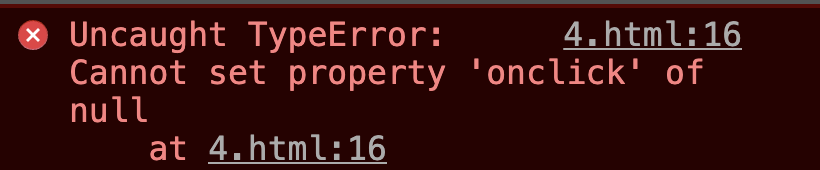
**<lecture5. javascript>**

**DOM Manipulation**

document.querySelector('button').onclick = count;

One thing to notice about what we’ve just done is that we’re not calling the count function by adding parentheses afterward, but instead just naming the function. This specifies that we only wish to call this function when the button is clicked. This works because, like Python, JavaScript supports functional programming, so functions can be treated as values themselves.

The above change alone is not enough though, as we can see by inspecting the page and looking at our browser’s console:



This error came up because when JavaScript searched for an element using document.querySelector('button'), it didn’t find anything. This is because it takes a small bit of time for the page to load, and our JavaScript code ran before the button had been rendered. To account for this, we can specify that code will run only after the page has loaded using the addEventListener function. This function takes in two arguments:

1. An event to listen for (eg: 'click')
2. A function to run when the event is detected (eg: hello from above)

We can use the function to only run the code once all content has loaded:

텍스트이(가) 표시된 사진

자동 생성된 설명

**dataset**

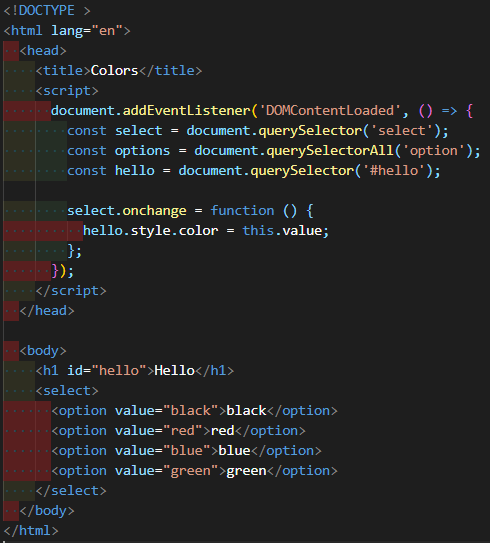


Some notes on the page above:

* We change the style of an element using the style.SOMETHING attribute.
* We use the data-SOMETHING attribute to assign data to an HTML element. We can later access that data in JavaScript using the element’s dataset property.
* We use the querySelectorAll function to get an Node List (similar to a Python list or a JavaScript array) with all elements that match the query.
* The forEach function in JavaScript takes in another function, and applies that function to each element in a list or array.

**querySelectorAll & this**

To get an idea about some other events we can use, let’s see how we can implement our color switcher using a dropdown menu instead of three separate buttons. We can detect changes in a select element using the onchange attribute. In JavaScript, this is a keyword that changes based on the context in which it’s used. In the case of an event handler, this refers to the object that triggered the event.



There are many other events we can detect in JavaScript including the common ones below:

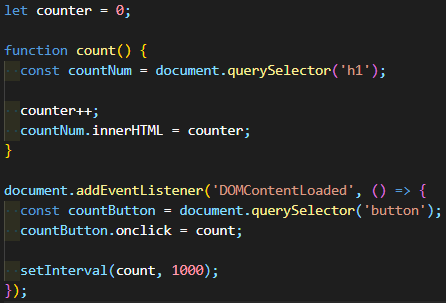
* onclick
* onmouseover
* onkeydown
* onkeyup
* onload
* onblur
* …

**Simple ToDoList**



* We can enable/disable a button by setting its disabled attribute to false/true.
* In JavaScript, we use .length to find the length of objects such as strings and arrays.
* At the end of the script, we add the line return false. This prevents the default submission of the form which involves either reloading the current page or redirecting to a new one.
* In JavaScript, we can create HTML elements using the createElement function. We can then add those elements to the DOM using the append function.

**Intervals**



the counter increments every second. To do this, we use the setInterval function, which takes as argument a function to be run, and a time (in milliseconds) between function runs.

**Local Storage**

we’ll want to be able to store information that we can use when a user returns to the site.

One way we can do this is by using Local Storage, or storing information on the user’s web browser that we can access later. This information is stored as a set of key-value pairs, almost like a Python dictionary. In order to use local storage, we’ll employ two key functions:

* localStorage.getItem(key): This function searches for an entry in local storage with a given key, and returns the value associated with that key.
* localStorage.setItem(key, value): This function sets and entry in local storage, associating the key with a new vlaue.



**Currency Exchange**

# currency.js

****

we’ll use something called AJAX, or Asynchronous JavaScript And XML, which allows us to access information from external pages even after our page has loaded. In order to do this, we’ll use the fetch function which will allow us to send an HTTP request. The fetch function returns a promise.

