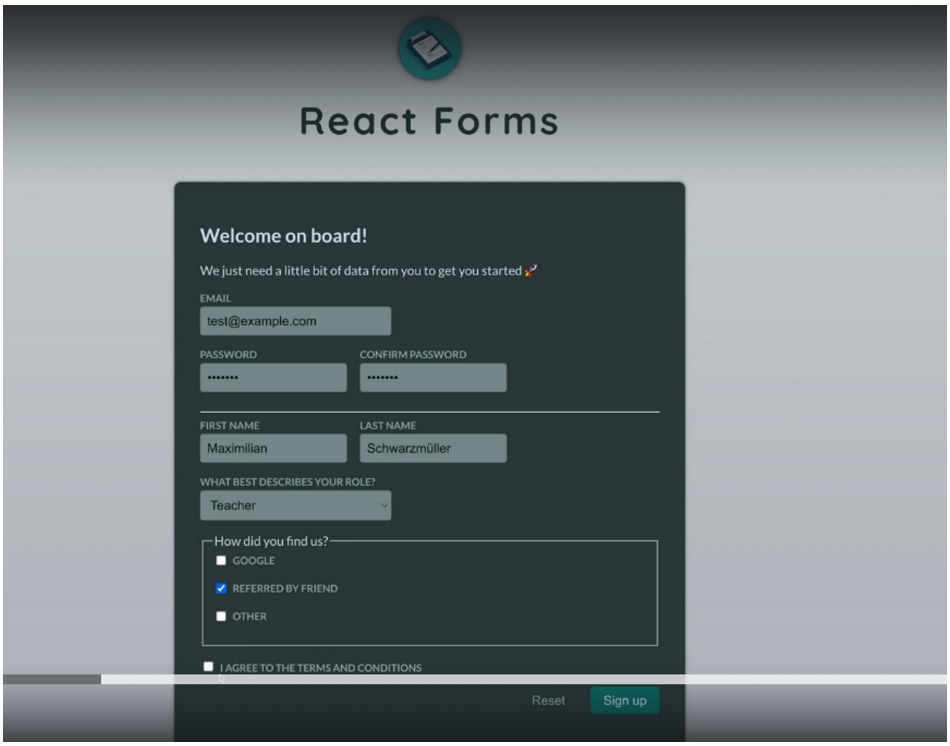
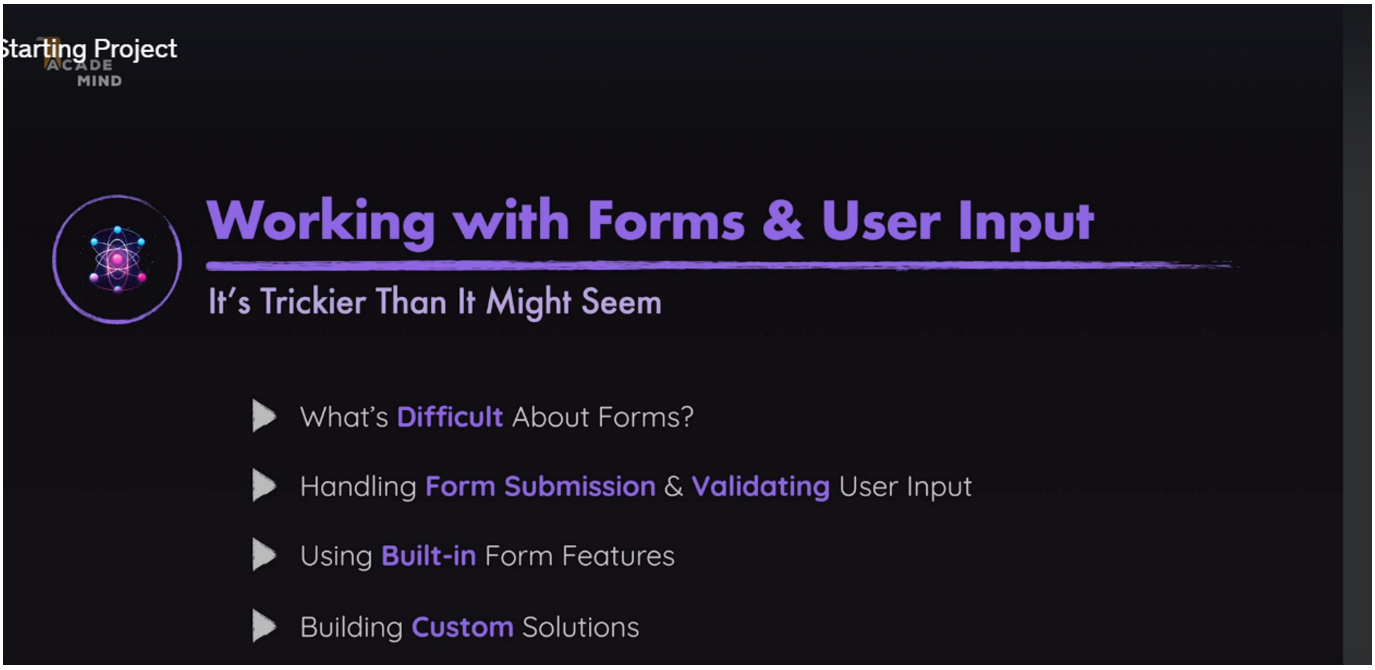
Section 17: Working with Forms & User Input

Saturday, October 5, 2024 1:25 PM





What's So Difficult?

Form Submission

Handling submission is relatively **easy**

Entered values can be managed via **state**

Alternatively, they can be extracted via **refs**

Or via **FormData** and native browser features

Input Validation

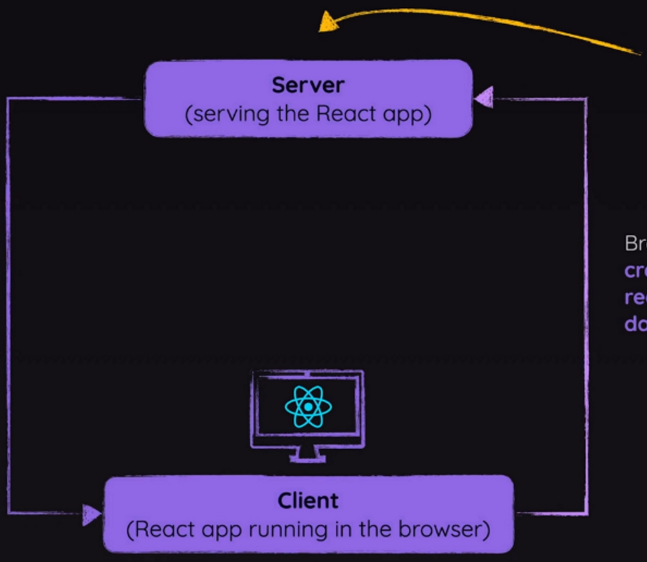
Providing a good user experience is **tricky**

You can **validate** on every **keystroke** → errors may be shown **too early**

You can **validate** on **lost focus** → errors may be shown **too long**

You can **validate** on **form submission** → errors may be shown **too late**

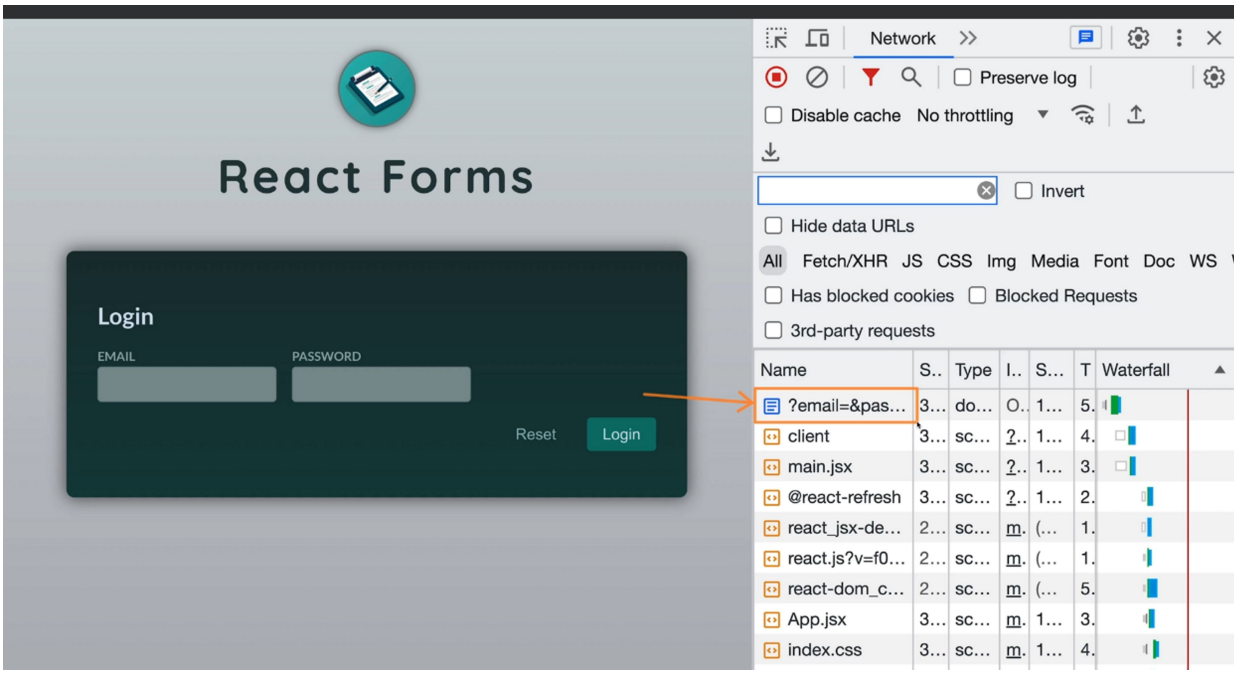
By Default The Browser Sends a HTTP Request



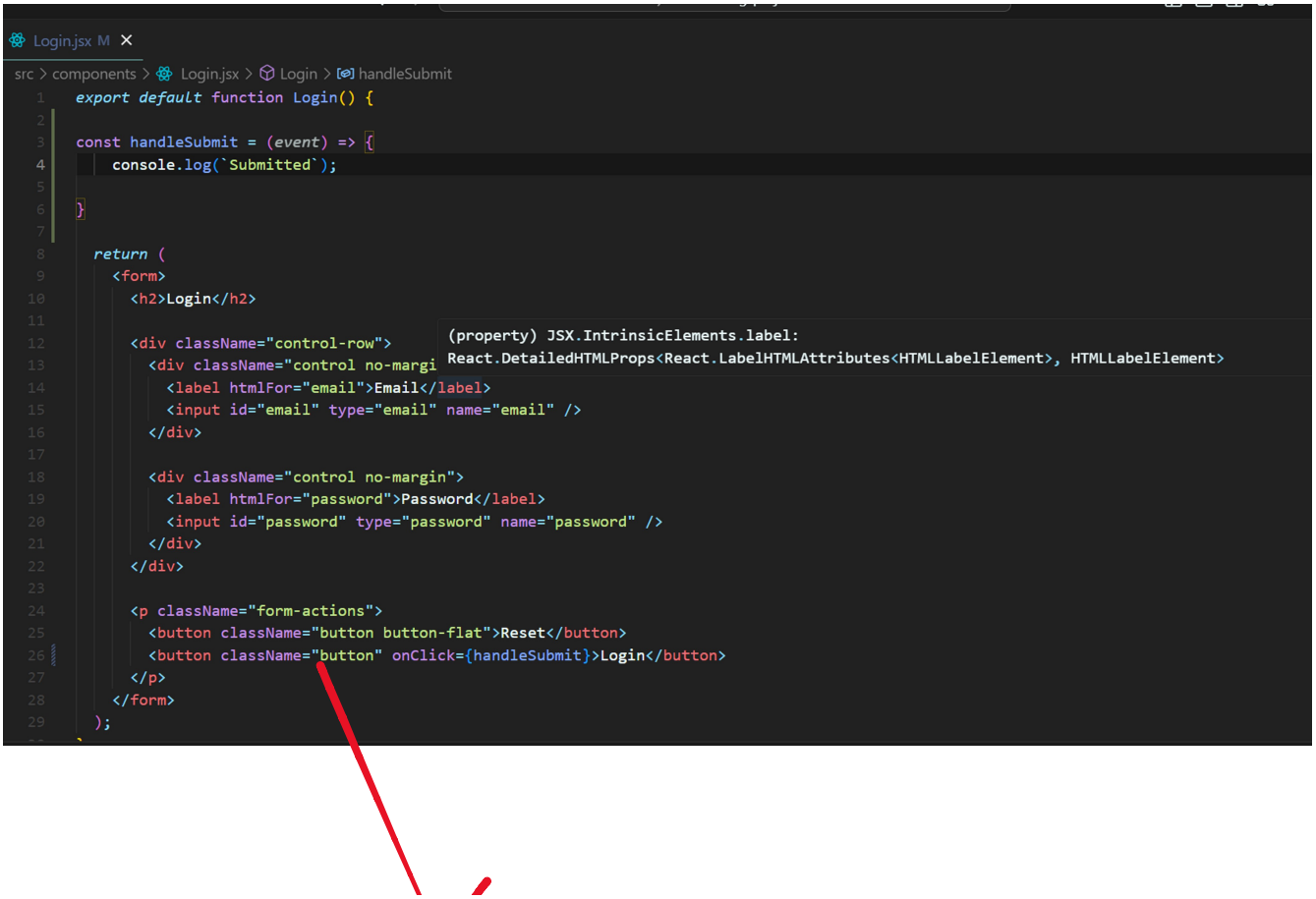
During development, that's the development server started via "npm run dev"

Serves the React app
JavaScript files + index.html (+ any CSS needed) to users visiting the website

Browser **automatically creates & sends a HTTP request** with entered **form data**



By default, button type is submit, it sends a req to the server and our page reloads. That's why we do not see the console.log() output to the console.



Default type is submit, it sends req. to the server.
<button type="submit" />

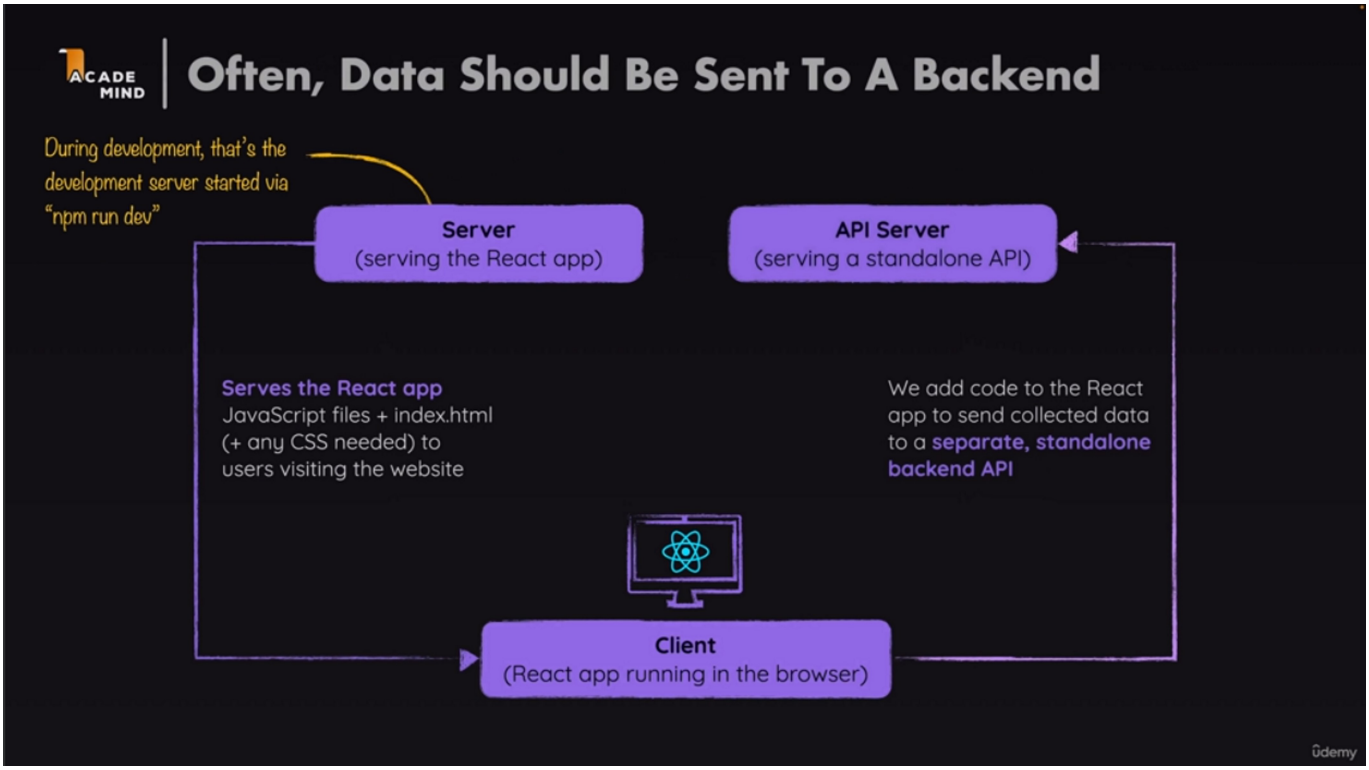
One way of preventing to send req is to add type button

```
    <button type="button" className="button" onClick={handleSubmit}>
      Login
    </button>
  </p>
```

<button type="button" />

Another way is adding,

event.preventDefault();



Resetting Forms

type='reset' actually resets our form.

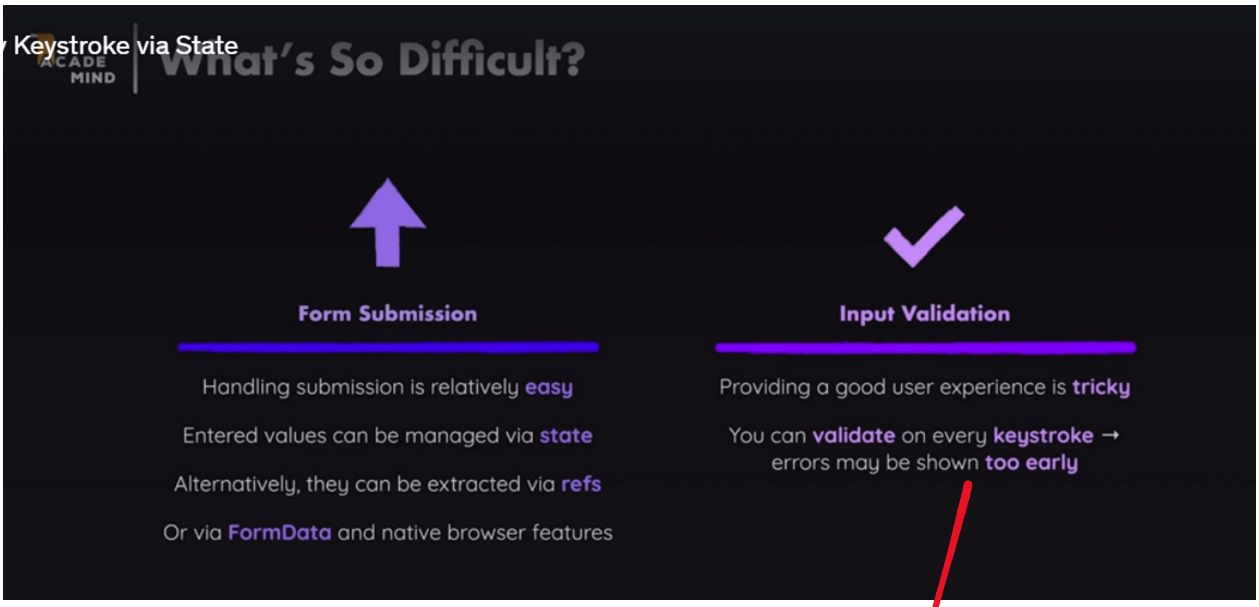
```
        <label htmlFor="terms-and-conditions">
          <input type="checkbox" id="terms-and-conditions" name="terms" />I
          agree to the terms and conditions
        </label>
      </div>

      <p className="form-actions">
        <button type="reset" className="button button-flat">
          Reset
        </button>
        <button type="submit" className="button">
          Sign up
        </button>
      </p>
    </form>
  );
```

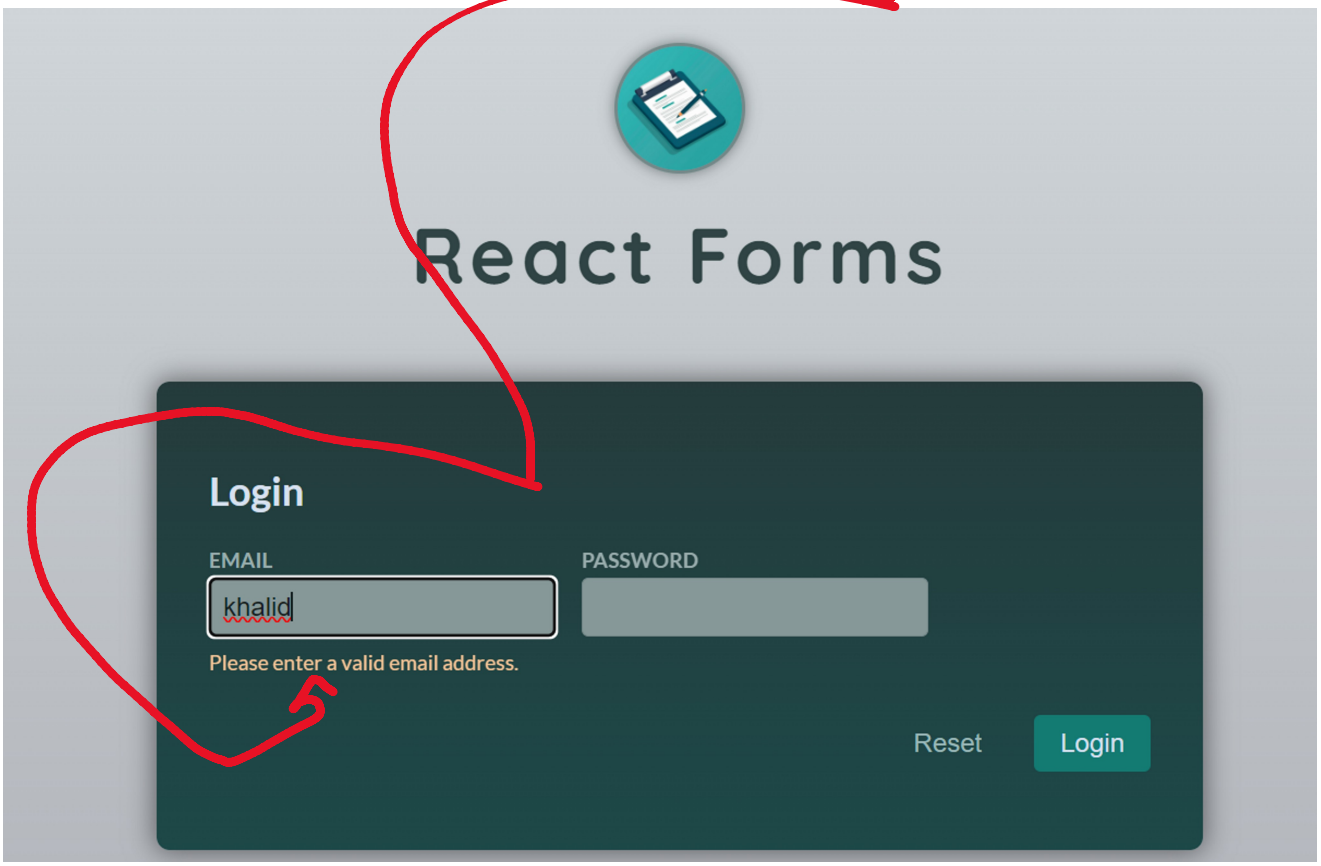
Or we can write like this way,

Signup.jsx M | StateLogin.jsx | Login.jsx

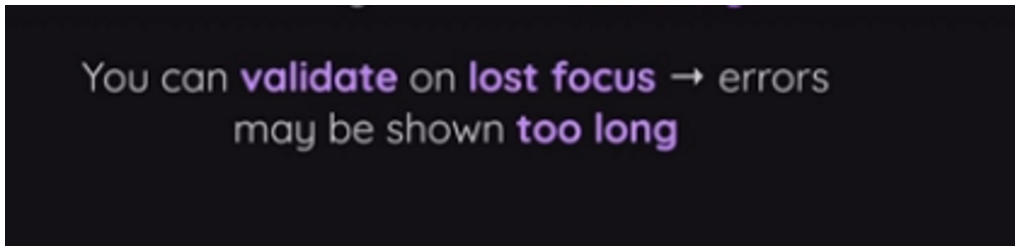
```
export default function Signup() {
  function handleSubmit(event) {
    event.preventDefault();

    const fd = new FormData(event.target);
    const acquisitionChannel = fd.getAll('acquisition')
    const data = Object.fromEntries(fd.entries());
    data.acquisition = acquisitionChannel;
    console.log(data);

    event.target.reset();
  }

  return (
```

Validating Input on Every Keystroke via State

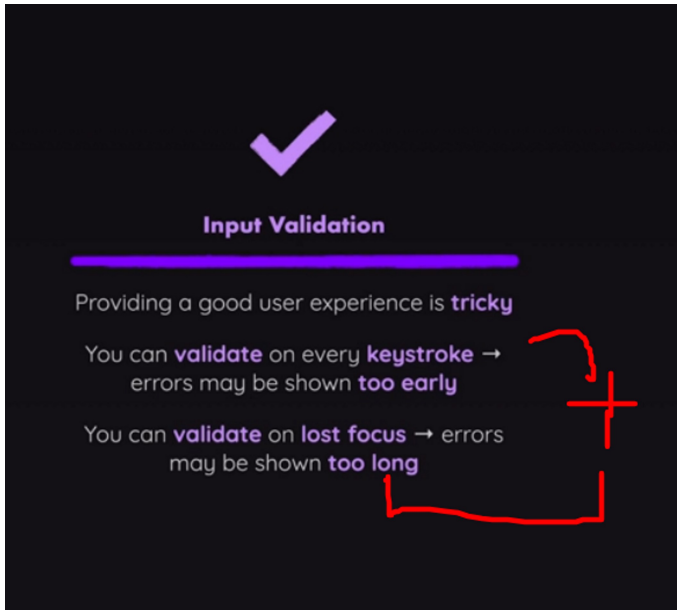




Validating Input Upon Lost Focus (Blur)



Email field theke type kore focus sorale error ashbe. Then abr type korte thakle o error thekey jabe.



So, we can combine on every keystroke + last focus

Using Third-Party Form Libraries