IT WORKSHOP I

JavaScript 2

25-Jan-23

Topics

- Built-in Objects

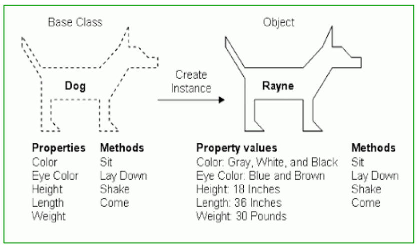
2

Built-in Objects in JavaScript

- JavaScript provides following built-in objects
  - Date
  - Math
  - Array
  - String

JavaScript Date Object

- Date object manipulates date and time.
- The JavaScript date object can be used to get year, month and day.
- You can display a timer on the webpage by the help of date object.
  - popular application is displaying a digital clock.
- Date objects are created with the new Date()
- Once object is created, developer can apply the various Date methods to get and set dates.
- get and set the year, month, day, hour, minute, second, and millisecond fields of the object



Date Object Creation

Syntax

```
var obj=new Date()
```

```
new Date()
new Date(year, month, day, hours, minutes, seconds, milliseconds)
new Date(milliseconds)
new Date(datestring)
```

```
var today = new Date()
var day = new Date(2022, 02, 22, 10, 23, 30);
var day = new Date(05, 11, 17)
var day = new Date(86400000)     //01-01-1970
var day = new Date('December 17, 1995 03:24:00')
var day = new Date('1995-12-17T03:24:00')
```

5

JavaScript Date Input Format

Type	Sample
ISO Date	"2021-02-22" (The International Standard)
Short Date	"02/22/2021"
Long Date	“Feb 22 2021" or "22 Feb 2021"

```
var d = new Date("2022-02-22");
var d = new Date("2022-02");
var d = new Date("2022");
var d = new Date("2015-03-25T12:00:00");
                //YYYY-MM-DDTHH:MM:SS
var d = new Date("02/22/2022");
var d = new Date("Feb 22 2022");
var d = new Date("22 Feb 2022");
```

6

Date get Methods

Method	Description
getFullYear()	returns the year in 4 digit e.g. 2015.
getMonth()	returns the month in 2 digit from 0 to 11. So it is better to use getMonth()+1 in your code.
getDate()	returns the date in 1 or 2 digit from 1 to 31.
getDay()	returns the day of week in 1 digit from 0 to 6.
getHours()	returns the hour (0-23)
getMinutes()	Returns the minutes (0-59)
getSeconds()	returns the seconds (0-59)
getMilliseconds()	returns the milliseconds.
Date.now()	Get the time //milliseconds

7

Date set Methods

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

8

Example

```
<html><body><script>
var d,day,month,year,h,m,s;
   d=new Date();
   day=d.getDate();
   month=d.getMonth()+1;
   year=d.getFullYear();
document.write("<br>Date is: "+day+"/"+month+"/"+year);
   h=d.getHours();
   m=d.getMinutes();
   s=d.getSeconds();
document.write("<br>"+h+":"+m+":"+s);
  d.setDate(11);
document.write("<br>Date is: "+d.getDate());
</script></body></html>
```

9

Example

```
<html><body><script>
var d,day,month,year,h,m,s;
   d=new Date();
   day=d.getDate();
   month=d.getMonth()+1;
   year=d.getFullYear();
document.write("<br>Date is: "+day+"/"+month+"/"+year);
   h=d.getHours();
   m=d.getMinutes();
   s=d.getSeconds();
document.write("<br>"+h+":"+m+":"+s);
  d.setDate(11);
document.write("<br>Date is: "+d.getDate());
</script></body></html>
```

```
Date is: 21/3/2022
20:41:52
Date is: 11
```

10

JavaScript Math Object

- The JavaScript math object provides several constants and methods to perform mathematical operation.
- Unlike date object, it doesn't have constructors.
- All the properties and methods of Math are static
- can be called by using Math as an object without creating it.

Math Properties

Math Property	Description
SQRT2	Returns square root of 2.
PI	Returns Π value.
E	Returns Euler's Constant.
LN2	Returns natural logarithm of 2.
LN10	Returns natural logarithm of 10.
LOG2E	Returns base 2 logarithm of E.
LOG10E	Returns 10 logarithm of E.

```
document.write("<br>The value of PI is: "+Math.PI);
```

Math Methods

Math Property	Description
abs()	Returns the absolute value of a number.
round()	Returns the value of a number rounded to the nearest integer.
ceil()	Returns the smallest integer greater than or equal to a number.
cos()	Returns cosine of a number.
floor()	Returns the largest integer less than or equal to a number.
random()	Returns a random number between 0 and 1.
max(), min()	Returns the largest of zero or more numbers.
sqrt()	Returns the square root of a number.
pow()	Returns base to the exponent power, that is, base exponent.

Math object Example

```
<script>
document.write(Math.sqrt(4));
document.write("<br> Randam no is:" +Math.random());
document.write("<br> power is:"+ Math.pow(2,4));
document.write("<br> floor is:" +Math.floor(4.6));
document.write("<br> ceil is:"+Math.ceil(4.6));
document.write("<br> Round of no is:"+Math.round(4.6))
document.write("<br> absolute no is"+Math.abs(-4))
document.write("<br> absolute no is"+Math.min(2,4,6,1,8))


document.write("<br>The Generating Number between 0 and
10 :  "+Math.floor(Math.random() * 11));
</script>
```

JavaScript Array Object

- JavaScript arrays are used to store multiple values in a single variable.
- JS Arrays can be created in three different ways,
  - Array literal
  - Direct Instantiation (new keyword, empty constructor)
  - Array Constructor (new keyword)

Array literal

- The simple and most efficient way to create an array
- Syntax
  - var variablename=[item1,item2,….]
- Item values are contained inside [ ] and separated by , (comma).

- Example: var cars=[“BMW",“Audi",“Ferrari“,”TATA”];

```
for (i=0;i<cars.length;i++){
document.write(cars[i] + "<br/>");
}
```

- The .length property returns the length of an array.
- Index starts from 0.

Direct Instantiation

- new keyword is used to create instance of array object.
- Syntax: var myArray=new Array()
- Values for the items can be added as,
  - myArray[0] = 3;
  - myArray[1] = "item 1";
  - myArray[2] = "item 2";
- Example

```
<script>
var cars = new Array();
cars[0] = “BMW";
cars[1] = “Ferrari";
cars[2] = “Audi";
for (i=0;i<cars.length;i++){
document.write(cars[i] + "<br>");
}
</script>
```

17

Array Constructor

- create instance of array by passing arguments in constructor
- Syntax: var myArray=new Array(“item1”,”item2”…)
- Example

```
<script>
var cars = new Array(“BMW”,”Audi”,”Ferrari”)
 for (i=0;i<cars.length;i++){
document.write(cars[i] + "<br>");
}
</script>
```

18

Array Constructor

- array element by referring to the index number.
- value of array item can be changed by simple value assignment
  - cars[0] = “Tata";
- to access or display complete array, arrayname cane be used.

```
document.write(cars + "<br>");
```

- the length property of an array returns the length of an array

Array Methods

Method	Description
concat()	Returns a new array comprised of this array joined with other array(s) and/or value(s).
pop()	Removes the last element from an array and returns that element.
push()	Adds one or more elements to the end of an array and returns the new length of the array.
reverse()	Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
shift()	Removes the first element from an array and returns that element.
slice()	Extracts a section of an array and returns a new array.
sort()	Sorts the elements of an array
unshift()	Adds one or more elements to the front of an array and returns the new length of the array.

Array Methods

```
<html>
<head>
</head>
<body>
<script language="javascript">

var cars = ["BMW", "audi", "Ferrari", "Tata"];
document.write("<h2> Before Array operations:</h2>"+cars);
document.write("<br><br>Popped value is "+cars.pop());
document.write("<h2> After POP():</h2>"+cars );
cars.push("Lamborghini");
document.write("<h2> After Push():</h2>"+cars );
cars.shift();
document.write("<h2> After shift():</h2>"+cars );

cars.unshift("Ford");
document.write("<h2> After unshift():</h2>"+cars );
```

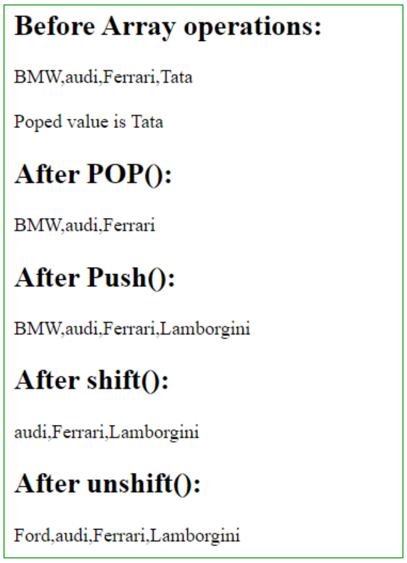
21

Array Methods

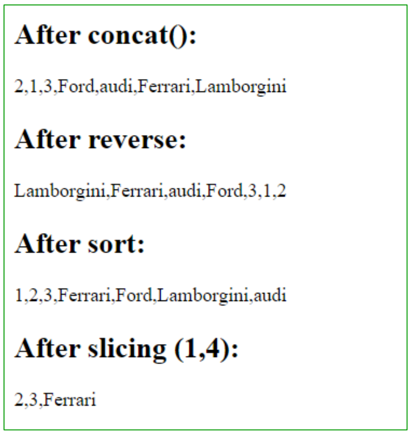
```
cars.unshift("Ford");
document.write("<h2> After unshift():</h2>"+cars );

var num=[2,1,3]
var my= num.concat(cars);

document.write("<h2> After concat():</h2>"+my);
document.write("<h2> After reverse:</h2>"+my.reverse());
document.write("<h2> After sort:</h2>"+my.sort());

document.write("<h2> After slicing (1,4):</h2>"+my.slice(1,4));

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

22

Array Methods



```
Before Array operations:

BMW,audi,Ferrari,Tata

Poped value is Tata

After POP():

BMW,audi,Ferrari

After Push():

BMW,audi,Ferrari,Lamborgini

After shift():

audi,Ferrari,Lamborgini

After unshift():

Ford,audi,Ferrari,Lamborgini
```



```
After concat():

2,1,3,Ford,audi,Ferrari,Lamborgini

After reverse:

Lamborgini,Ferrari,audi,Ford,3,1,2

After sort:

1,2,3,Ferrari,Ford,Lamborgini,audi

After slicing (1,4):

2,3,Ferrari
```

23

JavaScript String

- JavaScript strings are used for storing and handling text values.
- Object that represents a sequence of characters
- JS Arrays can be created in three different ways,
  - Array literal
  - Array Constructor (new keyword)

24

String Literals

- zero or more characters written inside quotes.
- syntax:
  - var stringname="string value";

```
<script>
var str="JavaScript String";
document.write(str);
</script>
```

25

String Constructor

- create instance of string by passing arguments in constructor
- Syntax: var mystring=new String("item1")
- Example

```
<script>
var str1 = new String("JavaScript String");
document.write(str1 + "<br>");
</script>
```

- The property length returns the number of characters in a string.

```
var n=str1.length
```

26

String Methods

Method	Description
charAt()	Returns the character at the specified index.
concat()	Combines the text of two strings and returns a new string.
replace()	Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.
slice()	Extracts a section of a string and returns a new string.
substr()	Returns the characters in a string beginning at the specified location through the specified number of characters.
toLowerCase()	Returns the calling string value converted to lower case.
toString()	Returns a string representing the specified object.
toUpperCase()	Returns the calling string value converted to uppercase.

27

Example

```
<script>
var s1="JavaScript";
 document.write("<h1>string length:</h1>"+s1.length);
document.write("<br>"+s1.charAt(2));
var s2="example";
var s3=s1.concat(s2);
document.write("<h1>after concat:</h1>"+s3);
var s4=s3.toUpperCase();
document.write("<h1>after Upper case:</h1>"+s4);
var s5=s4.toLowerCase();
document.write("<h1>after lower case:</h1>"+s5);
var s6=s5.substring(2,5);
 document.write("<h1>after substring:</h1>"+s6);
var s7=s5.substr(2,5);
 document.write("<h1>after substr:</h1>"+s7);
</script>
```

28

Note

```
Var a = “This works ‘fine’”;
var x = “This is a “JavaScript” example” ;

        var x = “This is a \“JavaScript\” example”;

var x = “JavaScript";
var y = new String("JavaScript");

                x==y?
                x===y?

            document.write(typeof(y));
```

29

Document Object Model

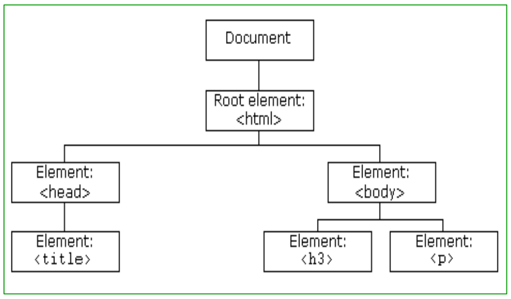
- an interface that allows developers to manipulate the content, structure and style of a website.
- A Document object represents the HTML document that is displayed in the browser window.
- programmers can create and build documents, navigate their structure, and add, modify, or delete elements and content
- HTML DOM model is constructed as a tree of Objects.
- When a web page is loaded, the web browser creates a Document Object Model of the web page.

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

30

Document Object Model

```
<html>

<head>
  <title>Page Title</title>
</head>

<body>
  <h3>Heading</h3>
  <p>This is a paragraph.</p>
</body>

</html>
```



- the places of the elements are referred to as **nodes**.
- also the attributes of elements and text get their own node

31

Document Object Model

- The model is built in a tree structure of objects and defines:
  - The properties of all HTML elements
  - The methods to access all HTML elements
  - The methods to create and delete
  - The events for all HTML elements
- The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

32

Document Object Methods

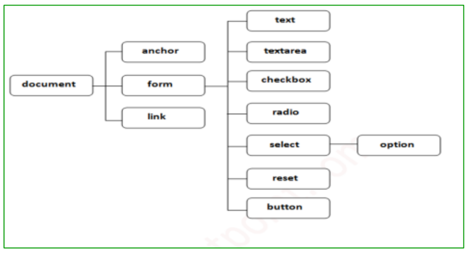
- Some important methods of document object

Method	Description
write("string")	prints the given string on the HTML document.
writeln("string")	prints the given string on the document with newline character at the end.
getElementById()	returns the element having the given 'id' as argument
getElementsByName()	returns all the elements having the given 'name'
getElementsByTagName()	returns all the elements having the given tag name.

33

Access Elements

- The following ways are common to access the HTML elements
  - Get element through node hierarchy
  - Get element by ID
  - Get element by tag name



- Access by Node Hierarchy
  - var name=document.parent.child;

```
<form name="form1">
Enter Name:<input type="text" name="name"/>
</form>
```

```
var name=document.form1.name.value;
```

34

Access Elements

- Get element by ID
- The *getElementById()* method is used to get a single element by its id

```
<form name="form1">
Enter Name:<input type="text" id="name"/>
</form>
```

```
var name=document.getElementById('name').value;
```

35

Access Elements

- Get element by Tag Name
- elements can be identified by tag name using the getElementsByTagName() method

```
<p>First Paragraph</p>
<p>Second Paragraph</p>
```

```
document.getElementsByTagName("p")[0].innerHTML;
```

36

Changing HTML Elements

- The innerHTML property can be used to change the content of an HTML element.

  document.getElementById(“id”).innerHTML = “Hello World!”;

- Changing a value of an attribute

  document.getElementsByTag(“img”).src = “test.jpg”;

- Changing the style

  document.getElementById(‘id’).style.color=“red”;

eval()

37

HTML Event Handlers

- The HTML DOM also allows Javascript to react to HTML events.
- JavaScript's interaction with HTML is handled through events
- Examples of HTML events:
  - When a user clicks the mouse
  - When a web page has loaded
  - When the mouse moves over an element
  - When an input field is changed
  - When an HTML form is submitted
- The term event handler is any function or object registered to be notified of events.

38

HTML Event Handlers

Event	Description
onclick	The user clicks an HTML element
onsubmit	occurs when form is submitted.
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page
onchange	An HTML element has been changed

HTML Event Handlers

```
<html>
<body>
<script>
function displayDate() {
   document.write(Date());
}
</script>
<p>Click the button to display the date.</p>
<input type="button" onclick="displayDate()"
value=click>
</body>
</html>
```

Form Validation

- Vital to validate the form submitted by the user for appropriate values
- JS facilitates to validate the form on the client side so processing will be fast than server side validation
- So, web developers prefer JavaScript form validation
- Through JavaScript, we can validate name, password, email, date, mobile number etc fields.
- Form validation is the process of making sure that data supplied by the user using a form, meets the conditions set for collecting data from the user

Form Validation

User Name Validation

Rule:

1. Name not empty.

```
<html><head>
<script>
function validation()
{
var a = document.form.name.value;
if(a=="")
{
alert("Please Enter  Name");
return false;
}}
</script></head>
<body>
<form    name="form"         onsubmit="
validation()">
 Your Name:<input type="text" name="name"">
<input type="submit" name="sub" value="Submit">
</form></body></html>
```

Form Validation

User Name Validation

Rules:

Name not empty

Only Characters.

Must be 5 to 15 Characters.

```
<html><head>
<script type="text/javascript">
function validation()
{
var a = document.form.name.value;
if(a=="")
{
alert("Please Enter Your Name");
return false;
}
if(!isNaN(a))
{
alert("Please Enter Only Characters");
return false;
}
if ((a.length < 5) || (a.length > 15))
{
alert("Your Character must be 5 to 15 Character");
return false;}}
</script></head>
<body>
<form name="form" method="post" onsubmit="return
validation()">
 Your Name:<input type="text" name="name"">
<input type="submit" name="sub" value="Submit">
</form></body></html>
```

Form Validation

Password Validation

Rules:

Password not empty

At least 8 characters.

```
<html><head><script>
function validation()
{
var a = document.form.pass.value;
if(a=="")
{
alert("Please Enter password");
return false;
}
if (a.length <8)
{
alert("Password atleast have 8 characters");
return false;}}
</script</head>
<body>
<form name="form"   onsubmit="
validation()">
 Your Name:<input type="password"
name="pass"">
<input type="submit" name="sub"
value="Submit">
</form></body></html>
```

Form Validation

Retype Password Validation

```
<html><head><script>
function matchpass(){
var firstpassword=document.f1.password.value;
var secondpassword=document.f1.password2.value;
if(firstpassword==secondpassword){
return true; }
else{
alert("password must be same!");
return false;}}
</script></head>
<body>
<form name="f1"  onsubmit="matchpass()">
Password:<input type="password" name="password" /><br/>
Re-enter Password:<input type="password" name="password2"/><br/>
<input type="submit">
</form></body></html>
```

Form Validation

Phone Number Validation

Numbers only

10 digit number only

```
<html><head><script>
function validate(){
var num=document.myform.num.value;
if (isNaN(num)){
  document.getElementById("numloc").innerHTML="Enter Numeric value
      only";
  return false; }
else if(num.length==10) {
 alert("Valid");   }
else{
document.getElementById("numloc").innerHTML="Enter 10 digit phno only";
  return false;  }}
</script</head>
<body>
<form name="myform"      onsubmit="return validate()" >
Number: <input type="text" name="num"><span id="numloc"></span><br/>
<input type="submit" value="submit">
</form></body></html>
```

End

47

Add/Remove HTML Elements

48

Add/Remove HTML Elements

- createElement() //parameter tag will be created
- createTextNode //content text will be created
- x.appendChild(y) // y will be added as a child to x
- z.insertBefore(x,y) //both x and y are child of z. insert x before the occurrence of tag y under z.
- x.remove() //delete the tag x
- x.removeChild(y) //delete the tag y which is child of x

Add/Remove HTML Elements

```
<form id="form1">
<label id="l1">This is a Label 1</label>
<label id="l2">This is Label 2</label>
</form>

<script>

var newlabel = document.createElement("label");                    //creating a Tag
var labeltext = document.createTextNode("This is new.");           //creating a Text value

newlabel.appendChild(labeltext);                                   //assigning the text value to the tag created

var formelement = document.getElementById("form1");                //parent tag to insert new one
formelement.appendChild(newlabel);                                //inserting the new tag as child of parent tag

//var label1element = document.getElementById("l1");
//formelement.insertBefore(newlabel,label1element);       // To insert new tag before any specific child tag of parent

var remelement = document.getElementById("l1");
remelement.remove()                                                //remove element by refering itself

//var parent = document.getElementById("form1");
//var child = document.getElementById("l1");
//parent.removeChild(child);                                       //remove element by refering parent of it

</script>
```