Java script lab2 report

Nabil AL-Harethi

**Learning reflection:**

while creating  contact.js to set up DOM element selection

The challenges I faced were:

1) Understanding the difference between getElementById and querySelector.
2) Dealing with null references due to typos in my id attributes.

**What I Learned**

I learned that it is better to store element references in variables at the top of the contact.js file. My first time through, I had document.getElementById() calls sprinkled throughout the file. This created a lot of unnecessary lines of code and a very messy code style, making things harder and slower for me. Storing my element references in variables at the top of my contact.js file will create a much cleaner and faster code style.

```
//store refrence to all form elements

const contactForm = document.getElementById("contact-form");
const firstNameInput = document.getElementById("first-name");
const lastNameInput = document.getElementById("last-name");
const emailInput = document.getElementById("email");
const phoneInput = document.getElementById("phone");
const subjectInput = document.getElementById("subject");
const messgeTextarea = document.getElementById("message");
const clearButton = document.getElementById("reset-button");
const successMessage = document.getElementById("success-message");
const successNameSpan = document.getElementById("successName");
const charCount = document.getElementById("char-counter");

console.log("All form elements referenced.:", {
    form: contactForm,
    firstName: firstNameInput,
    lastName: lastNameInput,
});
```

**What Were My Mistakes**

1) I initially typed getElementById('firstName') but my HTML element had id="first-name" (with a hyphen).

```
//store refrence to all form elements

const contactForm = document.getElementById("contactForm");
const firstNameInput = document.getElementById("firstName");
const lastNameInput = document.getElementById("lastName");
const emailInput = document.getElementById("email");
const phoneInput = document.getElementById("phone");
const subjectInput = document.getElementById("subject");
const messgeTextarea = document.getElementById("message");
const clearButton = document.getElementById("clearButton");
const successMessage = document.getElementById("successMessage"
const successNameSpan = document.getElementById("successName");
const charCount = document.getElementById("charCount");

console.log("All form elements referenced.:", {
    form: contactForm,
    firstName: firstNameInput,
    lastName: lastNameInput,
});
```

```
//store refrence to all form elements

const contactForm = document.getElementById("contact-form");
const firstNameInput = document.getElementById("first-name")
const lastNameInput = document.getElementById("last-name");
const emailInput = document.getElementById("email");
const phoneInput = document.getElementById("phone");
const subjectInput = document.getElementById("subject");
const messgeTextarea = document.getElementById("message");
const clearButton = document.getElementById("reset-button");
const successMessage = document.getElementById("success-mess
const successNameSpan = document.getElementById("successName
const charCount = document.getElementById("char-counter");

console.log("All form elements referenced.:", {
    form: contactForm,
    firstName: firstNameInput,
    lastName: lastNameInput,
});
```

2) I forgot to put the error span elements in my HTML so my getElementById() calls returned null.
3) I tried to access an element before it was created in the DOM.

**How I Fixed These Mistakes**

- I made sure that all of my id attributes in my HTML matched my .js file.
- I added the missing error spans in my HTML structure.
- I moved my element reference creation to the top of the file.

**The biggest challenge I faced**

was learning about event handling. When I first learned about it, I was confused because I didn't know how all of the pieces fit together. Specifically:

When does the code inside an event listener get executed?
What does the "event" argument mean?
What is the difference between using arrow functions versus traditional functions when writing an event handler?

**For a Fresh Start: How To Approach Differently**

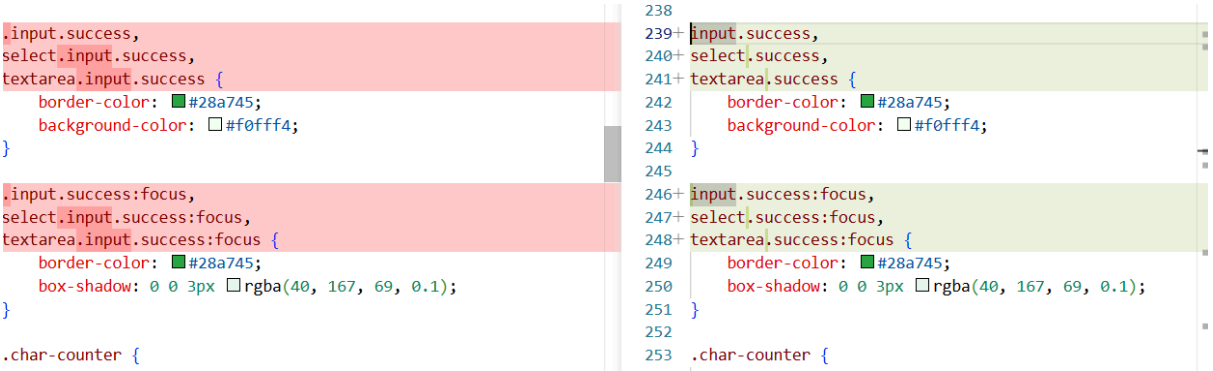
Test Automation

The majority of my testing was done manually by filling out the form numerous times. I would like to gain knowledge of how to create automated test cases to allow me to easily verify that my application is functioning correctly simultaneously.

```
//test validateForm function
console.log("testing validateForm()");
firstNameInput.value = "Nabil";
lastNameInput.value = "Smith";
emailInput.value = "nabil.smith@example.com";
messgeTextarea.value = "Hello, this is a valid message with more than twenty characters.";
phoneInput.value = "+1 (555) 123-4567";
console.log("All valid form test:", validateForm()); //should be true
firstNameInput.value = "A"; //invalid first name
console.log("Invalid first name test:", validateForm()); //should be false
firstNameInput.value = "Nabil"; //valid first name
emailInput.value = "invalid-email"; //invalid email
console.log("Invalid email test:", validateForm()); //should be false
```

Frequent Commits

I was sometimes engaged in coding for hours before creating any commits. By committing to smaller increments more frequently, I would have significantly simplified the process of undoing mistakes.



```
.input.success,
select.input.success,
textarea.input.success {
    border-color: #28a745;
    background-color: #f0fff4;
}

.input.success:focus,
select.input.success:focus,
textarea.input.success:focus {
    border-color: #28a745;
    box-shadow: 0 0 3px rgba(40, 167, 69, 0.1);
}

.char-counter {
```

```
238
239+ input.success,
240+ select.success,
241+ textarea.success {
242      border-color: #28a745;
243      background-color: #f0fff4;
244  }
245
246+ input.success:focus,
247+ select.success:focus,
248+ textarea.success:focus {
249      border-color: #28a745;
250      box-shadow: 0 0 3px rgba(40, 167, 69, 0.1);
251  }
252
253  .char-counter {
```