FUNCTION COMPONENTS

Stateless Functional Components

In the code editor, take a look at **Example.js**. The first `Example` component is defined as a JavaScript class, but it doesn't have to be! In React, we can also define components as JavaScript functions – we call them *function components* to differentiate them from *class components*.

In the latest versions of React, function components can do everything that class components can do. In most cases, however, function components offer a more elegant, concise way of creating React components. This lesson will focus on converting a class component to a function component and adding props, which are available in all versions of React.

Compare the `Example` class component and the `Example` function component. For the most basic function components, all you need to do is remove the beginning `render() {` and ending `}` of the `render()` method:

```
render()              { //              Delete              this
  return                                          <h1>Hello</h1>
} // Delete this
```

To put it in other words: the function component should return the same JSX that was originally returned by the `render()` method.

Instructions

1.

Select **Friend.js**.

Rewrite the `Friend` component class as a function component.

Use **Example.js** as a guide. Make sure to delete the original `Friend` class component when you're done.

Checkpoint 2 Passed

Hint

You'll be rewriting the `Friend` component in **Friend.js**. It will look something like this:

```
export const Friend = () => {
  // Return something here.
};
```

If you lose the image URL, here it is again:

```
https://content.codecademy.com/courses/React/react_photo-octopus.jpg
```

Friend.js

```
import React from 'react';
import ReactDOM from 'react-dom';

export const Friend = () => {
  return <img src="https://content.codecademy.com/courses/React/react_photo-octopus.jpg" />;
};

ReactDOM.render(
  <Friend />,
  document.getElementById('app')
);
```

Example.js

```
// A component class written in the usual way:
export class MyComponentClass extends React.Component {
  render() {
    return <h1>Hello world</h1>;
  }
}

// The same component class, written as a stateless functional component:
export const MyComponentClass = () => {
  return <h1>Hello world</h1>;
}

// Works the same either way:
ReactDOM.render(
  <MyComponentClass />,
```

```
  document.getElementById('app')
);
```

Function Components and Props

Like any component, function components can receive information via `props`.

To access these `props`, give your function component a parameter named `props`. Within the function body, you can access the props using this pattern: `props.propertyName`. You don't need to use the `this` keyword.

```
export function YesNoQuestion (props) {
  return (
    <div>
      <p>{props.prompt}</p>
      <input value="Yes" />
      <input value="No" />
    </div>
  );
}

ReactDOM.render(
  <YesNoQuestion prompt="Have you eaten an apple today?" />,
  document.getElementById('app');
);
```

In the above example, we pass a value of "Have you eaten an apple today?" as the `prompt` prop when rendering `YesNoQuestion`.

Instructions

1.

Open **NewFriend.js**.

Rewrite the `NewFriend` class component as a function component.

Make sure to delete the original `NewFriend` class when you're done.

Click Run and make sure that your new friend is still there!

Checkpoint 2 Passed

Hint

Make sure to include the `export` keyword.

Don't forget to define a `props` parameter in your new function and access the props using `props.src`.

Refer to **Example.js** for more examples.

NewFriend.js

```
import React from 'react';
import ReactDOM from 'react-dom';

export const NewFriend = (props) => {
    return (
      <div>
        <img src={props.src} />
      </div>
    )
}

ReactDOM.render(
  <NewFriend src="https://content.codecademy.com/courses/React/react_photo-squid.jpg" />,
  document.getElementById('app')
);
```

Example.js

```
// A component class written in the usual way:
export class MyComponentClass extends React.Component {
  render() {
    return <h1>Hello world</h1>;
  }
}

// The same component class, written as a stateless functional component:
```

```
export const MyComponentClass = () => {
  return <h1>Hello world</h1>;
}

// Works the same either way:
ReactDOM.render(
  <MyComponentClass />,
  document.getElementById('app')
);
```

Review

Well done! You've written your first function component. Here's a recap:

- *Function components* are React components defined as JavaScript functions
- Function components must return JSX
- Function components may accept a `props` parameter. Expect it to be a JavaScript object

Although function components and class components can do the same things, you'll see a lot of function components in the React documentation and example apps. Some developers prefer them over class components for their simplicity and straightforward features, like Hooks, which you'll learn later in your coding journey.

Instructions

Take a look at the example in the code editor. Make sure the code makes sense before moving on!