

BlueUpCode Guide for HTML Template

Get starting

1. First, you must have [Node.js](#) installed on your computer.
2. Extract the downloaded package from the marketplace.
3. Start terminal and go to `/template/{variation}/` directory from the package that was extracted.
4. This template uses **Gulp** as a task runner that helps you automate your time-consuming tasks in the development workflow. To install Gulp globally, you need to run `npm install --global gulp-cli`.
5. Install all dependencies needed by running `npm install`.
6. Run `gulp --version` to verify that Gulp is successfully installed, and the version of installed Gulp will appear and make sure you have installed Gulp 4.x version.
7. Try to run `gulp build` on the terminal to build all assets.
8. Start the development server by running `gulp serve` and keep the terminal open and visit the link on the terminal. The server will automatically build assets and reload the browser if you edit the source codes. Press `CTRL+C` to stop the server.

Directory structure

This template comes with a simple and organized directory structure for easy to understand and maintainability. Look at the treeview below to find the basic template directory structure.

Template directory

- **dist/** - contains compiled code and assets
 - **assets/** - contains all assets for supporting the pages
 - **app/** - contains compiled app scripts
 - **build/** - contains all compiled library assets
 - **scripts/** - contains compiled library JS files
 - **styles/** - contains compiled library CSS files
 - **fonts/** - contains fonts
 - **images/** - contains images
 - ***.html** - HTML pages from compiled Nunjucks
- **src/** - contains source code and assets
 - **app/** - contains source app scripts
 - **assets/** - contains all additional assets
 - **build/** - contains buildable library assets
 - **core/** - contains Bootstrap and core component source codes
 - **vendors/** - contains custom 3rd party library source codes
 - **pages/** - contains source Nunjucks codes
- **tool/** - contains build tool scripts
- **config.json** - build configuration
- **gulpfile.js** - Gulp main script

Build tools

This template uses Gulp to power the build tools. The build tool provides an easy way to organize, compile, and bundle assets. Below are all the build tool commands.

Command	Description
<code>gulp clean</code>	Delete <code>dist/</code> directory
<code>gulp assets</code>	Copy all additional assets such as images, fonts that linked on <code>config.json</code> or inside <code>src/assets/</code> into <code>dist/assets/</code> directory
<code>gulp build:style-core</code>	Compile and bundle Bootstrap and core SCSS files that linked under <code>build.core.styles</code> object in <code>config.json</code> to <code>(ltr rtl)-core.css</code>
<code>gulp build:style-vendor</code>	Compile and bundle 3rd party library SCSS files that linked under <code>build.vendors.optional.[package-name].styles</code> object in <code>config.json</code> to <code>(ltr rtl)-vendor.css</code>
<code>gulp build:style</code>	Run all SCSS build tasks: <code>gulp build:(style-core style-vendor)</code>
<code>gulp build:script-core</code>	Bundle Bootstrap and core JS files that linked under <code>build.core.scripts</code> object in <code>config.json</code> to single file <code>core.js</code>
<code>gulp build:script-mandatory</code>	Bundle mandatory library JS files that linked under <code>build.vendors.mandatory.[package-name].scripts</code> object in <code>config.json</code> to single file <code>mandatory.js</code>
<code>gulp build:script-vendor</code>	Bundle 3rd party library JS files that linked under <code>build.vendors.optional.[package-name].scripts</code> object in <code>config.json</code> to single file <code>vendor.js</code>
<code>gulp build:script-app</code>	Compile with Babel all JS files in <code>src/app/</code> and moved into <code>dist/assets/app/</code> directory
<code>gulp build:script</code>	Run all JS build tasks: <code>gulp build:(script-core script-mandatory script-vendor script-app)</code>
<code>gulp build:page</code>	Compile all Nunjucks page codes in <code>src/pages/public/</code> and moved into <code>dist/</code> directory
<code>gulp build</code>	Run all build tasks <code>gulp (build:style build:script build:page assets)</code> parallely
<code>gulp serve</code>	Start the development server and watch source codes for live reloading. Press <code>CTRL+C</code> to stop the server

Configuration

All build tool configurations are located at `config.json` and you can customize the configuration.

Field	Type	Description
<code>config.port</code>	<code>integer</code>	Local development server port
<code>config.production</code>	<code>boolean</code>	Enable/disable production mode. By default, this option is <code>false</code> to speed up live reload. If you want to build for deployment, you must enable this option
<code>config.css_style</code>	<code>string</code>	CSS output structure, you can set this option with <code>nested</code> <code>expanded</code> <code>compact</code> <code>compressed</code>
<code>config.html_beautify</code>	<code>boolean</code>	Beautify compiled HTML files. You can disable this option if you want to increase build tool performance. This option will be automatically disabled if you enable production mode
<code>config.sourcemaps</code>	<code>boolean</code>	Enable/disable sourcemap. This option will be automatically disabled if you enable production mode
<code>config.skip</code>	<code>array</code>	An array of 3rd party libraries to be skipped from being compiled, you can skip the libraries under <code>build.vendors.optional</code> . example: <code>["apexcharts", "autosize", "datepicker"]</code>
<code>config.path</code>	<code>object</code>	Collection of paths
<code>config.output</code>	<code>object</code>	An object of build output file names
<code>build.core</code>	<code>object</code>	The object specifies Bootstrap and core buildable assets
<code>build.vendors</code>	<code>object</code>	The object specifies 3rd party library buildable assets
<code>build.vendors.mandatory</code>	<code>object</code>	Assets list under this node is required, you can't skipped to be compiled
<code>build.vendors.optional</code>	<code>object</code>	Assets list under this node can be skipped to be compiled

Page customization

This template uses Nunjucks to generate HTML pages. Nunjucks provides many useful features like template inheritance, variables, modulation, and more to make your development easier. You can find all Nunjucks source codes in `src/pages/` directory, look at tree view below to understand the directory structure.

Nunjucks source code directory `src/pages/`

- **components/** - contains all supporting components
 - **[component]/** - contains specific page components
 - **template.njk** - base template for inheriting
 - **variables.njk** - Nunjucks general variables
- **public/** - contains main pages that will be compiled

If you run the `gulp build:page` command, all Nunjucks files inside `src/pages/public/` will be compiled and moved into `dist/` directory. Don't make your pages from sketches, just start by customizing the prebuilt pages.

Tips: To set dark theme as default theme, you have to set `defaults.theme` variable in `variables.njk` with `dark` value.

Maybe your code editor doesn't support the `jinja` syntax highlighting of Nunjucks. There are plugins to support `jinja` syntax highlighting for some code editors.

- atom <https://github.com/alohaas/language-nunjucks>
- vim <https://github.com/niftylettuce/vim-jinja>
- brackets <https://github.com/axelboc/nunjucks-brackets>
- sublime <https://github.com/mogga/sublime-nunjucks/blob/master/Nunjucks.tmLanguage>
- emacs <http://web-mode.org>
- vscode <https://github.com/ronnidc/vscode-nunjucks>

Stylesheet customization

This template uses SCSS for handling the CSS efficiently. You can find all Bootstrap and core SCSS files in `src/build/core/styles/`. We remove several Bootstrap components because we replace them with our custom components. Find the custom 3rd party library SCSS files in `src/build/vendors/(package-name)/styles/`.

You can easily customize the color palette and other options by editing the variables in `variables.scss`.

SCSS source directory structure `src/build/core/styles/`

- **components/** - contains component styles
- **mixins/** - contains SCSS mixins
- **utilities/** - contains utilities
- **widgets/** - contains widgets
- **_components.scss** - concatenate all components
- **_functions.scss** - helper functions
- **_mixins.scss** - concatenate all SCSS mixins
- **_reboot.scss** - reset default page style
- **_utilities.scss** - concatenate all utilities
- **_variables.scss** - global variables
- **_widgets.scss** - concatenate all widgets
- **index.scss** - the main file to be compiled and concatenate all parts

Script customization

There are 2 types of the JS scripts: **Bundlable script** and **App script**. Bundlable script is all library scripts that will be bundled to a single file. App script is the script that is written for specific pages.

Bundlable script

Bundlable script is all JS files that are linked under `build` object in `config.json`. Bundlable scripts include all core and 3rd party library scripts, all the scripts will be bundled and moved into `dist/assets/build/scripts/`.

Find all Bootstrap and core component scripts in `src/build/core/scripts/`. This template takes 3rd party library scripts from node modules directory, but several library scripts have been customized. All the custom 3rd party library scripts are located inside `src/build/vendors/(package-name)/scripts/`.

App script

All app scripts are located in `src/app/`, all scripts inside this directory will be compiled with Babel and moved into `dist/assets/app/`. These scripts are written for a specific page. You can use ES6 syntax for these scripts.