JavaScript

JavaScript is one of the **3 languages** all web developers **must** learn:

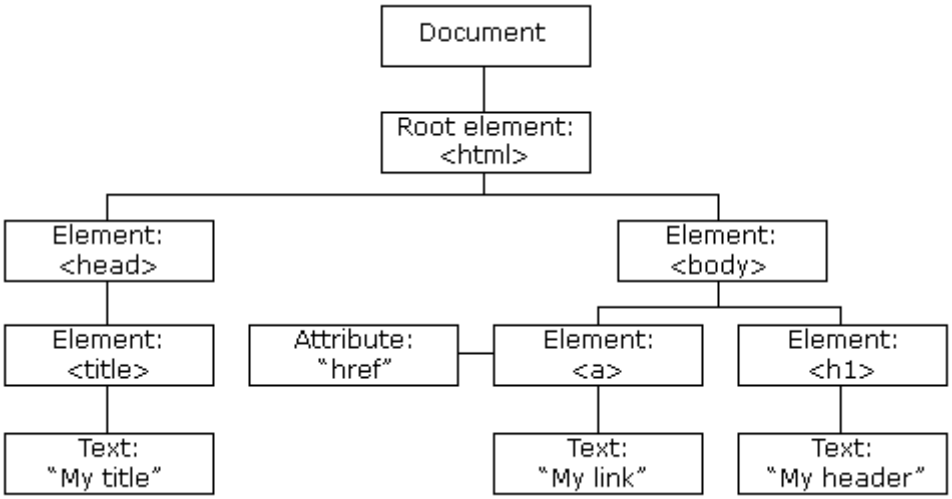
1. **HTML** to define the content/structure of web pages
2. **CSS** to specify the layout/style of web pages
3. **JavaScript** to program the behavior of web pages

JavaScript HTML DOM

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

The HTML DOM (Document Object Model):

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page. The **HTML DOM** model is constructed as a tree of **Objects**:



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 is the 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

What is the HTML DOM?

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as **objects**
- The **properties** of all HTML elements
- The **methods** to access all HTML elements
- The **events** for all HTML elements

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

JavaScript - HTML DOM Methods

The HTML DOM can be accessed with JavaScript (and with other programming languages).

In the DOM, all HTML elements are defined as **objects**.

HTML DOM properties are **values** of HTML Elements that you can set or change (like changing the content of an HTML element).

HTML DOM methods are **actions** you can perform (like add or deleting an HTML element).

The <script> Tag:

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

```
<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
```

Example: The following example changes the content of the <p> element with id="demo".

getElementById is a **method**, while innerHTML is a **property**.

The most common way to access an HTML element is to use the id of the element.
The getElementById method used id="demo" to find the element.

- The HTML document contains a `<p>` element with `id="demo"`
- We use the HTML DOM to get the element with `id="demo"`
- A JavaScript changes the content (`innerHTML`) of that element to "New text!"

```
<body>
<p id="demo">A Paragraph</p>

<script>
document.getElementById("demo").innerHTML = "New text!
</body>
```

JavaScript HTML DOM Document

The HTML DOM document object is the owner of all other objects in your web page.The document object represents your web page. If you want to access any element in an HTML page, you always start with accessing the document object.

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Finding HTML Element by Id

The easiest way to find an HTML element in the DOM, is by using the element id. This example finds the element with `id="intro"`:

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

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

Finding HTML Elements by Tag Name

This example finds all `<p>` elements:

```
<p>Hello</p>

<script>
document.getElementsByTagName("p")[0].innerHTML = "Hello  World!";
</script>
```

Finding HTML Elements by Class Name

If you want to find all HTML elements with the same class name, use `getElementsByClassName()`. This example returns a list of all elements with `class="intro"`.

```
<p class="test"></p>

<script>
document.getElementsByClassName("test")[0].innerHTML = "Hello";
</script>
```

Changing HTML Content

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

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

document.getElementById(*id*).innerHTML = *new HTML*

This example changes the content of a `<p>` element:

```
<html>
<body>
<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML = "New text!";
</script>

</body>
</html>
```

Changing the Value of an Attribute

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

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

This example changes the value of the src attribute of an `<img>` element:

```
<!DOCTYPE html>
<html>
<body>
<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>

</body>
</html>
```

Changing CSS

Changing HTML Style

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

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

The following example changes the style of a `<p>` element:

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

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

<p>The paragraph above was changed by a script.</p>

</body>
</html>
```

DOM Events

The HTML DOM allows you to execute code when an event occurs.

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*

This example changes the style of the HTML element with `id="id1"`, when the user clicks a button:

```
<body>
<h1 id="id1">My Heading 1</h1>

<button type="button"
onclick="document.getElementById('id1').style.color = 'red'">  OR .style.fontSize = "35px"
Click Me!</button>
</body>
```

In this example JavaScript changes the value of the `src` (source) attribute of an `<img>` tag:

```
<body>
<p>JavaScript can change HTML attribute values.</p>
<button onclick="document.getElementById('myImage').src='pic_bulbon.gif'">On </button>
<img id="myImage" src="pic_bulboff.gif" style="width:100px">
<button onclick="document.getElementById('myImage').src='pic_bulboff.gif'">Off</button>
</body>
```

JavaScript Can Hide HTML Elements

Hiding HTML elements can be done by changing the `display` style:

```
<!DOCTYPE html>
<html>
<body>
<p id="id1">JavaScript can hide HTML elements.</p>

<button type="button"
onclick="document.getElementById('id1').style.display='none'">
Click Me!</button>
</body>
</html>
```

JavaScript Can Show HTML Elements

Showing hidden HTML elements can also be done by changing the `display` style:

```
<!DOCTYPE html>
<html>
<body>
<p>JavaScript can show hidden HTML elements.</p>
<p id="id" style="display:none">Hello JavaScript!</p>

<button type="button"
onclick="document.getElementById('id1').style.display='block'">
Click Me!</button>

</body>
</html>
```

JavaScript Functions and Events

A JavaScript `function` is a block of JavaScript code, that can be executed when "called" for.

For example, a function can be called when an **event** occurs, like when the user clicks a button.

```
<body>
<p id="demo">Click button to execute the displayDate() function.</p>
<button id="myBtn">Try it</button>


<script>
document.getElementById("myBtn").onclick = displayDate;

function displayDate() {
  document.getElementById("demo").innerHTML = Date();
}
</script>

</body>
```

JavaScript in <head> or <body>

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.

A JavaScript function is a block of JavaScript code, that can be executed when "called" for.

For example, a function can be called when an **event** occurs, like when the user clicks a button.

JavaScript in <head>

In this example, a JavaScript function is placed in the <head> section of an HTML page.

The function is invoked (called) when a button is clicked:

```
<html>

<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>

<body>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
</body>
</html>
```

JavaScript in <body>

In this example, a JavaScript function is placed in the <body> section of an HTML page.

The function is invoked (called) when a button is clicked:

```
<html>
<body>

<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

<script>
function myFunction() {
 document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>

</body>
</html>
```

External JavaScript

Scripts can also be placed in external files: **External file: myScript.js:**

```
function myFunction()
{
 document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

External scripts are practical when the same code is used in many different web pages. To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

```
<script src="myScript.js"></script>
```

You can place an external script reference in <head> or <body> as you like. The script will behave as if it was located exactly where the <script> tag is located. External scripts cannot contain <script> tags.

JavaScript Form Validation

HTML form validation can be done by JavaScript. It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user. JavaScript provides facility to validate the form on the client-side so data processing will be faster than server-side validation. Most of the web developers prefer JavaScript form validation. Through JavaScript, we can validate name, password, email, mobile numbers and more fields.

Example: We are going to validate the name and password. The name can't be empty and password can't be less than 6 characters long, this function alerts a message, and returns false, to prevent the form from being submitted: (https://www.javatpoint.com/javascript-form-validation)

```
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;

if (name==null || name==""){
  alert("Name can't be blank");
  return false;
}else if(password.length<6){
  alert("Password must be at least 6 characters long.");
  return false;
  }
}
</script>
<body>
<form name="myform" method="post" action="abc.jsp" onsubmit="return validateform()
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
```

Example: JavaScript Retype Password Validation

```
<script type="text/javascript">
function matchpass(){
var firstpassword=document.f1.password.value;
var secondpassword=document.f1.password2.value;

if(firstpassword==secondpassword){
return true;
}
else{
alert("password must be same!");
return false;
}
}
</script>

<form name="f1" action="register.jsp" onsubmit="return matchpass()">
Password:
<input type="password" name="password" /><br/>
Re-enter Password:<input type="password" name="password2"/><br/>
<input type="submit">
</form>
```