WEB TECHNOLOGIES

React JS – Component States

Prof. Vinay Joshi and Dr. Sarasvathi V
Department of Computer Science and Engineering

Acknowledgement
The slides are created from various internet resources with valuable contributions from multiple professors and teaching assistants in the university.

Component States

Agenda

- Introduction
- Conventions of using states
- Demo of differences between props and states

React.JS – Stateful Components
Introduction

- Components need to change based on
  - User actions (clicks, keyboard inputs, etc.)
  - Other triggers (responses received from server, timers, etc.)
- Static or stateless Components don’t undergo state changes.

React.JS – Stateful Components
Introduction

- In ReactJS, stateful components are components that maintain and manage internal state.
- Unlike stateless components, which simply render data based on props, stateful components actively respond to changes in their internal state, leading to dynamic updates of the user interface.

The state within a component can evolve based on:

1. **User actions:** Inputs such as button clicks, form entries, or keyboard interactions can trigger state changes. These updates allow the UI to react in real time, such as enabling buttons, showing validation errors, or updating content.
2. **External triggers:** Events such as receiving data from an API, completing a timer, or listening to changes in other components or systems can also cause a component's state to update. This makes the UI reactive to asynchronous events, such as displaying loading indicators or updated data upon successful API calls.

React.JS – Stateful Components

Introduction

- In contrast, **static or stateless components** are purely presentational and do not manage or track any state internally. They simply render based on the props passed down to them and do not undergo any state changes during their lifecycle.

- Stateful components offer more flexibility in managing complex logic and UI interactions, making them essential for creating dynamic, interactive applications.

- React provides hooks such as useState and useEffect to manage and handle state changes efficiently.

Component States

States - Introduction

Props	State
Immutable. once set the props cannot be changed	observable object that is to be used to hold data that may change over time and to control the behavior after each change.
Props are set by the parent component.	State is generally updated by event handlers.
Props don’t have this limitation.	Used in Class Components

Both *props* and *state* are plain JavaScript objects.

- While both of them hold information that influences the output of render, they are different in their functionality with respect to component.
- Props get passed to the component similar to function parameters whereas state is managed within the component similar to variables declared within a function.

Component States

States - Introduction

- State is used with React Component Classes to make them dynamic.
- It enables the component to keep track of changing information in between renders.
- Used to track component's internal state which may change over time
- An instance of React Component Class can be defined as an object of a set of observable properties that control the behavior of the component
- An object that holds some information that may change over the lifetime of the component and to control the behavior after each change
- Can only be used in class components.
- Can only be accessed and modified inside the component and directly by the component

Component States

Conventions of using states

- Must first have some **initial state**, should define the state in the constructor of the component’s class.

```
    class MyClass extends React.Component {
            constructor(props)  {
                 super(props);
           this.state = { attribute : "value" };    }
  }
```

Component States

Conventions of using states

Constructor:

- The constructor is the first method called when a class component is instantiated. It's where we initialize the component's state and bind any event handler methods if necessary.

State:

- The state object is initialized in the constructor. The state holds data that can change over time and influence how the component renders.
- State is initialized in the constructor using **this.state = {}**.

Component States

Conventions of using states

Super

Will initiate parent's constructor method and allows component to inherit methods from its parent. Call constructor of its parent class, needed to access some variables of its parent class.

- The **super(props)** call invokes the constructor of the parent class (**React.Component**). This is necessary because React components that are class-based need to call the parent's constructor to properly initialize and access **this.props**.

- Without calling **super(props)**, **this.props** will be undefined within the constructor.

Component States

Conventions of using states

- **The state object can contain as many properties as you like**
- State should **never be updated explicitly**. Use **setstate()**
  - Takes a single parameter and expects an object which should contain the set of values to be updated.
  - Once the update is done the method implicitly calls the render() method to repaint the page.

setState() *can take either an object or a function.* ***setState()*** *merges the new state with the existing one and schedules a re-render of the component to reflect the changes in the UI.*

Component States

Demo of differences : props and state

- Generate the Digital clock using reactJS

THANK YOU

Vinay Joshi and Dr.Sarasvathi V

Department of Computer Science and Engineering

vinayj@pes.edu

sarsvathiv@pes.edu

Acknowledgement

The slides are created from various internet resources with valuable contributions from multiple professors and teaching assistants in the university.