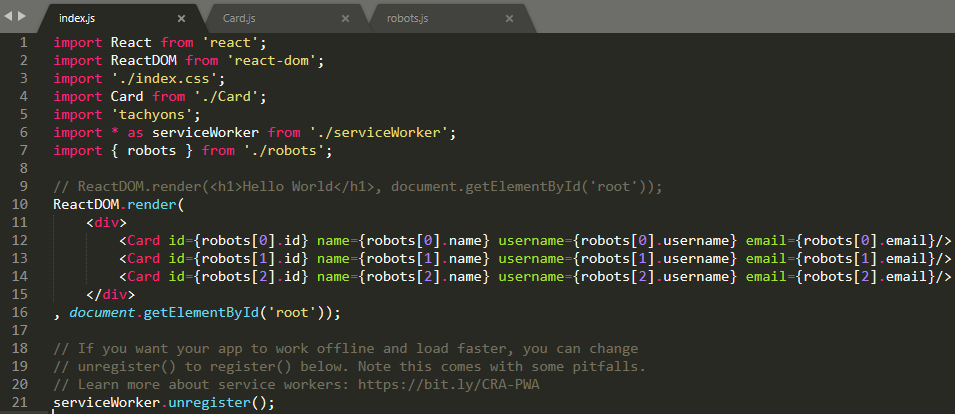
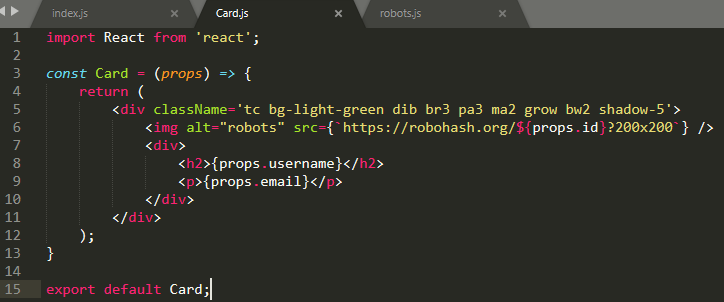
**ROBOFRIENDS (note that custom component should be started to capital letter)**

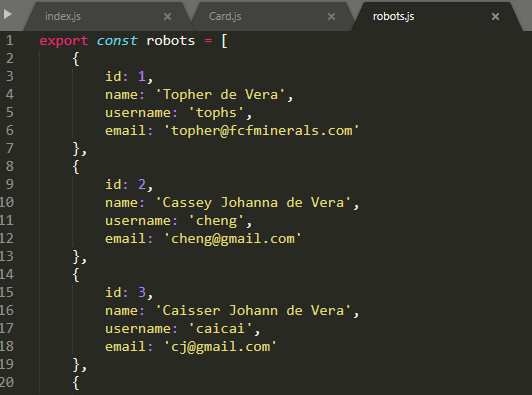
Index.js



Card.js



Robots.js

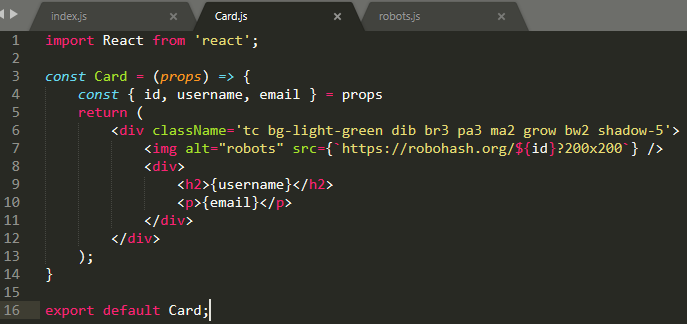


**EchmaSript 6**

Template String – can apply “ **{`https://robohash.org/${props.id}?200x200`}** ”, need to wrap it in a curly brackets

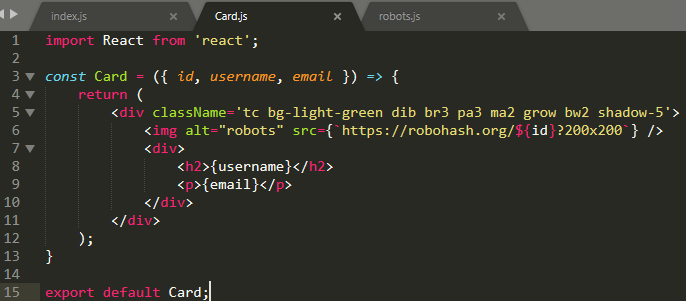


**Destructuring**

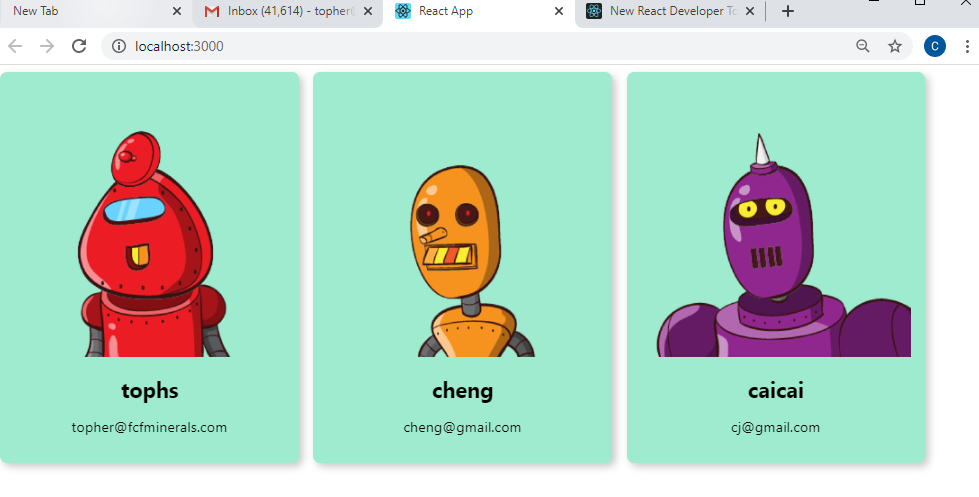


Observe the syntax become much cleaner compare on the first line above.

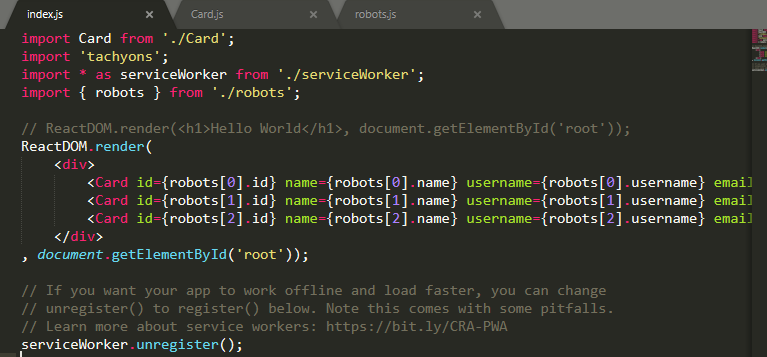
Or much cleaner is this one



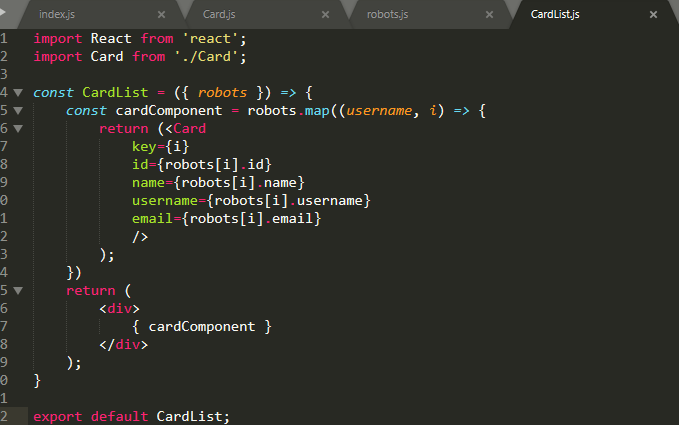
**Result**

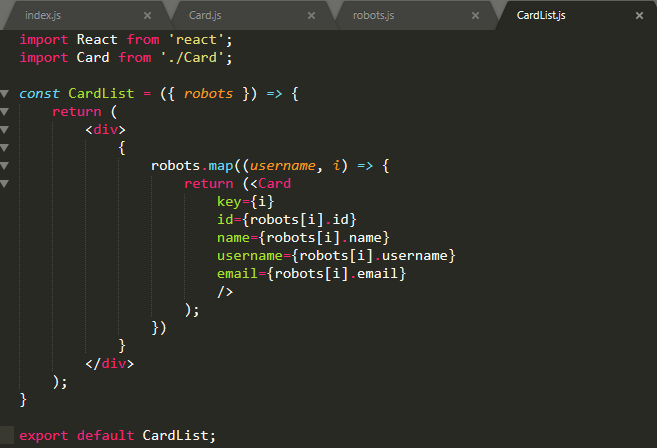


Base on the JSX syntax located on index.js, it show only 3 and need to view all 10 robofriends



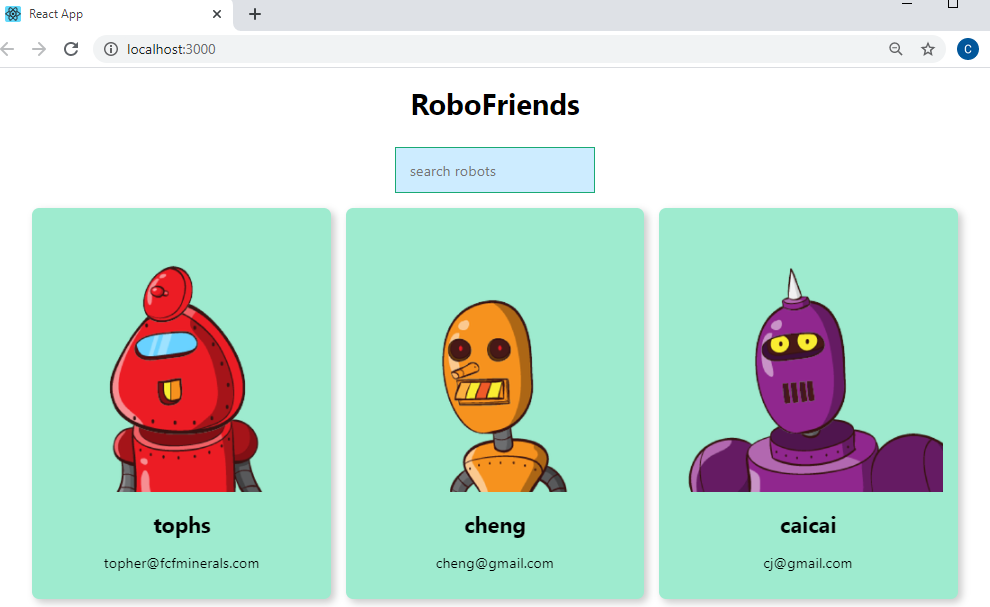
**Need a sort of looping**

* Need to import new component naming to “CardList”

Or

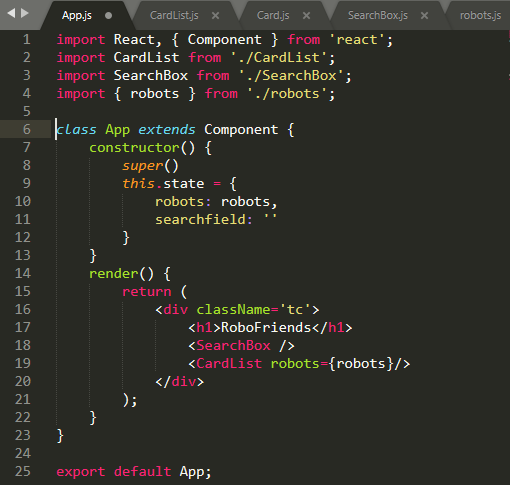
**STATE** describes the status of the entire program or an individual object. It could be text, a number, a boolean, or another data type. It's a common tool for coordinating code. For example, once you update **state**, a bunch of different functions can instantly react to that change.

**Props**, on the other hand, make components reusable by giving components the ability to receive data from their parent component in the form of **props**.

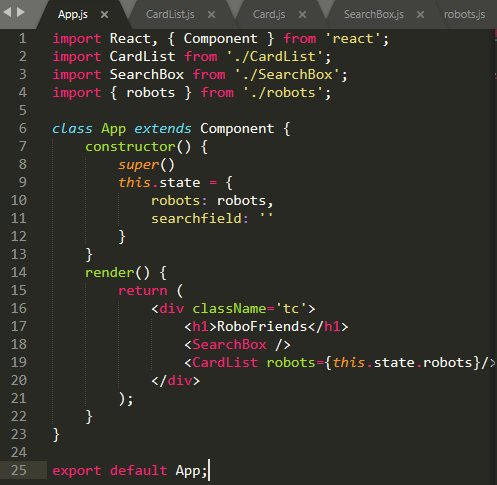


Now need that both “searchbox” and “cardlist” know each other, on their parents “App” it should have STATE declaration.

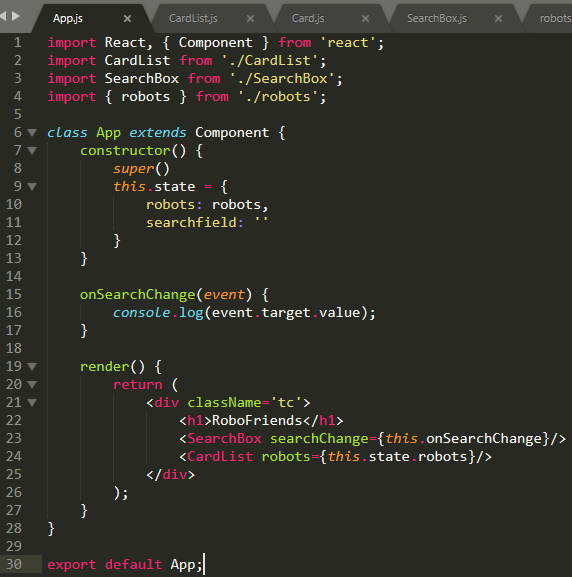
* First convert back the “App” to class syntax and create state to for both “robots and searchfield”

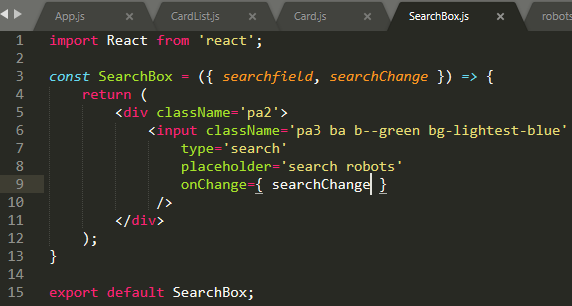


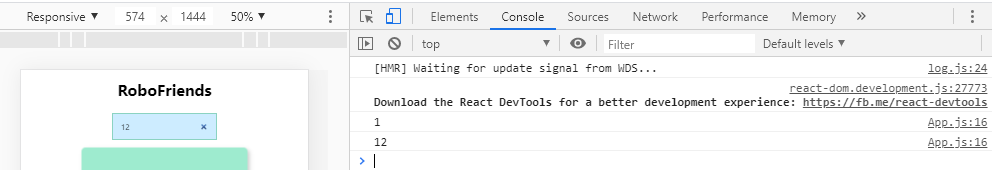
* Now under “App” component, we are not accessing robots components using the “import” instead we can use the “this.state” under class App, by this syntax we can now communicate “robots” using state

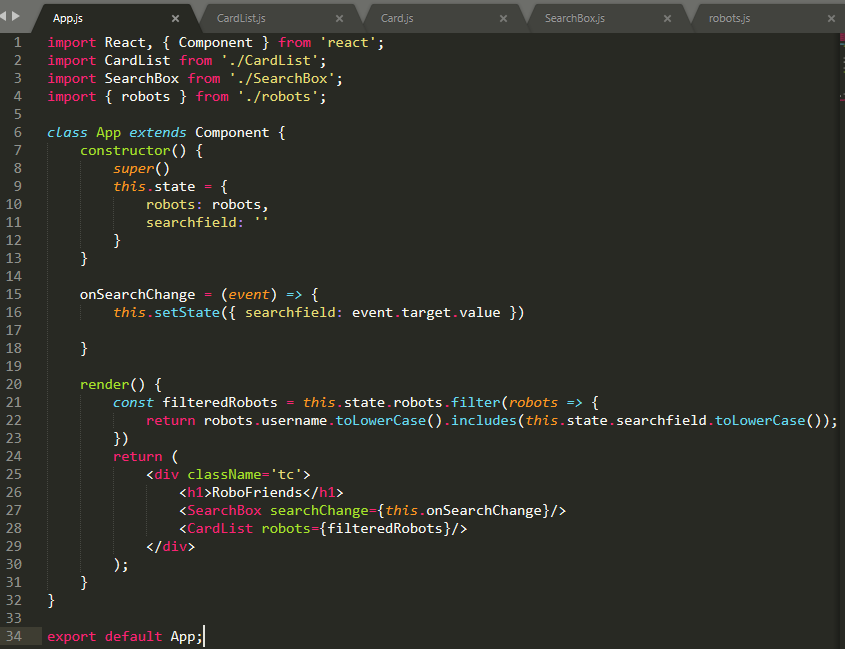


* For searchfield – we need that searchfield recognize all the input that we entered was captured thru console



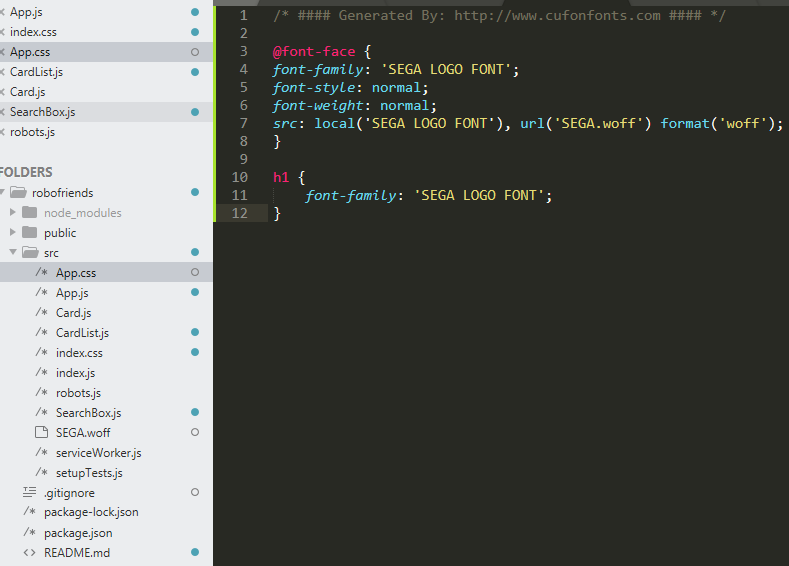






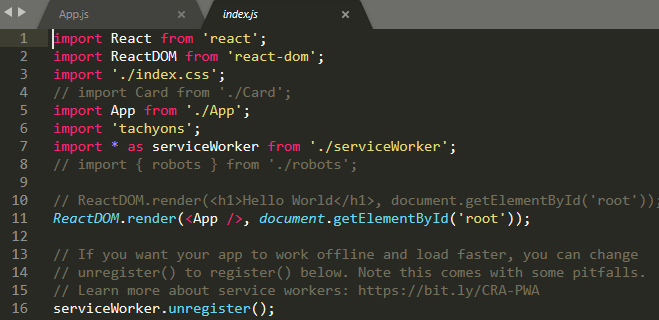
**Styling React App**

* Download SOGO Font from this link <https://www.cufonfonts.com/font/sega-logo-font>
* Add the .woff file into .src folder
* Copy the css syntax into App.css (create new if not exist)

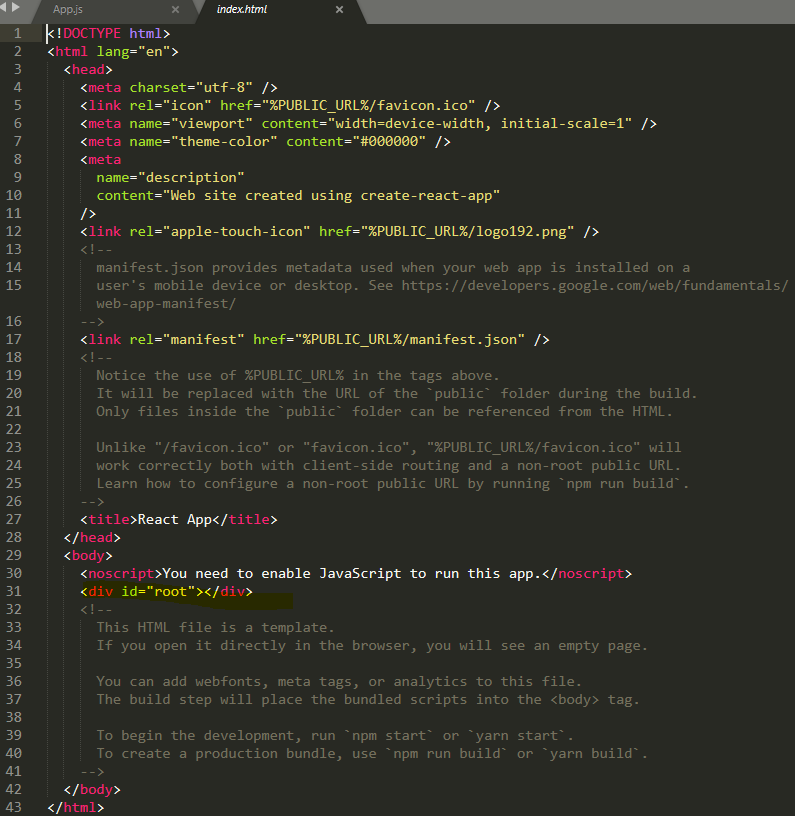


**Life Cycle Method**

* **Mounting -** These methods are called in the following order when an instance of a component is being created and inserted into the DOM:

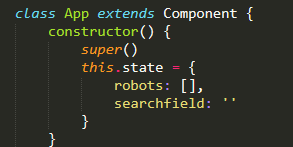


The <App /> will replace or be rendered on ‘root’ in index.html

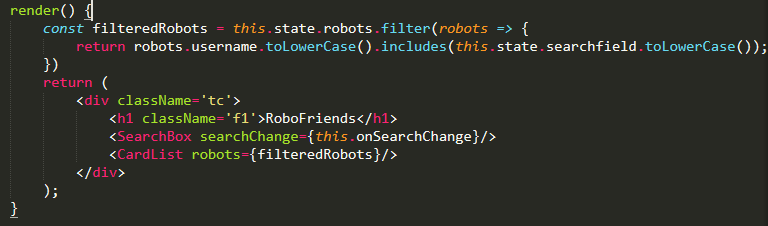


***Under mounting, here is the life cycle hooks***

1. constructor() – it will check the components if the constructor is existing



1. componentWillMount() – if doesn’t, ignore it
2. render()

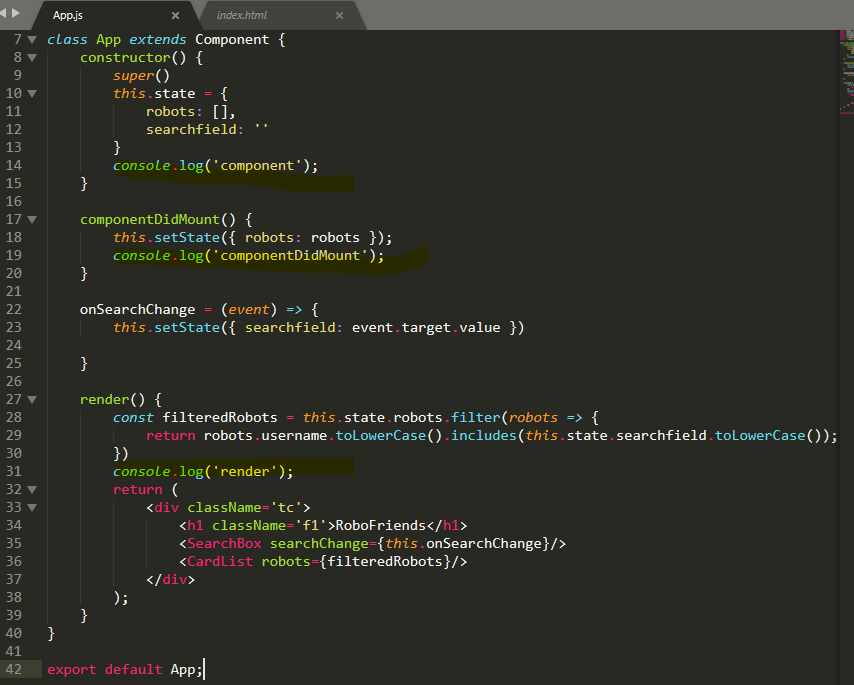


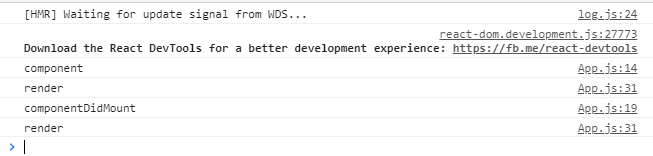
1. componentDidMount – if doesn’t, ignore it

* **Updating -** An update can be caused by changes to props or state. These methods are called in the following order when a component is being re-rendered:

1. Static getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

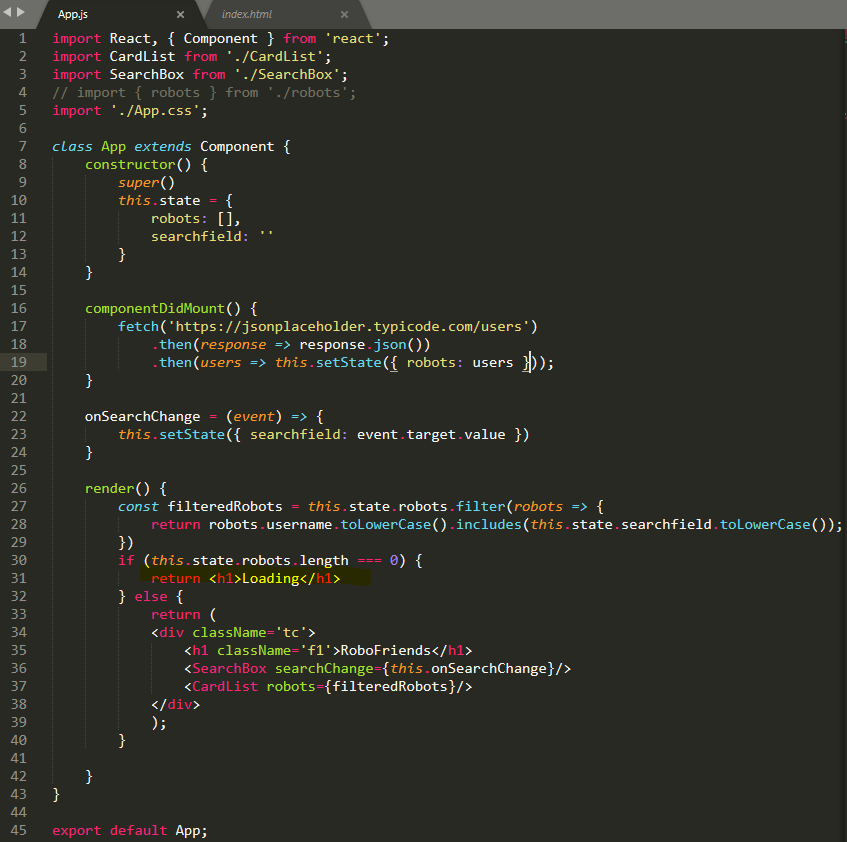
**To view the actual compilation of the App program**



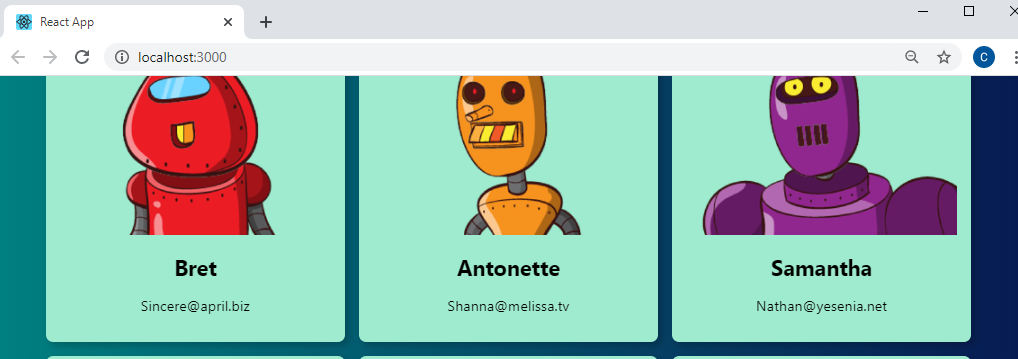


**To make it more realistic, we have to link to API – external link** [**https://jsonplaceholder.typicode.com/users**](https://jsonplaceholder.typicode.com/users) **to get more user profiles.**

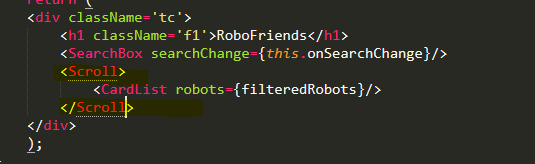
But what if the link takes time to response, make the App say something “LOADING”, this will happen once there is no user profile to load.



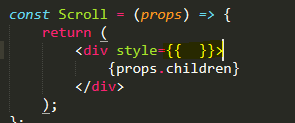
**Problem –** though the site is responsive yet but the searchbox and title were scrolled up.



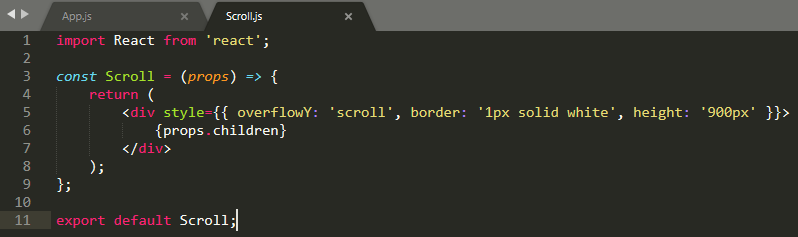
**Need to create a SCROLL component that will wrap the CardList component.**



**NOTE:** *double curly brackets, the first bracket is for javascript and the second which is inside is the STYLE/CSS properties*

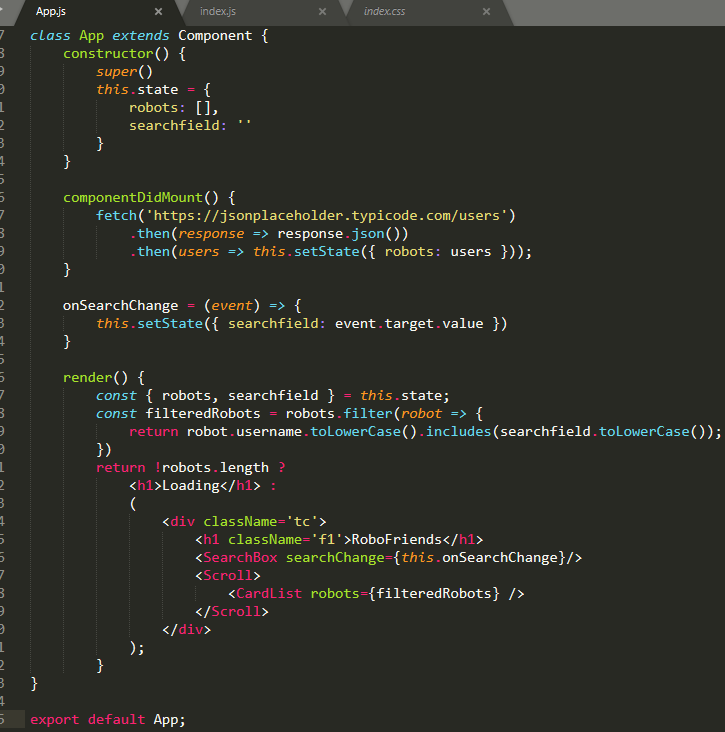


**The new added component now is SCROLL.js**

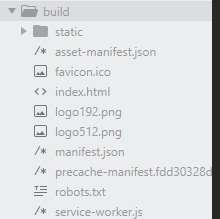


**Project Finalization**

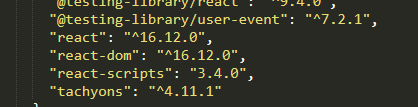
* Create folder naming “components and container”
* Components (Card.js, CardList.js, Scroll.js, SearchBox.js)
* Container (App.css, App.js, SEGA.woff)
* App.css syntax refructuring



* run command “npm run build” – this is the folder “build” that we need deploy on the website



* Ensure also that the version of REACT is up to date. "npm update” and followed by “npm audit fix –force



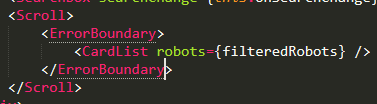
**Error Boundary in REACT**

Usually use to show the graceful view if there is an error with the created page or component

New command “componentDidCatch()”

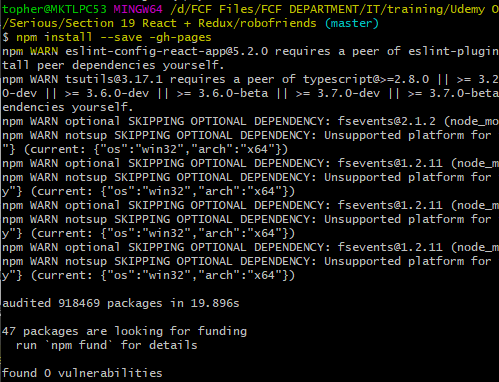


Just create a new component name “ErrorBoundary.js” and within this, we need to wrap the CardList

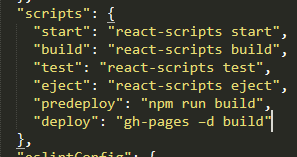


**Deploying Our REACT App on GIT HUB -** [**https://create-react-app.dev/docs/deployment/**](https://create-react-app.dev/docs/deployment/)

1. Create home page in package.json file ( "homepage": "https:/topherdevera.github.io/robofriends", )
2. Npm install –save gh-pages



1. Add scripts on package.json file (“predeploy”: “npm run build”, and “deploy”: gh-pages –d build”,)

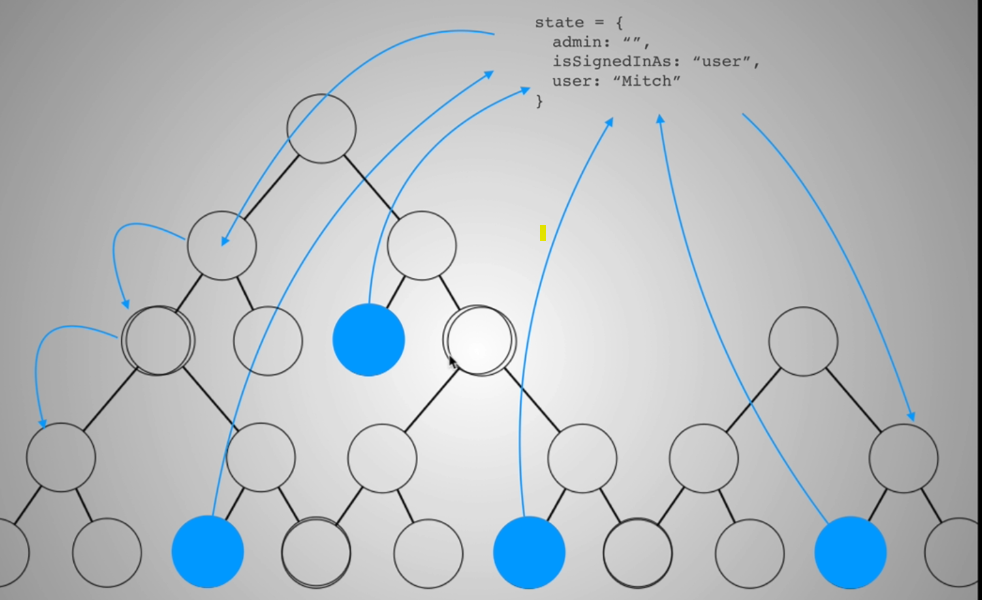


1. Run “nmp run deploy”

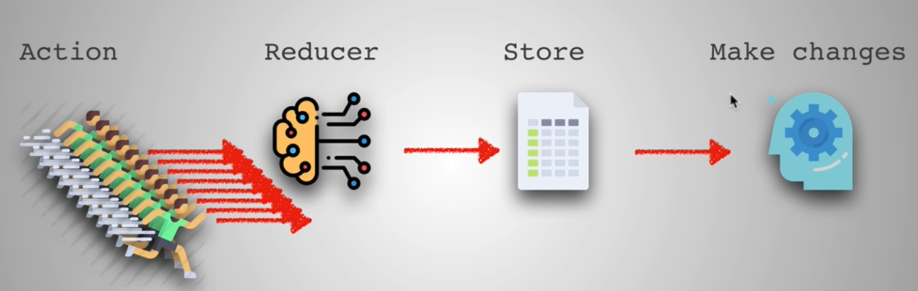
**REDUX –** state management, sharing data between containers.

Three (3) Principles of redux

1. Single source of truth
2. State is read only
3. Changes using pure functions

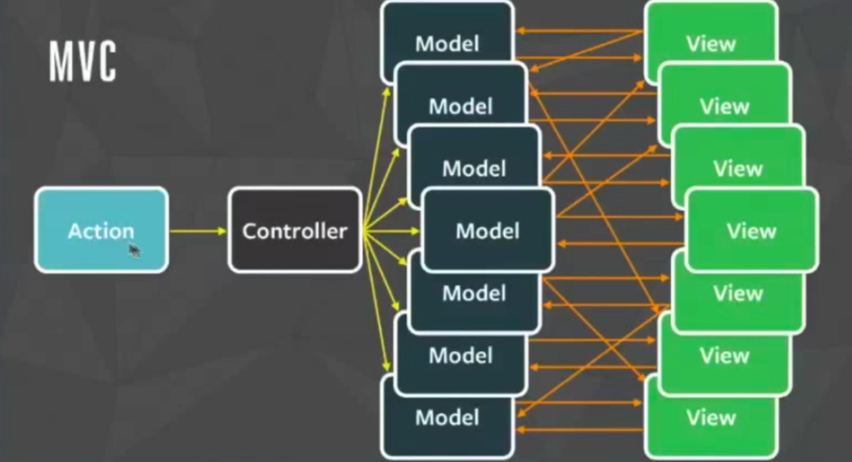




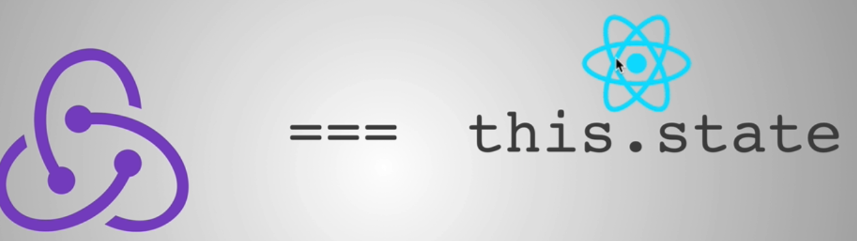


**REDUX** used an architectural pattern called “FLUX PATTERN”

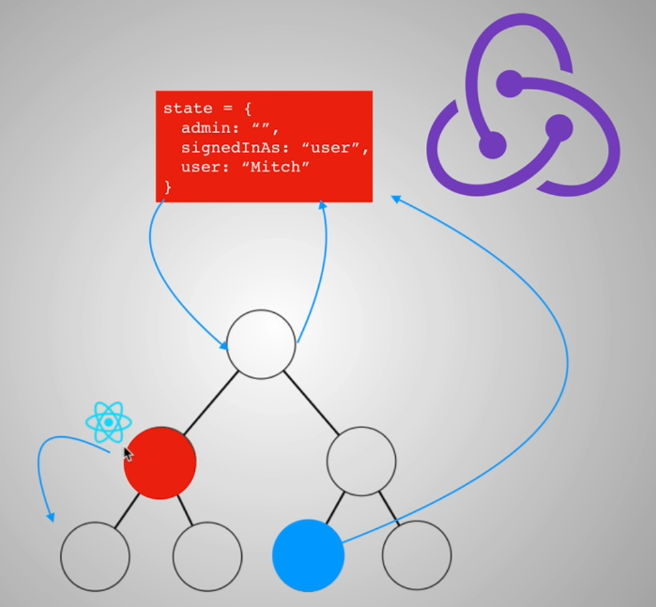
It as a once way data flow

**MVC –** Model View Controller

REDUX === this.state (React)



**REDUX Data flow diagram**



By this image, still need to use sometimes a REACT but mostly it is a REDUX

**REDUX Installation**

1. Before anything else, we need to run “**npm install**” making sure that NPM was installed properly
2. Run “**npm start**”
3. Run “**npm install redux**”



1. REACT is going to use REDUX, to run “**npm install react-redux”**

