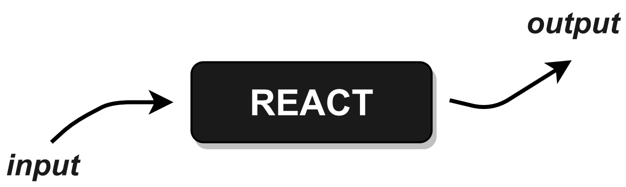
More on Props

An introduction to react components and how they communicate with one another.

WE'LL COVER THE FOLLOWING

Every React component is like a small system that operates on its own. It has its own state, input, and output. In this section, we will explore these characteristics.



React Component

As discussed in the previous chapter, the input of React components are **props**, which are the means through which data is passed to them. For example, in the following code playground, the `Title` component has only one input or prop - `text`. The parent component should provide it as an attribute while using the `<Title>` tag. The parent component in the example below is `App` and it passes “Hello React” to it.

```
import React from 'react';
import Title from './title';

export default class App extends React.Component {
  render() {
    return (
      <div>
          <Title text="Hello React" />
      </div>
    );
  }
}
```

`propTypes`

In addition to the component definition, we also have to define, at the very least, `propTypes`. Within `propTypes`, we define the type of every property and React provides hints in the console if something unexpected gets sent. Have a look at **lines 19-22** in `titles.js` where the `text` prop is set to be a string.

`defaultProps` is another useful option. We may use it to set a default value of a component's props so that if the developer forgets to pass them, we have meaningful values available. Have a look at **lines 24-27** to see how to use `defaultProps`. Try removing the input passed to `Title` in the `App` component and running the code!

Passing components as props

React does not strictly define what should and shouldn't be passed as a prop. It may be anything we want. It could even be another component,

```
import React from 'react';

export default class SomethingElse extends React.Component {
  constructor(props) {
    super(props);
  }

  render() {
  return <div>The answer is { this.props.answer }</div>;
  }
}
```

Here, `Answer` is passed to `SomethingElse` as a prop on **line 9** in `index.js`.

There is also a `props.children` property that gives us access to the child components passed by the owner of the component. For example:

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

export default class Title extends React.Component
{

  render()
  {
    return (
      <h1>
```

```
        {this.props.text}
        {this.props.children}
      </h1>

    );
  }
};
Title.propTypes =
{
  text: PropTypes.string,
  children: PropTypes.any

};

Title.defaultProps =
{
  text: PropTypes.string
};
```

In this example, `<span>community</span>` in the `App` component is `children` in the `Title` component. Notice that if we don't return `{ children }` as part of the `Title`'s body the `<span>` tag will not be rendered.

(prior v16.3) An indirect input to a component may also be the so-called `context`. The whole React tree may have a `context` object which is accessible by every component. We will further elaborate on that in the dependency injection section.

Quick Quiz on Props!

1 You can pass functions, components, and text as props

COMPLETED 0%

1 of 3