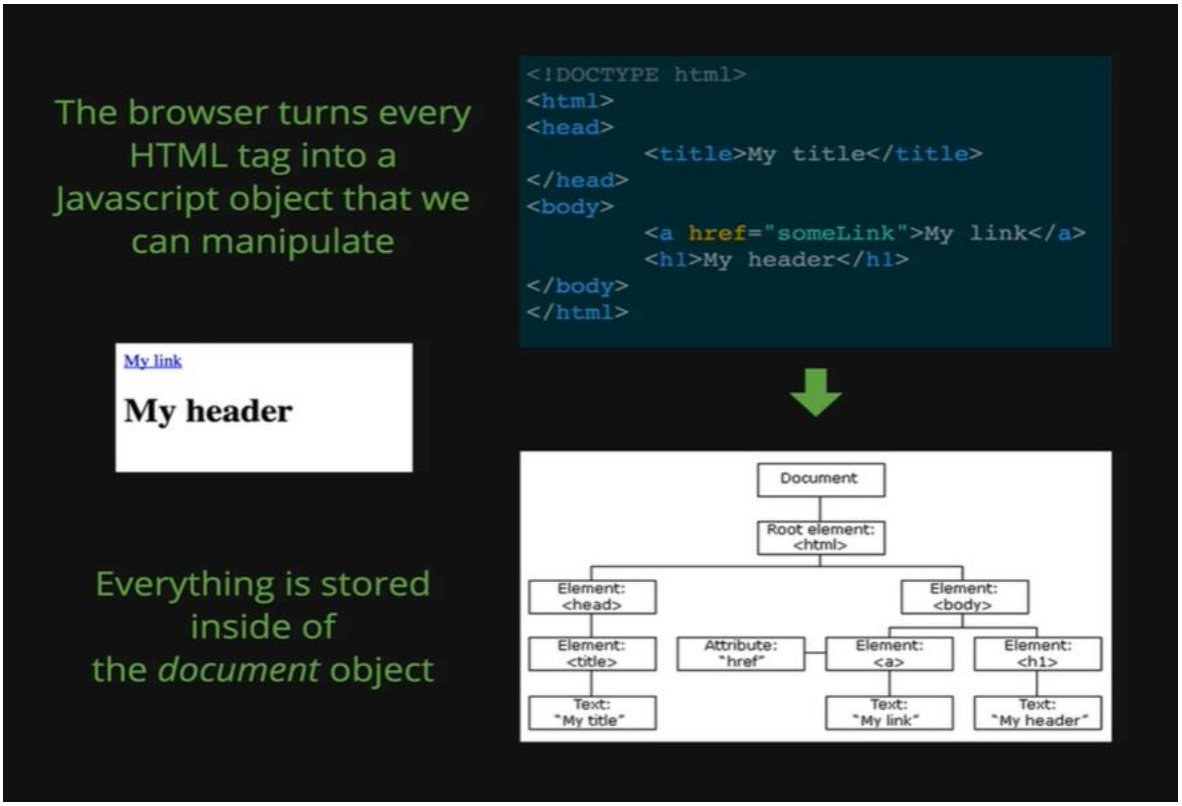
DOM – Document Object Model

- *The Document Object Model is the interface between our JavaScript and HTML+CSS*

At a higher level the DOM is the interface between our JavaScript and our HTML and CSS. Its basically a bunch of special JavaScript objects, methods and functions that we can use to interact with our HTML and CSS. We can change things, we can add elements, remove elements, change colors, animate things, all sort of exciting ways to interact with our HTML and CSS.



Here we will talk about how the DOM is constructed, and why it is called the *Document Object Model.* When we load up an HTML page, it looks like the My Header text on the white background on the browser and it is just some simple HTML, so we have title, anchor tag with My link as the text, and an h1 with My header as the text and that's it.

When we open that in the browser, we see the My Header text on the white background, but behind the scene the DOM is created, the HTML that we wrote gets constructed as the document object as we can see the third image on the picture. The HTML gets turn into a bunch of JavaScript objects. Each of the object models are one of the elements from the HTML code.

Again, to remind ourselves, we load up the HTML on the browser, the browser displays things to us, but behind the scene it makes the model of every single element with a JavaScript object. Just like we modeled the dog Rusty using a JavaScript object, he had properties like name, breed and age; these objects are fundamentally the same thing, except the content is different. Here we are going to have things like type of tag, texts, color, background color, text size all different properties that all contained inside of different objects.

All we want to emphasize here is that we load up HTML, its displayed in the browser and the browser takes that HTML and takes each element and turns it into an object. That's covered the *Object Model* part but what about the *Document*? We can see on the third image of the picture; it is a diagram of the type of structure that we get from the DOM. The top-level object that everything is contained inside of. Our entire model of *HTML* and *CSS* is called the *Document.*

We can go to any webpage at all. For example, we go to the site of Google and we have the console open where we type document as below.

Console coding:

document

- ➔ *#document*
  - ➢ *<html itemscope itemtype=http://schema.org/WebPage lang="en">…</html>*

So, when we do that, we get a big thing, that comes back, a big object, except it does not look like a JavaScript Object, it is not what we expect, that is because the JavaScript is hiding the full Object from us. What it is showing us is what the HTML looks like as text, but it is not showing us the Object that is modeled inside this HTML.

To get that we need, we need to use another console object.

Console coding:

console.dir(document)

- ➔ *#document*
  - ➢ *URL: https://www.google.com/*
    ……
    ……
    ……

This prints out the entire document object in the object syntax that we are familiar with, it's a regular object like the one we showed on the picture, except there are a lot of information at this one. We have things like fonts, images, links and we also have things like body. This body represents the entire body, all the elements that we see on our site is inside of the body. If we

open the section of body, we will see whole bunch of other properties, including one called childNodes. The childNodes shows us that there's a div, there's a script tag and then there's another div inside of the body. If we go to 'elements', we would see that the body, inside the body we have the first div, we have a script tag and then we have a second div and if we open up that second div we would see two more div inside of that second div, and if we go back to the 'console' from the 'element' and we look at the second div and then we look at childNodes, we can see two more div.

So, as we can see the DOM gets huge in a really short time. There are hundreds and hundreds of different properties and methods and different nodes in this whole crazy structure of nested objects, there are just many stuffs in it.

We need to be comfortable not knowing everything, we can tell that most developers, do not know half of the properties inside the DOM. They just work with 10% of those properties at most. There are many stuffs in there that we almost never need to use, and if sometimes down the line something comes up then we need to Google it and find out what property is meant to do what task. This is what it likes to be a developer, up until now with the JavaScript that we have been working with at a small controlled world, where we are learning about functions, objects, variables and there's a finite number of ways of doing things and there's a specific subset of knowledge that we need to know. With the DOM now we have a huge ecosystem to work in, so many properties.

Lets just reiterate, there is a lot in a DOM, and we are going to focus in the important pieces, but the goal is not to memorize, and the goal is not to master every single property. Again, 10% at most, is all that we need to become a component web developer.

The DOM stands for *Document Object Model.* It's the interface between our JavaScript, and our HTML and CSS. Its how we can use our JavaScript to make things interactive. Its how we do things like add a new element to the page or change the *href* attribute of a link or change the background image of our body or how we can get the value out of a form or how we can do custom validations and animations, add interactive elements where we click and some code runs. All these different possibilities, all stem from the fact that the Document Object Model connects our JavaScript with our HTML and CSS. The *Object Model* part in *Document Object Model* tells us that it's a bunch of JavaScript objects those are modelling elements in our HTML. So, if we have an anchor tag, we end up with an object inside our DOM representing the anchor tag, and then there is another object that is representing the h1 tag. In this way it contains all the tags and elements of HTML and CSS structured in *Document Object Model.*

Lastly, it's called the *Document Object Model,* because everything lives inside that one object called the *Document*. It's the root node where everything lives inside of.