JavaScript Fundamentals

JavaScript:

- JavaScript is a most popular lightweight, integrated high level programming language.
- JavaScript helps to develop the front-end as well as the back-end software by providing different JavaScript frameworks such as Jquery, Node.js etc.
- JavaScript makes a website dynamic and Interactive with the users.
- JavaScript usage has extended to mobile app development, Desktop app development and Game development.
- It comes installed in all modern web browsers.

Add JavaScript to Web page:

- JavaScript is implemented using the JavaScript statements that are placed within the <script>....</script> HTML tags.
- 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.
- The **script** tag provides a mechanism to allow you to store JavaScript in an external file and then include it into your HTML files.

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

Language Basics:

Variables:

- JavaScript allows to work with three primitive data types :
  1. **Numbers,** eg. 123, 120.50 etc.
  2. **Strings** of text e.g. "This text string" etc.

3. **Boolean** e.g. true or false.

- JavaScript also defines two trivial data types, **null** and **undefined,** each of which defines only a single value.
- JavaScript supports a composite data type known as **object**.
- Variables are declared with the **var** keyword :

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

        var name="john";

        var age=23;

        var ismarried=false;

</script>
```

Operators:

JavaScript supports the following types of operators.

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

Conditions and Loops:

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

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

Switch statement:

The **switch** statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each **case** against the value of the expression until a match is found. If nothing matches, a **default** condition will be used. The **break** statements indicate the end of a particular case. If they were omitted, the interpreter would continue executing each statement in each of the following cases.

```
switch (expression) {
  case condition 1: statement(s)
  break;
  case condition 2: statement(s)
  break;
  ...

  case condition n: statement(s)
  break;

  default: statement(s)
}
```

While Loop:

The purpose of a **while** loop is to execute a statement or code block repeatedly as long as an **expression** is true. Once the expression becomes **false,** the loop terminates.

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

For Loop:

The syntax of **for** loop is JavaScript is as follows −

```
for (initialization; test condition; iteration statement) {
            Statement(s) to be executed if test condition is true
}
```

For … in Loop:

The syntax of 'for..in' loop is −

```
for (variablename in object) {
            statement or block to execute
}
```

In each iteration, one property from **object** is assigned to **variablename** and this loop continues till all the properties of the object are exhausted.

Functions:

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.

Syntax:

```
<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.

Syntax:

```
functionName();
```

Events:

- 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.
- Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

Some of the Events:

1. onclick
2. onsubmit
3. onmouseout
4. onmouseover
5. onfocus
6. onselect

JavaScript Debugging:

Use a JavaScript Validator:

One way to check your JavaScript code for strange bugs is to run it through a program that checks it to make sure it is valid and that it follows the official syntax rules of the language. These programs are called validating parsers or just validators for short, and often come with commercial HTML and JavaScript editors.

Add Debugging Code to Your Programs

You can use the **alert()** or **document.write()** methods in your program to debug your code.

Use a JavaScript Debugger:

A debugger is an application that places all aspects of script execution under the control of the programmer. Debuggers provide fine-grained control over the state of the script through an interface that allows you to examine and set values as well as control the flow of execution.

Strings:

The **String** object lets you work with a series of characters; it wraps Javascript's string primitive data type with a number of helper methods.

As JavaScript automatically converts between string primitives and String objects, you can call any of the helper methods of the String object on a string primitive.

Syntax to create a String object −

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

String Methods:

String Concatenation: Combines text of two strings and returns a new string.

string.concat(string2, string3[, ..., stringN]);

String length: Returns number of characters in a String.

```
String.length;
```

indexOf: Returns the index of found occurrence, otherwise -1;

```
string.indexOf(searchValue[, fromIndex])
```

Slice: Extracts a section of a string and returns a new string.

```
string.slice( beginslice [, endSlice] );
```

toUpperCase: Returns a calling string value converted to upper case.

```
string.toUpperCase();
```

toLowerCase: Returns a calling string value converted to lower case.

```
String.toLowerCase();
```

Replace: It simply returns a new changed string.

```
string.replace(regexp/substr, newSubStr/function[, flags]);
```

Arrays:

The **Array** object lets you store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Syntax:

syntax to create an **Array** object −

```
var fruits = new Array( "apple", "orange", "mango" );
```

we can create array by simply assigning values as follows −

```
var fruits = [ "apple", "orange", "mango" ];
```

we will use ordinal numbers to access and to set values inside an array as follows.

```
fruits[0] is the first element
fruits[1] is the second element
fruits[2] is the third element
```

Array Methods:

indexOf():

Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.

```
array.indexOf(searchElement[, fromIndex]);
```

pop()

Removes the last element from an array and returns that element

```
array.pop();
```

Push()

Adds one or more elements to the end of an array and returns the new length of the array.

```
array.push(element1, ..., elementN);
```

shift()

Removes the first element from an array and returns that element.

```
array.shift();
```

unshift():

Adds one or more elements to the front of an array and returns the new length of the array.

```
array.unshift( element1, ..., elementN );
```

Objects:

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.

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.

```
var employee = new Object();
```

Number Object:

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

Boolean Object:

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

String Object:

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

Array Object:

```
var fruits = new Array( "apple", "orange", "mango" );
```

Date Object:

```
new Date();
```

Math Object:

```
var pi_val = Math.PI;
var sine_val = Math.sin(30);
```

This Keyword:

The JavaScript this keyword refers to the object it belongs to.

It has different values depending on where it is used:

- In a method, this refers to the **owner object**.
- Alone, this refers to the **global object**.
- In a function, this refers to the **global object**.
- In a function, in strict mode, this is undefined.
- In an event, this refers to the **element** that received the event.
- Methods like call(), and apply() can refer this to **any object**.