JavaScript Basics

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.

- 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

Client-side JavaScript

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.

The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.

JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

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.

Limitations of JavaScript

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features –

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multithreading or multiprocessor capabilities.

Once again, JavaScript is a lightweight, interpreted programming language that allows you to build interactivity into otherwise static HTML pages.

JavaScript Syntax

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

You can place the **<script>** tags, containing your JavaScript, anywhere within you web page, but it is normally recommended that you should keep it within the **<head>** tags.

The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

```
<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>
```

Your First JavaScript Script

Let us take a sample example to print out "Hello World".

we call a function **document.write** which writes a string into our HTML document.

his function can be used to write text, HTML, or both. Take a look at the following code.

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

This code will produce the following result –

```
Hello World!
```

Whitespace and Line Breaks

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. You can use spaces, tabs, and newlines freely in your program and you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

Semicolons are Optional

Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if each of your statements are placed on a separate line. For example, the following code could be written without semicolons.

```
<script language="javascript" type="text/javascript">
      var1 = 10
      var2 = 20
</script>
```

But when formatted in a single line as follows, you must use semicolons –

```
<script language="javascript" type="text/javascript">
      var1 = 10; var2 = 20;
</script>
```

Note – It is a good programming practice to use semicolons.

Case Sensitivity

JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

So the identifiers **Time** and **TIME** will convey different meanings in JavaScript.

NOTE – Care should be taken while writing variable and function names in JavaScript.

Comments in JavaScript

JavaScript supports both C-style and C++-style comments, Thus –

- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

Example

The following example shows how to use comments in JavaScript.

```
<script language="javascript" type="text/javascript">
   <!--
      // This is a comment. It is similar to comments in C++
      /*
      * This is a multiline comment in JavaScript
      * It is very similar to comments in C Programming
      */
   //-->
</script>
```

JavaScript - Placement in HTML File

There is a flexibility given to include JavaScript code anywhere in an HTML document. However the most preferred ways to include JavaScript in an HTML file are as follows −

- Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections.
- Script in an external file and then include in <head>...</head> section.

In the following section, we will see how we can place JavaScript in an HTML file in different ways.

JavaScript in <head>...</head> section

If you want to have a script run on some event, such as when a user clicks somewhere, then you will place that script in the head as follows −

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

   </head>

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

```
</html>
```

JavaScript in <body>...</body> section

If you need a script to run as the page loads so that the script generates content in the page, then the script goes in the <body> portion of the document. In this case, you would not have any function defined using JavaScript. Take a look at the following code.

```
<html>

   <head>
   </head>

   <body>

      <script type="text/javascript">
            document.write("Hello World")
      </script>
      <p>This is web page body </p>

   </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>
```

JavaScript in External File

Here is an example to show how you can include an external JavaScript file in your HTML code using **script** tag and its **src** attribute.

```
<html>

   <head>
      <script type="text/javascript" src="filename.js" ></script>
   </head>

   <body>
```

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

To use JavaScript from an external file source, you need to write all your JavaScript source code in a simple text file with the extension ".js" and then include that file as shown above.

For example, you can keep the following content in **filename.js** file and then you can use **sayHello** function in your HTML file after including the filename.js file.

```
function sayHello() {
   alert("Hello World")
}
```

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

- Writing into an HTML element, using **innerHTML**.
- Writing into the HTML output using **document.write()**.
- Writing into an alert box, using **window.alert()**.
- Writing into the browser console, using **console.log()**.

Using innerHTML

To access an HTML element, JavaScript can use the document.getElementById(id) method.

The **id** attribute defines the HTML element. The **innerHTML** property defines the HTML content:

Example:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

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

<script>
document.getElementById("demo").innerHTML = “welcome”;
</script>

</body>
</html>
```

Using document.write()

Example:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
document.write("welcome to javascript");
</script>

</body>
</html>
```

Note: Using document.write() after an HTML document is fully loaded, will **delete all existing HTML**

Using window.alert()

You can use an alert box to display data:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
window.alert("welcome");
</script>

</body>
</html>
```

Using console.log()

For debugging purposes, you can use the **console.log()** method to display data.

```
<!DOCTYPE html>
<html>
<body>

<script>
console.log("welcome");
</script>

</body>
</html>
```

JavaScript Variables

Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container.

Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the **var** keyword as follows.

```
<script type="text/javascript">
      var money;
      var name;
</script>
```

You can also declare multiple variables with the same **var** keyword as follows –

```
<script type="text/javascript">
      var money, name;
</script>
```

Storing a value in a variable is called **variable initialization**. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

For instance, you might create a variable named **money** and assign the value 2000.50 to it later. For another variable, you can assign a value at the time of initialization as follows.

```
<script type="text/javascript">
      var name = "Ali";
      var money;
      money = 2000.50;
</script>
```

Note – Use the **var** keyword only for declaration or initialization, once for the life of any variable name in a document. You should not re-declare same variable twice.

JavaScript is **untyped** language. This means that a JavaScript variable can hold a value of any data type. Unlike many other languages, you don't have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

JavaScript Variable Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

- **Global Variables** – A global variable has global scope which means it can be defined anywhere in your JavaScript code.
- **Local Variables** – A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Within the body of a function, a local variable takes precedence over a global variable with the same name. If you declare a local variable or function parameter with the same name as a global variable, you effectively hide the global variable. Take a look into the following example.

```
<html>
   <body onload = checkscope();>
      <script type = "text/javascript">
            var myVar = "global"; // Declare a global variable
            function checkscope( ) {
               var myVar = "local";  // Declare a local variable
               document.write(myVar);
            }
      </script>
   </body>
</html>
```

JavaScript Variable Names

While naming your variables in JavaScript, keep the following rules in mind.

- You should not use any of the JavaScript reserved keywords as a variable name. These keywords are mentioned in the next section. For example, **break** or **boolean** variable names are not valid.
- JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or an underscore character. For example, **123test** is an invalid variable name but **_123test** is a valid one.
- JavaScript variable names are case-sensitive. For example, **Name** and **name** are two different variables.

JavaScript – Objects

JavaScript is an Object Oriented Programming (OOP) language. A programming language can be called object-oriented if it provides four basic capabilities to developers –

- **Encapsulation** – the capability to store related information, whether data or methods, together in an object.
- **Aggregation** – the capability to store one object inside another object.
- **Inheritance** – the capability of a class to rely upon another class (or number of classes) for some of its properties and methods.
- **Polymorphism** – the capability to write one function or method that works in a variety of different ways.

Objects are composed of attributes. If an attribute contains a function, it is considered to be a method of the object, otherwise the attribute is considered a property.

Object Properties

Object properties can be any of the three primitive data types, or any of the abstract data types, such as another object. Object properties are usually variables that are used internally in the object's methods, but can also be globally visible variables that are used throughout the page.

The syntax for adding a property to an object is –

```
objectName.objectProperty = propertyValue;
```

For example – The following code gets the document title using the **"title"** property of the **document** object.

```
var str = document.title;
```

Object Methods

Methods are the functions that let the object do something or let something be done to it. There is a small difference between a function and a method – at a function is a standalone unit of statements and a method is attached to an object and can be referenced by the **this** keyword.

Methods are useful for everything from displaying the contents of the object to the screen to performing complex mathematical operations on a group of local properties and parameters.

For example – Following is a simple example to show how to use the **write()** method of document object to write any content on the document.

```
document.write("This is test");
```

User-Defined Objects

All user-defined objects and built-in objects are descendants of an object called **Object**.

The new Operator

The **new** operator is used to create an instance of an object. To create an object, the **new** operator is followed by the constructor method.

In the following example, the constructor methods are Object(), Array(), and Date(). These constructors are built-in JavaScript functions.

```
var employee = new Object();
var books = new Array("C++", "Perl", "Java");
var day = new Date("August 15, 1947");
```

The Object() Constructor

A constructor is a function that creates and initializes an object. JavaScript provides a special constructor function called **Object()** to build the object. The return value of the **Object()** constructor is assigned to a variable.

The variable contains a reference to the new object. The properties assigned to the object are not variables and are not defined with the **var** keyword.

Example 1

Try the following example; it demonstrates how to create an Object.

```
<html>
   <head>
      <title>User-defined objects</title>
      <script type="text/javascript">
         var book = new Object();   // Create the object
         book.subject = "Perl"; // Assign properties to the object
         book.author  = "Mohtashim";
      </script>
   </head>
   <body>
      <script type="text/javascript">
         document.write("Book name is : " + book.subject + "<br>");
         document.write("Book author is : " + book.author + "<br>");
```

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

Output

```
Book name is : Perl
Book author is : Mohtashim
```

Example 2

This example demonstrates how to create an object with a User-Defined Function. Here **this** keyword is used to refer to the object that has been passed to a function.

```
<html>
   <head>
   <title>User-defined objects</title>
      <script type="text/javascript">
         function book(title, author){
            this.title = title;
            this.author  = author;
         }
      </script>

   </head>
   <body>

      <script type="text/javascript">
         var myBook = new book("Perl", "Mohtashim");
         document.write("Book title is : " + myBook.title + "<br>");
         document.write("Book author is : " + myBook.author + "<br>");
      </script>

   </body>
</html>
```

Output

```
Book title is : Perl
Book author is : Mohtashim
```

Defining Methods for an Object

The previous examples demonstrate how the constructor creates the object and assigns properties. But we need to complete the definition of an object by assigning methods to it.

Example

Try the following example; it shows how to add a function along with an object.

```
<html>
   <head>
   <title>User-defined objects</title>
      <script type="text/javascript">
         // Define a function which will work as a method
         function addPrice(amount){
            this.price = amount;
         }
         function book(title, author){
            this.title = title;
            this.author  = author;
            this.addPrice = addPrice; // Assign that method as property.
         }
      </script>
   </head>
   <body>
      <script type="text/javascript">
         var myBook = new book("Perl", "Mohtashim");
         myBook.addPrice(100);
         document.write("Book title is : " + myBook.title + "<br>");
         document.write("Book author is : " + myBook.author + "<br>");
         document.write("Book price is : " + myBook.price + "<br>");
      </script>
   </body>
</html>
```

Output

```
Book title is : Perl
Book author is : Mohtashim
Book price is : 100
```

The 'with' Keyword

The **'with'** keyword is used as a kind of shorthand for referencing an object's properties or methods.

The object specified as an argument to **with** becomes the default object for the duration of the block that follows. The properties and methods for the object can be used without naming the object.

Syntax

The syntax for with object is as follows –

```
with (object){
   properties used without the object name and dot
}
```

Example

Try the following example.

```
<html>
   <head>
   <title>User-defined objects</title>
      <script type="text/javascript">
         // Define a function which will work as a method
         function addPrice(amount){
            with(this){
               price = amount;
            }
         }
         function book(title, author){
            this.title = title;
            this.author  = author;
            this.price = 0;
            this.addPrice = addPrice; // Assign that method as property.
         }
      </script>
   </head>
   <body>
      <script type="text/javascript">
         var myBook = new book("Perl", "Mohtashim");
         myBook.addPrice(100);

         document.write("Book title is : " + myBook.title + "<br>");
         document.write("Book author is : " + myBook.author + "<br>");
```

```
         document.write("Book price is : " + myBook.price + "<br>");
      </script>
   </body>
</html>
```

Output

```
Book title is : Perl
Book author is : Mohtashim
Book price is : 100
```

JavaScript form validation

Java Script Form validation for Empty fields

```
<html>
  <head>
  <script>
  function validateForm()
  {
        var un = document.registration.uname.value;
        var pass = document.registration.pwd.value;
        var gen = document.registration.gender.value;
        var age = document.registration.age.value;
        var mail = document.registration.email.value;
        var cntry = document.registration.country.value;
        var lan = document.registration.lang;
        if(un=="" || un=="null")
        {
              alert("please enter ur name");
              document.registration.uname.focus();
              return false;

        }
        if(pass=="" || pass=="null")
        {
              alert("please enter ur password");
              document.registration.pwd.focus();
              return false;
        }
       if(gen=="" || gen=="null")
       {
              alert("Please select the gender");
              document.registration.gender[0].focus();
              return false;

       }
       if(age=="" || age=="null")
       {
              alert("please enter your age");
              document.registration.age.focus();
              return false;

       }
```

```
        if(mail=="" || mail=="null")
        {
                alert("please enter ur email");
                document.registration.email.focus();
                return false;
        }
        if(cntry == "Default")
        {
                 alert("please select your country")
                document.registration.country.focus();
                return false;
        }
        for (var i=0; i<lan.length; i++)
       {
                if(lan[i].checked)
                break;
       }
       if(i==lan.length)
       {
                alert("Please choose your languages");
                document.registration.lang[0].focus();
                return false;
       }
       alert("Registration successfull");
       return true;
  }
  </script>
  </head>
  <body>
  <form name="registration" onsubmit="return validateForm();">
  Enter your name: <input type="text" name="uname"><br>
  Enter your password: <input type="password" name="pwd"><br>
  Choose your Gender: <input type="radio" name="gender" value="Male">Male
                  <input type="radio" name="gender" value="Female">Female<br>
  Enter your Age: <input type="number" name="age"><br>
  Enter your email: <input type="text" name="email"><br>
  Choose ur country:
  <select name="country">
        <option value="Default">(Please select a country)</option>
        <option value="aus">Australia</option>
        <option value="can">Canada</option>
        <option value="ind">India</option>
        <option value="rus">Russia</option>
        <option value="usa">USA</option>
  </select><br>
  Languages Known:
  Telugu: <input type="checkbox" name="lang" value="Telugu"/>
  English: <input type="checkbox" name="lang" value="English"/>
  Hindi: <input type="checkbox" name="lang" value="Hindi"/><br>

  <input type="submit"><input type="reset">
  </form>
  </body>
</html>
```

JavaScript Form validation:

a) **User Name:** Please enter alphabets only

b) **Age :** please enter numbers only

c) Password: at least one lowercase, uppercase & special character.

```
<html>
 <head>
 <script>
 function validateForm()
 {
   var un = document.registration.uname.value;
   var age = document.registration.age.value;
   var pass = document.registration.pwd.value;
   var letters  = /^[a-zA-Z\s]+$/;
   var numbers = /^[0-9]+$/;
   var checkSpecial = /[*@!#%&]+/.test(pass);
   var checkUpper = /[A-Z]+/.test(pass);
   var checkLower = /[a-z]+/.test(pass);
   var flag = 0;
   if(un=="" || un=="null")
   {
      alert("please enter ur name");
      document.registration.uname.focus();
      return false;
   }
   if(!un.match(letters))
   {
      alert("please enter alphabets only")
      document.registration.uname.focus();
      return false;
   }
   if(age=="" || age=="null")
   {
      alert("please enter ur age");
      document.registration.age.focus();
      return false;

   }
   if(!age.match(numbers))
   {
      alert("please enter numbers only")
      document.registration.age.focus();
      return false;
   }
   if(pass=="" || pass=="null")
   {
      alert("please enter ur password");
      document.registration.pwd.focus();
      return false;
   }

   if(checkSpecial && checkLower && checkUpper)
   {
      flag = 1;
   }
   if(flag == 0)
   {
      alert("Password should be the combination of atleast one lowercase letter, uppercase
letter & special symbols like ($, & #)");
      document.registration.pwd.focus();
```

```
        return false;
     }
     alert("Registration successfull");
     return true;
  }
  </script>
  </head>
  <body>
  <form name="registration" onsubmit="return validateForm();">
  Enter your name: <input type="text" name="uname"> Please enter alphabets only<br>
  Enter your age: <input type="text" name="age"> pleae enter numbers only<br>
  Enter your password: <input type="text" name="pwd">atleast one lowercase, uppercase &
special character<br>
  <input type="submit"><input type="reset">
  </form>
  </body>
</html>
```

JavaScript Email validation:

```
<html>
  <head>
  <script>
  function validateForm()
  {
     var mail = document.registration.email.value;
        if(mail=="" || mail=="null")
        {
              alert("please enter your email");
              document.registration.email.focus();
              return false;
        }
        var atpos = mail.indexOf("@");
        var dotpos = mail.lastIndexOf(".");
        if (atpos < 1 || dotpos < atpos + 2 || dotpos + 2 >= mail.length)
        {
              alert("Not a valid e-mail address");
              return false;
     }
     alert("Registration successfull");
     return true;
  }
  </script>
  </head>
  <body>
  <form name="registration" onsubmit="return validateForm();">
    Enter your email: <input type="text" name="email"><br>
  <input type="submit"><input type="reset">
  </form>
  </body>
</html>
```