Javascript for web development, HKR
Elin Eugenia Nilsson
Email: elin_eugenia.nilsson0168@stud.hkr.se
GitHub-link: https://github.com/webbelin/HKR-lab2

Reflections, Lab 2

JavaScript, like HTML and CSS, sticks with me better through practice than by just reading. I learn best by tackling problems and searching for solutions. It's more effective for me to engage with a challenge and then figure out how to solve it.

I've found that W3Schools, ChatGPT, and using "console.log" a lot have been really helpful in moving forward. I prefer W3Schools over MDN, for example, because I find MDN too cluttered. Like ChatGPT, W3Schools keeps things simple and provides clear explanations.

ChatGPT is also a great tool for debugging code. A couple of times, I got stuck because I missed adding a simple ".value" to my code. Instead, I got an empty string and couldn't understand why.

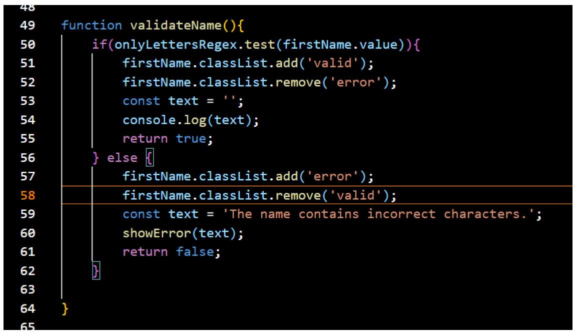
I also did not get the error messages to print, only because I didn't use preventDefault() before running all validation functions. Thanks to ChatGPT, I discovered the importance of adding preventDefault.

One of the hardest things was showing the error message below the right input field while using just one function for all errors. At first, all the error messages ended up in the same place because I didn't know how to target the right error <p>-tag. Later, I realized I could use the input field's id to target the correct error <p>-tag.

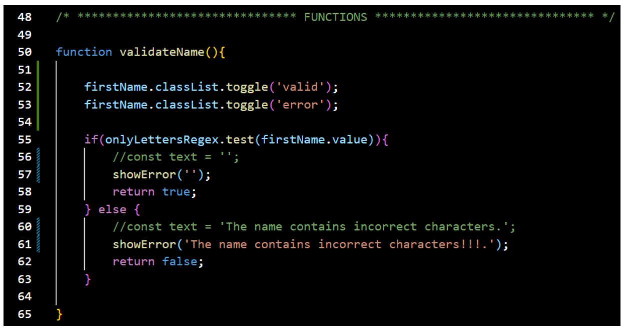
I only worked with one branch in Git. I committed often, whenever small pieces were done, and always made sure to push up a working flow in case I wanted to experiment with my local code. This ensured that if I messed things up, I could do a "pull" and get back to a working version.The goal has been to use Commit Message Prefixes with every commit.

Screenshot 1 - Toggle vs Add/Remove

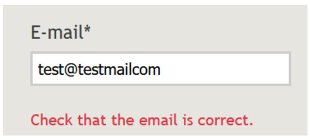
I wanted to minimize the number of lines of code, so I thought I could replace add/remove with toggle and move it outside the if-statement to reduce it from four lines to two in each validation function. Later, I realized there was an issue with this because it meant the border around the box would change color regardless of whether the content in the box had been corrected or not. Therefore, I later went back to using add/remove again.



```
function validateName(){
    if(onlyLettersRegex.test(firstName.value)){
        firstName.classList.add('valid');
        firstName.classList.remove('error');
        const text = '';
        console.log(text);
        return true;
    } else {
        firstName.classList.add('error');
        firstName.classList.remove('valid');
        const text = 'The name contains incorrect characters.';
        showError(text);
        return false;
    }

}
```



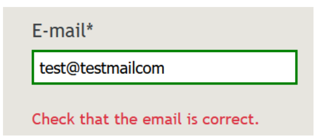
```
/* ****************************** FUNCTIONS ****************************** */

function validateName(){

    firstName.classList.toggle('valid');
    firstName.classList.toggle('error');

    if(onlyLettersRegex.test(firstName.value)){
        //const text = '';
        showError('');
        return true;
    } else {
        //const text = 'The name contains incorrect characters.';
        showError('The name contains incorrect characters!!!.');
        return false;
    }

}
```



E-mail*

test@testmailcom

Check that the email is correct.



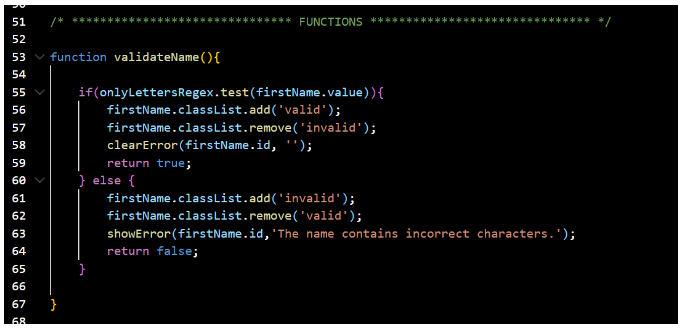
E-mail*

test@testmailcom

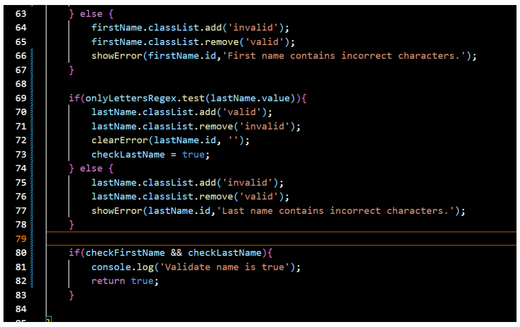
Check that the email is correct.

Screenshot 2 - validating first- & last name

When validating the name, I initially focused on making sure the first name was fully functional and validated. Later, I realized that simply copying and pasting the code for the last name wouldn't work, since I had used a return statement inside the conditional (if/else). I got stuck for a while, but with help from ChatGPT, I was able to debug the issue and come up with a solution for validating multiple fields with the same regex conditions within a single function.



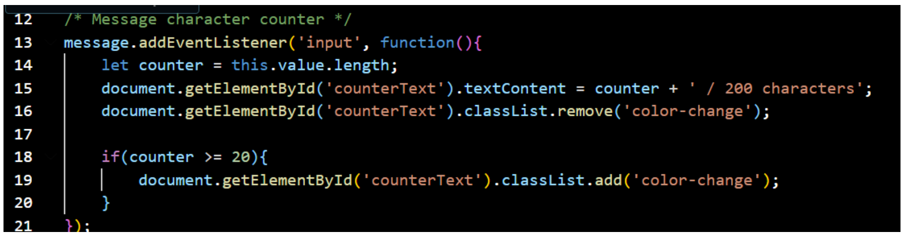
```
/* ****************************** FUNCTIONS ****************************** */

function validateName(){

    if(onlyLettersRegex.test(firstName.value)){
        firstName.classList.add('valid');
        firstName.classList.remove('invalid');
        clearError(firstName.id, '');
        return true;
    } else {
        firstName.classList.add('invalid');
        firstName.classList.remove('valid');
        showError(firstName.id,'The name contains incorrect characters.');
        return false;
    }
}
```



```
    } else {
        firstName.classList.add('invalid');
        firstName.classList.remove('valid');
        showError(firstName.id,'First name contains incorrect characters.');
    }

    if(onlyLettersRegex.test(lastName.value)){
        lastName.classList.add('valid');
        lastName.classList.remove('invalid');
        clearError(lastName.id, '');
        checkLastName = true;
    } else {
        lastName.classList.add('invalid');
        lastName.classList.remove('valid');
        showError(lastName.id,'Last name contains incorrect characters.');
    }

    if(checkFirstName && checkLastName){
        console.log('Validate name is true');
        return true;
    }

```

Screenshot 3 - keyup / input

One of the last things I changed was when I noticed that the character counter didn't update when I pasted 20+ characters into the "message" box using the mouse. To work around that and cover all cases, I learned that I could use "input" instead of "keyup" as the trigger.



```
/* Message character counter */
message.addEventListener('input', function(){
    let counter = this.value.length;
    document.getElementById('counterText').textContent = counter + ' / 200 characters';
    document.getElementById('counterText').classList.remove('color-change');

    if(counter >= 20){
        document.getElementById('counterText').classList.add('color-change');
    }
});
```