What is JavaScript ?

Javascript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript was first known as **LiveScript,** but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name**LiveScript**. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

The ECMA-262 Specification defined a standard version of the core JavaScript language.

- 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

Advantages of JavaScript

The merits of using JavaScript are –

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

Syntax

JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.

```
<script ...>
   JavaScript code
</script>
```

The script tag takes two important attributes –

- **Language** – This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- **Type** – This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

So your JavaScript segment will look like –

```
<script language="javascript" type="text/javascript">
   JavaScript code
</script>
```

First JavaScript Script

Let us take a sample example to print out "Hello World". We added an optional HTML comment that surrounds our JavaScript code. This is to save our code from a browser that does not support JavaScript. The comment ends with a "//-->". Here "//" signifies a comment in JavaScript, so we add that to prevent a browser from reading the end of the HTML comment as a piece of JavaScript code. Next, we call a function document.write which writes a string into our HTML document.

```
<html>
   <body>
      <script language="javascript" type="text/javascript">
         <!--
            document.write("Hello World!")
         //-->
      </script>
   </body>
</html>
```

JavaScript in <body> and <head> Sections

You can put your JavaScript code in <head> and <body> section altogether as follows –

```
<html>
   <head>
      <script type="text/javascript">
         <!--
            function sayHello() {
               alert("Hello World")
            }
         //-->
      </script>
   </head>

   <body>
      <script type="text/javascript">
         <!--
            document.write("Hello World")
         //-->
      </script>

      <input type="button" onclick="sayHello()" value="Say Hello" />

   </body>
</html>
```

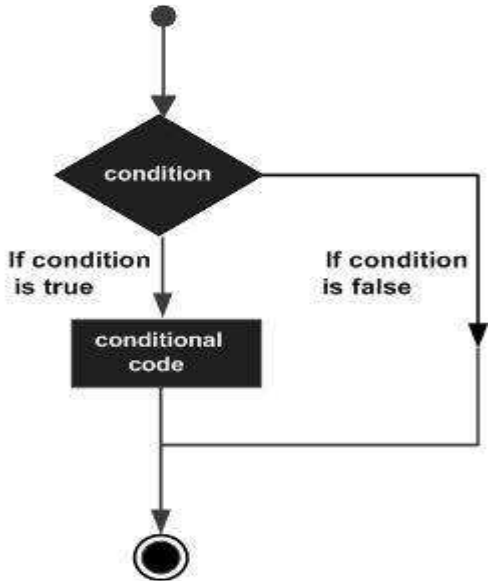
What is an operator?

Let us take a simple expression **4 + 5 is equal to 9**. Here 4 and 5 are called**operands** and '+' is called the **operator**. JavaScript supports the following types of operators.

- Arithmetic Operators
- Comparision Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Flow Chart of if-else

The following flow chart shows how the if-else statement works.



JavaScript supports the following forms of **if..else** statement –

- if statement
- if...else statement
- if...else if... statement.

if statement

The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

```
if (expression){
   Statement(s) to be executed if expression is true
}
```

if...else statement:

The 'if...else' statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.

```
if (expression){
   Statement(s) to be executed if expression is true
}

else{
   Statement(s) to be executed if expression is false
}
```

if...else if... statement

The if...else if... statement is an advanced form of if…else that allows JavaScript to make a correct decision out of several conditions.

```
if (expression 1){
   Statement(s) to be executed if expression 1 is true
}

else if (expression 2){
   Statement(s) to be executed if expression 2 is true
}

else if (expression 3){
   Statement(s) to be executed if expression 3 is true
}

else{
   Statement(s) to be executed if no expression is true
}
```

Function Definition

Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the function keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

```
<script type="text/javascript">
   <!--
      function functionname(parameter-list)
      {
         statements
      }
   //-->
</script>
```

Calling a Function

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

```
<html>
```

```
   <head>

      <script type="text/javascript">
         function sayHello()
         {
            document.write ("Hello there!");
         }
      </script>

   </head>
   <body>
      <p>Click the following button to call the function</p>

      <form>
         <input type="button" onclick="sayHello()" value="Say Hello">
      </form>

      <p>Use different text in write method and then try...</p>
   </body>
</html>
```

Function Parameters

Till now, we have seen functions without parameters. But there is a facility to pass different parameters while calling a function. These passed parameters can be captured inside the function and any manipulation can be done over those parameters. A function can take multiple parameters separated by comma.

```
<html>
   <head>

      <script type="text/javascript">
         function sayHello(name, age)
         {
            document.write (name + " is " + age + " years old.");
         }
      </script>

   </head>
   <body>
      <p>Click the following button to call the function</p>

      <form>
         <input type="button" onclick="sayHello('Zara', 7)" value="Say Hello">
      </form>

      <p>Use different parameters inside the function and then try...</p>
   </body>
</html>
```

The return Statement

A JavaScript function can have an optional return statement. This is required if you want to return a value from a function. This statement should be the last statement in a function.

```
<html>
   <head>

      <script type="text/javascript">
         function concatenate(first, last)
         {
            var full;
            full = first + last;
            return full;
         }

         function secondFunction()
         {
            var result;
            result = concatenate('Zara', 'Ali');
            document.write (result );
         }
      </script>

   </head>

   <body>
      <p>Click the following button to call the function</p>

      <form>
         <input type="button" onclick="secondFunction()" value="Call Function">
      </form>

      <p>Use different parameters inside the function and then try...</p>

  </body>
</html>
```

What is an Event ?

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.

onclick Event Type

```
<html>
   <head>

      <script type="text/javascript">
         <!--
            function sayHello() {
               alert("Hello World")
            }
         //-->
```

```
      </script>

   </head>

   <body>
      <p>Click the following button and see result</p>

      <form>
         <input type="button" onclick="sayHello()" value="Say Hello" />
      </form>

   </body>
</html>
```

onsubmit Event type

```
<html>
   <head>

      <script type="text/javascript">
         <!--
            function validation() {
               all validation goes here
               .........
               return either true or false
            }
         //-->
      </script>

   </head>
   <body>

      <form method="POST" action="t.cgi" onsubmit="return validate()">
         .......
         <input type="submit" value="Submit" />
      </form>

   </body>
</html>
```

onmouseover and onmouseout

```
<html>
   <head>

      <script type="text/javascript">
         <!--
            function over() {
               document.write ("Mouse Over");
            }

            function out() {
               document.write ("Mouse Out");
            }

         //-->
      </script>

   </head>
   <body>
      <p>Bring your mouse inside the division to see the result:</p>

      <div onmouseover="over()" onmouseout="out()">
         <h2> This is inside the division </h2>
      </div>

   </body>
</html>
```

What is Page Redirection ?

You might have encountered a situation where you clicked a URL to reach a page X but internally you were directed to another page Y. It happens due to**page redirection**. This concept is different from JavaScript Page Refresh.

There could be various reasons why you would like to redirect a user from the original page. We are listing down a few of the reasons –

- You did not like the name of your domain and you are moving to a new one. In such a scenario, you may want to direct all your visitors to the new site. Here you can maintain your old domain but put a single page with a page redirection such that all your old domain visitors can come to your new domain.

- You have built-up various pages based on browser versions or their names or may be based on different countries, then instead of using your server-side page redirection, you can use client-side page redirection to land your users on the appropriate page.

- The Search Engines may have already indexed your pages. But while moving to another domain, you would not like to lose your visitors coming through search engines. So you can use client-side page redirection. But keep in mind this should not be done to fool the search engine, it could lead your site to get banned.

```
<html>
```

```
   <head>

      <script type="text/javascript">
         <!--
            function Redirect() {
               window.location="http://www.tutorialspoint.com";
            }
         //-->
      </script>

   </head>

   <body>
      <p>Click the following button, you will be redirected to home page.</p>

      <form>
         <input type="button" value="Redirect Me" onclick="Redirect();" />
      </form>

   </body>
</html>
```