Web Designing
Introduction to Java Script

Car Skeleton (only body) is HTML
Car Painted or Decorated is CSS
Car Engine and Internal logic is JS

JavaScript Functions

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it.
- A JavaScript function is defined with the **function** keyword, followed by a **name**, followed by parentheses **()**.
- The parentheses may include parameter names separated by commas: (parameter1, parameter2, ...)
- The code to be executed, by the function, is placed inside curly brackets.
- Example:

Code
```
function myFunction(p1, p2) {
         return p1 * p2;
}
```

- **Note** - The arguments inside functions are used as local variables.

JavaScript Functions

- 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.
- The code inside the function will execute when "something" invokes (calls) the function:
  - When an event occurs (when a user clicks a button)
  - When it is invoked (called) from JavaScript code
  - Automatically (self invoked)
- Here is an example of defining a function in JavaScript:
- <script type="text/javascript"> function funName(parameter-list) { statements list } </script>

3

JavaScript Functions

- Let's look at the following example of JavaScript functions, which takes no parameters called fun1 is defined below :

```
<script type="text/javascript">
        function fun1()
        {
          alert("Hello JavaScript, I am Statements of Function
          Definition");
        }
</script>
```

4

JavaScript Functions

- **Call a Function in JavaScript** : To call a function in JavaScript, you have to simply write the name of the function which is going to be called. Here is an example:

```
<script type="text/javascript">
   fun1();
</script>
```

- After calling the above function fun1() which is define in the second above code fragment, the function will tells the browser to give an alert box containing the string "Hello JavaScript, I am Statements of Function Definition“

```
<body>
   <form>
<input type = "button" onclick = “fun1()" value = "Say Hello"> </form>
</body>
```

Using simple function (simple)

```
<!DOCTYPE HTML>
<HTML>
   <HEAD>
         <TITLE>Function</TITLE>
         <SCRIPT language="JavaScript">
                  function simple()                 {
                           alert("This is an alert box. Thank You!!")}
         </SCRIPT>
    </HEAD>
    <BODY onload="simple()">
         <H1>Using Simple JavaScript Function</H1>
         <P>Here we are using a simple JavaScript function that
         displays an alert box on the load event of the Web
         page.</P>
         </BODY>
</HTML>
```

JavaScript Functions

- **JavaScript Function with Parameters or Arguments** : A function with parameters or arguments accepts some values when it is called.
- Basically arguments are the values that you pass to a function, which has corresponding parameters to store them.

Function with argument

```
<!DOCTYPE HTML>
<HTML>
   <HEAD>
         <TITLE>Function with Arguments</TITLE>
   </HEAD>
   <BODY>
         <H1>Passing Arguments to the Functions </H1>
         <SCRIPT language="JavaScript">
                  function total(a,b,c)                    {
                    var sum = a+b+c;
                     document.write("total="+sum);
                  }
                  document.write("The sum is calculated by passing
                  arguments to the function<BR/>");
                  total(4,5,6);
         </SCRIPT>    </BODY>   </HTML>
```

Using parameter

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
        <SCRIPT type="text/JavaScript">
          function parameter(value)          {
                alert(value);         }
        </SCRIPT>
   </HEAD>
   <BODY>
        <FORM>
          <INPUT type="button" onclick="parameter('JavaScript
          function with parameter')" value="Call function">
        </FORM>
           <P>This is an example of JavaScript function with
           parameters</P>
     </BODY>  </HTML>
```

9

Built in Global Functions in JS

Function	Description
eval()	The **eval()** function evaluates JavaScript code represented as a string.
isFinite()	The global **isFinite()** function determines whether the passed value is a finite number. If needed, the parameter is first converted to a number.
isNaN()	The **isNaN()** function determines whether a value is NaN or not. Note, coercion inside the isNaN function has interesting rules; you may alternatively want to use Number.is.Nan(), as defined in ECMAScript 2015.
parseFloat()	The **parseFloat()** function parses an argument (converting it to a string first if needed) and returns a floating point number.
parseInt()	The **parseInt()** function parses a string argument and returns an integer of the specified radix (the base in mathematical numeral systems).
escape	Returns the hexadecimal encoding of an argument in the ISO Latin-1 character set.
unescape	Returns the ASCII string for the specified value.
Number()	Converts a value of an object into a number.

10

JavaScript Functions

- **JavaScript return Statement :** The return statement in JavaScript returns a value from a function.

```
<script type="text/javascript">
    function calmult(a, b)
    {
            var mult = a*b;
            return mult;
    }
    document.write("The multiplication result of any two
    number is calculated using calmult() function<br/>");
    var m = calmult(10, 20);
    document.write("Multiplication Result = " + m );
</script>
```

11

Using Return Statement

```
<!DOCTYPE HTML>
<HTML>
   <HEAD>
         <TITLE>RETURN</TITLE>
   </HEAD>
   <BODY>
         <H1>Using the return Statement </H1>
         <SCRIPT language="JavaScript">
                 function area (l, b)                      {
                         var rectangle = l*b;
                         return rectangle;                          }
                 document.write("The area of the rectangle is
               calculated using the area() function <BR/>");
                 var rect=area (4,6);
                 document.write("Area of rectangle is: "+rect);
         </SCRIPT>
   </BODY>
</HTML>
```

12

JavaScript Functions

- **JavaScript Variable Scope :** In JavaScript, the variable's scope is basically the region of your program where it is defined. There are following two types of variable scopes available in JavaScript:
  - Local Variables
  - Global Variables
- **JavaScript Local Variables :** A local variable in JavaScript, will be visible only that function where it is declared.
- **JavaScript Global Variables :** A global variable in JavaScript, will be visible anywhere in the JavaScript program.

13

Function Scope and Closure

```
<!DOCTYPE HTML>
<HTML>
   <HEAD>
         <TITLE>Closures</TITLE>
  </HEAD>
  <BODY>
         <H1>Using the Function Scope and Closure</H1>
         <SCRIPT language="JavaScript">
            // call function to display greeting message
           Showmessage();
           // "Hello, World" function
           function Showmessage()
          {
             // declaring new variables
             var date = new Date();
             var hour = date.getHours();
```

14

Function Scope and Closure

```
  var min = date.getMinutes();
  var month = date.getMonth() + 1
  var day = date.getDate()
  var year = date.getFullYear()
      // call DisplayGreeting
  Display();
// display greeting
function Display(){
      if (hour >= 22 || hour <= 5)
          document.write("Goodnight, world!  BR/>");
      else
              document.write("Hello, world! <BR/>");
      document.write ("Today is " + month + "/" + day +
       "/" + year + "<BR/>");
      document.write ("The current time is " + hour + ":"
      + min );                                  }
```

15

Function Scope and Closure

```
                }
          </SCRIPT>
      </BODY>
</HTML>
```

16

JavaScript Functions

- **JavaScript setTimeout() Method :** The setTimeout() method in JavaScript is used to specify the time interval after which the code executes.
- You can use the setTimeout() method to delay the execution of specific code.
- **JavaScript setInterval() Method :** The setInterval() method in JavaScript is used to execute the code after every specified time intervals.
- **JavaScript clearTimeout(timer) Method:** Deactivates or cancels the timer that is set using the setTimeout().
- **JavaScript clearInterval() Method :** Deactivates or cancels the timer that is set using the setInterval().

Time Message

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
        <SCRIPT type="text/javascript">
        function timedMsg(){
                var t=setInterval("alert('1seconds!')",1000);}
        </SCRIPT>
  </HEAD>
  <BODY>
     <FORM>
        <INPUT type="button" value=" timed alert box!"
        onClick="timedMsg()" />
     </FORM>
  </BODY>
</HTML>
```

Using timer

```
<!DOCTYPE HTML>
<HTML>
   <HEAD>
       <SCRIPT type="text/javascript">
         function timedMsg(){
           var t=setTimeout("alert('5 seconds!')",5000);}
       </SCRIPT>
   </HEAD>
   <BODY>
     <FORM>
         <INPUT type="button" value=" timed alert box!"
         onClick="timedMsg()" />
     </FORM>
   </BODY>
</HTML>
```

19

JavaScript Events

- Events in JavaScript, refer to actions that are detected by a JavaScript program when you perform a particular task.
- For example, **onclick** event is detected by the program when you click the mouse button. Here, the following code fragment shows how to create an event handler in JavaScript:
- onEvent = "code to handle the event"

20

Form Events

Attribute	Description
onblur	Fires the moment that the element loses focus
onchange	Fires the moment when the value of the element is changed
oncontextmenu	Script to be run when a context menu is triggered
onfocus	Fires the moment when the element gets focus
oninput	Script to be run when an element gets user input
onforminput	Triggers when input is provided on a form
onformchange	Triggers when a form changes
oninvalid	Script to be run when an element is invalid
onreset	Fires when the Reset button in a form is clicked
onsearch	Fires when the user writes something in a search field (for <input="search">)
onselect	Fires after some text has been selected in an element
onsubmit	Fires when a form is submitted

Onreset Event

```
<!DOCTYPE HTML>
<HTML>
    <HEAD>
        <TITLE>onreset Event</TITLE>
    </HEAD>
    <BODY>
        <H1>The onreset Event</H1>
        <FORM onreset="alert('onreset event is triggered when the
         user clicks the RESET button')">
            <B>Application Form</B>
            <BR>
            <BR>
             <B>Name:</B><INPUT type="text" name="fname"
             value=" ">
             <BR>
```

Onreset Event

```
        <B>Roll Number:</B><INPUT type="text" id="fnumber"
         value=" ">
          <BR>
          <B>Address:</B><INPUt type="text" name="faddress"
           value=" ">
          <BR>
          <B>Email id:</B><INPUT type="text" name="femail"
          value=" ">
          <BR>
          <INPUt type="reset" value="RESET">
    </FORM>
  </BODY>
</HTML>
```

23

Onsubmit Event

```
<!DOCTYPE HTML>
<HTML>
    <HEAD>
          <TITLE>onsubmit  Event</TITLE>
    </HEAD>
   <BODY>
          <H1>The onsubmit Event</H1>
          <FORM onsubmit ="alert('All the details of this form are
          submitted')">
             <B>Application Form</B>
             <P>Fill the form and click the SUBMIT button</P>
             <BR>
              <B>Name:</B><INPUT type="text" name="fname"
              value="">
              <BR>
```

24

Onsubmit Event

```
            <B>Roll Number:</B><INPUT type="text" id="fnumber"
             value="">
             <BR>
              <B>Address:</B><INPUt type="text" name="faddress"
              value="">
              <BR>
              <B>Email id:</B><INPUT type="text" name="femail"
              value="">
              <BR>
              <INPUT type="submit" value="SUBMIT">
           </FORM>
     </BODY>
</HTML>
```

Keyboard Events

Event	Attribute	Description
keydown	onkeydown	The event occurs when the user is pressing a key or holding down a key
keypress	onkeypress	The event occurs when the user is pressing a key or holding down a key
keyup	onkeyup	The event occurs when a keyboard key is released

Mouse Events

Event	Attribute	Description
click	onclick	The event occurs when the user clicks on an element
dblclick	ondblclick	The event occurs when the user double-clicks on an element
mousedown	onmousedown	The event occurs when a user presses a mouse button over an element
mousemove	onmousemove	The event occurs when a user moves the mouse pointer over an element
mouseover	onmouseover	The event occurs when a user mouse over an element
mouseout	onmouseout	The event occurs when a user moves the mouse pointer out of an element
mouseup	onmouseup	The event occurs when a user releases a mouse button over an element

Onclick Event

```
<!DOCTYPE HTML>
<HTML>
        <HEAD>
                <TITLE>onclick Event</TITLE>
        </HEAD>
        <BODY>
                <H1>Using the onclick Event </H1>
                <FORM name="form">
                        Name:<INPUT type="text" id="field"
                        value=" " ><BR/>
                        <BUTTON onclick="alert('Welcome, ' +
                        document.getElementById('field').value)">
                        Click me </BUTTON>
                </FORM>
        </BODY>
</HTML>
```

Mouse down and up Event

```
<!DOCTYPE HTML>
<HTML>
        <HEAD>
                <TITLE>onload Event</TITLE>
        </HEAD>
        <BODY>
                <H1>Using the Mouse Events </H1>
                <P> click this Button </P>
                <BUTTON onmousedown =
                   "document.images['img'].src='car.jpg’”
                    onmouseup =
                   "document.images['img'].src='go.png'" >
                    <IMG NAME="img" >
                </BUTTON>
        </BODY>
</HTML>
```

29

Frame/Object Events

Event	Attribute	Description
abort	onabort	The event occurs when an image is stopped from loading before completely loaded (for <object>)
error	onerror	The event occurs when an image does not load properly (for <object>, <body> and <frameset>)
load	onload	The event occurs when a document, frameset, or <object> has been loaded
resize	onresize	The event occurs when a document view is resized
scroll	onscroll	The event occurs when a document view is scrolled
unload	onunload	The event occurs when a document is removed from a window or frame (for <body> and <frameset>)

30

Media Events

- Events triggered by medias like videos, images and audio (applies to all HTML elements, but is most common in media elements, like <audio>, <embed>, <img>, <object>, and <video>).

Attribute	Description
onabort	Script to be run on abort
oncanplay	Script to be run when a file is ready to start playing (when it has buffered enough to begin)
oncanplaythrough	Script to be run when a file can be played all the way to the end without pausing for buffering
oncuechange	Script to be run when the cue changes in a <track> element
ondurationchange	Script to be run when the length of the media changes
onemptied	Script to be run when something bad happens and the file is suddenly unavailable (like unexpectedly disconnects)

Media Events

Attribute	Description
onended	Script to be run when the media has reach the end (a useful event for messages like "thanks for listening")
onerror	Script to be run when an error occurs when the file is being loaded
onloadeddata	Script to be run when media data is loaded
onloadedmetadata	Script to be run when meta data (like dimensions and duration) are loaded
onloadstart	Script to be run just as the file begins to load before anything is actually loaded
onpause	Script to be run when the media is paused either by the user or programmatically
onplay	Script to be run when the media is ready to start playing

Media Events

Attribute	Description
onplaying	Script to be run when the media actually has started playing
onprogress	Script to be run when the browser is in the progress of getting the media data
onratechange	Script to be run each time the playback rate changes (like when a user switches to a slow motion or fast forward mode)
onseeked	Script to be run when the seeking attribute is set to false indicating that seeking has ended
onseeking	Script to be run when the seeking attribute is set to true indicating that seeking is active
onstalled	Script to be run when the browser is unable to fetch the media data for whatever reason
onsuspend	Script to be run when fetching the media data is stopped before it is completely loaded for whatever reason

33

Media Events

Attribute	Description
ontimeupdate	Script to be run when the playing position has changed (like when the user fast forwards to a different point in the media)
onvolumechange	Script to be run each time the volume is changed which (includes setting the volume to "mute")
onwaiting	Script to be run when the media has paused but is expected to resume (like when the media pauses to buffer more data)

34

Window Events

- Events triggered for the browser/window object (applies to the <body> tag):

Attribute	Description
onafterprint	Script to be run after the document is printed
onbeforeprint	Script to be run before the document is printed
onbeforeunload	Script to be run when the document is about to be unloaded
onerror	Script to be run when an error occurs
onhashchange	Script to be run when there has been changes to the anchor part of the a URL
onload	Fires after the page is finished loading
onmessage	Script to be run when the message is triggered
onoffline	Script to be run when the browser starts to work offline

35

Window Events

Attribute	Description
ononline	Script to be run when the browser starts to work online
onpagehide	Script to be run when a user navigates away from a page
onpageshow	Script to be run when a user navigates to a page
onpopstate	Script to be run when the window's history changes
onresize	Fires when the browser window is resized
onstorage	Script to be run when a Web Storage area is updated
onunload	Fires once a page has unloaded (or the browser window has been closed)

36

Onload Event

```
<!DOCTYPE HTML>
<HTML>

        <HEAD>
                <TITLE>onload Event</TITLE>
        </HEAD>

        <BODY onload = "alert ('Welcome to javascript!')">
                <H1>Using the onload Event </H1>
                <P>This example displays an alert box at the load
                 event of the Web page.</P>
        </BODY>

</HTML>
```

37

Onresize Event

```
<!DOCTYPE HTML>
<html>
  <head>
    <script type="text/javascript">
        function OnResizeDocument () {
             document.body.style.backgroundColor = "red";       }
       </script>
  </head>
  <body onresize="OnResizeDocument ()">
    <br /><br />
    When the size of the document changes, an onresize event is
    fired on the body element in Internet Explorer.In Firefox, Opera,
    Google Chrome and Safari, an onresize event is fired on the
    body element when the browser window is resized. Resizing the
    browser window fires the onresize event in Internet Explorer,
    too.Try it, resize the browser window!    <br /><br />
  </body>
</html>
```

38

JavaScript in Image map

- An image map in JavaScript is an image on a web page that provides various links to navigate to other web pages or some sections of the same web page. You can say those various links as hotspots.
- In an image map in JavaScript to provide hotspot link, you can use any shape such as rectangle, circle, or polygon.
- If you want to create a rectangular image map, then you need two different co-ordinates, such as top right and bottom left.
- If you want to create a circular image map, then you need centre co-ordinate. If you want to create a polygon image map, then you need different number of co-ordinates.
- And last, if you want to create a pentagon shape image map, then you need five co-ordinates.

39

JavaScript in Image map

- Use MAP element to define image maps. Every image map has a unique name.
- Therefore, the name attribute is required in the MAP element. You can add usemap attribute to the IMG element to associate an image map with an image.
- You can also add events to the image map using the AREA element.

40

JavaScript in image map

```
<!DOCUMENT HTML>
<HTML>
   <HEAD>
         <TITLE>Using javaScript in Image map</title>
         <script type="text/javascript">
            function details(txt)          {
              document.getElementById("description").src=txt
            }
         </SCRIPT>
   </HEAD>
   <BODY>
     <IMG src ="flower.jpg" width ="300" height ="300"
       alt="flowers" usemap="#flowermap" />
     <MAP name="flowermap">
          <AREA shape ="rect" coords ="2,7,22,50"
         onmouseover="details('flower1.jpg')" />
```

41

JavaScript in image map

```
         <AREA shape ="circle" coords ="90,60,10"
          onmouseover="details('flower2.jpg')" />
         <AREA shape ="circle" coords ="90,18,16"
          onmouseover="details('flower3.jpg')" />
     </MAP>
     <BR>
     <IMG id="description" width ="300" height ="300" >
     <P id="description1"></P>
   </BODY>
</HTML>
```

42

JavaScript Animation

- Animation is a type of optical illusion where the rapid display of a sequence of images or frames of 2D or 3D artwork creates an illusion of movement. An animation movie or cartoon film is an example of animation.
- Two types of Animation
  - **Sprite Animation** : Sprite animation defines a rectangular image in which the parts of the image are made transparent where you want to show the background.
    - It has the ability to move an image over another image.
  - **Frame Animation :** Frame animation allows you to create fast slide shows. In this animation, you can create a number of slides or frames by making small changes in each frame.
    - As the slide sets are displayed in quick succession, therefore, the changes appears as motion.

43

JavaScript Animation

- A high-quality animation contains the combination of both the animations, that is, the sprite animation and the frame animation.
- JavaScript provides timing functions to create animation which are listed in the following table.

Function	Description
setTimeout(function, duration)	This function is used to execute the code sometime in the future
setInterval(function, duration)	This function is used to execute the code after specified intervals of time
clearTimeout(setTimeout_variable)	This function is used to clear the timer set by the setTimeout() function

44

JavaScript Animation

```
<!DOCTYPE HTML>
<HTML>
   <HEAD>
         <TITLE>JavaScript Animation</TITLE>
         <SCRIPT type="text/javascript">
            var imgObj = null;
            var animate ;
            function init(){
               imgObj = document.getElementById('myImage');
               imgObj.style.position= 'relative';
               imgObj.style.left = '0px';
               imgObj.style.right = '0px';
             }
            function moveRight(){
                imgObj.style.left = parseInt(imgObj.style.left) + 10 +
                'px';
```

JavaScript Animation

```
         animate = setTimeout(moveRight,200);
         // call moveRight in 20msec
       }
      function moveLeft(){
         imgObj.style.left = parseInt(imgObj.style.left) - 10 + 'px';
         imgObj.style.right = parseInt(imgObj.style.right) + 10 +
         'px';
         animate = setTimeout(moveLeft,200);
         // call moveLeft in 20msec
      }
      function stop(){
         clearTimeout(animate);
         //imgObj.style.left = '0px’;
      }
      window.onload =init;
    </SCRIPT>          </HEAD>
```

JavaScript Animation

```
    <BODY>
      <FORM>
              <IMG id="myImage" src="Car.jpg" />
              <P>Click the Start button to start aniamtion</P>
              <INPUT type="button" value="Start"
               onclick="moveRight();" />
              <INPUT type="button" value="Left"
              onclick="moveLeft();" />
              <INPUT type="button" value="Stop"
              onclick="stop();" />
      </FORM>
   </BODY>
</HTML>
```

47

JavaScript Object

- As you know that the JavaScript language is totally based on objects.
- An object is just a special kind of data, with properties and methods.
- In JavaScript, you have the following two options to create an object:
  - by creating a direct instance of object
  - by creating an object using a function
- A direct instance of an object in JavaScript, is created by using the new keyword.
- Here is the general form shows how to create an object in JavaScript by creating a direct instance of object:
- `Obj = new object();`

48

JavaScript Object

- You are free to add properties and methods to an object in JavaScript, just by using the dot (.) followed by a property or method name as shown in the below code fragment:
  - Obj.name="Deepak";
  - Obj.branch="CSE";
  - Obj.rollno=15;
  - Obj.getValue();
- From the above code fragment, Obj is the newly created object, name, branch and rollno are its properties and getValue() is its method.

49

JavaScript Object

- **Accessing Object Properties**
  - Properties are the values associated with an object.
  - The syntax for accessing the property of an object is below
    - objectName.propertyName
  - This example uses the length property of the Javascript's inbuilt object(String) to find the length of a string:

```
var message="Hello World!";
var x=message.length;
```

- **Accessing Object Methods**
- Methods are the actions that can be performed on objects.
- You can call a method with the following syntax.

```
objectName.methodName()
```

- This example uses the toUpperCase method of the String object to convert string to upper case:

```
var message="Hello World!";
var x=message.toUpperCase();
```

50

JavaScript Object

- Another way to create an instance of an object is by creating a function template of the object.
- After defining the function template for an object, you need to create an instance of the object by using the **new** keyword.
- A function template for an object is created by using the function keyword.
- You can also add properties to the function template by using this keyword.
- Example of creating function template.

```
<script type = "text/javascript">
function book(title, author)
{
        this.title = title;
        this.author = author;
}
```

51

Object() Constructor

- A constructor is a function that creates and initializes an object. JavaScript provides a special constructor function called Object() to build the object. The return value of the Object() constructor is assigned to a variable.
- The variable contains a reference to the new object. The properties assigned to the object are not variables and are not defined with the var keyword.

52

Object() Constructor

```
<head>
    <title>User-defined objects</title>
    <script type = "text/javascript">
      var book = new Object();   // Create the object
      book.subject = "Perl";     // Assign properties to the object
      book.author  = "Mohtashim";
    </script>
  </head>
  <body>
    <script type = "text/javascript">
      document.write("Book name is : " + book.subject + "<br>");
      document.write("Book author is : " + book.author + "<br>");
    </script>    </body>
```

53

Defining methods for an Object

- The previous examples demonstrate how the constructor creates the object and assigns properties.
- But we need to complete the definition of an object by assigning methods to it.

54

Defining methods for an Object

```
<script type = "text/javascript">
        // Define a function which will work as a method
        function addPrice(amount) {
          this.price = amount;          }
        function book(title, author) {
          this.title = title;
          this.author  = author;
          this.addPrice = addPrice;  // Assign that method as property.
        }
  </script>
  </head>
```

55

Defining methods for an Object

```
<body>
     <script type = "text/javascript">
        var myBook = new book("Perl", "Mohtashim");
        myBook.addPrice(100);
        document.write("Book title is : " + myBook.title + "<br>");
        document.write("Book author is : " + myBook.author +
"<br>");
        document.write("Book price is : " + myBook.price + "<br>");
     </script>
</body>
```

56

Object Method

```
<HTML>
   <head>
      <title> methods using an object</title>
       <script type = "text/javascript">
           // Define a function which will work as a method
           function addPrice(amount) {
                this.price = amount;          }
           function book(title, author) {
               this.title = title;
               this.author  = author;
               this.addPrice = addPrice;
               // Assign that method as property.         }
        </script>
  </head>
  <body>
```

57

Object Method

```
     <script type = "text/javascript">
          var a = '20';
          var b = a = 30;
          document.write(a+b); // will print 60
          var myBook = new book("Perl", "Mohtashim");
          myBook.addPrice(100);
          document.write("Book title is : " + myBook.title + "<br>");
          document.write("Book author is : " + myBook.author +
         "<br>");
         document.write("Book price is : " + myBook.price +
        "<br>");
       var myBook1 = new book("HTML5", "Dream Tech");
       myBook1.addPrice(500);
       document.write("Book title is : " + myBook1.title + "<br>");
       document.write("Book author is : " + myBook1.author +
      "<br>");
```

58

Object Method

```
      document.write("Book price is : " + myBook1.price + "<br>");
    </script>
  </body>
</html>
```

Built in JavaScript Objects

- Number Object
- Array Object
- String Object
- Boolean Object
- Math Object
- RegExp Object
- Date Object

JavaScript String

- A string can be defined as a sequence of letters, digits, punctuation and so on.
- A string in a JavaScript is wrapped with single or double quotes
- Strings can be joined together with the + operator, which is called concatenation.
- For Example : mystring = " my college name is " + " DDU ";
- All the strings in JavaScript are represent as instances of the String object.
- As string is an object type it also has some useful features.
- For Example : lenStr = mystring.length;
    - Which returns the length of the string in integer

61

JavaScript String Object Properties

Property	Description
constructor	gives the function, creates the prototype of a String object
length	gives the length of a string
prototype	adds properties and methods to an object

62

String Property

```
<!DOCTYPE HTML>
<HTML>
   <HEAD>
      <TITLE>String Properties Demo</TITLE>
   </HEAD>
   <BODY>
      <SCRIPT type="text/javascript">
         function employee(name,jobtitle,salary){
              this.name=name;
              this.jobtitle=jobtitle;
              this.salary=salary; }
          var emp1=new employee("Nilima Aahuja","Technical
          writer",10000);
          document.write("Name : "+emp1.name+ "<BR/>
          Designation : "+emp1.jobtitle+"<BR/>  Salary :
          Rs."+emp1.salary);
```

String Property

```
         document.write("<BR/> --------------------<BR/>");
         var emp2=new employee("Shilpa Verma","HR
         manager",40000);
         employee.prototype.doj=2007;
         document.write("Name : "+emp2.name+"<BR/>
         Designation : "+emp2.jobtitle+"<BR/>  Date of Joining :
         "+emp2.doj+"<BR/>Salary : Rs."+emp2.salary);
     </SCRIPT>
    </BODY>
</HTML>
```

String Object Methods

Method	Description
charAt()	gives the character in the specified index
charCodeAt()	gives the Unicode equivalence of the character in the specified index
concat()	joins two strings
fromCharCode ()	converts Unicode to character
indexOf()	gives the position of the first occurrence of the specified character in a string
lastindexof()	gives the position of the last occurrence of a specified value in a string
match()	looks for the match between a regular expression and a string and returns the matches
quote()	writes quotes in JavaScript
replace()	searches for the match between a regular expression and a string, and replaces the matched substring with a new substring

String Object Methods

Method	Description
search()	searches for a match between a regular expression and a string, and returns the position of the match
slice()	gives a part of a string as new string
split()	splits a string into substrings
substr()	extracts characters from a string beginning at the specified start index for the specified number of characters
substring()	gives characters in a string between two specified indices
toLowerCase()	converts a String value into a lowercase letter
toSource()	returns an object literal representing the specified object
toString()	returns a string representing the specified object
toUpperCase()	converts a string into uppercase letter
valueOf()	gives the primitive value of String object

String Functions

```
<!DOCTYPE HTML>
<HTML>
    <HEAD>
        <TITLE>Different String Functions</TITLE>
   </HEAD>
   <BODY>
        <SCRIPT type="text/javascript">
            var mystring="Twinke Twinkle Little Star";
            document.write("String:  "+mystring+"<BR/>");
            document.write("Length:   "+mystring.length+"<BR/>");
            document.write("Bold text:   "+mystring.bold()+"<BR/>");
            document.write("Uppercase text:
            "+mystring.toUpperCase()+"<BR/>");
            document.write("Big text:   "+mystring.big()+"<BR/>");
            document.write("Lowercase text:
            "+mystring.toLowerCase()+"<BR/>");
```

String Functions

```
            document.write("Strike text:
            "+mystring.strike()+"<BR/>");
            document.write("Small text:
            "+mystring.small()+"<BR/>");
            document.write("Substring after 8 characters:
            "+mystring.substring(8) +      "<BR/>")
            document.write("Splitted text:   "+mystring.split('Twinkle')
            + "<BR/>")
            document.write("Index of text:
            "+mystring.indexOf('Twinkle') + "<BR/>")
            document.write("Character at 8th position:
            "+mystring.charAt(8) + "<BR/>")
            document.write("Blinking text:   "+mystring.blink() +
            "<BR/>")
     </SCRIPT>
   </BODY>  </HTML>
```

JavaScript String

- **Find Length of String in JavaScript :** use JavaScript built-in property length. Here is an example:
  - var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
  - var sln = txt.length;
- Here the variable sln will hold the value, 26, which is the length of the string ABCDEFGHIJKLMNOPQRSTUVWXYZ
- **JavaScript Special Characters**
  - As you know that, JavaScript string must be in quotes. Now let's look at the following code:
  - var y = "He is a "Computer Hacker" from New Delhi."
  - Here, the above string will be chopped to "He is a ".
  - To avoid this problem, is simply to use \ escape character. Here is an example:

69

Escape Sequence

- An escape sequence is a sequence of characters that does not represent itself when used inside a character or string, but is translated into another character or a sequence of characters that may be difficult or impossible to represent directly.
- Some Useful Escape sequences :

Character Code	Output
\n	newline
\t	tab
\r	carriage return
\'	single quote
\"	double quote
\\	backslash
\f	form feed
\b	backspace

70

Wrapper method that returns a String Object

- anchor() - Creates an HTML anchor that is used as a hypertext target.
- big() - Creates a string to be displayed in a big font as if it were in a <big> tag.
- blink() - Creates a string to blink as if it were in a <blink> tag.
- bold() - Creates a string to be displayed as bold as if it were in a <b> tag.
- fixed() - Causes a string to be displayed in fixed-pitch font as if it were in a <tt> tag
- fontcolor() - Causes a string to be displayed in the specified color as if it were in a <font color="color"> tag.
- fontsize() - Causes a string to be displayed in the specified font size as if it were in a <font size="size"> tag.
- italics() - Causes a string to be italic, as if it were in an <i> tag.

Wrapper method that returns a String Object

- link() - Creates an HTML hypertext link that requests another URL.
- small() - Causes a string to be displayed in a small font, as if it were in a <small> tag.
- strike() - Causes a string to be displayed as struck-out text, as if it were in a <strike> tag.
- sub() - Causes a string to be displayed as a subscript, as if it were in a <sub> tag
- sup() - Causes a string to be displayed as a superscript, as if it were in a <sup> tag

Form with string object

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
    <TITLE>Form with Sting Object</TITLE>
  </HEAD>
  <BODY style="background-color:orange">
     <H3>Enter your text in the given text area</H3>
     <FORM name="wordcount">
        <TEXTAREA rows="12" name="wordcount2" cols="38"
        wrap="virtual" style="color:pink;background-
        color:blue;font-size:20px"></textarea><br>
       <INPUT type="button" value="Calculate Words"
        onClick="countit()">
       <INPUT type="text" name="wordcount3" size="20">
    </FORM>
```

Form with string object

```
    <SCRIPT type="text/javascript">
         function countit(){
            var formcontent=document.wordcount.wordcount2.value
            formcontent=formcontent.split(" ")
   document.wordcount.wordcount3.value=formcontent.length
          }
      </SCRIPT>
   </BODY>
</HTML>
```

JavaScript Boolean Object

- The JavaScript Boolean object is a wrapper class and a member of global objects.
- It is used to convert the non-boolean values into Boolean values.
- Boolean object has following two values:
  - true
  - false
- **JavaScript Boolean Object Properties**
  - **Constructor--** returns the function which has created the prototype of the Boolean object
  - **Prototype--** allows you to add properties and methods to an object

JavaScript Boolean Object

- **JavaScript Boolean Object Methods**
  - toString() -- converts the boolean value into string and returns the string
  - stringvalueOf() -- returns the primitive value of a Boolean object
- Here is the general form to create a boolean object in JavaScript: var val = new Boolean(value);
- **JavaScript Boolean() Function**
  - The JavaScript Boolean() function is used to find out if an expression or variable is true or not.
  - Here is an example: Boolean(20 > 9)   // returns true

Boolean Test

```
<!DOCTYPE HTML>
<HTML>
   <HEAD>
     <TITLE>Boolean test</TITLE>
   </HEAD>
   <BODY>
      <SCRIPT type="text/javascript">
         var bool1=new Boolean(0);
         var bool2=new Boolean(8);
         var bool3=new Boolean(1);
         var bool4=new Boolean("");
         var bool5=new Boolean("Hello");
         var bool6=new Boolean('False’);
         var bool7=new Boolean(null);
         var bool8=new Boolean(NaN);
         document.write("0 value is boolean "+bool1+"<br>");
```

Boolean Test

```
         document.write("8 value is boolean "+bool2+"<br>");
         document.write("1 value is boolean "+bool3+"<br>");
         document.write("An empty string is boolean
         "+bool4+"<br>");
         document.write("String \"Hello\" is boolean
         "+bool5+"<br>");
         document.write("String 'False' is boolean "+bool6+"<br>");
         document.write("null is boolean "+bool7+"<br>");
         document.write("NaN is boolean "+bool8);
     </SCRIPT>
   </BODY>
</HTML>
```

JavaScript Number Object

- The Number object represents numerical data, either integers or floating-point numbers.
- In general, you do not need to worry about Number objects because the browser automatically converts number literals to instances of the number class.
- The syntax for creating a number object is as follows −
  - var val = new Number(number);
- In the place of number, if you provide any non-number argument, then the argument cannot be converted into a number, it returns NaN (Not-a-Number).
  - var num= new Number("12" , "12");

79

Number Object Properties

Property	Description
constructor	holds the value of the constructor function that has created the object
MAX VALUE	gives the maximum value
MIN VALUE	gives the minimum value
NEGATIVE INFINITY	represents the value of negative infinity
POSITIVE INFINITY	represents the value of infinity
prototype	adds properties and methods to the Number object

80

Number Object Methods

Method	Description
toExponential(x)	converts a number into an exponential notation
toFixed(x)	rounds up a number to x digits after the decimal
toPrecision(x)	rounds up a number to a length of x digits
toString()	gives a string value for the Number object
valueOf()	gives a primitive value for the Number object

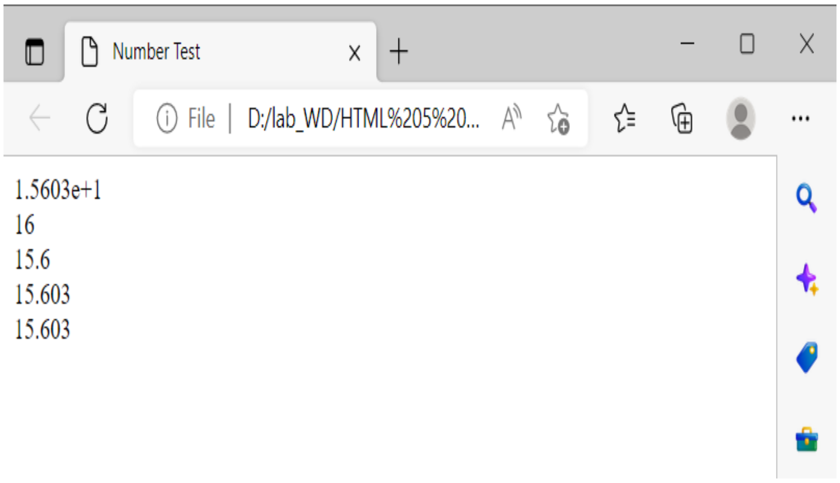
81

Number Test

```
<!DOCTYPE HTML>
<HTML>
    <HEAD>
        <TITLE>Number Test</TITLE>
     </HEAD>
     <BODY>
        <SCRIPT type="text/javascript">
             num= new Number('15.603’)
             document.write(num.toExponential()+ "<br>")
             document.write(num.toFixed()+ "<br>")
             document.write(num.toPrecision(3)+"<br>")
             document.write(num.toString()+"<br>")
             document.write(num.valueOf()+"<br>")
        </SCRIPT>
    </BODY>
</HTML>
```

82

output



```
1.5603e+1
16
15.6
15.603
15.603
```

JavaScript Array Object

- An array is a collection of data, each item in array has an index to access it.
- Ways to use array in JavaScript
  - var myArray = new Array();
  - myArray[0] = " ddu ";
  - myArray[1] = 222;
  - myArray[2] = false;
  - var myArray = new Array(" ddu " , 123 , true);

JavaScript Array Object

- Here is an example, shows how to create an array in JavaScript:
  - First, using array constructor

    ```
    var fruits = new Array( "Guava", "Apple", "Orange" );
    ```
  - Second, using the array literal notation

    ```
    var fruits = [ "Guava", "Apple", "Orange" ];
    ```
  - The fruits[0] represents the first element which is "Guava" here, and the fruits[2] is the third element which is "Orange" here.

85

Array Object Properties

Property	Description
constructor	holds the value of the constructor function that has created the object
length	holds the number of elements in an array
prototype	adds properties and methods to the Array object

86

Array Object Methods

Method	Description
concat()	joins two or more arrays
join()	joins all the elements of an array into a string
pop()	removes the last element of an array and returns that element
push()	adds new element as the last element and the returns the length of the new array
reverse()	reverses the order of list of element in an array
shift()	removes the first element of an array and then returns that element
slice()	selects a part of an array and then returns that part as a new array
sort()	sorts the elements of an array
splice()	adds or removes the elements of an array
tostring()	converts an array into a string and then returns the string
unshift()	adds new elements to an array and then returns the new length
valueof()	returns the primitive value of an Array object

87

Array Functions

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
     <TITLE>Array Functions</TITLE>
  </HEAD>
  <BODY>
     <SCRIPT type="text/javascript">
       myarray1=new Array("Sunday", "Monday",
       "Tuesday","Wednesday","Thursday");
       myarray2=new Array(1,2,3,4,5);
       document.write("Array 1 : "+myarray1+"<BR/>")
       document.write("Array 2 : "+myarray2+"<BR/>")
       document.write("Concatenated two arrays :
       "+myarray1.concat(myarray2)+"<BR/>")
      document.write("Joined two arrays :
      "+myarray1.join(myarray2)+"<BR/>")
```

88

Array Functions

```
 document.write("Array 1 pop out :
 "+myarray1.pop()+"<BR/>")
 document.write("Reversed array 1:
 "+myarray1.reverse()+"<BR/>")
 document.write("Shifted array 1 :
"+myarray1.shift()+"<BR/>")
document.write("Sorted array 1 : "+myarray1.sort()+"<BR/>")
document.write("Array 2 in string format
"+myarray2.toString()+"<BR/>")
 </SCRIPT>
 </BODY>
</HTML>
```

89

JavaScript Splice()

- The JavaScript splice() method does following three jobs:
  - adds new elements to the array
  - removes specified number of elements from array
  - adds new elements and removes specified number of elements from specified array, at once

90

JavaScript Splice() Example

```
<!DOCTYPE html>
<html>
<body>
  <script>
    const myarr = ["One", "Two", "Three", "Four", "Five", "Six"];
    myarr.splice(1, 3, "MCA", "DDU");
    document.write(myarr);
  </script>
</body>
</html>
```

- Output

```
One,MCA,DDU,Five,Six
```

JavaScript Join()

- array.join(separator)
- The separator parameter is used to separate each items of the array when creating the string. The default value is comma (,).

```
<!DOCTYPE html>
<html>
<body>
  <script>
    const cities = ["Tokyo", "Los Angeles", "Bangkok"];
    let str = cities.join("and");
    document.write(str);
  </script>
</body>  </html>
```

Output : TokyoandLos AnglesandBangkok

JavaScript Math Object

- The JavaScript Math object is used to perform simple and complex arithmetic operations.
- The Math object includes several mathematical constants and methods.
- Example for using properties/methods of Math:

Code

```
<script>
               var x=Math.PI;
              var y=Math.sqrt(16);
</script>
```

Math Object Properties

Property	Description
E	Euler's number (approx. value is 2.718)
LN2	natural logarithm of 2 (approx. value is 0.693)
LN10	natural logarithm of 10 (approx. value is 2.302)
LOG2E	base-2 logarithm of E (approx. value is 1.442)
LOG10E	base-10 logarithm of E (approx. value is 0.434)
PI	numerical value of PI (approx. value is 3.142)
SQRT1_2	square root of 1/2 (approx. value is 0.707)
SQRT2	square root of 2 (approx. value is 1.414)

Math Object Methods

Method	Description
abs(x)	gives the absolute value of x
acos(x)	gives arccosine of x (in radian)
asin(x)	gives arcsine of x (in radian)
atan(x)	gives the arctangent of x
atan2(y,x)	gives the arctangent of the quotient on dividing y and x
ceil(x)	rounds up x to the nearest bigger integer
cos(x)	gives cosine value of x
exp(x)	gives the value of ex
floor(x)	rounds up x to the nearest smaller integer
log(x)	gives the natural logarithmic value of x

95

Math Object Methods

Method	Description
max(x,y,z,...,n)	gives the highest number from the given list
min(x,y,z,...,n	gives the lowest number from the given list
pow(x,y)	returns x to the power of y
random()	returns a random number between 0 and 1
round(x)	rounds up x to the nearest integer
sin(x)	gives the sine value of x
sqrt(x)	gives the square root of x
tan(x)	gives the tangent value of x

96

Math Functions

```
<!DOCTYPE HTML>
<HTML>
   <HEAD>
     <TITLE>Math Functions</TITLE>
    </HEAD>
    <BODY>
        <SCRIPT type="text/javascript">
           document.write("Floor : "+Math.floor(12.6)+"<BR/>");
           document.write("Log : "+Math.log(12.6)+"<BR/>");
           document.write("Max :
           "+Math.max(12,34,15,65)+"<BR/>");
           document.write("Min :
           "+Math.min(12,34,15,65)+"<BR/>");
           document.write("Pow : "+Math.pow(10,2)+"<BR/>");
           document.write("Random : "+Math.random()+"<BR/>");
           document.write("Round : "+Math.round(12.6)+"<BR/>");
```

Math Functions (mathfunction)

```
           document.write("Sin : "+Math.sin(45)+"<BR/>");
           document.write("Sqrt : "+Math.sqrt(16)+"<BR/>");
           document.write("Tan : "+Math.tan(45)+"<BR/>");

       </SCRIPT>
   </BODY>
</HTML>
```

JavaScript Date Object

- The Date object in JavaScript is used to display a date on a Web page.
  - **Date Object Properties**

Property	Description
constructor	holds the value of the constructor function that has created the object
prototype	adds properties and methods to the Date object

Date Object Method

Method	Description
getDate()	returns the day of the month (ranges from 1 to 31)
getDay()	returns the numerical equivalence of the day of a week (ranges from 0 to 6). 0 for Monday
getFullYear()	returns the numerical equivalence of the year (in 4 digits)
getHours()	returns the hours (ranges from 0 to 23)
getMilliseconds()	returns the milliseconds (ranges from 0 to 999)
getMinutes()	returns minutes (ranges from 0 to 59)
getMonth()	returns the numerical equivalence of month (ranges from 0 to 11)
getSeconds()	returns the seconds (ranges from 0 to 59)
getTime()	returns the number of milliseconds since midnight Jan 1, 1970

Date Object Method

Method	Description
getTimezoneOffset()	returns the difference of time in minutes between GMT and the local time
getUTCDate()	returns the day of the month (ranges from 1 to 31) as per the universal time
getUTCDay()	returns the numerical equivalence of the day of the week (ranges from 0 to 6) as per the universal time
getUTCFullYear()	returns the year in four digits as per the universal time
getUTCHours()	returns the hour (ranges from 0 to 23) as per the universal time
getUTCMilliseconds()	returns the milliseconds (ranges from 0 to 9999) as per the universal time
getUTCMinutes()	returns the minutes (ranges from 0 to 59) as per the universal time
getUTCMonth()	returns the numerical equivalence of month (ranges from 1 to 31) as per the universal time

Date Object Method

Method	Description
getUTCSeconds()	returns the seconds (ranges from 0 to 59) as per the universal time
Parse()	parses a date string and returns the number of millisecond since the midnight of January 1, 1970
setDate()	sets the day of a month (ranges from 1 to 31)
setFullYear()	sets the year in four digits
setHours()	sets the hours (ranges from 0 to 23)
setMilliseconds()	sets the milliseconds (ranges from 0 to 999)
setMinutes()	sets the minutes (ranges from 0 to 59)
setMonth()	sets the numerical equivalence of month (ranges from 0 to 11)
setSeconds()	sets the seconds (ranges 0 from 59)

Date Object Method

Method	Description
setTime()	sets a date and time by adding or subtracting specified milliseconds to/from midnight January 1, 1970
setUTCDate()	sets the day of the month (ranges from 1 to 1) as per the universal time
setUTCFullYear()	sets the year in four digits as per the universal time
setUTCHours()	sets the hours (ranges from 0 to 23) as per the universal time
setUTCMilliseconds()	sets the milliseconds (ranges from 0 to 999)
setUTCMinutes()	sets the minutes (ranges from 0 to 59) as per the universal time
setUTCMonth()	sets the numerical equivalence of month (ranges from 0 to 11) as per universal time
setUTCSeconds()	sets the seconds (ranges from 0 to 59) as per the universal time

Date Object Method

Method	Description
toDateString()	converts the date into a string
toLocalDateString()	converts the date into a string as per local conventions
toLocalTimeString( )	converts the time into a string as per local convention
toLocalString()	converts the Date object into a string as per local convention
toString()	converts the Date object into a string
toTimeString()	converts time into a string
toUTCString()	converts the Date object into a string as per the universal time
UTC()	holds the number of millisecond since the midnight of January 1, 1970, as per the universal time
valueOf()	returns the primitive value of the Date object

Date Functions

```
<!DOCTYPE HTML>
<HTML>
   <HEAD>
     <TITLE>Date Functions</TITLE>
   </HEAD>
   <BODY>
      <SCRIPT type="text/javascript">
         var mydate= new Date();
         document.write("Today the date
         is:"+mydate.getDate()+"/"+
         (mydate.getMonth()+1)+"/"+mydate.getFullYear()+
         "<BR/>");
        document.write("The time is:"+mydate.getHours()+":"
       +mydate.getMinutes()+":"+mydate.getSeconds()+"<BR/>");
     </SCRIPT>
  </BODY>   </HTML>
```

Validation

- Validation is the process of checking data against a standard or requirement.
- Form validation normally used to occur at the server, after client entered necessary data and then pressed the Submit button.
- If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information.
- This was really a lengthy process which used to put a lot of burden on the server.
- JavaScript provides a way to validate form's data on the client's computer before sending it to the web server.

Validation

- Form validation generally performs two functions.
  1. Basic Validation
     - Emptiness
     - Confirm Password
     - Length Validation etc……
     - Data Format Validation
  2. Secondly, the data that is entered must be checked for correct form and value.
     - Email Validation
     - Mobile Number Validation
     - Enrollment Number Validation etc….

JavaScript RegExp

- A regular expression is an object that describes a pattern of characters.
- The JavaScript RegExp class represents regular expressions, and both String and RegExp define methods that use regular expressions to perform powerful pattern-matching and search-and-replace functions on text.
- A regular expression could be defined with the RegExp () constructor, as follows −

```
var pattern = new RegExp(pattern, attributes);
```

- or simply

```
var pattern = /pattern/attributes;
```

JavaScript RegExp

- Here is the description of the parameters −
  - pattern − A string that specifies the pattern of the regular expression or another regular expression.
  - attributes − An optional string containing any of the "g", "i", and "m" attributes that specify global, case-insensitive, and multi-line matches, respectively.
- Example:

```
var pattern = "^ [\\w]$";   // will allow only words in the string
var regex = new RegExp(pattern);
If(regex.test(testString)){
    //Valid
} else {
    //Invalid
}
```

109

JavaScript RegExp

- Following are three types of flag/attributes of Java Script:
  - **i** : Perform case-insensitive matching.
  - **m** : Specifies that if the string has newline or carriage return characters, the ^ and $ operators will now match against a newline boundary, instead of a string boundary
  - **g** : Performs a global match that is, find all matches rather than stopping after the first match.
  - **Brackets** : Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

110

JavaScript RegExp

Expression & Description	
[...]	Any one character between the brackets.
[^...]	Any one character not between the brackets.
[0-9]	It matches any decimal digit from 0 through 9.
[a-z]	It matches any character from lowercase **a** through lowercase **z**.
[A-Z]	It matches any character from uppercase **A** through uppercase **Z**.
[a-Z]	It matches any character from lowercase **a** through uppercase **Z**.

- The ranges shown above are general; you could also use the range [0-3] to match any decimal digit ranging from 0 through 3, or the range [b-v] to match any lowercase character ranging from b through v.

111

RegExp Object Property

Property	Description
global	refers that the g modifier is set
ignoreCase	refers that the i modifier is set
lastIndex	refers the index to start the next match
multiline	refers that the m modifier is set
source	refers the text of the RegExp pattern

112

RegExp Object Method

Method	Description
compile()	compiles the RegExp object
exec()	tests for a match in a string and returns a result array
test()	tests for a match in a string and returns true or false

```
^(?:(?:\+|0{0,2})91(\s*[\-]\s*)?|[0]?)?[789]\d{9}$
```

Using RegExp

JavaScript

```
<script>
        // Creating regex literal
        var regex = "/India/g";
        var str = "We live in India.";
        // search for india
        var matches = str.match(regex);
        document.write(matches)
</script>

Output
India
```

Email Validation Using RegExp

JavaScript

```
let str = "abc@gmail.com";
// regex pattern to match valid email
let re = /^\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$/
// test the
let result = re.test(str);
if(result)
        document.write("Email is valid")
else
        document.write("Email is not valid")
```

JavaScript User Defined Object

- JavaScript allows you to create your own objects.
- The first step is to use the new operator.
  - var myObj= new Object();
- This creates an empty object.
- This can then be used to start a new object that you can then give new properties and methods.
- In object- oriented programming such a new object is usually given a constructor to initialize values when it is first created.
- However, it is also possible to assign values when it is made with literal values.

JavaScript User Defined Object

example

```
<script>
        person={
                firstname: "DDU",
                lastname: "College",
                        age: 50,
                        eyecolor: "blue"
                }
                alert(person.firstname + " is " + person.age +
        " years
                old.");
          </script>
```

JavaScript User Defined Object Example

```
<!doctype html> <HTML>
<head>
      <title>Create User-defined Object in JavaScript</title>
</head>
<body>
     <script>
          /* defining a new constructor function */
          function Bike(company, model, year) {
            this.company = company;
           this.model = model;
           this.year = year;          }
         /* creating the object */
        let myBike = new Bike("KTM", "Duke", 2010);
```

JavaScript User Defined Object Example

```
            document.write(myBike.company+"<br/>"+myBike.model
+"<br/>"+myBike.year);
          </script>
    </body>
</html>
```

119

120