18BCS53C – HTML & JavaScript
UNIT III

Introduction to JavaScript: - Core JavaScript - Client-Side JavaScript - Lexical Structure : Character Set - Comments - Literals Identifiers and Reserved Words Optional Semicolons types, values, and variables-.Expressions and Operators. Statements: Expression Statements - Compound and Empty Statements - Declaration Statements Conditionals - Loops - Jumps -Objects - Creating Objects - Querying and Setting Properties -Deleting Properties -Testing Properties - Enumerating Properties - Object Attributes - - Object Methods Arrays:- Creating Arrays - Reading and Writing Array Elements - -Adding and Deleting Array Elements - Iterating Arrays

---

What is JavaScript? Complete Introduction with Hello World! Example

- What is JavaScript?
- Javascript History
- How to Run JavaScript?
- Tools You Need
- A Simple JavaScript Program
- Hello World! Example

What is JavaScript?

- JavaScript is a very powerful client-side scripting language.
- JavaScript is used mainly for enhancing the interaction of a user with the webpage.
- JavaScript is also being used widely in game development and Mobile application development.

JavaScript History

- JavaScript was developed by Brendan Eich in 1995, which appeared in Netscape, a popular browser of that time.
- The language was initially called LiveScript and was later renamed JavaScript.
- There are many programmers who think that JavaScript and Java are the same.
- In fact, JavaScript and Java are very much unrelated.
- Java is a very complex programming language whereas JavaScript is only a scripting language.
- The syntax of JavaScript is mostly influenced by the programming language C.

How to Run JavaScript?

- Being a scripting language, JavaScript cannot run on its own.
- In fact, the browser is responsible for running JavaScript code.
- When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it is up to the browser to execute it.
- The main advantage of JavaScript is that all modern web browsers support JavaScript.
- JavaScript runs on any operating system including Windows, Linux or Mac.
- JavaScript overcomes the main disadvantages of VBScript which is limited to just IE and Windows.

A Simple JavaScript Program

- All JavaScript code should be within **<script> tags** (<script> and </script>).
- Other client-side scripting languages (Example: VBScript), is highly recommended to specify the scripting language. you use.
- The use of type attribute within the <script> tag and set its value to text/javascript like this:
    - <script type="text/javascript">

Hello World Example:

```
<html>
<head>
   <title>My First JavaScript code!!!</title>
   <script type="text/javascript">
      alert("Hello World!");
   </script>
</head>
<body>
</body>
</html>
```

JavaScript Variable: Declare, Assign a Value with Example

- Variables are used to **store values** (name = "John") **or expressions** (sum = x + y).

Declare Variables in JavaScript

- Before using a variable, you first need to declare it. You have to use the keyword**var** to declare a variable like this:
- var name;

Assign a Value to the Variable

- You can assign a value to the variable either while declaring the variable or after declaring the variable.
- var name = "John";
- OR
- var name;
- name = "John";

Naming Variables

- It is a good programming practice to give descriptive and meaningful names to the variables.
- Variable names should start with a letter and they are case sensitive. Hence the variables student name and studentName are different because the letter n in a name is different (n and N).

```
<html>
<head>
<title>Variables!!!</title>
<script type="text/javascript">
var one = 22;
var two = 3;
```

```
var add = one + two;
var minus = one - two;
var multiply = one * two;
var divide = one/two;
   document.write("First No: = " + one + "<br />Second No: = " + two + " <br />");
   document.write(one + " + " + two + " = " + add + "<br/>");
   document.write(one + " - " + two + " = " + minus + "<br/>");
   document.write(one + " * " + two + " = " + multiply + "<br/>");
   document.write(one + " / " + two + " = " + divide + "<br/>");
</script>
</head>
<body>
</body>
</html>
```

JavaScript Array Methods: Create with Example
What is an Array?

- An array is an object that can store a **collection of items**.
- Arrays become really useful when you need to store large amounts of data of the same type.
- The items in an array can be referred by its **index number** and the index of the first element of an array is zero.

JavaScript Create Array

- You can create an array in JavaScript as given below.

```
var students = ["John", "Ann", "Kevin"];
```

Here, you are initializing your array as and when it is created with values “John”, “Ann” and “Kevin”.The index of “John”, “Ann” and “Kevin” is 0, 1 and 2 respectively. If you want to add more elements to the students array, you can do it like this:

```
students[3] = "Emma";
students[4] = "Rose";
```

You can also create an array using Array constructor like this:

```
var students = new Array("John", "Ann", "Kevin");
```

OR

```
var students = new Array();

students[0] = "John";
students[1] = "Ann";
students[2] = "Kevin";
```

JavaScript Array Methods

- The Array object has many properties and methods which help developers to handle arrays easily and efficiently.
- You can get the value of a property by specifying arrayname.property and the output of a method by specifying arrayname.method().
    1. **length property** --> If you want to know the number of elements in an array, you can use the length property.

2. **prototype property** --> If you want to add new properties and methods, you can use the prototype property.
3. **reverse method** --> You can reverse the order of items in an array using a reverse method.
4. **sort method -->** You can sort the items in an array using sort method.
5. **pop method** --> You can remove the last item of an array using a pop method.
6. **shift method** --> You can remove the first item of an array using shift method.
7. **push method** --> You can add a value as the last item of the array.

```
<html>
<head>
   <title>Arrays!!!</title>
   <script type="text/javascript">
      var students = new Array("John", "Ann", "Aaron", "Edwin",
"Elizabeth");
      Array.prototype.displayItems=function()
      {
         for (i=0;i<this.length;i++)
         {
            document.write(this[i] + "<br />");
         }
      }
      document.write("students array<br />");
      students.displayItems();
      document.write("<br />The number of items in students array is " +
students.length + "<br />");
      document.write("<br />The SORTED students array<br />");
      students.sort();
      students.displayItems();
      document.write("<br />The REVERSED students array<br />");
      students.reverse();
      students.displayItems();
      document.write("<br />THE students array after REMOVING the LAST
item<br />");
      students.pop();
      students.displayItems();
      document.write("<br />THE students array after PUSH<br />");
      students.push("New Stuff");
      students.displayItems();
   </script>
</head>
<body>
</body>
</html>
```

For, While and Do While LOOP in JavaScript (with Example)

- How to use Loop?
- Different Types of Loops

- for loop
- while loop
- do…while loop

How to use Loop?

- Loops are useful when you have to execute the same lines of code repeatedly, for a specific number of times or as long as a specific condition is true.

Different Types of Loops

There are mainly four types of loops in JavaScript.

1. for loop
2. for/in a loop (explained later)
3. while loop
4. do…while loop

for loop

Syntax:\

```
For(statement1, statement2, statement3)
{
lines of code to be executed
}
```

1. The statement1 is executed first even before executing the looping code. So, this statement is normally used to assign values to variables that will be used inside the loop.
2. The statement2 is the condition to execute the loop.
3. The statement3 is executed every time after the looping code is executed.

```
<html>
<head>
  <script type="text/javascript">
    var students = new Array("John", "Ann", "Aaron", "Edwin", "Elizabeth");
    document.write("<b>Using for loops </b><br />");
    for (i=0;i<students.length;i++)
    {
    document.write(students[i] + "<br />");
    }
  </script>
</head>
<body>
</body>
</html>
```

while loop

Syntax:

```
while(condition)
{
lines of code to be executed
}
```

1. The “while loop” is executed as long as the specified condition is true.

2. Inside the while loop, you should include the statement that will end the loop at some point of time.
3. Otherwise, your loop will never end and your browser may crash.

```
<html>
<head>
  <script type="text/javascript">
    document.write("<b>Using while loops </b><br />");
    var i = 0, j = 1, k;
    document.write("Fibonacci series less than 40<br />");
    while(i<40)
    {
      document.write(i + "<br />");
      k = i+j;
      i = j;
      j = k;
    }
  </script>
</head>
<body>
</body>
</html>
```

do…while loop

Syntax:

```
do
{
block of code to be executed
} while (condition)
```

- The do…while loop is very similar to while loop.
- The only difference is that in do…while loop, the block of code gets executed once even before checking the condition.

```
<html>
<head>
  <script type="text/javascript">
    document.write("<b>Using do...while loops </b><br />");
    var i = 2;
    document.write("Even numbers less than 20<br />");
    do
    {
      document.write(i + "<br />");
      i = i + 2;
    }while(i<20)
  </script>
</head>
<body>
</body>
</html>
```

JavaScript Conditional Statements: IF, Else, Else IF (Example)

- How to use Conditional Statements
- Different Types of Conditional Statements
- If statement
- If…Else statement
- If…Else If…Else statement
- Switch statement

How to use Conditional Statements

- Conditional statements are used to decide the flow of execution based on different conditions.
- If a condition is true, you can perform one action and if the condition is false, you can perform another action.

Different Types of Conditional Statements

There are mainly three types of conditional statements in JavaScript.

1. If statement
2. If…Else statement
3. If…Else If…Else statement

If statement

Syntax:

```
 if (condition)
{
Lines of code to be executed if condition is true
}
```

```
<html>
<head>
   <title>IF Statments!!!</title>
   <script type="text/javascript">
      var age = prompt("Please enter your age");
      if(age>=18)
      document.write("You are an adult <br />");
      if(age<18)
      document.write("You are NOT an adult <br />");
   </script>
</head>
<body>
</body>
</html>
```

If…Else statement

Syntax:

```
if (condition)
{
lines of code to be executed if the condition is true
```

```
}
Else
{
lines of code to be executed if the condition is true
}
    <html>
    <head>
      <title>If...Else Statments!!!</title>
      <script type="text/javascript">
        // Get the current hours
        var hours = new Date().getHours();
        if(hours<12)
        document.write("Good Morning!!!<br />");
        else
        document.write("Good Afternoon!!!<br />");
      </script>
    </head>
    <body>
    </body>
    </html>
```

If…Else If…Else statement

Syntax:

```
if (condition1)
{
lines of code to be executed if condition1 is true
}
else if(condition2)
{
lines of code to be executed if condition2 is true
}
else
{
lines of code to be executed if condition1 is false and condition2 is false
}

    <html>
    <head>
      <script type="text/javascript">
        var one = prompt("Enter the first number");
        var two = prompt("Enter the second number");
        one = parseInt(one);
        two = parseInt(two);
        if (one == two)
          document.write(one + " is equal to " + two + ".");
        else if (one<two)
          document.write(one + " is less than " + two + ".");
        else
```

```
            document.write(one + " is greater than " + two + ".");
        </script>
    </head>
    <body>
    </body>
    </html>
```

JavaScript Switch Statement

- The switch statement is used to perform different actions based on different conditions.
- Use the switch statement to select one of many code blocks to be executed.

Syntax

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

This is how it works:

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.
- If there is no match, the default code block is executed.

Example

The getDay() method returns the weekday as a number between 0 to 6.
(Sunday=0, Monday=1, Tuesday=2)
This example uses the weekday number to calculate the weekday name:

```
switch (new Date().getDay()) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
     day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
```

```
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
}
```

The result of day will be:
Monday

The break Keyword

- When JavaScript reaches a break keyword, it breaks out of the switch block.
- This will stop the execution of inside the block.
- It is not necessary to break the last case in a switch block.
- The block breaks (ends) there anyway.

The default Keyword

- The default keyword specifies the code to run if there is no case match:
- Example
    - The getDay() method returns the weekday as a number between 0 and 6.
      If today is neither Saturday (6) nor Sunday (0), write a default message:

```
switch (new Date().getDay()) {
  case 6:
    text = "Today is Saturday";
    break;
  case 0:
    text = "Today is Sunday";
    break;
  default:
    text = "Looking forward to the Weekend";
}
```

The result of text will be:
Looking forward to the Weekend

The default case does not have to be the last case in a switch block:
Example

```
switch (new Date().getDay()) {
  default:
    text = "Looking forward to the Weekend";
    break;
  case 6:
    text = "Today is Saturday";
    break;
```

```
  case 0:
    text = "Today is Sunday";
}
```

- If default is not the last case in the switch block, the case should be end with a break statement.

Switching Details

- If multiple cases matches a case value, the **first** case is selected.
- If no matching cases are found, the program continues to the **default** label.
- If no default label is found, the program continues to the statement(s) **after the switch**.

Strict Comparison

- Switch cases use **strict** comparison (===).
- The values must be of the same type to match.
- A strict comparison can only be true if the operands are of the same type.

JavaScript Objects

- Real Life Objects, Properties, and Methods
- In real life, a car is an **object**.
- A car has **properties** like weight and color, and **methods** like start and stop:
- All cars have the same **properties**, but the property **values** differ from car to car.
- All cars have the same **methods**, but the methods are performed **at different times**.
- This code assigns a **simple value** (Fiat) to a **variable** named car:

```
var car = "Fiat";
```

- Objects are variables too. But objects can contain many values.
- This code assigns **many values** (Fiat, 500, white) to a **variable** named car:

```
var car = {type:"Fiat", model:"500", color:"white"};
```

- The values are written as **name:value** pairs (name and value separated by a colon).
- JavaScript objects are containers for **named values** called properties or methods.

Object Definition

- To define (and create) a JavaScript object with an object literal
- Example

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

- Spaces and line breaks are not important. An object definition can span multiple lines:
- Example

```
var person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
```

Object Properties

- The **name:values** pairs in JavaScript objects are called **properties**:
Property	**Property Value**
firstName	John
lastName	Doe
Age	50
eyeColor	blue

Accessing Object Properties

- You can access object properties in two ways:

  objectName.propertyName

  or

  objectName["propertyName"]

  Example1

  person.lastName;

  Example2

  person["lastName"];

Object Methods

- Objects can also have **methods**.
- Methods are **actions** that can be performed on objects.
- Methods are stored in properties as **function definitions**.
Property	**Property Value**
firstName	John
lastName	Doe
Age	50
eyeColor	blue
fullName	function() {return this.firstName + " " + this.lastName;}
- A method is a function stored as a property.
- Example

```
var person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

this Keyword

- The **this** Keyword
- In a function definition, this refers to the "owner" of the function.
- In the example above, this is the **person object** that "owns" the fullName function.
- In other words, this.firstName means the firstName property of **this object**.

Accessing Object Methods

- You access an object method with the following syntax:

  objectName.methodName()

  Example

  name = person.fullName();

- If you access a method **without** the () parentheses, it will return the **function definition**:

  Example

  name = person.fullName;

- Do Not Declare Strings, Numbers, and Booleans as Objects
- When a JavaScript variable is declared with the keyword "new", the variable is created as an object:

  var x = new String(); // Declares x as a String object

  var y = new Number(); // Declares y as a Number object

  var z = new Boolean(); // Declares z as a Boolean object

- Avoid String, Number, and Boolean objects. They complicate your code and slow down execution speed.

REFERENCES

1. www.w3schools.com
2. www.guru99.com/javascript
3. www.tutorialspoint.com/javascript

----------xxxxxxxxxxxx----------