

 [Harpreetkaur199125](#) / [react-intro-and-props](#) Privateforked from [cb-wd-18/react-intro-and-props](#)[Code](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) [Settings](#) [master](#) ▼

...

[react-intro-and-props](#) / [new](#) / [README.md](#)**RonyKordahi** stuff that didn't commit the first time? 1 contributor

Intro to React & Props (New Workshop)

Setup

This workshop uses **Create React App**, a project from Facebook designed to make it easy to get started building a React application.

It comes with a full build system, and very little boilerplate. We will learn more about this system in the future, but for now we'll focus on using it, not understanding how it works.

1. `cd new` This navigates your console inside the folder marked `new` so that your next commands affect only the files pertaining to this workshop.
2. `yarn install` This installs the third-party dependencies (like React). There are alot! It is normal for this to take some time.
3. `yarn start` This starts the development server.

It works similarly to `Live Server` - when you save a file, it should auto-restart, and the app should auto-refresh.

Understanding the files

The directory structure of this project is on the left, and looks like:

```
├── __lecture
├── new
│   ├── public
│   │   ├── assets
│   │   │   └── [images]
│   │   └── index.html
│   ├── src
│   │   ├── components
│   │   │   └── [component js and css]
│   │   └── data.js
```

104 lines (66 sloc) | 3.26 KB

...

```
├── package.json
└── original
```

`package.json` is a **manifest** of our project and its dependencies. We *don't need to edit it* for this workshop.

`public` contains static files, like our output `index.html` and some image assets. We *don't need to edit anything in here* either!

Finally, `src` is where all the good stuff lives.

To bootstrap this workshop, we've created several JS and CSS files:

```
├── components
│   ├── App.css
│   ├── App.js
│   ├── Header.css
│   ├── Header.js
│   ├── Menu.css
│   └── Menu.js
├── data.js
└── index.js
```

`data.js` includes all the information we need about our food items. This includes their names, price and a short description.

`App.js` is our top-level component, the very top of the tree. It imports the data, and will render the main chunks of our website.

`styles.css` includes *global* styles. You probably don't need to edit this file; all the other CSS you add should go in the other CSS files.

`components` holds our set of components, with 1 JS and 1 CSS file per component. If we want to style the Header component, we'd put those styles in `Header.css`.

Our initial state gives us a loose structure, but very little in the way of UI.

Your job will be to build this out, using the structure provided!

Exercises

1. Rendering some food

Open this exercise file: [exercise-1.md](#)

2. Creating a `MenuSection` component

Open this exercise file: [exercise-2.md](#)



3. Can never have too many components

Open this exercise file: [exercise-3.md](#)

 - Minimally complete workshop (75%) - 

4. Getting a blackbelt in `.map()`

Open this exercise file: [exercise-4.md](#)

 - Complete workshop (100%) - 

5. Styling it up!

No `exercise-5.md` file this time. Just add some style to it and make it look like an awesome menu!