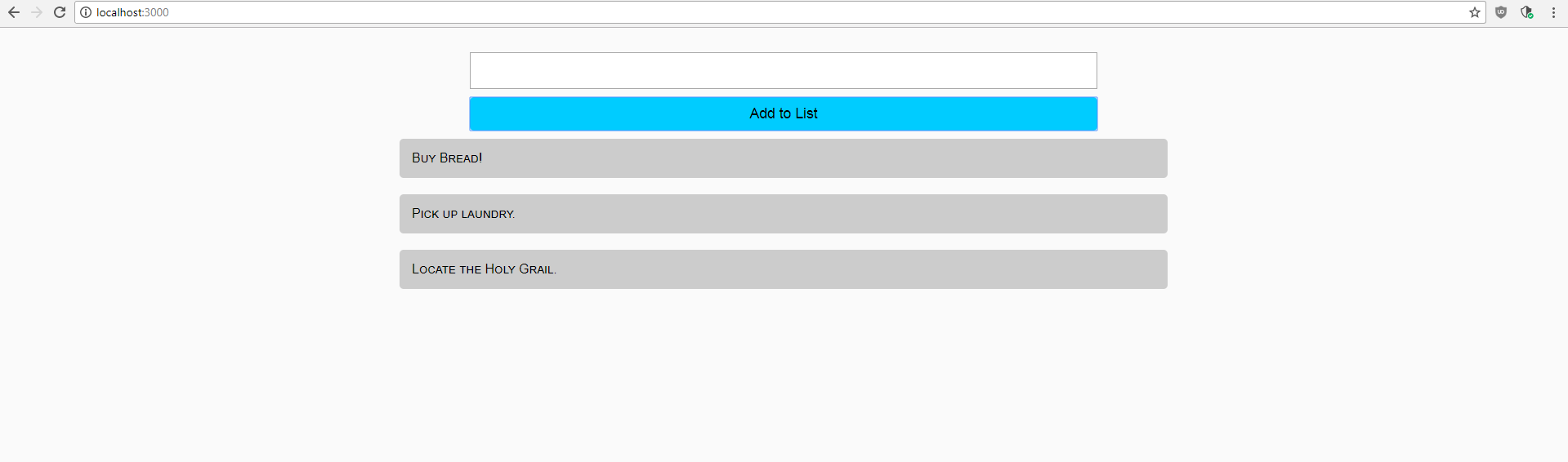
# **ToDo-List**

What we will be Creating

  
  
Setting up the Project

We are going to begin by cloning the React-Boilerplate repository.



Once you have cloned the Boilerplate onto your own computer, you can then rename the folder to “ToDo-List”. Inside of the “ToDo-List” folder, go ahead and open a Terminal. When using the Boilerplate, you will need to install the necessary JavaScript packages in order to begin development.



Then start the Front-End development server.



Creating the HTML Layout

We will create a column-style layout with an input box at the top so we can add more items to our ToDo List. First, let’s create the layout for the list. Start by making a new **<div></div>** element with a className attribute of “todoList”. Inside of this new **<div></div>,** create another **<div></div>** element with a className attribute of “listItem”, you can place some filler text inside of this element. Above your “todoList” className **<div></div>** element, go ahead and create another **<div></div>** element with a className of “inputContainer”, which will hold our **<input />** elements. Inside of this element, you will add the first **<input />** element with a type=”text” and a className of “todoInput”. Underneath this **<input/>** element, you will another **<input />** element with a type=”submit” and a className of “todoButton”.

*containers/Home/index.js*



Adding CSS to the List

In the *style.css* file, you can go ahead and add a class for todoList. We will set the **display:flex** and **flex-direction:column** so the list items stack on top of each other. We will also set the **width:50%** and **margin:0** auto to make the list as wide as half the page and center it.

Next, we are going to style the list items with a listItem class. Give the class a **background:#CCCCCC** so we can see our items, a **padding:15px** and **margin:10px** to space each item out. For added style, let’s also add a **border-radius:5px** and a **font-variant:small-caps** to our class.

*containers/Home/style.css*



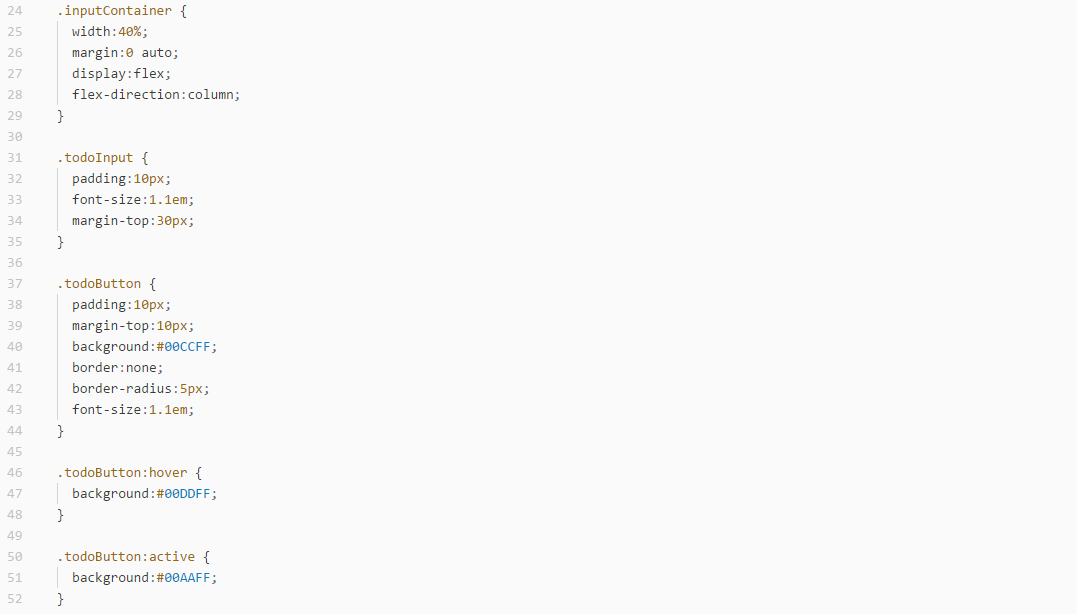
Now that the list is styled, we will go ahead and style the inputs. Starting with an inputContainer class, give it a **width:40%** and **margin:0 auto** to center it. We will also set **display:flex** and **flex-direction:column** so our input field and submit button stack on top of each other.

Create a todoInput class with a **padding:10px** and **margin-top:30px**. In order to see the text better, we will add a **font-size:1.1em**.

For the submit button, create a todoButton class with a **padding:10px** and **margin-top:10px**. Be sure to set **border:none** so the ugly default borders are removed. Give the submit button a **background:#00CCFF**, for a nice light blue color. Additionally, you can add **font-size:1.1em** and **border-radius:5px** for a more appealing button.

We will also add selectors to our button so the button changes color when a user hovers over the button and then clicks on it. Create another todoButton class with a **:hover** selector attached to it and give it a **background:#00DDFF**. Next, create another todoButton class with a :active selector attached to it and give it a **background:#00AAFF**. This way, when a user hovers over the button, it becomes slightly lighter and when they click on it, the button will become slightly darker.

*containers/Home/style.css*



Adding JavaScript

For our ToDo-List, we will need to use JavaScript to store our list items, handle our input field, and add new items. We will require three functions to make this work. The first function is our constructor, where we will define our state variables. The first state variable we require is “listItems” which is an array. This variable will hold all of the items that were already added to the list. The second state variable is our inputItem which is an empty string. This variable will hold a new item from the input field before the submit button is clicked and it is added to the listItems array.

*containers/Home/index.js*

**

We will also add an iteration to our HTML inside of the todoList **<div></div>** element. This iteration will contain the listItem **<div></div>** element and will iterate over our listItems state variable. It will output the singular iteration variable inside of the listItem **<div></div>** element.

*containers/Home/index.js*

**

You can test this by adding an array item or two inside of the listItems state variable.

The next function we need will be used to handle our input from the input field and store it into the inputItem state variable. We will call this function “handleItem” and it takes a parameter called “event”. This function will take the data from the input box based on the event.target.value and then use setState to the data into the inputItem state variable.

*containers/Home/index.js*



We will then add this function and state variable to the <input /> element using an onChange attribute and value attribute respectively.

*containers/Home/index.js*



The last function we will create is the storeItem function to take the inputItem state variable and add it to the listItems state array variable. We will create two variables in this function. The first variable will be called listItems which will be equal to this.state.listItems and the second will be called inputItem which will be equal to this.state.inputItem. We will then create an if-statement which will check to see if inputItem is an empty string. If inputItem is not an empty string, we will add the inputItem to the listItems array. In order to add anything to the end of an array, we will use a built-in JavaScript function called “push”. Then we will use setState to reapply the newly amended list and clear the inputItem state variable.

*containers/Home/index.js*

**

Then, we will add the storeItem function to the submit button using the onClick attribute.

*containers/Home/index.js*



Now, you should have a completed and fully functioning ToDo-List.