1) npm i -g create-react-app

Это устанавливает скрипт, который создает boilerplate react приложения, чтобы не писать с нуля все

2) create-react-app tory-app

Создаем наше приложение. Там он тебе сразу после создания высветит команды:

npm start

Starts the development server.

npm run build

Bundles the app into static files for production.

npm test

Starts the test runner.

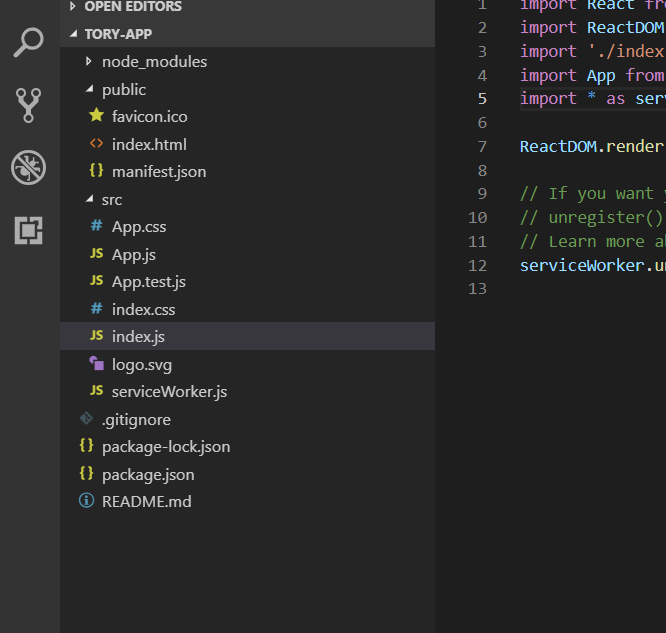
npm run eject

Removes this tool and copies build dependencies, configuration files

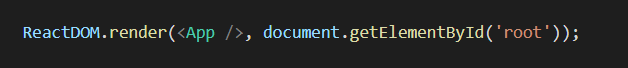
and scripts into the app directory. If you do this, you can’t go back!

Уже можно запускать с помощью npm start и посмотреть на приложение на http://localhost:3000/ Он запустит приложение на локальном сервере

Заходим в наше крутянское приложение, смотрим:

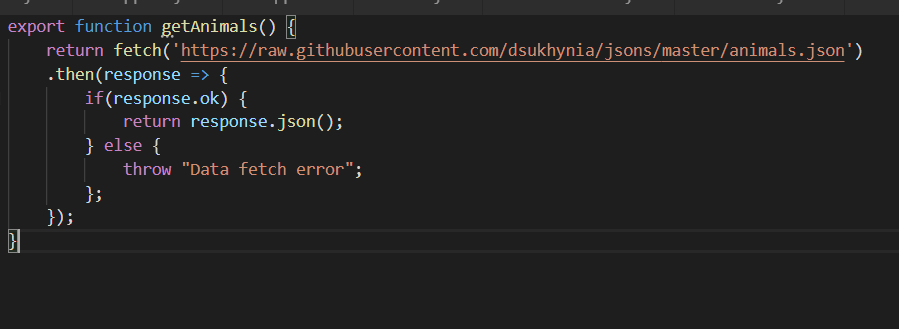


Файлы, на которые следует обратить внимание:

* Index.js – корневой JS файл. Он монтирует все наше приложени к определенному айдишнику корневого html:  
  
* Index.html – сам корневой html, к элементу root которого реакт цепляет динамически генерируемые элементы:  
  
* App.js – сам файл приложения, будем его модифицировать
* App.css – ну это поятно.

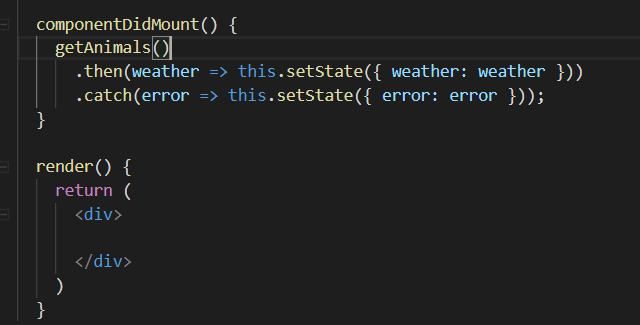
Там еще какой-то serviceWorker.js генерируется, хз что это вообще, он опциональный. Но удалять не будем.

Пишем код, который будет вытаскивать наши данные. Создаем файл services.js в /src



Тут такой синтаксис, потому что функция fetch возвращает Promise, это API для асинхронных вызовов. Читни доки если что, там немного.

Модифицируем App.js, сносим имеющийся код, добавляем код, который вытягивает данные. Это лучше всего делать, переопределив метод componendDidMount в компоненте. Это метод жизненного цикла, вызывается один раз, после того как компонент был «вмонтирован» в DOM



Тут мы вытащили наш json, и сохранили его в state компонента. В React у компонента есть state и props. State это изменяемое внутреннее состояние, props – неизменяемое внешнее, которое передается дочерним компонентам при создании. У нашего корневого компонента пропсов нет.

Пишем компонент для отображния таблицы. Создаем файл CoolTable.js:



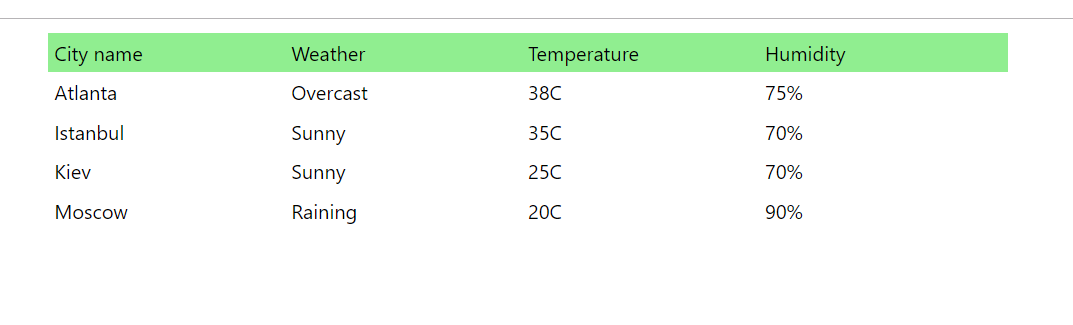
Как видишь, у него есть и пропсы, и стейт. Пропсы мы передаем из родительского компонента, это те самые данные о погоде, которые мы вытащили из ссылки. В стейте мы храним ключ сортировки, который используется в функции sortFunction чтобы отсортировать записи в таблице по заданному ключу. Можно покликать по цифрам с данным, и они будут сортироваться по данным в этой колонке.

Теперь импортируем этот компонент в наше приложение:



Если наш weather загрузился, то рендерим CoolTable, и передаем ему в пропсы заголовки для таблицы и сами данные.

Вот наша таблица с данными из сети:



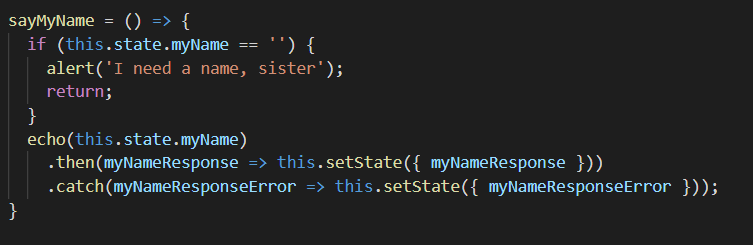
Теперь едем дальше, с формой отправки данных. Будем посылать данные на сервер postman-echo. Открываем наш файл services.js, добавляем в него новый вызов:



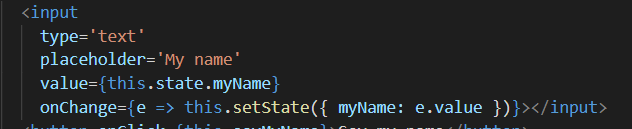
Теперь идем обратно в нашу аппликуху, добавляем див с кнопкой:



И метод, который будет использовать нашу echo-функцию:

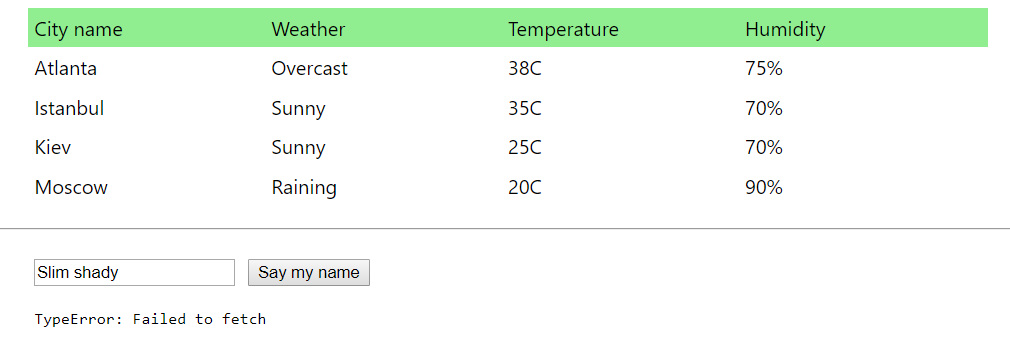


Обрати внимание на инпут, это характерно для форм контролов в реакте:



у него есть value для отображения текущего значения, которое ссылается на стейт компонента, и onChange, которое этот стейт апдейтит тем, что ввел пользователь. Таким образом из компонентов формы вытаскиваются вводимые данные.

Все вместе это выглядит так:



По посту у меня все время выбивает ошибку, которую ты видишь, пушо политика безопасности в хроме не позволяет отправлять данные на домены, отличные от того, с которого загружен сам документ. Это можно обойти либо хакнув хром, либо передав Access-Control-Allow-Origin: \* с сервера, но постман этого не умеет. Но в целом код правильый.