Lesson Weekend

Intermediate JavaScript (/intermediate-javascript) / Asynchrony and APIs (/intermediate-javascript/asynchrony-and-apis) / Protecting API Keys

Text

In the real world, API keys shouldn't be stored in client-side JavaScript files. This simply isn't secure, especially if the API has a rate limit, charges for use, or provides access to sensitive information. If your API key is accessible directly in the source code, anyone can access it, including bots!

Thankfully, we can easily conceal API keys in our applications, and that's just what we'll learn how to do in this lesson.

It's worth noting that we'll be learning a method of protecting and managing API keys for development. In production, your JavaScript application would typically make a request to your own web server, which safely stores your API key; then your web server would make the appropriate request to the API, and then handle delivering the response to your application.

Protecting API Keys

When we write code in development, we can store our API key in a `.env` file stored on our local machines. `.env` is a place where we can store any **environmental variables** we want to keep secret (such as an API key). These are variables that are specific to your

environment and shouldn't be shared with others. `.env` is a common file extension for a configuration file used to set variables containing sensitive information. In fact, the file is a convention used across many frameworks including Node and Rails.

Once we have a `.env` file, we need to add it to our `.gitignore`. This way, we don't accidentally push sensitive information to GitHub.

Anyone who wants to clone and use our application can simply create their own local `.env` file with their own API key. Because of this, **you are required to include instructions for getting and setting up an API key in your README for this section's independent project.** A README should have all instructions for setting up a project, so the reason for this requirement should be clear.

Adding `.env` to `.gitignore`

Before using an API key in our application or creating a `.env` file, we need to make we're ignoring it. This should always be your first step.

Let's add `.env` to our `.gitignore` file now. At this point, your `.gitignore` file should look something like this:

.gitignore

```
node_modules/
dist/
coverage/
.env
```

Creating a `.env` File

Next, let's create a `.env` file in the root directory of our project where we can store our sensitive variables:

.env

```
API_KEY=3d68a17ddd5ea407ab91f2d278342017
```

Note that the API key above is a fake one — you will need to replace it with your own. It's *not* a string — so don't use quotes or backticks!

The variable that we're storing the key in is `API_KEY`. Note that the naming convention here for variables is all uppercase with underscores between each words.

We can add as many variables as we want (such as if we are working on a project that uses multiple APIs and keys). While we are only protecting API keys, there are many other sensitive variables not related to APIs we might potentially want to store in the future.

Configuring webpack

Next, we'll use a webpack plugin called `dotenv-webpack` to make our environmental variables available inside our application:

```
$ npm install dotenv-webpack@2.0.0 --save-dev
```

Because it's a plugin, we need to first `require` it and then add it to the `plugins` array in `webpack.config.js`:

webpack.config.js

```
...
const Dotenv = require('dotenv-webpack');

module.exports = {
  ...
  plugins: [
    ...
    new Dotenv()
  ],
  ...
  }
};
```

Accessing Environmental Variables

To access environmental variables in our application, we need to preface the environmental variable with `process.env`. Here's an example using our API_KEY from above:

```
process.env.API_KEY
```

Note that it's necessary to use a template literal so the key is expressed as a string within our API call. (Remember, it's not a string in our `.env` file.) For that reason, it's added to our code like this: `${process.env.API_KEY}`. Here's how it should look if we add it to our API call from the last lesson:

src/index.js

```
function getWeather () {
  ...
  const url = `http://api.openweathermap.org/data/2.5/weath
er?q=${city}&appid=${process.env.API_KEY}`
  ...
}
```

Summary

You should *always* be careful storing any sensitive information in your application. Even if it's just the key for an API that's free for public use, you should always be in the habit of keeping this information private. This means storing all keys in a local `.env` file that's *never* pushed to GitHub.

If you create GitHub templates for projects that use webpack, make sure that `dotenv-webpack` is added to your `package.json` and `.env` is added to your `.gitignore`.

Once again, when you push projects to GitHub that require API keys, make sure to include instructions in the README so that other users can clone your project and use their own API keys to see what you've built. That means including *all* steps for getting a key — from the link to sign up for an account to any steps to getting an API key to the name of the API key variables that should be added to the `.env` file. Including accurate instructions for getting and setting up an API key is a requirement for this section's independent project.

Up next, we'll add error handling to our OpenWeather API app.

Previous (/intermediate-javascript/asynchrony-and-apis/making-api-calls-with-javascript)
Next (/intermediate-javascript/asynchrony-and-apis/exception-handling-for-api-calls)

disable dark mode

(http://www.epicodus.com)