CHAPTER -3

JAVA SCRIPT

COURSE OUTLINE

- JavaScript Syntax
- Types Of JavaScript
- Variables
- Arrays
- Functions
- Conditions & Loops
- Pop Up Boxes
- Javascript Objects And DOM,
- Javascript Inbuilt Functions,
- Javascript Validations,
- Regular Expressions,
- Event Handling With Javascript,
- Callbacks In Javascript,
- Function As Arguments In Javascript,
- Object Concepts In Javascript, JSON

WHAT IS JAVASCRIPT ?

- JavaScript is a lightweight, cross-platform and interpreted scripting language.
- JavaScript can be used for Client-side developments as well as Server-side developments.
- JavaScript is the most popular language on earth.
- With advances in browser technology and JavaScript having moved into the server with Node.js and other frameworks, JavaScript is capable of so much more. Here are a few things that we can do with JavaScript:
- JavaScript was created in the first place for DOM manipulation. Earlier websites were mostly static, after JS was created dynamic Web sites were made.
- Functions in JS are objects. They may have properties and methods just like another object. They can be passed as arguments in other functions.
- Can handle date and time.
- Performs Form Validation although the forms are created using HTML.
- No compiler needed.

Java script 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 your 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.
- Basic Syntax :

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

Example :

```
<script language =“javascript” type =“text/javascript”>
        Javascipt Code
</script>
```

First JavaScript Program

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

Output :

Hello World!

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.

```
<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>
```

JavaScript Data Types

- One of the most fundamental characteristics of a programming language is the set of data types it supports.
- These are the type of values that can be represented and manipulated in a programming language.
- JavaScript allows you to work with three primitive data types –
- **Numbers, eg. 123, 120.50 etc.**
- **Strings of text e.g. "This text string" etc.**
- **Boolean e.g. true or false.**
- JavaScript also defines two trivial data types, null and undefined, each of which defines only a single value. In addition to these primitive data types, JavaScript supports a composite data type known as object.
- We will cover objects in detail in a separate chapter."}; // Object

JavaScript Variables

- 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>
```

- 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.

JavaScript Output

- 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().
- innerHTML used 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:

JavaScript - if...else Statement

- While writing a program, there may be a situation when you need to adopt one out of a given set of paths.
- In such cases, you need to use conditional statements that allow your program to make correct decisions and perform right actions.
- JavaScript supports conditional statements which are used to perform different actions based on different conditions.

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

JavaScript Switch

- The JavaScript switch statement is used to execute one code from multiple expressions.
- It is just like else if statement that we have learned in previous page.
- But it is convenient than if..else..if because it can be used with numbers, characters etc.
- **Synatax :**

```
switch(expression) {
        case value1:
        code to be executed;
         break;
        case value 2:
        code to be executed;
        break;
        ......

        default:
        code to be executed if above values are not matched;
}
```

JavaScript Loops

- The JavaScript loops are used to iterate the piece of code using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.
- There are four types of loops in JavaScript.

- for loop
- while loop
- do-while loop
- for-in loop

JavaScript For loop

- The **JavaScript for loop** *iterates the elements for the fixed number of times*. It should be used if number of iteration is known. The syntax of for loop is given below.
- Synax :

```
for (initialization; condition; increment/decrement)
{
    code to be executed
}
```

- Example :

```
<script>
for (i=1; i<=5; i++)
{
document.write(i + "<br/>")
}
</script>
```

While loop in Java Script

- 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.
- The syntax of while loop in JavaScript is as follows −

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

Do...while Loop in Java Script

- The **do...while** loop is similar to the **while** loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is **false**.
- The syntax for do-while loop in JavaScript is as follows –

```
do {
   Statement(s) to be executed;
} while (expression);
```

for...in loop in Java Script

- The for...in loop is used to loop through an object's properties.
- 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.

Array in JavaScript

- 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 : var <array-name> = [element0, element1, element2,... elementN];**
- **Example : var fruits = new Array("apple", "orange", "mango");**
- You can create array by simply assigning values as follows −

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

Arrays and Memory

- In a computer's memory, an array represents multiple contiguous locations that all contain the values of the *same data type*.
- We represent each "slot" in an array with a number:
  - *Numbers start at zero.*
  - *An element's position (represented by a number) is called an index (or subscript)*
  - *For Ex. Fruits[0] =apple*

    fruits[1]=orange

    fruits[2]=mango

Array Elements

- Elements populate an array.
- Each element is a discrete value.
- We identify elements using a unique index (the element's "address").
- Array index numbering starts with zero and increments by one for each new element.

Adding Values to an Array

- To add values to an array, assign the new values while also identifying the index where you want JavaScript to store the new value:

```
fruits[0] = “mango”;
fruits[1] = “apple”;
fruits[2] = “grapes”;
fruits[3] = “banana”;
```

The Array.length Property

- Just like any JavaScript objects, arrays have properties.
- One of the most useful properties is the length property, which will give you a count of how many indexes the array has.

The Array.length Property

- Example of the Array.length property:

```
var fruits = new Array();
var fruits = 0;

fruits[0] = apple;
fruits[1] = orange;
fruits[2] = mango;

Size = fruits.length;
```

Using Array.length

- The following code averages the values stored in an array

```
for(var i=0; i<golfScores.length; i++)
{
    sumScores += golfScores[i]
}
avgScore = sumScores/golfScores.length;
```

JavaScript - Functions

- A function is a group of reusable code which can be called anywhere in your program.
- This eliminates the need of writing the same code again and again.
- It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.
- You must have seen functions like alert() and write() . We were using these functions again and again, but they had been written in core JavaScript only once.

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.
- Syntax : The basic syntax is shown here.

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

Example

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

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.

- For example, you can pass two numbers in a function and then you can expect the function to return their multiplication in your calling program.

JavaScript - The Function() Constructor

- The function statement is not the only way to define a new function; you can define your function dynamically using Function() constructor along with the new operator.
- Syntax : Following is the syntax to create a function using Function( ) constructor along with the new operator.

```
<script type = "text/javascript">
   <!--
      var variablename = new Function(Arg1,Arg2...,"Function Body");
   //-->
</script>
```

- The Function() constructor expects any number of string arguments. The last argument is the body of the function – it can contain arbitrary JavaScript statements, separated from each other by semicolons.
- Notice that the Function() constructor is not passed any argument that specifies a name for the function it creates. The unnamed functions created with the Function() constructor are called anonymous functions.

JavaScript - Dialog Boxes

- JavaScript supports three important types of dialog boxes. These dialog boxes can be used to raise and alert, or to get confirmation on any input or to have a kind of input from the users.
- Types of Dialog(PopUp) Box

1. Alert Box
2. Confirm Box
3. Prompt Box

Alert Dialog Box

- An alert dialog box is mostly used to give a warning message to the users.
- For example, if one input field requires to enter some text but the user does not provide any input, then as a part of validation, you can use an alert box to give a warning message.
- Nonetheless, an alert box can still be used for friendlier messages. Alert box gives only one button "OK" to select and proceed.

Confirmation Dialog Box

- A confirmation dialog box is mostly used to take user's consent on any option. It displays a dialog box with two buttons: OK and Cancel.
- If the user clicks on the OK button, the window method confirm() will return true. If the user clicks on the Cancel button, then confirm() returns false.

Prompt Dialog Box

- The prompt dialog box is very useful when you want to pop-up a text box to get user input. Thus, it enables you to interact with the user. The user needs to fill in the field and then click OK.

- This dialog box is displayed using a method called prompt() which takes two parameters: (i) a label which you want to display in the text box and (ii) a default string to display in the text box.

- This dialog box has two buttons: OK and Cancel. If the user clicks the OK button, the window method prompt() will return the entered value from the text box. If the user clicks the Cancel button, the window method prompt() returns null.

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** − Encapsulation is a process of binding the data (i.e. variables) with the functions acting on that data. It allows us to control the data and validate it.
- **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;
- Refer Example

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

- An object is just a special kind of data, with properties and methods.
- Accessing Object Properties
- Properties are the values associated with an object.
- The syntax for accessing the property of an object is below
- objectName.propertyName
- This example uses the length property of the Javascript's inbuilt object(String) to find the length of a string:
- var message="Hello World!";
var x=message.length;
- Refer Example

Accessing Object Methods

- Methods are the actions that can be performed on objects.
- You can call a method with the following syntax.objectName.methodName()
- This example uses the toUpperCase method of the String object to convert string to upper case:
- ```
  var message="Hello World!";
  var x=message.toUpperCase();
  ```
  ```

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.

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.

- Refer Example

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**

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

JavaScript’s inbuilt Objects

- JavaScript comes with some inbuilt objects which are,
- String
- Date
- Array
- Boolean
- Math
- RegExp

Math Object

- The Math object allows you to perform mathematical tasks.
- The Math object includes several mathematical constants and methods.
- Example for using properties/methods of Math:

```
<script>
        var x=Math.PI;
       var y=Math.sqrt(16);
</script>
```

Continue..

- Math object has some properties which are,

Properties	Description
E	Returns Euler's number(approx.2.718)
LN2	Returns the natural logarithm of 2 (approx.0.693)
LN10	Returns the natural logarithm of 10 (approx.2.302)
LOG2E	Returns the base-2 logarithm of E (approx.1.442)
LOG10E	Returns the base-10 logarithm of E (approx.0.434)
PI	Returns PI(approx.3.14)
SQRT1_2	Returns square root of ½
SQRT2	Returns square root of 2

Math Methods

- Here is a list of the methods associated with Math object and their description

Method Name	Description
exp()	Returns EN, where N is the argument, and E is Euler's constant, the base of the natural logarithm.
floor()	Returns the largest integer less than or equal to a number.
max()	Returns the largest of zero or more numbers.
min()	Returns the smallest of zero or more numbers.
Sqrt()	Returns the square root of a number.
tan()	Returns the tangent of a number.

Document Object Model (DOM)

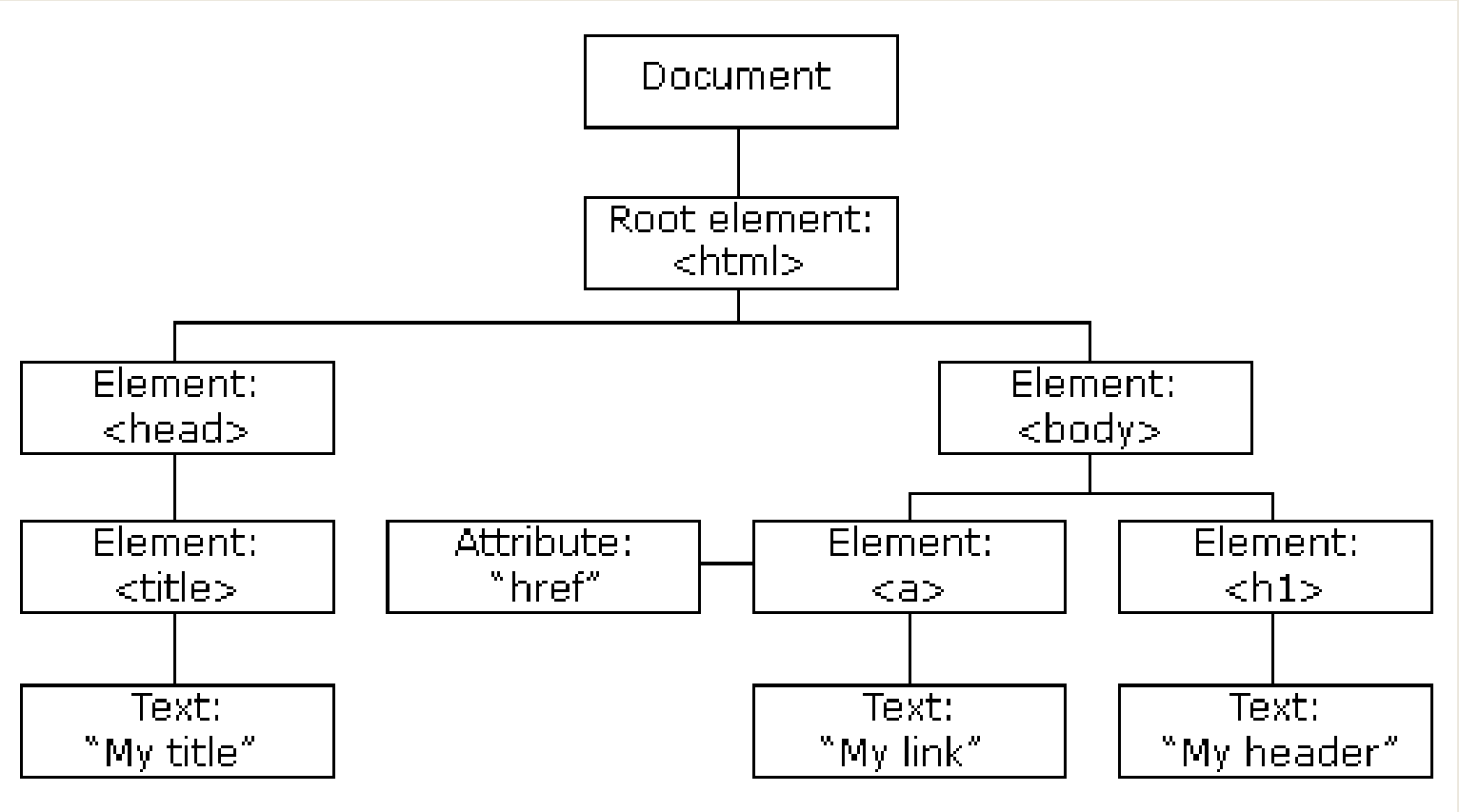
- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM defines a standard for accessing documents:
- "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."
- The W3C DOM standard is separated into 3 different parts:
- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents.

DOM

- The window object is the primary point from which most other objects come.
- From the current window object access and control can be given to most aspects of the browser features and the HTML document.
- When we write :
- document.write(“Hello World”);
- We are actually writing :
- window.document.write(“Hello World”);
- The window is just there by default

Continue..

- Here is a simple hierarchy of a few important objects –



Continue..

- There are several DOMs in existence. The following sections explain each of these DOMs in detail and describe how you can use them to access and modify document content.
- **The Legacy DOM** – This is the model which was introduced in early versions of JavaScript language. It is well supported by all browsers, but allows access only to certain key portions of documents, such as forms, form elements, and images.
- **The W3C DOM** – This document object model allows access and modification of all document content and is standardized by the World Wide Web Consortium (W3C). This model is supported by almost all the modern browsers.
- **The IE4 DOM** – This document object model was introduced in Version 4 of Microsoft's Internet Explorer browser. IE 5 and later versions include support for most basic W3C DOM features.

Document Object Properties

Method/Property	Description
Value	Return value of specified field
Write(“string”)	Write the specified string in the document
Writeln(“string”)	Write the specified string in the document with new line character at end
getElementById()	Returns the element having specified id value
getElementByName()	Returns all the element having specified name value
getElementByTagName()	Returns all the element having specified tag name
getElementByClassName()	Returns all the element having specified class name

getElementById()

- When we suppose to get the reference of the element from HTML in JavaScript using id specified in the HTML we can use this method.
- Refer Example

getElementsByName()

- When we suppose to get the reference of the elements from HTML in JavaScript using name specified in the HTML we can use this method.
- It will return the array of elements with the provided name.
- Refer Example

getElementsByTagName()

- When we suppose to get the reference of the elements from HTML in JavaScript using name of the tag specified in the HTML we can use this method.
- It will return the array of elements with the provided tag name.
- Refer Example

JavaScript - Form Validation

- Form validation normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button.
- If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information.
- This was really a lengthy process which used to put a lot of burden on the server.
- JavaScript provides a way to validate form's data on the client's computer before sending it to the web server. Form validation generally performs two functions.
- **Basic Validation** − First of all, the form must be checked to make sure all the mandatory fields are filled in. It would require just a loop through each field in the form and check for data.
- **Data Format Validation** − Secondly, the data that is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data.

Data Validation

- Data validation is the process of ensuring that user input is clean, correct, and useful.
- **Typical validation tasks are:**
- has the user filled in all required fields?
- has the user entered a valid date?
- has the user entered text in a numeric field?
- Most often, the purpose of data validation is to ensure correct user input.
- Validation can be defined by many different methods, and deployed in many different ways.
- **Server side validation** is performed by a web server, after input has been sent to the server.
- **Client side validation** is performed by a web browser, before input is sent to a web server.

JavaScript Form Validation

- HTML form validation can be done by JavaScript.
- If a form field (fname) is empty, this function alerts a message, and returns false, to prevent the form from being submitted:
- The function can be called when the form is submitted

Automatic HTML Form Validation

- HTML form validation can be performed automatically by the browser:
- If a form field (fname) is empty, the required attribute prevents this form from being submitted:

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

<form action="#" method="post">
  <input type="text" name="fname" required>
  <input type="submit" value="Submit">
</form>

<p>If you click submit, without filling out the text field,
your browser will display an error message.</p>

</body>
</html>
```

JavaScript email validation

- We can validate the email by the help of JavaScript.
- There are many criteria that need to be follow to validate the email id such as:
- email id must contain the @ and . character
- There must be at least one character before and after the @.
- There must be at least two characters after . (dot).

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.

- 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.

onclick Event Type

- This is the most frequently used event type which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type.

- Refer Example

onsubmit Event Type

- onsubmit is an event that occurs when you try to submit a form. You can put your form validation against this event type.
- Refer Example

onmouseover and onmouseout

- These two event types will help you create nice effects with images or even with text as well. The **onmouseover** event triggers when you bring your mouse over any element and the **onmouseout** triggers when you move your mouse out from that element.
- Refer Example

HTML 5 Standard Events

- The standard HTML 5 events are listed here for your reference. Here script indicates a Javascript function to be executed against that event.

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

JSON

- JSON — short for JavaScript Object Notation — is a format for sharing data. As its name suggests, JSON is derived from the JavaScript programming language, but it's available for use by many languages including Python, Ruby, PHP, and Java.
- JSON uses the .json extension when it stands alone. When it's defined in another file format (as in .html), it can appear inside of quotes as a JSON string, or it can be an object assigned to a variable. This format is easy to transmit between web server and client or browser.
- Very readable and lightweight, JSON offers a good alternative to XML and requires much less formatting.

Continue..

- A JSON object is a key-value data format that is typically rendered in curly braces. When you're working with JSON, you'll likely see JSON objects in a .json file, but they can also exist as a JSON object or string within the context of a program.
- A JSON object looks something like this:
- {
- "first_name" : "Sammy",
- "last_name" : "Shark",
- "location" : "Ocean",
- "online" : true,
- "followers" : 987
- }
- JSON keys are on the left side of the colon. They need to be wrapped in double quotation marks, as in "key", and can be any valid string. Within each object, keys need to be unique. These key strings can include whitespaces, as in "first name", but that can make it harder to access when you're programming, so it's best to use underscores, as in "first_name".

Advantages of JSON

1. JSON is Faster:

- JSON syntax is very easy to use. We have to use only -> as a syntax which provides us an easy parsing of the data and faster execution of the data. Since its syntax is very small and light weighted that's the reason that it executes the response in the faster way.

2. Schema Support:

- It has the wide range of supported browser compatibility with the operating systems so the applications made with the coding of JSON doesn't require much effort to make it all browser compatible. During development, the developer thinks for the different browsers but JSON provides that functionality.

3. Server Parsing:

- On the server side parsing is the important part that developers want if the parsing will be fast on the server side then the only user can get the fast response of their response so in this case JSON server-side parsing is the strong point that indicates us to use the JSON on the server side.

4. Tool for sharing data:

- JSON is the best tool for the sharing data of any size even audio, video etc. This is because JSON stores the data in the arrays so data transfer makes easier. For this reason, JSON is a superior file format for web APIs and for web development.