The ReadingTime component

Now it's time to start building our main component that will make this application do something!

Props

Up until now we've used the word `props` here and there, but what really are these mystical things called `props`? Props are simply properties that are passed into a React component for consumption by said component.

```
<Component prop1='foo' prop2={someFunction} />
```

In that short snippet, `prop1` and `prop2` would be available to use in the `Component` component.

```
class Component extends React.Component {
  render() {
    return (
      <div>
        <button onClick={this.props.prop2}>{this.props.prop1}</button>
      </div>
    );
  }
}
```

Easy, right? Anything we give to the component as an "HTML attribute" is available as a prop in the component. Unlike HTML attributes, we can pass JavaScript variables by replacing the quotes with curly braces.

propTypes

We briefly discussed propTypes in the previous section, but let's put it to use in real life for our new component. First, create a new file in the `src` directory called `reading-time.jsx`. Let's import React and create the skeleton for this component, remembering that there must **ALWAYS** be a render function in the component.

```
import React from 'react';

export default class ReadingTime extends React.Component {
  static propTypes = {}

  render() {
    return (
      <div></div>
    );
  }
}
```

Now we've got an empty React component. The next thing we need to do with it is define what propTypes this component will be expecting. One thing that we will definitely need to know is how many words per minute we expect people to be able to read.

It looks like we're going to need a number property on this component in order to be able to calculate the reading time. Let's go ahead and add this `propType`:

```
static propTypes = {
  wordsPerMinute: React.PropTypes.number
}
```

React will raise a warning in the console if one of these props is missing or not the proper type, but will not raise an exception or cause your program to stop running.

More ES6 syntax

You might be asking yourself, what's with this `export default` and `static` stuff? I thought you'd never ask.

Module support is baked into ES6, and we've already seen how we can *import* code into a module that we need, but how do we get code *out* of a module that we want to be available to other programs or our application? With the `export` keyword of course! Any function, variable, or class that we precede with `export` will be available when the module is imported from other places in the application, but you have to explicitly ask for it:

```
export add = function(num1, num2) {
  return num1 + num2;
}
```

Some time later, in a module far, far, away....

```
import MathFuncs from 'math';

MatchFuncs.add(1, 2);
```

This is where the `export default` comes in. If you tell the module to export something as the default, that's just what the variable name you define when importing becomes, instead of the entire module:

```
export default function(num1, num2) {
  return num1 + num2;
}
```

Some time much later, in a module much, much farther away....

```
import add from 'math';

add(1, 2);
```

See the difference there? Typically in a React application, each component module will export only the component class that is created, which is why we define the class with the `export default` prefix.

The static keyword is akin to class methods in other languages. It can be used to define functions and variables that are available without creating a new instance of the class:

```
class MyClass {
  static foo = 'This is very, very foo'
}

MyClass.foo
-> This is very, very foo
```

defaultProps

The next thing we're going want to address is default properties. We **should** be able to use this component without anything actually being passed in as props. We should define some sensible defaults that make it easy for people to use the component without too much configuration.

Add this to your component just below the propTypes declaration we just added:

```
static defaultProps = {
  wordsPerMinute: 270
}
```

That's it! Now we have some sensible defaults that we can use in the event that someone tries to use our component without giving it any props whatsoever.