Client-side form validation using JavaScript:

Client-side form validation using JavaScript is the process of validating form data before it is submitted to the server. This is done by checking the data for validity and displaying error messages to the user if any errors are found.

There are several reasons why it is important to perform client-side form validation:

- It provides immediate feedback to the user, so they can correct any errors before submitting the form.
- It can reduce the load on the server by preventing invalid form data from being submitted.
- It can improve the security of the application by preventing malicious users from submitting invalid data.

To implement client-side form validation using JavaScript, you can use the following steps:

1. **Add event listeners to the form elements.** You can use the addEventListener() method to add event listeners to the form elements that you want to validate. For example, you could add an event listener to the submit event of the form element to validate the form data before it is submitted.
2. **Write validation functions.** You need to write validation functions to check the form data for validity. For example, you could write a function to check if the email address is valid or if the password is at least 8 characters long.
3. **Display error messages.** If any errors are found, you need to display error messages to the user. You can use the alert() function or the document.querySelector() method to display error messages.

Here is an example of a simple client-side form validation using JavaScript:

```
const form = document.getElementById('form');

const username = document.getElementById('username');

const email = document.getElementById('email');

const password = document.getElementById('password');

const password2 = document.getElementById('password2');
```

```
form.addEventListener('submit', e =>

{

    e.preventDefault();

    checkInputs();

});



function checkInputs()

{

    // trim to remove the whitespaces

    const usernameValue = username.value.trim();

    const emailValue = email.value.trim();

    const passwordValue = password.value.trim();

    const password2Value = password2.value.trim();



    if(usernameValue === '')

        setErrorFor(username, 'Username cannot be blank');

    else

        setSuccessFor(username);
```

```
    if(emailValue === '')

        setErrorFor(email, 'Email cannot be blank');

    else if (!isEmail(emailValue))

        setErrorFor(email, 'Not a valid email');

       else

        setSuccessFor(email);




    if(passwordValue === '')

        setErrorFor(password, 'Password cannot be blank');

    else

        setSuccessFor(password);




    if(password2Value === '')

        setErrorFor(password2, 'Password2 cannot be blank');

    else if(passwordValue !== password2Value)

        setErrorFor(password2, 'Passwords does not match');

       else

         setSuccessFor(password2);
```

```
}



function setErrorFor(input, message)

{

    const formControl = input.parentElement;

    const small = formControl.querySelector('small');

    formControl.className = 'form-control error';

    small.innerText = message;

}



function setSuccessFor(input)

{

    const formControl = input.parentElement;

    formControl.className = 'form-control success';

}



function isEmail(email)

{

    return
/^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[
0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}])|(([a-zA-Z\-0-9]+\.)+
[a-zA-Z]{2,}))$/.test(email);
```

```
}
```

Line 1-5: This code declares four constants: form, username, email, password, and password2. These constants store references to the form element and the four input fields on the form.

Line 7-10: This code adds an event listener to the form element. The event listener will be executed when the form is submitted.

Line 12-24: This function validates the form data. It trims the whitespace from each input field and then checks to make sure that each field is not empty. If a field is empty, the function displays an error message and sets the className of the field to error.

The function also uses the isEmail() function to validate the email address. If the email address is not valid, the function displays an error message and sets the className of the email input field to error.

If all of the form data is valid, the function sets the className of each input field to success.

Line 26-31: This function sets the className of the input field to error and displays the specified error message.

Line 33-35: This function sets the className of the input field to success.

Line 37-43: This function validates the email address using a regular expression. The regular expression matches the following pattern:

^(([^<>()[]\.,;:\s@"]+(.[^<>()[]\.,;:\s@"]+)*)|(".+"))@(([[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}])|(([a-zA-Z-0-9]+.)+[a-zA-Z]{2,}))$

If the email address matches the pattern, the function returns true. Otherwise, the function returns false.

Example:

To use the code, you would first need to add it to your HTML file. Then, you would need to add the following HTML to your form:

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Form Validation</title>
    <!-- <link rel="preconnect"
href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com"
crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;7
00&display=swap" rel="stylesheet"> -->
    <link rel="stylesheet" href="./style.css">
    <script defer src="./index.js"></script>
</head>
<body>
    <div class="container">
        <div class="header">
            <h2>Create Account</h2>
        </div>
        <form id="form" class="form">
            <div class="form-control">
                <label for="username">Username</label>
                <input type="text" placeholder="florinpop17"
id="username" />
                <i class="fas fa-check-circle"></i>
                <i class="fas fa-exclamation-circle"></i>
                <small>Error message</small>
            </div>
            <div class="form-control">
                <label for="username">Email</label>
                <input type="email"
placeholder="a@florin-pop.com" id="email" />
                <i class="fas fa-check-circle"></i>
```

```
                <i class="fas fa-exclamation-circle"></i>
                <small>Error message</small>
            </div>
            <div class="form-control">
                <label for="username">Password</label>
                <input type="password" placeholder="Password"
id="password"/>
                <i class="fas fa-check-circle"></i>
                <i class="fas fa-exclamation-circle"></i>
                <small>Error message</small>
            </div>
            <div class="form-control">
                <label for="username">Password check</label>
                <input type="password" placeholder="Password two"
id="password2"/>
                <i class="fas fa-check-circle"></i>
                <i class="fas fa-exclamation-circle"></i>
                <small>Error message</small>
            </div>
            <button>Submit</button>
        </form>
    </div>
</body>
</html>
```

Once you have added the code and the HTML to your file, you can test the form validation by entering some data into the form and then clicking the submit button. If the form data is valid, the form will be submitted. Otherwise, you will see an error message.

style.css

```
body {

    background-color: #9b59b6;
```

```
    font-family: 'Open Sans', sans-serif;
    display: flex;
    align-items: center;
    justify-content: center;
    min-height: 100vh;
    margin: 0;
}

.container {
    background-color: #fff;
    border-radius: 5px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.3);
    overflow: hidden;
    width: 400px;
    max-width: 100%;
}

.header {
    border-bottom: 1px solid #f0f0f0;
```

```
    background-color: #f7f7f7;
    padding: 20px 40px;
}

.header h2 {
    margin: 0;
}

.form {
    padding: 30px 40px;
}

.form-control {
    margin-bottom: 10px;
    padding-bottom: 20px;
    position: relative;
}

.form-control label {
```

```
    display: inline-block;
    margin-bottom: 5px;
}


.form-control input {
    border: 2px solid #f0f0f0;
    border-radius: 4px;
    display: block;
    font-family: inherit;
    font-size: 14px;
    padding: 10px;
    width: 100%;
}


.form-control input:focus {
    outline: 0;
    border-color: #777;
}
```

```
.form-control.success input {

    border-color: #2ecc71;

}



.form-control.error input {

    border-color: #e74c3c;

}



.form-control i {

    visibility: hidden;

    position: absolute;

    top: 40px;

    right: 10px;

}



.form-control.success i.fa-check-circle {

    color: #2ecc71;

    visibility: visible;

}
```

```
.form-control.error i.fa-exclamation-circle {
    color: #e74c3c;
    visibility: visible;
}

.form-control small {
    color: #e74c3c;
    position: absolute;
    bottom: 0;
    left: 0;
    visibility: hidden;
}

.form-control.error small {
    visibility: visible;
}

.form button {
```

```
    background-color: #8e44ad;

    border: 2px solid #8e44ad;

    border-radius: 4px;

    color: #fff;

    display: block;

    font-family: inherit;

    font-size: 16px;

    padding: 10px;

    margin-top: 20px;

    width: 100%;

}
```