# dotenv dotenv-webpack

A secure webpack plugin that supports dotenv and other environment variables and \*\*only exposes what you choose and use\*\*.

     

## Installation Include the package locally in your repository. `npm install dotenv-webpack --save-dev` ## Description `dotenv-webpack` wraps `dotenv` and `Webpack.DefinePlugin`. As such, it does a text replace in the resulting bundle for any instances of `process.env`. Your `.env` files can include sensitive information. Because of this,`dotenv-webpack` will only expose environment variables that are \*\*explicitly referenced in your code\*\* to your final bundle. Interested in taking your environments to the next level? Check out the [Dotenv Organization](https://www.dotenv.org/get-started?r=3). ## Usage The plugin can be installed with little-to-no configuration needed. Once installed, you can access the variables within your code using `process.env` as you would with `dotenv`. The example bellow shows a standard use-case. ### Create a .env file ```dosini // .env DB\_HOST=127.0.0.1 DB\_PASS=foobar S3\_API=mysecretkey ``` ### Add it to your Webpack config file ```javascript // webpack.config.js const Dotenv = require('dotenv-webpack'); module.exports = { ... plugins: [ new Dotenv() ] ... }; ``` ### Use in your code ```javascript // file1.js console.log(process.env.DB\_HOST); // '127.0.0.1' ``` ### Resulting bundle ```javascript // bundle.js console.log('127.0.0.1'); ``` Note: the `.env` values for `DB\_PASS` and `S3\_API` are \*\*NOT\*\* present in our bundle, as they were never referenced (as `process.env.[VAR\_NAME]`) in the code. ## How Secure? By allowing you to define exactly where you are loading environment variables from and bundling only variables in your project that are explicitly referenced in your code, you can be sure that only what you need is included and you do not accidentally leak anything sensitive. ### Recommended Add `.env` to your `.gitignore` file ## Limitations Due to the fact that we use `webpack.DefinePlugin` under the hood, we cannot support destructing as that breaks how this plugin is meant to be used. Because of this, please reference your variables without destructing. For more information about this, please review the issue [here](https://github.com/mrsteele/dotenv-webpack/issues/70). ## `process.env` stubbing / replacing `process.env` is not polyfilled in Webpack 5+, leading to errors in environments where `process` is `null` (browsers). We automatically replace any remaining `process.env`s in these environments with `"MISSING\_ENV\_VAR"` to avoid these errors. When the `prefix` option is set, `process.env`s will not be stubbed. If you are running into issues where you or another package you use interfaces with `process.env`, it might be best to set `ignoreStub: true` and make sure you always reference variables that exist within your code (See [this issue](https://github.com/mrsteele/dotenv-webpack/issues/271) for more information). ## Properties Use the following properties to configure your instance. \* \*\*path\*\* (`'./.env'`) - The path to your environment variables. This same path applies to the `.env.example` and `.env.defaults` files. [Read more here](#about-path-settings). \* \*\*safe\*\* (`false`) - If true, load '.env.example' to verify the '.env' variables are all set. Can also be a string to a different file. \* \*\*allowEmptyValues\*\* (`false`) - Whether to allow empty strings in safe mode. If false, will throw an error if any env variables are empty (but only if safe mode is enabled). \* \*\*systemvars\*\* (`false`) - Set to true if you would rather load all system variables as well (useful for CI purposes). \* \*\*silent\*\* (`false`) - If true, all warnings will be suppressed. \* \*\*expand\*\* (`false`) - Allows your variables to be "expanded" for reusability within your `.env` file. \* \*\*defaults\*\* (`false`) - Adds support for `dotenv-defaults`. If set to `true`, uses `./.env.defaults`. If a string, uses that location for a defaults file. Read more at [npm](https://www.npmjs.com/package/dotenv-defaults). \* \*\*ignoreStub\*\* (`false`) - Override the automatic check whether to stub `process.env`. [Read more here](#user-content-processenv-stubbing--replacing). \* \*\*prefix\*\* (`'process.env.'`) - The prefix to use before the name of your env variables. The following example shows how to set any/all arguments. ```javascript module.exports = { ... plugins: [ new Dotenv({ path: './some.other.env', // load this now instead of the ones in '.env' safe: true, // load '.env.example' to verify the '.env' variables are all set. Can also be a string to a different file. allowEmptyValues: true, // allow empty variables (e.g. `FOO=`) (treat it as empty string, rather than missing) systemvars: true, // load all the predefined 'process.env' variables which will trump anything local per dotenv specs. silent: true, // hide any errors defaults: false, // load '.env.defaults' as the default values if empty. prefix: 'import.meta.env.' // reference your env variables as 'import.meta.env.ENV\_VAR'. }) ] ... }; ``` ## About `path` settings As previously mentioned, it is possible to customize the `path` where the `.env` file is located as well as its \*filename\* from the plugin settings: ```javascript module.exports = { ... plugins: [ new Dotenv({ path: './some.other.env', }) ] ... }; ``` It is important to mention that this same path and filename will be used for the location of the `.env.example` and `.env.defaults` files if they are configured, this will only add the `.example` and `.defaults` suffixes respectively: ```javascript module.exports = { ... plugins: [ new Dotenv({ path: '../../path/to/other.env', safe: true, // load '../../path/to/other.env.example' defaults: true, // load '../../path/to/other.env.defaults' }) ] ... }; ``` This is especially useful when working with [Monorepos](https://monorepo.tools/) where the same configuration can be shared within all sub-packages of the repository: ```bash . ├── packages/ │ ├── app/ │ │ └── webpack.config.js # { path: '../../.env' } │ └── libs/ │ └── webpack.config.js # { path: '../../.env' } ├── .env ├── .env.example └── .env.defaults ``` ## LICENSE MIT