

# React JS Training

(Intermediate to Advanced)



*Narasimha*

Sr. Corporate Trainer, Mentor  
tnrao.trainer@gmail.com

# Schedule for React JS Training

---

Day#	Date	Topic
Day-1	16-Apr-2024	State Management : Class Components
Day-2	17-Apr-2024	State Management : Functional Components, Hooks
Day-3	18-Apr-2024	Http Client Programming – AJAX Calls to APIs (Node JS)
Day-4	19-Apr-2024	Working with Forms, Validations, Services
Day-5	22-Apr-2024	Redux – State Management Library
Day-6	23-Apr-2024	Unit Testing React Application
Day-7	24-Apr-2024	Routing – SPA in React JS

# Day4

## Working with Forms



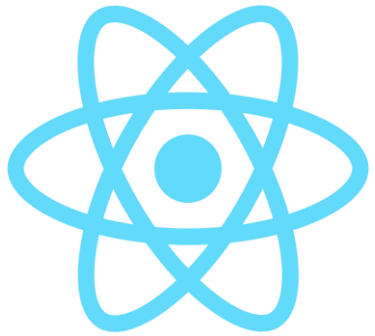
*Narasimha*

Sr. Corporate Trainer, Mentor  
tnrao.trainer@gmail.com

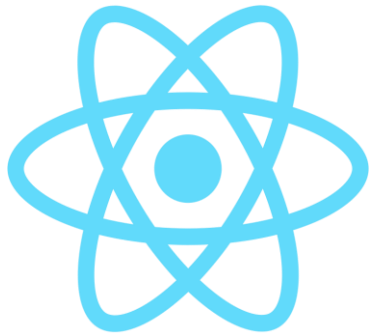
# Index – Day4

---

1. Recap : Reusable Services objects in React
2. Controlled and Uncontrolled components
3. React Forms
4. Custom Validations using React Forms



# useEffect Hook



# Controlled vs Uncontrolled Components

# Controlled vs Uncontrolled Components

---

- A Controlled Component is one that takes its current value through props and notifies changes through callbacks like `onChange`.
  - A parent component "controls" it by handling the callback and managing its own state and passing the new values as props to the controlled component.
- 
- A Uncontrolled Component is one that stores its own state internally, and you query the DOM using a `ref` to find its current value when you need it. This is a bit more like traditional HTML.

# Controlled vs Uncontrolled Components

---

- In most cases, we recommend using controlled components to implement forms.
- In a controlled component, form data is handled by a React component.
- The alternative is uncontrolled components, where form data is handled by the DOM itself.
- To write an uncontrolled component, instead of writing an event handler for every state update, you can use a ref to get form values from the DOM.



# Controlled vs Uncontrolled Components

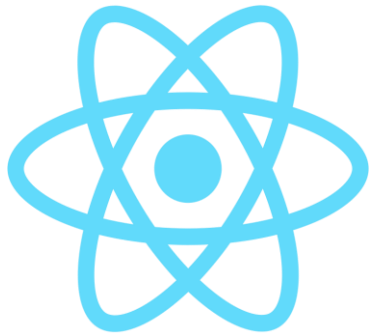
---

```
import {useRef} from 'react';
```

```
const inputRef = useRef(null);
```

```
<input ref={inputRef} type="text" value={uname} />
```

```
inputRef.current.focus();
```



# React Forms

# Working with Forms in React

---

- Just like in HTML, React uses forms to allow users to interact with the web page.
- You can add a form with React like any other html element
- This will work as normal, the form will submit and the page will refresh. But this is generally not what we want to happen in React.
- We want to prevent this default behavior and let React control the form.

## onSubmit event

---

```
const handleSubmit = (event) =>
{
  event.preventDefault();
  alert(`Form is submitted by user`);
}
```

```
<form onSubmit={handleSubmit}>
  .....
  <input type="submit" />
</form>
```

# Code Snippets

---

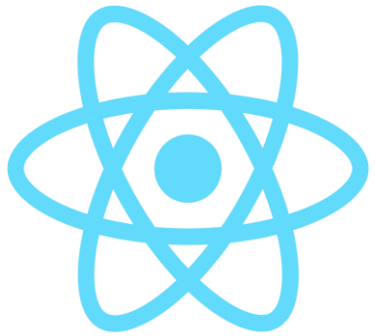
```
let tempErrorObj = Object.assign({}, errorsObj);  
tempErrorObj.username = (username.length == 0) ? "User Name is required" : "";  
  
let valuesArray = Object.values(tempErrorObj);  
  
let index = valuesArray.findIndex( item => item.length != 0 );
```

## Practice Hands-Ons

*Narasimha*

Sr. Corporate Trainer, Mentor

tnrao.trainer@gmail.com



# React Forms Validations using 3<sup>rd</sup> Party Library



*Narasimha*

Sr. Corporate Trainer, Mentor

[tnrao.trainer@gmail.com](mailto:tnrao.trainer@gmail.com)



