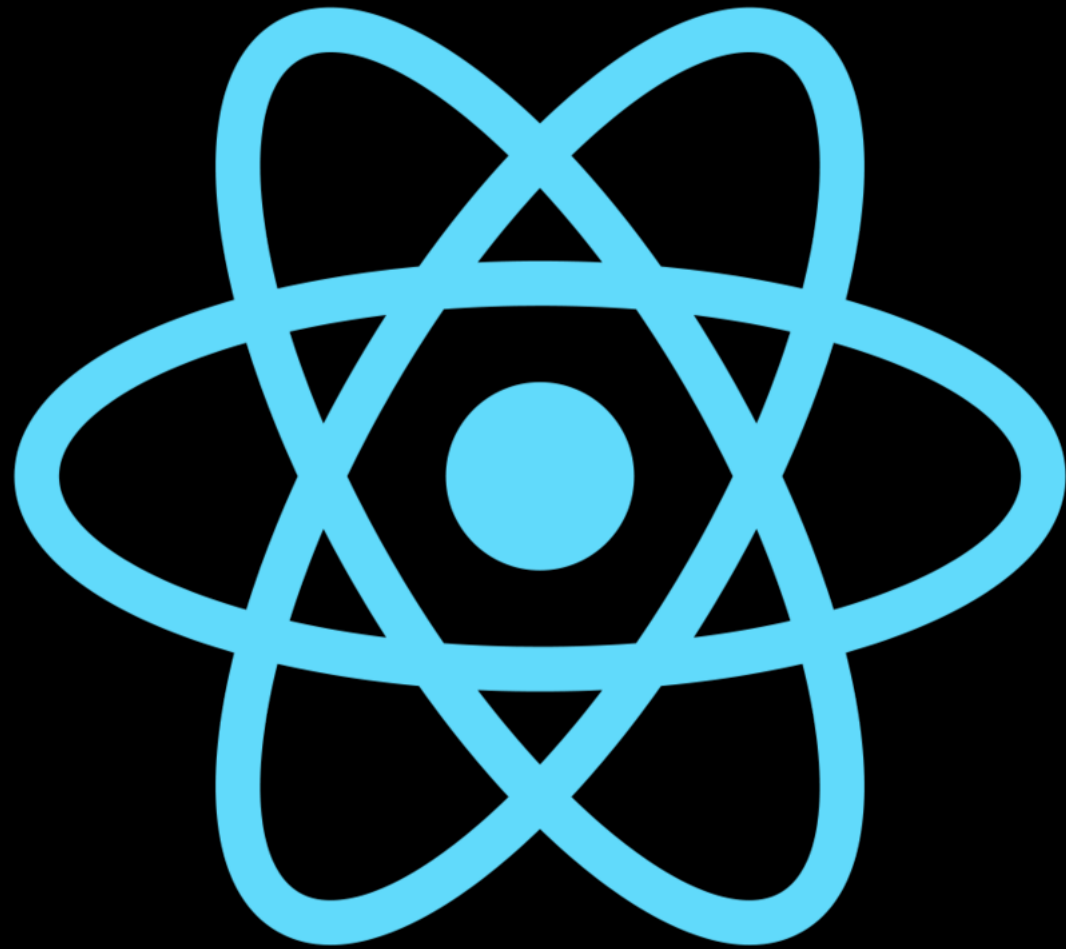


# LEARN TO CODE - REACT!



# OBJECTIVES

- ▶ **Describe Single Page Applications (SPA)**
  - ▶ **Name the key concepts in React**
  - ▶ **Explain components, props, and JSX**
- ▶ **Identify functional and class components**
  - ▶ **Know when to use state vs setState**
- ▶ **Lift state up to the closest common ancestor**

# **DESCRIBE SINGLE PAGE APPLICATIONS (SPA)**

**In a single-page application, all of the code needed is included in a single-page load. This includes all of the CSS, HTML, and JavaScript. The page does not reload, and it requires dynamic interaction with the Web server.**

# NAME THE KEY CONCEPTS IN REACT

- ▶ Web Components
- ▶ 1 way data binding
  - ▶ Virtual DOM
- ▶ JSX (which requires a "build" step)
- ▶ State management (Flux or Redux)

# COMPONENTS

- ▶ **Components let you split the UI into independent, reusable pieces, and think about each piece in isolation.**

# IDENTIFY "CONTAINER" (SMART) COMPONENTS

The screenshot shows the Galvanize Learn web application. The interface is divided into three main sections, each highlighted with a red box and a label:

- Header Component:** The top navigation bar, including the Galvanize logo, the word "LEARN", and user controls like "ADMIN", "COHORT: REACT", and "JEFF DEAN".
- Sidebar Component:** The left-hand navigation menu, showing "UNIT 3 Components" and a list of lessons and exercises, including "Unit Overview", "LESSON 01 React Components", "EXERCISE 01 Shopping Cart: Header / Footer", "LESSON 02 Props", "EXERCISE 02 Shopping Cart: Copyright with Props", and "LESSON 03 JSX Collections".
- Content Component:** The main content area, titled "LESSON 11 Identifying Components". It includes a "Github" button and a list of learning objectives: "By the end of this lesson you should be able to:" followed by "1. Take a wireframe or design and identify possible components" and "2. Identify list components".

At the bottom of the content area, there are links for "RESOURCES" and "NEXT: JSX REFERENCE".

# IDENTIFY "PRESENTATION" (DUMB) COMPONENTS

The screenshot shows the Galvanize Learn web application. Red boxes and arrows highlight various UI elements identified as presentation components:

- Navigation Bar:** Includes the Galvanize logo, "LEARN" text, and user profile "JEFF DEAN".
- Top Menu:** Contains links like DASHBOARD, UNITS, ACTIVITY, PERFORMANCES, STUDENTS, CURRICULUM, and COHORT SETUP.
- Left Sidebar:** Features a "Unit Overview" section with a list of lessons and exercises, including "LESSON 01 React Components", "EXERCISE 01 Shopping Cart: Header / Footer", "LESSON 02 Props", "EXERCISE 02 Shopping Cart: Copyright with Props", and "LESSON 03 JSX Collections".
- Main Content Area:** Displays "LESSON 11 Identifying Components". Below the title, it lists learning objectives: "By the end of this lesson you should be able to: 1. Take a wireframe or design and identify possible components, 2. Identify list components".
- Annotations:** Red boxes and arrows point to specific components: "NEXT: UNIT 4", "DropdownMenu Component", "ForwardButton Component", "LinkList Component", "Link Components", "Github", and "NEXT: JSX REFERENCE".

# PASS DATA AS PROPS AND RENDER DYNAMIC VALUES IN JSX

## ► Props are Read-Only

```
class Welcome extends React.Component {  
  render() {  
    return <h1>Hello, {this.props.name}</h1>;  
  }  
}
```



# DIFFERENTIATE BETWEEN FUNCTIONAL AND CLASS COMPONENTS

```
function Welcome(props) {  
  return <h1>Hello, {props.name}</h1>;  
}
```

VS

```
class Welcome extends React.Component {  
  render() {  
    return <h1>Hello, {this.props.name}</h1>;  
  }  
}
```

# KNOW WHEN TO USE STATE VS SETSTATE

- ▶ Use state to assign initial state in a constructor

```
constructor(props) {  
  super(props)  
  this.state = {  
    cards: CardData  
  }  
}
```

# KNOW WHEN TO USE STATE VS SETSTATE

- Use `setState` to update state and re-render the component

```
onSubmit = (card) => {  
  this.setState({  
    cards: this.state.cards.concat(card)  
  })  
}
```

# LIFT STATE UP TO THE CLOSEST COMMON ANCESTOR

- ▶ Single “source of truth” in React apps
  - ▶ Rely on the top-down data flow
- ▶ Avoid duplicating lifted state in child components

# CODE ALONG

- ▶ Clone (or Download) this repo
- ▶ We will work together to wire up these components using props, state, and setState
- ▶ Here is a video of the code along that you can refer to afterwards

# OBJECTIVES

- ▶ **Describe Single Page Applications (SPA)**
  - ▶ **Name the key concepts in React**
  - ▶ **Explain components, props, and JSX**
- ▶ **Identify functional and class components**
  - ▶ **Know when to use state vs setState**
- ▶ **Lift state up to the closest common ancestor**