Using a Class Component with contextType

In this lesson we will learn how we can use contextType within class components.

WE'LL COVER THE FOLLOWING

- The Perfect Solution?

React 16.6 introduced the ability to consume data from context without using the `Consumer` component directly. This helps cut down on unnecessary nesting in your components' JSX, making them easier to read. To take advantage of `contextType` you're required to work with a class component.

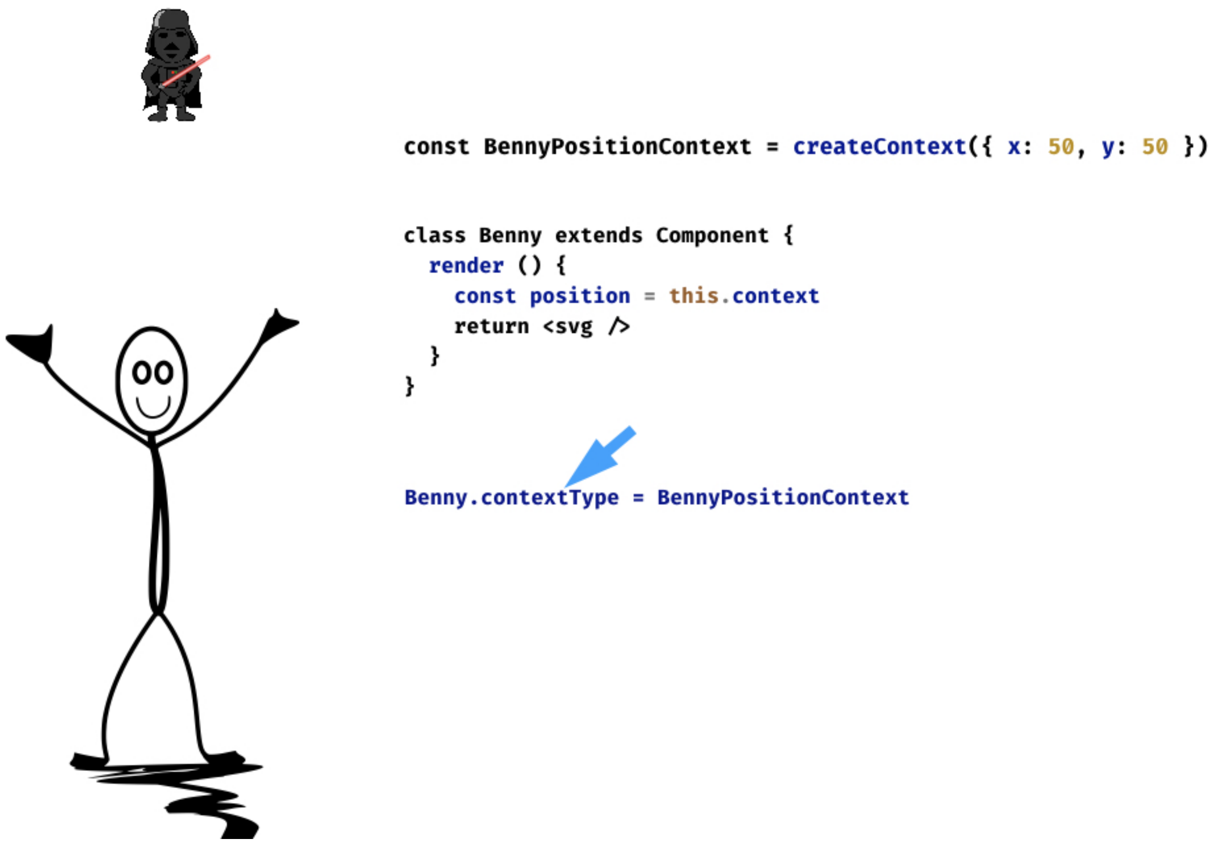
Consider the `Benny` component rewritten as a class component.

```
// create context object
const { Provider, Consumer } = createContext({ x: 50, y: 50 })
// Class component
class Benny extends Component {
  render () {
    return <Consumer>
    {position => <svg />}
  </Consumer>
  }
}
```

In this example, `Benny` consumes the initial context values `{ x: 50, y: 50 }` from the context object.

However, using a `Consumer` forces you to use a render prop API that may lead to nested code.

Let's get rid of the `Consumer` component by using the `contextType` class property.



```
const BennyPositionContext = createContext({ x: 50, y: 50 })

class Benny extends Component {
  render () {
    const position = this.context
    return <svg />
  }
}

Benny.contextType = BennyPositionContext
```

Getting this to work is fairly easy.

First, you set the `contextType` property of the class component to a context object.

```
const BennyPositionContext = createContext({ x: 50, y: 50 })
// Class Benny extends Component ...
// look here 👇
Benny.contextType = BennyPositionContext
```

After setting the `contextType` property, you can go ahead to consume values from the context object by using `this.context`.

For example, to retrieve the position values `{ x: 50, y: 50 }`:

```
class Benny extends Component {
  render () {
    // look here. No nesting!
    const position = this.context
    return <svg />
  }
}
```

```
}
```

The Perfect Solution?

Using the `contextType` class property is great, but not particularly the best solution in the world. You can only use one `contextType` within a class component. This means if you need to introduce multiple `Consumers`, you’ll still have some nested code.

Let’s conclude what we have learned so far in our next lesson.