Setup

Built-In Tools

From the Semantic directory you can setup gulp to build Semantic by running.

```
npm install
```

Semantic will automatically configure itself using a `post-install` script built into the package.

After set-up can use gulp to build your project's css:

```
# Watch files
gulp watch

# Build all files
gulp build
```

Visit the Getting Started Guide for more details on set-up

Custom Pipelines

Importing Gulp Tasks Each gulp task can be imported into your own Gulpfile using `require`

```
const watch = require('path/to/semantic/tasks/watch');

gulp.task('watch ui', 'Watch Semantic UI', watch));
```

Importing LESS

> LESS files do not contain vendor prefixes. If you are to use these files directly you must add them during your build step.

Before using source files you will need to create a `theme.config` by renaming `theme.config.example`, and a site folder by renaming `_site/` to `site/`

You can then import Semantic from your own LESS files:

```
/* Import all components */
@import 'src/semantic';
```

To import individual components you will have to create a scope for each import using `& {}`

```
/* Import a specific component */
& { @import 'src/definitions/elements/button'; }
```

Config Files

These files are generated automatically using install scripts, but must be manually renamed if you are using installing manually.

filename	usage	Initial Name
theme.config	config file that stores each element's current theme for LESS	theme.config.example
site/	folder storing all your site's variables and css overrides for each UI component	_site/
semantic.json	stores folder paths for build tools and current installed version for updates. Only necessary when using build tools	semantic.json.example

Workflow

You will only need to use Semantic's build tools while refining your UI. When designing pages, you can rely on the compiled css packages in `dist/`.

When creating your UI you can try downloading different themes, adjusting your site-wide settings (font-family, colors, etc) and tweaking components in your site's component overrides.

Files in the `examples/` folder of your project can be useful for testing out changes in your UI. For example, you might run `gulp watch` download a new theme to `src/site/themes/` then adjust your `theme.config` file with the name of the new theme and refresh `examples/kitchensink.html` to inspect changes in the theme.

Theming

Concepts

Inheritance There are three levels of inheritance in Semantic * Default theme - Semantic UI's neutral default theme * Packaged theme - A specified packaged theme, like "amazon", or "material" * Site theme - A theme specific to your site

Folder Structure

- `definitions/` contains the `css` and `javascript` definitions for each component
- `themes/` contains *pre-packaged themes* including Semantic's default theme
- `site/` contains your current site's theme

View the Theming Guide for a more in-depth look

Customizing

Basic Customization The best way to start customizing is to specify overriding variables in your site's `site.variables` file.

This is a blank stub file that lets you specify variables that overriding variables.

Some important values to customize: * Base font size * Named color hex codes * Header/Page Font-families * Primary and secondary colors * Grid column count

To find out what variables are available to modify, you can inspect the variables in the default theme in `themes/default/`

Advanced Configuration Each component has its own variable file, which can be used to modify any of the underlying variables for that component.

For example `/site/elements/button.variables`.

You may also specify your own custom LESS in `site/elements/button.overrides`. This file will have access to all underlying variables available for that component.

Using Pre-Packaged Themes You can modify `theme.config` to use any prepackaged theme available in `src/themes/`.

For example you can modify `theme.config` to use a `github` button theme by changing

```
@button: 'github';
```

View the Customization Guide to learn more