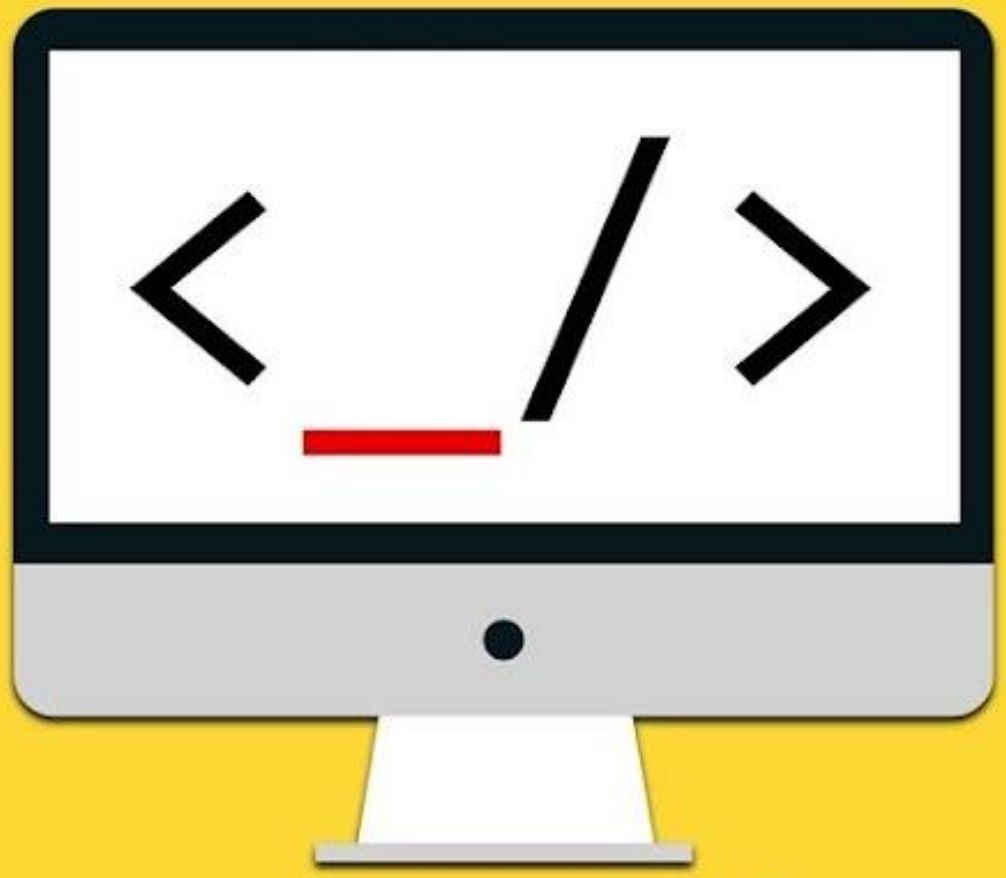


JavaScript

Introduction to JavaScript

JavaScript was created to add ‘life to webpages’. Just like HTML and CSS, JavaScript is written in plain text files with a .js extension.

JavaScript can be used to accomplish many useful operations on the frontend, like **validating forms**, **alerting users**, **storing temporary data**, and **performing calculations**—to name just a few.

JavaScript is a programming language, and like most programming languages, it has some basic constructs that we’ll look at.

A program in JavaScript is like a sequence of steps. Similar to how we give directions to a stranger, a computer needs detailed instructions, defined as steps, to accomplish any simple or complex action.

Writing JavaScript

In HTML, JavaScript code is inserted between <script> and </script> tags.

Example:

```
<script>

</script>
```

JavaScript in <head> or <body> or as an External File

You can place any number of scripts in an HTML document. Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

Adding Behaviour to Websites

Lets link our script file in our HTML. Just to make sure it works, write the following code into your script.js

e.g:

```
alert(“Hello World”);
```

Basic Arithmetic and Modulo Operators

Operators	Meaning	Example	Result
+	Addition	4+2	6
-	Subtraction	4-2	2
*	Multiplication	4*2	8
/	Division	4/2	2
%	Modulus operator to get remainder in integer division	5%2	1
+ +	Increment	A = 10; A+ +	11
− −	Decrement	A = 10; A− −	9

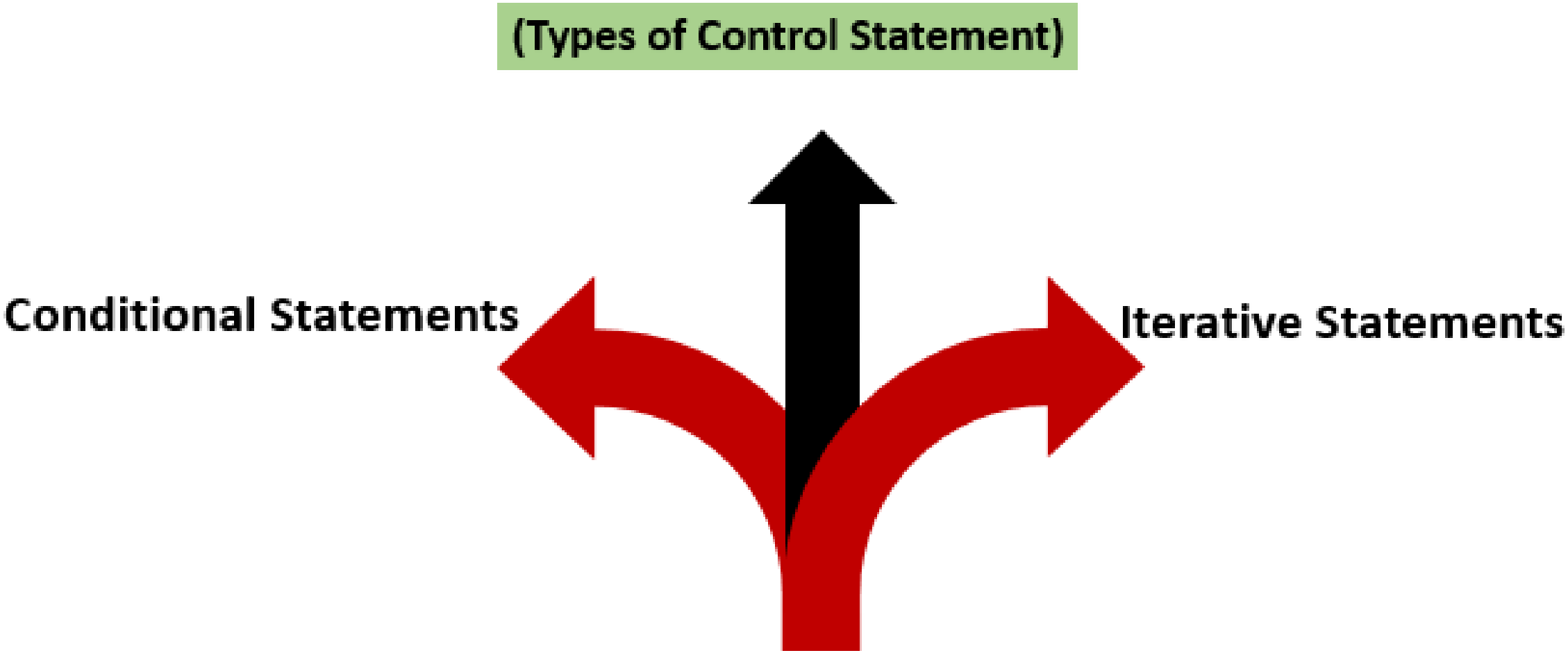
Increment & Decrement Operators

We use the increment & decrement operators to increase or decrease the variable's value by one. JavaScript uses the ++(increment) and –(decrement) to denote them

Syntax of Increment & Decrement Operator

Increment Operator ++x or x++, this is equal to x = x + 1;
Decrement Operator –x or x--, this is equal to x = x – 1;

Control Statements.



Control Statements

- **Conditional Statements:** based on an expression passed, a conditional statement makes a decision, which results in either YES or NO.

- **Iterative Statements (Loop):** These statements continue to repeat until the expression or condition is satisfied.

Conditional Statements

Conditional statements in a program decide the next step based on the result. They result in either True or False. The program moves to the next step if a condition is passed and true. However, if the condition is false, the program moves to another step. These statements are executed only once.

Following are the different types of Conditional Statements:

IF

When you want to check for a specific condition with the IF condition, the inner code block executes if the provided condition is satisfied.

If Statements

Syntax:

```
if (condition) { //code block to be executed if
condition is satisfied }
```

IF-ELSE

an extended version of IF. When you want to check a specific condition and two

Syntax:

If-Else Statements

```
if (condition)

{ // code to be executed of condition is true }

else { // code to be executed of condition is false }
```

As observed in an IF-ELSE statement, when the condition is satisfied, the first block of code executes, and if the condition isn’t satisfied, the second block of code executes.

Comparators and Equality

Operators	Meaning	Example	Result
<	Less than	5<2	False
>	Greater than	5>2	True
<=	Less than or equal to	5<=2	False
>=	Greater than or equal to	5>=2	True
==	Equal to	5==2	False
! =	Not equal to	5! =2	True
===	Equal value and same type	5 === 5	True
		5 === “5”	False
! ==	Not Equal value or Not same type	5 ! == 5	False
		5 ! == “5”	True

JavaScript Functions

JavaScript Functions

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).

Syntax

```
function nameOfFunction() {
  // code to be executed
}
//Function Call
nameOfFunction();
```

Function Invocation

- The code inside the function will execute when "something" **invokes** (calls) the function:
- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code

JavaScript Function Return

A parameter is a value that is passed when declaring a function.

When JavaScript reaches a return statement, the function will stop executing.

If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

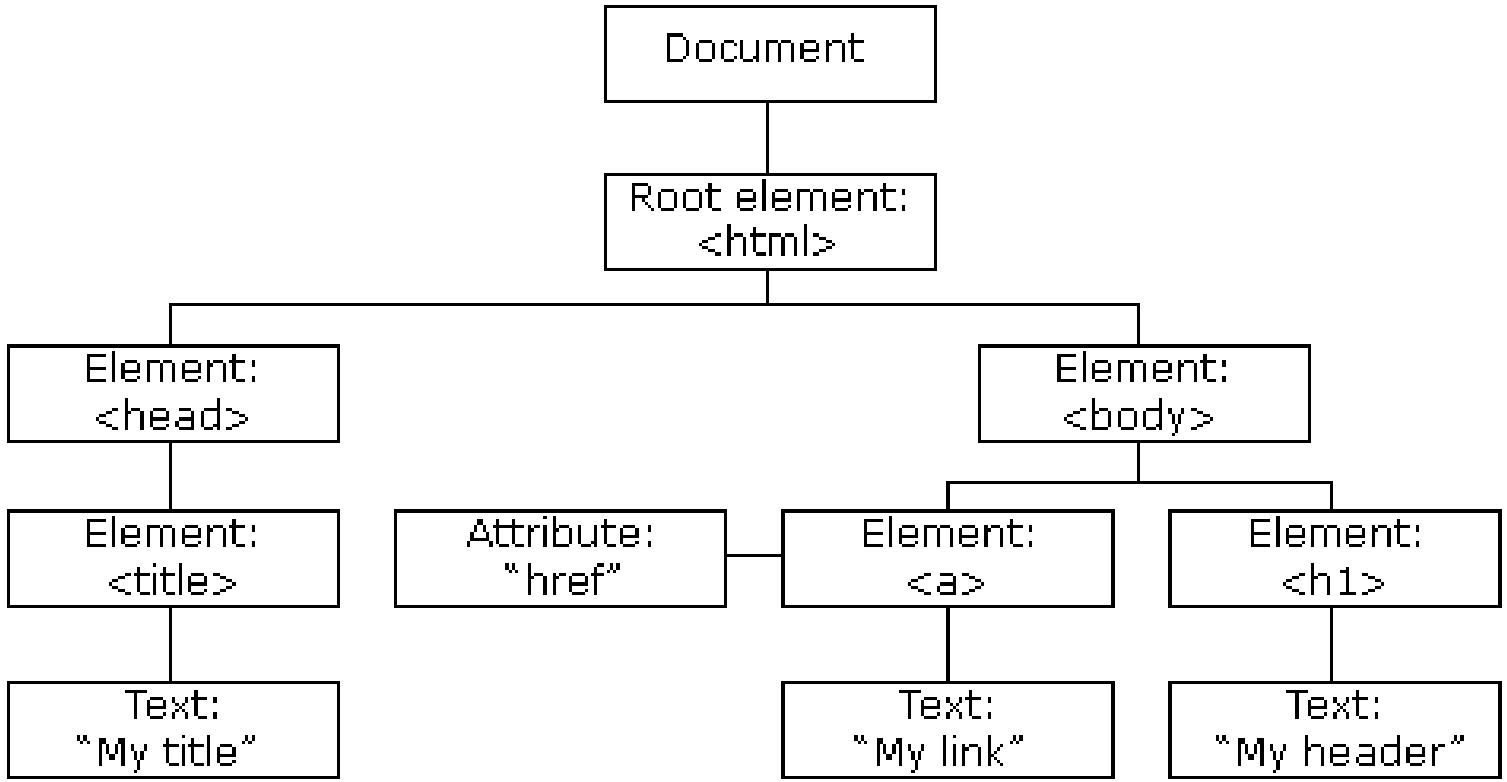
```
// Function is called, the return value will end up in x
let x = myFunction(4, 3);
Console.log(x)

function myFunction(a, b) {
// Function returns the product of a and b
  return a * b;
}
```

JavaScript Document Object Model (DOM)

The HTML DOM (Document Object Model)

With the HTML DOM, JavaScript can access and change all the elements of an HTML document.



The HTML DOM (Document Object Model)

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What we will Learn

- How to change the content of HTML elements
- How to change the style (CSS) of HTML elements
- How to react to HTML DOM events

Selecting HTML Elements with JavaScript using JavaScript DOM methods

Often, with JavaScript, you want to manipulate HTML elements.

To do so, you have to find the elements first. There are several ways to do this:

- Finding HTML elements by id
- Finding HTML elements by tag name
- Finding HTML elements by class name
- Finding HTML elements by CSS selectors

Examples

```
const element = document.getElementById("intro");

const element = document.getElementsByTagName("p");

const x = document.getElementsByClassName("intro");


const x = document.querySelector(".intro");
```

Changing HTML Elements

Property	Description
element.innerHTML	Changes the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element

The innerHTML Property

The easiest way to get the content of an element is by using the `innerHTML` property.

The `innerHTML` property is useful for getting or replacing the content of HTML elements.

Syntax

```
<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
```

Changing the Value of an Attribute

To change the value of an HTML attribute, use this syntax:

```
document.getElementById(id).attribute = new value
```

```
<img id="myImage" src="smiley.gif">

<script>

document.getElementById("myImage").src = "landscape.jpg";

</script>
```

Changing HTML Style

To change the style of an HTML element, use this syntax:

```
document.getElementById(id).style.property = new style
```

```
<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color = "blue";
</script>
```

HTML DOM Events

Reacting to Events

A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.

To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

`onclick=`*`JavaScript`*

Examples of HTML events:

- When a user clicks the mouse
- When a web page has loaded
- When an image has been loaded
- When the mouse moves over an element
- When an input field is changed
- When an HTML form is submitted
- When a user strokes a key

Reacting to Events

```
<h1 onclick="this.innerHTML = 'Ooops!'">Click on this
text!</h1>
```

In this example, a function is called from the event handler:

```
<h1 onclick="changeText(this)">Click on this text!</h1>

<script>
function changeText(id) {
  id.innerHTML = "Ooops!";
}
</script>
```

The onmouseover and onmouseout Events

```
<div onmouseover="mOver(this)" onmouseout="mOut(this)"
</div>

<script>
function mOver(obj) {
  obj.innerHTML = “Thank You";
}

Function mOut(obj){
      obj.innerHTML = “Mouse Over Me”
}
</script>
```

HTML DOM Event Listeners

The addEventListener() method

The addEventListener() method attaches an event handler to the specified element.

You can add many event handlers to one element.

Syntax

element.addEventListener(*event*, *function*);

The first parameter is the type of the event (like "click" or "mousedown" or any other HTML DOM Event.)

The second parameter is the function we want to call when the event occurs.

Add an Event Handler to an Element

Method 1:

```
element.addEventListener("click", function(){

alert("Hello World!");

 });
```

Method 2:

```
element.addEventListener("click", myFunction);

function myFunction() {
  alert ("Hello World!");
}
```

Add an Event Handler to an Element

Method 1:

```
element.addEventListener("click", function(){

alert("Hello World!");

 });
```

Method 2:

```
element.addEventListener("click", myFunction);

function myFunction() {
  alert ("Hello World!");
}
```

HOW TO

Validate input

```
<form action="" name="form" onsubmit="return(validate());">

        <input type="text" name="name" placeholder="name"><br>
        <span class="nameError"></span><br>
        <input type="text" name="email"
placeholder="email"><br>
        <input type="text" name="zip" placeholder="zip
code"><br>
        <input type="submit" value="submit">


    </form>
```

Validate input

```
function validate(){
    const name = document.form.name.value;
    const email = document.form.email.value;
    const zip = document.form.zip.value;

    if(name == ""){
        alert("Please provide your name");
        return false;
    }
    if(email == ""){
        alert("Please provide your email");
        return false;
    }
    if(zip == "" || isNaN(zip)){
        alert("Please provide a valid zip code");
        return false;
    }
    return (true);
}
```

Validate Email

```
//Validate Email
    const atpos = email.indexOf("@");
    const dotPos = email.lastIndexOf(".");


if(atpos < 1 || (dotPos - atpos < 2)){
        alert("Please enter valid email address");
        return false;

}
```

createElement() method

In an HTML document, the **document.createElement()** method creates the HTML element specified by *tagName*

Syntax:
createElement(tagname);

createElement() method

```
// create a new div element
const newDiv = document.createElement("div");

// and give it some content
const newContent = document.createTextNode("Hi there and greetings!");

// add the text node to the newly created div
newDiv.appendChild(newContent);

// add the newly created element and its content into the DOM
const currentDiv = document.getElementById("div1");
document.body.insertBefore(newDiv, currentDiv);
```