WEB TECHNOLOGIES

React JS – refs

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.

refs and keys

Agenda

- Introduction to refs
- Create and use refs
- Callback refs

refs and keys

Introduction to refs

- Provides a **way to directly access and interact with DOM nodes or React elements created in the render method**

- Used to return a reference to DOM node or React element, allowing us to perform operations directly on them

- Refs are generally used when we need to perform actions that are not easily achieved through declarative React patterns.

refs and keys

Introduction to refs

- Good use cases for refs to be used:
  - **Managing Focus, Text Selection, or Media Playback**: To programmatically manage focus, text selection, or control media elements.
  - **Triggering Imperative Animations**: Refs can be used to trigger animations that require direct manipulation of the DOM.
  - **Integrating with Third-Party DOM Libraries**: When integrating with libraries that interact with the DOM, refs provide a way to access and manipulate the DOM elements.

refs and keys

How to create and use refs?

- **Creating Refs:**
    - **React.createRef()** is used to create refs in React, and it's typically done in the constructor (or with a class field).
    - These refs can then be attached to a React element via the ref attribute.

- **Accessing Refs:**
    - After the ref is attached to an element in the render() method, the DOM node can be accessed via **this.myRef.current**.
    - This is useful when we need to directly interact with the DOM node, like focusing on an input element or retrieving its value.

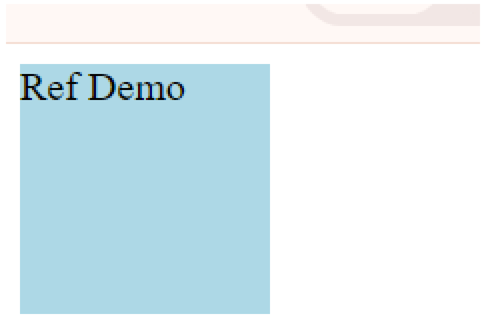
refs and keys

How to create and use refs? (coding example)

```
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    // Creating a ref using React.createRef()
    this.myRef = React.createRef();
  }

  componentDidMount() {
    // Accessing the DOM node when the component mounts
    const node = this.myRef.current;
    console.log('DOM Node:', node);
  }

  render() {
    return (
      // Attaching the ref to the div element
      <div ref={this.myRef} style={{ width: '100px', height: '100px', backgroundColor: 'lightblue' }}>
        Ref Demo
      </div>
    );
  }
}
```


Ref Demo

```
// Rendering the component
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<MyComponent />);
```

refs and keys

Callback refs

Fine-Grain Control:

- Callback refs give more control over when refs are set and unset. We can define a function that will be called when the element is rendered or removed.

Function as Ref:

- Instead of using React.createRef(), a function is used as the ref to assign the DOM element or component instance to a class property.
- This function receives the React component instance or HTML DOM element as an argument, allowing us to store and access the ref elsewhere.

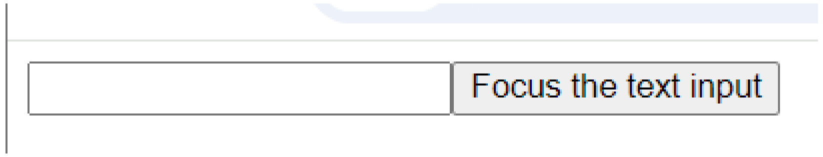
refs and keys

Callback refs (coding example)

```
class CustomTextInput extends React.Component {
 constructor(props) {
  super(props);
  this.textInput = null;

  // Callback ref function to assign the DOM element to this.textInput
  this.setTextInputRef = element => {
   this.textInput = element;
  };

  this.focusTextInput = () => {
   // Use the raw DOM API to focus the text input if it exists
   if (this.textInput) this.textInput.focus();
  };
 }
```

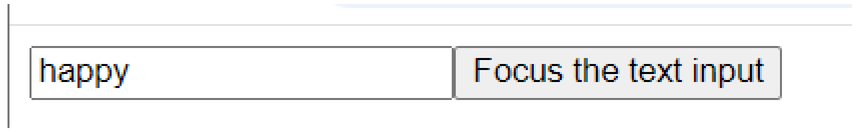


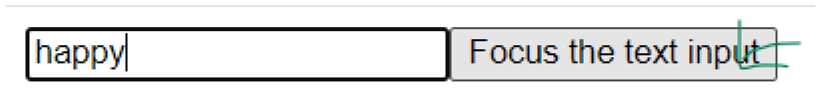
refs and keys

Callback refs (coding example) continuation...

```
componentDidMount() {
  // Automatically focus the input when the component mounts
  this.focusTextInput();
}

render() {
  return (
    <div>
      {/* Use the callback ref to get the input DOM element */}
      <input type="text" ref={this.setTextInputRef} />
      <input type="button" value="Focus the text input" onClick={this.focusTextInput} />
    </div>
  );
}}
```





THANK YOU

Vinay Joshi and 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.