Unit V

Java Script: Introduction to Client Side Scripting, Introduction to Java Script, Comments, Variables in JS, Global Variables, Data types, Operators in JS, Conditions Statements (If, If Else, Switch), Java Script Loops (For Loop, While Loop, Do While Loop), JS Popup Boxes (Alert, Prompt, Confirm), JS Events, JS Arrays, JS Objects.

The client-side environment used to run scripts is usually a browser. The processing takes place on the end users computer. The source code is transferred from the web server to the users computer over the internet and run directly in the browser.

BASIS FOR COMPARISON	SERVER-SIDE SCRIPTING	CLIENT-SIDE SCRIPTING
Basic	Works in the back end which could not be visible at the client end.	Works at the front end and script are visible among the users.
Processing	Requires server interaction.	Does not need interaction with the server.
Languages involved	PHP, ASP.net, Ruby on Rails, ColdFusion, Python, etcetera.	HTML, CSS, JavaScript, etc.
Affect	Could effectively customize the web pages and provide dynamic websites.	Can reduce the load to the server.
Security	Relatively secure.	Insecure

Comment

JavaScript comments are meaningful way to deliver message. It is used to add information about the code, warnings or suggestions so that end user can easily interpret the code.The JavaScript comment is ignored by the JavaScript engine i.e. embedded in the browser.

There are mainly two advantages of JavaScript comments.

1. **To make code easy to understand** It can be used to elaborate the code so that end user can easily understand the code.
2. **To avoid the unnecessary code** It can also be used to avoid the code being executed. Sometimes, we add the code to perform some action. But after sometime, there may be need to disable the code. In such case, it is better to use comments.

Types of JavaScript Comments

There are two types of comments in JavaScript.

1. Single-line Comment
2. Multi-line Comment

Single line Comment

It is represented by double forward slashes (//). It can be used before and after the statement.

Multi line Comment

It can be used to add single as well as multi line comments. So, it is more convenient.It is represented by forward slash with asterisk then asterisk with forward slash.

// It is single line comment	/* It is multi line comment. It will not be displayed */

Varibles: **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers)

1. Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.
2. After first letter we can use digits (0 to 9), for example value1.
3. JavaScript variables are case sensitive, for example x and X are different variables.

<script>
var x = 10;
var y = 20;
var z=x+y;
document.write(z);
</script>	

local variable : A JavaScript local variable is declared inside block or function. It is accessible within the function or block only.

global variable : A JavaScript global variable is accessible from any function. A variable i.e. declared outside the function or declared with window object is known as global variable.

<script>
function abc(){
var x=10;//local variable
}
</script>
	var data=200;//gloabal variable
	function a(){
	document.writeln(data);
	}

Data Types

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:

1. var a=40;//holding number
2. var b="Rahul";//holding string

Primitive data types

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

Non-primitive data types

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

Operators

JavaScript operators are symbols that are used to perform operations on operands. Operators are used to perform specific mathematical and logical computations on operands. In other words, we can say that an operator operates the operands. In JavaScript operators are used for compare values, perform arithmetic operations etc.

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Logical Operators
4. Assignment Operators
5. Special Operators

Arithmetic Operators : Arithmetic operators are used to perform arithmetic operations on the operands. (+ - * / %).

Comparison Operators : The JavaScript comparison operator compares the two operands. (< > == != <= > =)

Logical Operators : The following operators are known as JavaScript logical operators. (&& || !)

Assignment Operators : (=)

Special Operators : The following operators are known as JavaScript special operators.

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
new	creates an instance (object)
void	it discards the expression's return value.

If-else

The **JavaScript if-else statement** is used *to execute the code whether condition is true or false*. There are two forms of if statement in JavaScript.

1. If Statement
2. If else statement

If statement

It evaluates the content only if expression is true. Here a JavaScript expression is evaluated. If the resulting value is true, the given statement(s) are executed. If the expression is false, then no statement would be not executed. Most of the times, you will use comparison operators while making decisions.

If...else Statement

It evaluates the content whether condition is true of false. Here JavaScript expression is evaluated. If the resulting value is true, the given statement(s) in the 'if' block, are executed. If the expression is false, then the given statement(s) in the else block are executed.

```
<script>
var a=20;
if(a>10){
document.write("value of a is greater than 10");
}
</script>
```

```
<script>
var a=20;
if(a%2==0){
document.write("a is even number");
}
else{
document.write("a is odd number");
}
</script>
```

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.

The objective of a **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.Break statements play a major role in switch-case statements.

```
<script>
var grade='B';
var result;
switch(grade){
```

case 'A':
result="A Grade";
break;
case 'B':
result="B Grade";
break;
case 'C':
result="C Grade";
break;
default:
result="No Grade";
}
document.write(result);
</script>	

Loop :

While writing a program, you may encounter a situation where you need to perform an action over and over again. In such situations, you would need to write loop statements to reduce the number of lines.

JavaScript supports all the necessary loops to ease down the pressure of programming.

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.

1. for loop
2. while loop
3. do-while loop

while loop

The **JavaScript while loop** *iterates the elements for the infinite number of times*. It should be used if number of iteration is not known.

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.

do while loop

The **JavaScript do while loop** *iterates the elements for the infinite number of times* like while loop. But, code is *executed at least* once whether condition is true or false.

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**.

```
<script>
var i=1;
while (i<=10)
{
document.write(i + "<br/>");
i++;
}
</script>
```

```
<script>
var i=100;
do{
document.write(i + "<br/>");
i++;
}while (i<=10);
</script>
```

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 '**for**' loop is the most compact form of looping. It includes the following three important parts −

- The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
- The **iteration statement** where you can increase or decrease your counter.

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

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. JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Alert Box

An alert box is often used if you want to make sure information comes through to the user.When an alert box pops up, the user will have to click "OK" to proceed.

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.

Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.

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 Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

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**.

<html> <body> JavaScript Alert	<html> <body> JavaScript Confirm Box

```
<button onclick="myFunction()">Try
it</button>

<script>
function myFunction() {
  alert("I am an alert box!");
}
</script>

</body>
</html>
```

```
<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
  var txt;
  if (confirm("Press a button!")) {
    document.write( "You pressed OK!");
  } else {
    document.write( txt = "You pressed Cancel!");
  }

}
</script>

</body>
</html>
```

```
<html>
<body>

<h2>JavaScript Prompt</h2>

<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
  var txt;
  var person = prompt("Please enter your name:", "Harry Potter");
  if (person == null || person == "") {
document.write( "User cancelled the prompt.");
  } else {
    document.write( "Hello " + person + "! How are you today?");
  }
}
</script>

</body>
</html>
```

Array

JavaScript array is an object that represents a collection of similar type of elements. 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.

The **Array** parameter is a list of strings or integers. When you specify a single numeric parameter with the Array constructor, you specify the initial length of the array.

There are 3 ways to construct array in JavaScript

1. By array literal
2. By creating instance of Array directly (using new keyword)

Array literal

The syntax of creating array using array literal is given below:

var arrayname=[value1,value2.....valueN];

As you can see, values are contained inside [ ] and separated by , (comma).Let's see the simple example of creating and using array in JavaScript. The .length property returns the length of an array.

```
<script>
var emp=["Sonoo","Vimal","Ratan"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
```

Array directly (new keyword)

The syntax of creating array directly is given below:

var arrayname=new Array();

Here, **new keyword** is used to create instance of array.Let's see the example of creating array directly.

```
<script>
var i;
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";

for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

Objects

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create 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.

```
<script>
emp={id:102,name:"Shyam Kumar",salary:40000}
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

Events

The change in the state of an object is known as an **Event**. In html, there are various events which represents that some activity is performed by the user or by the browser. When javascript code is included in HTML, js react over these events and allow the execution. This process of reacting over the events is called **Event Handling**. Thus, js handles the HTML events via **Event Handlers**.

For example, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

Mouse events:

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element

Keyboard events:

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

Form events:

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form

```
<html>
<head> Javascript Events </head>
<body>
<script language="Javascript" type="text/Javascript">
   <!--
   function clickevent()
   {
      document.write("This is JavaTpoint");
   }
   //-->
</script>
<form>
<input type="button" onclick="clickevent()" value="click"/>
</form>
</body>
</html>
```