



# Full-Stack Software Development

**Course:** Advanced Frontend Development Using React

**Lecture on:** Forms and Material-UI

**Instructor:** Mrigank Kaushik



## In the last class, we discussed

- React Hooks for handling state management and side effects
- How to implement lifecycle methods of a component using React Hooks
- How React Hooks help segregate logically separate functionalities and help reduce the complexity of large components
- `useState`, `useEffect` and `useContext` hooks

# Poll 1

Which of the following built-in React Hooks are used for lifecycle methods in components?

- A. `useState`
- B. `useCallback`
- C. `useEffect`
- D. `useReducer`

# Poll 1 (Answer)

Which of the following built-in React Hooks are used for lifecycle methods in components?

- A. `useState`
- B. `useCallback`
- C. `useEffect`**
- D. `useReducer`

## Poll 2

Which of the following built-in Hooks is used to access the React context object?

- A. useContext
- B. useContextProvider
- C. createContext
- D. None of the above

## Poll 2 (Answer)

Which of the following built-in Hooks is used to access the React context object?

- A. useContext**
- B. useContextProvider
- C. createContext
- D. None of the above

# Today's Agenda

1. Implementing forms in React
  - Difference from traditional HTML forms
  - Controlled vs uncontrolled components
2. Form validation using Material-UI
  - Higher order validator components
  - Validation rules



# Implementing Forms in React

## Difference Between Forms in HTML and React

### HTML

- Form data is handled by the DOM
- Data is stored directly in the DOM elements

Therefore, accessing this data involves reading it directly from the DOM. For example:

```
<form>  
  <input type="text" name="email" />  
</form>
```

```
const email = document.querySelector('input[name="email"]').value;
```

## Difference Between Forms in HTML and React

### **React**

- Form data is managed in the components
- Data is stored in the component state

We will discuss an example of this approach shortly.

## Controlled Components

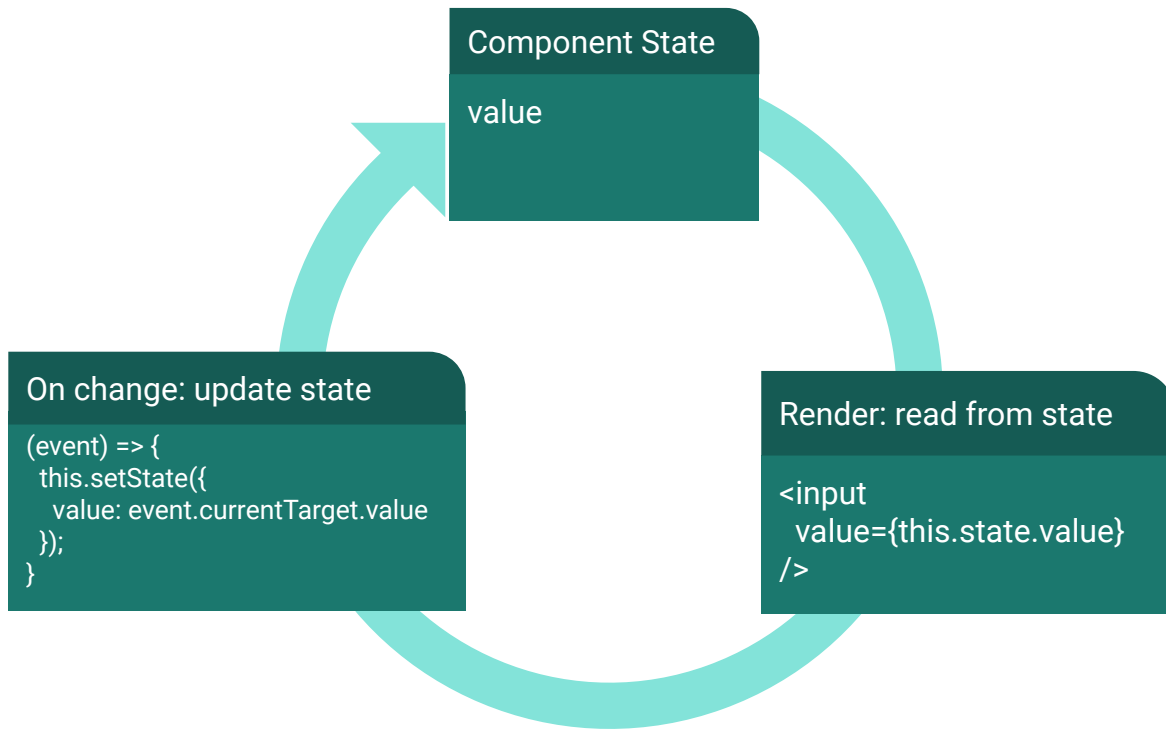
A 'controlled' React component renders a form with the following properties:

- React state acts as the single source of truth
- The component that renders the form also controls what happens on user input
- The form data is maintained in the internal state of the form component

## Controlled Form Component

```
class MyForm extends React.Component {
  onChange = (e) => {
    this.setState({ name: e.target.value });
  }
  onSubmit = (e) => {
    // Prevent the page from reloading
    e.preventDefault();
    // submit the data to the server...
  }
  render() {
    return (
      <form onSubmit={this.onSubmit}>
        <label>Enter name</label>
        <input type="text" name="name" value={this.state.name} onChange={this.onChange} />
        <input type="submit" value="Submit" />
      </form>
    );
  }
}
```

## How do Controlled Components Work?



## Textarea

In HTML, there is no value attribute for **textarea**, but to help us turn it into a controlled component, React supports a value prop.

Textarea syntax in HTML:

```
<textarea>  
  Input content of text area  
</textarea>
```

textarea syntax in React (value prop)

```
<textarea value="Input content of text area" />
```

## Select

React supports the value prop on select components to specify the currently selected option. For example:

```
<select value={this.state.value} onChange={e => this.setState({ value: e.target.value })}>
  <option value="foo">Foo</option>
  <option value="bar">Bar</option>
  <option value="baz">Baz</option>
</select>
```



## File Input

The value of a file input is read-only; hence, it is not possible for React to manage it.

Therefore, `<input type="file">` is an uncontrolled component in React.

## Form With Multiple Inputs (Generic)

```
class MyForm extends React.Component {
  onChange = (e) => {
    const value = e.target.type === 'checkbox' ? e.target.checked : e.target.value;
    this.setState({ [e.target.name]: value });
  }
  onSubmit = (e) => {
    e.preventDefault();
    // submit the data to the server...
  }
  render() {
    return (
      <form onSubmit={this.onSubmit}>
        <label>Enter name</label>
        <input type="text" name="name" value={this.state.name} onChange={this.onChange} />
        <label>Enter phone number</label>
        <input type="number" name="phone" value={this.state.phone} onChange={this.onChange} />
        <input type="submit" value="Submit" />
      </form>
    );
  }
}
```

## Poll 3

Which of the following is true for React form components?

- A. The data entered in the form is stored in the DOM.
- B. The data entered in the form is stored in the component's state.
- C. The data entered in the form is managed by refs.
- D. The data in a React form cannot change.

## Poll 3 (Answer)

Which of the following is true for React form components?

- A. The data entered in the form is stored in the DOM.
- B. The data entered in the form is stored in the component's state.**
- C. The data entered in the form is managed by refs.
- D. The data in a React form cannot change.

## Poll 4

What are form elements in React whose values derive from the state and user input leads to the updation of state with the user input value called?

- A. Uncontrolled components
- B. Stateful components
- C. Controlled components
- D. Pure components

## Poll 4 (Answer)

What are form elements in React whose values derive from the state and user input leads to the updation of state with the user input value called?

- A. Uncontrolled components
- B. Stateful components
- C. Controlled components**
- D. Pure components

# Form Validation Using Material-UI

## The Need for Validation

There are two primary reasons for which the data needs to be validated before it is sent to the server for persisting:

- Security: Being able to input random strings opens up our database and web application to a variety of attack vectors, such as SQL injection and cross-site scripting (XSS)
- Ensuring sanity of the input data: Validation helps ensure that some simple rules are being followed, such as a mandatory form field or the input format of a particular field. This, in turn, helps the user enter valid values and correct data to some extent



## Validation in React Forms

You can add data validation logic in either:

- onChange or
- onSubmit

To this end, you already have libraries available to help you without having to reinvent the wheel!

A React component library/design system: <https://material-ui.com/>

Material Design is a design language developed by Google in 2014. According to Google, it is a visual language that synthesises the classic principles of a good design, with the innovation of technology and science.

Provides a set of commonly used components with built-in theming support and other common functionalities.

## Component Example

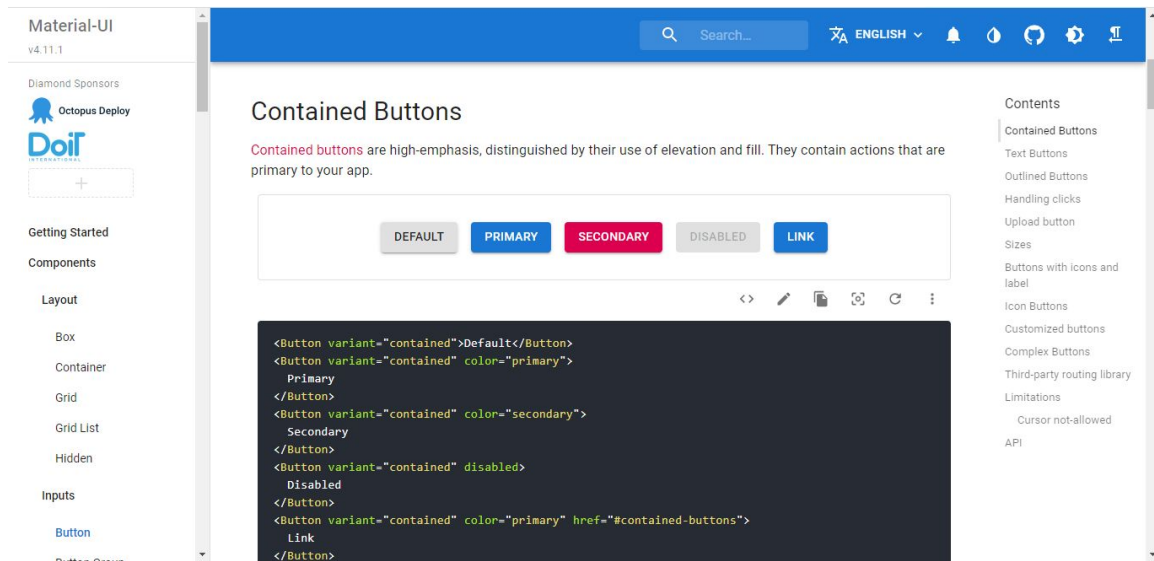
For example, you can implement a delete button with a corresponding icon in your component with a few lines of code depicted below-

```
import Button from '@material-ui/core/Button';  
import DeleteIcon from '@material-ui/icons/Delete';
```

```
<Button  
  variant="contained"  
  color="secondary"  
  className={classes.button}  
  startIcon={<DeleteIcon />}  
>  
  Delete  
</Button>
```

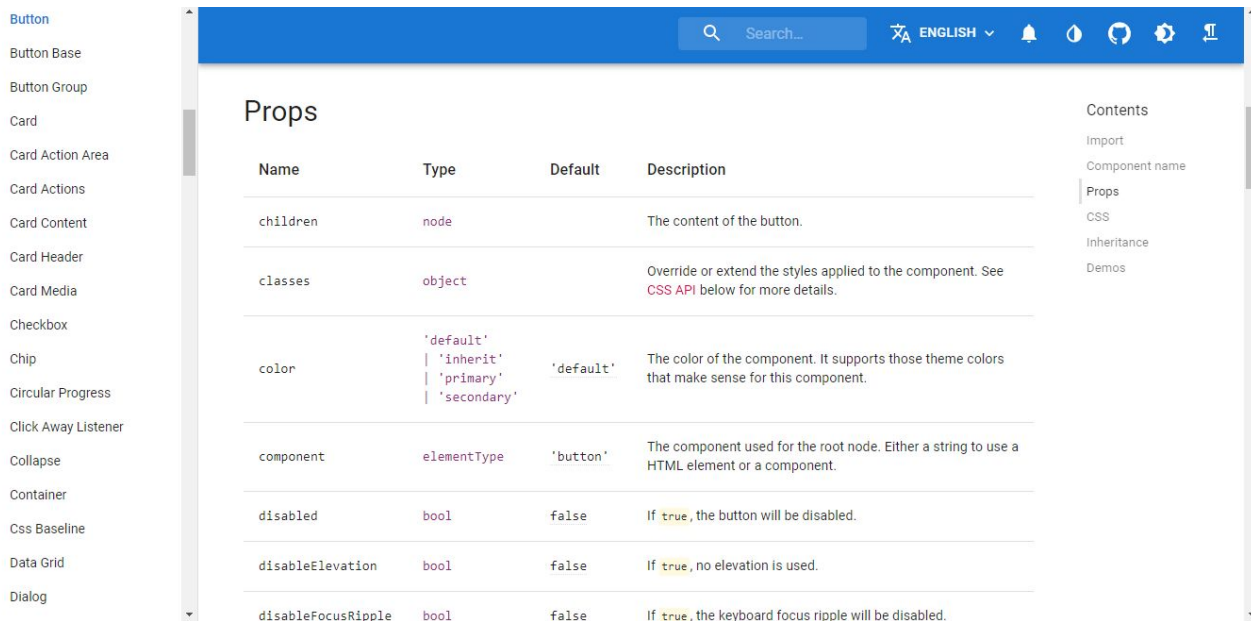
## Components

On the Material-UI website, there is full-fledged documentation of all the different variants of the components and how to configure them as per our use.



## Component API

Detailed API (props) of the individual components is also well-documented.



The screenshot shows the Material-UI documentation page for the `Button` component. The left sidebar lists various components, and the main content area displays the 'Props' table. The table has four columns: Name, Type, Default, and Description. The props listed are `children`, `classes`, `color`, `component`, `disabled`, `disableElevation`, and `disableFocusRipple`.

Name	Type	Default	Description
<code>children</code>	<code>node</code>		The content of the button.
<code>classes</code>	<code>object</code>		Override or extend the styles applied to the component. See <a href="#">CSS API</a> below for more details.
<code>color</code>	<code>'default'</code> <code>'inherit'</code> <code>'primary'</code> <code>'secondary'</code>	<code>'default'</code>	The color of the component. It supports those theme colors that make sense for this component.
<code>component</code>	<code>elementType</code>	<code>'button'</code>	The component used for the root node. Either a string to use a HTML element or a component.
<code>disabled</code>	<code>bool</code>	<code>false</code>	If <code>true</code> , the button will be disabled.
<code>disableElevation</code>	<code>bool</code>	<code>false</code>	If <code>true</code> , no elevation is used.
<code>disableFocusRipple</code>	<code>bool</code>	<code>false</code>	If <code>true</code> , the keyboard focus ripple will be disabled.

## Installation

Installation using NPM:

```
npm install @material-ui/core --save-dev
```

This code will download the material-ui core library as a dependency in the local project as well as add it to the project's package.json

## Sample Code

```
import { Button, Input } from '@material-ui/core';

function LoginForm() {
  const onChange = () => { ... } // event handler for user input in form
  const handleSubmit = () => { ... } // event handler for form submit

  return (
    <form noValidate>
      <Input type="text" label="Email" onChange={onChange} />
      <Input type="password" label="Password" onChange={onChange} />
      <Button onClick={handleSubmit} color="primary">Submit</Button>
    </form>
  );
}
```

## Library to Add Validation Functionality to Material-UI

Installation using NPM:

```
npm install react-material-ui-form-validator --save-dev
```

This code will download the react-material-ui-form-validator library as a dependency in the local project as well as add it to the project's package.json

Refer to the following link for more information:

<https://www.npmjs.com/package/react-material-ui-form-validator>



## Higher Order Validator Components

- TextValidator (for all text-based input fields)
- SelectValidator (for select input field)

Props:

- validators (list of predefined validator rules)
- errorMessages (list of error messages corresponding to validators)

## Validator Rules

*react-form-validator-core* supports a set of predefined rules, such as:

- matchRegexp
- isEmail
- required
- isNumber
- isPositive
- minNumber
- maxNumber
- isString
- minStringLength
- maxStringLength
- maxFileSize

## Example

Let's look at an example of a form that does the following:

- It expects two inputs from the user: email and password
- Both email and password are required to be entered to be able to submit the form
- If any of these fields are empty, that field is going to be highlighted in red with an error message
- The email address should also be checked whether it is valid. For example, helloworld is not a valid email address, however hello@world can be considered as valid a valid address

## Example Code

```
import { ValidatorForm, TextValidator } from 'react-material-ui-form-validator';
import { Button } from '@material-ui/core';

function LoginForm() {
  const handleSubmit = () => { ... }
  const onChange = () => { ... }
  return (
    <ValidatorForm onSubmit={handleSubmit} onError={errors => console.log(errors)}>
      <TextValidator
        label="Email" onChange={onChange} name="email"
        validators={['required', 'isEmail']}
        errorMessages={['Email is required', 'Invalid email']} />
      <TextValidator
        label="Password" onChange={onChange} name="password"
        validators={['required']}
        errorMessages={['Please enter your password']} />
      <Button onClick={handleSubmit} color="primary">Submit</Button>
    </ValidatorForm>
  );
}
```

## Phone Directory

In the Phone Directory app, the input to add a new contact is through a form in the *AddSubscriber* component. You can add some validations to ensure the basic sanity of the user input. For example, the name field should be a required field:

```
<TextValidator
  id="name"
  label="Enter Name"
  type="text"
  name="name"
  onChange={inputChangedHandler}
  value={name}
  validators={['required']}
  errorMessages={['Name cannot be empty']}
>
</TextValidator>
```

[Code Reference](#)

- Write a simple React app that takes information from a user through a form.
- The details to be captured are name, phone number, country, address and email id.
- All are text fields, except country, which is a select dropdown.
- Of these fields, address is not mandatory and rest are required fields.
- Ensure that the correct validations are in place when the user clicks on the Submit button.



Name

Phone Number

Country

Address

Email

SUBMIT

The stub code is provided [here](#).

The solution code is provided [here](#).

## Poll 5

How do you use React forms to monitor the modifications?

- A. By adding the event handlers
- B. They are monitored automatically.
- C. Both of the above
- D. None of the above

## Poll 5 (Answer)

How do you use React forms to monitor the modifications?

- A. By adding the event handlers**
- B. They are monitored automatically.
- C. Both of the above
- D. None of the above



## Poll 6

Where does it make sense to add validation logic for a form in React?

- A. The onLoad handler for the form
- B. The onClick handler for the form
- C. The onSubmit handler for the form
- D. The update state call for the form

## Poll 6 (Answer)

Where does it make sense to add validation logic for a form in React?

- A. The onLoad handler for the form
- B. The onClick handler for the form
- C. The onSubmit handler for the form**
- D. The update state call for the form

## Poll 7

Which of the following validator components are provided by the *react-material-ui-form-validator* library?

**(Note:** More than one option may be correct.)

- A. `DataValidator`
- B. `TypeValidator`
- C. `TextValidator`
- D. `SelectValidator`

## Poll 7 (Answer)

Which of the following validator components are provided by the *react-material-ui-form-validator* library?

**(Note:** More than one option may be correct.)

- A. `DataValidator`
- B. `TypeValidator`
- C. **`TextValidator`**
- D. **`SelectValidator`**

## Poll 8

Which of the following is not a commonly used valid prop for validator components offered by the react-material-ui-form-validator library?

- A. label
- B. onValidate
- C. validators
- D. errorMessages

## Poll 8 (Answer)

Which of the following is not a commonly used valid prop for validator components offered by the react-material-ui-form-validator library?

- A. label
- B. onValidate**
- C. validators
- D. errorMessages

All the code used in today's session can be found in the link provided below (branch Session9):

<https://github.com/upgrad-edu/react-hooks/tree/Session9>

# Doubt Clearance (5 minutes)



These tasks are to be completed after today's session:

MCQs
Coding Questions
Course Project (Part B) - Checkpoint 3

# Key Takeaways

- Forms in HTML handle data in the DOM, whereas the React forms manage the data within the internal state of the form component itself
- Such components are called 'controlled components'
- Material-UI provides a component library for a common set of out-of-the-box but customisable components can be used to build forms and other visual components
- Material-UI Form Validator library can be used to plug in data validation into the forms created using Material-UI

# In the next class, we will discuss

- Backend integration using APIs
  - Introduction to REST APIs
  - Session and Authentication
  - Using Fetch



Thank You!