Hack @ Brown 2020: React Starter Project

developed w/ love by Katherine Sang

WORKSHOP MATERIALS

Stencil Code: Code Sandbox / Github
Final Code: Code Sandbox / Github
Slide deck

SET UP

Go to the stencil code and click 'fork'. You're all set! :)

ESTABLISH THE UI

Add converter and header to App.js

In `App.js` render section, add a header and converter component with the following code.

```
<Header title="Unit converter" />
<Converter />
```

Notice that the Header passes a props called "title", whereas the converter has no props.

Props for header

In `Header.js` render section, replace filler text with

```
{props.title}
```

This makes your Header a reusable component. This practice is ***very*** common in React!

Create the input text box

Now, we are moving onto creating the UI for the converter.

In `Converter.js`, in the `input` div:

```
<input type="text" onChange={ e => {this.onTextInput(e) }} /> blue room muffin(s)
```

This creates a text box, that calls onTextInput when the user types anything into the text box. We refer to method contained in a component class by putting the `this` keyword. Note that we haven't written `onTextInput` yet, so it won't do anything right now. We use `{ e => {` … }} function notation because we want to pass the event into the function to receive the text inputted.

Create content for the output.

In `Converter.js`, in the `output` div:

```
{this.getOutput()}
```

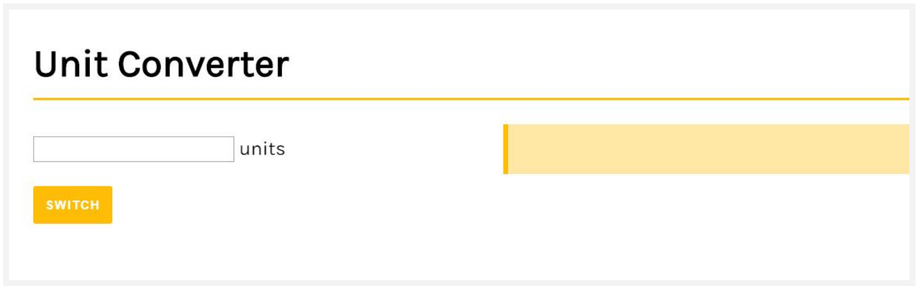
This will call our getOutput function that we will later write, and render whatever it returns. Notice it's in brackets because we are embedding Javascript into our HTML. If we did not have the brackets, it would literally print this.getOutput() into our web page.

Create the button

Now, we'll create a button to 'switch' our units from scili -> muffin to muffin -> scili.
In `Converter.js`, after the the `converter__content` div,

```
<button type="button" onClick={this.onSwitchUnits} class="button">Switch</button>
```

Checkpoint 1: Your web app should look like this, and do nothing :')



ADD INTERACTION WITH CONVERSION

Add converter functions

In `Converter.js`, before render, this is how your conversion functions (could potentially) look. We will use these functions in getOutput later.

```
   muffinToSciLi(muffins) {
        return parseFloat(muffins) / 847 + " scili(s)";
    }

    sciliToMuffin(scili) {
        return parseFloat(scili) * 847 + " blue room muffin(s)";
    }
```

Note: We use parseFloat because our inputs are strings, so we want to convert them into floats to perform math operations on them. We return it as a string because it'll be nice for our outputs to have a unit.

Set up state for text input, and store it when the input text updates so the component has knowledge of the update.

In `Converter.js` constructor, this is what your state looks like:

```
    this.state = {
        input: "0",
    }
```

You should also add these bind functions into your Converter.js function:

```
this.onSwitchUnits = this.onSwitchUnits.bind(this);
this.onTextInput = this.onTextInput.bind(this);
```

This is because we want to access the state of the component in this function. Without these two lines of code, if we type `this.state`, it'll refer to a different 'this'. This is a bit confusing but if you want to learn further rationale about this, you can read this article.

In `Converter.js`, this is how your onTextInput function looks like:

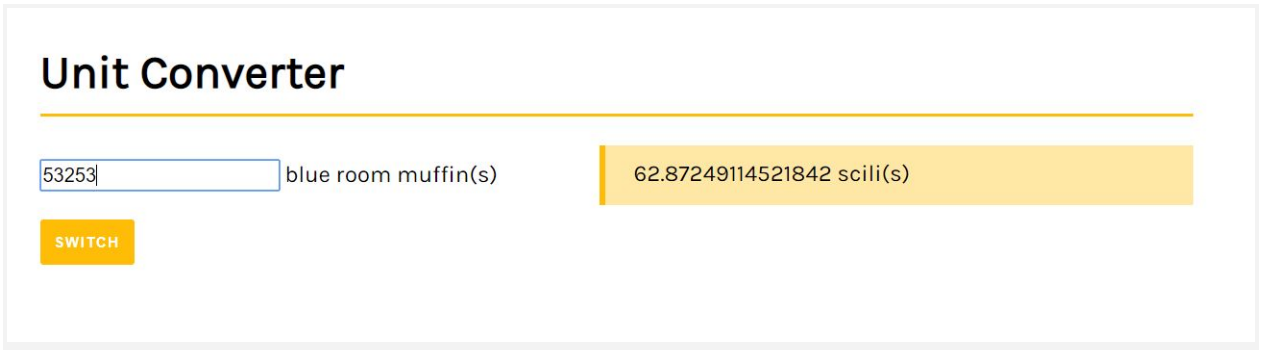
```
onTextInput(e) {
        if (e.target.value.length > 0) {
            this.setState({input: e.target.value});
        } else {
            this.setState({input: "0"});
        }
    }
```

Use the state to call the conversion function.

In `Converter.js`, update your getOutput() function to look like this:

```
   getOutput() {
       return this.muffinToSciLi(this.state.input);
    }
```

Checkpoint 2: Upon interaction, we can convert from muffins to scili! However, the switch button doesn't work yet :(


Unit Converter

53253 blue room muffin(s)

62.87249114521842 scili(s)

SET UP SWITCH FUNCTIONALITY

Set up the state so there's an input type.

In `Converter.js` constructor, this is what your state should be:

```
this.state = {
       input: "0",
       inputType: "blue room muffin"
}
```

Our state now contains an inputType so we know what our starter conversion is. The component needs to know this information in order to determine the function to do the conversion (muffinToScili or sciliToMuffin).

Write the onSwitchUnits to handle the button event

In `Converter.js`, in the `onSwitchUnits` function:

```
   onSwitchUnits() {
        this.setState({inputType: this.state.inputType === "blue room muffin" ? "scili" :
"blue room muffin"})
    }
```

When the user clicks 'Switch', they expect the unit conversions to change, thus our inputType changes!

Modify getOutput to account for which conversion mode we are in.

Right now, our converter is returning the same conversion (muffinToScili), even though our state now properly updates. Now, we need to use this information in our state to choose the correct function to do the conversion computing. Thus, we need to update getOutput.

```
   getOutput() {
        if (this.state.inputType === "blue room muffin") {
            return this.muffinToSciLi(this.state.input);
        } else {
            return this.sciliToMuffin(this.state.input);
        }
    }
```

Change input label to reflect the input type

Before, we hardcoded our input button output, but now it's dynamic because of the switch buttons. Let's reflect this by using the inputType state as the label! This is how your input div in render should look.

```
<div className="input">
    <input type="text" onChange={e => {this.onTextInput(e)}}/> {this.state.inputType}(s)
</div>
```

We're done! Your web app should now match the demo! :) Congrats!

WANT TO DO MORE WITH THIS PROJECT?

- When you click switch, it'll switch the input number and output numbers, because currently it doesn't do that :(
- How about handling unit conversion with multiple different units, similar to how Google's converter works? Scili to muffins? Joe's sandwiches to scili? The length of the main green to muffins? There are some ***big*** questions out there that need answering!
- Flash your CSS skills and make it mobile responsive, add images, etc.

WANT TO EXPLORE MORE WITH REACT?

- Make a to do list where you can archive tasks, mark them as done, or store them
- Make a filter for classes at Brown
- Make your own portfolio website with React and (bonus!) Gatsby
- Make tic-tac-toe or other fun board games
- If you're into vector art, try making your own React components like this library
- Look into React Router for page redirection

RESOURCES

- Set up your own web app with create-react-app
- Learn more about bind
- React documentation site
- Deploying your react app on netlify or general tutorial