STYLE

Advanced React Techniques

In this unit, you will learn a variety of useful techniques that React programmers are expected to know.

You'll learn how to make a *propType*, how to write a form, and how to use styles.

You'll also be introduced to your second programming pattern: dividing components into *presentational components* and *container components*.

Click Next to begin!

Instructions

In this video, an object is passed from the Post component to the User info component. The object contains CSS-like values which affect the styling applied to User info.

Inline Styles

There are many different ways to use styles in React. This lesson is focused on one of them: *inline styles*.

An inline style is a style that's written as an *attribute*, like this:

```
<h1 style={{ color: 'red' }}>Hello world</h1>
```

Notice the double curly braces. What are those for?

The *outer* curly braces inject JavaScript into JSX. They say, "everything between us should be read as JavaScript, not JSX."

The *inner* curly braces create a JavaScript object literal. They make this a valid JavaScript object:

```
{ color: 'red' }
```

If you inject an object literal into JSX, and your entire injection is *only* that object literal, then you will end up

with double curly braces. There's nothing unusual about how they work, but they look funny and can be confusing.

Instructions

1.

Click Run to see the `<h1></h1>` rendered to the browser. How can you add styles to this poor `<h1></h1>`?

Checkpoint 2 Passed

Hint

No need to edit any code here. Just click Run and see what happens!

2.

Let's give the `<h1></h1>` an *inline style*.

Give the `<h1></h1>` an attribute with a *name* of `style`. The attribute's *value* should evaluate to this object:

```
{ background: 'lightblue', color: 'darkred' }
```

Checkpoint 3 Passed

Hint

Your `<h1>` will have a new attribute, called `style`, with a value of the object in the instructions. The code will look like this:

```
<h1 style={{ background: 'lightblue', color: 'darkred' }}>
```

styleMe.js

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

const styleMe = <h1 style={{ background: 'lightblue', color:
 'darkred' }}>Please style me! I am so bland!</h1>;

ReactDOM.render(
  styleMe,
  document.getElementById('app')
);
```

Make A Style Object Variable

That's all that you need to apply basic styles in React! Simple and straightforward.

One problem with this approach is that it becomes obnoxious if you want to use more than just a few styles. An alternative that's often nicer is to store a style object in a *variable*, and then inject that variable into JSX.

Look in the code editor for an example. The style object is *defined* on lines 3-6, and then *injected* on line 11.

If you aren't used to using modules, then this code may have made you twitch uncontrollably:

```
const style = {
  color: 'darkcyan',
  background: 'mintcream'
};
```

Defining a variable named `style` in the top-level scope would be an extremely bad idea in many JavaScript environments! In React, however, it's totally fine.

Remember that every file is invisible to every other file, except for what you choose to expose via `module.exports`. You could have 100 different files, all with global variables named `style`, and there could be no conflicts.

Instructions

1.

Select **styleMe.js**. Make a new line after `import ReactDOM from 'react-dom';`.

On this new line, declare a new constant named `styles`. Set `styles` equal to this object:

```
{
  background: 'lightblue',
  color: 'darkred'
}
```

Checkpoint 2 Passed

Hint

If you wanted to declare a constant named `styles` with different colors, you might do something like this:

```
const styles = {
  background: 'orange',
  color: 'purple'
};
```

Your code will look similar but will use different colors. Refer to **Example.js** for another example.

2.

Change the *value* of your `<h1></h1>`'s `style` attribute. Make `style`'s value equal to your new `styles` variable.

Since you aren't injecting an object *literal* anymore, you will no longer need to use double curly braces.

Checkpoint 3 Passed

Hint

Set the `<h1>`'s `style` attribute to `styles` with `style={styles}`.

Example.js

```
import React from 'react';

const styles = {
  color: 'darkcyan',
  background: 'mintcream'
};

export class StyledClass extends React.Component {
  render() {
    return (
      <h1 style={styles}>
        Hello world
      </h1>
    );
  }
}
```

styleMe.js

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

const styles = {
  background: 'lightblue',
  color: 'darkred'
}

const styleMe = <h1 style={styles}>Please style me! I am so
bland!</h1>;

ReactDOM.render(
  styleMe,
  document.getElementById('app')
);
```

Style Name Syntax

In regular JavaScript, style *names* are written in hyphenated-lowercase:

```
const styles = {
  'margin-top': '20px',
  'background-color': 'green'
};
```

In React, those same names are instead written in camelCase:

```
const styles = {
  marginTop: '20px',
  backgroundColor: 'green'
};
```

This has zero effect on style property *values*, only on style property *names*.

Instructions

1.

Give your `styles` object two more properties:
a `marginTop` of `100px`, and a `fontSize` of `50px`.

Checkpoint 2 Passed

Hint

Your `styles` object currently has two properties: `background` and `color`. You'll add two more: `marginTop` and `fontSize`.

styleMe.js

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

const styles = {
  background: 'lightblue',
  color: 'darkred',
  marginTop: '100px',
  fontSize: '50px'
}

const styleMe = <h1 style={styles}>Please style me! I am so bland!</h1>;

ReactDOM.render(
  styleMe,
  document.getElementById('app')
);
```

Style Value Syntax

In the last exercise, you learned how style *names* are slightly different in React than they are in regular JavaScript.

In this exercise, you will learn how style *values* are slightly different in React than they are in regular JavaScript.

In regular JS, style *values* are almost always strings. Even if a style value is numeric, you usually have to write it as

a string so that you can specify a unit. For example, you have to write `"450px"` or `"20%"`.

In React, if you write a style value as a *number*, then the unit `"px"` is assumed.

How convenient! If you want a font size of 30px, you can write:

```
{ fontSize: 30 }
```

If you want to use units other than “px,” you can use a string:

```
{ fontSize: "2em" }
```

Specifying “px” with a string will still work, although it’s redundant.

A few specific styles will *not* automatically fill in the “px” for you. These are styles where you aren’t likely to use “px” anyway, so you don’t really have to worry about it. Here is a list of styles that don’t assume “px”.

Instructions

1.

In your `styles` object, change any property values that end in “px” from strings into numbers.

Checkpoint 2 Passed

Hint

You’ll change `marginTop`‘s `'100px'` to `100`, and do something very similar for `fontSize`.

styleMe.js

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

const styles = {
  background: 'lightblue',
  color: 'darkred',
  marginTop: 100,
  fontSize: 50
```

```
}

const styleMe = <h1 style={styles}>Please style me! I am so
bland!</h1>;

ReactDOM.render(
  styleMe,
  document.getElementById('app')
);
```

Share Styles Across Multiple Components

What if you want to reuse styles for several different components?

One way to make styles *reusable* is to keep them in a separate JavaScript file. This file should *export* the styles that you want to reuse, via `export`. You can then `import` your styles into any component that wants them.

In the code editor, move back and forth between **facebookStyles.js** and **FacebookColorThief.js** to see a styles file in action.

Instructions

1.

You have a meeting in 5 minutes with an important client. It is *crucial* that you impress them by showing off a beautiful React website.

You haven't used any styles at all yet! Oh, no!!!

You *do* have some styles that you wrote one night while lost in a deep dream. Open **styles.js** to see your subconscious styles.

Intriguing! But your dream-styles aren't connected to the rest of your app, so at the moment they're worthless. Can you apply your styles before it's too late?

Click Run to see the current state of things. Maybe you don’t need to apply your dream-styles! Maybe your site is impressive enough already!

Checkpoint 2 Passed

Hint

Click Run to see the app in action. No need to edit any code yet!

2.

No, definitely not impressive enough.

You need to export your styles from **styles.js**, so that you can import them into your other files.

In **styles.js**, export your styles:

```
export const styles = {
  fontFamily: fontFamily,
  background: background,
  fontSize: fontSize,
  padding: padding,
  color: color
};
```

Checkpoint 3 Passed

Hint

Carefully copy the code snippet at the bottom of **styles.js**.

3.

These styles are all you’ve got! You’re going to have to use them in both **Home.js** and **AttentionGrabber.js**.

In **Home.js**, create a new line after `import { AttentionGrabber } from './AttentionGrabber';`. On your new line, `import styles` from **styles.js**.

styles.js, **Home.js**, and **AttentionGrabber.js** all share the same parent directory.

Checkpoint 4 Passed

Hint

In **Home.js**, add a new line after `import { AttentionGrabber } from './AttentionGrabber';`. On your new line, `import { styles } from './styles.js';`.

4.

Now select **AttentionGrabber.js**.

Create a new line after `import React from 'react';`. On your new line, import your exported `styles` variable again.

Checkpoint 5 Passed

Hint

In **AttentionGrabber.js**, add a new line after `import React from 'react';`. On your new line, `import { styles } from './styles.js';`—same as in the previous step.

5.

Now you can use your dream-styles!

Select **Home.js**. In between the `styles` import and the `Home` component declaration, define a new variable:

```
const divStyle = {
  background: styles.background,
  height: '100%'
};
```

Now select **AttentionGrabber.js**. In between the `styles` import and the `AttentionGrabber` component declaration, define a different new variable:

```
const h1Style = {
  color: styles.color,
  fontSize: styles.fontSize,
  fontFamily: styles.fontFamily,
  padding: styles.padding,
  margin: 0,
};
```

Checkpoint 6 Passed

Hint

In **Home.js**, between the `styles` import and the `Home` component declaration, define a new variable like this:

```
const divStyle = {
  background: styles.background,
  height: '100%'
};
```

In **AttentionGrabber.js**, we'll do something similar. Between the `styles` import and the `AttentionGrabber` component declaration, define another variable:

```
const h1Style = {
  color: styles.color,
  fontSize: styles.fontSize,
  fontFamily: styles.fontFamily,
  padding: styles.padding,
  margin: 0,
};
```

6.

The client is walking into the conference room! Quick, use those styles!

In `Home`'s render function, give the `<div>` a `style` attribute of `{divStyle}`.

Checkpoint 7 Passed

Hint

In **Home.js**, make sure that the `<div>` has an attribubte of `style={divStyle}`.

7.

In `AttentionGrabber`'s render function, give the `<h1>` a `style` attribute of `{h1Style}`.

Click Run and expand the browser.

The client falls to pieces over your minimalist, elegant design! Raises for everyone!

Checkpoint 8 Passed

Hint

In **AttentionGrabber.js**, make sure that the `<h1>` has an attribute of `style={h1Style}`.

facebookStyles.js

```
// facebook color palette

const blue = 'rgb(139, 157, 195)';
const darkBlue = 'rgb(059, 089, 152)';
const lightBlue = 'rgb(223, 227, 238)';
const grey = 'rgb(247, 247, 247)';
const white = 'rgb(255, 255, 255)';

const colorStyles = {
  blue: blue,
  darkBlue: darkBlue,
  lightBlue: lightBlue,
  grey: grey,
  white: white
};
```

FacebookColorThief.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { colorStyles } from './facebookStyles';

let divStyle = {
  backgroundColor: styles.darkBlue,
  color: styles.white
};

export class Wow extends React.Component {
  render() {
    return (
      <div style={divStyle}>
        Wow, I stole these colors from Facebook!
      </div>
    );
  }
}

ReactDOM.render(
  <Wow />,
```

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

styles.js

```
const fontFamily = 'Comic Sans MS, Lucida Handwriting, cursi
ve';
const background = 'pink url("https://content.codecademy.com
/programs/react/images/welcome-to-my-homepage.gif") fixed';
const fontSize = '4em';
const padding = '45px 0';
const color = 'green';

export const styles = {
  fontFamily: fontFamily,
  background: background,
  fontSize: fontSize,
  padding: padding,
  color: color
};
```

Home.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { AttentionGrabber } from './AttentionGrabber';
import { styles } from './styles.js';

const divStyle = {
  background: styles.background,
  height: '100%'
};

export class Home extends React.Component {
  render() {
    return (
      <div style={divStyle}>
        <AttentionGrabber />
        <footer>THANK YOU FOR VISITING MY HOMEPAGE!</footer>
      </div>
```

```
    );
  }
}

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

AttentionGraber.js

```
import React from 'react';
import { styles } from './styles.js';

const h1Style = {
  color: styles.color,
  fontSize: styles.fontSize,
  fontFamily: styles.fontFamily,
  padding: styles.padding,
  margin: 0,
};

export class AttentionGrabber extends React.Component {
  render() {
    return <h1 style={h1Style}>WELCOME TO MY HOMEPAGE!</h1>;
  }
}
```