

Theory 08

React Form, Hook Form, Yup

WEEK 02

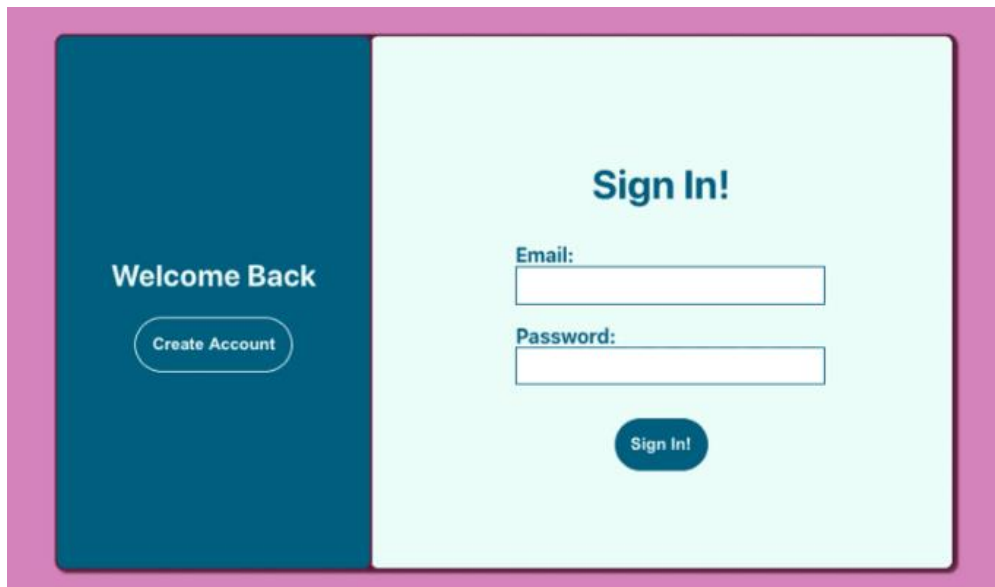
Nội dung chính

- | | |
|----------------------------------|----|
| ❑ THAO TÁC FORM TRONG REACTJS | 03 |
| ❑ XỬ LÝ DỮ LIỆU TRÊN FORM | 05 |
| ❑ THAO TÁC SUBMIT FORM | 07 |
| ❑ THAO TÁC MULTIPLE INPUT FIELDS | 08 |
| ❑ THAO TÁC FORM VALIDATION | 09 |
| ❑ THƯ VIỆN REACT HOOK FORM, YUP | 15 |



Thao tác Form trong ReactJS

- ❑ Giống như HTML, React sử dụng Form để cho phép người dùng tương tác với trang Web: tìm kiếm, nhập thông tin liên lạc, đăng nhập, đăng ký,...
- ❑ Các phần tử trong Form tương đối đa dạng.
- ❑ Một số phần tử thông dụng như input, checkbox, radio button, submit button,...



The image shows a web form mockup with a pink border. It is divided into two main sections: a dark teal section on the left and a light teal section on the right.

Left Section (Dark Teal):

- Text: "Welcome Back"
- Button: "Create Account" (white text on a rounded teal button)

Right Section (Light Teal):

- Text: "Sign In!" (bold, dark teal)
- Form Fields:
 - "Email:" followed by a white input box with a teal border.
 - "Password:" followed by a white input box with a teal border.
- Button: "Sign In!" (white text on a dark teal rounded button)

Thao tác Form trong ReactJS

❑ Ví dụ thêm một Form với ReactJS:

```
import React from 'react';  
const MyForm = () => {  
  return (  
    <form>  
      <h1>Hello</h1>  
      <p>Enter your name:</p>  
      <input type="text"/>  
    </form>;)  
}  
export default MyForm;
```

Xử lý dữ liệu trên Form

- ❑ Là cách xử lý dữ liệu khi giá trị được thay đổi hoặc đã gửi đi
- ❑ Trong **HTML**, dữ liệu của form được quản lý bởi **DOM**.
- ❑ Trong **React**, dữ liệu của form được quản lý bởi các component
- ❑ Khi dữ liệu được xử lý bởi các **component**, tất cả dữ liệu được lưu trữ ở **state**
- ❑ Kiểm soát các thay đổi bằng cách thêm trình xử lý sự kiện trong **onChange**

Bill To / Billing Address

Full Name	John Newman	✓
Street Address	2125 Chestnut st	✓
	optional	
Zip Code	9412	Enter Zip for City & State The specified ZIP is invalid
Phone		
Email		

Send me exclusive offers, deals and expert reviews.

Xử lý dữ liệu trên Form

❑ Ví dụ cập nhật state:

```
import React, {useState} from 'react';  
const MyForm = () => {  
  const [username, setUsername] = useState("");  
  const handleChange = event => setUsername(event.target.value);  
  return (  
    <form>  
      <h1>Hello {username}</h1>  
      <p>Enter your name:</p>  
      <input type="text" value={username} onChange={handleChange}/>  
    </form>;)  
}  
export default MyForm;
```

Thao tác Submit Form

- ❑ Chúng ta có thể kiểm soát hành động gửi dữ liệu Form đi bằng cách thêm trình xử lý sự kiện trong thuộc tính **onSubmit**



React form

Name

Age

Email

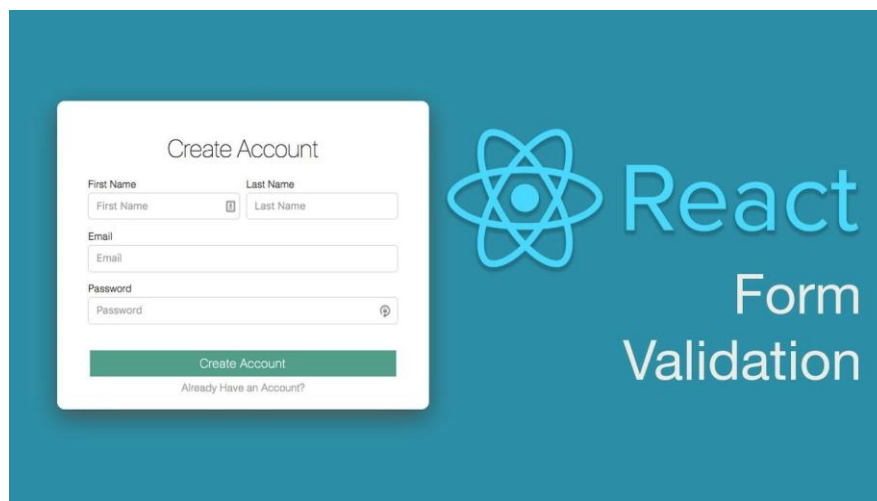
SUBMIT

Thao tác Multiple Input Fields

- ❑ Bạn có thể kiểm soát các giá trị của nhiều đầu vào bằng cách thêm thuộc tính tên vào mỗi phần tử
- ❑ Khi bạn khởi tạo **state** trong hàm tạo, hãy sử dụng tên.
- ❑ Để truy cập các trình xử lý sự kiện, hãy sử dụng cú pháp **event.target.name** và **event.target.value**
- ❑ Để cập nhật **state** trong **Class Component**, chúng ta sử dụng phương thức **this.setState** hoặc sử dụng **setState** trong **Functional Component**

Thao tác Form Validation

- ❑ Trong React, **Form validation** cho phép hiển thị thông báo lỗi nếu người dùng điền không chính xác vào biểu mẫu với loại đầu vào mong đợi
- ❑ Để validate **Form** trong React, tạo ra các hàm **validation** với các quy tắc xác thực
- ❑ Bạn có thể xác thực đầu vào biểu mẫu khi người dùng đang nhập hoặc bạn có thể đợi cho đến khi biểu mẫu được gửi



Thao tác Form Validation

❑ Ví dụ khai báo một component **App.js** với **App.css** như sau:

```
JS App.js M X
practice-03-validate-login-form > src > JS App.js > App
1 import React, { useState } from "react";
2 import "../App.css";
3
4 function App() {
5   > const MESSAGE_ERROR = { ...
8   };
9
10  > const REGEX = { ...
13  };
14
15   const [form, setForm] = useState({});
16
17   > function handleChange(event) { ...
25   }
26
27   > function handleSubmit() { ...
36   }
37
38   > return ( ...
78   );
79
80
81   export default App;

```

```
# App.css M X
practice-03-validate-login-form > src > # App.css > ...
1 label {
2   display: block;
3 }
4
5 .custom-input {
6   padding-bottom: 2px;
7 }
8
9 .custom-input-error label {
10  color: red;
11 }
12
13 .custom-input-error input {
14   border: 2px solid red;
15   border-radius: 4px;
16 }
17
18 .error {
19   color: red;
20   margin: 0;
21 }
22

```

Thao tác Form Validation

❑ Tạo 3 biến **MESSAGE_ERROR**, **REGEX**, **state form** để hiện thực **validate form**:

```
4  function App() {  
5    const MESSAGE_ERROR = {  
6      email: "Email error",  
7      password: "Password error"  
8    };  
9  
10   const REGEX = {  
11     email: /^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$/,  
12     password: /^[a-zA-Z0-9!@#\$%^\&*\\)\(+=._-]{6,}$/  
13   };  
14  
15   const [form, setForm] = useState({});  
16  
17   > function handleChange(event) { ...  
25     }  
26  
27   > function handleSubmit() { ...  
36     }  
37  
38   > return ( ...  
78     );  
79   }
```

- ☐ value
- ☐ event
- ☐ error message

```

return (
  <div>
    <h1>Login</h1>
    <form>
      <div>
        className={`custom-input ${form.email &&
          form.email.error &&
            "custom-input-error"}`}
      >
        <label>Email </label>
        <input
          name="email"
          value={{form.email && form.email.value} || ""}
          onChange={handleChange}
        />
        {form.email && form.email.error && (
          <p className="error">Email error</p>
        )}
      </div>
      <div>
        className={`custom-input ${form.password &&
          form.password.error &&
            "custom-input-error"}`}
      >
        <label>Password </label>
        <input
          type="password"
          name="password"
          value={{form.password && form.password.value} || ""}
          onChange={handleChange}
        />
        {form.password && form.password.error && (
          <p className="error">Password error</p>
        )}
      </div>
      <button type="button" onClick={handleSubmit}>
        Submit
      </button>
    </form>
  </div>
);

```

Thao tác Form Validation

❑ Tạo hàm xử lý **handleChange** để **validate** dữ liệu các trường **input** trên **Fom**:

```
function handleChange(event) {  
  let error = REGEX[event.target.name].test(event.target.value)  
  ? ""  
  : MESSAGE_ERROR[event.target.name];  
  setForm({  
    ...form,  
    [event.target.name]: { value: event.target.value, error: error }  
  });  
}  
  
function handleSubmit() {  
  const isFilled =  
    form.email && form.email.value && form.password && form.password.value;  
  const isError = isFilled && (form.email.error || form.password.error);  
  alert(  
    isFilled && !isError  
    ? "Login in successfully!!!"  
    : "Please fill out all the fields!!!"  
  );  
}
```

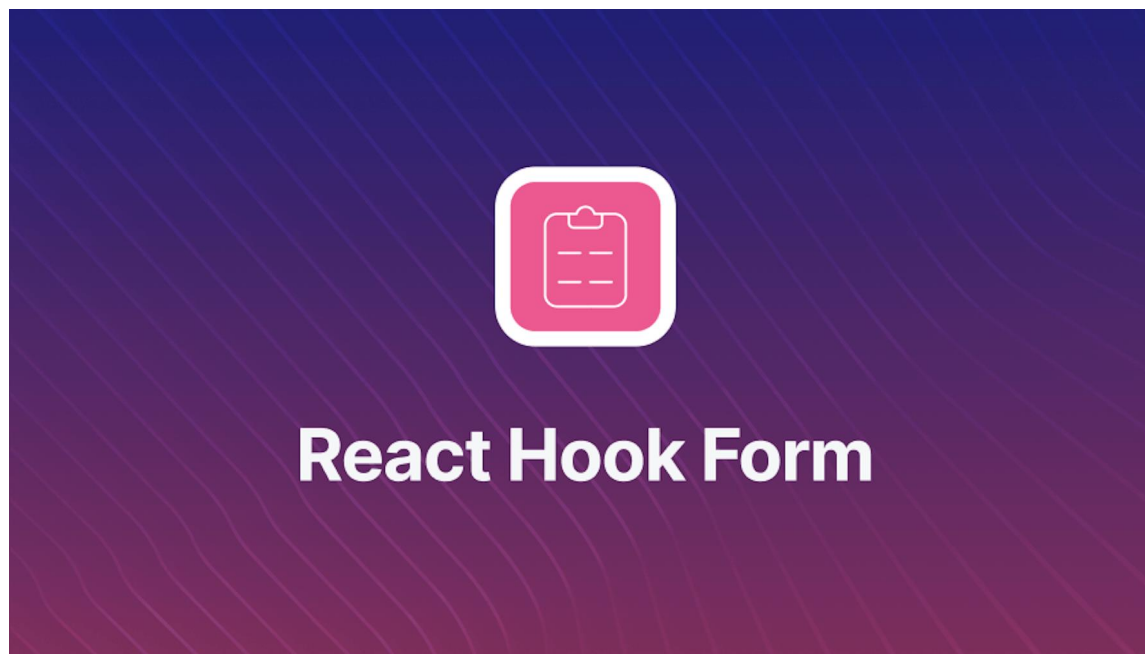
Thao tác Form Validation

❑ Tạo hàm xử lý **handleSubmit** để **submit** các dữ liệu đã điền trên **Form**:

```
function handleChange(event) {  
  let error = REGEX[event.target.name].test(event.target.value)  
  ? ""  
  : MESSAGE_ERROR[event.target.name];  
  setForm({  
    ...form,  
    [event.target.name]: { value: event.target.value, error: error }  
  });  
}  
  
function handleSubmit() {  
  const isFilled =  
    form.email && form.email.value && form.password && form.password.value;  
  const isError = isFilled && (form.email.error || form.password.error);  
  alert(  
    isFilled && !isError  
    ? "Login in successfully!!!"  
    : "Please fill out all the fields!!!"  
  );  
}
```

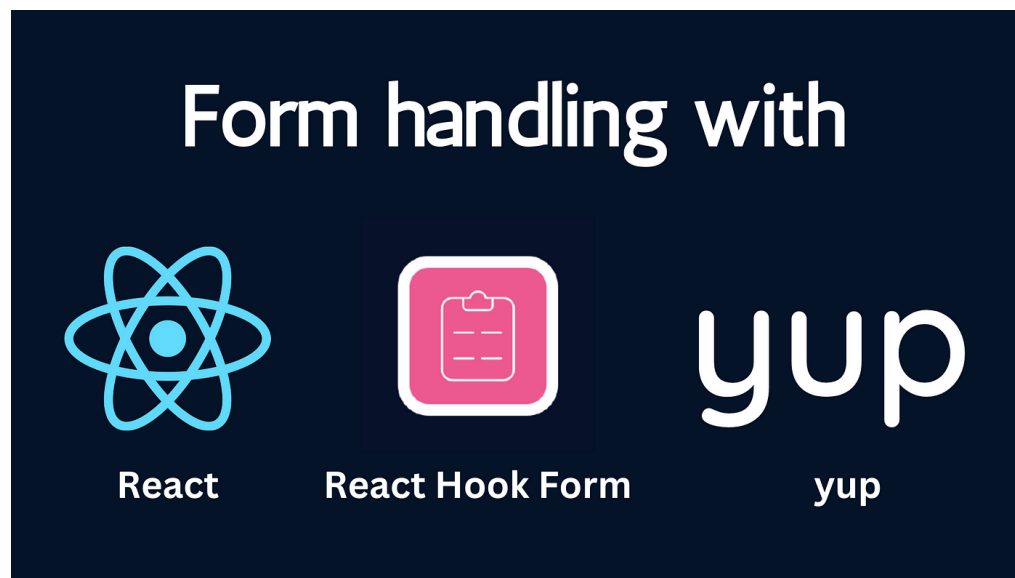
Thư viện React Hook Form, Yup

- ❑ **React Hook Form (RHF)**: thư viện giúp quản lý Form trong React theo cách dễ dàng, tối ưu hiệu năng, ít re-render.
- ❑ **React Hook Form** hỗ trợ tốt cho TypeScript, Validation và hiệu suất cao.



Thư viện React Hook Form, Yup

- ❑ **Yup**: thư viện schema validation mạnh mẽ, dễ dùng để định nghĩa và validate dữ liệu trên Form.
- ❑ Kết hợp **React Hook Form** và **Yup** là best practice hiện nay trong các dự án thường dùng để quản lý Form và validation trong ReactJS.



Thư viện React Hook Form, Yup

❑ Cài đặt RHF, Yup:

`npm install react-hook-form @hookform/resolvers yup`

Form handling with



React



React Hook Form

yup

yup

Thư viện React Hook Form, Yup

❑ Ví dụ tạo LoginForm component sử dụng RHF, Yup:

- Import các thư viện cần thiết để sử dụng RHF, Yup trong React Component:

```
import React from 'react';  
import { useForm } from 'react-hook-form';  
import { yupResolver } from '@hookform/resolvers/yup';  
import * as yup from 'yup';
```

Thư viện React Hook Form, Yup

❑ Ví dụ tạo LoginForm component sử dụng RHF, Yup:

○ Tạo Component **LoginForm**:

```
const LoginForm: React.FC = () => {  
  return (  
    <form onSubmit={handleSubmit(onSubmit)}  
      className="p-4 max-w-md mx-auto border rounded">  
      // 2 trường Email, Mật khẩu...  
    </form>  
  );  
}  
export default LoginForm;
```

Thư viện React Hook Form, Yup

❑ Ví dụ tạo LoginForm component sử dụng RHF, Yup:

- Nội dung 2 trường dữ liệu Email, Mật khẩu trong LoginForm:

```
<div>
  <label>Email</label>
  <input {...register('email')} className="border p-2 w-full" />
  <p className="text-red-500">{errors.email?.message}</p>
</div>
```

```
<div className="mt-4">
  <label>Mật khẩu</label>
  <input type="password" {...register('password')} className="border p-2 w-full" />
  <p className="text-red-500">{errors.password?.message}</p>
</div>
```

```
<button type="submit" className="mt-4 p-2 bg-blue-500 text-white rounded">Đăng nhập</button>
```

Thư viện React Hook Form, Yup

❑ Ví dụ tạo LoginForm component sử dụng RHF, Yup:

○ Định nghĩa **schema validation** bằng **Yup** nằm ngoài, phía trên **LoginForm**:

```
const schema = yup.object({  
  email: yup.string().email('Email không hợp lệ').required('Bắt buộc nhập email'),  
  password: yup.string().min(6, 'Mật khẩu tối thiểu 6 ký tự').required('Bắt buộc nhập mật khẩu'),  
}).required();
```

```
type FormData = yup.InferType<typeof schema>;
```

Thư viện React Hook Form, Yup

❑ Ví dụ tạo LoginForm component sử dụng RHF, Yup:

- Khai báo **yup resolver schema** để **validation** trên phần **return** trong **LoginForm**:

```
const { register, handleSubmit, formState: { errors } } = useForm<FormData>({  
  resolver: yupResolver(schema),  
});
```

```
const onSubmit = (data: FormData) => {  
  console.log('Dữ liệu hợp lệ:', data);  
};
```

Tóm tắt bài học

- ☐ Thao tác Form trong ReactJS
- ☐ Xử lý dữ liệu trên Form
- ☐ Thao tác Submit Form
- ☐ Thao tác Multiple Input Fields
- ☐ Thao tác Form Validation
- ☐ Thư viện React Hook Form, Yup

