Seconds: " +my_date.getSeconds()+"<br>");
```

```
</SCRIPT>

</center>

</Body>

</HTML>
```



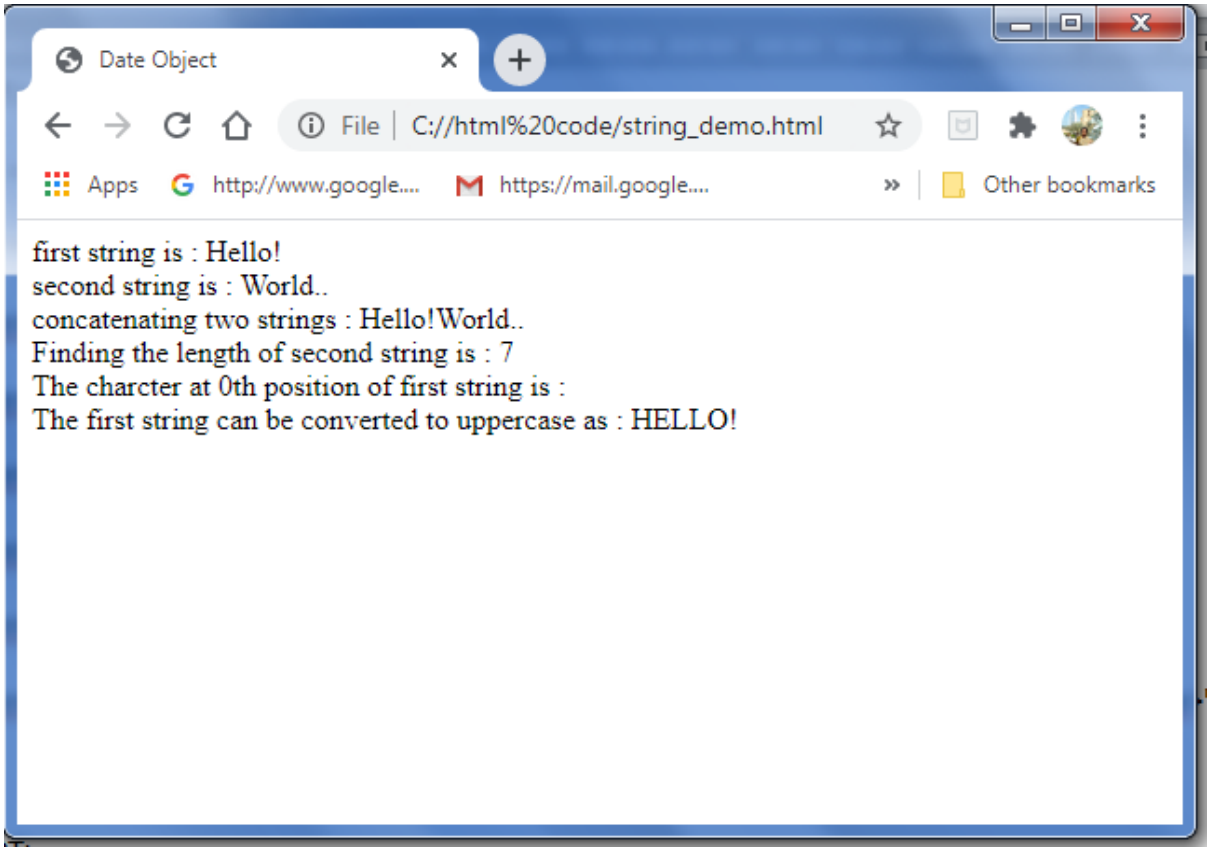
String

String is a collection of characters. In javascript using string object many useful string related functionalities can be exposed off. Here, are some commonly used string functions.

Methods	Description
charAt()	It provides the char value present at the specified index.
concat()	It provides a combination of two or more strings.
replace()	It replaces a given string with the specified replacement.
substr()	It is used to fetch the part of the given string on the basis of the specified starting position and length.
substring()	It is used to fetch the part of the given string on the basis of the specified index.
toLowerCase()	It converts the given string into lowercase letter.

toUpperCase()	It converts the given string into uppercase letter.
valueOf()	It provides the primitive value of string object.
split()	It splits a string into substring array, then returns that newly created array.
trim()	It trims the white space from the left and right side of the string.

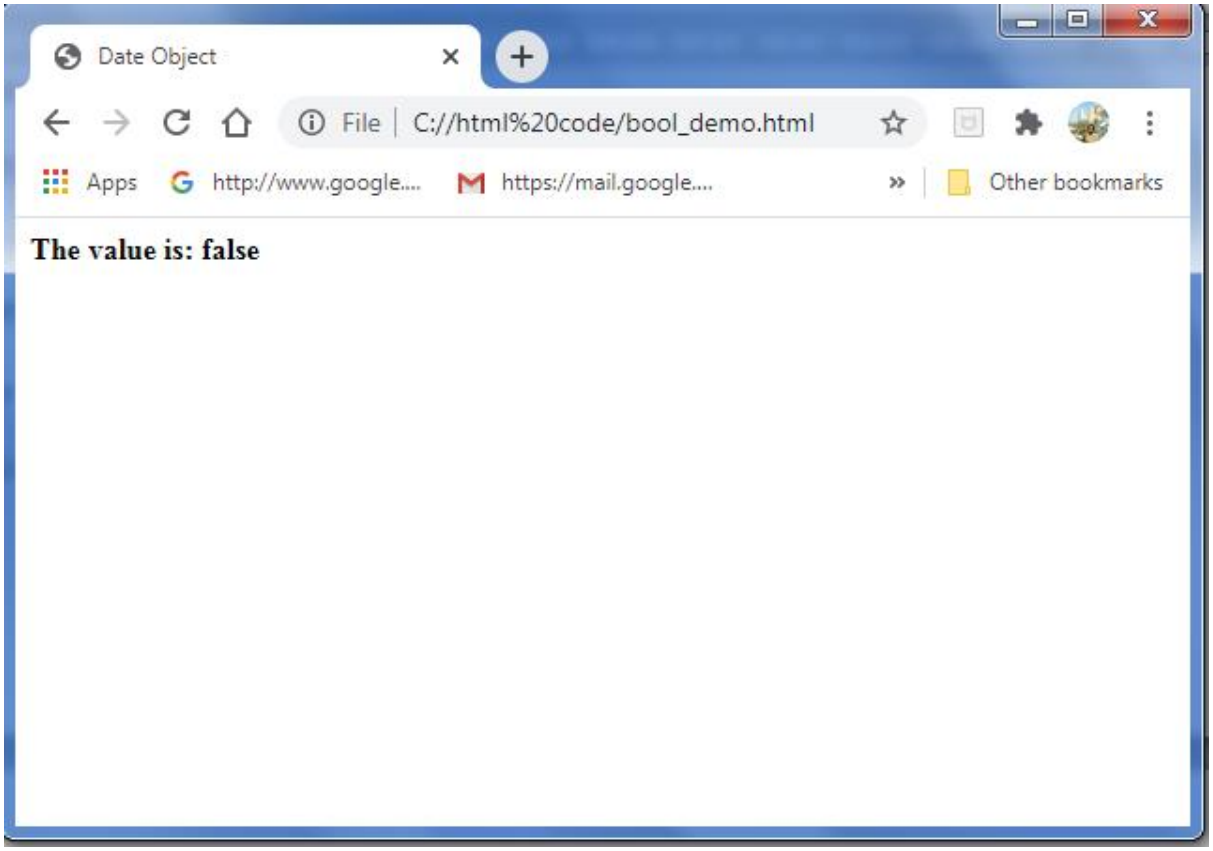
```
<HTML>
<head>
<TITLE> Date Object </TITLE>
</head>
<Body>
<SCRIPT>
var str1 = " Hello!";
var str2 = "World..";
document.write("first string is : " +str1 + "<br>");
document.write("second string is : " +str2 + "<br>");
document.write("concatenating two strings : " +str1.concat(str2) + "<br>");
document.write("Finding the length of second string is : " +str2.length + "<br>");
document.write("The charcter at 0th position of first string is : " +str1.charAt(0) + "<br>");
document.write("The first string can be converted to uppercase as : " +str1.toUpperCase() + "<br>");

</SCRIPT>
</Body>
</HTML>
```



Boolean

The Boolean object is simplest kind of object which is used especially when we want to represent true and false values. Here is the simple javascript in which Boolean type is used-

```
<HTML>

<head>

<TITLE> Boolean Object </TITLE>

</head>

<Body>

<SCRIPT>

var temp = new Boolean(false);

document.write("<b>"+"The value is: ");

document.write(temp.toString());

</SCRIPT>

</Body>

</HTML>
```



Math Object

Math object provides some useful methods to perform mathematical calculations. For example if we want to provide a minimum of two numbers then we can write-

Document.write(math.min(4,5));

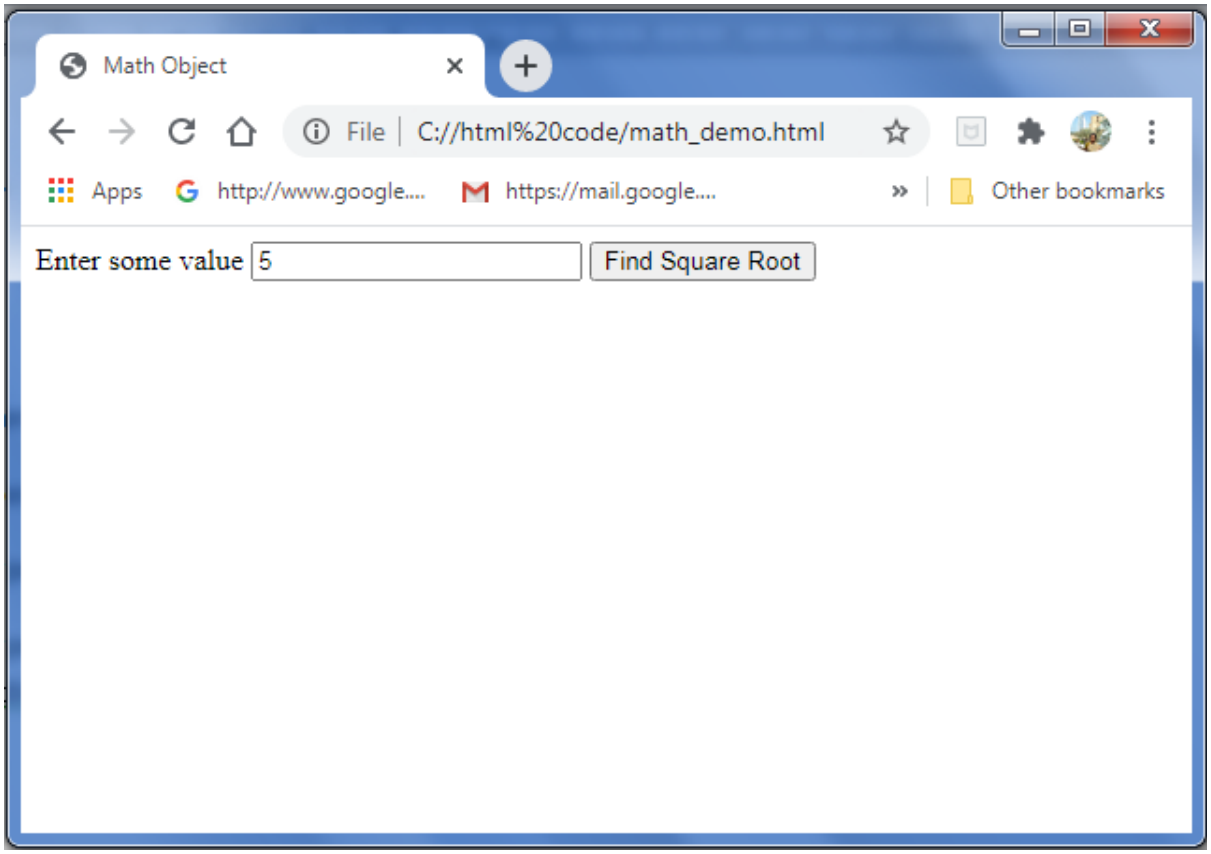
The above statement will result 4. Here are some commonly used methods for math function.

Methods	Description
abs()	It returns the absolute value of the given number.
exp()	It returns the exponential form of the given number.
floor()	It returns largest integer value, lower than or equal to the given number.
max()	It returns maximum value of the given numbers.
min()	It returns minimum value of the given numbers.

sqrt()	It returns the square root of the given number

Here is the example of javascript which uses math() function methods.

```
<HTML>

<head>

<TITLE> Math Object </TITLE>

<script>

function display(num)

{

alert(Math.sqrt(num));

}

</script>

</head>

<Body>

<form name="form1")

<h3> Enter some value </h3>

<input type="text" name="text1">

<input type="button" name="button1" value="Find Square Root"
onclick=display(form1.text1.value)>

</form>

</Body>

</HTML>
```



Dynamic HTML

DHTML is used by the web developer to control how to display and position HTML elements in a browser window.

Difference between HTML and DHTML

- HTML is used to create static web pages and DHTML is used to create dynamic web pages.
- HTML is consists of simple HTML tags, on the other hand DHTML is made of HTML tags+Cascading Style Sheets+ Javascripts.

Cascading Style Sheets(CSS)

Cascading Style Sheets helps us to specify presentation of elements on the webpage. Hence, using CSS we can determine the style and layout of the web page.

There are 3 types of Style sheets—

1. Inline Style Sheets
2. Embedded Style Sheets
3. External Style Sheets

1. Inline Style Sheets

An Inline CSS is used to apply various unique style to a single element. It is also used to define a style for a special types of elements, add a class attribute to the element.

The syntax is

Selector(property:value)

For ex:

```
P
{
Font-family:Ariel;
Color red;
}
```

Here selector p is used for which the properties such as font-family and color are assigned with the values arial and red.

Some times we can group some selectors and set the common properties to them.

For ex:

H1,h2,h3

{

Font-family: arial

}

Note: If you want to define more than one property then we have to separate these properties by the semi colon.

Let us discuss an example CSS program to demonstrate inline style sheets.

<HTML>

<head>

<TITLE> Inline Style sheets </TITLE>

</head>

<Body>

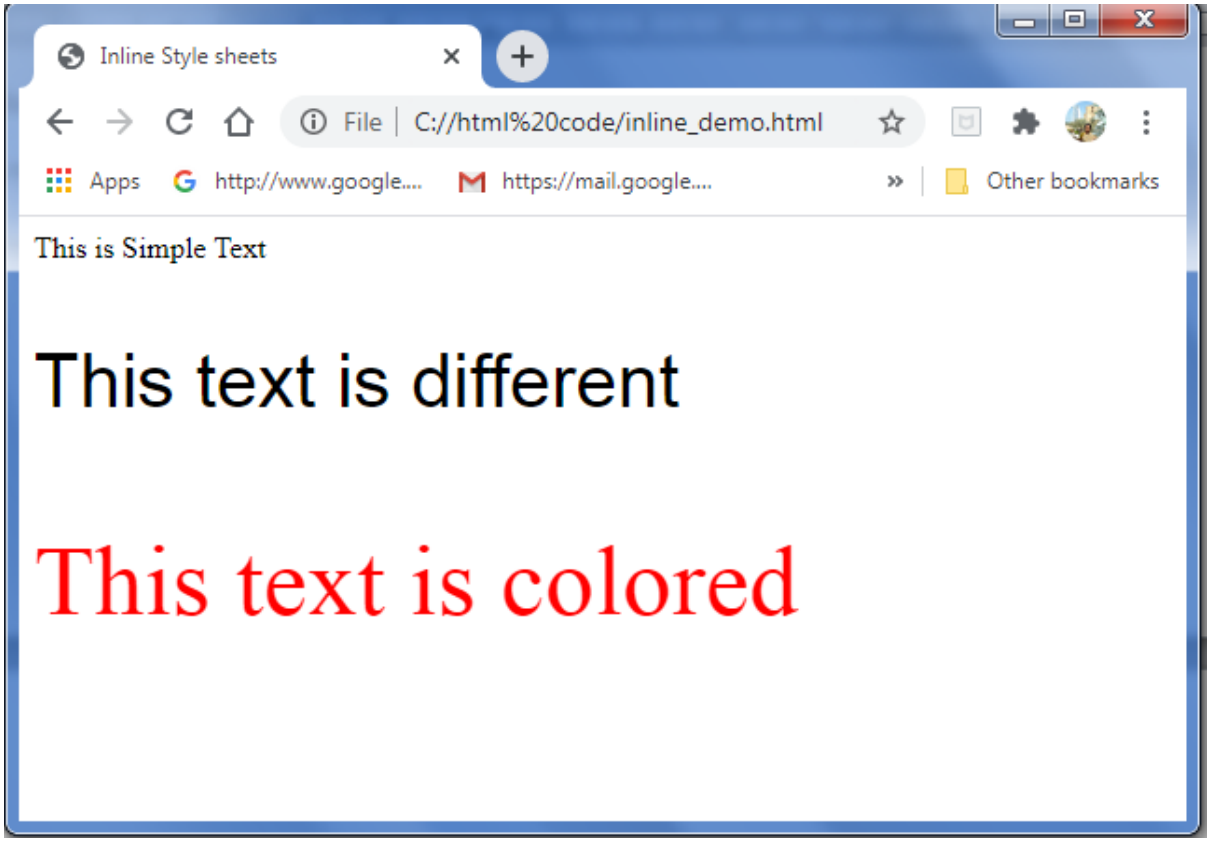
 This is Simple Text

 This text is different

 This text is colored

</Body>

</HTML>



Embedded Style Sheets

For embedded style sheet, we write all desired selectors along with the properties and values in the head section. And in the body section then newly defined selector tags are used with the actual contents.

In the following DHTML script we have deined h1,h2,h3 and p selectors with different properties and values.

Note: to define Embedded style sheets we should mention **style type=”test/css”** in head section.

<HTML>

<head>

<TITLE> Embedded Style sheets </TITLE>

<style type="text/css">

h1,h2,h3{

font-family:arial;}

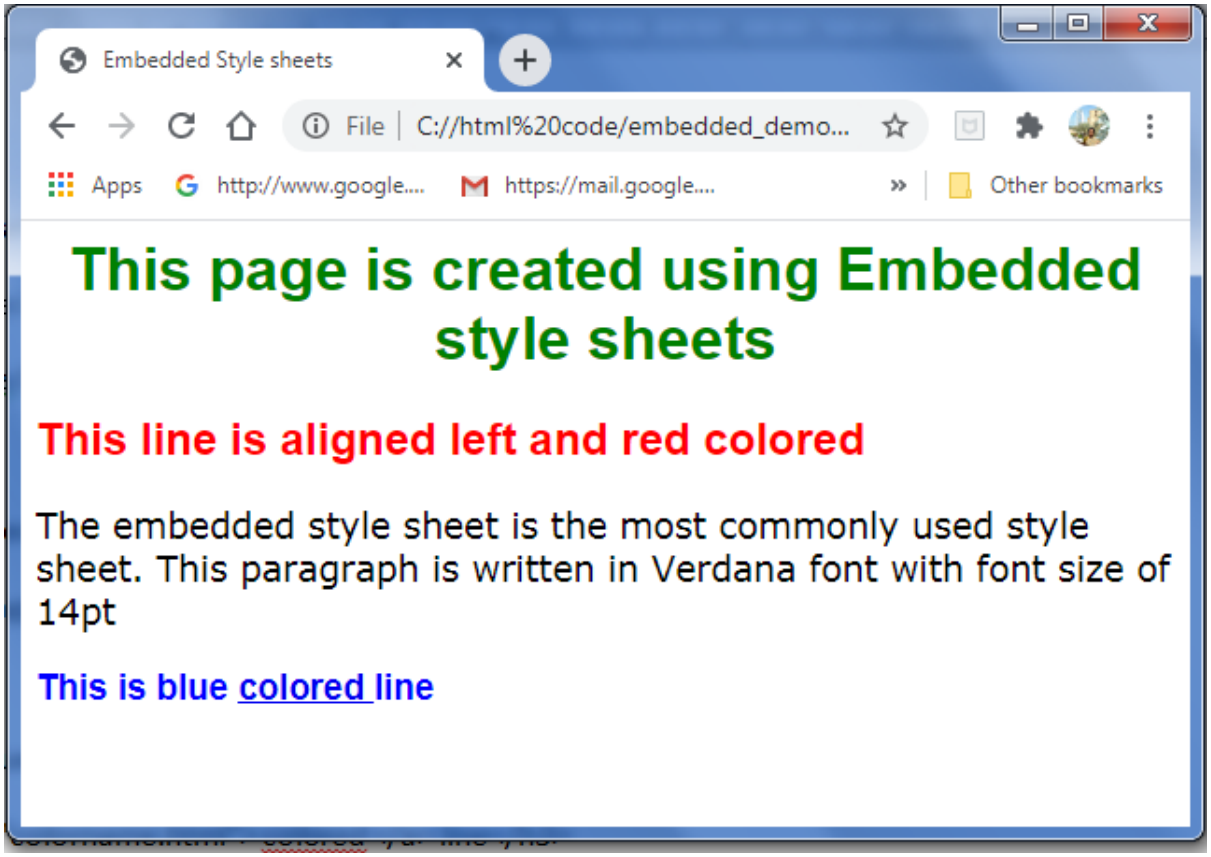
h2 {

color:red;

left:20px }

h3 {

```
color:blue ;
}
p {
font-size:14pt;
font-family:verdana;
}
.special {
color:green}
</style>
</head>
<Body>
<h1 class="special"> <center>
This page is created using Embedded style sheets </center> </h1>
<h2> This line is aligned left and red colored </h2>
<p>
The embedded style sheet is the most commonly used style sheet.
This paragraph is written in Verdana font with font size of 14pt</p>
<h3>
This is blue
<a href="colorname.html"> colored </a> line</h3>
</Body>
</HTML>
```



External Style sheets

Some times we need to apply the particular style to more than one web page in such cases external style sheets can be used. The central idea in this type of style sheet is that it is stored in one .css file. And the name of that file has to be mentioned in our web pages. Then the styles defined in .css are applied to these web pages.

Here is the simple program in which external style sheets are used:

<HTML>

<head>

<TITLE> External Style sheets </TITLE>

<link rel="stylesheet" type="txt/css" href=C:\html code\ex1.css"/>

</head>

<Body>

<h1 class="special">

This page is created using External style sheets </h1>

<h2> This line is aligned left and red colored </h2>

<p>

The embedded style sheet is the most commonly used style sheet.

This paragraph is written in Verdana font with font size of 14pt</p>

<h3>

This is blue

 colored line</h3>

</Body>

</HTML>

// Create a file with name ex1.css

h1 {

font-family:arial;

}

h2 {

font-family:times new roman;

color:red;

left:20px;

}

h3 {

font-family:arial;

color:blue

}

p {

font-size:14pt;

font-family:cambria;

}

special

{

color:green }