FirstBitSolutions.com
...Learn IT Bit by Bit...

JavaScript

Topic

- What is JavaScript?,Why to Learn JavaScript
- Application of JavaScript, Features of JavaScript
- Where to define JavaScript
- JavaScript Syntax, Variables, Functions, DataTypes, Operators, Comments
- Conditional Statement, Switch case
- What is Loop?, Types of Loop
- JavaScript String, JavaScript Object
- JavaScript Array, Date Object
- Window Object, Form Validation

What is JavaScript?

FirstBitSolutions.com
...Learn IT Bit by Bit...

- JavaScript is designed for beginners and professionals both. JavaScript is used to create client-side dynamic pages.
- JavaScript is an object-based scripting language which is lightweight and cross-platform.
- JavaScript is not a compiled language, but it is a translated language. The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.
- JavaScript is a high level, interpreted, programming language used to make web pages more interactive.

Why Study JavaScript?

FirstBitSolutions.com
...Learn IT Bit by Bit...

- JavaScript is one of the 3 languages all web developers must learn:
  - HTML to define the content of web pages
  - CSS to specify the layout of web pages
  - JavaScript to program the behavior of web pages

Application of JavaScript

- JavaScript is used to create interactive websites. It is mainly used for:
- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

Features of JavaScript

- All popular web browsers support JavaScript as they provide built-in execution environments.
- It is a case-sensitive language.
- JavaScript is supportable in several operating systems including, Windows, macOS, etc.
- It provides good control to the users over the web browsers.
- JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
- It doesn’t have to be compiled like Java and C which require a compiler.

Where to define JavaScript

FirstBitSolutions.com
...Learn IT Bit by Bit...

- Scripts can be placed in the <body>-----</body> tag.
- Scripts can be placed in the <head>-----</head> tag.
- Scripts can be placed with External Link.

JavaScript Syntax

JavaScript syntax is the set of rules, how JavaScript programs are constructed.

```
var x, y, z;       // Declare Variables
x = 5; y = 6;      // Assign Values
z = x + y;         // Compute Values
```

JavaScript Values

- The JavaScript syntax defines two types of values:
  - Fixed values
  - Variable values
- Fixed values are called Literals.
- Variable values are called Variables.

JavaScript Literals

FirstBitSolutions.com
...Learn IT Bit by Bit...

- The two most important syntax rules for fixed values are

1. Numbers are written with or without decimals.

10.50
1001

2. Strings are text, written within double or single quotes

“Amit”
“Suresh”

Variables

- Variable is a name given to a memory location which acts as a container for storing data temporarily. They are nothing but reserved memory locations to store values.
- To declare a variable in JavaScript use the ‘let’ ,’var’ and ‘const’ keyword.

```
let age;
age=22;

var name=“Suresh”;
const pi=3.14;
```

Difference between var, let and const

FirstBitSolutions.com
...Learn IT Bit by Bit...

- var declarations are globally scoped or function scoped while let and const are block scoped.
- var variables can be updated and re-declared within its scope; let variables can be updated but not re-declared; const variables can neither be updated nor re-declared.
- var variables are initialized with undefined, let and const variables are not initialized.
- While var and let can be declared without being initialized, const must be initialized during declaration.

Constants

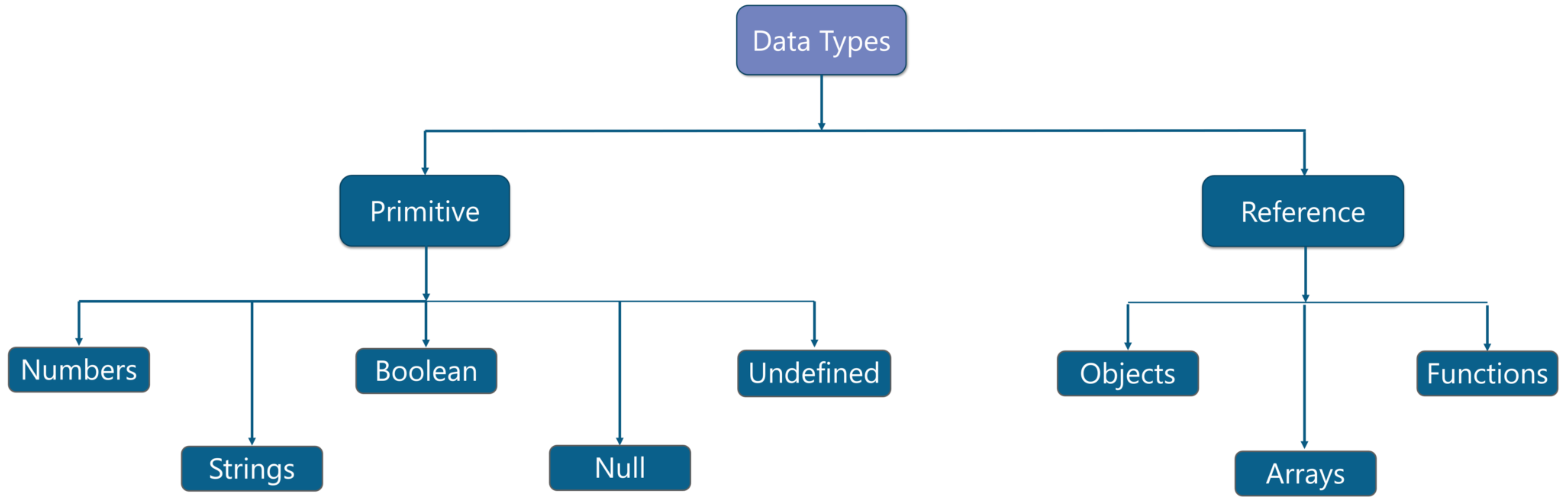
- Constants are fixed values that don’t change during execution time.
- To declare a constant in JavaScript use the ‘const’ keyword.

```
const pi=3.14;
```

- Data types

  You can assign different types of values to a variable such as a number or a string. In JavaScript, there are two categories of data types :

Data Types


Data Types
Primitive
Reference
Numbers
Strings
Boolean
Null
Undefined
Objects
Arrays
Functions

Functions

- A function is a block of organised, reusable code that is used to perform an action.
- To declare a function in JavaScript use the ‘function’ keyword.

```
 function product(a, b)
 {
return a*b;
 }


function show()
{

}
```

JavaScript Operator

JavaScript uses **arithmetic operators** (+ - * /) to **compute** values

(5 + 6) * 10

JavaScript uses an assignment operator **(=) to** assign **values to variables**

```
var x, y;
x = 5;
y = 6;
```

JavaScript Keywords

JavaScript **keywords** are used to identify actions to be performed.
The var keyword tells the browser to create variables

```
var x, y;
x = 5 + 6;
y = x * 10;
```

JavaScript Expressions

- An expression is a combination of values, variables, and operators, which computes to a value.
- The computation is called an evaluation.
- For example, 5 * 10 evaluates to 50

- **Expressions can also contain variable values**

  x * 10

  The values can be of various types, such as numbers and strings.

For example, “Amit” + " " + "Salunke", evaluates to "Amit Salunke"

JavaScript Comments

- Not all JavaScript statements are "executed".

- Code after double slashes // or between /* and */ is treated as a comment.

- Comments are ignored, and will not be executed
- var x = 5;   // I will be executed

  // var x = 6;   I will NOT be executed

JavaScript Identifiers

- Identifiers are names.
- In JavaScript, identifiers are used to name variables (and keywords, and functions, and labels).
- The rules for legal names are much the same in most programming languages.
- In JavaScript, the first character must be a letter, or an underscore (_), or a dollar sign ($).
- Subsequent characters may be letters, digits, underscores, or dollar signs.
- Numbers are not allowed as the first character.
  This way JavaScript can easily distinguish identifiers from numbers.

JavaScript is Case Sensitive

- All JavaScript identifiers are case sensitive.
- The variables lastName and lastname, are two different variables

```
var lastname, lastName;
lastName = "Amit";
lastname = "Salunke";
```

- JavaScript does not interpret VAR or Var as the keyword var.

JavaScript and Camel Case

- Programmers have used different ways of joining multiple words into one variable name:
- Hyphens:
- first-name, last-name, master-card, inter-city.
- Hyphens are not allowed in JavaScript. They are reserved for subtractions.
- Underscore:
- first_name, last_name, master_card, inter_city.
- Upper Camel Case (Pascal Case):
- FirstName, LastName, MasterCard, InterCity.

JavaScript and Camel Case

- Lower Camel Case:
- JavaScript programmers tend to use camel case that starts with a lowercase letter:
- firstName, lastName, masterCard, interCity.

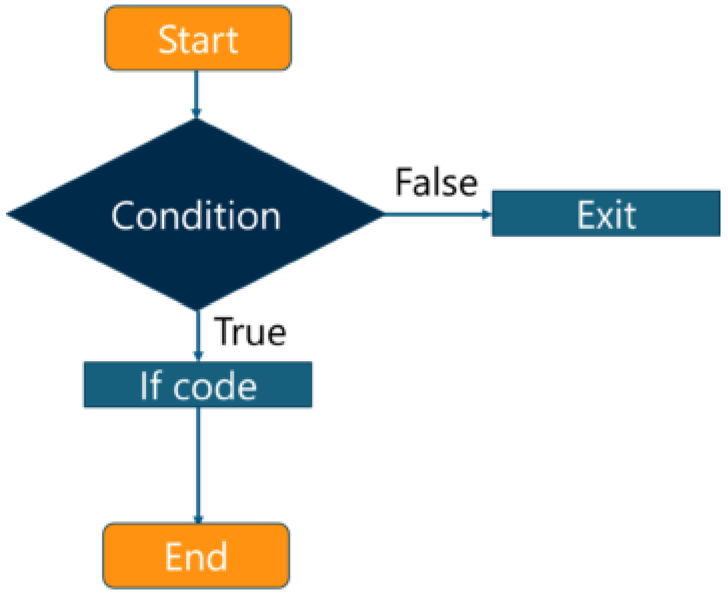
JavaScript Character Set

- JavaScript uses the **Unicode** character set.
- Unicode covers (almost) all the characters, punctuations, and symbols in the world.

Conditional statements – if

- Conditional statement is a set of rules performed if a certain condition is met. The ‘if’ statement is used to execute a block of code, only if the condition specified holds true.
- To declare an if statement in JavaScript use the ‘if’ keyword.

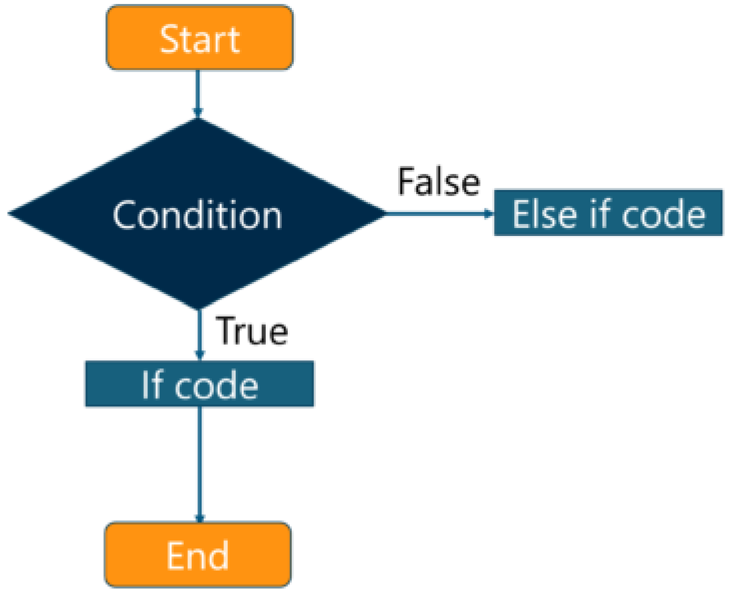
```
 if(condition)
{
      statement;
 }
```



Conditional statements- Else if

- Else if statement is used to execute a block of code if the condition is false.

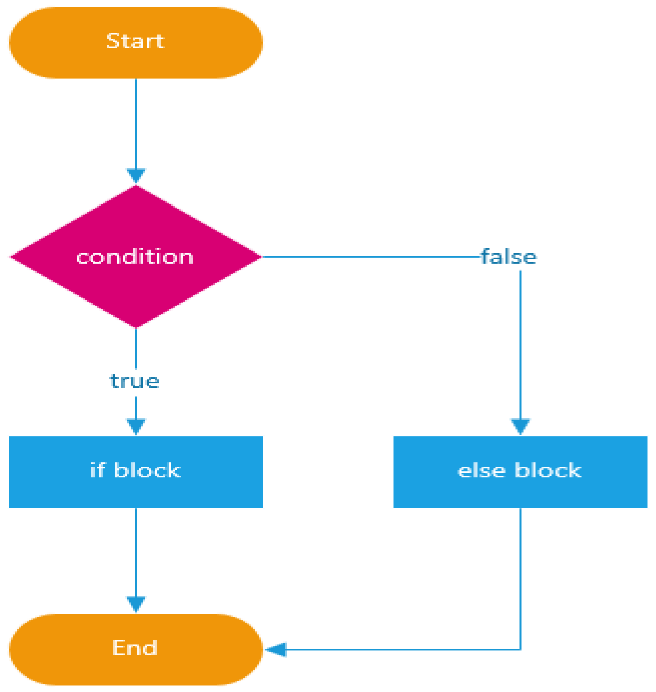
```
if(condition)
{
statement a;
}
else if(condition)
{
statement b;
}
```



Else statement

- Else statement is used to execute a block of code if the condition is false.

```
if(condition)
{
statement a;
}
else
{
statement b;
}
```

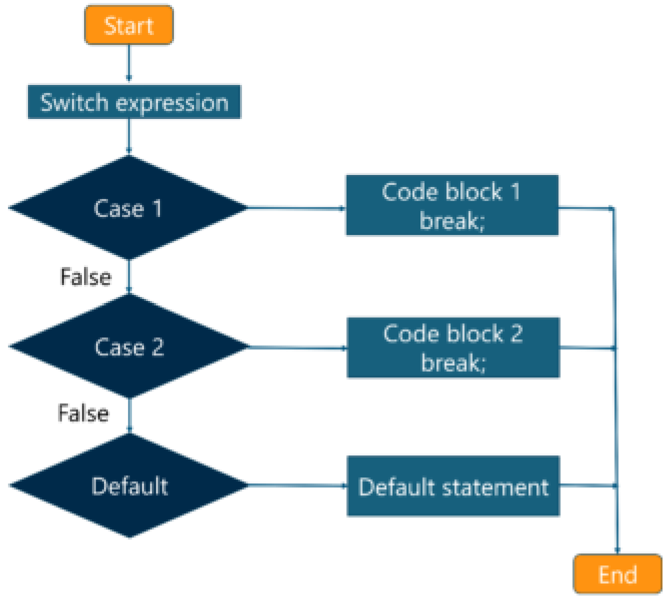


Switch case

- The switch statement is used to perform different actions based on different conditions.
- The value of the expression is then compared with the values of each case in the structure. If there is a match, the associated block of code is executed.

```
switch(expression)
{
case 1:
code block 1
break;
case 2:
code block 2
break;

default:
code block 3
break;
}
```



Loops

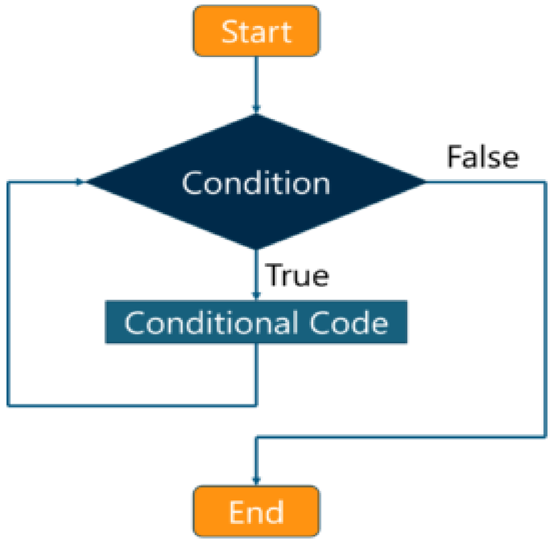
- Loop is used to execute the block of code several times according to the condition given becomes false. It means it executes the same code multiple times so it saves code.
- There are three categories of loops in JavaScript :

1. while loop
2. do while loop
3. for loop

While loop

While loop execute the code until condition is false.

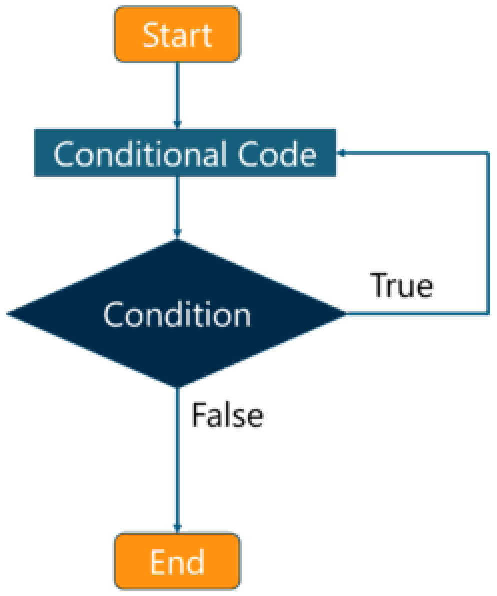
```
while(condition) {
loop code;
}
```



Do while loop

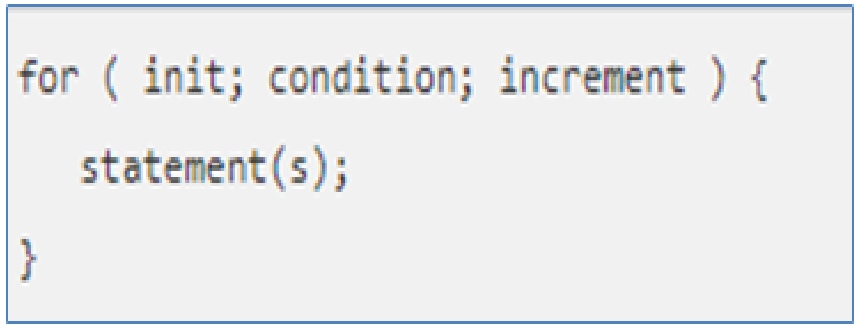
- It also executes the code until condition is false.
- In do…while loop at least once, code is executed whether condition is true or false but this is not the case with while. While loop is executed only when the condition is true.

```
do {
loop code;
} while(condition);
```

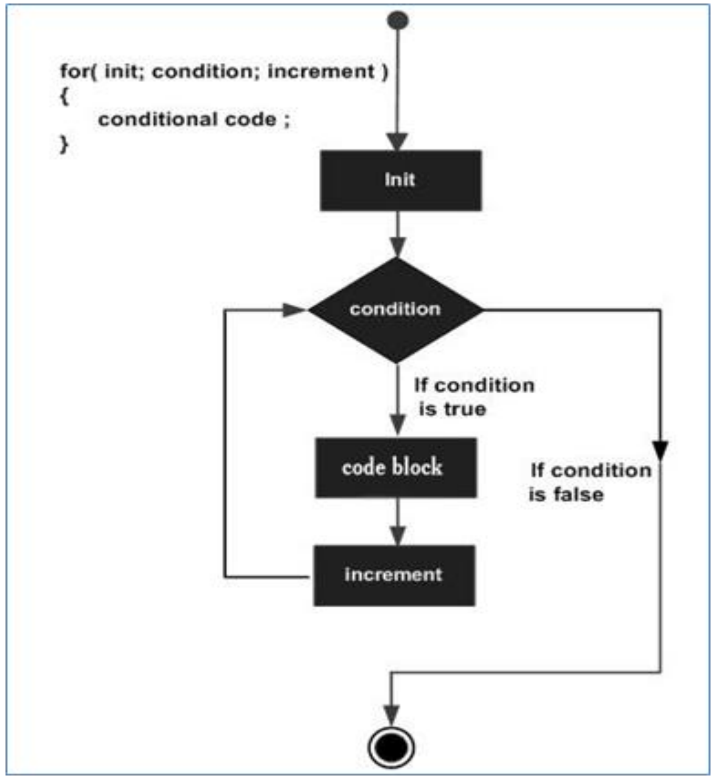


For loop

- The for loop repeatedly executes the loop code while a given condition is TRUE. It tests the condition before executing the loop body.
- In this three parameters are given that is
  - Initialization
  - Condition
  - Increment/Decrement



```
for ( init; condition; increment ) {
   statement(s);
}
```



```
for( init; condition; increment )
{
    conditional code ;
}
```

JavaScript String

- The JavaScript string is an object that represents a sequence of characters.
- There are 2 ways to create string in JavaScript

1. By string literal
2. By string object (using new keyword)

1) By string literal

The string literal is created using double quotes

```
<script>
var str="This is string literal";
document.write(str);
</script>
```

JavaScript String

2) By string object (using new keyword)

The syntax of creating string object using new keyword

```
<script>
var stringname=new String("hello javascript string");
document.write(stringname);
</script>
```

String Methods

Methods	**Description**
charAt()	It provides the char value present at the specified index.
charCodeAt()	It provides the Unicode value of a character present at the specified index.
concat()	It provides a combination of two or more strings.
indexOf()	It provides the position of a char value present in the given string.
lastIndexOf()	It provides the position of a char value present in the given string by
searching a	character from the last position.
search()	It searches a specified regular expression in a given string and returns its
position if	a match occurs.

String Methods

match()	It searches a specified regular expression in a given string and returns that regular expression if a match occurs.
replace()	It replaces a given string with the specified replacement.
substr()	It is used to fetch the part of the given string on the basis of the specified starting position and length.
substring()	It is used to fetch the part of the given string on the basis of the specified index.
slice()	It is used to fetch the part of the given string. It allows us to assign positive as well negative index.
toLowerCase()	It converts the given string into lowercase letter.
toLocaleLowerCase()	It converts the given string into lowercase letter on the basis of host?s current locale.
toUpperCase()	It converts the given string into uppercase letter.
toLocaleUpperCase()	It converts the given string into uppercase letter on the basis of host?s current locale.
toString()	It provides a string representing the particular object.
valueOf()	It provides the primitive value of string object.
split()	It splits a string into substring array, then returns that newly created array.
trim()	It trims the white space from the left and right side of the string.

JavaScript String Example

String **charAt()** method returns the character at the given index.

```
<script>
var str="javascript";
document.write(str.charAt(2));
</script>
```

String **concat(str)** method concatenates or joins two strings.

```
<script>
var s1="javascript ";
var s2="concat example";
var s3=s1.concat(s2);
document.write(s3);
</script>
```

String **indexOf(str)** method returns the index position of the given string.

```
<script>
var s1="javascript from indexof";
var n=s1.indexOf("from");
document.write(n);
</script>
```

JavaScript String Example

String **lastIndexOf(str)** method returns the last index position of the given string.

```
<script>
var s1="javascript from indexof";
var n=s1.lastIndexOf("java");
document.write(n);
</script>
```

String **toLowerCase()** method returns the given string in lowercase letters.

```
<script>
var s1="JavaScript toLowerCase Example";
var s2=s1.toLowerCase();
document.write(s2);
</script>
```

```
<script>
var s1="JavaScript toUpperCase Example";
var s2=s1.toUpperCase();
document.write(s2);
</script>
```

JavaScript String Example

String **slice(beginIndex, endIndex)** method returns the parts of string from given beginIndex to endIndex

```
<script>
var s1="This Is slice method example";
var s2=s1.slice(2,5);
document.write(s2);
</script>
```

String **trim()** method removes leading and trailing whitespaces from the string.

```
<script>
var s1="     javascript trim     ";
var s2=s1.trim();
document.write(s2);
</script>
```

```
<script>
var str="This is split method example";
document.write(str.split(" ")); //splits the given string.
</script>
```

JavaScript Objects

- A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.
- JavaScript is an object-based language. Everything is an object in JavaScript.
- JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

JavaScript Objects

- Creating Objects in JavaScript
- There are 3 ways to create objects.
  - By object literal
  - By creating instance of Object directly (using new keyword)
  - By using an object constructor (using new keyword)

1. JavaScript Object by object literal

```
<script>
emp={id:102,name:"Shyam Kumar",salary:40000}
document.write(emp.id+" "+emp.name+" "+emp.salary);

</script>
```

JavaScript Objects

FirstBitSolutions.com
...Learn IT Bit by Bit...

2. By creating instance of Object

```
<script>
var emp=new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
document.write(emp.id+" "+emp.name+" "+emp.salary);

</script>
```

3. By using an Object constructor

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Vimal Jaiswal",30000);
document.write(e.id+" "+e.name+" "+e.salary);
</script>
```

JavaScript Array

- JavaScript array is an object that represents a collection of similar type of elements.
- There are 3 ways to construct array in JavaScript
  1. By array literal
  2. By creating instance of Array directly (using new keyword)
  3. By using an Array constructor (using new keyword)

1. JavaScript array literal

```
<script>
var emp=["Suresh",“Amit","Ratan"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
```

JavaScript Array

2. JavaScript Array directly (new keyword)

```
<script>
var i;
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";

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

JavaScript Array

3. JavaScript Array

```
<script>
var emp=new Array("Jai","Vijay","Smith");
for (i=0;i<emp.length;i++)
{
document.write(emp[i] + "<br>");
}
</script>
```

JavaScript Date Object

- 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 JavaScript date object.
- You can use different Date constructors to create date object. It provides methods to get and set day, month, year, hour, minute and seconds.
- Constructor
- You can use 4 variant of Date constructor to create date object.

1. Date()
2. Date(milliseconds)
3. Date(dateString)
4. Date(year, month, day, hours, minutes, seconds, milliseconds)

JavaScript Date Methods

Methods	Description
getDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of local time.
getDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of local time.
getFullYears()	It returns the integer value that represents the year on the basis of local time.
getHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of local time.
getMilliseconds()	It returns the integer value between 0 and 999 that represents the milliseconds on the basis of local time.
getMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of local time.
getMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of local time.
getSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of local time.

Example of Date Objects

```
Current Date and Time: <span id="txt"></span>

<script>
var today=new Date();
document.getElementById('txt').innerHTML=toda
y;
</script>
```

```
Current Time: <span id="txt"></span>
<script>
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
document.getElementById('txt').innerHTML=h+":
"+m+":"+s;
</script>
```

```
<script>
var date=new Date();
var day=date.getDate();
var month=date.getMonth()+1;
var year=date.getFullYear();
document.write("<br>Date is: "+day+"/"+month+"/"+y
ear);
</script>
```

Browser Object Model

- The Browser Object Model (BOM) is used to interact with the browser.
- The default object of browser is window means you can call all the functions of window by specifying window or directly.

Window Object

- The window object represents a window in browser. An object of window is created automatically by the browser.
- Window is the object of browser, it is not the object of JavaScript. The JavaScript objects are string, array, date etc.

Methods of window object

Method	Description
alert()	displays the alert box containing message with ok button.
confirm()	displays the confirm dialog box containing message with ok and cancel button.
prompt()	displays a dialog box to get input from the user.
open()	opens the new window.
close()	closes the current window.
setTimeout()	performs action after specified time like calling function, evaluating expressions etc.

Alert() JavaScript

- It displays alert dialog box. It has message and ok button.

```
<script type="text/javascript">
function msg()
{
 alert("Hello Alert Box");
}
</script>
<input type="button" value="click" onclick="msg()"/>
```

confirm Method

- It displays the confirm dialog box. It has message with ok and cancel buttons.

```
<script type="text/javascript">
function msg()
{
var v= confirm("Are u sure?");
if(v==true)
{
alert("ok");
}
else{
alert("cancel");
}
}
</script>
<input type="button" value="delete record" onclick="msg()"/>
```

Prompt method

- It displays prompt dialog box for input. It has message and textfield.

```
<script type="text/javascript">
function msg()
{
var v= prompt("Who are you?");
alert("I am "+v);
 }
</script>
<input type="button" value="click" onclick="msg()"/>
```

Open method

- It displays the content in a new window.

```
<script type="text/javascript">
function msg()
{
open("http://www.google.com");
}
</script>
<input type="button" value=“Click" onclick="msg()"/>
```

setTimeout() method

- It performs its task after the given milliseconds.

```
<script type="text/javascript">
function msg()
{
setTimeout(
function()
{
alert("Welcome to Java after 2 seconds")
},2000);

}
</script>

<input type="button" value="click" onclick="msg()"/>
```

Form validation

- It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user.
- JavaScript provides facility to validate the form on the client-side so data processing will be faster than server-side validation. Most of the web developers prefer JavaScript form validation.
- Through JavaScript, we can validate name, password, email, date, mobile numbers and more fields.

```
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;

if (name==null || name==""){
  alert("Name can't be blank");
  return false;
}else if(password.length<6){
  alert("Password must be at least 6 characters long.");
  return false;
  }
}
</script>
```

Form validation

```
<body>
<form name="myform" method="post" action="abc.jsp" onsubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
</body>
```

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

Form validation

```
<form name="f1" action="register.jsp" onsubmit="return matchpass()">
Password:<input type="password" name="password" /><br/>
Re-enter Password:<input type="password" name="password2"/><br/>
<input type="submit">
</form>
```

```
<script>
function validate(){
var num=document.myform.num.value;
if (isNaN(num)){
  document.getElementById("numloc").innerHTML="Enter Numeric va
lue only";
  return false;
}else{
  return true;
  }
}
</script>
```