Is it possible to use dotenv in a react project?

I am trying to set some environment variables (for making API calls to dev/prod endpoints, keys depending on dev/prod, etc.) and I'm wondering if using dotenv will work.

I've installed dotenv, and I am using webpack.

My webpack entry is `main.js` , so in this file I've put `require('dotenv').config()`

Then, in my webpack config, I've put this:

```
new webpack.EnvironmentPlugin([
  'NODE_ENV',
  '__DEV_BASE_URL__'  //base url for dev api endpoints
])
```

However, it is still undefined. How can I do this correctly?

javascript reactjs webpack

asked Feb 11 '17 at 23:44
user1354934
1,552 4 16 31

1 Answer

The short answer is no. A browser cannot access local or server environment variables so dotenv has nothing to look for. Instead, you specify ordinary variables in your React application, usually in a settings module.

Webpack can be made to take environment variables from the build machine and bake them into your settings files. However, it works be actually replacing strings at build-time, not run-time. So each build of your application will have the values hard-coded into it. These values would then be accessible through the `process.env` object.

```
var nodeEnv = process.env.NODE_ENV;
```

Additionally, you could use the `DefinePlugin` for webpack which lets you explicitly specify different values depending on your build target (dev, prod, etc.). Note that you have to `JSON.stringify` all values passed into it.

```
new webpack.DefinePlugin({
    'process.env.NODE_ENV': JSON.stringify(process.env.NODE_ENV || 'development')
}),
```

This is fine for any sort of public details but should **never be used for any sort of private keys, passwords or API secrets**. This is because any values baked in are publicly accessible and could be used maliciously if they contain sensitive details. For those sorts of things, you need to write some server-side code and build a simple API which can authenticate with the 3rd party API using the secrets, then pass the relevant details along to your client-side application. Your server-side API acts as an intermediary, protecting your secrets while still getting the data you need.

edited Feb 11 '17 at 23:57

answered Feb 11 '17 at 23:53
Soviut
50.8k 30 129 196

1 Thanks. How can I do this for stuff like API keys? For example, google maps API key. I mean, that one is fine to expose I guess since only one domain origin is whitelisted. But just curious anyway. TY! – user1354934 Feb 11 '17 at 23:56

I've added an explanation of how to do this in my answer. The short answer is write a server-side application your client-app can talk to and do all the private/secret stuff in there. – Soviut Feb 12 '17 at 0:01

Thanks. Sorry if this sounds dumb, but do you mean something like what I have done right now - my back end (express app) has an /api-auth endpoint which sends over a JWT which is then stored in LS. I use that if its valid (otherwise refresh it) so that the client need to worry about getting a valid token to access any api response/data. – user1354934 Feb 12 '17 at 0:05

Sort of. You authenticate your user, but you also can use your API to proxy to other APIs. So all API requests, even ones to Google API, could be proxies through your server. The client would only care about your API and whether it was authenticated with it. – Soviut Feb 12 '17 at 0:09