PROPTYPES

propTypes

In this lesson, you will learn to use an important React feature called `propTypes`.

`propTypes` are useful for two reasons. The first reason is *prop validation*.

Validation can ensure that your `props` are doing what they're supposed to be doing. If `props` are missing, or if they're present but they aren't what you're expecting, then a warning will print in the console.

This is useful, but reason #2 is arguably more useful: *documentation*.

Documenting `props` makes it easier to glance at a file and quickly understand the component class inside. When you have a lot of files, and you will, this can be a huge benefit.

Click Next to learn how to use `propTypes`!

Instructions

In this video, the Stats component has a `propTypes` object that requires its `comments` prop to be a number.

When a string is passed to Stats, an error is raised. When a number is passed to Stats, the prop is accepted and the component updates with the new value.

Apply PropTypes

In the code editor, take a look at `MessageDisplayer`'s render function.

Notice the expression `this.props.message`. From this expression, you can deduce that `MessageDisplayer` expects to get passed a `prop` named `message`. Somewhere, at some time, this code is expected to execute:

```
<MessageDisplayer message="something" />
```

If a component class expects a `prop`, then you can use `propTypes` for that component class!

In order to start using `propTypes`, we need to import the `'prop-types'` library.

```
import PropTypes from 'prop-types';
```

Then, you can declare `propTypes` as a static property for your component after the component has been defined. See the example of a `propTypes` property on lines 11-13. Notice that the *value* of `propTypes` is an object, not a function!

The second step is to add properties to the `propTypes` object. For each `prop` that your component class expects to receive, there can be one property on your `propTypes` object.

`MessageDisplayer` only expects one `prop`: `message`. Therefore, its `propTypes` object only has one property.

Instructions

1.

Select **BestSeller.js**.

Import the `'prop-types'` library as `PropTypes` on line 2.

Checkpoint 2 Passed

Hint

Take a look at how the `'prop-types'` library was imported on line 2 of **MessageDisplayer.js**. Make sure `'prop-types'` is a string and that the library is imported as `PropTypes`.

2.

Give the `BestSeller` component class a `propTypes` property. For now, set `propTypes` equal to an empty object literal.

Checkpoint 3 Passed

Hint

Check out **MessageDisplayer.js** to see how `propTypes` is attached to the `MessageDisplayer` component.

MessageDisplayer.js

```
import React from 'react';
import PropTypes from 'prop-types';

export class MessageDisplayer extends React.Component {
  render() {
    return <h1>{this.props.message}</h1>;
  }
}

// This propTypes object should have
// one property for each expected prop:
MessageDisplayer.propTypes = {
  message: PropTypes.string
};
```

BestSeller.js

```
import React from 'react';
import PropTypes from 'prop-types';


export class BestSeller extends React.Component {
  render() {
    return (
      <li>
        Title: <span>
          {this.props.title}
        </span><br />

        Author: <span>
          {this.props.author}
        </span><br />

        Weeks: <span>
          {this.props.weeksOnList}
        </span>
      </li>
    );
  }
}
```

```
BestSeller.propTypes = {

};
```

Add Properties to PropTypes

In the code editor, look at the property on `MessageDisplayer`'s `propTypes` object:

```
message: PropTypes.string
```

What are the properties on `propTypes` supposed to *be*, exactly?

The *name* of each property in `propTypes` should be the name of an expected `prop`. In our case, `MessageDisplayer` expects a `prop` named `message`, so our property's *name* is `message`.

The *value* of each property in `propTypes` should fit this pattern:

```
PropTypes.expected_data_type_goes_here
```

Since `message` is presumably going to be a string, we chose `PropTypes.string`. You can see this on line 12. Notice the difference in capitalization between the `propTypes` object and `PropTypes`!

Each property on the `propTypes` object is called a `propType`.

Select the next file in the code editor, **Runner.js**. Find `Runner`'s `propTypes` object.

`Runner` has six `propTypes`! Look at each one. Note that `bool` and `func` are abbreviated, but all other data types are spelled normally.

If you add `.isRequired` to a `propType`, then you will get a console warning if that `prop` *isn't* sent.

Try to find all six `props` from the `propTypes` object in `Runner`'s `render` function: `this.props.message`, `this.props.style`, etc.

Instructions

1.

Select **BestSeller.js**.

In `BestSeller`'s `propTypes` object, write one `propType` for each `prop` that `BestSeller` is expecting: `title`, `author`, and `weeksOnList`.

Make `title` and `author` strings. Make `weeksOnList` a number. Make all three `isRequired`.

If you get stuck, look to **Runner.js** for guidance.

Checkpoint 2 Passed

Hint

Look at **Runner.js** for an example of how to do something similar.

2.

Good! You just gave `BestSeller` three `propTypes`.

In the code editor, open the last file, **BookList.js**.

At the bottom of the file, render `<BookList />` using `ReactDOM.render`.

Checkpoint 3 Passed

Hint

Call `ReactDOM.render()` with two arguments: `<BookList />` and `document.getElementById('app')`.

MessageDisplayer

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

export class BookList extends React.Component {
  render() {
```

```
    return (
      <div>
        <h1>Best Sellers</h1>
        <div>
          <ol>
            <BestSeller
              title="Glory and War Stuff for Dads"
              author="Sir Eldrich Van Hoorsgaard"
              weeksOnList={10} />
            <BestSeller
              title="The Crime Criminals!"
              author="Brenda Sqrentun"
              weeksOnList={2} />
            <BestSeller
              title="Subprime Lending For Punk Rockers"
              author="Malcolm McLaren"
              weeksOnList={600} />
          </ol>
        </div>
      </div>
    );
  }
}

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

Runner.js

```
import React from 'react';
import PropTypes from 'prop-types';

export class Runner extends React.Component {
  render() {
    let miles = this.props.miles;
    let km = this.props.milesToKM(miles);
    let races = this.props.races.map(function(race, i){
      return <li key={race + i}>{race}</li>;
    });
```

```
    return (
      <div style={this.props.style}>
        <h1>{this.props.message}</h1>
        { this.props.isMetric &&
          <h2>One Time I Ran {km} Kilometers!</h2> }
        { !this.props.isMetric &&
          <h2>One Time I Ran {miles} Miles!</h2> }
        <h3>Races I've Run</h3>
        <ul id="races">{races}</ul>
      </div>
    );
  }
}

Runner.propTypes = {
  message:   PropTypes.string.isRequired,
  style:     PropTypes.object.isRequired,
  isMetric:  PropTypes.bool.isRequired,
  miles:     PropTypes.number.isRequired,
  milesToKM: PropTypes.func.isRequired,
  races:     PropTypes.array.isRequired
};
```

BestSeller.js

```
import React from 'react';
import PropTypes from 'prop-types';

export class BestSeller extends React.Component {
  render() {
    return (
      <li>
        Title: <span>
          {this.props.title}
        </span><br />

        Author: <span>
          {this.props.author}
        </span><br />
```

```
        Weeks: <span>
          {this.props.weeksOnList}
        </span>
      </li>
    );
  }
}

BestSeller.propTypes = {
  title: PropTypes.string.isRequired,
  author: PropTypes.string.isRequired,
  weeksOnList: PropTypes.number.isRequired
};
```

BookList.js

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

export class BookList extends React.Component {
  render() {
    return (
      <div>
        <h1>Best Sellers</h1>
        <div>
          <ol>
            <BestSeller
              title="Glory and War Stuff for Dads"
              author="Sir Eldrich Van Hoorsgaard"
              weeksOnList={10} />
            <BestSeller
              title="The Crime Criminals!"
              author="Brenda Sqrentun"
              weeksOnList={2} />
            <BestSeller
              title="Subprime Lending For Punk Rockers"
              author="Malcolm McLaren"
              weeksOnList={600} />
          </ol>
```

```
        </div>
      </div>
    );
  }
}

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

PropTypes in Function Components

Remember *function components?* You can see some familiar ones in **Example.js.**

How could you write `propTypes` for a function component?

```
// Usual way:
class Example extends React.component{

}

Example.propTypes = {

};
...

// Function component way:
const Example = (props) => {
  // ummm ??????
}
```

It turns out the process is fairly similar. To write `propTypes` for a function component, you define a `propTypes` object as a property of *the function component itself.* Here's what that looks like:

```
const Example = (props) => {
  return <h1>{props.message}</h1>;
}

Example.propTypes = {
  message: PropTypes.string.isRequired
};
```

Instructions

1.

Select **GuineaPigs.js**.

You can see your GuineaPigs *function component* from earlier. Let's give it a propType.

After the GuineaPigs class declaration, define a propTypes property on GuineaPigs. Use the example code above as a guide.

GuineaPigs is only expecting one prop, so it should only get one propType.

Give GuineaPigs one propType, matching its expected prop. Make the propType isRequired.

If you aren't sure what prop GuineaPigs is expecting, check the render function in **GuineaPigsContainer.js**.

Checkpoint 2 Passed

Hint

In previous exercises, you've added propTypes to components to specify which props are required. GuineaPig is no different, even though it is a functional component.

You can refer to the sample code above as a guide.

Example.js

```
// Normal way to display a prop:
export class MyComponentClass extends React.Component {
  render() {
    return <h1>{this.props.title}</h1>;
  }
}

// Functional component way to display a prop:
export const MyComponentClass = (props) => {
  return <h1>{props.title}</h1>;
}
```

```
// Normal way to display a prop using a variable:
export class MyComponentClass extends React.component {
  render() {
    let title = this.props.title;
    return <h1>{title}</h1>;
  }
}

// Functional component way to display a prop using a variab
le:
export const MyComponentClass = (props) => {
  let title = props.title;
  return <h1>{title}</h1>;
}
```

GuineaPigs.js

```
import React from 'react';
import PropTypes from 'prop-types';

export const GuineaPigs = (props) => {
  let src = props.src;

  return (
    <div>
      <h1>Cute Guinea Pigs</h1>
      <img src={src} />
    </div>
  );
}

GuineaPigs.propTypes = {
    src: PropTypes.string.isRequired
};
```

GuineaPigsContainer.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { GuineaPigs } from '../components/GuineaPigs';
```

```
const GUINEAPATHS = [
  'https://content.codecademy.com/courses/React/react_photo-guineapig-1.jpg',
  'https://content.codecademy.com/courses/React/react_photo-guineapig-2.jpg',
  'https://content.codecademy.com/courses/React/react_photo-guineapig-3.jpg',
  'https://content.codecademy.com/courses/React/react_photo-guineapig-4.jpg'
];

export class GuineaPigsContainer extends React.Component {
  constructor(props) {
    super(props);

    this.state = { currentGP: 0 };
  }

  nextGP() {
    let current = this.state.currentGP;
    let next = ++current % GUINEAPATHS.length;
    this.setState({ currentGP: next });
  }

  componentDidMount() {
    this.interval = setInterval(this.nextGP, 5000);
  }

  componentWillUnmount() {
    clearInterval(this.interval);
  }

  render() {
    let src = GUINEAPATHS[this.state.currentGP];
    return <GuineaPigs src={src} />;
  }
});

ReactDOM.render(
```

```
  <GuineaPigsContainer />,
  document.getElementById('app')
);
```