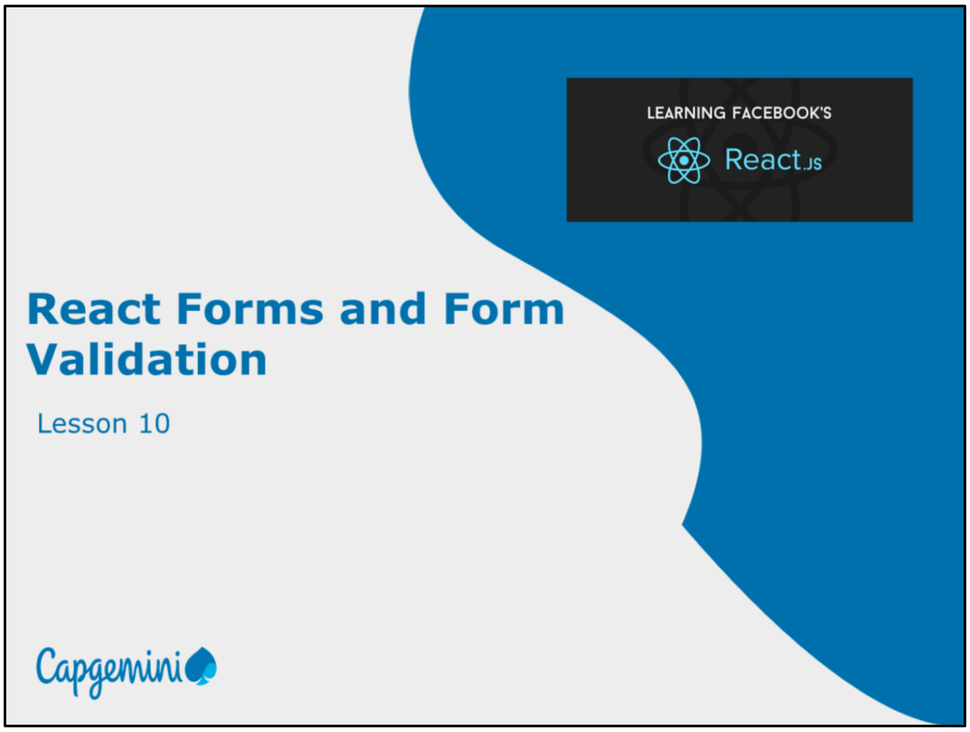
Instructor Notes:

Add instructor notes here.



LEARNING FACEBOOK'S React.JS

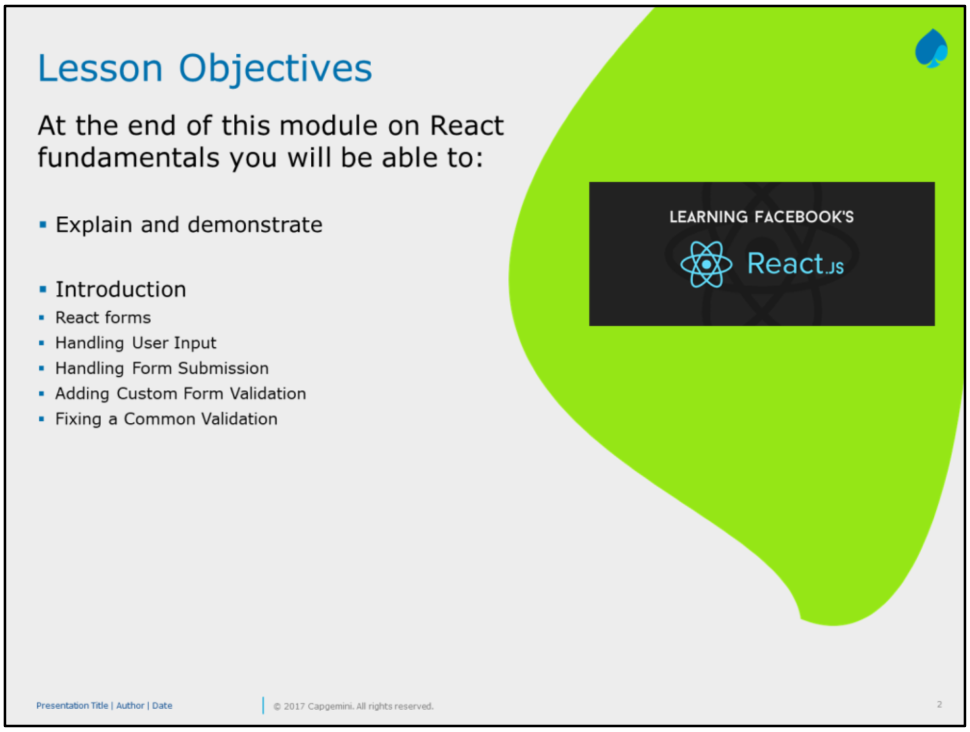
React Forms and Form Validation

Lesson 10

Capgemini

Instructor Notes:

Add instructor notes here.



Lesson Objectives

At the end of this module on React fundamentals you will be able to:

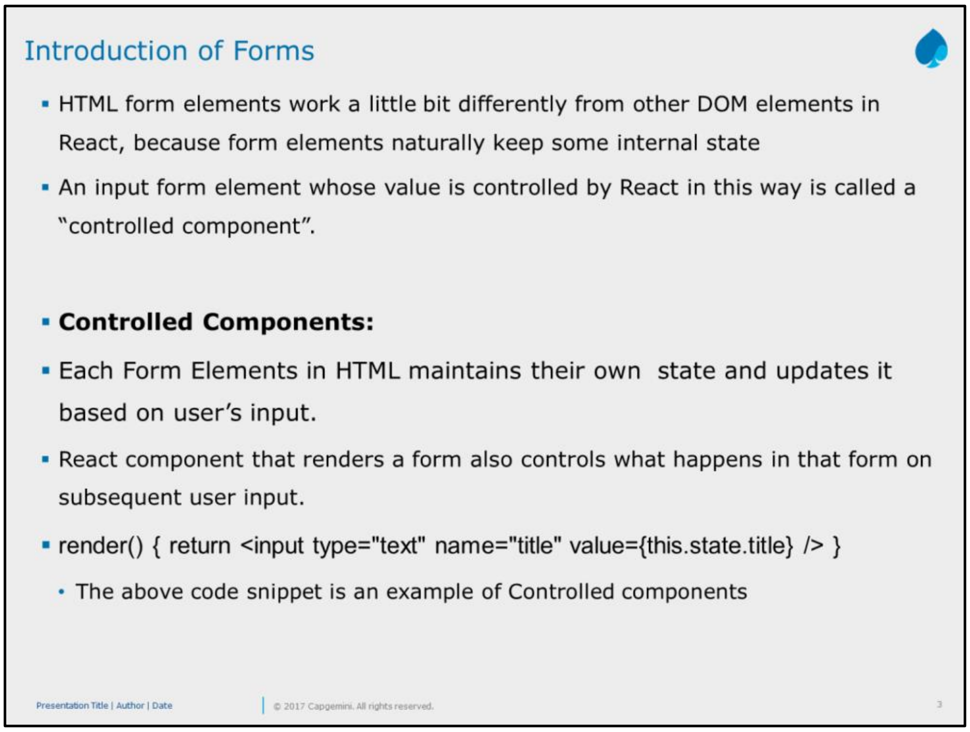
- Explain and demonstrate
- Introduction
- React forms
- Handling User Input
- Handling Form Submission
- Adding Custom Form Validation
- Fixing a Common Validation

LEARNING FACEBOOK'S
React.JS

Presentation Title | Author | Date © 2017 Capgemini. All rights reserved. 2

Instructor Notes:

Slide explains that communication between service provider and consumer happen via SOAP messages



Introduction of Forms

- HTML form elements work a little bit differently from other DOM elements in React, because form elements naturally keep some internal state
- An input form element whose value is controlled by React in this way is called a "controlled component".

- **Controlled Components:**
- Each Form Elements in HTML maintains their own state and updates it based on user's input.
- React component that renders a form also controls what happens in that form on subsequent user input.
- render() { return <input type="text" name="title" value={this.state.title} /> }
  - The above code snippet is an example of Controlled components

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 3

In the traditional HTML form elements, the state of the elements will change with the user input.

React uses a declarative approach to describe the UI. The input needs to be dynamic to reflect the state properly.

render() { return <input type="text" name="compName" value="capgemini" /> }

The code above represents the view at any state, and the value will always be "capgemini".

With input fields, they must change in response to the user keystrokes. Given these points, let's make the value dynamic.

render() { return <input type="text" name="title" value={this.state.title} /> }

Developers need to implement an event handler to capture changes with onChange.

The best practice is for developers to implement the following things to sync the internal state with the view

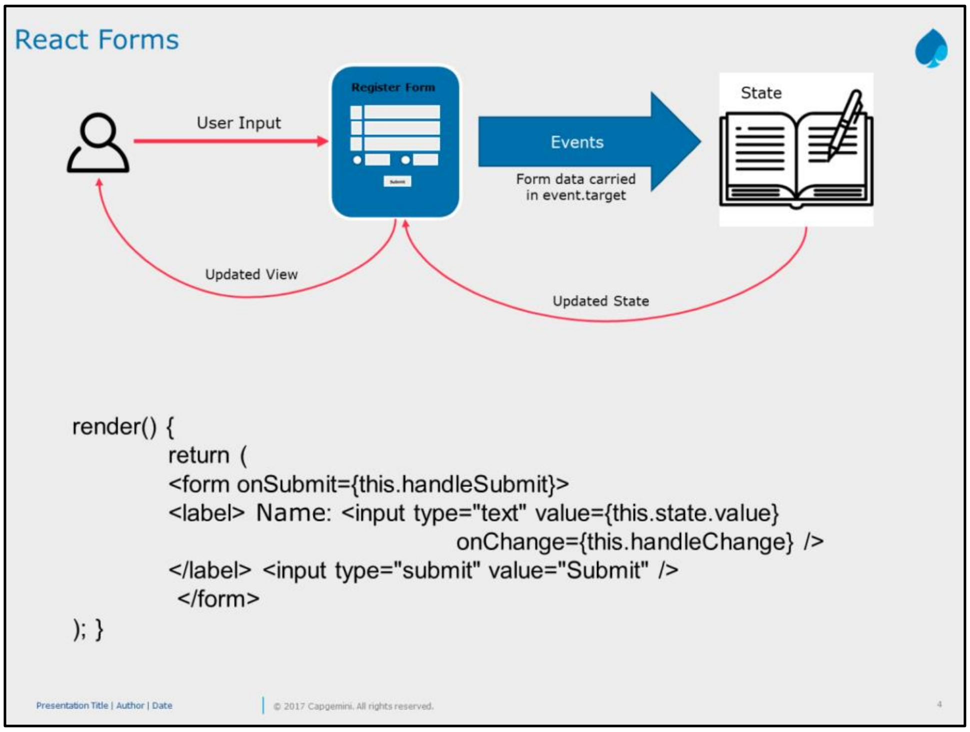
Controlled Components

In normal HTML form, it has the default behavior of browsing to a new page when the user submits the form. Same behavior is also available in React. But in most cases, what if a JavaScript function that handles the submission of the form and has access to the data that the user entered into the form. The standard way to achieve this is with a technique called "**controlled components**".

In HTML Form elements like <input>, <textarea>, and <select> have their own state and updates based on user input.
But in React, mutable state is typically kept in the state property of components, and only updated with setState().

Instructor Notes:

The slide explains in brief, the components which make up the web service



React Forms

```
render() {
            return (
            <form onSubmit={this.handleSubmit}>
            <label> Name: <input type="text" value={this.state.value}
                                      onChange={this.handleChange} />
            </label> <input type="submit" value="Submit" />
             </form>
); }
```

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 4

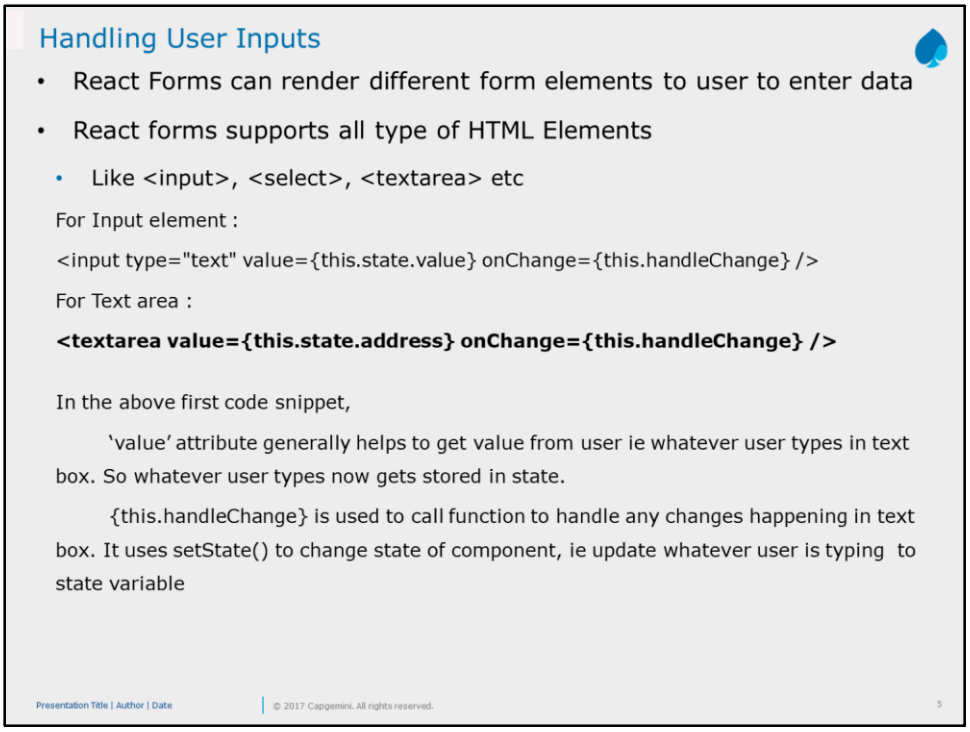
Flow of internal state with the view :

Define elements in render() using values from state
capture changes of a form element using onChange() as they happen
update the internal state in event handler
save new values in state and then update the view with a new render()

Instructor Notes:

Slide explains about WSDL and UDDI registry.

Also explains regarding web service communication



Handling User Inputs

- React Forms can render different form elements to user to enter data
- React forms supports all type of HTML Elements
  - Like <input>, <select>, <textarea> etc

For Input element :

```
<input type="text" value={this.state.value} onChange={this.handleChange} />
```

For Text area :

```
<textarea value={this.state.address} onChange={this.handleChange} />
```

In the above first code snippet,

'value' attribute generally helps to get value from user ie whatever user types in text box. So whatever user types now gets stored in state.

{this.handleChange} is used to call function to handle any changes happening in text box. It uses setState() to change state of component, ie update whatever user is typing to state variable

Presentation Title | Author | Date　© 2017 Capgemini. All rights reserved.　5

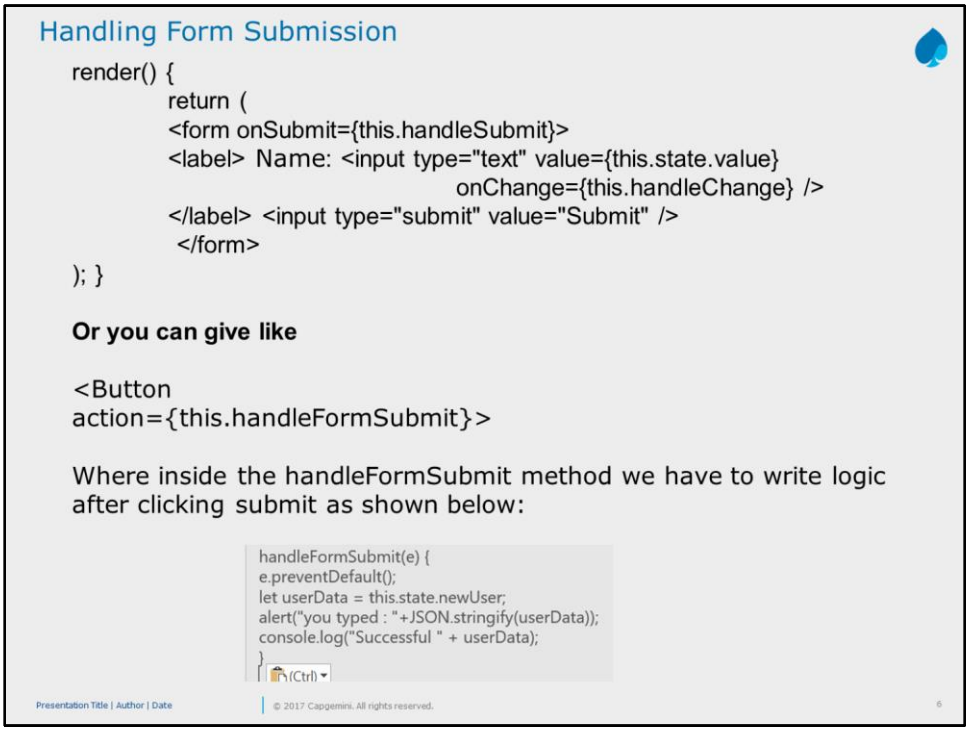
Any submit or button click is triggered in the form will trigger handleClick

```
handleClick=()=>{
                    this.setState({
                    displayValue:this.state.value.toUpperCase(),
                    value:''
      });
      }
```

For creating react <select> ie., Drop down:

```
<select value={this.state.value} onChange={this.handleChange}>
      <option value="grapefruit">Grapefruit</option>
      <option value="lime">Lime</option>
      <option value="coconut">Coconut</option>
      <option value="mango">Mango</option>
    </select>
```

Instructor Notes:



Handling Form Submission

```
render() {
        return (
        <form onSubmit={this.handleSubmit}>
        <label> Name: <input type="text" value={this.state.value}
                        onChange={this.handleChange} />
        </label> <input type="submit" value="Submit" />
         </form>
); }
```

Or you can give like

```
<Button
action={this.handleFormSubmit}>
```

Where inside the handleFormSubmit method we have to write logic after clicking submit as shown below:

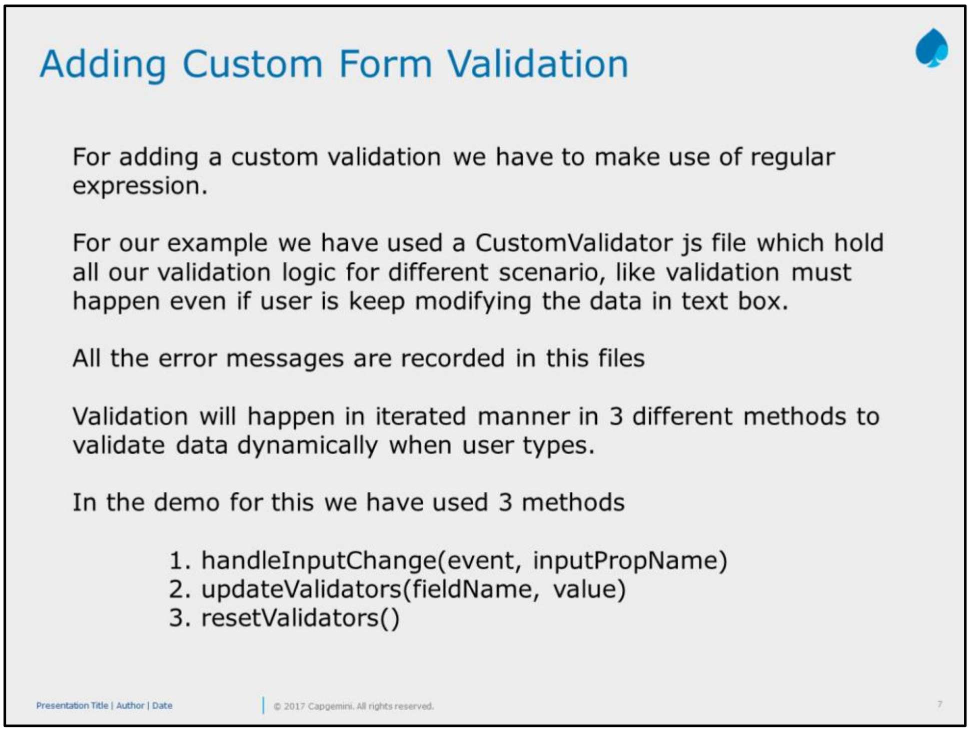
```
handleFormSubmit(e) {
e.preventDefault();
let userData = this.state.newUser;
alert("you typed : "+JSON.stringify(userData));
console.log("Successful " + userData);
}
```

Presentation Title | Author | Date © 2017 Capgemini. All rights reserved. 6

You've got three options
Add a constructor and do the binding there (recommended):
this.handleSubmit = this.handleSubmit.bind(this);
Bind directly:
onSubmit={this.handleSubmit.bind(this)}
Use arrow => functions
onSubmit={() => this.handleSubmit}

```
handleFormSubmit(e) {
e.preventDefault();
let userData = this.state.newUser;
alert("you typed : "+JSON.stringify(userData));
console.log("Successful " + userData);
}
```

Instructor Notes:



Adding Custom Form Validation

For adding a custom validation we have to make use of regular expression.

For our example we have used a CustomValidator js file which hold all our validation logic for different scenario, like validation must happen even if user is keep modifying the data in text box.

All the error messages are recorded in this files

Validation will happen in iterated manner in 3 different methods to validate data dynamically when user types.

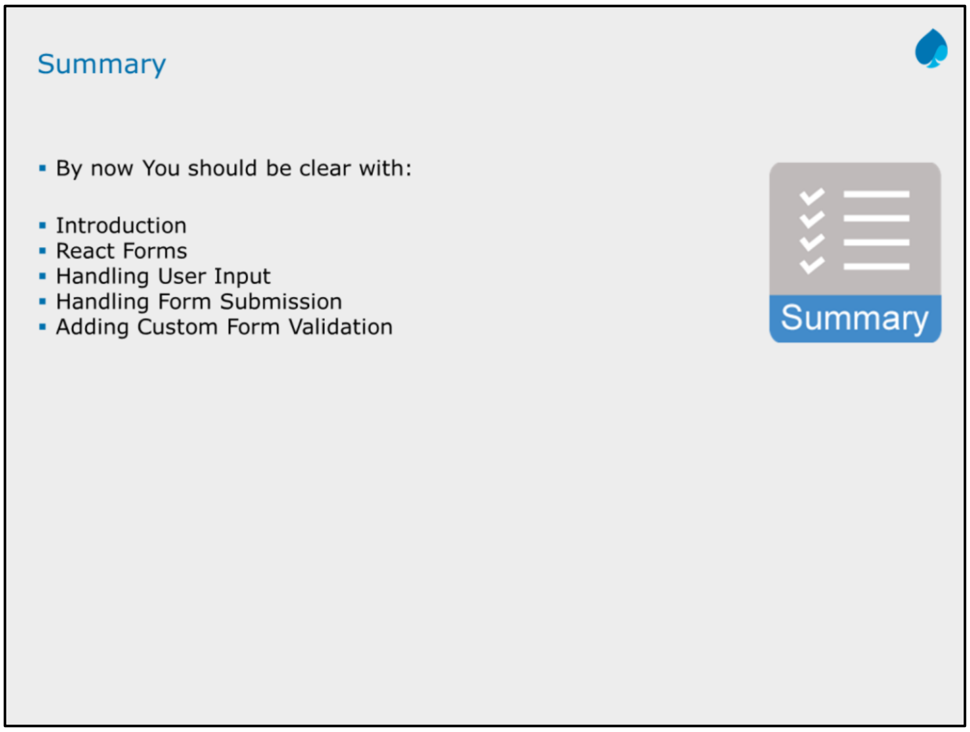
In the demo for this we have used 3 methods

1. handleInputChange(event, inputPropName)
2. updateValidators(fieldName, value)
3. resetValidators()

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 7

Instructor Notes:

Add instructor notes here.

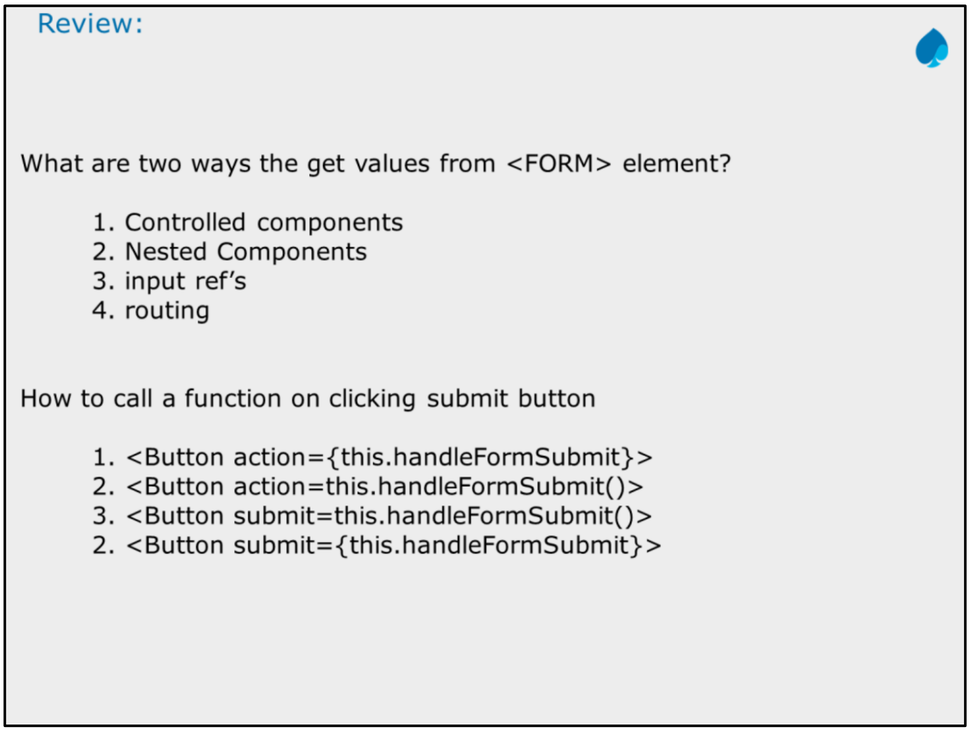


Summary

- By now You should be clear with:
- Introduction
- React Forms
- Handling User Input
- Handling Form Submission
- Adding Custom Form Validation

Add the notes here.

Instructor Notes:



Review:

What are two ways the get values from <FORM> element?

1. Controlled components
2. Nested Components
3. input ref’s
4. routing

How to call a function on clicking submit button

1. <Button action={this.handleFormSubmit}>
2. <Button action=this.handleFormSubmit()>
3. <Button submit=this.handleFormSubmit()>
2. <Button submit={this.handleFormSubmit}>