Class Components Intro

[(https://github.com/learn-co-curriculum/react-hooks-class-components-intro)](https://github.com/learn-co-curriculum/react-hooks-class-components-intro) [(https://github.com/learn-co-curriculum/react-hooks-class-components-intro/issues/new)](https://github.com/learn-co-curriculum/react-hooks-class-components-intro/issues/new)

Learning Goals

- Translate function component syntax to class component syntax

Introduction

So far throughout this phase, we've been writing all our components as **function components**. Function components take in props, and return JSX. React also gives us another way of writing components using **classes** instead of functions.

For many years, the only way to use **state** and **lifecycle methods** in a React component was via **class components**. Since the introduction of React hooks in late 2018, that is no longer the case: function components can do everything classes can do! (Well, almost).

You are still very likely to encounter the class components when reading about React, and even are likely to see them on your first React job. So let's take a look at the syntax and learn some of the ins and outs of writing class components.

Class Component Syntax

When we're writing a function component, the most basic version (without any hooks) looks something like this:

```
function BlogPost(props) {
  return (
    <article>
      <h1>{props.title}</h1>
      <p>{props.content}</p>
    </article>
  );
}
```

Here's what the same component looks like using the class syntax:

```
import React from "react";

class BlogPost extends React.Component {
  render() {
    return (
      <article>
        <h1>{this.props.title}</h1>
        <p>{this.props.content}</p>
      </article>
    );
  }
}
```

Note that regardless of which syntax we use to define the component, we can use both versions of the component the same way:

```
ReactDOM.render(
  // doesn't matter if BlogPost is a class component or a function component!
  <BlogPost title="Hello" content="World" />,
  document.getElementById("root")
);
```

Here are some notes about this class syntax:

- `class BlogPost` : the **class declaration (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes)** gives us a template for creating JavaScript objects. In this case, the kind of object we're creating is a React component.
- `extends React.Component` : the `extends` keyword is JavaScript's way of providing inheritance to our class definitions. Our components **must** inherit from the `React.Component` class; this is where we'll get some additional behavior that we'll see later.

- `render()` : the `render` method is a special **lifecycle method** that must be defined on all of our class components. Just like with our function components, `render` is responsible for returning JSX!
- `this.props` : When React runs, any time we use `<BlogPost>` in a parent component, React will make a new **object** by calling constructor function `new BlogPost(props)` for us. All of the props passed down from the parent will be saved to that newly created object! So to access the props, we can use `this.props` inside of any method defined in our component. `props` alone won't work! We must use `this.props` .

```
// when we write this:
ReactDOM.render(
  <BlogPost title="Hello" content="World" />,
  document.getElementById("root")
);

// something like this happens in React's internal code
const component = new BlogPost({ title: "Hello", content: "World" });

// so inside the component, the props object is saved to the object:
class Component {
  constructor(props) {
    this.props = props;
  }
}
```

Conclusion

You don't have to worry about what's happening under the hood in React, but for working with class components, here are the key takeaways so far:

- use `class MyComponent extends React.Component` to define your class component
- write a `render()` method that returns JSX
- any time you need access to props, use `this.props`

Resources

- **MDN Classes (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes)**
- **React Component API (https://reactjs.org/docs/react-component.html)**