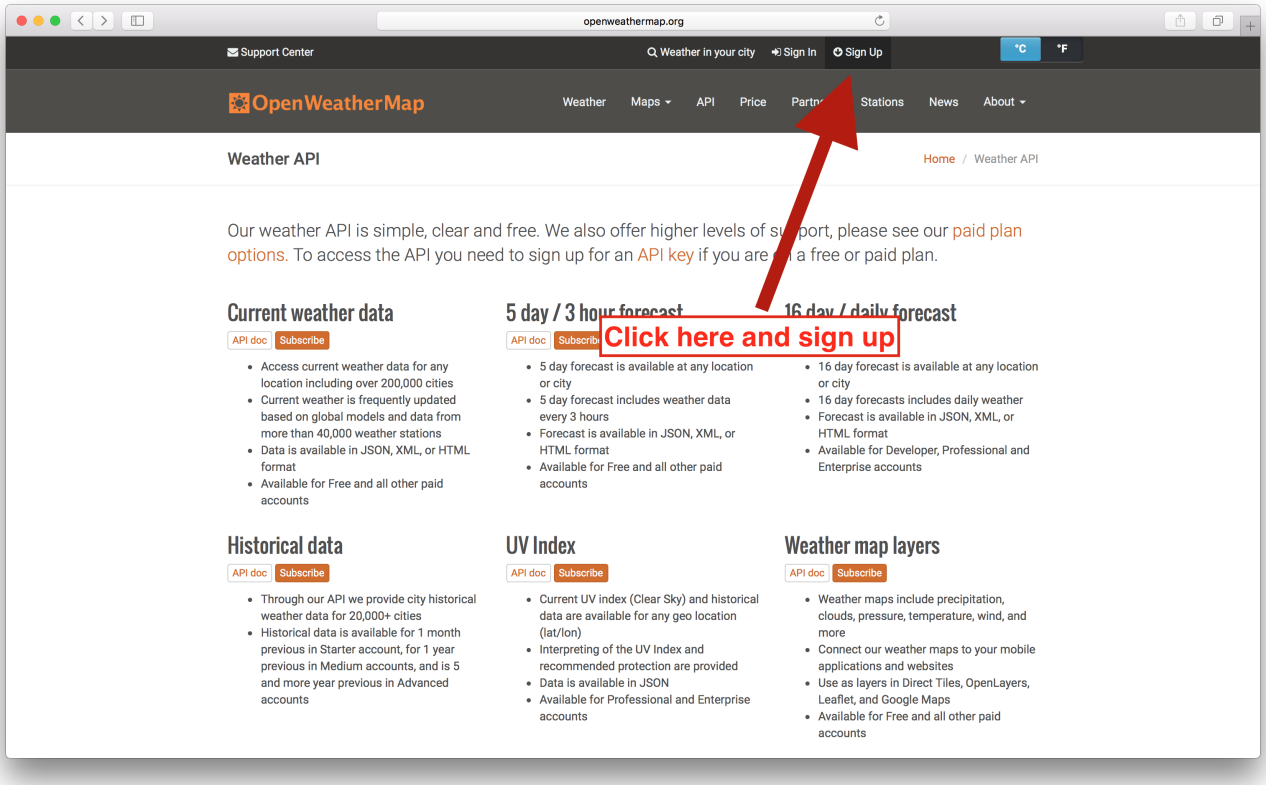
Fetching data

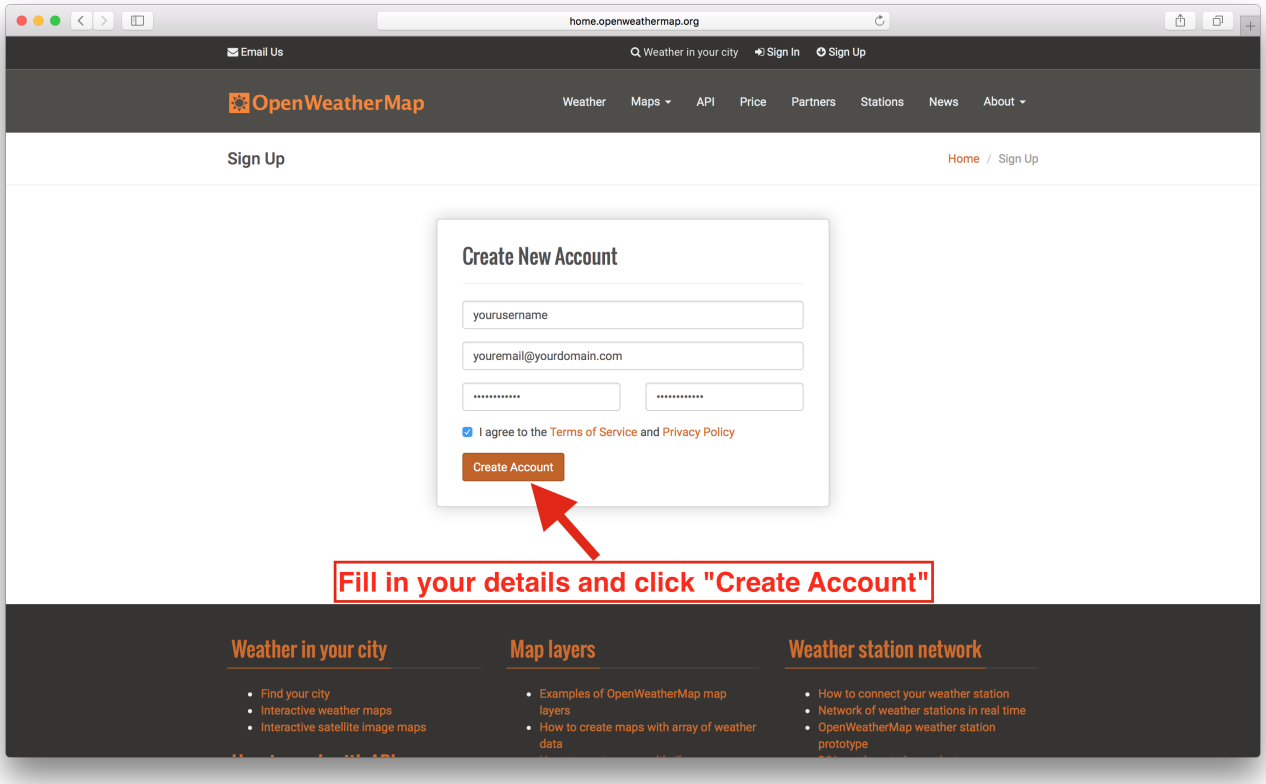
How are we going to fetch data for our weather app using React.js

WE'LL COVER THE FOLLOWING

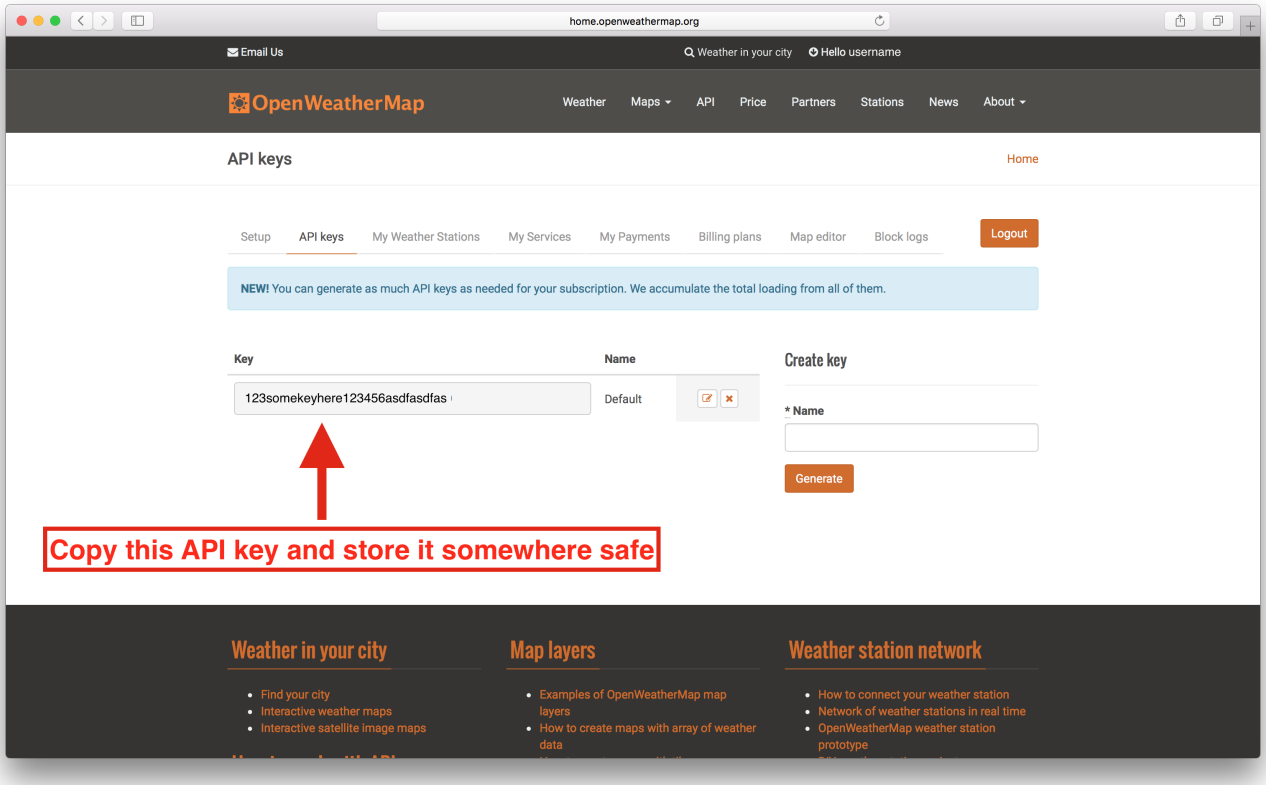
Setting OpenWeatherMap account

Let's get into fetching data. Instead of console logging a text, we need to get some weather information. We'll be using the OpenWeatherMap API for this task, which is a free service that provides access to data for basically all locations all around the world. You'll need to get an API key from it, so head over to openweathermap.org/api, press "Sign Up" in the top bar and register for a free account:





As soon as you've done that go to your API key page by going to home.openweathermap.org/api_keys, copy the `API key` from there and keep it somewhere safe.



Go ahead, signup. I'll wait for you here.

Now that we have access to all the weather data our heart could desire, let's get on with our app!

npm module: xhr

Inside our `fetchData` function, we'll have to make a request to the API. I like to use a npm module called `xhr` for this, a wrapper around the JavaScript XMLHttpRequest that makes said requests a lot easier

The general usage of `xhr` looks like this:

```
xhr({
  url: 'someURL'
}, function (err, data) {
  /* Called when the request is finished */
});
```

Parsing data from OpenWeatherMap

To get the weather forecast data, the structure of the URL we'll request looks like this:

```
http://api.openweathermap.org/data/2.5/forecast?q=CITY,COUNTRY&APPID=YOURAPIKEY&units=metric
```

Replace `CITY,COUNTRY` with a city and country combination of choice, replace `YOURAPIKEY` with your copied API key and open that URL in your browser. (it should look something like this: `http://api.openweathermap.org/data/2.5/forecast?q=Vienna,Austria&APPID=asdf123&units=metric`)

What you'll get is a JSON object that has the following structure:

```
"city": {
  "id": 2761369,
  "name": "Vienna",
  "coord": {
    "lon": 16.37208,
    "lat": 48.208488
  },
  "country": "AT",
  "population": 0,
  "sys": {
    "population": 0
  }
},
"cod": "200",
"message": 0.0046,
"cnt": 40,
"list": [ /* Hundreds of objects here */ ]
```

The top level `list` array contains time sorted weather data reaching forward 5 days. One of those weather objects looks like this: (only relevant lines shown)

```
{
  "dt": 1460235600,
  "main": {
    "temp": 6.94,
    "temp_min": 6.4,
    "temp_max": 6.94
  },
  "weather": [
    {
      "main": "Rain",
      /* …more data here */
```

```
  ],

  /* …more data here */
}
```

The five properties we care about are:

1. `dt_txt` - the time of the weather prediction.
2. `temp` - the expected temperature.
3. `temp_min` - the minimum expected temperature.
4. `temp_max` - the maximum expected temperature.
5. `weather[0].main` - a string description of the weather at that time.

OpenWeatherMap gives us a lot more data than that though, and I encourage you to snoop around a bit more and see what you could use to make the application more comprehensive!

Creating the Url that fetches weather data

As you can see, everything we really need to take care of is constructing the url and saving the returned data somewhere!

We know that the URL has a prefix that's always the same, `http://api.openweathermap.org/data/2.5/forecast?q=`, and a suffix that's always the same, `&APPID=YOURAPIKEY&units=metric`. The sole thing we need to do is insert the location the user entered into the URL!

> You can also change the units you get back by setting `units` to `imperial`: `http://api.openweathermap.org/data/2.5/forecast?q=something&APPID=YOURAPIKEY&units=imperial`

Now, if you're thinking this through you know what might happen – the user might enter spaces in the input! URLs with spaces aren't valid, so it wouldn't work and everything would break! While that is true, JavaScript gives us a very handy method to escape non-URL-friendly characters. It is called `encodeURIComponent()`, and this is how one uses it:

```
encodeURIComponent('My string with spaces'); // -> 'My%20string%20with%20spaces'
```

Show current temperature

We are going to modify our app to show the current temperature of the location that the user specifies.

Let's start writing some code

First, let's encode the location from the state and construct the URL that we need using that escaped location (*notice how we have a variable to assign the API key when the time comes*).

```
const API_KEY = "";
class App extends React.Component {
  state = { /* … */ };

  fetchData = (evt) => {
    evt.preventDefault();
    let location = encodeURIComponent(this.state.location);

    let location = encodeURIComponent(this.state.location);
    let urlPrefix = 'http://api.openweathermap.org/data/2.5/forecast?q=';
    let urlSuffix = '&APPID=' + API_KEY + '&units=metric';
    let url = urlPrefix + location + urlSuffix;
  };

  changeLocation = (evt) => { /* … */ };

  render() { /* … */ }
}
```

The last thing we need to do to get the data from the server by calling `fetch` with that url!

```
class App extends React.Component {
  fetchData = (evt) => {
        ...

    let url = urlPrefix + location + urlSuffix;

    fetch(url).then(r => r.json())
          .then(json => this.setState({
        data: json
        }))
        .catch(e => console.log("error"))
```

```
  };

  changeLocation = (evt) => { /* … */ };

  render() { /* … */ }
}
```

Now that we've got the weather data for the location we want in our component state, we can use it in our render method! Remember, the data for the current weather is in the `list` array, sorted by time. The first element of said array is thus the current temperature, so let's try to render that first. Here's how our render method will look:

```
render() {
    let currentTemp = 'Specify a location';
    if (this.state.data.list) {
      currentTemp = this.state.data.list[0].main.temp;
    }
    return (
      <div>
        <h1>Weather</h1>
        <form onSubmit={this.fetchData}>
          <label>I want to know the weather for
            <input
              placeholder={"City, Country"}
              type="text"
              value={this.state.location}
              onChange={this.changeLocation}
            />
          </label>
        </form>
        <p className="temp-wrapper">
          <span className="temp">{ currentTemp }</span>
          <span className="temp-symbol">°C</span>
        </p>
      </div>
    );
  }
```

Now it's time to use your `OpenWeatherMAP API key`. Assign your key to the variable API_KEY in the example below and then enter a location and you'll see the current temperature! 🎉 *Awesome!*).

```
import React from 'react';
import './App.css';
import xhr from 'xhr';


const API_KEY = "82f40c24bce69950c7aa3d09e07b391b";

class App extends React.Component {
  state = {
      location: '',
      data: {}
  };

  fetchData = (evt) => {
    evt.preventDefault();

    if (!API_KEY) {
      console.log('Enter your API_KEY and the enter location');
      return;
    }

    let location = encodeURIComponent(this.state.location);
    let urlPrefix = '/cors/http://api.openweathermap.org/data/2.5/forecast?q=';
    let urlSuffix = '&APPID=' + API_KEY + '&units=metric';
    let url = urlPrefix + location + urlSuffix;

    xhr({
      url: url
    }, (err, data) => {
      if (err) {
        console.log('Error:', err);
        return;
      }

      this.setState({
        data: JSON.parse(data.body)
      });
    });
  };

  changeLocation = (evt) => {
    this.setState({
      location: evt.target.value
    });
  };

  render() {
    let currentTemp = 'Specify a location';
    if (this.state.data.list) {
      currentTemp = this.state.data.list[0].main.temp;
    }
    return (
      <div>
        <h1>Weather</h1>
        <form onSubmit={this.fetchData}>
          <label>I want to know the weather for
            <input
              placeholder={"City, Country"}
              type="text"
              value={this.state.location}
              onChange={this.changeLocation}
            />
          </label>
```

```
        </form>
        <p className="temp-wrapper">
          <span className="temp">{ currentTemp }</span>
          <span className="temp-symbol">°C</span>
        </p>
      </div>
    );
  }
}

export default App;
```

Summary

We learned how to structure our application and then we created a controlled text input and used that to fetch our first data using `fetch`!

Now that we have the current temperature, we need to render the 5 day forecast! Continue with Chapter 3: Adding a Forecast Graph where we'll learn how to draw a graph!

Additional Material

- Official create-react-app docs
- What APIs Are And Why They're Important