FRONT END TRAINING – DAY 3

Impact Byte - Nexsoft

Today’s Material

- React Overview
- React Components
- JSX
- Hands-on Practice

Web Development – Before React

- Focus is based on print practice – everything is a page and we make the page interactive
- Page is the single smallest element that we design and develop
- Layouting UI elements
- Make it responsive according to the page width
- Troublesome to make a consistent UI

Web Development – After React

- Focus is based on creating application
- Component is the single smallest element of work
- Component is reusable throughout the project, generally doesn't care whether it's website, web app, mobile web app, desktop app, etc
- Responsive aspect is still handled by Media Query

React

- React is an open source JavaScript library introduced by Facebook
- It’s a UI component library. The UI components are created with React using JavaScript, not a special template language. This approach is called creating composable UIs, and it’s fundamental to React’s philosophy.
- React UI components are highly self-contained, concern-specific blocks of functionality.
- For example, there could be components for date-picker, captcha, address, and ZIP code elements. Such components have both a visual representation and dynamic logic.
- Some components can even talk to the server on their own: for example, an autocomplete component might fetch the autocompletion list from the server.2

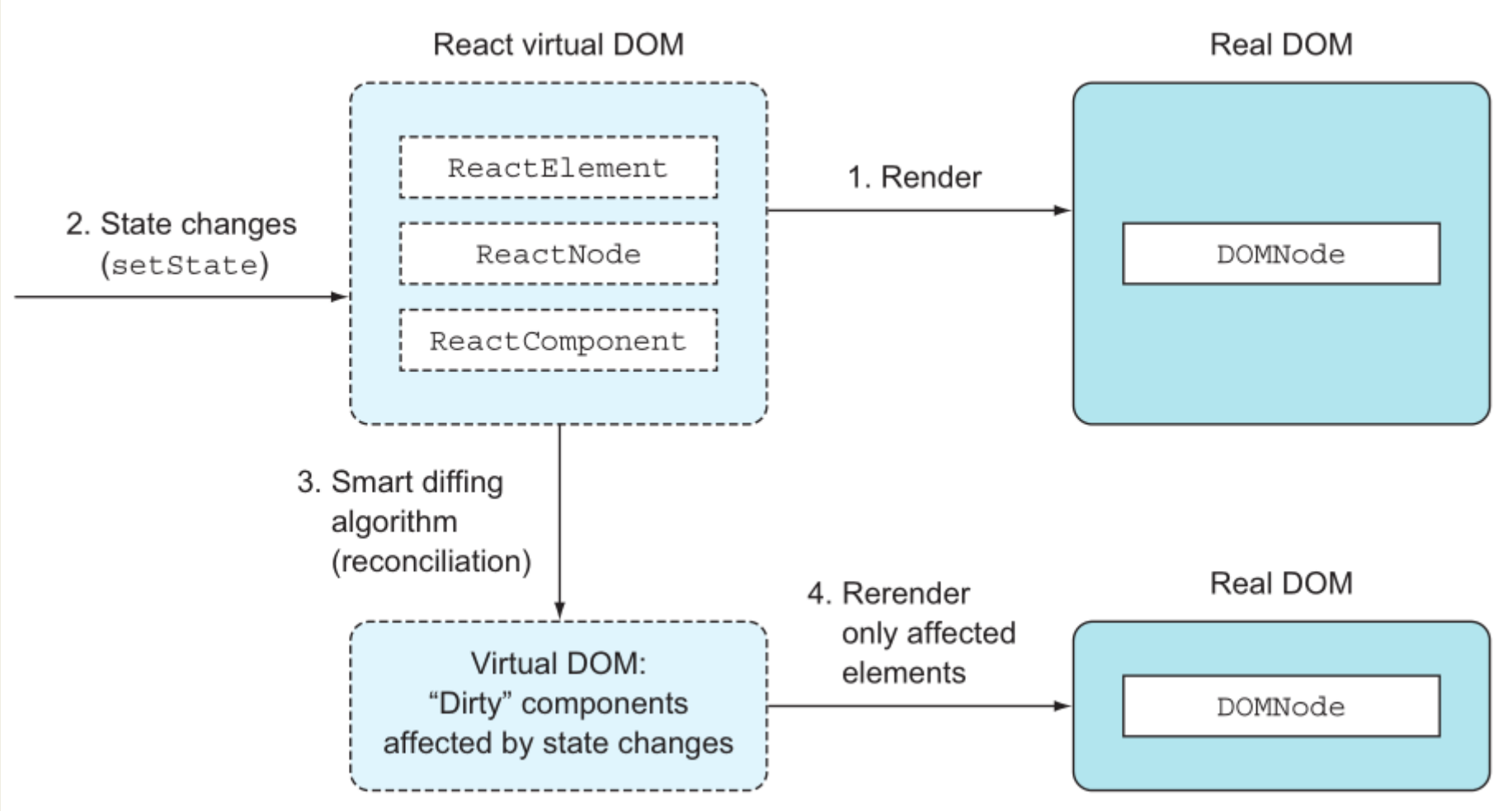
The Problem that React Solves

- Building and managing complex web UIs for front-end applications
- Handling UI for large applications with data that changes over time

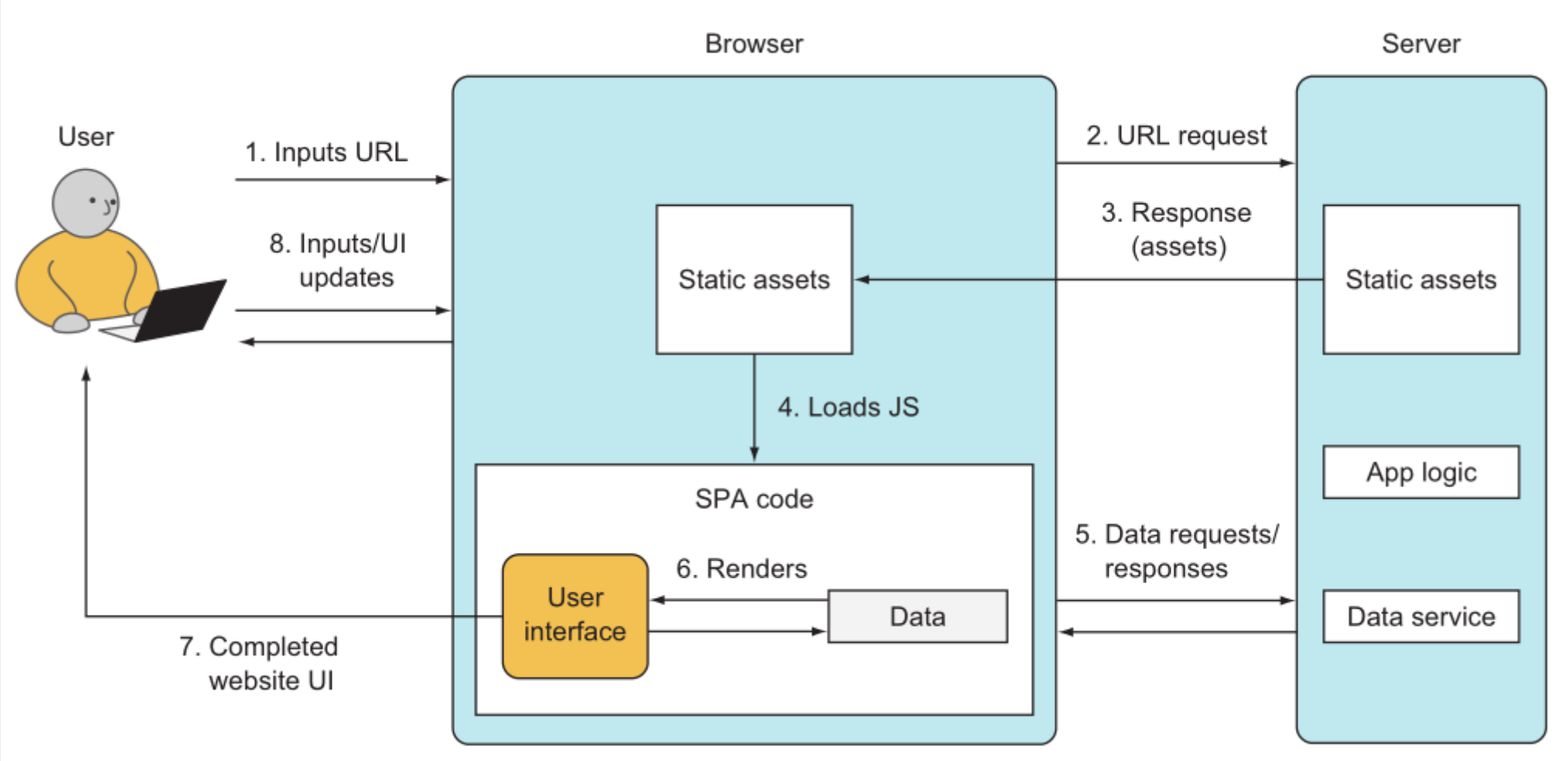
Benefits of Using React

- Simplicity – Declarative, Component-based Architecture in JS, powerful abstractions
- Speed and Testability
- Ecosystem and Community

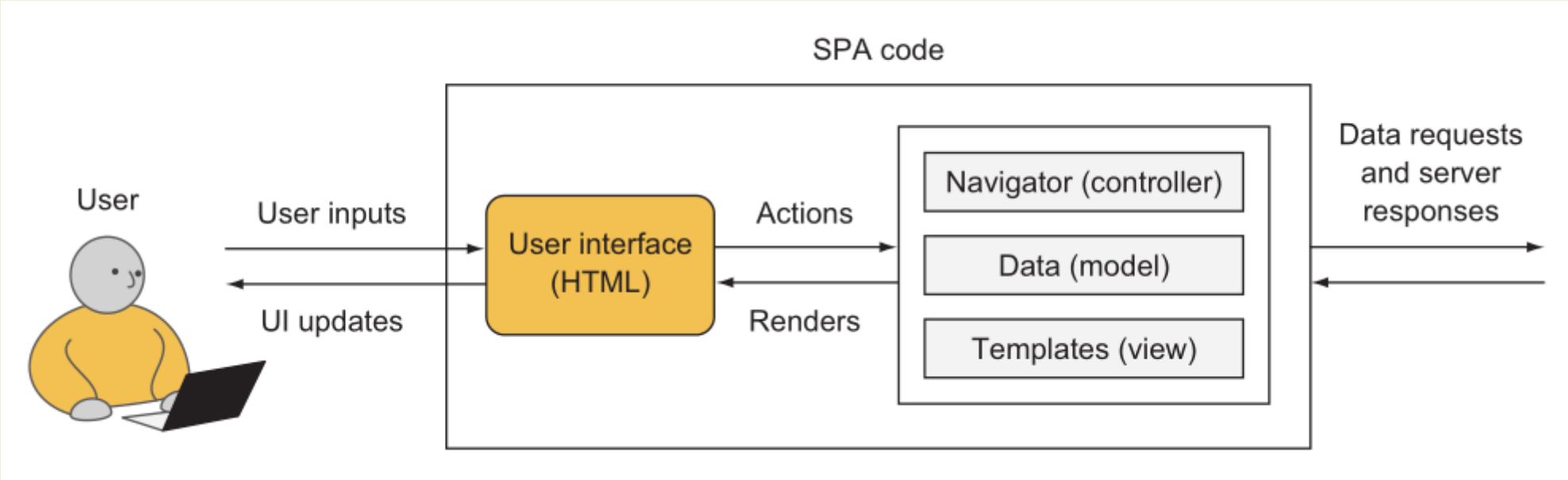
React and DOM



Typical React Architecture



Front End Architecture

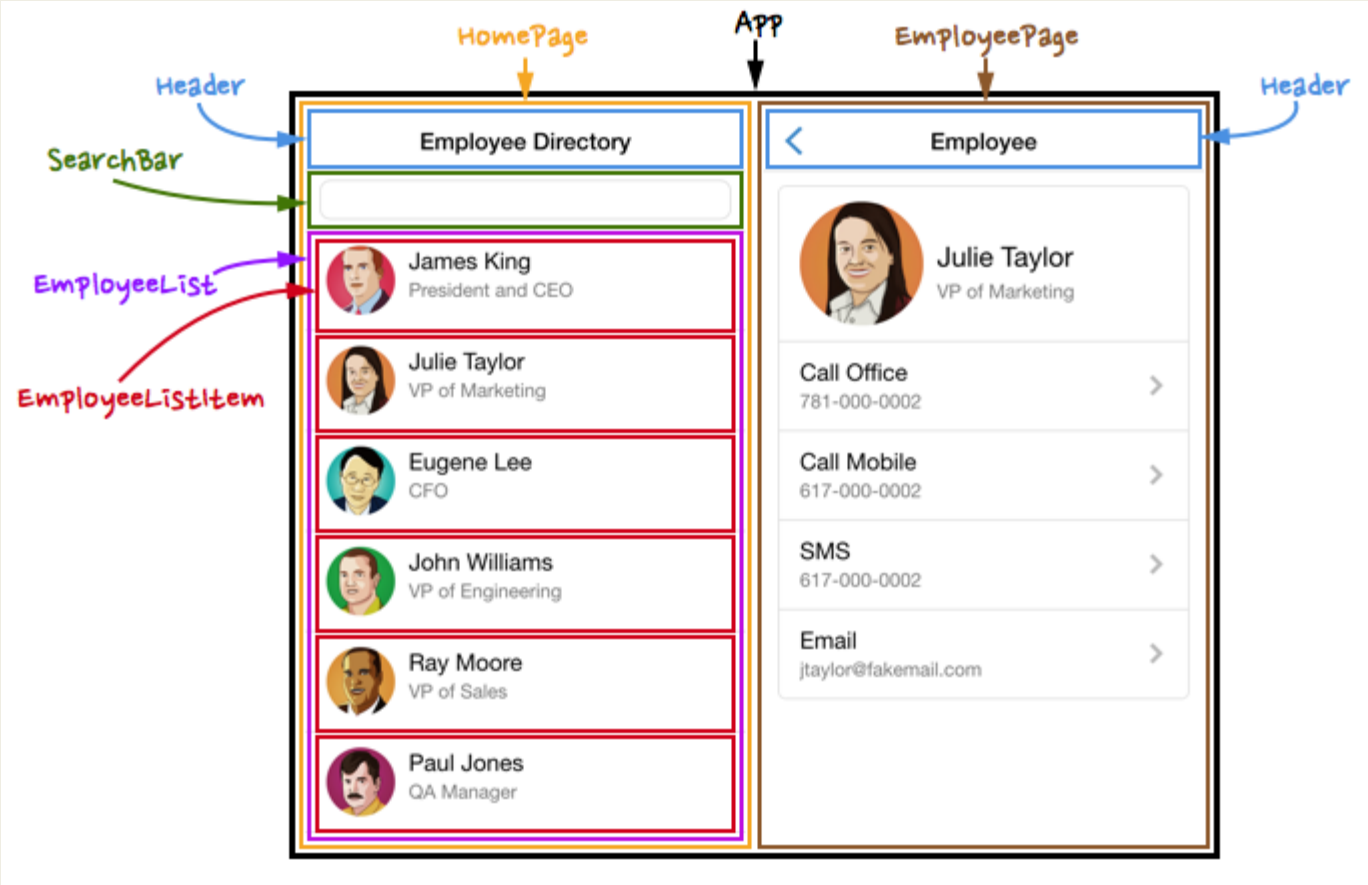


LET’S SETUP REACT AND CREATE A HELLO WORLD

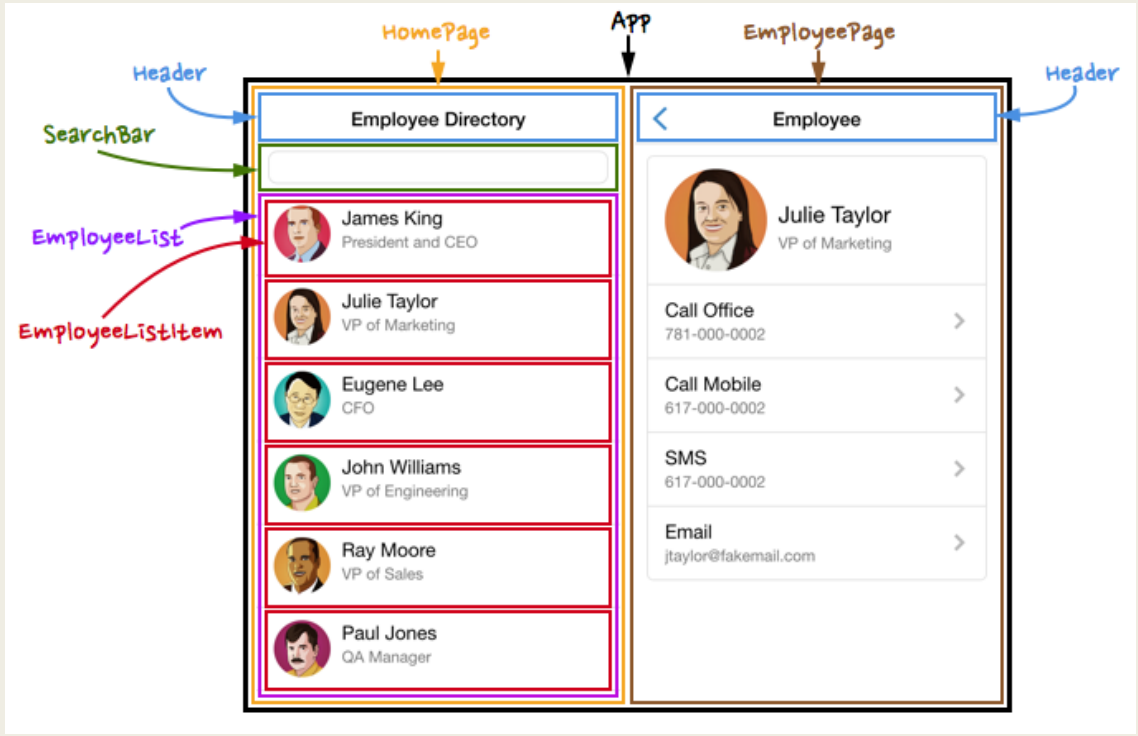
React Component

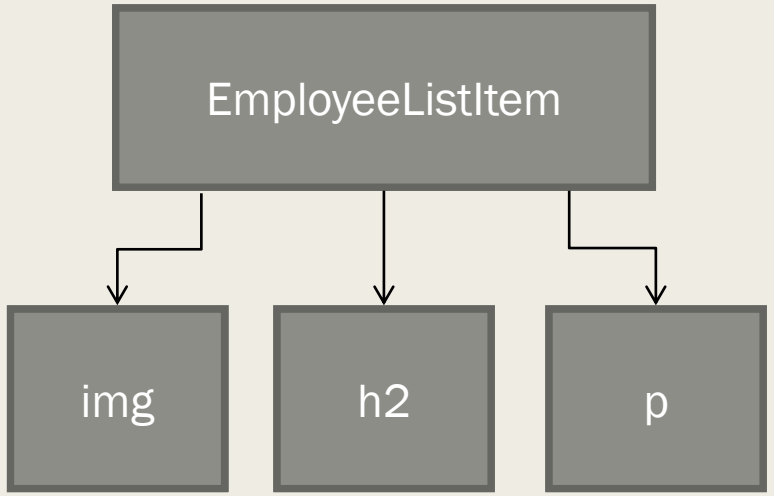
- React Component is a single smallest bit of working item in React
- It can and should be reusable
- It can also be composed to form a bigger UI component
- A single React Component is represented as a single HTML element (usually a)

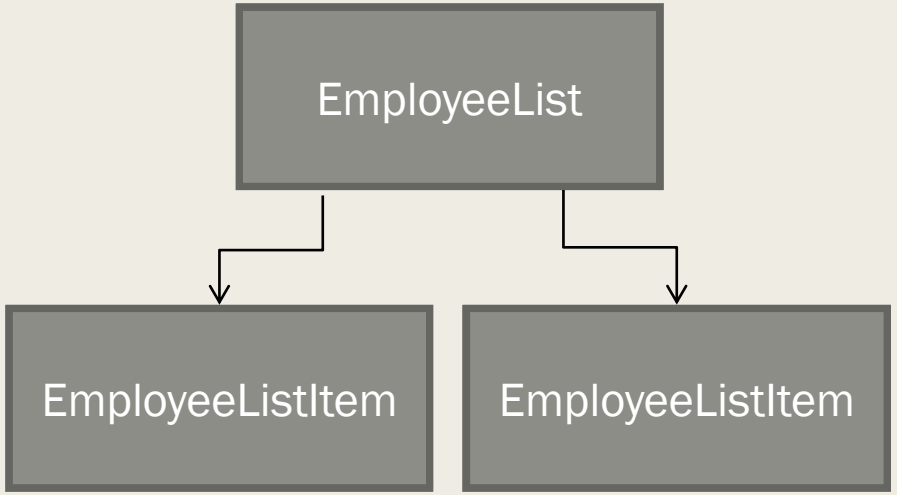
React Component



React Component

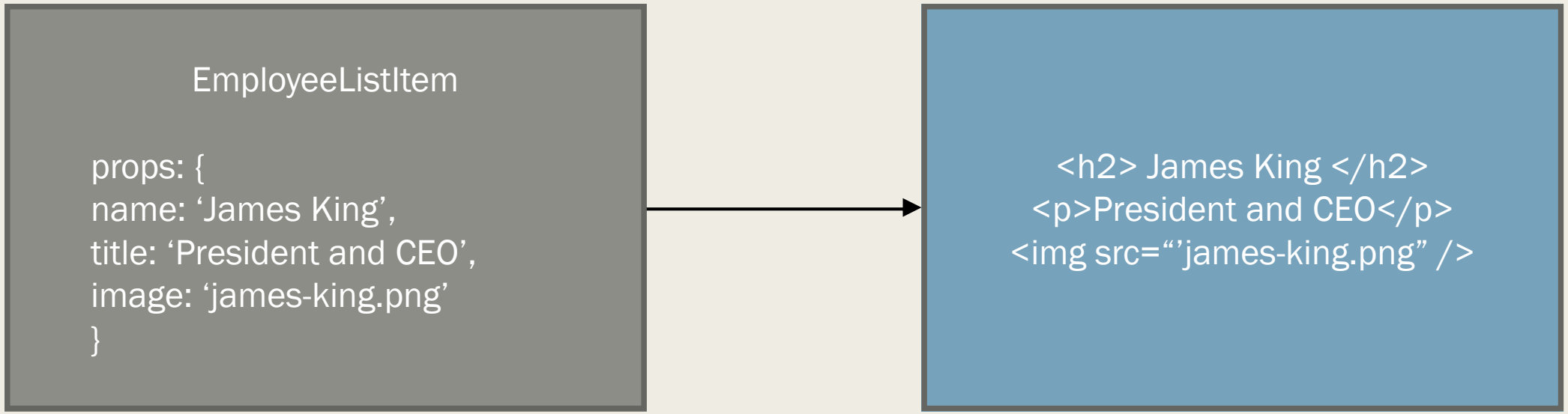






React Props

- A detailed property of a React component, basically tells a component what it has to display
- Essential to create a reusable component



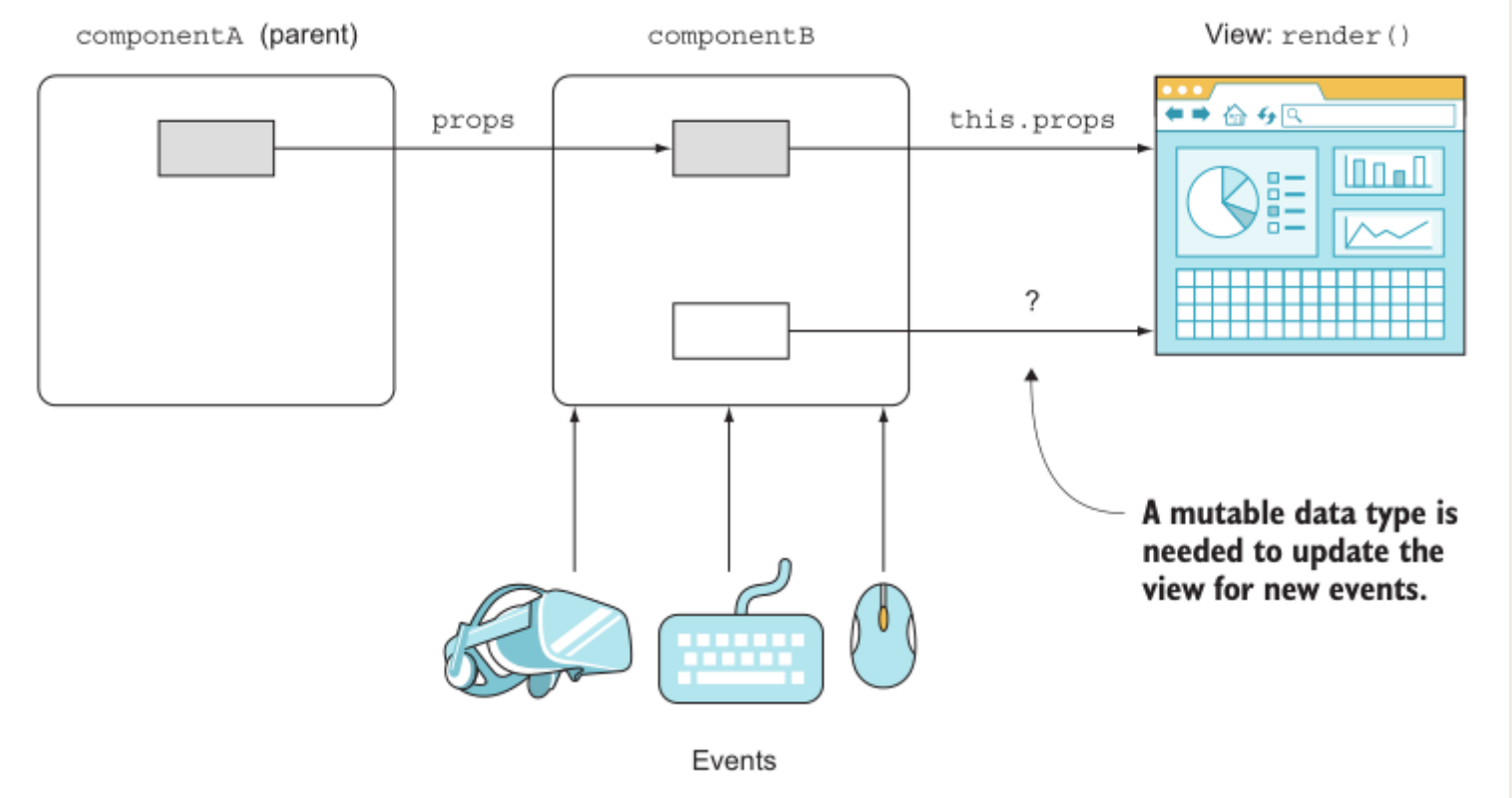
React State

- A React state is a mutable data store of components—self-contained, functionality centric blocks of UI and logic.
- Mutable means state values can change.
- By using state in a view (render()) and changing values later, you can affect the view’s representation.

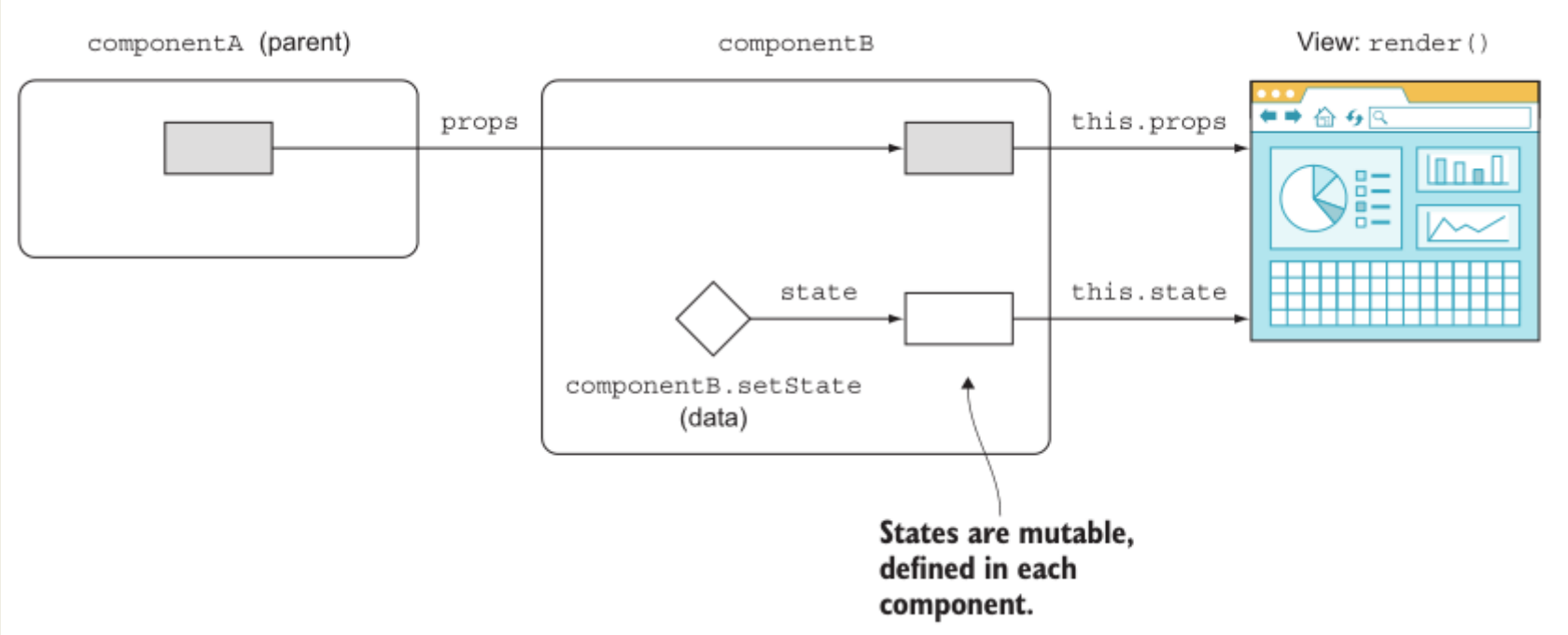
React State and Props

- State is mutable while props is immutable
- you pass properties from **parent components**,
- whereas you define states in the **component itself**, not its parent.

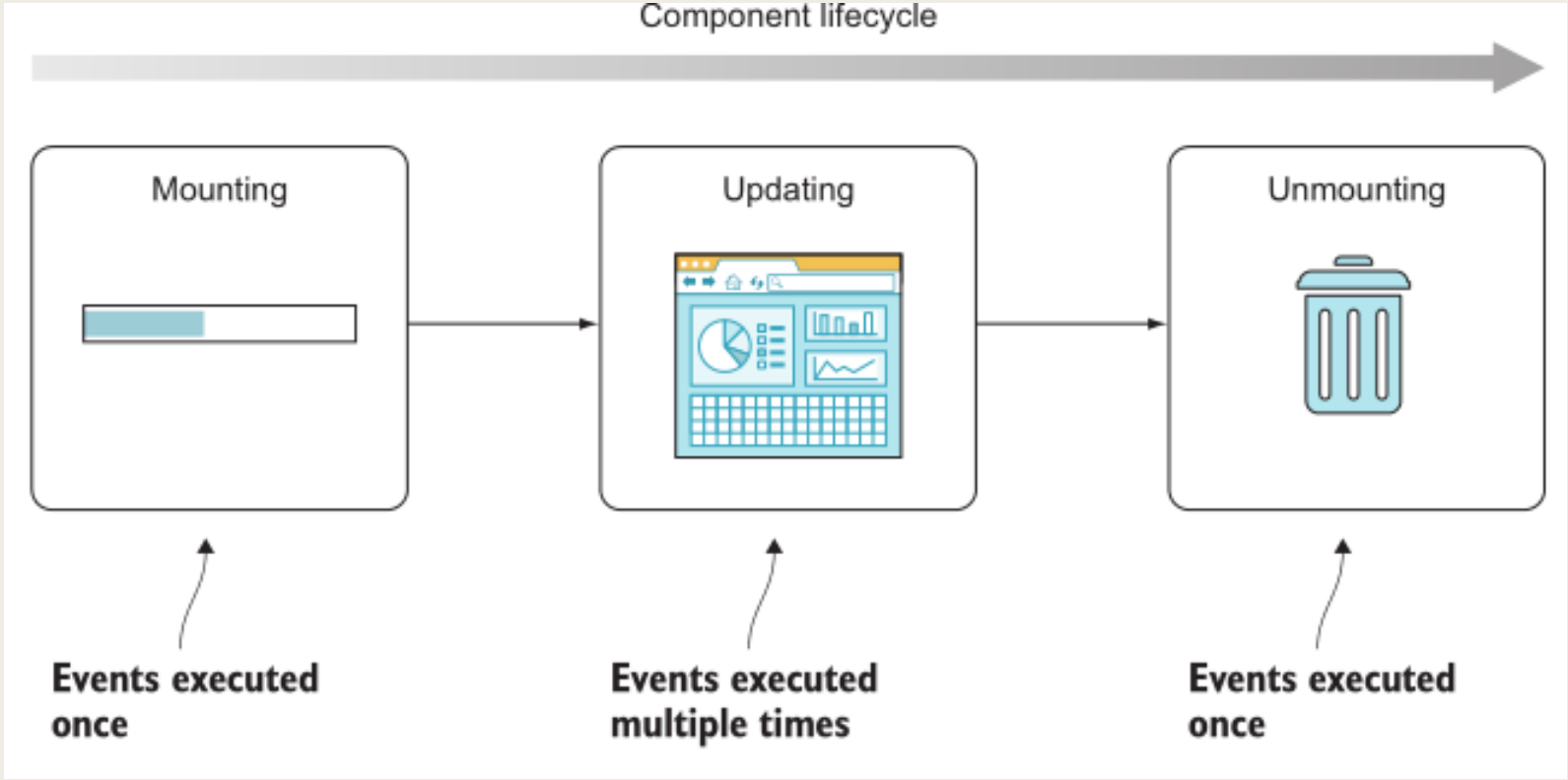
React State


componentA (parent)
componentB
View: render()
props
this.props
?
A mutable data type is needed to update the view for new events.
Events

React State


componentA (parent)
componentB
View: render()
props
this.props
state
this.state
componentB.setState
(data)
States are mutable,
defined in each
component.

React Lifecyle



FRONT END TRAINING – DAY 4

Impact Byte - Nexsoft

Today’s Material

- JS Array Operation
- React Router
- Create A React To-Do App

JavaScript Array Operators

The filter() method creates a new array with all elements that pass the test implemented by the provided function.

```
const words = ["spray", "limit", "elite", "television",
"destruction", "present"];

const longWords = words.filter(word => word.length > 6);
```

JavaScript Array Operators

- The map() method creates a new array with the results of calling a provided function on every element in the calling array.

```
const numbers = [2, 4, 8, 10];
const halves = numbers.map(x => x / 2);


let halves = [];
for(let i = 0; I < numbers.length; i++) {
  newNum = numbers[i]/2;
  halves.push(newNum);
}
```

JavaScript Array Operators

- forEach() is an Array iteration method that we can use to execute a function on each element in an array.

```
const numbers = [2, 4, 8, 10];

numbers.forEach(x => x / 2);
```

JavaScript Array Operators

- The reduce() method applies a function against an accumulator and each element in the array (from left to right) to reduce it to a single value.

```
const total = [0, 1, 2, 3].reduce((sum, value) => sum + value, 1);
```

React Router

- A React library that is used specifically to create route inside a single page application, so user can go to specific page/view just like going between pages inside a web site
- Basically, React Router generates all route as a React component
- Thus, using it is like using a React component

React Router Workflow

1. **Create a mapping in which URL s will translate into React components** (which turn into markup on a web page). In React Router, this is achieved by passing the path and component properties as well as nesting Route . The mapping is done in JSX by declaring and nesting Route components.
2. **Use the React Router's Router and Route components**, which perform the magic of changing views according to changes in URL s
3. **Render Router** on a web page by mounting it with ReactDOM.render() like a regular React element

React Router Example

```
render() {
    <Router>
        <Route path="/" component={Content} >
            <Route path="/about" component={About} />
                <Route path="/about/company" .../>
                <Route path="/about/author" .../>
            <Route path="/posts" component={Posts} />
            <Route path="/posts/:id" component={Post}/>
            <Route path="/contact" component={Contact} />
        </Route>
    </Router
}
```

React Router in Action

- Example, we have 3 components:
  - Home -> renders ‘Home’
  - Users -> renders ‘User Name’
  - Contact -> renders ‘User Contact’
- These 3 will be rendered in index.js

```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'
import { Route, Link, BrowserRouter as Router } from
'react-router-dom'
import Home from ‘./Home'
import Users from './users'
import Contact from './contact’

const routing = (
  <Router>
    <div>
      <Route path="/" component={Home} />
      <Route path="/users" component={Users} />
      <Route path="/contact" component={Contact} />
    </div>
  </Router>
)
ReactDOM.render(routing,
document.getElementById('root'))
```

React Router in Action

- But, if we go to localhost:3000/users, we can see Home is rendered as well
- React Router can use exact to go to the exact path

```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'
import { Route, Link, BrowserRouter as Router } from
'react-router-dom'
import Home from ‘./Home'
import Users from './users'
import Contact from './contact’

const routing = (
  <Router>
    <div>
      <Route exact path="/" component={Home} />
      <Route path="/users" component={Users} />
      <Route path="/contact" component={Contact} />
    </div>
  </Router>
)
ReactDOM.render(routing,
document.getElementById('root'))
```

React Router in Action

- To implement link that lets user go to a certain route, we use the Link component

```
<Router>
  <div>
    <ul>
      <li>
        <Link to="/">Home</Link>
      </li>
      <li>
        <Link to="/users">Users</Link>
      </li>
      <li>
        <Link to="/contact">Contact</Link>
      </li>
    </ul>
    <Route exact path="/" component={Home} />
    <Route path="/users" component={Users} />
    <Route path="/contact" component={Contact} />
  </div>
</Router>
```

React Router in Action

- To render only available component and create a fallback when the requested route is not available, we can use Switch component
- In this case we can use our custom `NotFound` component to generate a 404 page

```
<Router>
   <div>
     <ul>
       <li>
         <Link to="/">Home</Link>
       </li>
       <li>
         <Link to="/users">Users</Link>
       </li>
       <li>
         <Link to="/contact">Contact</Link>
       </li>
     </ul>
     <Switch>
       <Route exact path="/" component={App} />
       <Route path="/users" component={Users} />
       <Route path="/contact" component={Contact} />
       <Route component={Notfound} />
     </Switch>
   </div>
 </Router>
```

React Router in Action

- To generate dynamic URL, say we want to create custom content for each `User` component, we can use URL Parameter inside React Router
- Then the passed parameter will be processed in the `User` component with match

```
class Users extends Component {
  render() {
    const { params } = this.props.match
    return (
      <div>
        <h1>Users</h1>
        <p>{params.id}</p>
      </div>
    )
  }
}
```

```
<Router>
    <div>
      <ul>
        <li>
          <Link to="/">Home</Link>
        </li>
        <li>
          <Link to="/users">Users</Link>
        </li>
        <li>
          <Link to="/contact">Contact</Link>
        </li>
      </ul>
      <Switch>
        <Route exact path="/" component={App} />
        <Route path="/users/:id" component={Users} />
        <Route path="/contact" component={Contact} />
        <Route component={Notfound} />
      </Switch>
    </div>
  </Router>
```

React Router in Action

- To implement nested routing (i.e. we want to create specific route for /users/1, /users/2, we can expand the previous method
- The main implementation is in the Users.js file

```
<Router>
   <div>
     <ul>
       <li>
         <Link to="/">Home</Link>
       </li>
       <li>
         <Link to="/users">Users</Link>
       </li>
       <li>
         <Link to="/contact">Contact</Link>
       </li>
     </ul>
     <Switch>
       <Route exact path="/" component={App} />
       <Route exact path="/users" component={Users} />
       <Route path="/contact" component={Contact} />
       <Route component={Notfound} />
     </Switch>
   </div>
 </Router>
```

React Router in Action

- To implement nested routing (i.e. we want to create specific route for /users/1, /users/2, we can expand the previous method
- The main implementation is in the Users.js file
- In the end, we are just nesting React Components

```
import React from 'react'
import { Route, Link } from 'react-router-dom'
const User = ({ match }) => <p>{match.params.id}</p>
class Users extends React.Component {
  render() {
    const { url } = this.props.match
    return (
      <div>
        <h1>Users</h1>
        <strong>select a user</strong>
        <ul>
          <li>
            <Link to="/users/1">User 1 </Link>
          </li>
          <li>
            <Link to="/users/2">User 2 </Link>
          </li>
          <li>
            <Link to="/users/3">User 3 </Link>
          </li>
        </ul>
        <Route path="/users/:id" component={User} />
      </div>
    )
  }
}
export default Users
```