Miscellaneous

—

.gitignore

HTML5 Boilerplate includes a basic project-level `.gitignore`. This should primarily be used to avoid certain project-level files and directories from being kept under source control. Different development-environments will benefit from different collections of ignores.

OS-specific and editor-specific files should be ignored using a "global ignore" that applies to all repositories on your system.

For example, add the following to your `~/.gitconfig`, where the `.gitignore` in your HOME directory contains the files and directories you'd like to globally ignore:

```
[core]
    excludesfile = ~/.gitignore
```

- More on global ignores: https://help.github.com/articles/ignoring-files/
- Comprehensive set of ignores on GitHub: https://github.com/github/gitignore

.editorconfig

The `.editorconfig` file is provided in order to encourage and help you and your team define and maintain consistent coding styles between different editors and IDEs.

By default, `.editorconfig` includes some basic properties that reflect the coding styles from the files provided by default, but you can easily change them to better suit your needs.

In order for your editor/IDE to apply the properties from the `.editorconfig` file, you may need to install a plugin.

N.B. If you aren't using the server configurations provided by HTML5 Boilerplate, we highly encourage you to configure your server to block access to `.editorconfig` files, as they can disclose sensitive information!

For more details, please refer to the EditorConfig project.

Server Configuration

H5BP includes a `.htaccess` file for the Apache HTTP server. If you are not using Apache as your web server, then you are encouraged to download a server configuration that corresponds to your web server and environment.

A `.htaccess` (hypertext access) file is an Apache HTTP server configuration file. The `.htaccess` file is mostly used for:

- Rewriting URLs
- Controlling cache
- Authentication
- Server-side includes
- Redirects
- Gzipping

If you have access to the main server configuration file (usually called `httpd.conf`), you should add the logic from the `.htaccess` file in, for example, a `<Directory>` section in the main configuration file. This is usually the recommended way, as using .htaccess files slows down Apache!

To enable Apache modules locally, please see the Apache modules documentation

In the repo the `.htaccess` is used for:

- Allowing cross-origin access to web fonts
- CORS header for images when browsers request it
- Enable `404.html` as 404 error document
- Making the website experience better for IE users better
- Media UTF-8 as character encoding for `text/html` and `text/plain`
- Enabling the rewrite URLs engine
- Forcing or removing the `www.` at the begin of a URL
- It blocks access to directories without a default document
- It blocks access to files that can expose sensitive information.
- It reduces MIME type security risks
- It forces compressing (gzipping)
- It tells the browser whether they should request a specific file from the server or whether they should grab it from the browser's cache

When using `.htaccess` we recommend reading all inline comments (the rules after a `#`) in the file once. There is a bunch of optional stuff in it.

If you want to know more about the `.htaccess` file check out the Apache HTTP server docs or more specifically the htaccess section.

Notice that the original repo for the `.htaccess` file is this one.

robots.txt

The `robots.txt` file is used to give instructions to web robots on what can be crawled from the website.

By default, the file provided by this project includes the next two lines:

- `User-agent: *` - the following rules apply to all web robots
- `Disallow:` - everything on the website is allowed to be crawled

If you want to disallow certain pages you will need to specify the path in a `Disallow` directive (e.g.: `Disallow: /path`) or, if you want to disallow crawling of all content, use `Disallow: /`.

The `/robots.txt` file is not intended for access control, so don't try to use it as such. Think of it as a "No Entry" sign, rather than a locked door. URLs disallowed by the `robots.txt` file might still be indexed without being crawled, and the content from within the `robots.txt` file can be viewed by anyone, potentially disclosing the location of your private content! So, if you want to block access to private content, use proper authentication instead.

For more information about `robots.txt`, please see:

- robotstxt.org
- How Google handles the `robots.txt` file

browserconfig.xml

The `browserconfig.xml` file is used to customize the tile displayed when users pin your site to the Windows 8.1 start screen. In there you can define custom tile colors, custom images or even live tiles.

By default, the file points to 2 placeholder tile images:

- `tile.png` (558x558px): used for `Small`, `Medium` and `Large` tiles. This image resizes automatically when necessary.
- `tile-wide.png` (558x270px): user for `Wide` tiles.

Notice that IE11 uses the same images when adding a site to the `favorites`.

For more in-depth information about the `browserconfig.xml` file, please see MSDN.

package.json

`package.json` is used to define attributes of your site or application for use in modern JavaScript development. The full documentation is available if you're interested. The fields we provide are as follows:

- `title` - the title of your project. If you expect to publish your application to npm, then the name needs to follow certain guidelines and be unique.

- `version` - indicates the version of your site application using semantic versioning (semver)
- `description` - describes your site.
- `scripts` - is a JavaScript object containing commands that can be run in a node environment. There are many built-in keys related to the package lifecycle that node understands automatically. You can also define custom scripts for use with your application development. We provide three custom scripts that work with WebPack to get you up and running quickly with a bundler for your assets and a simple development server.
  - `start` serves your `index.html` with a simple development server
- `keywords` - an array of keywords used to discover your app in the npm registry
- `author` - defines the author of a package. There is also an alternative contributors field if there's more than one author.
- `license` - the license for your application. Must conform to specific rules
- `devDependencies` - development dependencies for your package. In our case we have several dependencies used by WebPack, which we use as a simple development server.