> *Note: this feature is available with `react-scripts@0.2.3` and higher.*

Your project can consume variables declared in your environment as if they were declared locally in your JS files. By default you will have `NODE_ENV` defined for you, and any other environment variables starting with `REACT_APP_`.

> *WARNING: Do not store any secrets (such as private API keys) in your React app!*
>
> *Environment variables are embedded into the build, meaning anyone can view them by inspecting your app's files.*

The environment variables are embedded during the build time. Since Create React App produces a static HTML/CSS/JS bundle, it can't possibly read them at runtime. To read them at runtime, you would need to load HTML into memory on the server and replace placeholders in runtime, as described here. Alternatively you can rebuild the app on the server anytime you change them.

> *Note: You must create custom environment variables beginning with `REACT_APP_`. Any other variables except `NODE_ENV` will be ignored to avoid accidentally exposing a private key on the machine that could have the same name. Changing any environment variables will require you to restart the development server if it is running.*

These environment variables will be defined for you on `process.env`. For example, having an environment variable named `REACT_APP_NOT_SECRET_CODE` will be exposed in your JS as `process.env.REACT_APP_NOT_SECRET_CODE`.

There is also a built-in environment variable called `NODE_ENV`. You can read it from `process.env.NODE_ENV`. When you run `npm start`, it is always equal to `'development'`, when you run `npm test` it is always equal to `'test'`, and when you run `npm run build` to make a production bundle, it is always equal to `'production'`. **You cannot override `NODE_ENV` manually.** This prevents developers from accidentally deploying a slow development build to production.

These environment variables can be useful for displaying information conditionally based on where the project is deployed or consuming sensitive data that lives outside of version control.

First, you need to have environment variables defined. For example, let's say you wanted to consume an environment variable inside a `<form>`:

```
render() {
  return (
    <div>
      <small>You are running this application in <b>{process.env.NODE_ENV}</b> mode.
</small>
      <form>
        <input type="hidden" defaultValue={process.env.REACT_APP_NOT_SECRET_CODE} />
      </form>
    </div>
  );
}
```

During the build, `process.env.REACT_APP_NOT_SECRET_CODE` will be replaced with the current value of the `REACT_APP_NOT_SECRET_CODE` environment variable. Remember that the `NODE_ENV` variable will be set for you automatically.

When you load the app in the browser and inspect the `<input>`, you will see its value set to `abcdef`, and the bold text will show the environment provided when using `npm start`:

```
<div>
  <small>You are running this application in <b>development</b> mode.</small>
  <form>
    <input type="hidden" value="abcdef" />
  </form>
</div>
```

The above form is looking for a variable called `REACT_APP_NOT_SECRET_CODE` from the environment. In order to consume this value, we need to have it defined in the environment. This can be done using two ways: either in your shell or in a `.env` file. Both of these ways are described in the next few sections.

Having access to the `NODE_ENV` is also useful for performing actions conditionally:

```
if (process.env.NODE_ENV !== 'production') {
  analytics.disable();
}
```

When you compile the app with `npm run build`, the minification step will strip out this condition, and the resulting bundle will be smaller.

Referencing Environment Variables in the HTML

> *Note: this feature is available with `react-scripts@0.9.0` and higher.*

You can also access the environment variables starting with `REACT_APP_` in the `public/index.html`. For example:

```
<title>%REACT_APP_WEBSITE_NAME%</title>
```

Note that the caveats from the above section apply:

- Apart from a few built-in variables (`NODE_ENV` and `PUBLIC_URL`), variable names must start with `REACT_APP_` to work.
- The environment variables are injected at build time. If you need to inject them at runtime, follow this approach instead.

Adding Temporary Environment Variables In Your Shell

Defining environment variables can vary between OSes. It's also important to know that this manner is temporary for the life of the shell session.

Windows (cmd.exe)

```
set "REACT_APP_NOT_SECRET_CODE=abcdef" && npm start
```

(Note: Quotes around the variable assignment are required to avoid a trailing whitespace.)

Windows (Powershell)

```
($env:REACT_APP_NOT_SECRET_CODE = "abcdef") -and (npm start)
```

Linux, macOS (Bash)

```
REACT_APP_NOT_SECRET_CODE=abcdef npm start
```

Adding Development Environment Variables In `.env`

> *Note: this feature is available with `react-scripts@0.5.0` and higher.*

To define permanent environment variables, create a file called `.env` in the root of your project:

```
REACT_APP_NOT_SECRET_CODE=abcdef
```

> *Note: You must create custom environment variables beginning with `REACT_APP_`. Any other variables except `NODE_ENV` will be ignored to avoid accidentally exposing a private key on the machine that could have the same name. Changing any environment variables will require you to restart the development server if it is running.*

> *Note: You need to restart the development server after changing `.env` files.*

`.env` files **should be** checked into source control (with the exclusion of `.env*.local`).

What other `.env` files can be used?

> *Note: this feature is **available with `react-scripts@1.0.0` and higher**.*

- `.env`: Default.
- `.env.local`: Local overrides. **This file is loaded for all environments except test.**
- `.env.development`, `.env.test`, `.env.production`: Environment-specific settings.
- `.env.development.local`, `.env.test.local`, `.env.production.local`: Local overrides of environment-specific settings.

Files on the left have more priority than files on the right:

- `npm start`: `.env.development.local`, `.env.local`, `.env.development`, `.env`
- `npm run build`: `.env.production.local`, `.env.local`, `.env.production`, `.env`
- `npm test`: `.env.test.local`, `.env.test`, `.env` (note `.env.local` is missing)

These variables will act as the defaults if the machine does not explicitly set them.

Please refer to the dotenv documentation for more details.

> *Note: If you are defining environment variables for development, your CI and/or hosting platform will most likely need these defined as well. Consult their documentation how to do this. For example, see the documentation for Travis CI or Heroku.*

Expanding Environment Variables In `.env`

> *Note: this feature is available with `react-scripts@1.1.0` and higher.*

Expand variables already on your machine for use in your `.env` file (using dotenv-expand).

For example, to get the environment variable `npm_package_version`:

```
REACT_APP_VERSION=$npm_package_version
# also works:
# REACT_APP_VERSION=${npm_package_version}
```

Or expand variables local to the current `.env` file:

```
DOMAIN=www.example.com
REACT_APP_FOO=$DOMAIN/foo
REACT_APP_BAR=$DOMAIN/bar
```