

# A Guide to Styling with JavaScript

## □ The Document Object Model (DOM)

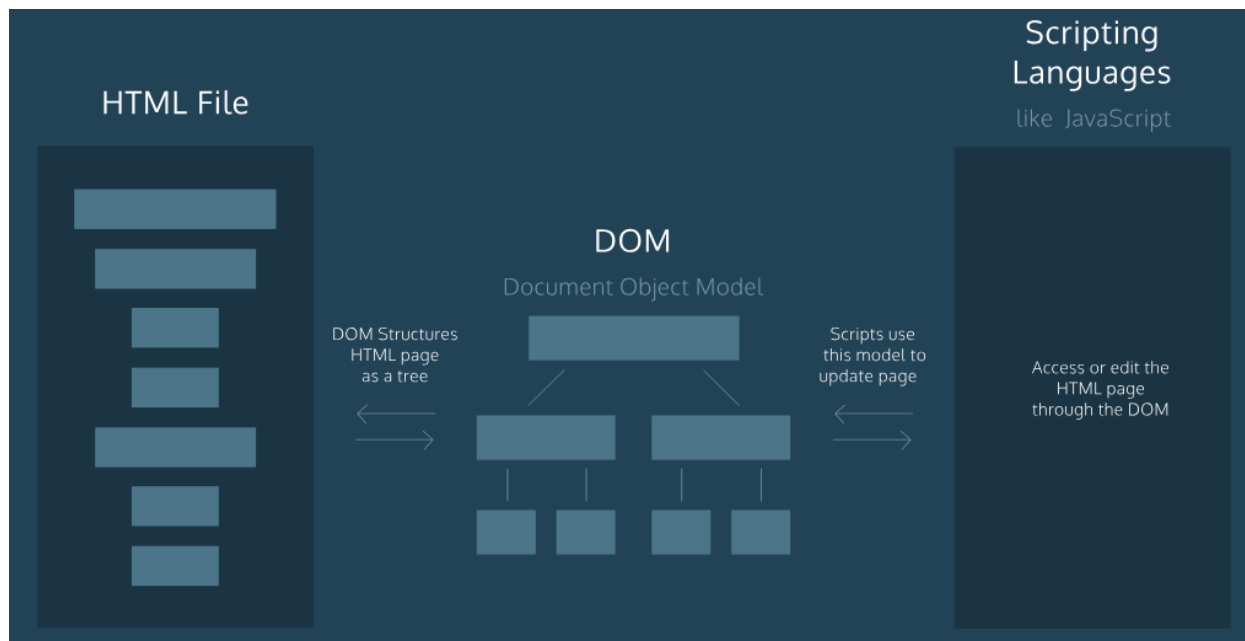
The Document Object Model, abbreviated DOM, is a powerful tree-like structure that allows programmers to conceptualize hierarchy and access the elements on a web page.

The DOM is one of the better-named acronyms in the field of Web Development. In fact, a useful way to understand what DOM does is by breaking down the acronym but out of order:

- The DOM is a logical tree-like **Model** that organizes a web page's HTML **Document** as an **Object**.

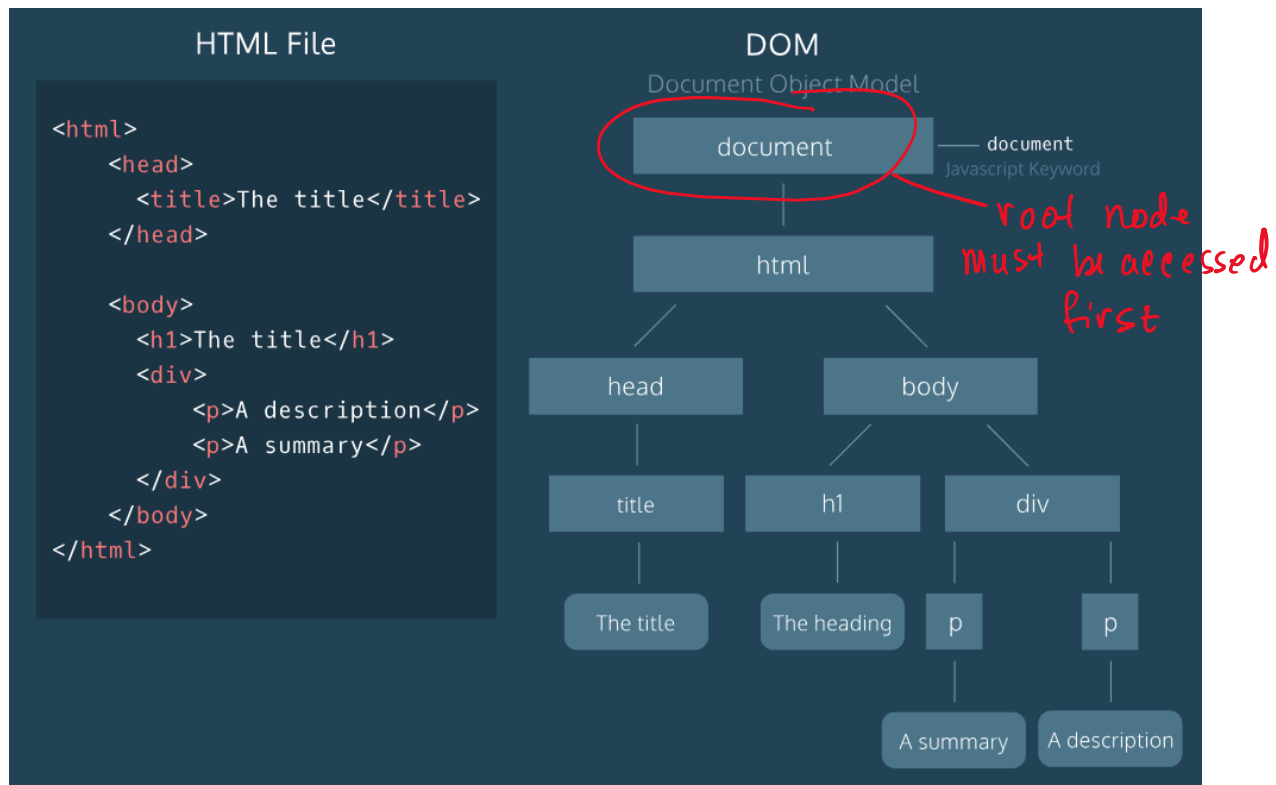
The DOM is a language-agnostic structure implemented by browsers to allow for web scripting languages, like JavaScript, to access, modify, and update the structure of an HTML web page in an organized way.

For this reason, we like to think of the DOM as the link between an HTML web page and scripting languages.



## ❑ Selecting DOM elements

Before we dive in to writing code to select DOM elements, let's first revisit how the DOM can be used to represent a simple HTML file.



Notice in the above example, that at the root of our model is the *document* object. In JavaScript, the *document* object is the door to the DOM structure. Before you can access a specific element within the document you must first access this root object. Consider the following code which logs the document object to the console. Within the document you can see the entire contents of the webpage.

Code	Output
<pre>console.log(document);</pre>	<pre>▼ #document   &lt;!DOCTYPE html&gt;   &lt;html&gt;     ► &lt;head&gt;...&lt;/head&gt;     ► &lt;body&gt;...&lt;/body&gt;   &lt;/html&gt;</pre>

The elements within the document of the page can be selected using the *dot* notation. Below are some examples,

Code	Output
<pre>console.log(document.head);</pre>	<pre>▼ &lt;head&gt;   &lt;script src="Scripts/App.js" defer&gt;&lt;/script&gt;   &lt;title&gt;Davie JR's Menu&lt;/title&gt; &lt;/head&gt;</pre>
<pre>console.log(document.body);</pre>	<pre>▼ &lt;body&gt;   &lt;img id="foodLogoImage" src="Images/foodlogo.png"&gt;   &lt;div id="description"&gt;&lt;/div&gt;   ► &lt;div id="nav"&gt;...&lt;/div&gt;   ► &lt;div id="content"&gt;...&lt;/div&gt; &lt;/body&gt;</pre>
<pre>console.log(document.title);</pre>	Davie JR's Menu

In addition to selecting elements by their name, we can also select elements by their id. Consider the output for the following code snippets which reference the HTML page below,

Code	Output
<pre>console.log(document.getElementById("nav"));</pre>	<pre>▼ &lt;div id="nav"&gt;   &lt;img id="logoImage" src="Images/logo.svg"&gt;   &lt;a href="#menu"&gt;MENU&lt;/a&gt;   &lt;a href="#nutrition"&gt;NUTRITION&lt;/a&gt;   &lt;a href="#order"&gt;ORDER&lt;/a&gt;   &lt;a href="#locations"&gt;LOCATIONS&lt;/a&gt; &lt;/div&gt;</pre>
<pre>console.log(document.getElementById("content"));</pre>	<pre>▼ &lt;div id="content"&gt;   ► &lt;div id="menu"&gt;...&lt;/div&gt;   ► &lt;div id="nutrition"&gt;...&lt;/div&gt; &lt;/div&gt;</pre>
<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;   &lt;head&gt;     &lt;script src="App.js" defer&gt;&lt;/script&gt;     &lt;title&gt;Davie JR's Menu&lt;/title&gt;   &lt;/head&gt;   &lt;body&gt;     &lt;img id = "foodLogoImage" src="foodlogo.png"&gt;     &lt;div id="description"&gt;&lt;/div&gt;     &lt;div id = "nav"&gt;       &lt;img id = "logoImage" src="logo.svg"&gt;       &lt;a href="#menu"&gt;MENU&lt;/a&gt;       &lt;a href="#nutrition"&gt;NUTRITION&lt;/a&gt;       &lt;a href="#order"&gt;ORDER&lt;/a&gt;       &lt;a href="#locations"&gt;LOCATIONS&lt;/a&gt;     &lt;/div&gt;     &lt;div id="content"&gt;       &lt;div id="menu"&gt;         &lt;img id = "burgerImage" src = "burger.jpg"&gt;         &lt;h1&gt;BBQ BACON BURGER&lt;/h1&gt;         &lt;p class="item"&gt;BBQ Bacon Burger&lt;/p&gt;         &lt;a href="#" id="button"&gt;ORDER NOW&lt;/a&gt;       &lt;/div&gt;       &lt;div id="nutrition"&gt;         &lt;p class="cals"&gt;CALORIES&lt;/p&gt;         &lt;p class="value"&gt;678&lt;/p&gt;       &lt;/div&gt;     &lt;/div&gt;   &lt;/body&gt; &lt;/html&gt;</pre>	

## ❑ Changing the contents of an element

Once an element is selected, we can alter its properties. You can access and set the contents of an element with the `.innerHTML` property.

For example, the following code reassigns the inner HTML of the body element to the text 'The cat loves the dog':

```
document.body.innerHTML = "The cat hates the dog";
```

The `.innerHTML` property can also add any valid HTML, including properly formatted elements. The following example assigns an `h2` element inside the `<body>` element:

```
document.body.innerHTML = '<h2>This is a heading</h2>';
```

Below are more examples,


Code	Output
<pre>var someText = "Yummy!"; document.getElementById("description").innerHTML = someText; console.log(document.getElementById("description"));</pre>	<pre>&lt;div id="description"&gt;Yummy!&lt;/div&gt;</pre>
<pre>var someText = "Not available"; document.getElementById("button").innerHTML = someText; console.log(document.getElementById("button"));</pre>	<pre>&lt;a href="#" id="button"&gt;Not available&lt;/a&gt;</pre>
<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;   &lt;head&gt;     &lt;script src="App.js" defer&gt;&lt;/script&gt;     &lt;title&gt;Davie JR's Menu&lt;/title&gt;   &lt;/head&gt;   &lt;body&gt;     &lt;img id = "foodLogoImage" src="foodlogo.png"&gt;     &lt;div id="description"&gt;&lt;/div&gt;     &lt;div id = "nav"&gt;       &lt;img id = "logoImage" src="logo.svg"&gt;       &lt;a href="#menu"&gt;MENU&lt;/a&gt;       &lt;a href="#nutrition"&gt;NUTRITION&lt;/a&gt;       &lt;a href="#order"&gt;ORDER&lt;/a&gt;       &lt;a href="#locations"&gt;LOCATIONS&lt;/a&gt;     &lt;/div&gt;     &lt;div id="content"&gt;       &lt;div id="menu"&gt;         &lt;img id = "burgerImage" src = "burger.jpg"&gt;         &lt;h1&gt;BBQ BACON BURGER&lt;/h1&gt;         &lt;p class="item"&gt;BBQ Bacon Burger&lt;/p&gt;         &lt;a href="#" id="button"&gt;ORDER NOW&lt;/a&gt;       &lt;/div&gt;       &lt;div id="nutrition"&gt;         &lt;p class="cals"&gt;CALORIES&lt;/p&gt;         &lt;p class="value"&gt;678&lt;/p&gt;       &lt;/div&gt;     &lt;/div&gt;   &lt;/body&gt; &lt;/html&gt;</pre>	

## ❑ Changing the style of an element

Another way to modify an element is by changing its style. The *.style* property of a DOM element provides access to the inline style of that HTML tag. There are many ways we can style an element, in this section we will discuss a few of the more common applications.

### Font Color

To change the color of text we use the *.color* property. This is illustrated in the following example. Notice that we simply typed the color “red” and the browser new how to interpret it. Modern browsers support 140 named colors, which are listed here, <https://htmlcolorcodes.com/color-names/>

<pre> &lt;div id="menu"&gt;   &lt;img id = "burgerImage" src = "burger.jpg"&gt;   &lt;h1&gt;BBQ BACON BURGER&lt;/h1&gt;   &lt;p class="item"&gt;BBQ Bacon Burger&lt;/p&gt;   &lt;a href="#" id="button"&gt;ORDER NOW&lt;/a&gt; &lt;/div&gt; </pre>	
Code	Output
<pre> var content = document.getElementById("menu"); content.style.color = "red"; </pre>	

### Background Color



To change the background color of an element we use the `.backgroundColor` property.

<pre> &lt;div id="menu"&gt;   &lt;img id = "burgerImage" src = "burger.jpg"&gt;   &lt;h1&gt;BBQ BACON BURGER&lt;/h1&gt;   &lt;p class="item"&gt;BBQ Bacon Burger&lt;/p&gt;   &lt;a href="#" id="button"&gt;ORDER NOW&lt;/a&gt; &lt;/div&gt; </pre>	
Code	Output
<pre> var content = document.getElementById("menu"); content.style.color = "red"; content.style.backgroundColor = "Gainsboro"; </pre>	

### Dimensions

The width and height of elements can also be specified. When specifying the width or height of an element you can either indicate the value in pixels (px) or as a percentage (%). If you are resizing an image, it is best to resize only one of the dimensions to ensure it scales and doesn't appear distorted.

You may be wondering why we would want to define the dimensions as percentages. When you consider the diversity in screen sizes, percentages make sense, because different screen sizes will render sizes specified in pixels differently.

<pre> &lt;img id = "burgerImage" src = "burger.jpg"&gt; </pre>	
Code	Output
<pre> var image = document.getElementById("burgerImage"); image.style.width = "100px"; </pre>	 <p><b>BBQ BACON BURGER</b></p>
<pre> var image = document.getElementById("burgerImage"); image.style.width = "20%"; </pre>	 <p><b>BBQ BACON BURGER</b></p>

## Font Size

The size of font can be specified with the `fontSize` property. Just as with the width and height, we can also specify the font relative to the screen we are on by using units *em* as opposed to pixels (px). If you specify the *fontSize* to 2 em, the font will appear 2x as large. If you specify the *fontSize* to 1.5 em, the font will appear 1.5x as large, etc.

<pre>&lt;div id="menu"&gt;   &lt;img id = "burgerImage" src = "burger.jpg"&gt;   &lt;h1&gt;BBQ BACON BURGER&lt;/h1&gt;   &lt;p class="item"&gt;BBQ Bacon Burger&lt;/p&gt;   &lt;a href="#" id="button"&gt;ORDER NOW&lt;/a&gt; &lt;/div&gt;</pre>	
Code	Output
<pre>var content = document.getElementById("menu"); content.style.fontSize = "2em";</pre>	<b>BBQ BACON BURGER</b>  BBQ Bacon Burger  <a href="#">ORDER NOW</a>

## Font Style


There are three basic font styles you can apply to text: normal, italic, or oblique. These can be applied with the *fontStyle* property.

<pre>&lt;div id="menu"&gt;   &lt;img id = "burgerImage" src = "burger.jpg"&gt;   &lt;h1&gt;BBQ BACON BURGER&lt;/h1&gt;   &lt;p class="item"&gt;BBQ Bacon Burger&lt;/p&gt;   &lt;a href="#" id="button"&gt;ORDER NOW&lt;/a&gt; &lt;/div&gt;</pre>	
Code	Output
<pre>var content = document.getElementById("menu"); content.style.fontStyle = "oblique";</pre>	<b><i>BBQ BACON BURGER</i></b>  <i>BBQ Bacon Burger</i>  <a href="#">ORDER NOW</a>

## Borders


Borders can be applied to any element using the *border* property. For a border to appear however requires that you specify three values: style, thickness, and color.

There are three different styles you can apply: dotted, dashed, solid. The thickness can be specified in pixels (px) or em. The color can be specified as before.

<pre> &lt;div id="menu"&gt;   &lt;img id = "burgerImage" src = "burger.jpg"&gt;   &lt;h1&gt;BBQ BACON BURGER&lt;/h1&gt;   &lt;p class="item"&gt;BBQ Bacon Burger&lt;/p&gt;   &lt;a href="#" id="button"&gt;ORDER NOW&lt;/a&gt; &lt;/div&gt; </pre>	
Code	Output
<pre> var content = document.getElementById("menu"); content.style.border = "dotted 2em green"; </pre>	

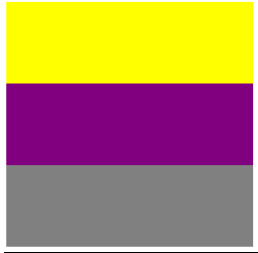
### Text Alignment

The text within an element can be aligned as left, right, or center using the `.textAlign` property.

<pre> &lt;div id="menu"&gt;   &lt;img id = "burgerImage" src = "burger.jpg"&gt;   &lt;h1&gt;BBQ BACON BURGER&lt;/h1&gt;   &lt;p class="item"&gt;BBQ Bacon Burger&lt;/p&gt;   &lt;a href="#" id="button"&gt;ORDER NOW&lt;/a&gt; &lt;/div&gt; </pre>	
Code	Output
<pre> var content = document.getElementById("menu"); content.style.border = "dotted 2em green"; content.style.textAlign = "center"; </pre>	

## Position

The majority of the elements you create on a webpage are called block-level elements. Block-level elements like paragraphs, or divs take up the full width of their parents and they prevent other elements from appearing in the same horizontal space. Consider the following code and output,

<pre>&lt;div id = "one"&gt;&lt;/div&gt; &lt;div id = "two"&gt;&lt;/div&gt; &lt;div id = "three"&gt;&lt;/div&gt;</pre>	
Code	Output
<pre>var one = document.getElementById("one"); var two = document.getElementById("two"); var three = document.getElementById("three");  one.style.width = "50%"; two.style.width = "50%"; three.style.width = "50%";  one.style.height = "5em"; two.style.height = "5em"; three.style.height = "5em";  one.style.backgroundColor = "yellow"; two.style.backgroundColor = "purple"; three.style.backgroundColor = "gray";</pre>	


In the example above the block-level elements also appear on the left side of the browser. This is the default *position* for block-level elements.

The default position of an element can be changed by setting its *position* property. The *position* property can take one of four values:

1. static- the default value (it does not need to be specified)
2. relative
3. absolute
4. fixed


The *relative* value allows you to position an element relative to its default position on the web page.

In the example below, div three is shifted 10px from the top and left border relative to its default location.

<pre>&lt;div id = "one"&gt;&lt;/div&gt; &lt;div id = "two"&gt;&lt;/div&gt; &lt;div id = "three"&gt;&lt;/div&gt;</pre>	
Code	Output
<pre>three.style.position = "relative"; three.style.top = "10px"; three.style.left = "10px";</pre>	



When an element's position is set to *absolute* you can place the element exactly where you want relative to the page. All elements on the page will ignore an element that is positioned with the *absolute* designation.

<pre>&lt;div id = "one"&gt;&lt;/div&gt; &lt;div id = "two"&gt;&lt;/div&gt; &lt;div id = "three"&gt;&lt;/div&gt;</pre>	
Code	Output
<pre>two.style.position = "absolute"; two.style.top = "50px"; two.style.left = "50px";  three.style.position = "absolute"; three.style.top = "100px"; three.style.left = "100px";</pre>	

We can fix an element to a specific position on the page (regardless of user scrolling) by setting its position to *fixed*.

<pre>&lt;div id = "one"&gt;&lt;/div&gt; &lt;div id = "two"&gt;&lt;/div&gt; &lt;div id = "three"&gt;&lt;/div&gt;</pre>	
Code	Output
<pre>one.style.backgroundColor = "yellow"; two.style.backgroundColor = "purple"; three.style.backgroundColor = "gray";  two.style.position = "absolute"; two.style.top = "50px"; two.style.left = "50px";  three.style.position = "fixed"; three.style.top = "100px"; three.style.left = "100px";</pre>	