

Zaynab Mohamud

669382

## Procedural programming

### Procedural HTML

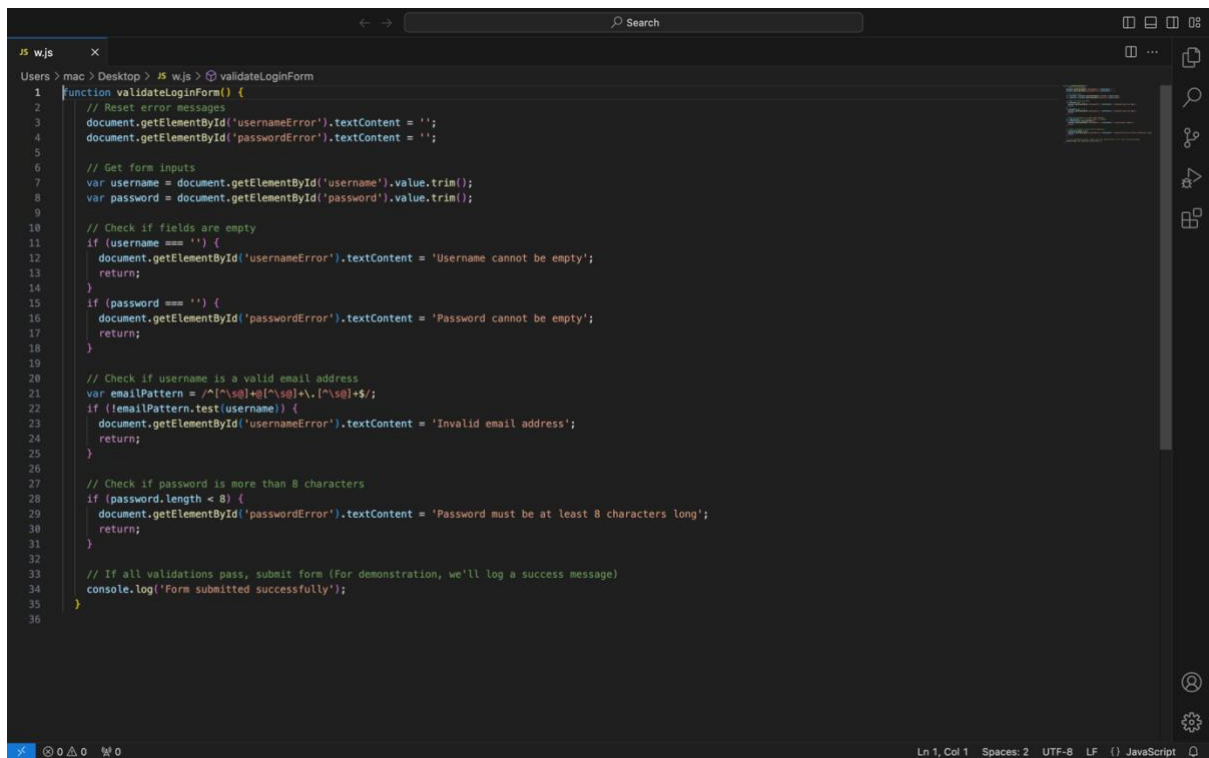
```
Users > mac > Desktop > w.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Login</title>
7 <link rel="stylesheet" href="w.css">
8 </head>
9 <body>
10 <div class="container">
11 <form id="loginForm">
12 <h2>Login</h2>
13 <div class="input-group">
14 <label for="username">Username:</label>
15 <input type="text" id="username" name="username">
16 <span class="error" id="usernameError"></span>
17 </div>
18 <div class="input-group">
19 <label for="password">Password:</label>
20 <input type="password" id="password" name="password">
21 <span class="error" id="passwordError"></span>
22 </div>
23 <button type="button" onclick="validateLoginForm()">Create account</button>
24 </form>
25 </div>
26 <script src="w.js"></script>
27 </body>
28 </html>
29
```

### ProceduralCSS

```
Users > mac > Desktop > # w.css > body
1 body {
2 font-family: Arial, sans-serif;
3 color: #ccc;
4 background-color: #black;
5 }
6
7 .container {
8 max-width: 400px;
9 margin: 50px auto;
10 background-color: #beige;
11 padding: 50px;
12 border: 1px solid #f8f3f3;
13 border-radius: 5px;
14 color: #black;
15 }
16
17 .input-group {
18 margin-bottom: 20px;
19 }
20
21 label {
22 display: block;
23 margin-bottom: 5px;
24 }
25
26 input {
27 width: 100%;
28 padding: 8px;
29 border: 1px solid #ccc;
30 border-radius: 4px;
31 }
32
33 .error {
34 color: #red;
35 font-size: 0.8em;
36 display: block;
37 margin-top: 5px;
38 }
39
40 button {
41 padding: 10px 20px;
42 background-color: #gold;
43 color: #black;
44 border: none;

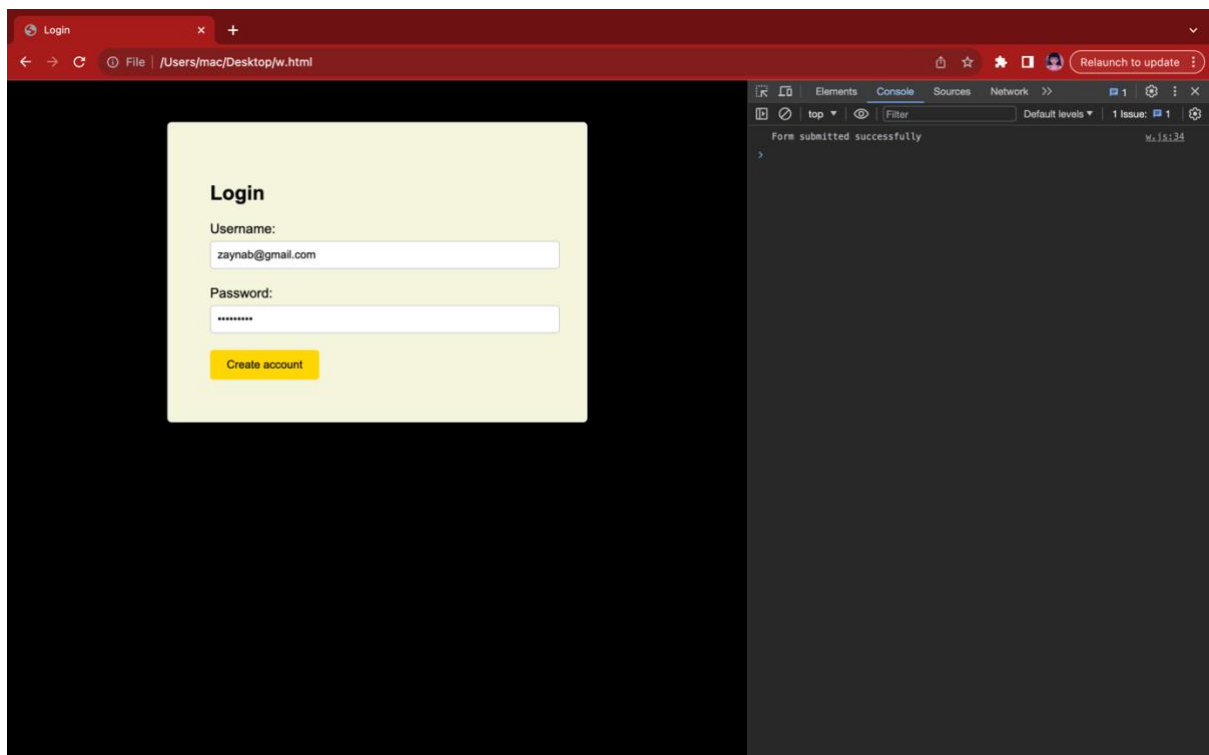
```

## Procedural JS

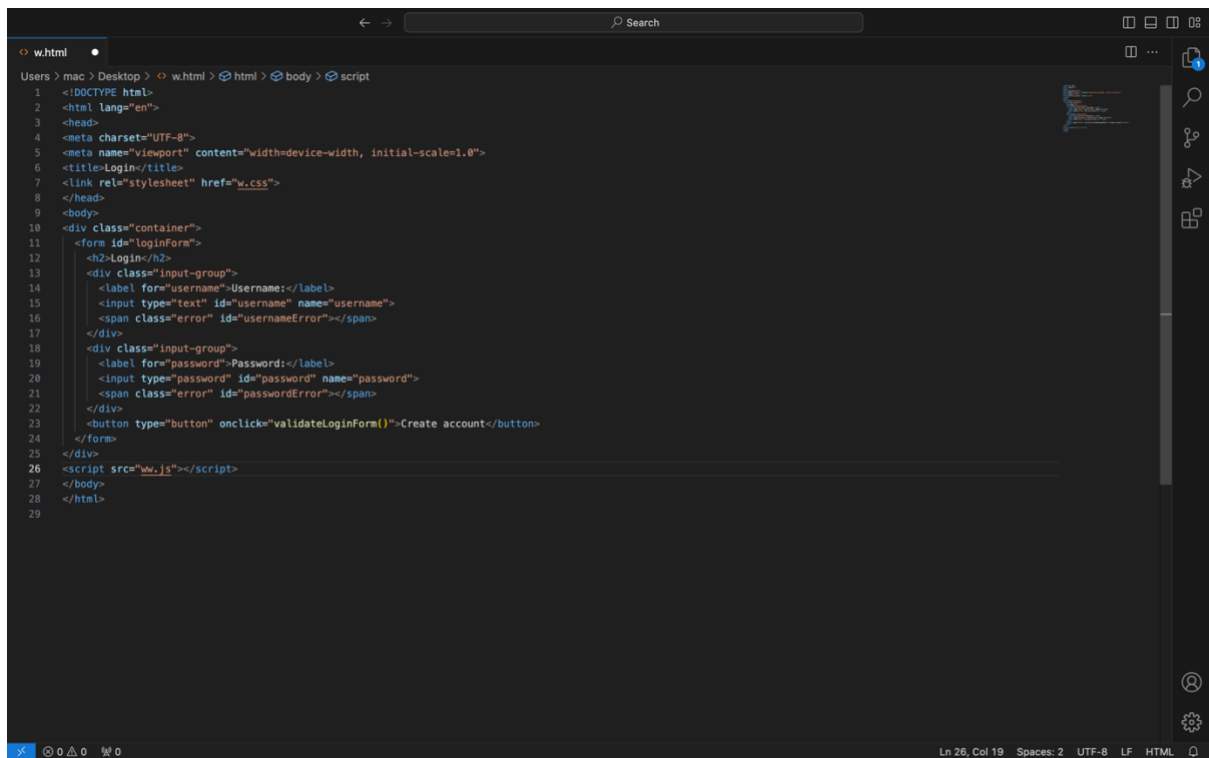


```
1 function validateLoginForm() {
2   // Reset error messages
3   document.getElementById('usernameError').textContent = '';
4   document.getElementById('passwordError').textContent = '';
5
6   // Get form inputs
7   var username = document.getElementById('username').value.trim();
8   var password = document.getElementById('password').value.trim();
9
10  // Check if fields are empty
11  if (username === '') {
12    document.getElementById('usernameError').textContent = 'Username cannot be empty';
13    return;
14  }
15  if (password === '') {
16    document.getElementById('passwordError').textContent = 'Password cannot be empty';
17    return;
18  }
19
20  // Check if username is a valid email address
21  var emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
22  if (!emailPattern.test(username)) {
23    document.getElementById('usernameError').textContent = 'Invalid email address';
24    return;
25  }
26
27  // Check if password is more than 8 characters
28  if (password.length < 8) {
29    document.getElementById('passwordError').textContent = 'Password must be at least 8 characters long';
30    return;
31  }
32
33  // If all validations pass, submit form (For demonstration, we'll log a success message)
34  console.log('Form submitted successfully');
35 }
36
```

## Procedural Console



## OPP-HTML

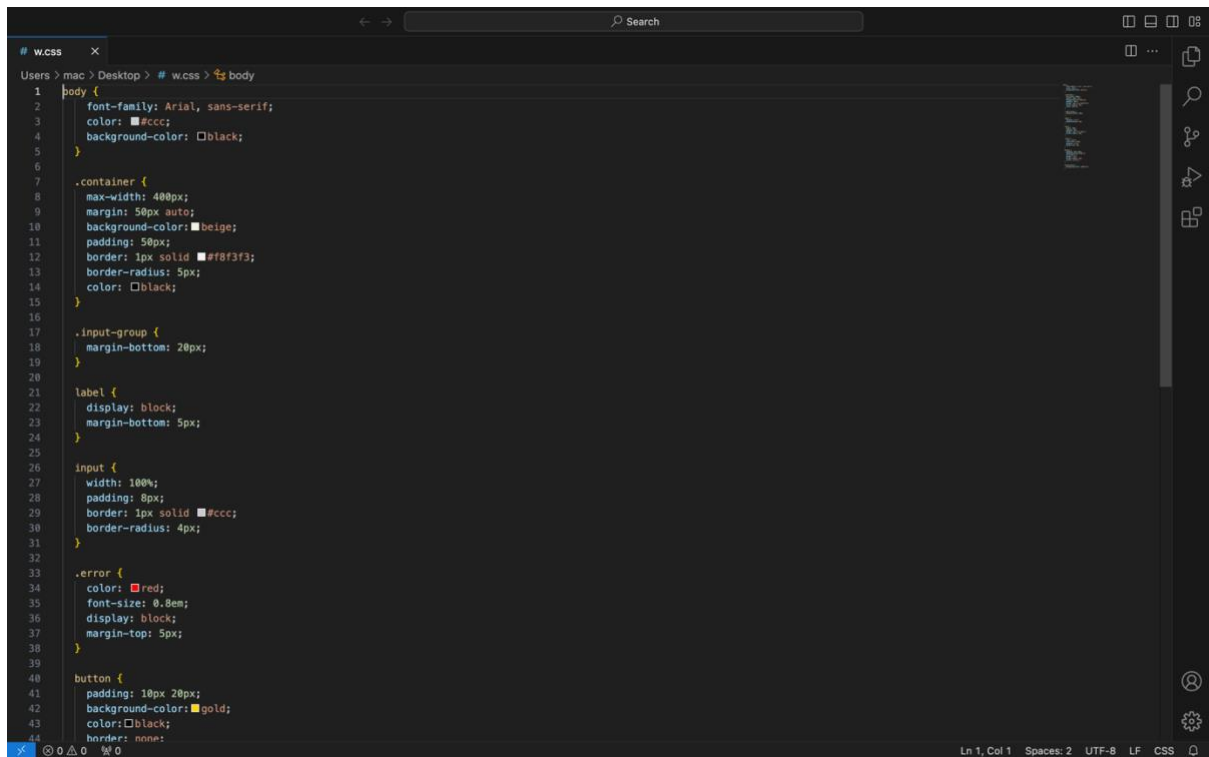


The screenshot shows a code editor with a dark theme. The file 'w.html' is open, and the breadcrumb navigation at the top indicates the path: Users > mac > Desktop > w.html > html > body > script. The HTML code is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Login</title>
7 <link rel="stylesheet" href="w.css">
8 </head>
9 <body>
10 <div class="container">
11 <form id="loginForm">
12 <h2>Login</h2>
13 <div class="input-group">
14 <label for="username">Username:</label>
15 <input type="text" id="username" name="username">
16 <span class="error" id="usernameError"></span>
17 </div>
18 <div class="input-group">
19 <label for="password">Password:</label>
20 <input type="password" id="password" name="password">
21 <span class="error" id="passwordError"></span>
22 </div>
23 <button type="button" onclick="validateLoginForm()">Create account</button>
24 </form>
25 </div>
26 <script src="ww.js"></script>
27 </body>
28 </html>
29
```

The status bar at the bottom shows 'Ln 26, Col 19', 'Spaces: 2', 'UTF-8', 'LF', and 'HTML'.

## OPP-CSS



The screenshot shows a code editor with a dark theme. The file 'w.css' is open, and the breadcrumb navigation at the top indicates the path: Users > mac > Desktop > w.css > body. The CSS code is as follows:

```
1 body {
2   font-family: Arial, sans-serif;
3   color: #ccc;
4   background-color: #000;
5 }
6
7 .container {
8   max-width: 400px;
9   margin: 50px auto;
10  background-color: #f0f0f0;
11  padding: 50px;
12  border: 1px solid #8f8f8f;
13  border-radius: 5px;
14  color: #000;
15 }
16
17 .input-group {
18   margin-bottom: 20px;
19 }
20
21 label {
22   display: block;
23   margin-bottom: 5px;
24 }
25
26 input {
27   width: 100%;
28   padding: 8px;
29   border: 1px solid #ccc;
30   border-radius: 4px;
31 }
32
33 .error {
34   color: #f00;
35   font-size: 0.8em;
36   display: block;
37   margin-top: 5px;
38 }
39
40 button {
41   padding: 10px 20px;
42   background-color: #f0f0f0;
43   color: #000;
44   border: none;
```

The status bar at the bottom shows 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', and 'CSS'.

## OPP-JS

```
JS ww.js x
Users > mac > Desktop > JS ww.js > ...
1  const Validator = {
2    REQUIRED: 'REQUIRED',
3    MIN_LENGTH: 'MIN_LENGTH',
4
5    validate(value, flag, validatorValue) {
6      if (flag === this.REQUIRED) {
7        return value.trim().length > 0;
8      }
9      if (flag === this.MIN_LENGTH) {
10       return value.trim().length >= validatorValue;
11     }
12     return false;
13   }
14 };
15
16 function User(username, password) {
17   this.username = username;
18   this.password = password;
19 }
20
21 User.prototype.greet = function() {
22   console.log('Hi, I am ${this.username}');
23 };
24
25
26 function signupHandler(event) {
27   event.preventDefault();
28
29   const enteredUsername = document.getElementById('username').value;
30   const enteredPassword = document.getElementById('password').value;
31
32   if (
33     !Validator.validate(enteredUsername, Validator.REQUIRED) ||
34     !Validator.validate(enteredPassword, Validator.REQUIRED) ||
35     !Validator.validate(enteredPassword, Validator.MIN_LENGTH, 6)
36   ) {
37     alert('Invalid input - username or password is wrong (password should be at least 6 characters long).');
38     return;
39   }
40
41   const newUser = new User(enteredUsername, enteredPassword);
42   newUser.greet();
43   console.log(newUser);
44 }
```

## OPP-Console

file:///Users/mac/Desktop/w.html

SWISH - SWI-Prolog for SHaring

Login

Username:  
Zeynab@gmail

Password:  
\*\*\*\*\*

Create account

Console opened at 08:57:09

ReferenceError: Can't find variable: validateLoginForm  
onclick - w.html:23

Selected Element  
<html lang="en"></html> = \$1