7

Props in React

MentorLabs℠

Objectives

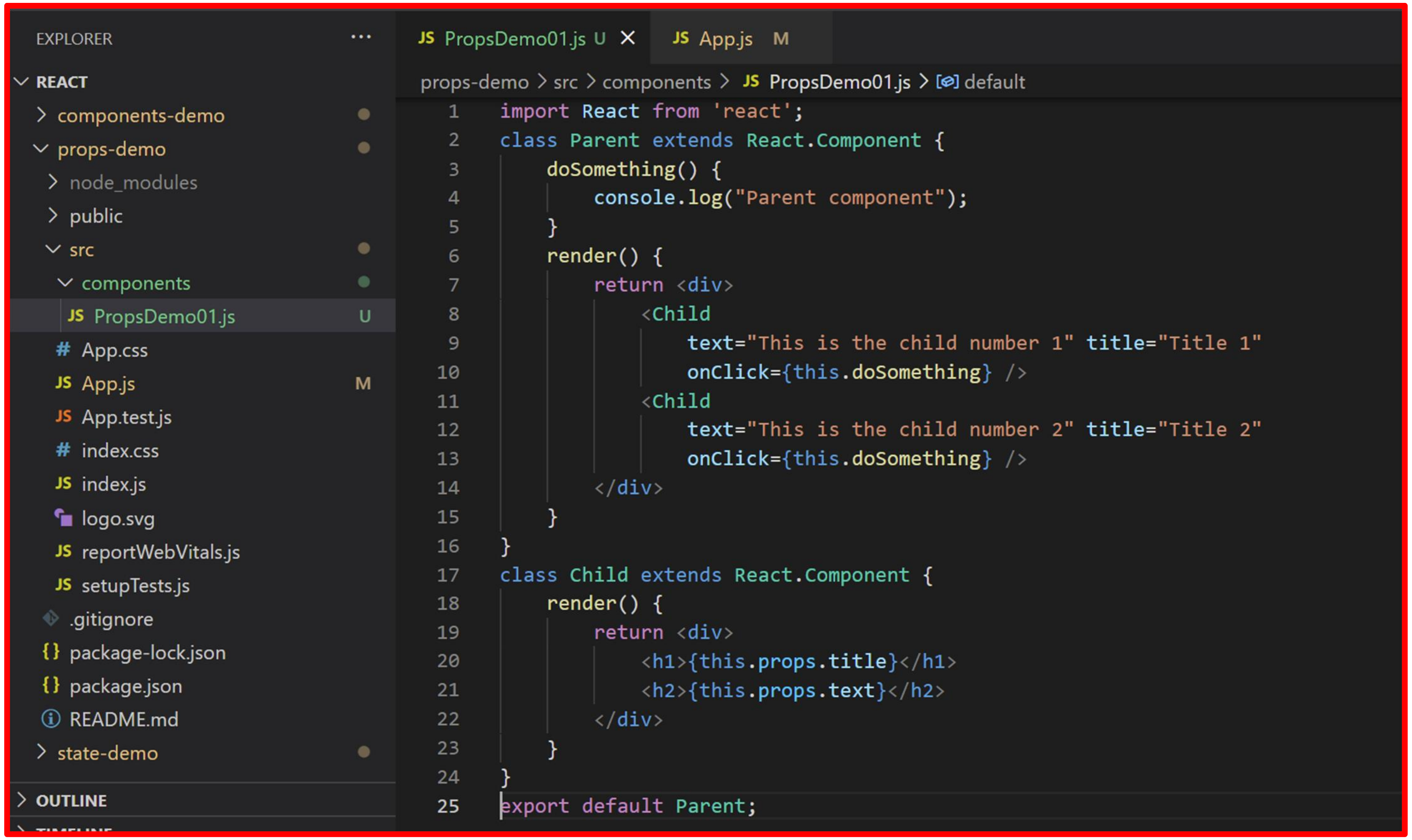
After completing this lesson, you should be able to do the following:

- Default Props
- PropsTypes
- Passing Down Props Using Spread Operator

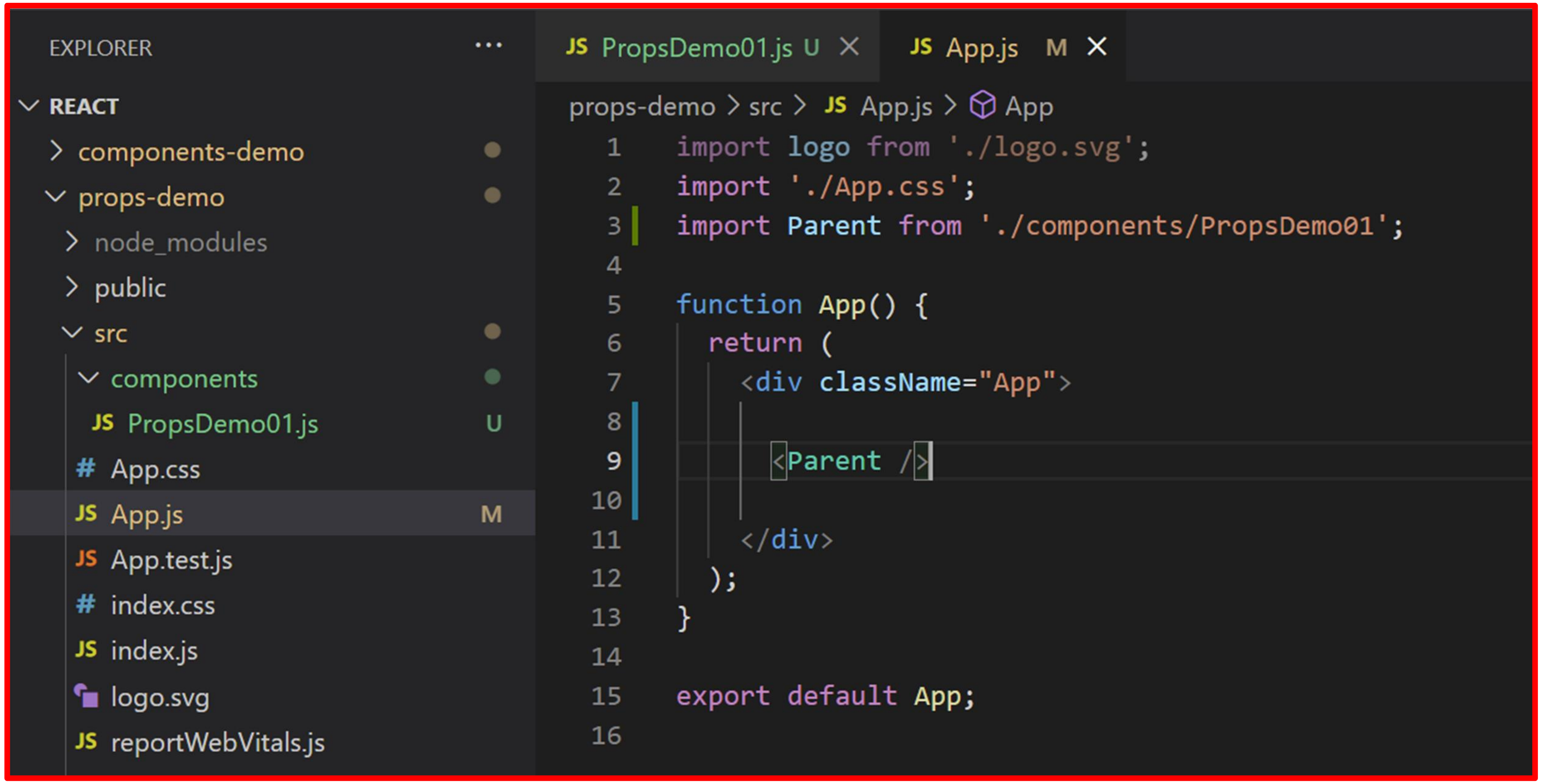
Introduction to Props

props are used to pass data and methods from a parent component to a child component.

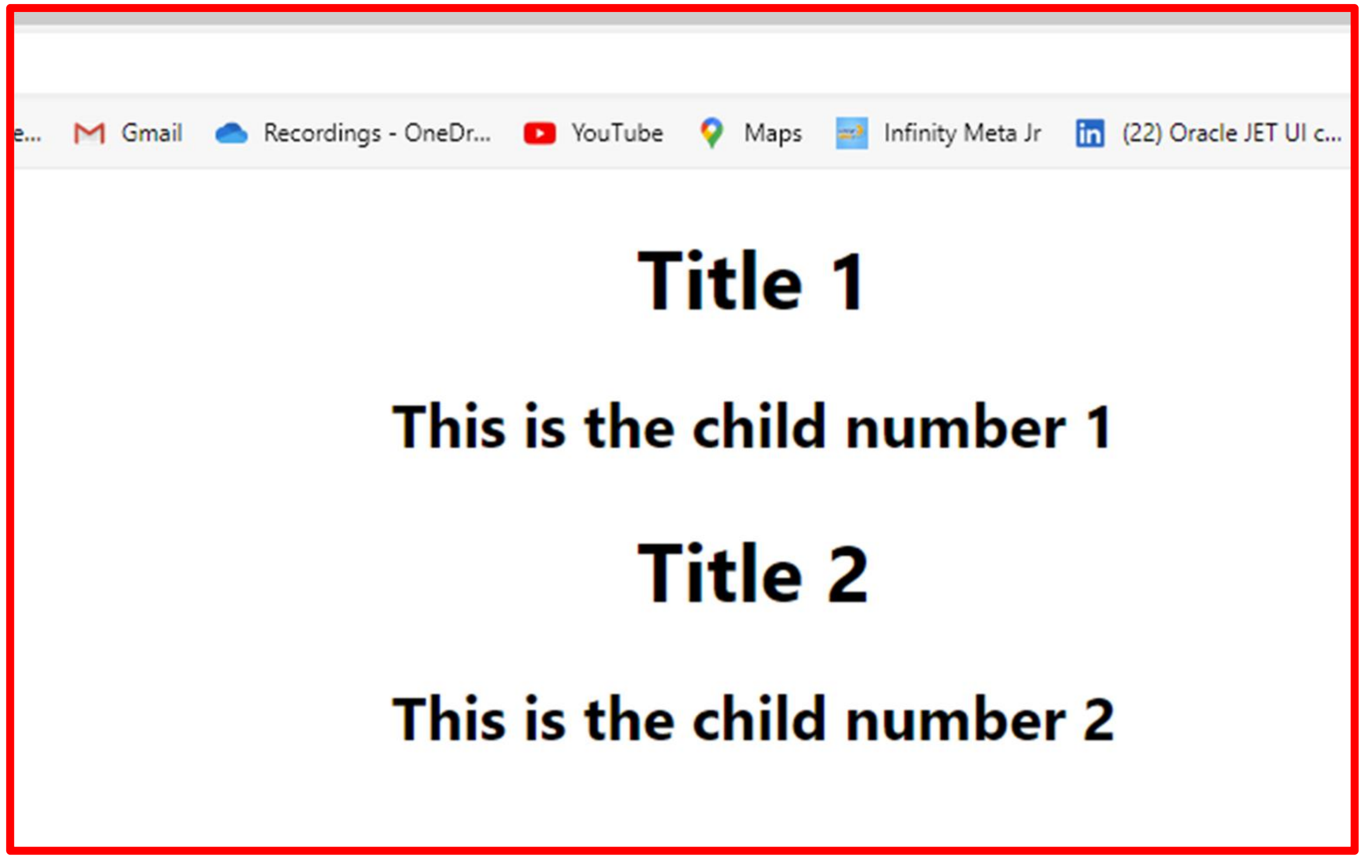
- **Interesting things about props**
- 1. They are immutable.
- 2. They allow us to create reusable components.



```
import React from 'react';
class Parent extends React.Component {
    doSomething() {
        console.log("Parent component");
    }
    render() {
        return <div>
            <Child
                text="This is the child number 1" title="Title 1"
                onClick={this.doSomething} />
            <Child
                text="This is the child number 2" title="Title 2"
                onClick={this.doSomething} />
        </div>
    }
}
class Child extends React.Component {
    render() {
        return <div>
            <h1>{this.props.title}</h1>
            <h2>{this.props.text}</h2>
        </div>
    }
}
export default Parent;
```



```
import logo from './logo.svg';
import './App.css';
import Parent from './components/PropsDemo01';

function App() {
  return (
    <div className="App">

      <Parent />

    </div>
  );
}

export default App;
```



Title 1

This is the child number 1

Title 2

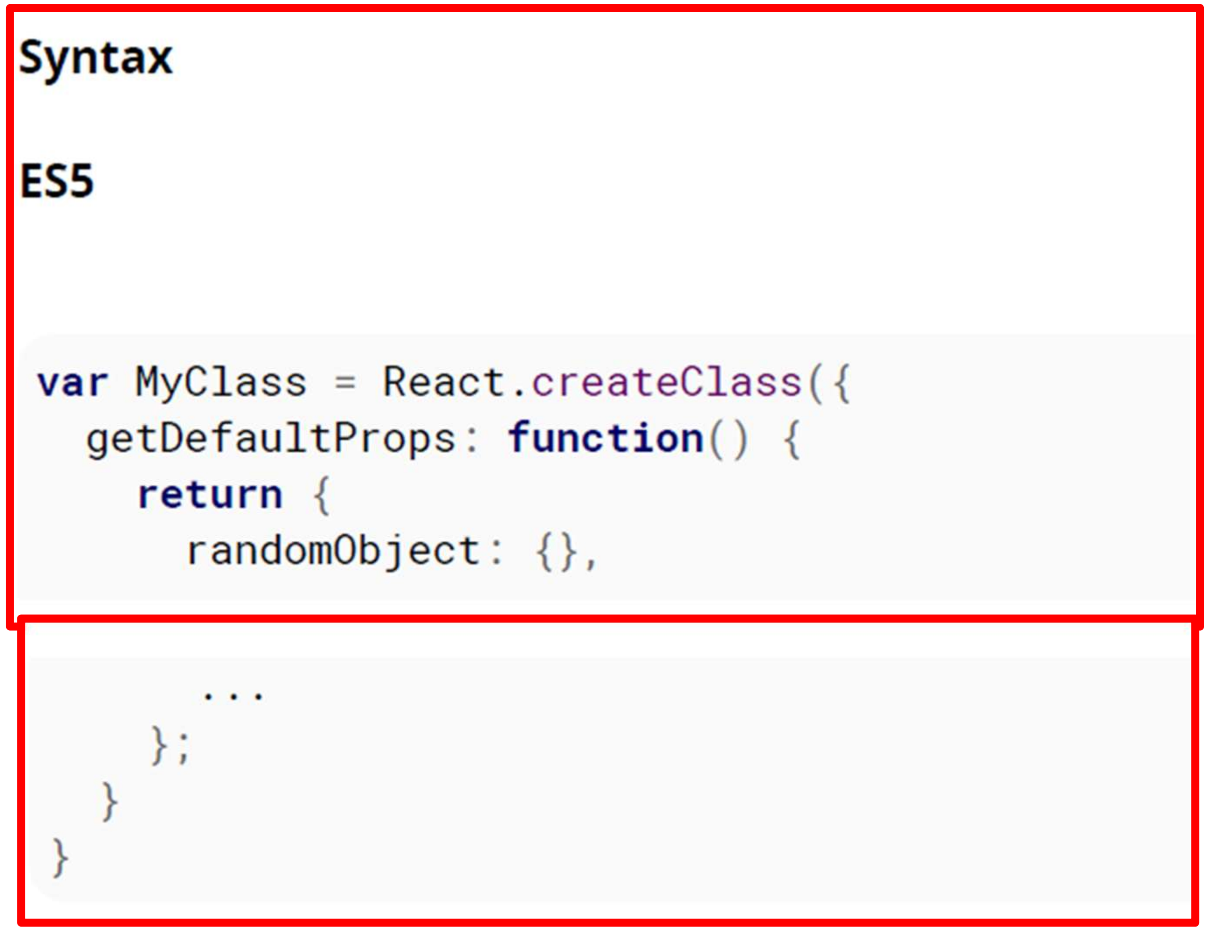
This is the child number 2

Default props

- defaultProps allows you to set default, or fallback, values for your component props. defaultProps are useful when you call components from different views with fixed props, but in some views you need to pass different value.

Syntax

ES5



```
var MyClass = React.createClass({
  getDefaultProps: function() {
    return {
      randomObject: {},
```

```
      ...
    };
  }
}
```

ES6

```
class MyClass extends React.Component {...}

MyClass.defaultProps = {
    randomObject: {},
    ...
}
```

ES7

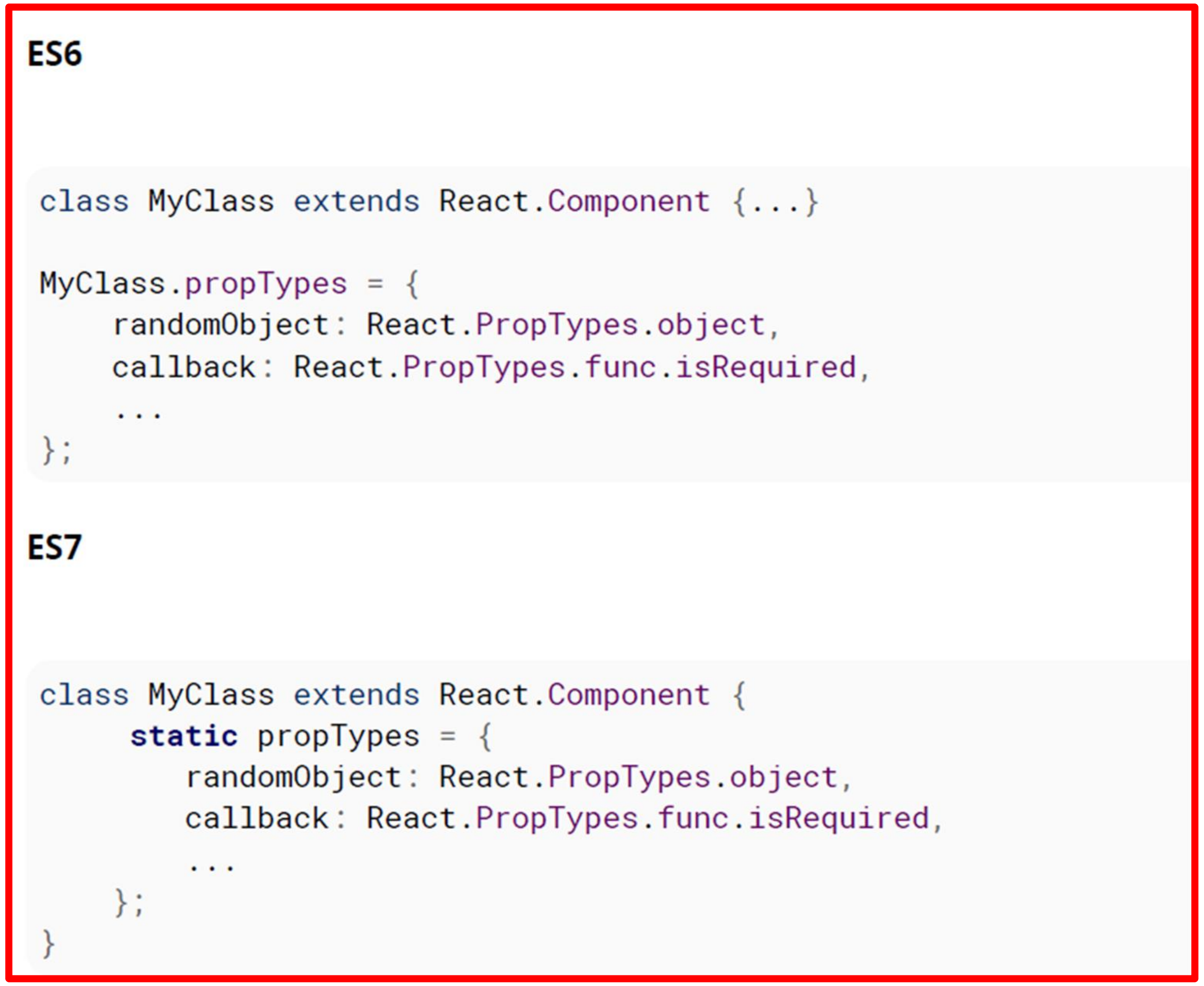
```
class MyClass extends React.Component {
    static defaultProps = {
        randomObject: {},
        ...
    };
}
```

- The result of getDefaultProps() or defaultProps will be cached and used to ensure that **this**.props.randomObject will have a value if it was not specified by the parent component.

PropTypes

- propTypes allows you to specify what props your component needs and the type they should be. Your component
- will work without setting propTypes, but it is good practice to define these as it will make your component more
- readable, act as documentation to other developers who are reading your component, and during development,
- React will warn you if you you try to set a prop which is a different type to the definition you have set for it.
- Some primitive propTypes and commonly useable propTypes are -

- If you attach isRequired to any propType then that prop must be supplied while creating the instance of that component.
- If you don't provide the **required** propTypes then component instance can not be created.



ES6

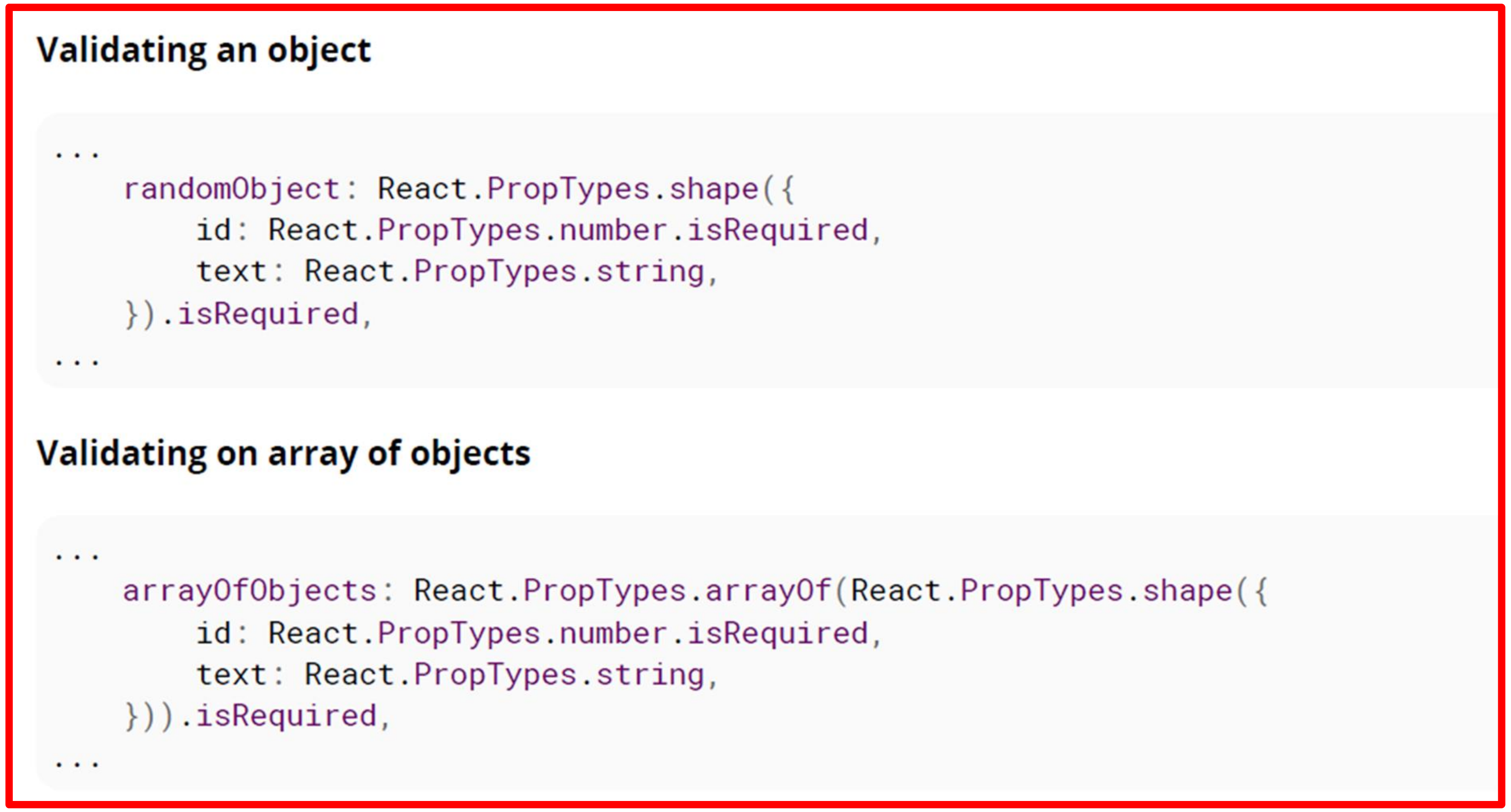
```
class MyClass extends React.Component {...}

MyClass.propTypes = {
    randomObject: React.PropTypes.object,
    callback: React.PropTypes.func.isRequired,
    ...
};
```

ES7

```
class MyClass extends React.Component {
     static propTypes = {
        randomObject: React.PropTypes.object,
        callback: React.PropTypes.func.isRequired,
        ...
    };
}
```

More complex props validation

- In the same way, PropTypes allows you to specify more complex validation



Validating an object

```
...
    randomObject: React.PropTypes.shape({
        id: React.PropTypes.number.isRequired,
        text: React.PropTypes.string,
    }).isRequired,
...
```

Validating on array of objects

```
...
    arrayOfObjects: React.PropTypes.arrayOf(React.PropTypes.shape({
        id: React.PropTypes.number.isRequired,
        text: React.PropTypes.string,
    })).isRequired,
...
```

Passing down props using spread operator

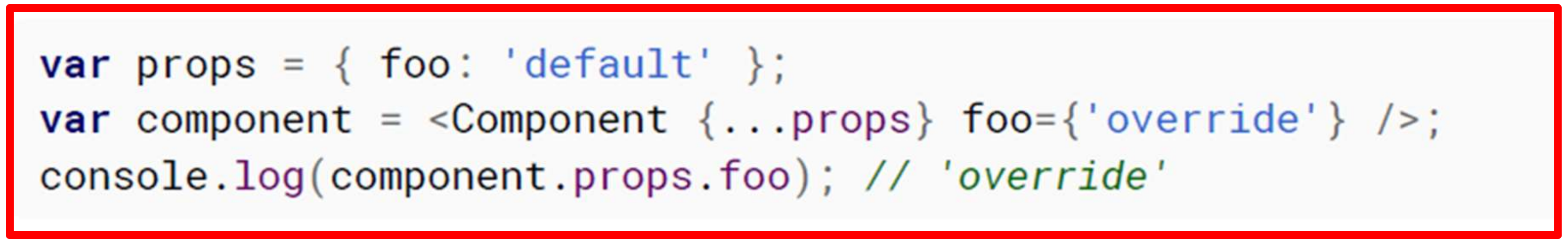
- Instead of

```
var component = <Component foo={this.props.x} bar={this.props.y} />;
```

- Where each property needs to be passed as a single prop value you could use the spread operator ... Supported for arrays in ES6 to pass down all your values.

```
var component = <Component {...props} />;
```

- Remember that the properties of the object that you pass in are copied onto the component's props.
- The order is important. Later attributes override previous ones.



```
var props = { foo: 'default' };
var component = <Component {...props} foo={'override'} />;
console.log(component.props.foo); // 'override'
```

Summary

In this lesson, you should have learned how to:

- Default Props
- PropsTypes
- Passing Down Props Using Spread Operator