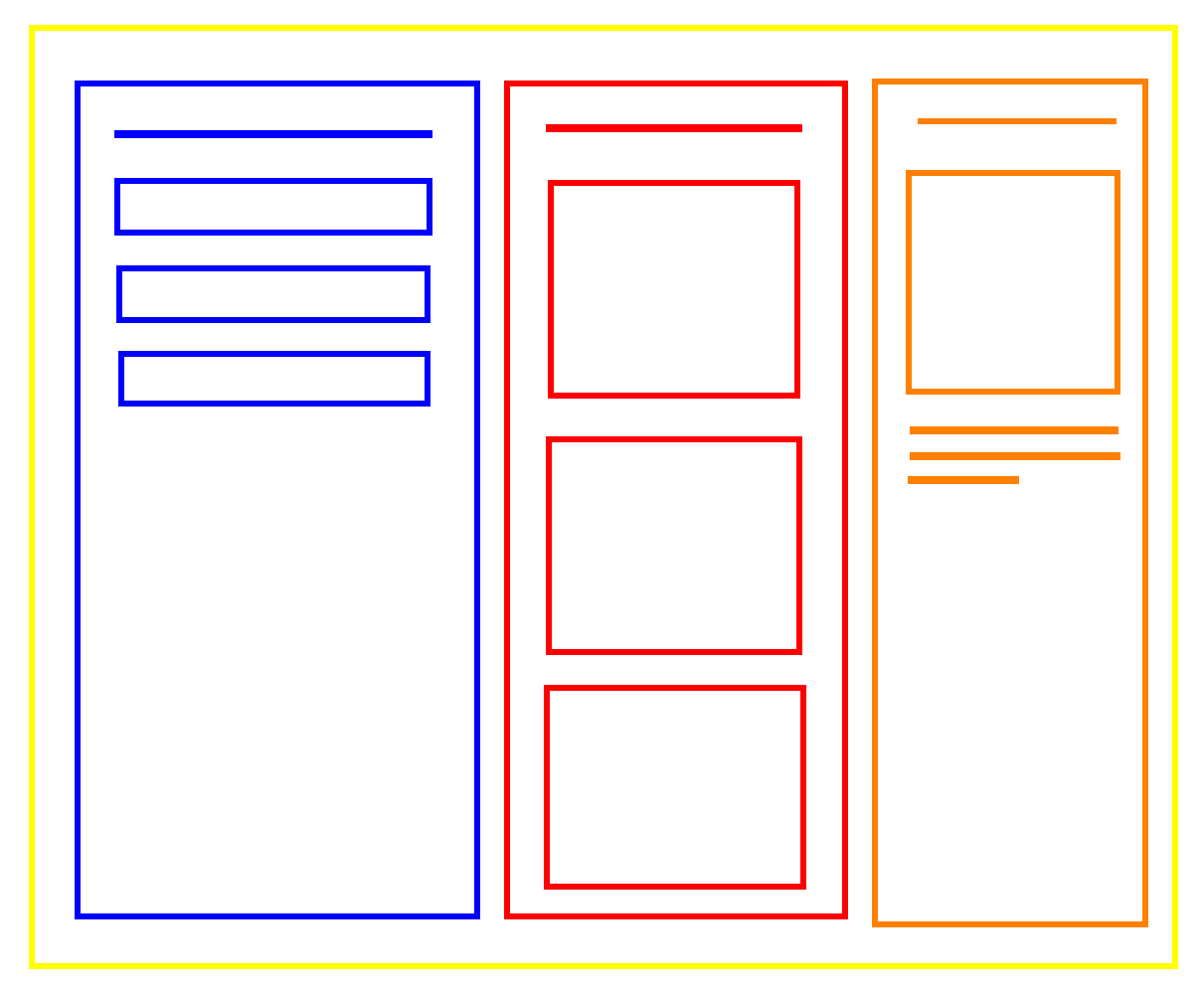
# Exercise: React Components

This is an exercise for the ["ReactJS Fundamentals Course"](https://softuni.bg/opencourses/reactjs-fundamentals) at [SoftUni.](https://softuni.bg)

## Task Requirements

Create a page with 3 sections. It should have the following structure:

* **App** Component
  + **Streets** Section
  + **Houses** Section
  + **HouseDetails** Section



On the **blue side**, you have a list of **street names**, in the **middle** you can see all the **pictures of the houses that are in the currently hovered street**, on the **right we have the currently hovered house** with the **type, picture, description and price**.

## Database

Use the provided **REST API** build with **Express.js** and **MongoDB** that works with **streets** **and houses**.

You can download **MongoDB** from [here](https://www.mongodb.com/download-center/community) and use a GUI with it like [Robo3T](https://robomongo.org/)

Each **Street** should have a **location (string)** and a **collection of houses (objectId)**.

Each house should have a **type (string), a description (string), imageUrl (string), price (number)**.

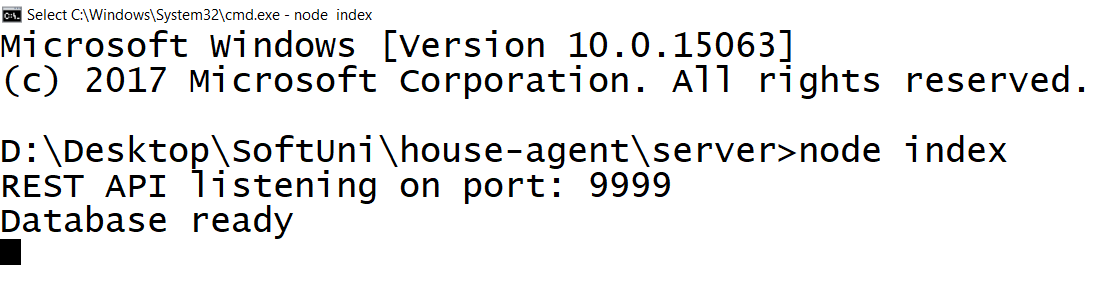
Use **Postman** to create some entities or use **"mongoimport"** to [import the data](https://stackoverflow.com/questions/15171622/mongoimport-of-json-file) that is provided to you.

## Initialize a React App

Use the **"react-create-app"** command to create a new React App.

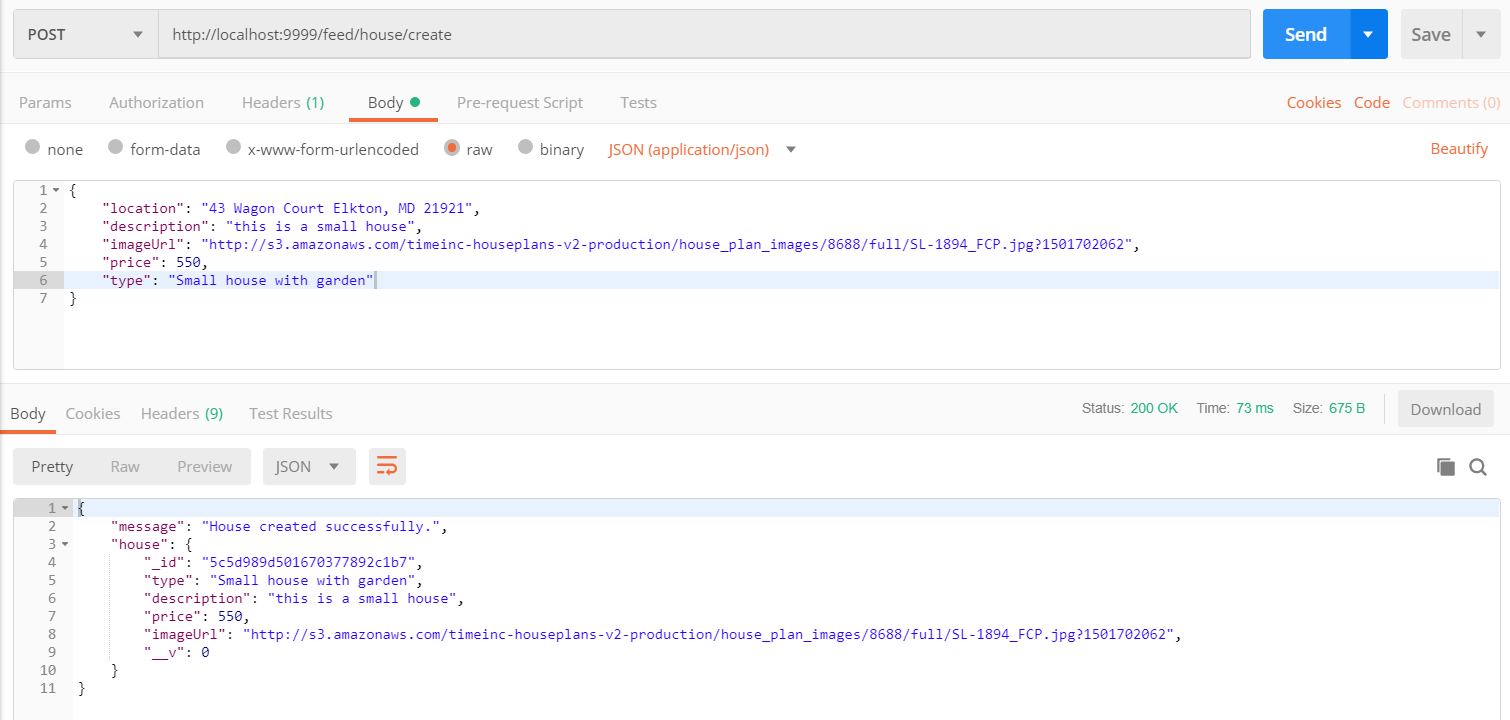
### Start the Provided Server

The server will listen on port **9999** by default. In order to fetch the data that you fed in the database you will have to make a **get request** on **'http://localhost:9999/feed/street/all'.** We will see how we can do that later. For now, just start the server.

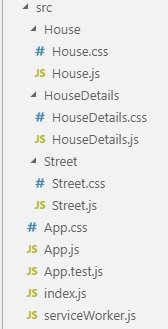


### Test the server

Using **Postman** and try to create some **houses or streets**:



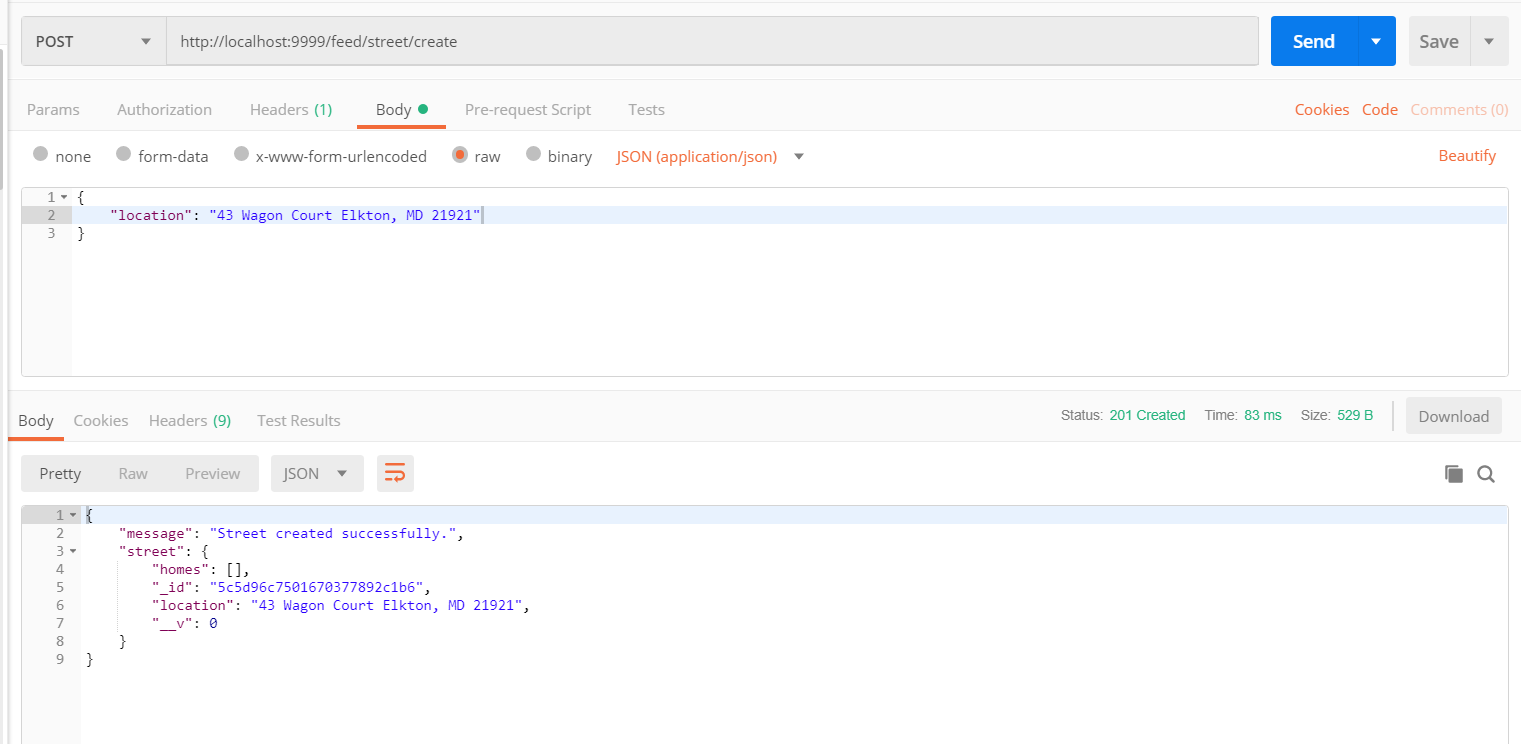
## Folder Structure

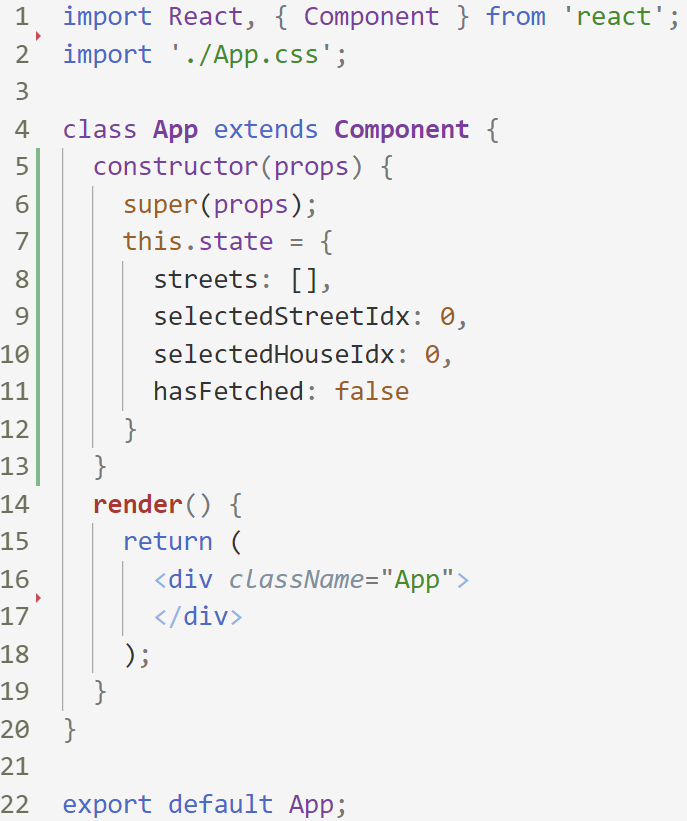


As you can see, for **each component**, we have a **separate folder** that includes the **component and the style** for that component.

**Create** all of the files as shown in the picture. We are going to write code in them just in a bit

## Fetching Data

In the **App.js** folder remove the unnecessary code and add a **constructor** for the app, which will initialize a **state of the component**



We are going to need an **array with all of the streets**, the **currently selected indices** of the **house and the street** in order to know which **piece of information to render**. The final thing we need to know is if we have loaded all the info from the database (since **fetching takes time**)

Now add the **componentWillMount** function to fetch the data before we even try to render it:



Here, we **fetch** the data from our server; we **convert it to json** and **set the new state of our component**

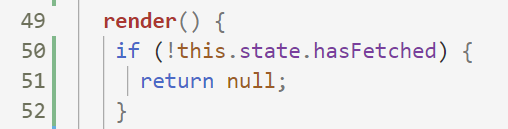
## Streets Section

In the **Street.js** file add the following code:



For now our **Street component** will only get its **location**.

Now let us go to our App component and add the following:



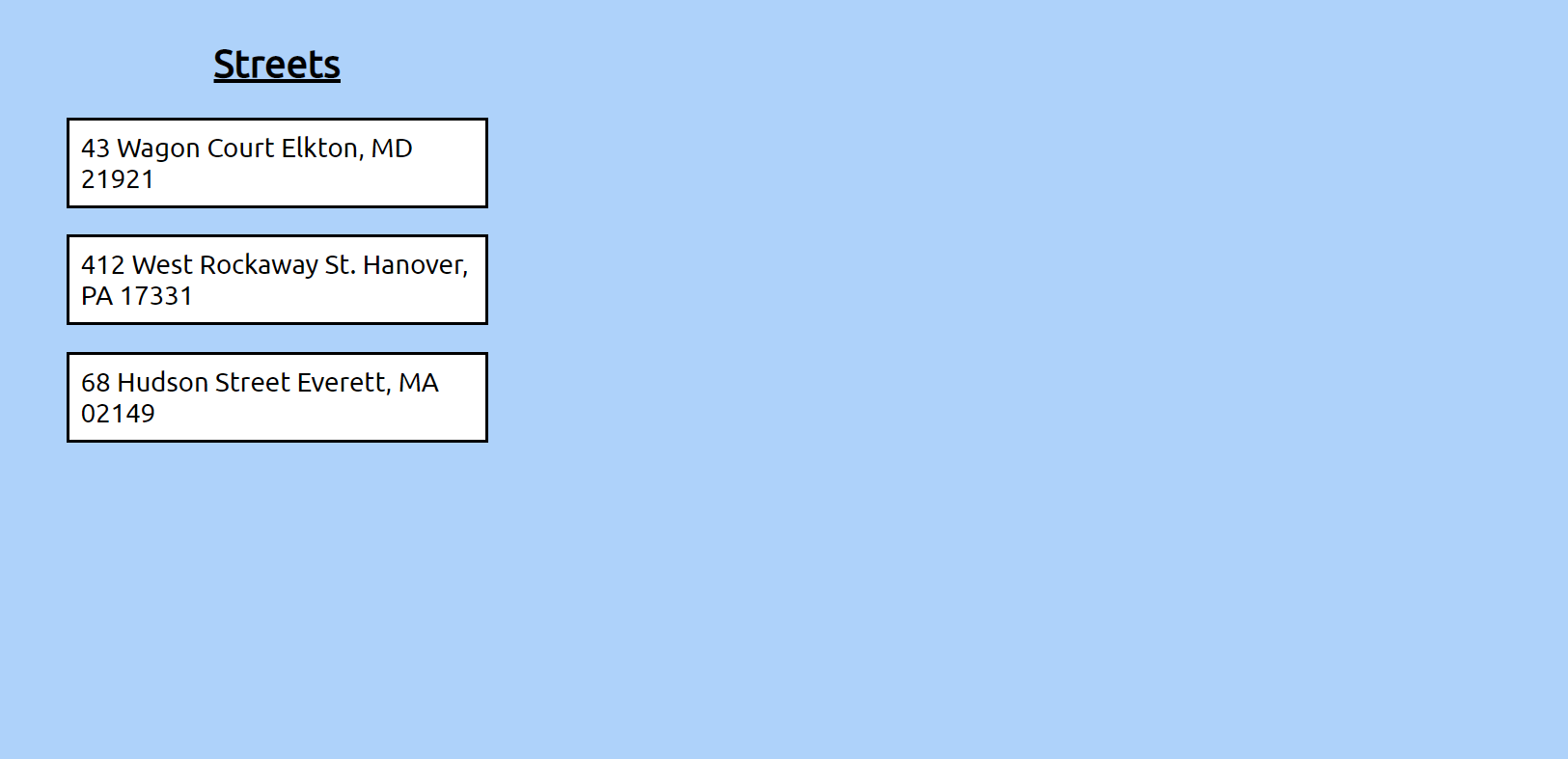
* This will make sure that we **do not proceed** to the **rendering** if we have **not yet fetched the data**

Before we test our app, let us render our streets. Add the following code:



Here we loop through all the streets and for each of them we render the street component (**don't forget to import it though**). We pass as properties the location of the street, a key, and the id (the index of the street).

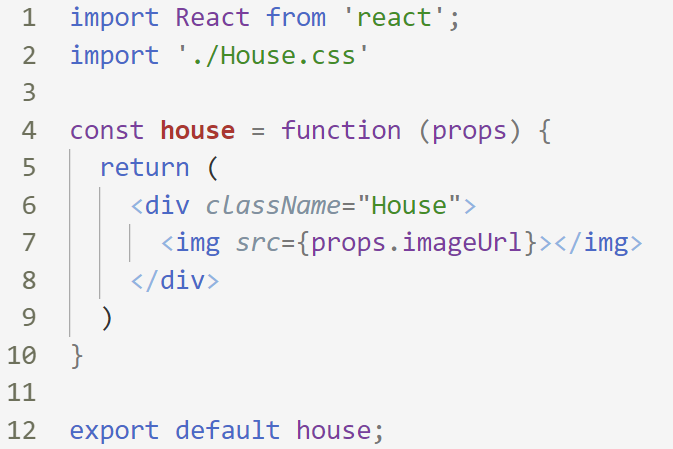
***Note:*** we do all of that only if the length of our streets array is greater than 0, otherwise we will get an error



**Success!** For now, let us leave the **Street** component and go to the **House** component

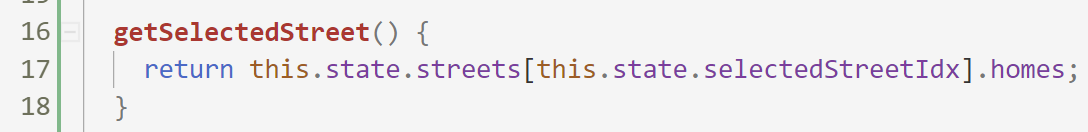
## Houses Section

Add the following code to the **House.js** file:



Since we **won't keep** any **state** in this component, we can **make it functional**. It is better to use functional components in that case. **Use them as often as possible**. You can now **go back to the Street** component and **make it functional** also (since there we don't keep any state either)

Now go back to the App.js and add the following helper function:

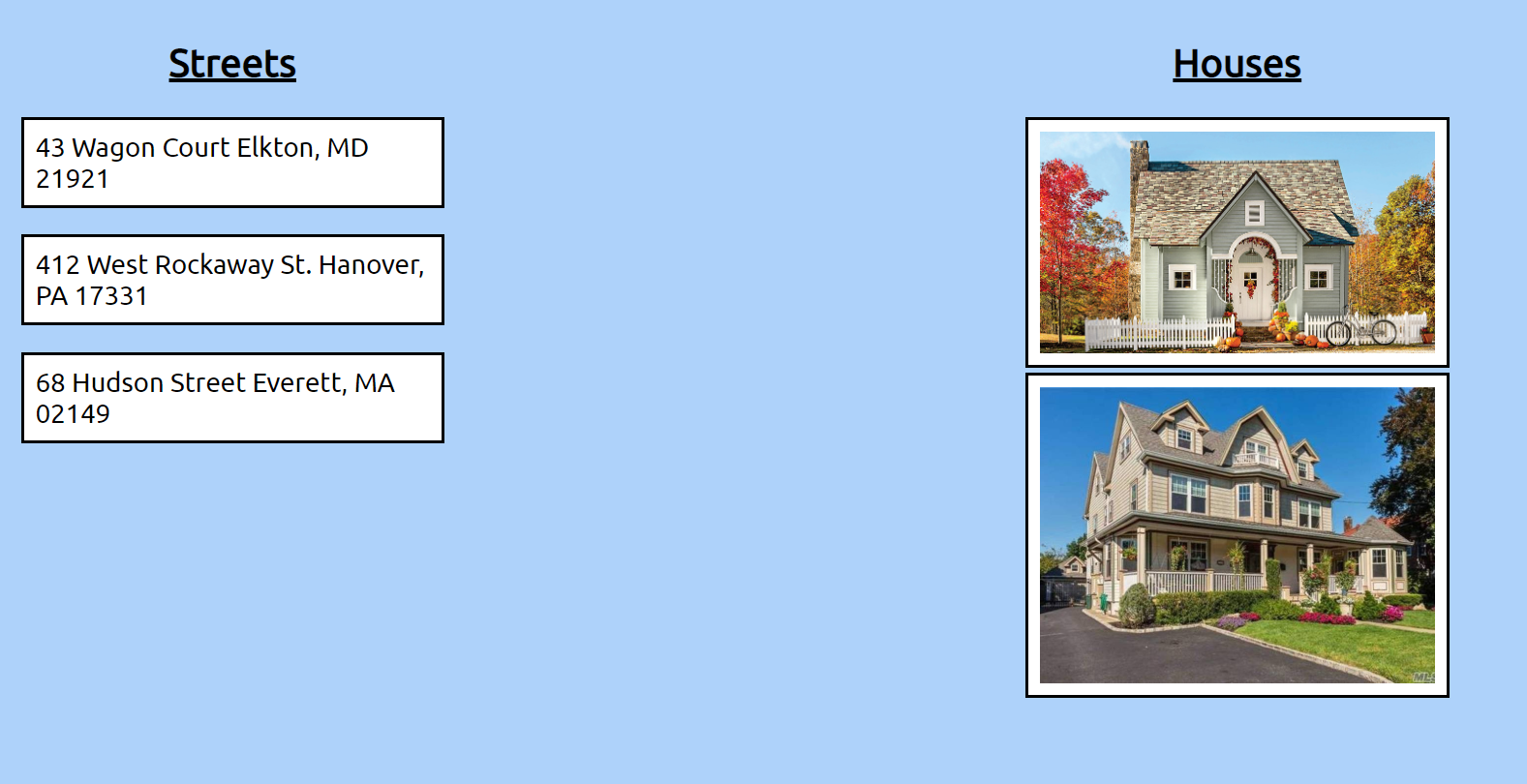


This function will return us **all the houses** in the **currently selected street**.

And now we have to get them and render them:



Now you should see something like that:



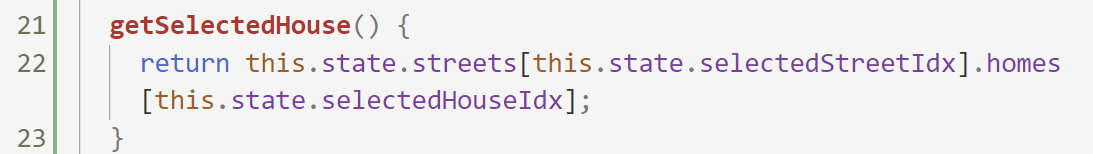
We will come back to this component later. For now, let us proceed to the **HouseDetails.js**

## House Details Section

Add the following code to the **HouseDetails.js**



Return to the App.js and add the following helper function:



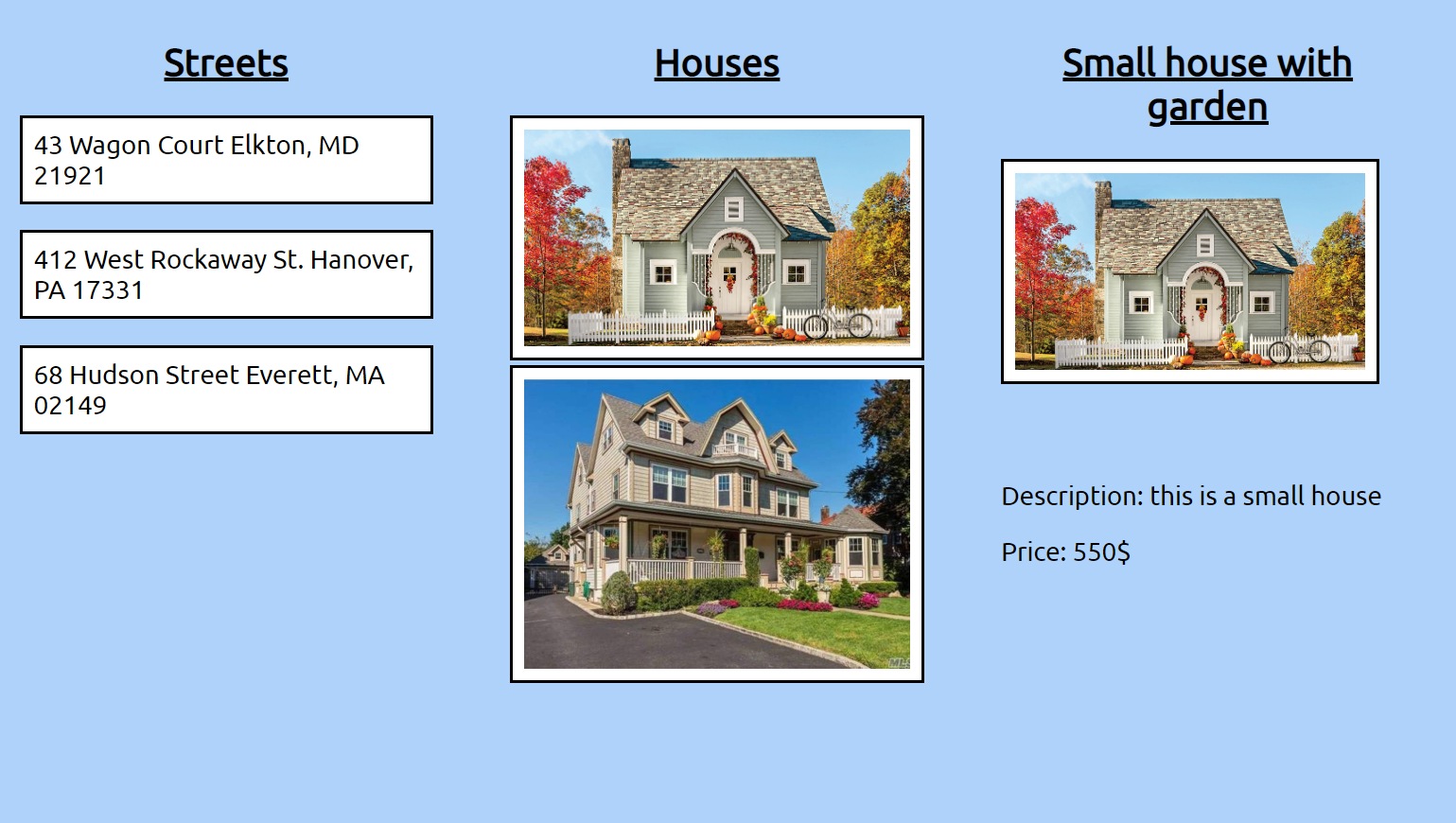
This function will return us the **currently selected house**

Now let us render it:

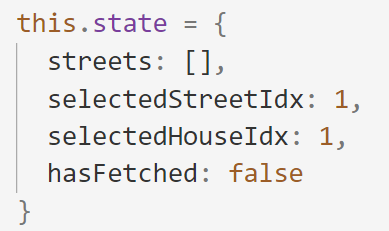
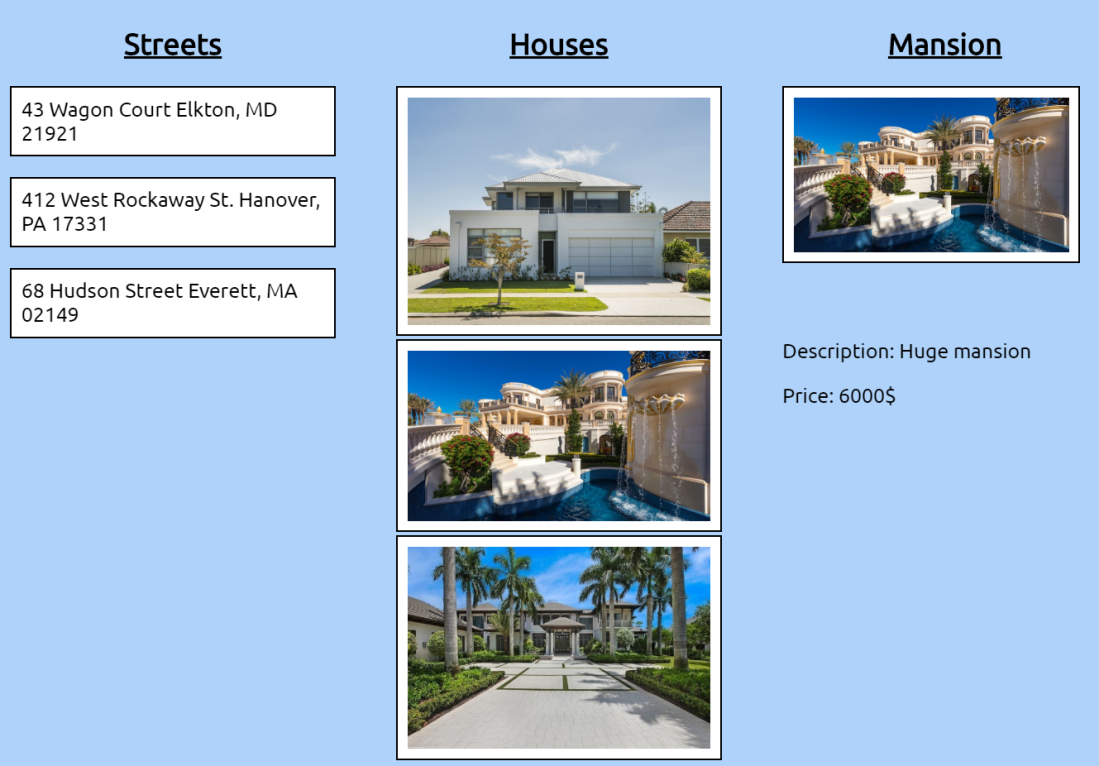


***Note:*** don't forget to import the **HouseDetails.js**

If you open the app now you should see something like this:



Try changing the indices in the state, you should see different results when doing that.



## On Hover Events

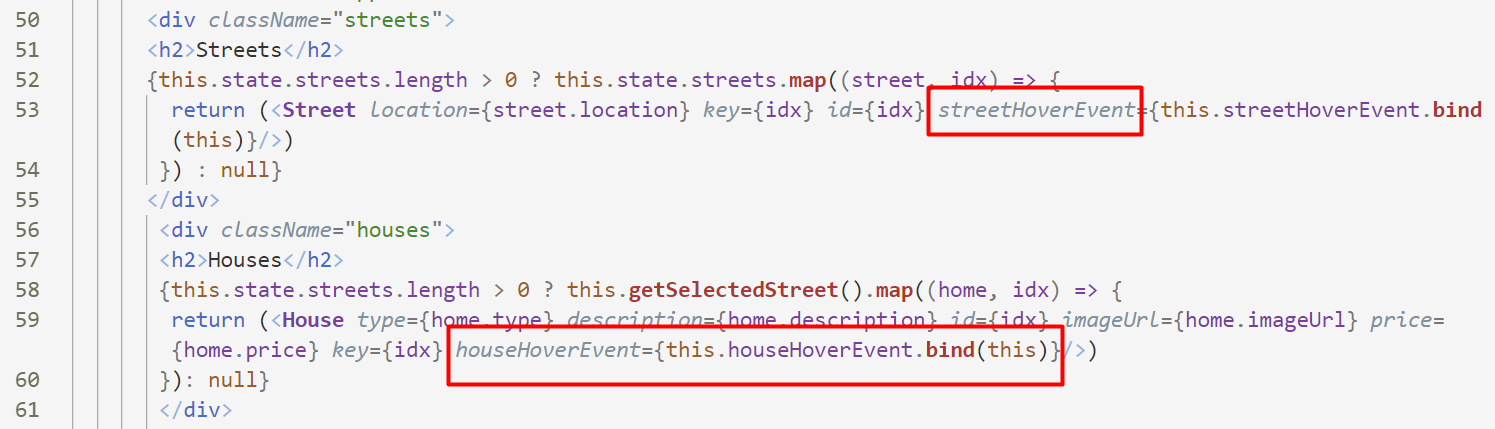
Of course, we don’t want to hardcode those indices; we want to **change them** when we **hover** with the mouse on a **street or a house**

First, add those two helper functions to the **App.js**:



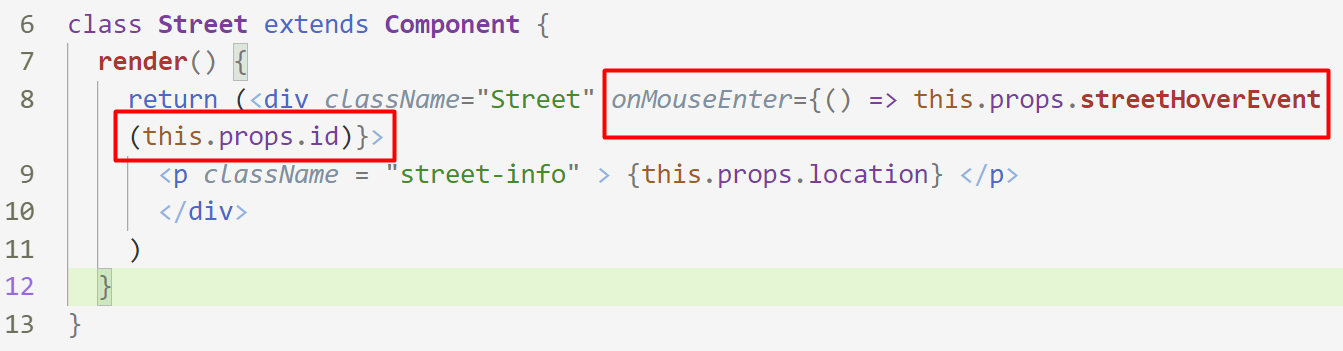
These functions will receive a new index and will update the state.

In order to do that however, we need to **pass the streetHoverEvent** function to each **Street** and the **houseHoverEvent** function to each **House** of the street. So in the render function of our app, add this:



We use **bind(this)**, so we can access the **this.state** of our **App** component

Now go to the **Street component** and add this:



Here we **call the function** passed from our App component **with the index of the current street** to **change it back in the App component**

**Do the same in the House component but call the houseHoverEvent instead!**

