PROJECT

Copycat

In this project, we'll build a program that lets users type into a textbox and allows them to visualize the immediate effect that these changes have on the web page.

Our program will display a textbox with a picture of a cat beneath it. When users type into the textbox, a copy of the text will appear below the cat image, suggesting that the cat is being a copycat. Clicking on the image of the cat will toggle whether or not it is able to copy the user.

Let's get started!

If you get stuck during this project or would like to see an experienced developer work through it, click **"Get Unstuck"** to see a **project walkthrough video**.

Tasks

12/12 Complete

Mark the tasks as complete by checking them off

Split into Presentational and Container Components
1.

Take a look at the code in **components/CopyCat.js**. We're going to divide `CopyCat` into a *presentational* component and a *container* component.

Inside of the **containers** folder, there is a *file* named **CopyCatContainer.js**. Copy all of the contents from **components/CopyCat.js** and paste them into **containers/CopyCatContainer.js**.

If you click Save, you'll see the component in **CopyCatContainer.js** render in the browser!
Hint

In this programming pattern, the container component does the work of figuring out what to display. The presentational component does the work of actually displaying it. If a component does a significant amount of work in both areas, then that's a sign that you should use this pattern!

2.

Select **components/CopyCat.js**. This is going to be our *presentational* component class.

On line 2, delete the line `import ReactDOM from 'react-dom';`. At the bottom of the file, delete the `ReactDOM.render()` call.

Inside of `CopyCat`'s class definition, delete everything except for the render function.

Finally, export `CopyCat`.

Hint

Remember, a presentational component's only job is to render HTML-like JSX.

3.

Select **containers/CopyCatContainer.js**. This is going to be our *container* component class.

Import `CopyCat` at the top of the file.

Now, down where the class is being defined, change `CopyCat` to `CopyCatContainer`. Be sure to update this in the `ReactDOM.render` call at the bottom of the file as well.

Inside of `CopyCatContainer`'s render function, delete everything inside of the `return` statement. Instead, just return an instance of `CopyCat`.

Hint

This component is supposed to be the opposite of `CopyCat`. It holds the logic for how things work. The only thing its `render()` method should return is an instance of `CopyCat`.

4.

Take a close look at `CopyCat`. You'll see that the component needs access to the `copying` state as well as the `toggleTape` method that now only exists in `CopyCatContainer`.

Inside of **containers/CopyCatContainer.js**, pass `copying` and `toggleTape` as props to `<CopyCat/>`.

Now switch back to `CopyCat` in **components/CopyCat.js**. Make sure the `render` method is grabbing `copying` and `toggleTape` from the props.

Then click Save. If everything was done correctly, the app should look the exact same in the browser as it did in step 1. Don't worry that it doesn't look pretty. We'll spruce it up once we add some style!

Hint

Since the state is not stored anywhere in the presentational component, it needs to be passed down as props from the container component. Instead of using `this.state` to access what we need, we'll use `this.props` instead.

Add Styles

5.

Select **styles.js**.

In this file, you'll see a number of style properties defined. Underneath these, there are two objects, each of which contain a selection of these properties: `divStyles` and `imgStyles`.

At the bottom of this file, create a const `styles`. Set its value to be an object that holds `divStyles` and `imgStyles`. Export this `styles` object.

Hint

In React, you can export JavaScript objects containing the styles you want for your components. Take a look at how the styles are written in **styles.js**. Here are a few things to take note of:

- In React, style names are written in camelCase, not hyphenated as is the case in CSS.
- Style values are almost always strings.
- When number values are used for style properties, React automatically adds "px" to make them pixel values.

6.

Go back to **CopyCat.js** and import `styles`.

In the JSX in the `render()` method, set the `<div>` element's style to `divStyles`. Use `imgStyles` for the `<img/>` element.

Give `<h1>` a single style property: `marginBottom: 80`.
Hint

React supports inline CSS styles for components. Styles are supplied as a `style` prop to components. The `style` prop should be an object with `StylePropertyName: StyleValue` values.
Add a Form
7.

In **CopyCat.js**, create an `<input>` element between the `<h1>` and `<img>` elements. Give it three attributes: `type`, `value`, `onChange`. Set `type` to be `'text'`.

The values of `value` and `onChange` will be acquired from the props, but we currently aren't passing anything down for those. For now, use empty braces as their values.
Hint

You don't even need to use a `<form>` element! Your "form" was actually just an `<input />`.
8.

Navigate to **CopyCatContainer.js**.

Add `input` to the state and set its initial value to an empty string.

Next, write an event handler function called `handleChange` which takes the event `e` as an argument. The function should update the state with `<input>`'s value whenever it changes.

Don't forget to bind `handleChange` in the constructor method!
Hint

When a user types or deletes in the input field, then that will trigger a change event. `handleChange` should set `this.state.input` equal to whatever text is currently in the input field.
9.

Next, pass `input` and `handleChange` as props to `<CopyCat>`.

Navigate back to `CopyCat.js` and update the values for `<input>`'s `value` and `onChange` attributes.
Hint

Remember, props are passed from one component to another by defining them as attributes on a component instance.

10.

Still working in **CopyCat.js**, add a `<p>` element after the `<img>` element in the JSX.

Inside of the `<p></p>` tags, write a conditional to check the value of `copying` and decide whether or not to display `value` here. If `copying` evaluates to `true`, `value` should show up. If `copying` evaluates to `false`, `value` should NOT show up.

When you're done, save your code. Type into the text box and see if anything happens. Click on the cat to toggle whether or not it's copying you!

Hint

This would be a good place for the && operator…

Typecheck with PropTypes

11.

Let's give our `CopyCat` component class a propTypes property!

The first thing we'll need to do is import `PropTypes` at the top of the file.

Next, declare a `propTypes` property after the close of the component declaration. In the `propTypes` object, write one `propType` for each `prop` that `CopyCat` is expecting.

Make sure each `propType` has an `isRequired` constraint.

Hint

The name of each property in propTypes should be the name of an expected prop. The value of each property in propTypes should fit this pattern: `PropTypes.expected-data-type-goes-here`

12.

Finally, let's add an optional `name` prop to define the cat's name.

Inside of the `<h1></h1>` tags, add a condition immediately after "Copy Cat". If a name is passed down in `props`, that name will be displayed. If not, the name that will be displayed will default to "Tom".

Add a `propType` for `name` to `propTypes`.
Hint

`propTypes` don't need the `.isRequired` property if the prop is considered optional by the component.

CopyCat.js

```
import React from 'react';
import { styles } from '../styles';
import PropTypes from 'prop-types';

const images = {
  copycat: 'https://content.codecademy.com/courses/React/react_photo_copycat.png',
  quietcat: 'https://content.codecademy.com/courses/React/react_photo_quietcat.png'
};

export class CopyCat extends React.Component {
  render() {
    const { copying, value, toggleTape, handleChange, name }
 = this.props;

    return (
      <div style={styles.divStyles}>
        <h1 style= {{marginBottom: 80}}>
        Copy Cat {name || 'Tom'}</h1>
        <input
            type='text'
            value={value}
            onChange={handleChange}
            />
        <img
          style={styles.imgStyles}
          alt='cat'
          src={copying ? images.copycat : images.quietcat}
          onClick={toggleTape}
```

```
        />
        <p>
        {copying && value}
        </p>
      </div>
    );
  };
}

CopyCat.propTypes = {
  copying: PropTypes.bool.isRequired,
  toggleTape: PropTypes.func.isRequired,
  value: PropTypes.string.isRequired,
  handleChange: PropTypes.func.isRequired,
  name: PropTypes.string
}
```

CopyCatContainer.js

```
import React from 'react';
import { styles } from '../styles';
import PropTypes from 'prop-types';

const images = {
  copycat: 'https://content.codecademy.com/courses/React/rea
ct_photo_copycat.png',
  quietcat: 'https://content.codecademy.com/courses/React/re
act_photo_quietcat.png'
};

export class CopyCat extends React.Component {
  render() {
    const { copying, value, toggleTape, handleChange, name }
 = this.props;

    return (
      <div style={styles.divStyles}>
        <h1 style= {{marginBottom: 80}}>
        Copy Cat {name || 'Tom'}</h1>
        <input
```

```
            type='text'
            value={value}
            onChange={handleChange}
            />
        <img
          style={styles.imgStyles}
          alt='cat'
          src={copying ? images.copycat : images.quietcat}
          onClick={toggleTape}
        />
        <p>
        {copying && value}
        </p>
      </div>
    );
  };
}

CopyCat.propTypes = {
  copying: PropTypes.bool.isRequired,
  toggleTape: PropTypes.func.isRequired,
  value: PropTypes.string.isRequired,
  handleChange: PropTypes.func.isRequired,
  name: PropTypes.string
}
```

styles.js

```
const fontFamily = 'Comic Sans MS, Lucida Handwriting, cursi
ve';
const fontSize = '5vh';
const backgroundColor = '#282c34';
const minHeight = '100vh';
const minWidth = 400;
const display = 'flex';
const flexDirection = 'column';
const alignItems = 'center';
const justifyContent = 'center';
const color = 'white';
const marginTop = '20px';
```

```
const width = '50%';

const divStyles = {
  fontFamily: fontFamily,
  fontSize: fontSize,
  color: color,
  backgroundColor: backgroundColor,
  minHeight: minHeight,
  minWidth: minWidth,
  display: display,
  flexDirection: flexDirection,
  alignItems: alignItems,
  justifyContent: justifyContent,
};

const imgStyles = {
  marginTop: marginTop,
  width: width
};

export const styles = {
  divStyles: divStyles,
  imgStyles: imgStyles
}
```