React 4: Events & Conditional Rendering

IN608: Intermediate Application Development Concepts

Kaiako: Tom Clark & Grayson Orr

Last Session’s Content

- State
- Lifecycle methods
  - Mounting
  - Updating
  - Unmounting
- Data flow
- React hooks

Today’s Content

- File structure
- Events
  - Binding
- Conditional rendering

Code Examples

- After today's lecture, we will no longer be using class components as code examples
- Instead, we will use function components & hooks code code examples
- For today's lecture, we have provided code examples in both component types

Events

Events

- Event handling with React elements is very similar to event handling on DOM elements
- There are two key syntax differences:
  - Camelcase is the naming convention for React events, not lowercase
  - With JSX, a function is passed as the event handler, not a string

```
// HTML
<button onclick="someEvent()"></button>

// JSX
<button onClick={someEvent}></button>
```

Binding via Constructor

- Create a new component file called `Register.js`
- Binding `this.handleRegisteredChange` to the component via constructor
- Renders a button which the user can toggle between a “Registered” & “Unregistered” state

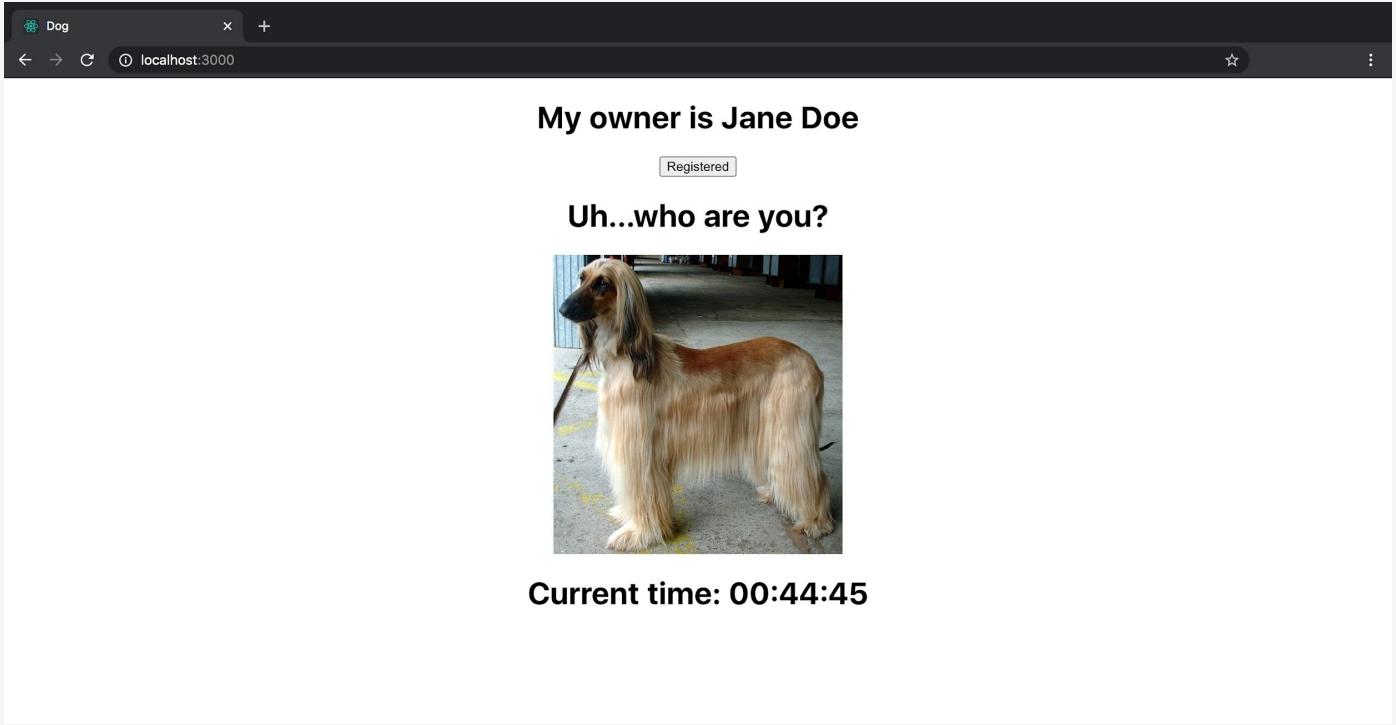
```
import React from 'react'

class Register extends React.Component {
  constructor(props) {
    super(props)
    this.state = { isRegistered: true }
    this.handleRegisteredChange = this.handleRegisteredChange.bind(this)
  }

  handleRegisteredChange() {
    this.setState((state) => ({ isRegistered: !state.isRegistered }))
  }

  render() {
    return (
      <button onClick={this.handleRegisteredChange}>
        {this.state.isRegistered ? 'Registered' : 'Unregistered'}
      </button>
    )
  }
}

export default Register
```

Events

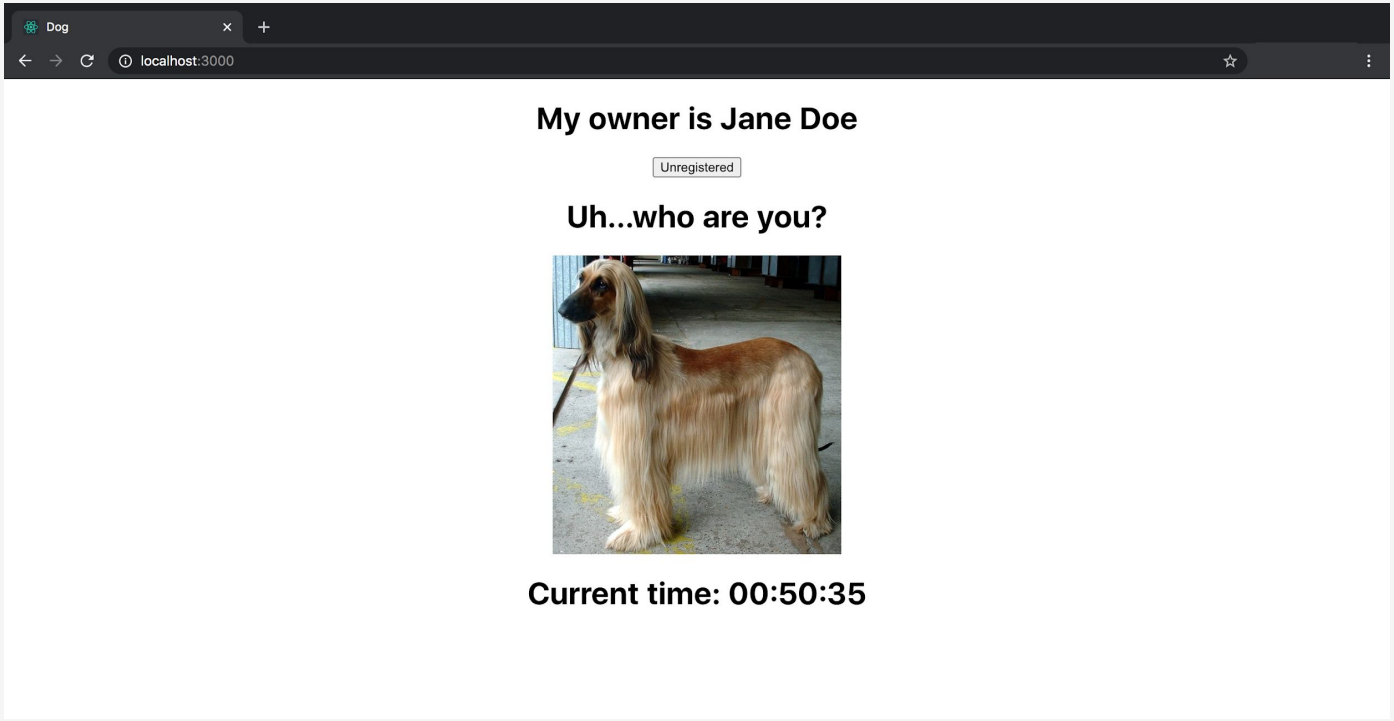
- In `App.js`

```
import React from 'react'
import Clock from './Clock'
import Owner from './Owner'
import Register from './Register'
import afghanHoundImg from '../img/afghan-hound.jpg'

const dog = {
  name: 'Bingo',
  breed: 'Afghan Hound',
  img: afghanHoundImg
}

function formatDog(dog) {
  return `Woof woof, my name is ${dog.name} & my breed is an ${dog.breed}`
}

function getGreeting(dog) {
  if (dog) {
    return <h1>{formatDog(dog)}</h1>
  }
  return <h1>Uh...who are you?</h1>
}

function App() {
  return (
    <div className='container'>
      <Owner />
      <Register />
      {getGreeting()}
      <img src={dog.img} alt='afghan hound' width='300' />
      <Clock />
    </div>
  )
}

export default App
```

Events


Dog
localhost:3000
My owner is Jane Doe
Registered
Uh...who are you?
Current time: 00:44:45

Events


Dog
localhost:3000
My owner is Jane Doe
Unregistered
Uh...who are you?
Current time: 00:50:35

Events

- In JavaScript, class methods are not bound by default
- If you do not bind, i.e., `this.handleRegisteredChange` & pass it to `onClick`, `this` will be `undefined` when the `bind` function is called
- Alternatively, there are three other ways to bind a callback without explicitly calling the `bind()` function in the `constructor()`

```
import React from 'react'

class Register extends React.Component {
  constructor(props) {
    super(props)
    this.state = { isRegistered: true }
  }

  handleRegisteredChange() {
    console.log(this) // undefined
  }

  render() {
    return (
      <button onClick={this.handleRegisteredChange}>
        {this.state.isRegistered ? 'Registered' : 'Unregistered'}
      </button>
    )
  }
}

export default Register
```

Binding via Render

- Binding `this.handleRegisteredChange` to the component via render

```
import React from 'react'

class Register extends React.Component {
  constructor(props) {
    super(props)
    this.state = { isRegistered: true }
  }

  handleRegisteredChange() {
    this.setState((state) => ({ isRegistered: !state.isRegistered }))
  }

  render() {
    return (
      <button onClick={this.handleRegisteredChange.bind(this)}>
        {this.state.isRegistered ? 'Registered' : 'Unregistered'}
      </button>
    )
  }
}

export default Register
```

Binding via Class Properties

- Binding `this.handleRegisteredChange` to the component via class properties (experiential)

```
import React from 'react'

class Register extends React.Component {
  constructor(props) {
    super(props)
    this.state = { isRegistered: true }
  }

  handleRegisteredChange = () => {
    this.setState((state) => ({ isRegistered: !state.isRegistered }))
  }

  render() {
    return (
      <button onClick={this.handleRegisteredChange}>
        {this.state.isRegistered ? 'Registered' : 'Unregistered'}
      </button>
    )
  }
}

export default Register
```

Binding via Arrow Function

- Binding `this.handleRegisteredChange` to the component via arrow function, i.e., `() => this.handleRegisteredChange()`

```
import React from 'react'

class Register extends React.Component {
  constructor(props) {
    super(props)
    this.state = { isRegistered: true }
  }

  handleRegisteredChange() {
    this.setState((state) => ({ isRegistered: !state.isRegistered }))
  }

  render() {
    return (
      <button onClick={() => this.handleRegisteredChange()}>
        {this.state.isRegistered ? 'Registered' : 'Unregistered'}
      </button>
    )
  }
}

export default Register
```

Conditional Rendering

Conditional Rendering

- Conditional rendering in React works the same as in JavaScript
  - You can use conditional statements, i.e., `if`, `else`, `else if`, `switch` or the ternary operator
- A component decides based on one or several conditions which element it will return, then later render
- Consider the following two components:

```
import React from 'react'

class GuestGreeting extends React.Component {
  render() {
    return <h1>Please sign up</h1>
  }
}

class UserGreeting extends React.Component {
  render() {
    return <h1>Welcome back</h1>
  }
}

export default GuestGreeting
export default UserGreeting
```

- **Note:** These two components are in separate files. `GuestGreeting.js` & `UserGreeting.js`
- We will create a component which renders either `GuestGreeting` or `UserGreeting` based on whether a user is logged in

Conditional Rendering

- Create a new component file called `Greeting.js`
- Renders a different greeting depending on the value of `isLoggedIn` prop

```
import React from 'react'
import GuestGreeting from './GuestGreeting'
import UserGreeting from './UserGreeting'

class Greeting extends React.Component {
  render() {
    const isLoggedIn = this.props.isLoggedIn
    return isLoggedIn ? <UserGreeting /> : <GuestGreeting />
  }
}

export default Greeting
```

Conditional Rendering

- Consider the following two components:

```
import React from 'react'

class LoginButton extends React.Component {
  render() {
    return <button onClick={this.props.onClick}>Login</button>
  }
}

class LogoutButton extends React.Component {
  render() {
    return <button onClick={this.props.onClick}>Logout</button>
  }
}

export default LoginButton
export default LogoutButton
```

- **Note:** These two components are in separate files

Conditional Rendering

- Create a new component file called `LoginControl.js`

```
import React from 'react'
import Greeting from './Greeting'
import LoginButton from './LoginButton'
import LogoutButton from './LogoutButton'

class LoginControl extends React.Component {
  constructor(props) {
    super(props)
    this.state = { isLoggedIn: false }
  }

  handleLoginClick() {
    this.setState({ isLoggedIn: true })
  }

  handleLogoutClick() {
    this.setState({ isLoggedIn: false })
  }

  render() {
    const { isLoggedIn } = this.state
    const button = isLoggedIn ? (
      <LogoutButton onClick={() => this.handleLogoutClick()} />
    ) : (
      <LoginButton onClick={() => this.handleLoginClick()} />
    )

    return (
      <React.Fragment>
        <Greeting isLoggedIn={isLoggedIn} />
        {button}
      </React.Fragment>
    )
  }
}

export default LoginControl
```

Conditional Rendering

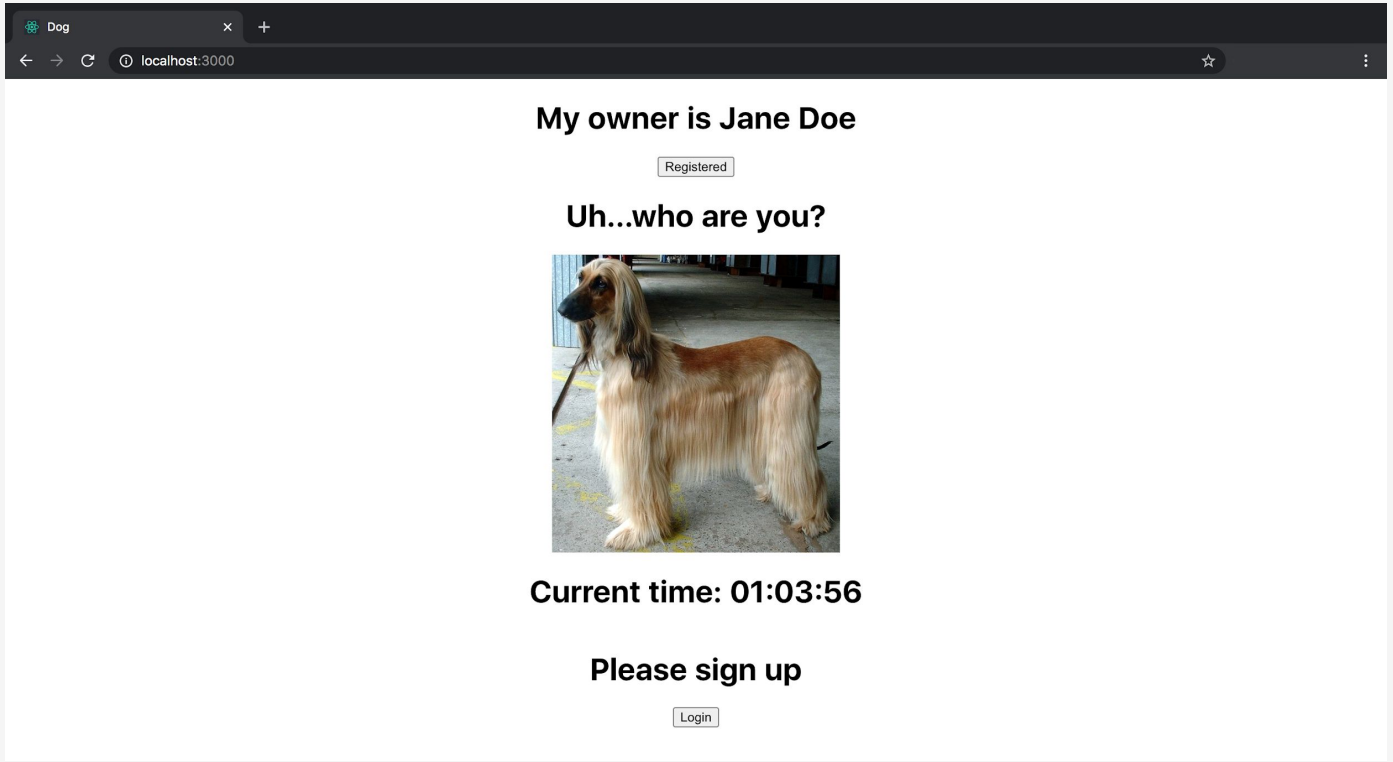
- Renders `Greeting` & either `LoginButton` or `LogoutButton` depending on its current state, i.e., `isLoggedIn`
- `React.Fragment` - group a list of children without adding extra node to the DOM

Conditional Rendering

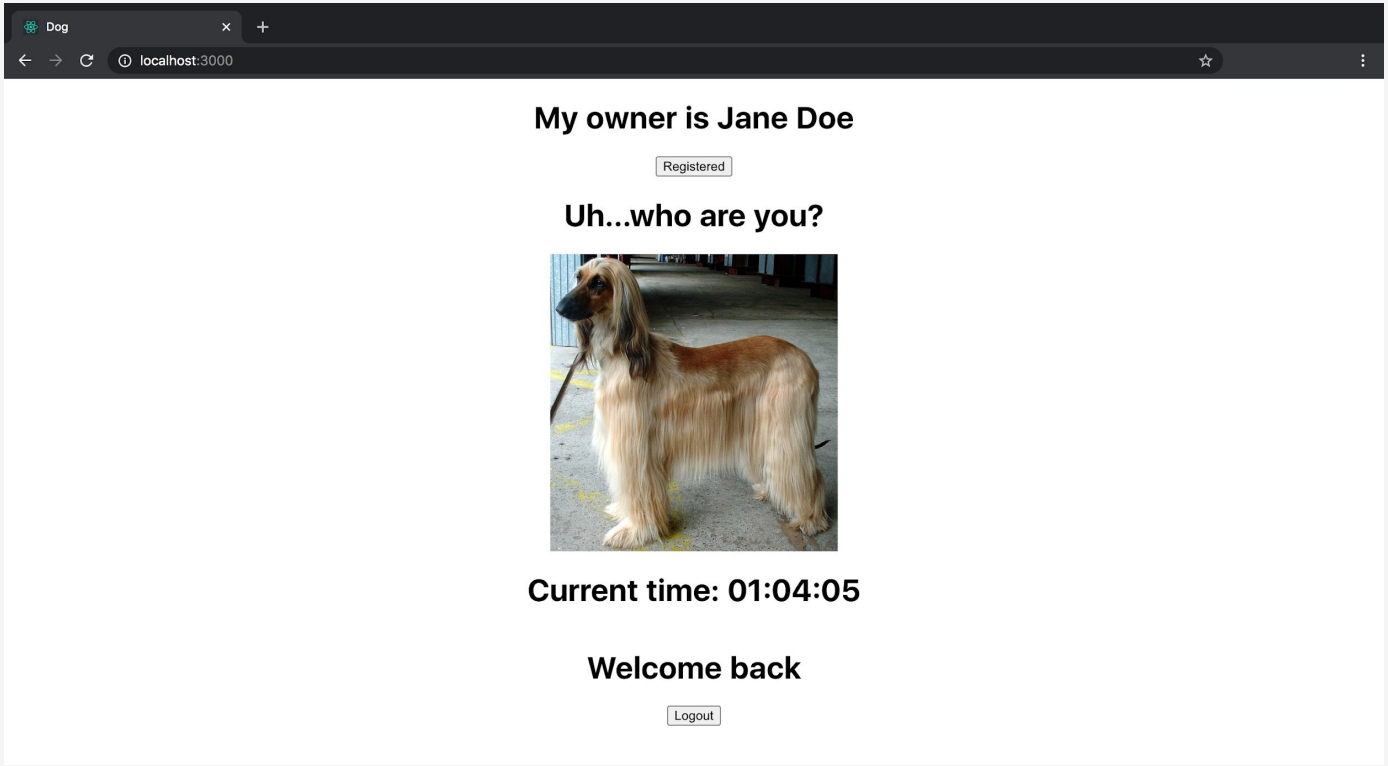
- In `App.js`
- `import LoginControl from './LoginControl'`
- Declare `<LoginControl />` in `App()`

```
function App() {
  return (
    <div className='container'>
      <Owner />
      <Register />
      {getGreeting()}
      <img src={dog.img} alt='afghan hound' width='300' />
      <Clock />
      <LoginControl />
    </div>
  )
}
```

Conditional Rendering



Conditional Rendering



Programming Activity

- Checkout to master - `git checkout master`
- Create a new branch called 19-practical - `git checkout -b 19-practical`
- Open the file `19-practical.pdf` and work on the tasks described