**JavaScript**

* HTML is used to create the static web pages.
* Javascript is used to add the behaviour to static web pages to make web pages alive.
* To avoid the hitting of server
* To control exception handling
* Front end validation
* Javascript is a dynamic lightweight programming language
* Javascript is a case sensitive.

**Advantages of Javascript**

* Pre compile execution.
* Using javascript we can control html elements.
* Using Jsp we can create dynamic web page.
* To achieve event handling
* To validate html page.

**Disadvantages of JavaScript**

* Not a secured language
* Not support for multi threading concept
* Not recommended to developed Network based application.
* No file handling concept.

**History of Javascript**:

## JavaScript / ECMAScript / JScript

**JavaScript** was developed for Netscape. The first browser to run JavaScript was Netscape 2 in 1996. After Netscape the Mozilla foundation continued to develop JavaScript for the Firefox browser. Java Script version numbers run from 1.0 to 1.8.

**ECMAScript** was developed by Ecma International after the organization adopted JavaScript. The first edition of ECMAScript was released in 1997. ECMAScript version numbers run from 1 to 7.

**JScript** was developed by Microsoft as a compatible JavaScript language for Internet Explorer in 1996. JScript version numbers runs from 1.0 to 9.0.

At present, JavaScript can execute not only in the browser, but also on the server, or actually on any device where there exists a special program called [the JavaScript engine](https://en.wikipedia.org/wiki/JavaScript_engine).

The browser has an embedded engine, sometimes it’s also called a “JavaScript virtual machine”.

Different engines have different “codenames”, for example:

* [V8](https://en.wikipedia.org/wiki/V8_(JavaScript_engine)) – in Chrome and Opera.
* [SpiderMonkey](https://en.wikipedia.org/wiki/SpiderMonkey) – in Firefox.

Engines are complicated. But the basics are easy.

1. The engine (embedded if it’s a browser) reads (“parses”) the script.
2. Then it converts (“compiles”) the script to the machine language.
3. And then the machine code runs, pretty fast.

The engine applies optimizations on every stage of the process. It even watches the compiled script as it runs, analyzes the data that flows through it and applies optimizations to the machine code based on that knowledge. At the end, scripts are quite fast.

**ECMAScript Browser Implementations**

|  |  |  |
| --- | --- | --- |
| **Engine** | **ECMA** | **Browser** |
| V8 | 6 | Chrome (Partial Support) |
| SpiderMonkey | 6 | Firefox (Partial Support) |
| Chakra | 6 | Edge (Partial Support) |
| Nitro | 6 | Safari (Partial Support) |
| V8 | 6 | Opera (Partial Support) |
| V8 | 5 | Chrome 23 |
| SpiderMonkey | 5 | Firefox 21 |
| JavaScript 1.8.5 | 5 | Firefox 4 |
| Nitro | 5 | Safari 6 |
| V8 | 5 | Opera 15 |
| Chakra | 5 | Edge 12 |
| Chakra | 5 | IE 10 |

Example 1:

<!DOCTYPE html>

<html>

<body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">.</p>

<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>Click Me!</button>

</body>

</html>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change the style of an HTML element.</p>

<button type="button" onclick="document.getElementById('demo').style.fontSize='35px'">Click Me!</button>

</body>

</html>

## The <script> Tag

In HTML, JavaScript code must be inserted between <script> and </script> tags.

## JavaScript in <head> or <body>

You can place any number of scripts in an HTML document.

Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

Example 1:

<!DOCTYPE  html>  
<html>

<head>  
<script>  
function myFunction() {  
    document.getElementById("demo").innerHTML = "hooray won the match.";  
}  
</script>  
</head>

<body>

<h1>A Web Page</h1>  
<p id="demo"></p>  
<button type="button" onclick="myFunction()">Try it</button>

</body>  
</html>

## JavaScript in <body>

In this example, a JavaScript function is placed in the <body> section of an HTML page.

The function is invoked (called) when a button is clicked:

Example: <!DOCTYPE html>  
<html>  
<body>   
  
<h1>A Web Page</h1>  
<p id="demo">A Paragraph</p>  
<button type="button" onclick="myFunction()">Try it</button>  
  
<script>  
function myFunction() {  
   document.getElementById("demo").innerHTML = "Paragraph changed.";  
}  
</script>  
  
</body>  
</html>

## External JavaScript

Scripts can also be placed in external files:

function myFunction() {  
   document.getElementById("demo").innerHTML = "Paragraph changed.";  
}

External scripts are practical when the same code is used in many different web pages.

JavaScript files have the file extension**.js**.

To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

Ex:

<script src="myScript.js"></script>

Differences between javascript and java

|  |  |
| --- | --- |
| **JavaScript** | **Java** |
| Object-oriented. No distinction between types of objects. Inheritance is through the prototype mechanism, and properties and methods can be added to any object dynamically. | Class-based. Objects are divided into classes and instances with all inheritance through the class hierarchy. Classes and instances cannot have properties or methods |
| Variable data types are not declared (dynamic typing,losely typed). | Variable data types must be declared (static typing,strongly typed). |
| Cannot automatically write to hard disk | Can automatically write to hard disk. |

**KeyWords in JavaScript**

|  |  |
| --- | --- |
| **Keyword** | ***Description*** |
| Break | *Terminates a switch or a loop* |
| Continue | *Jumps out of a loop and starts at the top* |
| Debugger | *Stops the execution of JavaScript, and calls (if available) the debugging function* |
| do ... while | *Executes a block of statements, and repeats the block, while a condition is true* |
| For | *Marks a block of statements to be executed, as long as a condition is true* |
| Function | *Declares a function* |
| if ... else | *Marks a block of statements to be executed, depending on a condition* |
| Return | *Exits a function* |
| Switch | *Marks a block of statements to be executed, depending on different cases* |
| try ... catch | *Implements error handling to a block of statements* |
| Var | *Declares a variable* |

## JavaScript Arithmetic Operators

|  |  |
| --- | --- |
| **Operator** | **Description** |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| / | Division |
| % | Modulus (Remainder) |
| ++ | Increment |
| -- | Decrement |

## JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Example** | **Same As** |
| = | x = y | x = y |
| += | x += y | x = x + y |
| -= | x -= y | x = x - y |
| \*= | x \*= y | x = x \* y |
| /= | x /= y | x = x / y |
| %= | x %= y | x = x % y |

Example:

<!DOCTYPE html>

<html>

<body>

<h2>The += Operator</h2>

<p id="demo"></p>

<script>

var x = 10;

x += 5;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

## JavaScript Comparison Operators

|  |  |
| --- | --- |
| == | equal to |
| === | equal value and equal type |
| != | not equal |
| !== | not equal value or not equal type |
| > | greater than |
| < | less than |
| >= | greater than or equal to |
| <= | less than or equal to |
| ? | ternary operator |

## JavaScript Logical Operators

|  |  |
| --- | --- |
| **Operator** | **Description** |
| && | logical and |
| || | logical or |
| ! | logical not |

## JavaScript Type Operators

|  |  |
| --- | --- |
| **Operator** | **Description** |
| Typeof | Returns the type of a variable |
| instanceof | Returns true if an object is an instance of an object type |

**Java Script Arrays**

* Arrays is homogenous collection of indexed based elements.

Example: <!DOCTYPE html>

<html>

<body>

<h2>JavaScript Arrays</h2>

<p>Array indexes are zero-based, which means the first item is [0].</p>

<p id="demo"></p>

<script>

var cars = ["Rolls Royce","lamborgini","Audi"];

document.getElementById("demo").innerHTML = cars[0];

</script>

</body>

</html>

## JavaScript Objects

JavaScript objects are written with curly braces.

Object properties are written as name:value pairs, separated by commas.

Example:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Objects</h2>

<p id="demo"></p>

<script>

var person = {

firstName : "John",

lastName : "Doe",

age : 50,

eyeColor : "blue"

};

document.getElementById("demo").innerHTML =

person.firstName + " is " + person.age + " years old.";

</script>

</body>

</html>

## The typeof Operator

You can use the JavaScript **typeof** operator to find the type of a JavaScript variable.

The **typeof** operator returns the type of a variable or an expression:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript typeof</h2>

<p>The typeof operator returns the type of a variable or an expression.</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML =

typeof "" + "<br>" +

typeof "John" + "<br>" +

typeof "John Doe";

</script>

</body>

</html>

## Undefined

In JavaScript, a variable without a value, has the value**undefined**. The typeof is also **undefined**.

Example:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript</h2>

<p>The value (and the data type) of a variable with no value is <b>undefined</b>.</p>

<p id="demo"></p>

<script>

var car;

document.getElementById("demo").innerHTML =

car + "<br>" + typeof car;

</script>

</body>

</html>