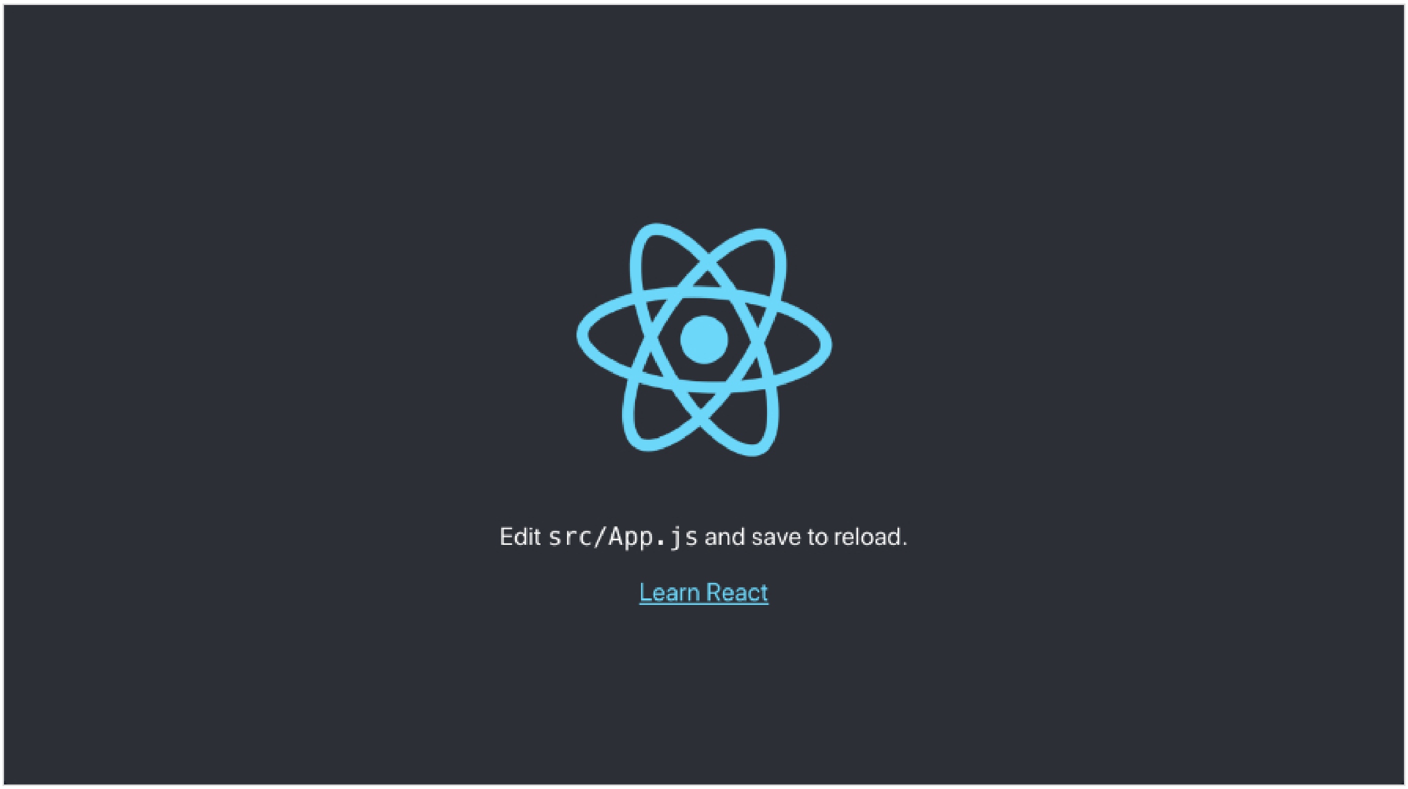
React

Project Setup

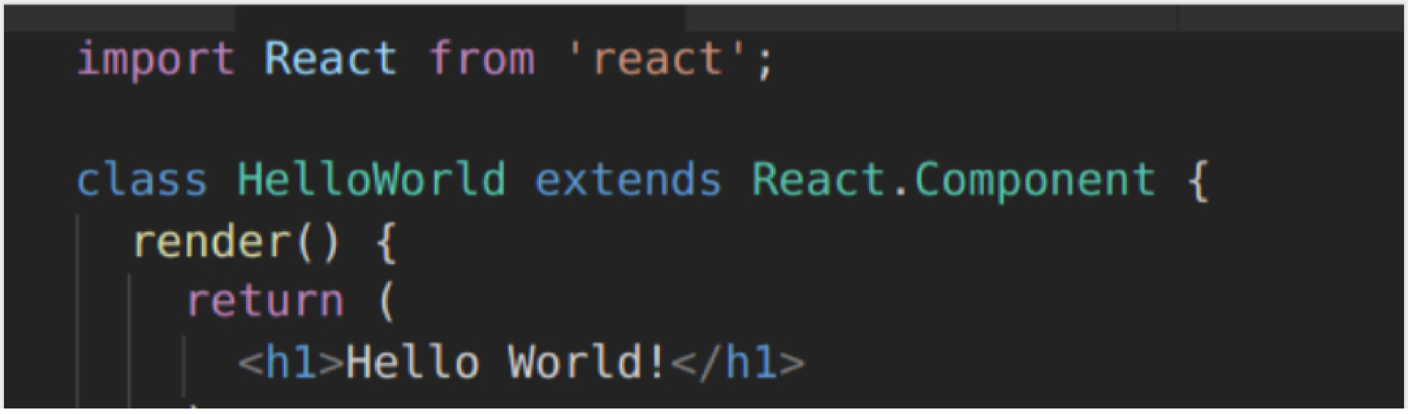


Edit `src/App.js` and save to reload.

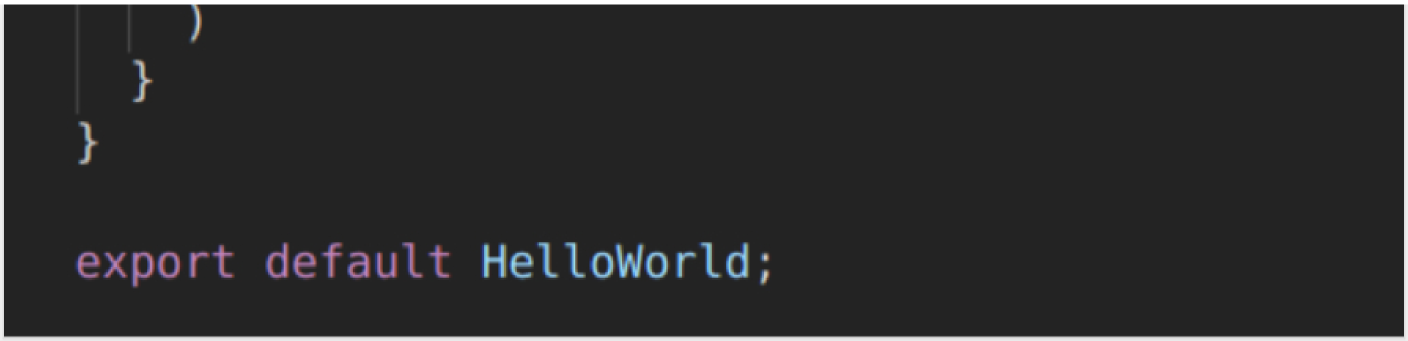
Learn React

By running the command 'npx create-react-app app-name' and entire application is created using React and by default presents you with this screen. There is only one link on this page that leads to the React website which has all the information you need in order to get started with React.

Hello World Component



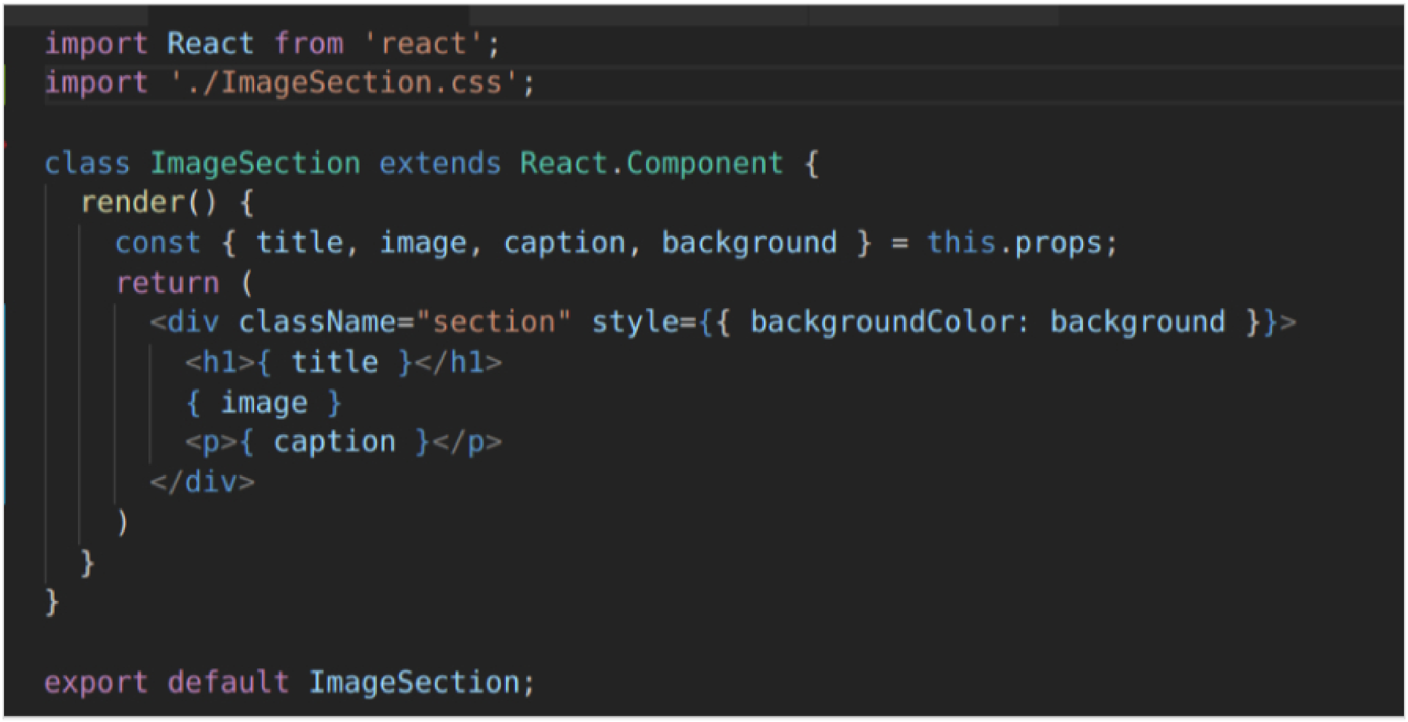
```
import React from 'react';

class HelloWorld extends React.Component {
  render() {
    return (
      <h1>Hello World!</h1>
```



```
      )
    }
  }

export default HelloWorld;
```

What you are looking at here is a basic React component that displays the text 'Hello World!' There are three main parts to focus on here: the import, the class definition, and the export. You have to import React into a file if you want to make it a React component. You define a JavaScript class that extends all the properties provided by the React.Component in order to make a component. Then finally you need to export your component to use it elsewhere.

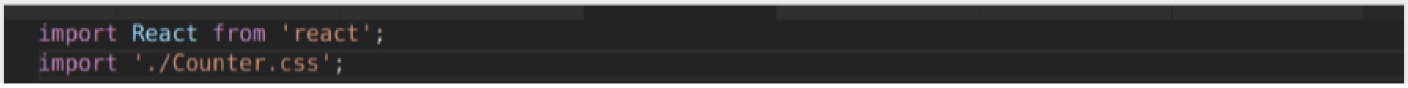
Passing Data Through Props



```
import React from 'react';
import './ImageSection.css';

class ImageSection extends React.Component {
  render() {
    const { title, image, caption, background } = this.props;
    return (
      <div className="section" style={{ backgroundColor: background }}>
        <h1>{ title }</h1>
        { image }
        <p>{ caption }</p>
      </div>
    )
  }
}

export default ImageSection;
```

The only way to pass data from a parent component to it's child is through the use of props. In order to access these props in the child component, all you need to do is define a variable with the same name as your prop and then define it by referencing this.props. In this case 'this' references the component you are currently in. You can use this data passes to the props by simply referencing the prop keyword inside curly brackets as such { prop-name }.

Responsive Data



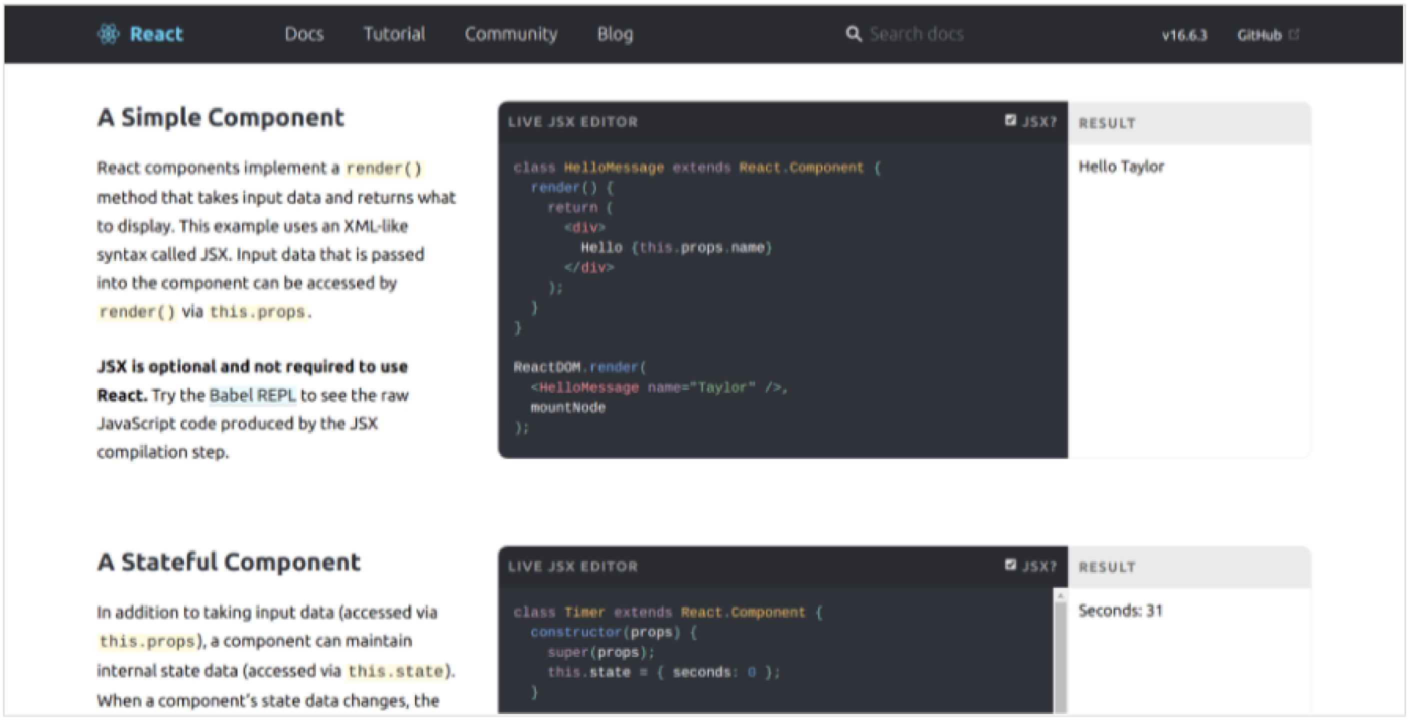
```
import React from 'react';
import './Counter.css';
```

```
class Counter extends React.Component {
  state = {
    count: 0
  }

  render() {
    return (
      <div className="counter">
        <p>{this.state.count}</p>
        <button onClick={() => {this.setState({ count: this.state.count + 1 })}}>Increment</button>
      </div>
    )
  }
}

export default Counter;
```

In order to add data to your component that you can resonsively modify all you have to do is define a components 'state' at the beginning of the class. This allows you to reference the state later on anywhere in the component by simply typing 'this.state.state-name' anywhere. In order to modify the state of a React component you call to use the built-in function 'this.setState()' and pass in the new values you want your state to be as a parameter. The state is contained within each component no matter how many instances you render so youll never run into the problem of two components modifying the same data.

0 Increment

Learning Curve



A Simple Component

React components implement a `render()` method that takes input data and returns what to display. This example uses an XML-like syntax called JSX. Input data that is passed into the component can be accessed by `render()` via `this.props`.

JSX is optional and not required to use React. Try the Babel REPL to see the raw JavaScript code produced by the JSX compilation step.

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  mountNode
);
```

Hello Taylor

A Stateful Component

In addition to taking input data (accessed via `this.props`), a component can maintain internal state data (accessed via `this.state`). When a component's state data changes, the

```
class Timer extends React.Component {
  constructor(props) {
    super(props);
    this.state = { seconds: 0 };
  }
```

Seconds: 31

This is a screenshot taken from the Reactjs website that features very basic tutorials of how to make a component. The documentation for React dives very deep, but it also starts you off with simple examples that even a beginner can understand. All of React is made in JavaScript so anything you can do in JavaScript you can do in React; this is not the case for Vue.

You should use React if:

- You're looking for a **powerful** frontend library
- You're making a **complex** application
- You want **extensive community support**