JavaScript DOM

CCAPDEV

Document Object Model

Allows programs and scripts to dynamically access and update the content, structure, and style of a document.

- Can change all HTML elements
- Can change all HTML attributes
- Can change all CSS styles

The **HTML DOM API** essentially provides write access to the HTML document through JavaScript

Document Object Model

HTML DOM is a standard object model and programming interface for HTML.

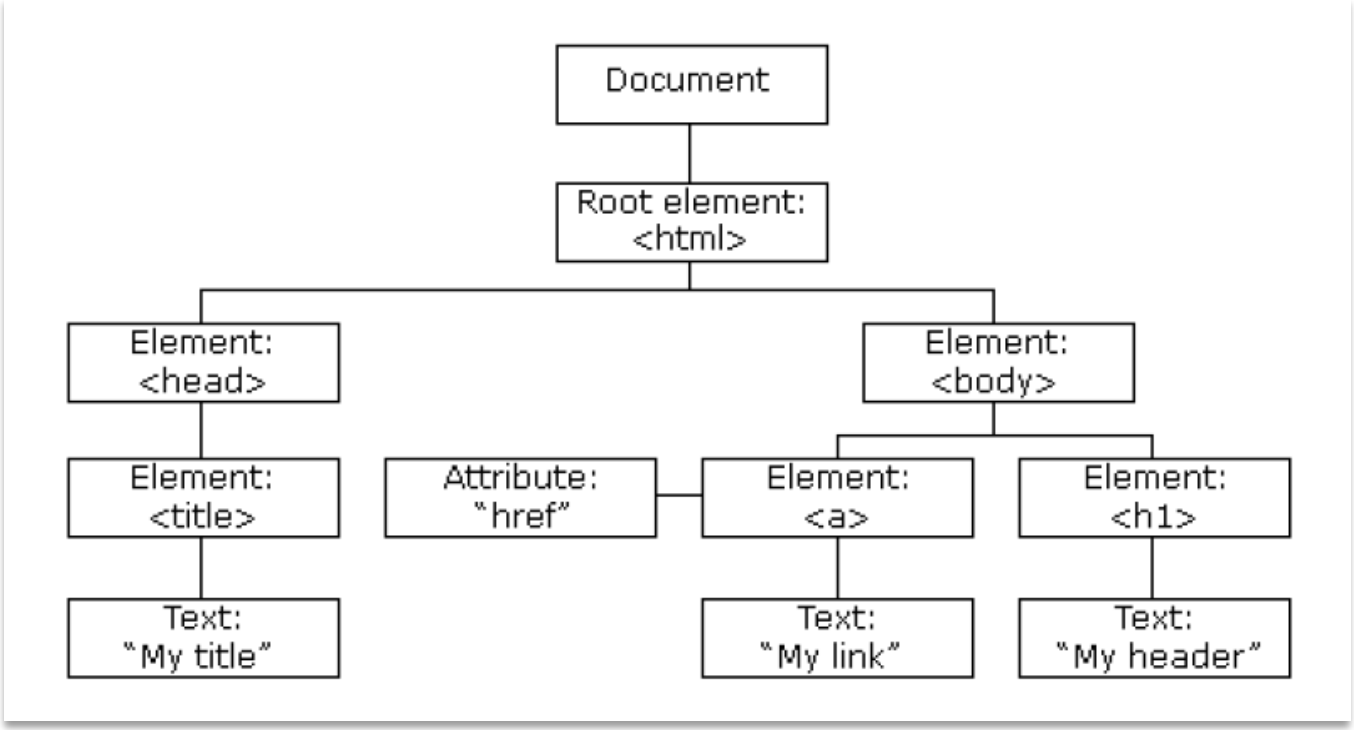


Figure 1. An HTML document in DOM format

DOM Methods

Finding HTML elements:

- `document.getElementById(id)`
- `document.getElementsByTagName(name)`
- `document.getElementsByClassName(class)`

Get Element By **ID**

The easiest way to find an HTML element in the DOM is by using the element `id`

```
<p id=“txt1”> Text 1 </p>
<p id=“txt2”> Text 2 </p>

document.getElementById(“txt1”).innerHTML=“Hi!”;
```

Get Elements By **TAG**

This will return an array of objects of a specific `<tag>`

```
<p id="intro"> Hello world! </p>
<p> Hello universe! </p>
<p class="comment"> New comment </p>

document.getElementsByTagname("p");
```

Get Elements By **CLASS**

This will return an array of objects of a specific `class` name.

```
<p class=“txt”> Text 1 </p>
<p> Hello universe! </p>
<p class=“txt”> Text 2 </p>

var txtElements = document.getElementsByClassName(“txt”);
```

After getting references to the elements...

You can change the following:

- Content
- Attributes
- Styles

Changing Content

HTML

```
<p id=“p1”> Hello World </p>
```

JS

```
document.getElementById(“p1”).innerHTML = “New Text!”;
```

Changing Content

HTML

```
<p id=“p1”> Hello World </p>
```

JS

```
var x = document.getElementById(“p1”);
x.innerHTML = “New Text!”;
```

Changing Content

HTML

```
<p class=“post-title”> Title </p>
<p id=“txt”> Text 1 </p>
<p id=“txt”> Text 2 </p>
```

JS

```
var x = document.getElementsByClassName(“txt”);
for(var i = 0; i < x.length; i++)
        x[i].innerHTML = “New Text” + i;
```

Changing Content

HTML

```
<h1> Heading </h1>
<p> Text 2 </p>
<p> Text 3 </p>
```

JS

```
var x = document.getElementsByTagName("p");
for(var i = 0; i < x.length; i++)
        x[i].innerHTML = "New Text" + i;
```

Changing Attributes

HTML

```
<img src=“logo.jpg” id=“icon”>
```

JS

```
document.getElementById(“icon”).src = “new.jpg”;
```

Changing Attributes

HTML

```
<img src=“logo.jpg” id=“icon”>
<img src=“act1.jpg” id=“act”>
<img src=“act2.jpg” id=“act”>
```

JS

```
var x = document.getElementsByClassName(“act”);
for (var i = 0; i < x.length; i++)
        x[i].src = “act_hover.jpg”;
```

Changing Attributes

HTML

```
<h1> Heading 1 </h1>
<img src=“act1.jpg” id=“act”>
<img src=“act2.jpg” id=“act”>
```

JS

```
var x = document.getElementsByTagName(“img”);
for (var i = 0; i < x.length; i++)
        x[i].src = “act_hover.jpg”;
```

Changing Styles

HTML

```
<h1 id=“title” style=“color: blue”> Title </h1>
```

JS

```
document.getElementById(“title”).style.color = “red”;
```

Changing Styles

HTML

```
<h1 id=“title” style=“color: blue”> Title </h1>
<p class=“note” style=“color: blue”> Note </p>
```

JS

```
var x = document.getElementsByClassName(“note”);
for(var i = 0; i < x.length; i++)
        x[i].style.color = “red”;
```

Changing Styles

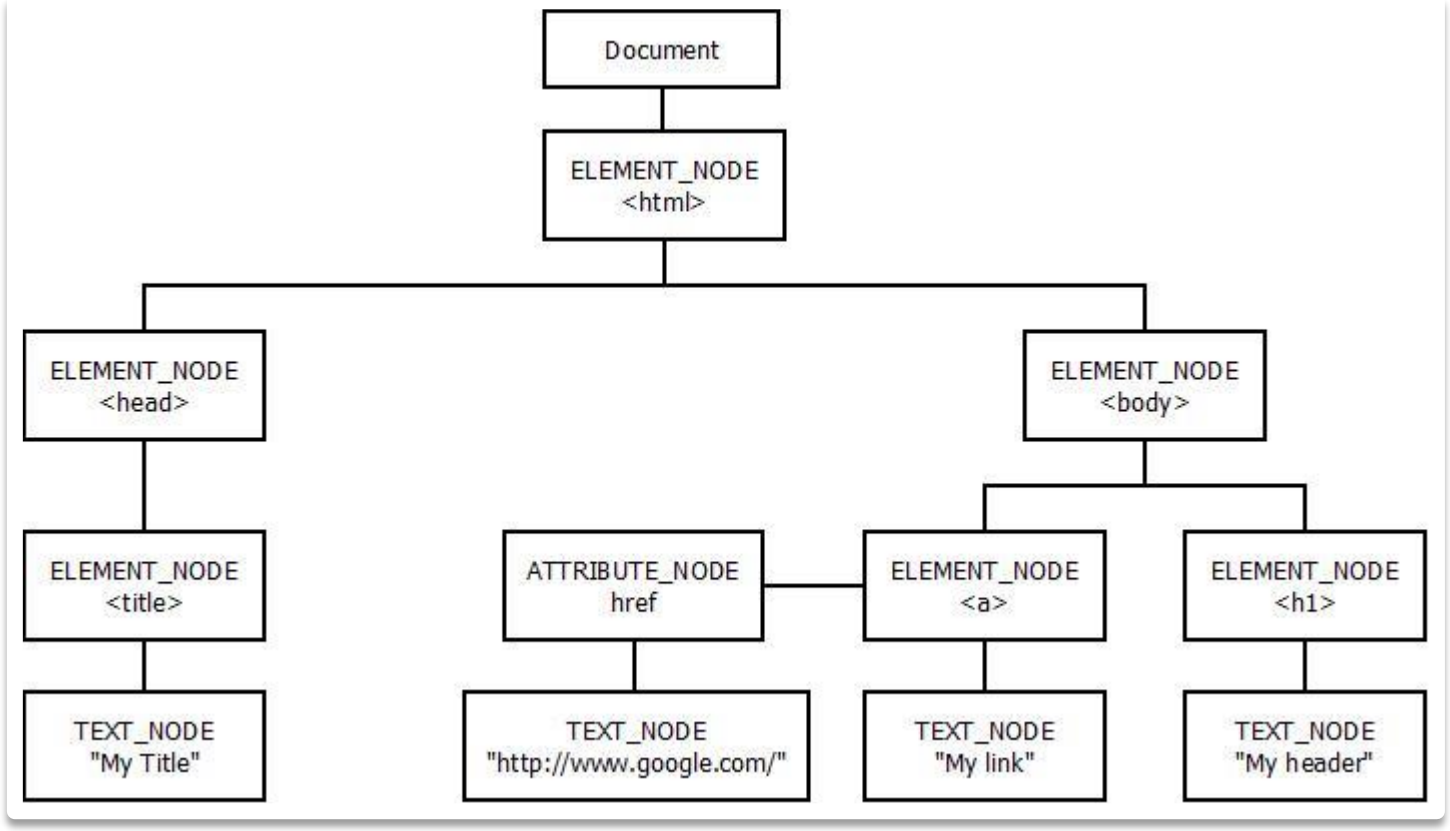
HTML

```
<h1 id="title" style="color: blue"> Title </h1>
<p class="note" style="color: blue"> Note </p>
```

JS

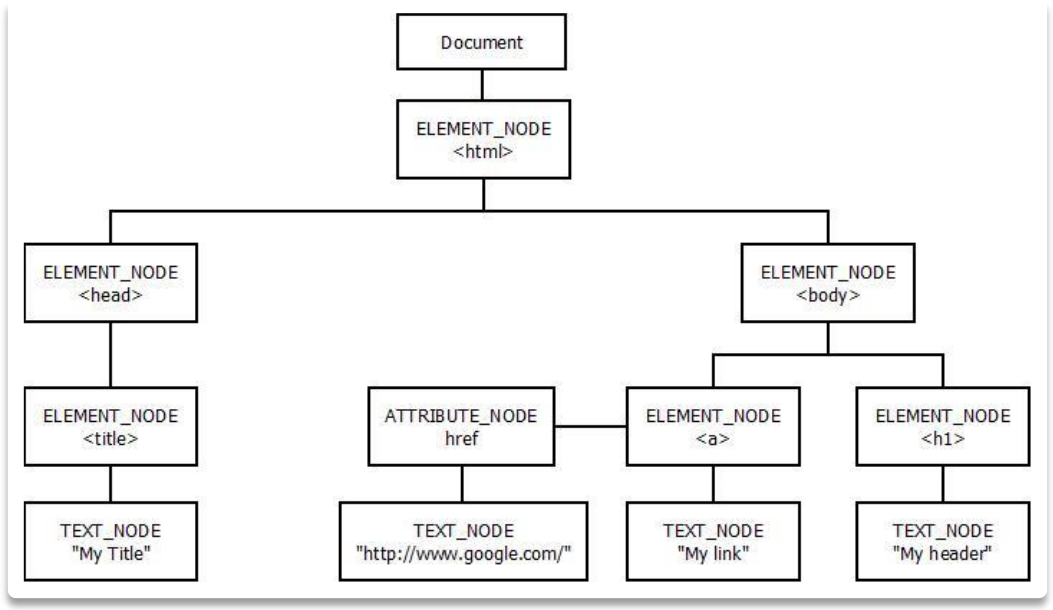
```
var x = document.getElementsByTagName("h1");
for(var i = 0; i < x.length; i++)
        x[i].style.color = "red";
```

DOM Nodes


Document
ELEMENT_NODE
<html>
ELEMENT_NODE
<head>
ELEMENT_NODE
<body>
ELEMENT_NODE
<title>
ATTRIBUTE_NODE
href
ELEMENT_NODE
<a>
ELEMENT_NODE
<h1>
TEXT_NODE
"My Title"
TEXT_NODE
"http://www.google.com/"
TEXT_NODE
"My link"
TEXT_NODE
"My header"

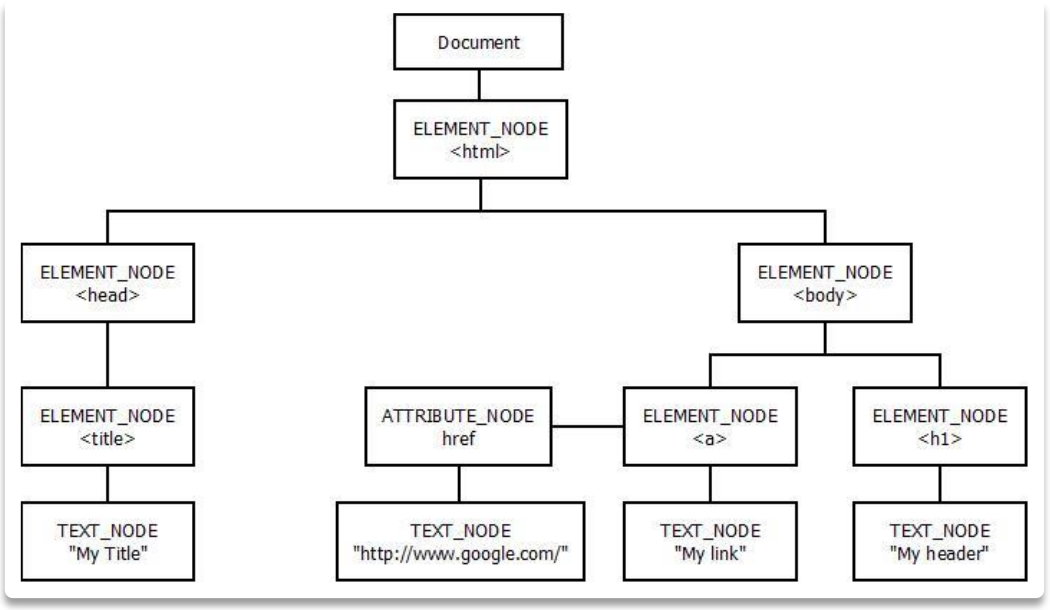
DOM Nodes

- The building blocks of the Document Object Model
- Can be of different types. Examples are:
  - ELEMENT_NODE
    - the name/tag of the element
  - TEXT_NODE
    - A text/string inside the document. Can be the content inside an element or the value of an attribute.
  - ATTRIBUTE_NODE
    - A specified attribute of an element.



DOM Nodes

- With the Node interface, you can systematically create HTML elements using JavaScript



Adding Elements using Node interface

HTML

```
<div id=“div1”>
   <p id=“p1”> This is a paragraph. </p>
</div>
```

Adding Elements using Node interface

JS

```
var p2 = document.createElement(“p”);
var node = document.createTextNode(“New text”);
p2.appendChild(node);
var element = document.getElementById(“div1”);
element.appendChild(p2);
```

Adding Elements through innerHTML

JS

```
var element = document.getElementById(“div1”);
var newP = “\n<p> New Text </p>”;
element.innerHTML += newP;
```

NOTE

use of innerHTML is generally not recommended as it is prone to security issues

Adding Elements using Node interface

JS

```
var image = document.createElement(“img”);
var source = document.createAttribute(“src”);
source.value = “0.png”;
image.setAttributeNode(source);
var element = document.getElementById(“div1”);
element.appendChild(image);
```

Removing Elements

HTML

```
<div id=“div1”>
   <p id=“p1”> This is a paragraph. </p>
</div>
```

JS

```
var x = document.getElementById(“p1”);
x.remove();
```

Removing Elements using Node interface

HTML

```
<div id=“div1”>
   <p id=“p1”> This is a paragraph. </p>
   <p id=“p2”> This is a paragraph. </p>
</div>
```

JS

```
var x = document.getElementById(“div1”);
var y = document.getElementById(“p1”);
x.removeChild(y);
```

Removing Elements using Node interface

HTML

```
<div id=“div1”>
   <p id=“p1”> This is a paragraph. </p>
   <p id=“p2”> This is a paragraph. </p>
</div>
```

JS

```
var x = document.getElementById(“div1”);
var y = document.getElementById(“p1”);
x.removeChild(y);
```

JavaScript DOM

CCAPDEV