Javascript

Shristi Technology Labs

Contents

Introduction to javascript

JavaScript is

- a lightweight, interpreted programming language
- Designed for creating network-centric applications
- Complementary to and integrated with Java
- Complementary to and integrated with HTML
- Open and cross-platform
- an Object Oriented Programming (OOP) language.

Advantages

- **Less server interaction:**
  - user input is validated before sending the page off to the server.
  - saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors:**
  - don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity:**
  - can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces:**
  - Can include items as drag-and-drop components and sliders using javascript to give a Rich Interface to the website.

<script> tag

- JavaScript statements are placed within the **<script>... </script>** in a html page.
- Write javascript in an external file and include the file in the html page
- **<script>** tag can be placed
  - Between head tags for event handling
  - Between body tags when the script must run as the page loads
- **<script>** tag
  - alerts the browser to interpret all the text between these tags as a script.

```
<script src="courses.js"></script>
<script>
    // javascript code
</script>
```

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Insert title here</title>
    <script>
        document.write("welcome to JS");
    </script>

</head>
<body>

</body>
</html>
```

Features

- Is case-sensitive.
- Whitespaces, tabs, spaces are ignored
- Semicolons are ignored
- // is single line comment
- /* */ is multi line comment
- Untyped Language

Data Types

Three primitive data types:

- Numbers **e.g.,** 123, 120.50
- Strings of text **e.g.,** “Hello world"
- Boolean **e.g.,** true or false

Two trivial data types

- *null* and *undefined*, each of which defines only a single value.

Composite data type

- *object*.

- does not make distinction between integer values & floating-point values.
- All numbers are represented as floating-point values.

Variables

- Variables are containers of data.
- Variables are declared with the **var** keyword
- Can’t start with numbers or keywords
- Can start with letters or _
- Is case sensitive.
- can do variable initialization at the time of creation or later

```
<script>
    var num =100;
    var message = 'Hello World';
</script>
```

Untyped Language

- variable can hold a value of any data type.
- value type of a variable can change during the execution of a program and JavaScript takes care of it automatically

```
<script>
    var num =100;
    // few lines of code
    num = 'Hundred';
</script>
```

Operators

Supports following type of operators

- Arithmetic Operators **(+,-,*)**
- Comparison Operators**(==,!=,>,<)**
- Logical (or Relational) Operators**(&&,||)**
- Assignment Operators**(+=,-=)**
- TernaryOperators**(?)**
- **===** returns true if value and data type are equal
- typeof operator

typeof operator

- returns the type of a variable, object, function or expression
- is placed before its single operand, which can be of any type.
- value is a string indicating the data type of the operand
- evaluates to "number", "string", "boolean“ ,”object”

```
var  name= "Ramana";
result = (typeof name == "string" ? "String" : "Numeric");
var  num= 100;
result1 = (typeof num == "number" ? "is a number" : "String");
```

Dialog box

- **alert()** – to give a message to user
- **prompt()** – to get user input
- **confirm()** – to get a confirmation from user for propagating to the next page

Example – using alert

```
<!DOCTYPE html>
<html>
<head>
 <title>Using Alert</title>

 <script>
  var  message = "Have a good day";
  alert(message)
 </script>
</head>
<body>

</body>
</html>
```

Example – using prompt

```
<html>
<head>
    <title>Using Prompt</title>
</head>
<body>
    <h1>This is in the HTML Page </h1>
    <script>
        name=prompt("Enter your name ","Ram ");
        alert("Welcome" +name)
        document.write("<h2>Hello "+name+" </h2>")
    </script>

</body>
</html>
```

Control Flow Statements

- **Selection Statements**
  - if – else
  - switch
- **Iteration Statements**
  - for
  - while
  - do – while
  - for-in
- **Jump Statements**
  - break
  - continue

Example – If else

```
<script >
    var book = prompt("Enter the book name", " ");
    alert(book + " " +isNaN(book))
    if (!isNaN(book)) {
        document.write("<b>enter  a string</b>");
    } else if (book.toUpperCase() == "JAVA") {
        document.write(book + " book is selected ");
    } else if (book.toUpperCase() == "JSP") {
        document.write("Book Selected " + book);
    } else if (book.toUpperCase() == "SPRING") {
        document.write(book + " book selected");
    } else {
        document.write("book not available");
    }
</script>
```

Example – switch

```
<script>
    var val = prompt("ENTER THE DAY "," ")
    val2 = val.toUpperCase();
    switch (val2) {
        case 'MONDAY':
        case 'TUESDAY':
        alert("This is a working day");
        break;
        case 'SATURDAY':
        case 'SUNDAY':
        alert("Happy weekend");
        break;
        default:
        alert('please enter valid day')
    }
</script>
```

Example – for, for-in

```
<script>
 for(var x=0;x<10;x++) {
  document.write(x);
 }
 // iterating an array
 var arr=[16,9,33,264,45,86];
 for(var i=0;i<arr.length;i++){
  document.write(arr[i]+" ");
 }
</script>
```

```
 <script>
      // iterating an array
      var arr=[16,9,33,264,45,86];
      for(var i in arr){
          document.write(arr[i]+" ");
      }
 </script>
```

Example – while, do-while

```
<script>
  //while
  var x=0;
  while(x<10){
    document.write(x +"<br/>");
    x++;
  }
  //do-while
  do{
    document.write(x+"<br/>");
    x--;
  }while(x>=0);

</script>
```

Example – break

```
<script>
    //if username exists in array, welcome the user(login)
    var names=["Priya","Ram","Raj","Tom","Sam","Gigi"];
    var username = prompt("Enter your name","");
    var flag = false;
    for ( var x = 0; x < names.length; x++) {
        if (username == names[x]) {
            flag = true;
            break;// comes out of the loop
        }
    }
    if(flag){
        alert("welcome "+username);
    }else{
        alert("Wrong username ");
    }
</script>
```

Example – continue

```
<script>
  //if username exists in array ask for a new name(register)
  var names=["Priya","Ram","Raj","Tom","Sam","Gigi"];
  var username = prompt("Enter your name","");
  var flag = true;
  for ( var x = 0; x < names.length; x++) {
    if (username != names[x]) {
      continue;// skip and continue next iteration
    }else{
      flag =false;
    }
    alert("checking...")
  }
  if(flag){
    alert("You are registered "+username);
  }else{
    alert("Name already exists ");
  }
</script>
```

Functions

- Function is a reusable code which can be called anywhere.
- No need to declare return types or argument types
- While calling the function, can pass any number of arguments regardless of the arguments in the function
- Functions are first-class data types and can be global
- Can create anonymous functions (closures)

e.g.,

// function declaration

```
function greet(...) {
 //js statements
}
var greet = function(...) {...}
```

// function expression

Example – functions

```
<!DOCTYPE html >
<html>
<head>
 <title>Insert title here</title>
 <script>
  var name=prompt("enter you name ","");
  sayHello();
  //function declaration
  function sayHello(){
   alert("welcome "+name);
  }
  //function expression
  var greet = function(){
   alert("hello "+name);
  }
  greet();
 </script>
</head>
<body></body>
</html>
```

Variables - type and scope

Global Variables

- Variables declared without var keyword has global scope
- It can be accessed everywhere within the js file.

Local variables

- Variables declared with var keyword is visible only within a function.
- Function parameters are always local to that function.
- They take precedence over global variables if the name is same.

Example – variable scope

```
<script>
  var lnum =10;// local to this file
  gnum=20;//global
  function declareVar() {
    myglobal =200;  // no var so global
    var mylocal=30;  // local
  }

  var message = "outside function";
  function printVar() {
    var message = "inside function";
    document.write("with var - local "+lnum+"<br>");
    document.write("without var - global "+gnum+"<br>");
    document.write("without var inside function - global "+myglobal+"<br>");
    //document.write("with var inside function - local "+mylocal+"<br>");
    document.write(message+"<br>");
  }
  document.write("Print First-  "+message+"<br>");
  declareVar();
  printVar();
</script>
```

Event Handling

- Events occur when the user or browser manipulates the page
- Events can be handled
  - on interaction with the user
  - on loading of page
  - on click of button
  - on submitting a form
  - on key up, down
  - on mouse movements

Example – Handling onload event

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Using onLoad</title>
 <script>
  var name=prompt("your name please ","");
 </script>
</head>
<body  onLoad="alert('welcome  '+name)" >
 <h3><i>Have a great day</i></h3></body>
</body>
</html>
```

Example – Handling onclick event

```
<!DOCTYPE html >
<html>
<head>
 <title>Getting values</title>
 <script>
  function sayHello(){
   name = prompt("Enter name ","");
   alert("Have a great day "+name);
  }
 </script>
</head>
<body>
 <input type="button" onclick="sayHello()"  value="Show ">
</body>
</html>
```

Example – using getElementById

```
<!DOCTYPE html >
<html>
<head>
 <title>Getting values</title>
 <script>
  function sayHello(){
   name = document.getElementById("username").value;
   alert("Have a great day "+name);
  }
 </script>
</head>
<body>
 Enter Name<input type="text" id = "username">
 <input type="button" onclick="sayHello()"  value="Show ">
</body>
</html>
```

Example – using innerHTML

```
<!DOCTYPE html >
<html>
<head>
 <title>Getting values</title>
 <script>
  function sayHello(){
   name = document.getElementById("username").value;
   alert("Have a great day "+name);
   document.getElementById("mydiv").innerHTML = "Have a great day " +name;
  }
 </script>
</head>
<body>
 Enter Name<input type="text" id = "username">
 <input type="button" onclick="sayHello()"  value="Show ">
 <div id="mydiv"></div>
</body>
</html>
```

Example – Handling onkeyup event

```
<!Doctype html >
<html>
<head>
 <title>Using OnKeyup</title>
</head>
<body>
 Enter comments
 <textarea rows="5" cols="10" id="poppy" onkeyup="printComments()"></textarea>
 <div id="mydiv" style="background-color:yellow;color:green"></div>
 <script >
  function printComments() {
   var message=document.getElementById("poppy").value;
   document.getElementById("mydiv").innerHTML= message;
  }
 </script>
</body>
</html>
```

Example – Using confirm

```
<!DOCTYPE html >
<html>
<head>
 <script>
  function call(){
   if(confirm("You have unsaved messages,Do you want to leave ?")){
    alert("Bye Bye");
    document.myform.submit();
   }
   else{
    alert("Thanks for staying back");
   }
  }
 </script>
</head>
<body>
 <form action="demo1.html" name="myform">
  Enter Comments<textarea id = "comments"></textarea>
  <input type="button" value="Click " onclick="call()">
 </form>
</body>
</html>
```

Objects

- ***new*** operator is used to create an instance of a class.
- **Inbuilt Objects**
  - Boolean
  - Number
  - String
  - Array
  - Math
  - Date
- **User-defined Objects**
  - Can be created as object literals or using function constructor

Boolean

- Is to create a boolean object
- Represents either true or false

Syntax

```
var val = new Boolean(value);
```

- If *value* parameter is **omitted, 0, -0, null, false, NaN, undefined, or the empty string ("")** the object has an initial value of false.

```
var nu = new Boolean(undefined);
document.write(nu+"<br>");
nu = new Boolean(null);
document.write(nu+"<br>");
```

Number

- represents numerical date, either integers or floating-point numbers.
- Automatically converts number literals to instances of Number class.

Syntax:

```
var value = new Number(number);
```

- If the argument cannot be converted into a number, it returns ***NaN (Not-a-Number).***

```
var num = new Number(10.28);
document.write(num.toLocaleString()+"<br>");
document.write((num+100)+"<br>");
document.write(num.toString()+100+"<br>");
document.write(num.toFixed(3)+"<br>");
```

String

- Works with a series of characters
- Automatically converts string primitives to String objects

Syntax:

```
var val = new String(string)
```

- The *string* parameter is series of characters that has been properly encoded

Example

```
var name = new String("This is  javascript demo");
document.write(name.length+"<br>");
document.write(name.charAt(1)+"<br>");
document.write(name.concat("bye")+"<br>");
document.write(name.toUpperCase()+"<br>");
document.write(name.slice(3,9)+"<br>");
document.write("<b>"+name+"</b>"+"<br>")
document.write(name.bold().italics()+"<br>");
document.write("foo"=='foo');  //returns true
document.write("<br>");
 s="hello"; s1='hello';
 document.write("abc"+3+5+"<br>");
document.write(3+7+"abc"+5+"<br>");
document.write(parseInt("123abc")+"<br>");
document.write(parseInt("abc123")+"<br>"); // returns nan
document.write(s == new String("hello"));    // true
document.write("<br>")
document.write(s === new String("hello"));//false
```

Array

- are used to store multiple values in a single variable
- Can be created as array literal or using new keyword

- Creating an array using new keyword may give complicated results

```
var arr = new Array(20,30);
document.write(arr[0]);
arr = new Array(20);
document.write(arr[0]);
```

```
<script>
 var nums=[16,9,33,264,45,86];
 for(var i=0;i<nums.length;i++){
  document.write(nums[i]+" ");
 }
 document.write("<br>");
 for(var a in nums){
  document.write(nums[a]+"<br>");
 }
 var names = new Array(4);
 names[0]="Ram";
 names[1]="Tom";
 names[2]="Shyam";
 names[3]="Manu";
 names[9]="Oggy";
 for ( var i = 0; i < names.length; i++) {
  document.write("<br>"+names[i]);
 }
</script>
```

Example

Properties

- length

Methods

- push()
- reverse()
- sort()

```
nums=[16,9,33,264,45,86];
document.write("<br>Adding element to the array ");
nums.push(98);
document.write(nums);
document.write("<br>reversing array ");
nums.reverse();
document.write(nums);
document.write("<br>sorting array ");
nums.sort();
document.write(nums+"<br>");
nums.sort(sortArray);
function sortArray(a,b){
    return a-b;
 }
document.write(nums+"<br>");
```

Date

- The Date object works with dates (years, months, days, hours, minutes, seconds, and milliseconds
- Date objects are created with the **new Date()** constructor.

```
<script>
    //gives the current date and time
    var date = new Date()
    document.write(date+"<br>");
    //passing date string
    date = new Date("February 17, 2003 6:15:02");
    document.write(date+"<br>");
    //year,month(starts at 0),day,hour,min,sec
    date = new Date(2007,8,06,12,40,00);
    document.write(date+"<br>");
</script>
```

Date Format

Type	Example
ISO Date	"2015-03-25" (The International Standard)
Short Date	"03/25/2015"
Long Date	"Mar 25 2015" or "25 Mar 2015"
Full Date	"Wednesday March 25 2015"

```
date = new Date("2016-02-24"); //iso complete date
document.write("ISO Complete Date:   "+date+"<br>");
date = new Date("2016-02"); //iso date year and month
document.write("ISO Date:   "+date+"<br>");
date = new Date("02/24/2016")//short date
document.write("Short Date:   "+date+"<br>");
date = new Date("2016-02-24T02:30:00") //date and time separated by T
document.write("Date & Time:   "+date+"<br>");
date = new Date("Feb 24 2016")//long date
document.write("Long Date:   "+date+"<br>");
```

Math

- Math object allows you to perform mathematical tasks on numbers
- Methods and properties are static.

e.g.,

Functions

– Math.abs, Math.ceil, Math.sqrt,

Constants

– Math.PI

User defined Objects

- Javascript is a prototype based language
- User defined objects can be created using function constructor or as object literals
- New properties can be attached to the object after the object is created

User defined Objects – Object Literals

```
<script>
 var student = {
  name : 'Raj',
  age : 20,
  hobbies:['sports','music'],
  address:{
   city:'Bangalore',
   state:'KAR'
  }
 }
 document.write(student.name+"<br>");
 student.mobile = 9876543210;
 for(var i in student){
  var st = student[i];
  if(typeof(st)==Array || st instanceof Object){
   for(var k in st){
    document.write(k+":  "+st[k]+"<br>");
   }
  }else{
   document.write(i+":  "+ st+"<br>");
  }
 }
</script>
```

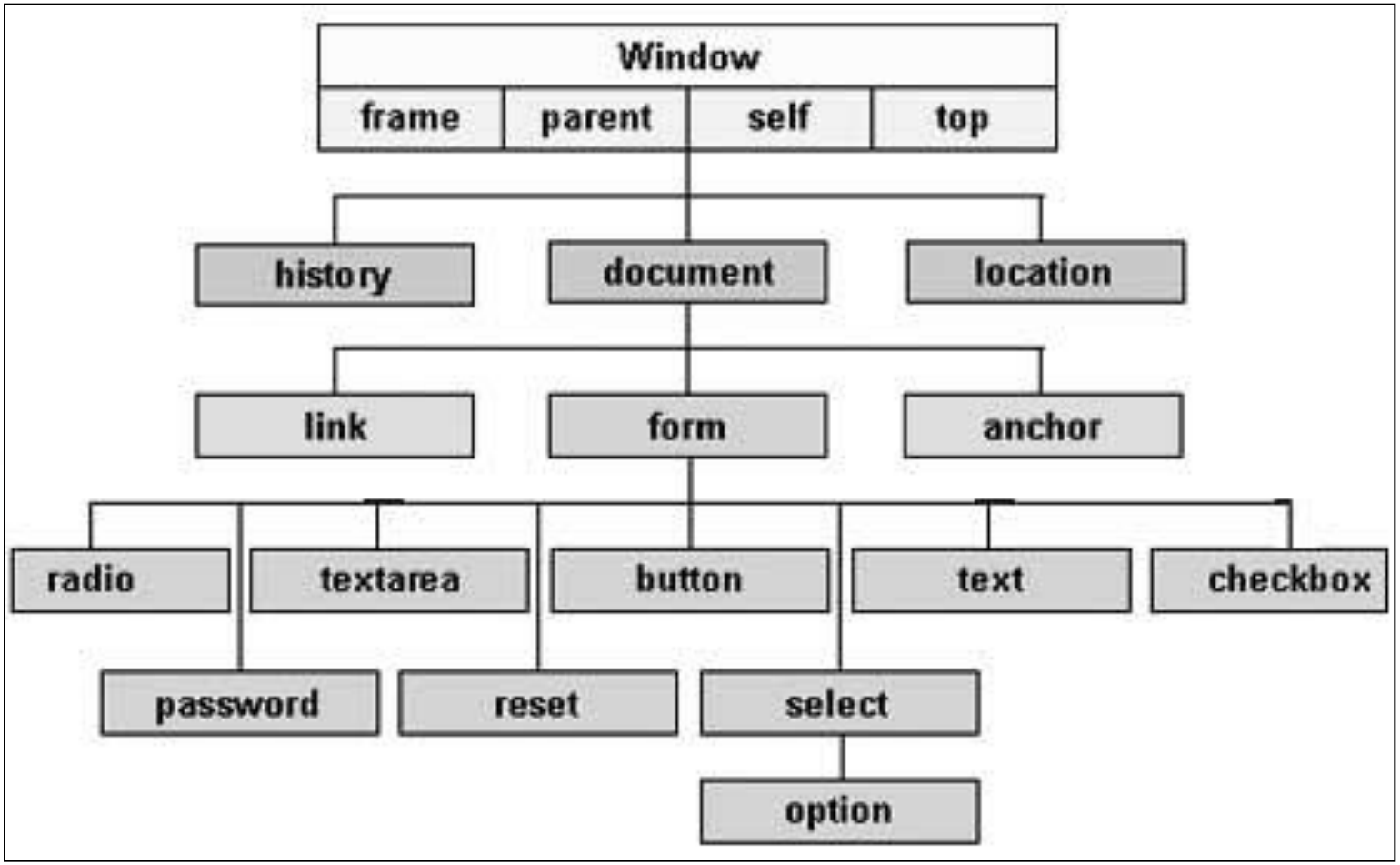
User defined Objects – Function Constructor

```
<script >
 var Book = function (title,author){
  this.title = title;
  this.author   = author;
  this.getDetails = function () {
   document.write("Title "+this.title +"<br>");
  }
 }
 // attaching a property to class prototype
 Book.prototype.price=2000;
 Book.prototype.showDiscount=function(amt){
   document.write("Discounted price "+amt);
  };
 //creating an instance of book
 var book = new Book("Spring","Rod");
 document.write(book.title+"<br>");
 document.write(book.author+"<br>");
 document.write(book.price+"<br>");
 book.getDetails();
 book.showDiscount(900);
</script>
```

Example – Array of objects

```
<script>
var studentList = [ {
  name : 'Ram',hobbies:['sports','music'],address:{ city:'Bangalore',state:'Karnataka'}
  }, {
  name : 'Tom', age : 20
 }, {
  name : 'Raj', age : 19
 }]
for ( var i in studentList) {
 var student = studentList[i];
 for(var j in student){
  var st = student[j];
   if(typeof(st)==Array || st instanceof Object){
   for(var k in st){
    document.write(st[k]+"<br>");
   }
  }else{
   document.write(st+"<br>");
  }
 }
}
</script>
```

DOM


Window
frame
parent
self
top
history
document
location
link
form
anchor
radio
textarea
button
text
checkbox
password
reset
select
option

window Object

window is the highest level JavaScript object which corresponds to the web browser window.

Properties

```
window.status
window.closed
```

Methods

```
window.alert("mess")
window.back()
window.close()
window.blur()
```

navigator Object

window.navigator object contains information about the browser

appCodeName	Returns the code name of the browser
appName	Returns the name of the browser
appVersion	Returns the version of the browser
language	Returns the language of the browser
platform	Returns for which platform the browser is compiled

Properties

```
navigator.appName
Navigator.appVersion
```

navigator.geolocation

- The geolocation property returns a Geolocation object that is used to locate the user's position.
- The position is not available unless the user approves it, as it can compromise user’s privacy.
- To get the user’s location,call ***getCurrentPosition()*** method
- The ***getCurrentPosition()*** method returns an object on success. The latitude, longitude and accuracy properties are always returned.

location Object

window.location object can be used to get the current page address (URL) and to redirect the browser to a new page

location object helps to get the pathname, port, protocol, querystring of a url

Properties

`location.href`

`location.hostname`

Methods

`location.reload()`

`location.replace()`

`location.assign()`

history Object

window.history object contains the URLs visited by the user (within a browser window)

This is an array of URL strings which reflect the entries in the History object.

Methods

```
history.back()
history.forward()
history.go(num)
```

Example

```
<h3><a href="hispage1.html">click for page1</a></h3><br>
<h3><a href="hispage2.html">click for page2</a></h3><br>
```

hisdemo.html

```
<input type="button" onclick="history.back()" value="go back" />
<input type="button" onclick="history.forward()" value="go forward" />
```

hispage1.html

```
<input type="button" onclick="history.back()" value="Back" />
<input type="button" onclick="history.go(-2)" value="Older history" />
```

hispage2.html

Cookie

- A cookie is a small text file that's stored in your browser.
- It contains data like
  - A name-value pair containing the actual data
  - An expiry date after which it is no longer valid
  - The domain and path of the server it should be sent to

```
// to create a cookie
document.cookie = "username=Kathy"
```

```
//to retrieve the cookie
document.write(document.cookie);
```

Summary

- Introduction
- Datatypes and variables
- Operators and Control Flow Statements
- Functions
- Event Handling
- Validation of forms using javascript
- InBuilt Objects - String, Number,Array, Math, Date
- User-defined Objects
- DOM – Window, Navigator, Location, History
- Cookie