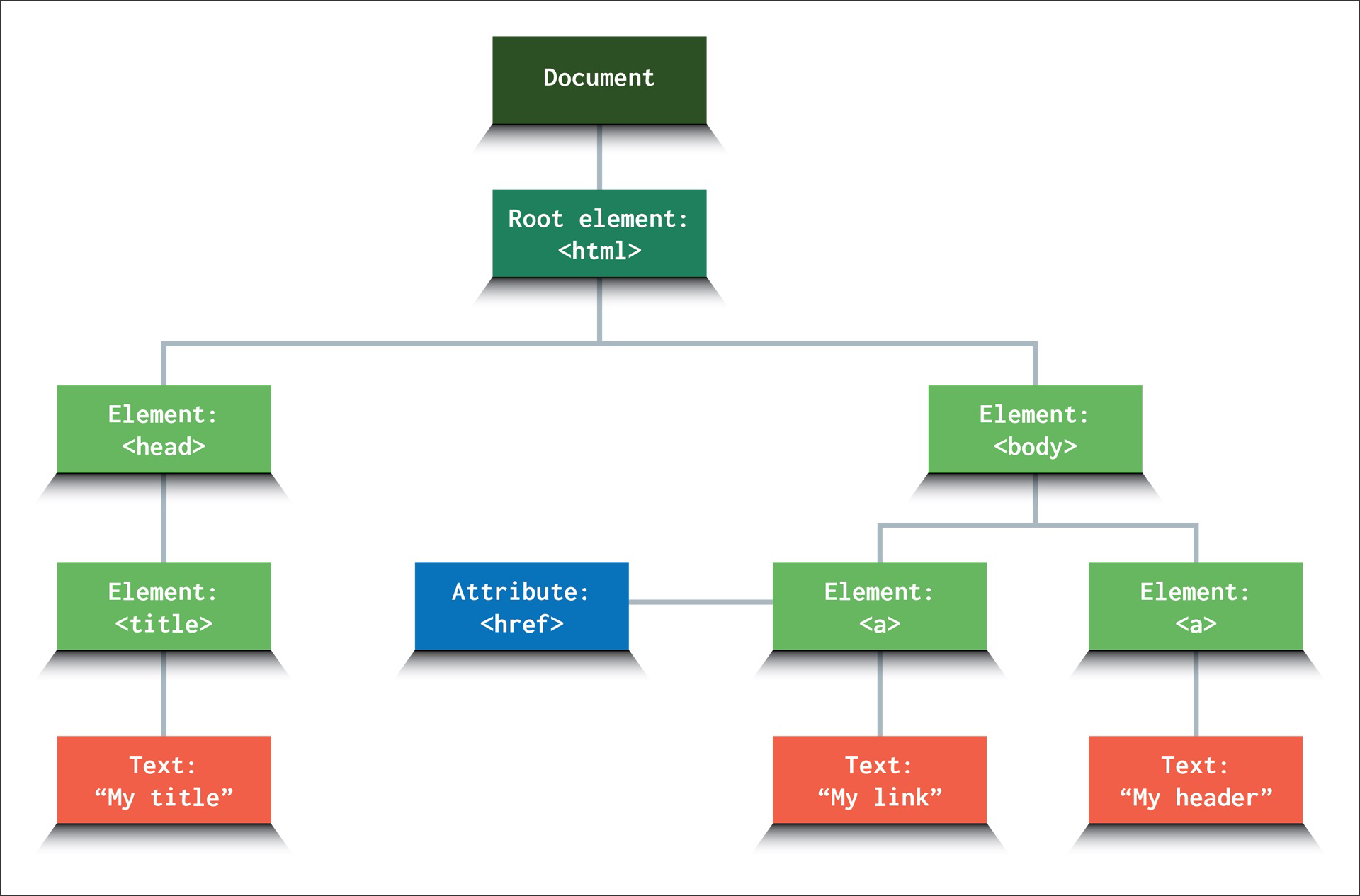
console.log(window.document);

Although this result may look like HTML, it's actually the object representation of the webpage. We call this the **Document Object Model**, or **DOM**. The document object represents the root element or the highest-level element of the webpage, which is the <html> element in the index.html file. The rest of the elements are the descendant elements of the document object.

To better illustrate the object representation of the elements, type the following code

console.dir(window.document);

Unlike console.log, which displays the element's HTML, console.dir() displays the HTML element as an object, known as a **DOM element**. And because it is an object, that means we can access its properties and methods using dot notation. In fact, the document object allows us to access everything on the webpage, including all of the elements and their attributes, text content, metadata, and much more.



The DOM has various methods that allow us to use JavaScript to find specific elements within it. Let's use one of those methods, querySelector(), to find the <button> and log the results in the console.

Type the following code in the browser console for the index.html file:

window.document.querySelector("button");

This looks like HTML, but it's actually an object representation of the <button> element. We can verify that this is actually a DOM element by typing the following into the browser console:

var btn = window.document.querySelector("button");

console.dir(btn);

Did you notice the document prefix on the querySelector() method? The document is the root DOM element, which represents the index.html file that's open in the browser. We can use querySelector()to select any element in the HTML, including the <button> that we need here.

The querySelector() method searches down from the document object through all the descendant elements. In addition to element types—like <button>—the versatile querySelector() method can also search selectors and attributes.

Experiment in the console by selecting different elements. Try to target the <body> or <main>. You'll find that all the elements in HTML are available to target.

To select a class attribute, you need to add a dot (.) prefix, as shown in the following code snippet:

document.querySelector(".btn");

The same object will be displayed in the console for the <button> element, even though we chose a different selector.

Notice that we didn't add window before document in the expression above. A window prefix is unnecessary because we opened index.html in the browser, where window is a global object.

Even though the display in the console for the <button> query looks like raw HTML, it's actually an object representation of this element. And because it's an object, we have access to its built-in properties and methods.

Let's investigate one of these properties—textContent—by typing the following code into the console:

document.querySelector("button").textContent;

As you can see, the console displays "Add Task". Thus we can conclude that the aptly named textContent property returns the text content of the element!

You were able to select and get the text content of the button, but if you added more buttons to the page, how would you distinguish this button from the rest? To do this, we can use an id attribute and assign it an id name. Let's try that now.

In the index.html file, add the id="save-task" attribute to the button element so that it looks like the following code:

<button class="btn" id="save-task">Add Task</button>

Save the file and refresh the browser so that the DOM has the id attribute on the element. Then update the querySelector() call to look for the id attribute instead of the generic element.

Type the following code into the console to see how the result looks:

document.querySelector("#save-task");

The result should look like the <button class="btn" id="save-task"> element that you just updated in the HTML.

Having successfully targeted the HTML element, now we can add this code into our JavaScript file, script.js. To do that, delete console.dir(window.document); and then add the following code to assign the button element object representation to a variable in the file:

DOM notes

document.getElementById

document.getElementById(“text-header”)

<h1 id=”text-header”>This is a header</h1>

document.createElement

document.createElement(“div”)

<div></div>

element.appendChild

var newTextElement = document.createElement(“p”);

document.body.appendChild (newTextElement);

<p></p>

**HTML** is a structured text language.

The **DOM** is a parsed version of HTML … that can be manipulated using JavaScript.

P B Q D T N R L K

U Z O E H G C W M

B E G I N N E R G L O B A L B U N G A L O W

C R O W N Y E L L O W S T E P

D E D I C A T E D C O R D B E Y O N D

A S T E R O I D O R E B R E A D

P O T A T O C H E F T O N G U E

A F I J S U V X Y N I C E L Y W A L T Z

$370.00