React 2: Components & Props

IN608: Intermediate Application Development Concepts

Kaiako: Tom Clark & Grayson Orr

Last Session's Content

- React
 - Overview
 - o Node.js installation
 - NPM (Node package manager)
- Create React App
- JSX
 - Embedding expressions
 - Attributes
 - Injection attacks
 - Objects

Today's Content

- File structure
- Components
 - Function
 - o Class
- Props

File Structure

File Structure

- React does not have opinions on how you put files into directories
- There are a couple common approaches you may want to consider:
 - Group by features or routes
 - o Group by file type we are going to follow this approach

File Structure - Group By Features or Routes

Group by features or routes

```
common/
   APIUtil.js
   APIUtil.test.is
    Avatar.css
   Avatar.js
    Avatar.test.js
feed/
    Feed css
   Feed.js
   Feed.test.js
   FeedAPI.js
   FeedAPI.test.js
profile/
    Profile.css
   Profile.is
   Profile.test.is
   ProfileAPI.js
   ProfileAPI.test.js
```

File Structure - Group By File Type

Group by file type

```
api/
    APIUtil.js
    APIUtil.test.js
    FeedAPI.js
    FeedAPI.test.js
    ProfileAPI.js
    ProfileAPI.test.js
components/
    Avatar.css
    Avatar.js
    Avatar.test.js
    Feed.css
   Feed.js
    Feed.test.js
    Profile.css
    Profile.js
    Profile.test.js
```

- What are components?
 - User-interface split into independent, reusable chunks of code
 - Each chunk of code is self-contained
 - Accepts an arbitrary arguments called props & returns a React element
- A component can be written either as a function or a class
- React treats components starting with lowercase letters as DOM tags, i.e., <div />
- <App /> is an example of a component starts with a capital letter
- The following examples from React's point of view are equivalent
- Functional & class components both have additional features. We will look at this in the next session
- All components are store in src/components you will need to create the components directory

Function Component

• Function component example in Owner.js

```
import React from 'react'
const Owner = () => <h1>My owner is John Doe</h1>
export default Owner
```

Class Component

- Class component example in Owner.js
- Extends React.Component
- Avoid creating your own base component class
 - o In React components, code reuse is achieved through composition rather than inheritance

```
import React from 'react'

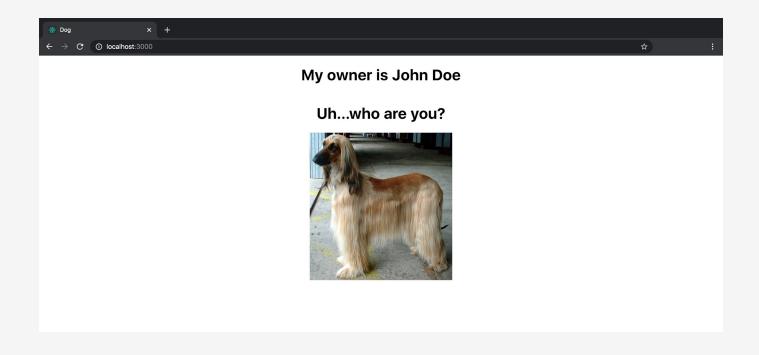
class Owner extends React.Component {
  render() {
    return <h1>My owner is John Doe</h1>
  }
}

export default Owner
```

• In App.js

```
import React from 'react'
import Owner from './Owner'
import afghanHoundImg from '../img/afghan-hound.jpg'
const App = () => {
 const dog = {
   name: 'Bingo',
   breed: 'Afghan Hound',
   img: afghanHoundImg,
 const formatDog = (dog) =>
    `Woof woof, my name is ${dog.name} & my breed is an ${dog.breed}`
  const getGreeting = (dog) => {
   if (dog) {
      return <h1>{formatDog(dog)}</h1>
    return <h1>Uh...who are you?</h1>
  return (
    <div className='main-container'>
      <Owner />
      {getGreeting()}
      <img src={dog.img} alt='afghan hound' width='300' />
    </div>
export default App
```

- What is happening in index.js?
 - ReactDOM.render() is called with the <App /> element
 - Returns a reference to the App component
 - ReactDOM updates the DOM



- When a function or class component is declared, its props (properties) must never be modified
- All React components must act like pure functions with respect to their props
- What does this mean? Its return value is the same for the same arguments & no side effects
- Function component with props example in Owner.js accepts props as an argument

```
import React from 'react'
const Owner = (props) => <h1>My owner is {props.name}</h1>
export default Owner
```

- Class component with props example in Owner.js
- No arguments. Use of the this keyword

```
import React from 'react'

class Owner extends React.Component {
  render() {
    return <h1>My owner is {this.props.name}</h1>
  }
}

export default Owner
```

- In App.js
- Updates the DOM by replacing props.name & this.props.name with Jane Doe

```
import React from 'react'
import Owner from './Owner'
import afghanHoundImg from '../img/afghan-hound.jpg'
const App = () => {
  const dog = {
    name: 'Bingo',
    breed: 'Afghan Hound',
    img: afghanHoundImg,
  const formatDog = (dog) =>
    `Woof woof, my name is ${dog.name} & my breed is an ${dog.breed}`
  const getGreeting = (dog) => {
    if (dog) {
      return <h1>{formatDog(dog)}</h1>
    return <h1>Uh...who are you?</h1>
  return (
    <div className='main-container'>
      <Owner name='Jane Doe' />
      {getGreeting()}
      <img src={dog.img} alt='afghan hound' width='300' />
    </div>
export default App
```

