React 2: Components & Props

IN608: Intermediate Application Development Concepts

Kaiako: Tom Clark & Grayson Orr

Last Session's Content

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

Today’s Content

- Components
  - Function
  - Class
- Props

Components

Components

- 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 elements
- 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 & requires `App` to be in scope. All you have to know is that it is a convention
- The following examples from React's point of view are equivalent
- In `src/`, create directory called `components` for your component files
  - Move `App.js` into `components`
  - Create a new component file called `Owner.js`
- Function & class components both have additional features. We will look at this in the next session

Function Component

- Function component example in `Owner.js`

```
import React from 'react'

function Owner() {
  return (
    <div className='container'>
      <h1>My owner is John Doe</h1>
    </div>
  )
}

export default Owner
```

Class Component

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

```
import React from 'react'

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

export default Owner
```

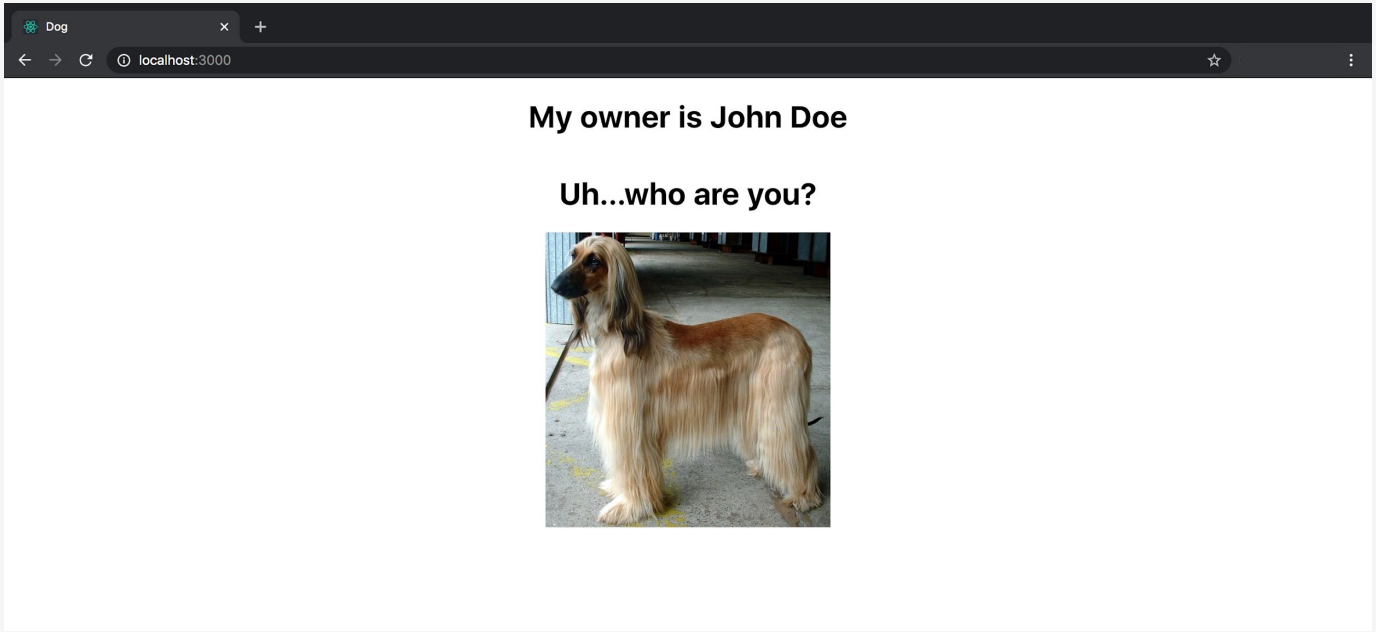
Components

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

```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'
import App from './App'

ReactDOM.render(
  <React.StrictMode>
    <Owner />
    <App />
  </React.StrictMode>,
  document.getElementById('root')
)
```

Components



Props

Props

- 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'

function Owner(props) {
  return (
    <div className='container'>
      <h1>My owner is {props.name}</h1>
    </div>
  )
}

export default Owner
```

Props

- 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 (
      <div className='container'>
        <h1>My owner is {this.props.name}</h1>
      </div>
    )
  }
}

export default Owner
```

Props

- What is happening in `index.js`?
  - `ReactDOM.render()` is called with the `<Owner name='Jane Doe' />` & `<App />` element
  - React calls the `Owner` component with `{name: 'Jane Doe'}` & `App` component
  - Returns a reference to the `Owner` & `App` component
  - `ReactDOM` updates the DOM by replacing `props.name` & `this.props.name` with `Jane Doe`

```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'
import App from './components/App'
import Owner from './components/Owner'

ReactDOM.render(
  <React.StrictMode>
    <Owner name="Jane Doe" />
    <App />
  </React.StrictMode>,
  document.getElementById('root')
)
```

Props