Week 2 Workshop

React Course

Agenda

Activity	Time
Get Prepared: Log in to Nucamp Learning Portal • Slack • Screenshare	10 minutes
Check-In	10 minutes
Weekly Recap: JSX, Elements, Components, Props, State & More!	40 minutes
Task 1 & 2	60 minutes
BREAK	15 minutes
Task 2 & 3	90 minutes
Check-Out	15 minutes

Check-In

- How was this week? Any particular challenges or accomplishments?
- Did you understand the Exercises and were you able to complete them?
- You must complete all Exercises before beginning the Workshop Assignment. We will review some of the Exercises together next, along with this week's concepts.

Welcome to React!

Week 1 Recap – Overview

Most of the New Concepts You Learned This Week

- create-react-app
- Reactstrap
- JSX & React Elements
- React Components
- Import & Export

- Props & State
- Lists and Keys
- Lifting State Up
- React Dev Tools

Next slides will review these concepts,
along with the Exercises you did this week.

create-react-app *Exercise Review: Getting Started with React*

You learned to create a new React app. You:

- Used yarn/npm to install create-react-app globally
  - Whichever you choose, yarn or npm, stick with the same one throughout your course! Switching midway will cause issues.
- Used `create-react-app nucampsite` to create the project folder named "**nucampsite**" a basic React app scaffolded out as a starting point.
- You do not have to use React this way, but it's a good place to start
- `yarn start` or `npm start`

JavaScript Import and Export

- import/export allows you to share code between JavaScript files
- Files that have at least one export are called modules
- You can have default or named exports
- Named exports must be imported with { } around the name
  - Example: `import { Navbar } from 'reactstrap';`
- Default exports do not need the { }
  - Example: `import App from './App';`
- You can get away with just using the package name after the 'from' for modules that are kept in the node_modules folder, such as 'reactstrap' and 'react'.
- For modules outside the node_modules folder, such as in your Components or shared folders, you need to give the filepath relative to the current file. Examples:
  - `'./App'` means to look for an App.js file in the same folder as the file that's doing the importing
  - `'./shared/campsites'` means look for a folder named shared in the same folder as the file that's doing the importing, then look for a campsites.js file in it.

React App Overview

- Create-react-app added:
  - App.js, index.js, index.html
- In App.js, note:
  - Last line: `export default App;`
- In index.js, note:
  - `import App from './App';`
  - `ReactDOM.render(<App />, document.getElementById('root'));`
- In index.html, note:
  - line 28: `<div id=”root”>`
- In summary: App.js exports App component, index.js uses the div id "root" inside index.html to send the data from App.js there. See next slide for a visual guide.

React App Overview

App.js

```
export default App
```

index.js

```
import App
ReactDOM.render(<App />,
document.getElementById('root'));
```

index.html

```
<div id=”root”>
```

reactstrap *Exercise Recap: Configuring React*

- Library of React + Bootstrap components.
  - Integrates Bootstrap 4 components and React together
  - Not the only way to do this, there are others, but we will use this one
  - We will also use just plain Bootstrap at times, without use of Reactstrap
- In the *Configuring React* Exercise, you installed reactstrap & used it to add a Navbar

JSX (JavaScript syntax extension)

- Helps visualize UI inside JavaScript, easier to read
- Similar to HTML syntax, some differences:
  - attribute `className` instead of `class`, `htmlFor` instead of `for`
  - Multi-word attribute names in camelCase, e.g. `onClick`, `encType`, `autoPlay`
- Transpiles to JavaScript: Fundamentally, JSX is syntactic sugar for the `React.createElement(component, props, ...children)` function (makes it simpler to use). No actual HTML here, just smoke and mirrors.
- Used in React to create JavaScript elements (which represent DOM elements)
- Elements are the smallest building block of a React component

OK but what's the point

- Why are we going to all this trouble to make everything into JavaScript objects? Wasn't a fine the way it was, just living out its life in the HTML world?
- React elements are used to create a Virtual DOM that mirrors the real DOM in memory
- The Virtual DOM is a key concept in React. Its main purpose is to make it faster and more efficient to control and update the real DOM.
- Vue, Ember, and other JS libraries/frameworks also use a Virtual DOM. You'll learn more about it later!

React Components

- React Components
  - Each must return a single React element or component (which can contain more elements/components)
  - Each component represents a part of the view in the browser
  - Can contain information about application state, can pass that information to child components
  - Enable you to split your UI into independent, reusable pieces
  - Always capitalized, like App or Directory
  - Called in React using JSX, can be self-closing or have both start and end tags,
    - e.g. `<App />` or `<NavBar> … </NavBar>`

React Components *(cont)*

- Basic structure of a React class component:
  - `import React, { Component } from 'react';`
  - `class Directory extends Component`
  - Add a constructor (*optional*) with `constructor(props)` and `super(props)`
  - Add `render()` method which `returns` a React element or component.
  - `Export` the component so it can be imported elsewhere , e.g. `export default App;`

Exercise Recap: React Components Part 1

- In this exercise, you:
  - Created a Directory component and stored an array of campsites in its constructor
  - Rendered each campsite from the array using the map method, giving each campsite a unique key
  - Imported and used the Directory component in App

Lists and Keys

- To build a list from an array, you will typically use the map method on the array, then in the callback function, format each array item with JSX.
- Note that "list" here does not necessarily mean the type of list you build with <ol>   elements (though it can!) - it means any time you go through an array, pull out each array item, and display them in an identical way, whether that's through s or s or s or <p>s, etc.
- Keys must be given to the top-level element inside the array while mapping
  - Helps identify which items have changed, are added, or removed
  - Key can be anything but must be unique (within the list) and unchanging. Usually, you would have ID numbers in the data that can be used.
- A couple examples:

```
<div>
    {users.map(user =>
        <p key={user.id}>
            {user.name}
        </p>
    }
</div>
```

```
<ul>
    {users.map(user =>
        <li key={user.id}>
            <p>{user.name}</p>
            <p>{user.city}</p>
        </li>
    )}
</ul>
```

Props

- Read-only data passed from parent to child component from parent component's state
- Short for 'properties'.
- When a parent component passes props to a child, it looks similar to an HTML element with attributes:
  - Format: `<ChildComponentName attribute1=someData attribute2=someMoreData />`
- Even if multiple attributes are passed in, only one props object is received inside the child component
- Access the data from inside child like so: `this.props.attribute1`
- Ex.: If the parent component renders: `<Instructor name="Karim" />` then `this.props.name` is "Karim" inside the child Instructor component.

Props (cont)

More examples:

```
<Album date={myDate} />
```

- Here, we are passing the value of *myDate* into the Album component, where it can be accessed as: `this.props.date`

```
<Directory campsites={this.state.campsites} />
```

- `This.state.campsites` is passed into the Directory component, where it can be accessed as `this.props.campsites`

```
<CampsiteInfo campsite={this.state.campsite} comments={this.state.comments}
/>
```

- Here, the state campsite and comment data is passed into the CampsiteInfo component and can be accessed there as `this.props.campsite` and `this.props.comments`

State

- A class component can store its own local, private information in its **state** (usually in constructor as **this.state**). Example:

To update - cannot directly change object properties, must go through **this.setState()**, e.g.:

```
this.setState({selectedCampsite: campsite});
```

Never do something like:

~~`this.state.selectedCampsite = campsite;`~~

```
class Welcome extends Component {
  constructor(props) {
    super(props);
    this.state = {name: "Nucamp"};
  }
  render() {
    return (
        <h1>Welcome to {this.state.name}!</h1>
    );
  }
}
```

Lifting State Up

- When multiple components need access to the same data that is stored in one as state, then that state is "lifted up" to the state of a common parent component.
- The common parent component then passes the state data as props to its child components.

Exercise Recap: React Components Part 2

- In this exercise, you saw an example of lifting state information from a child component to its parent component.
  - You set up a file named campsites.js inside a **shared** folder and removed the campsites data from DirectoryComponent.js.
  - You updated the Directory component to get its campsites data from **props** passed in by its parent component (App).
  - You also updated the App component to grab its **state** data from the campsites.js file then pass it down to its child component Directory as props.

React Dev Tools

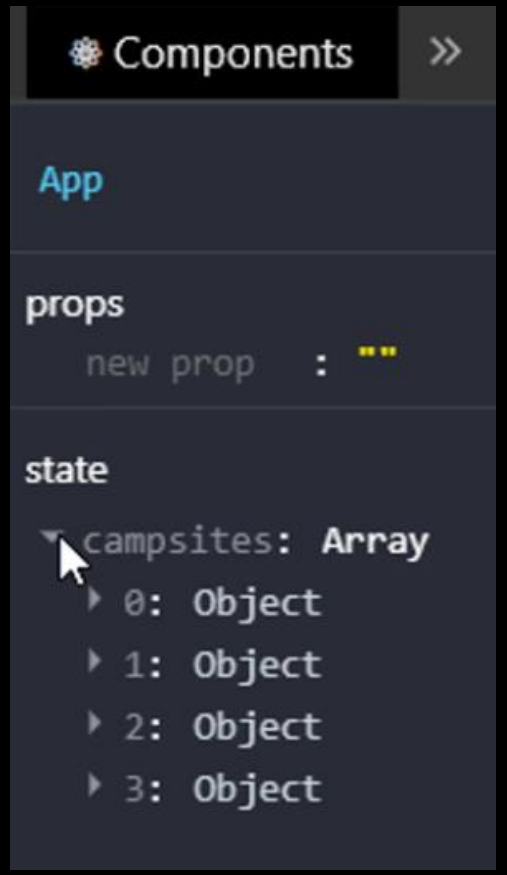
- Has everyone installed it?
- If not, take a moment to install the version for your browser now
- Start your app with yarn or npm from your project folder (3-React/nucampsite)
- Open your React Dev Tools Components tab in the same browser tab as your app (continued next slide)

React Dev Tools (cont)

In the Devtools' Components tab, click the App component, and note that State contains the campsites array of 4 objects, as seen to the right.

Next, click the Directory component in the Components tab and note that it has been passed the campsites array as Props, and its state has selectedCampsite as null initially.

Click a photo of a campsite in the app and observe what happens to Directory's state.



Workshop Assignment

- It's time to start the workshop assignment!
- Sit near your workshop partner.
  - Your instructor may assign partners, or have you choose.
- Work closely with each other.
  - 10-minute rule does *not* apply to talking to your partner, you should consult each other throughout.
- Follow the workshop instructions very closely
  - both the video and written instructions.
- Talk to your instructor if any of the instructions are unclear to you.

Check-Out

- Submit your two assignment files in the learning portal:
  - CampsiteInfoComponent.js
  - DirectoryComponent.js
- Wrap up – Retrospective
  - What went well
  - What could improve
  - Action items
- If there is time remaining, review any remaining questions from Week 2, or start Week 3, or work on your Portfolio Project.