TECH FEST 2022

"Coding for progress"

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 cannot run on its own. In fact, the browser is responsible for running JavaScript code.
- The main advantage of JavaScript is that all modern web browsers support JavaScript.

First Javascript Program

- You should place all your JavaScript code within <script> tags within the HTML document itself.
- This helps your browser distinguish your JavaScript code from the rest of the code.
- You have to use the type attribute within the <script> tag and set its value to text/javascript like this:
<script type="text/javascript">

home.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Javascript</title>
    <script type="text/javascript">
      alert("Hello World!");
    </script>
  </head>
  <body>
    <h1>Hello World</h1>
  </body>
</html>
```

Internal & External JavaScript

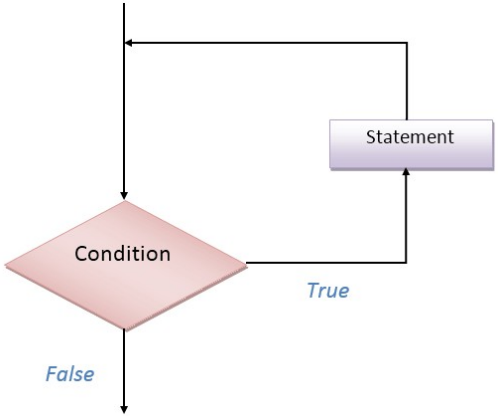
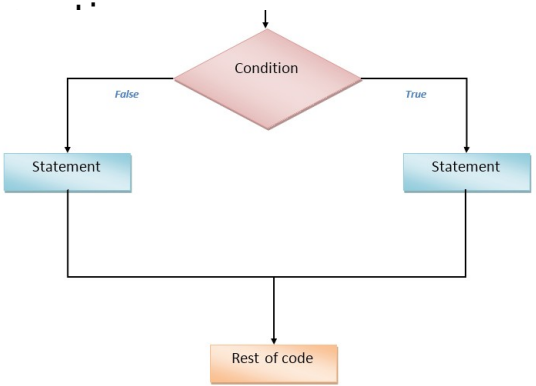
- You can use JavaScript code in two ways.
- You can either include the JavaScript code **internally within your HTML document** itself.
  Eg. <script type="text/javascript"></script>
- You can keep the JavaScript code in **a separate external file** and then point to that file from your HTML document.
  <script type="text/javascript" src="script.js">

JavaScript Variable

- Variables are used to **store values** (name = “Ram”) **or expressions** (sum = x + y).
- **Declare Variables in JavaScript:**
  var name;
- **Assign a Value to the Variable:**
  var name = "John";

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

  var age = 20;
</script>
```

JavaScript 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.
- You can access the items in an array by referring to its **index number** and the index of the first element of an array is zero.
- ```
  var students = ["John", "Ann", "Kevin"];
  students[3] = "Emma";
  students[4] = "Rose";
  console.log(students);
  console.log(students[0]);
  ```
  ```

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().

- **length property** –> If you want to know the number of elements in an array, you can use the length property.
- **push method** –> You can add a value as the last item of the array.
- **pop method** –> You can remove the last item of an array using a pop method.
- **shift method** –> You can remove the first item of an array using shift method.
- **sort method –>** You can sort the items in an array using sort method.
- **reverse method** –> You can reverse the order of items in an array using a reverse method.

Javascript Loops

- 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.
- Suppose you want to type a ‘Hello’ message 100 times in your webpage.
- Of course, you will have to copy and paste the san line 100 times. Instead, if you use loops, you can complete this task in just 3 or 4 lines.



Types of loops

1. For Loop
2. While Loop
3. Do While Loop

For loop

```
<script type="text/javascript">
  var students = [1, 2, 3];
  for (var i = 0; i < students.length; i++) {
    console.log(students[i]);
  }
</script>
```

> **Syntax:**

```
for(statement1; statement2; statment3){
        // lines of code to be executed
}
```

- 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.
- The statement2 is the condition to execute the loop.
- The statement3 is executed every time after the looping code is executed.

While Loop

- The “while loop” is executed as long as the specified condition is true.
- Inside the while loop, you should include the statement that will end the loop at some point of time. Otherwise, your loop will never end and your browser may crash.
- **Syntax:**

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

```
<script type="text/javascript">
  var i = 0;
  while (i < 5) {
    console.log(i);
    i++;
  }
</script>
```

Do while Loop

- 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.
- **Syntax:**

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

```
<script type="text/javascript">
  var i = 0;
  do {
    console.log(i);
    i++;
  } while (i < 5);
</script>
```

JavaScript 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 anothe



Types of Conditional Statements

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

If Statements

- You can use If statement if you want to check only a specific condition.
- **Syntax:**

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

```
<script type="text/javascript">
  var num = 1;

  if (num === 1) {
    console.log(num);
  }
</script>
```

If...Else statement

- You can use If….Else statement if you have to check two conditions and execute a different set of codes.
- **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 false
}
```

```
<script type="text/javascript">
  var num = 1;

  if (num === 1) {
    console.log("Number is 1");
  } else {
    console.log("Number is not 1");
  }
</script>
```

If…Else If…Else statement

- You can use If….Else If….Else statement if you want to check more than two conditions.

- **Syntax:**

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

```
<script type="text/javascript">
  var num = 1;

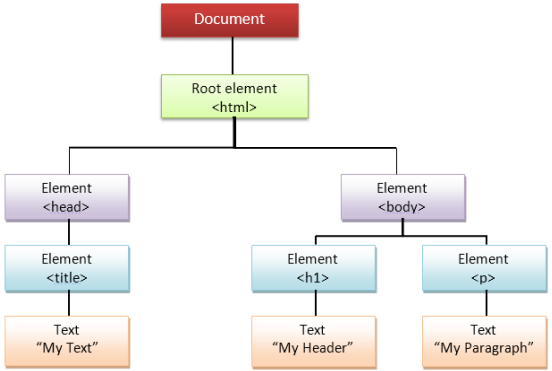
  if (num === 1) {
    console.log("Number is 1");
  } else if (num === 2) {
    console.log("Number is 2");
  } else {
    console.log("Number is nor 1 or 2");
  }
</script>
```

Javascript Functions

- JavaScript function is a block of code designed to perform a particular task.
- JavaScript function is executed when "something" calls it.
- Syntax:

```
function name(parameter1, parameter2, parameter3) {
    // code to be executed
}
```

```
<script>
  function add(num1, num2) {
    return num1 + num2;
  }

  var result = add(3, 4);
  console.log(result);
</script>
```

- Example:

```
function myFunction(p1, p2) {
      return p1 * p2;   // The function returns the product of p1 and p2
}
```

JavaScript Objects

- Objects are variables too. But objects can contain many values.
- For eg: A car has **properties** like weight and color, and **methods** like start and stop.
- ```
  var car = {
        type:"Fiat",
        model:"500",
        Color:"white"
  };
  ```
- The values are written as **name:value** pairs (name and value separated by a colon).

```
<script>
  var car = {
    type: "Fiat",
    model: "500",
    Color: "white"
  };

  console.log(car["type"]);
</script>
```
```

JavaScript DOM

- JavaScript can access all the elements in a webpage making use of Document Object Model (DOM).
- In fact, the web browser creates a DOM of the webpage when the page is loaded.
- The DOM model is created as a tree of objects like this:



How to use DOM

> Using DOM, JavaScript can perform multiple tasks.

> It can create new elements and attributes, change the existing elements and attributes and even remove existing elements and attributes.

- **getElementById**: To access elements and attributes whose id is set.
- **getElementsByTagName**: To access elements and attributes using tag name.
- **getElementByClass**: To access elements and attributes whose class is set.

Any Questions?

Learning Resources:
w3schools
codecademy