SP2013Framework Documentation

Integrating Modern JavaScript tools into SharePoint 2013

# Installation Process

### Step 1: Install Global Dependencies

Make sure the following open source programs are installed on your local machine globally before starting the installation process. Please refer to their documentation for instructions on how to install them. You must install NodeJS before anything else.

|  |  |
| --- | --- |
| **Program** | **URL** |
| NodeJs | <https://nodejs.org/en/download/> |
| Gulp | <https://github.com/gulpjs/gulp/blob/master/docs/getting-started.md> |
| Yeoman | <http://yeoman.io/learning/index.html> |
| Typescript | <https://www.typescriptlang.org/index.html#download-links> |
| Bower | <https://bower.io/> |

### Step 2: Register Yeoman Generator

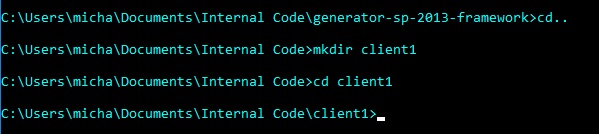
Since this project is not in the npm public registry, you must manually install the generator’s dependencies and link the project to npm.

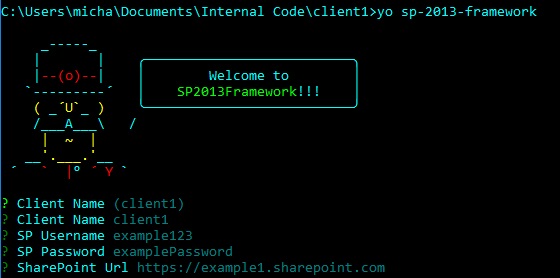
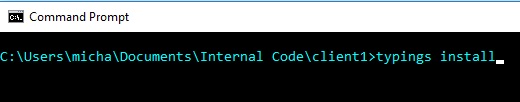
1. Navigate to the generator’s root directory and run “npm install”npm_install
2. Once finished, run “npm link” to register the project’s name to npm’s global symlink.npm_link

### Step 3: Using the Yeoman Generator

Now that the generator is linked to npm, you will be able to create a new project using Yeoman from the command line in any directory.

1. Create a new directory using the client’s name.



1. You will now be able to run the Yeoman generator which will ask you for your account credentials. These credentials must be correct for the framework to interact with the SharePoint site otherwise you will get a nasty error message in Step 4.
   1. If you are installing this setup on a sub site or site collection, the full url will be required in the SharePoint Url parameter.
   2. Once you are done, Yeoman will automatically install all node and bower dependencies.
2. Since Yeoman won’t automatically install typescript’s “definitely typed” files, you will have to install them yourself by running “typings install”

### Step 4: Install the framework on the SharePoint site

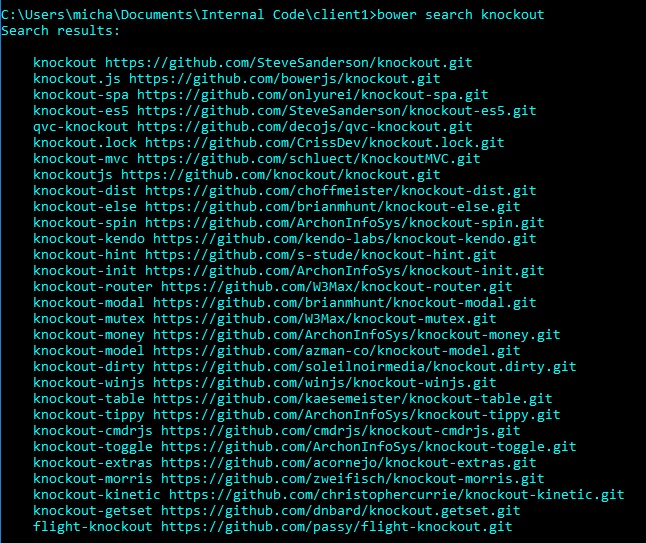
To make sure everything was installed correctly, it is time to test the link from the framework to your SharePoint site. Run the command “gulp push:sharepoint” to add all theme files to the SharePoint site.C:\Users\micha\AppData\Local\Microsoft\Windows\INetCacheContent.Word\push_sharepoint.jpg

All files will now be pushed to the SharePoint site if everything was installed correctly. In this case, you will find a new masterpage named “custom.html” inside the “client1” folder in the sharepoint masterpage directory. To view a full list of gulp commands, see the “gulpfile.js” on the root of your project.

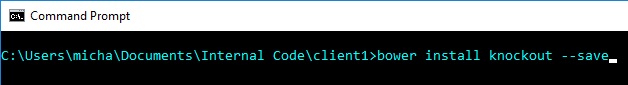
# Package Manager – Adding 3rd party libraries

SP2013Framework uses bower for managing front end dependencies. Each library will also require its own definitely type file so typescript can register their properties in the function calls. To keep track of the dependencies, it is recommended to first use bower to install the library and then use typescript to get the library’s definitely typed files. To demonstrate this, we are going to install the knockout.js framework to our example client1’s sharepoint site but this will work on any library registered with bower and typescript.

### Step 1: Install Knockout using bower

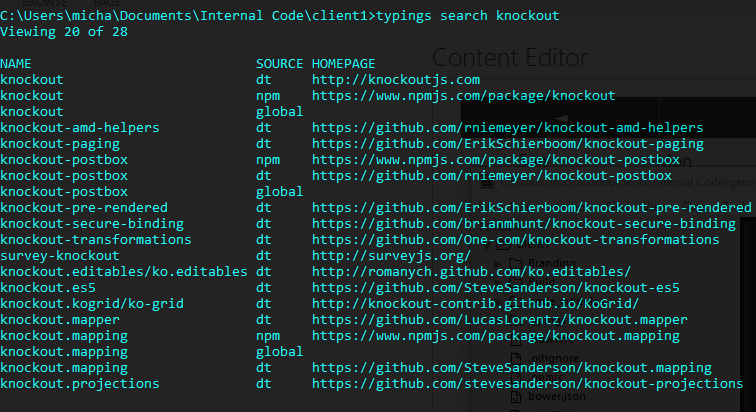
To check if bower has a library in its registry, you can run something like “bower search knockout” to pull up a list of projects bower can install.

For now, we will only need to install the first one. Run “bower install knockout –save” to install knockout and save its url to your bower.json file. It is important to run this command with the –save flag so our framework can keep track of dependencies and will make it easier if you need to move the project to another machine.



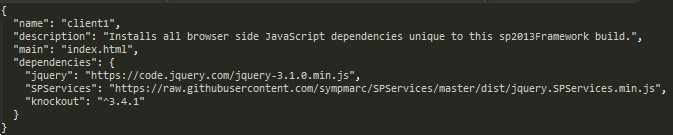
### Step 2: Install Knockout’s definitely typed file

To find out if a definitely typed file exists for knockout, we can search for it in a similar way we did for step 1. Simply run “typings search knockout” to see if a definitely typed file can be downloaded from the command line.

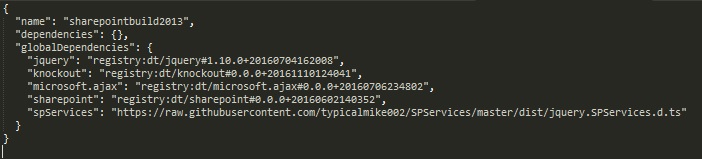


Once you know the definitely typed file exists, you will have to install the typings (.dt) files from the dt source directory as indicated from the search command. This framework requires you to run this command with the –global and –save flag so for this example, run “typings install dt~knockout –global –save”install_knockout_typings

### Step 3: Making sure everything worked

If you ran the bower command in step 1 with the –save flag, inside the “bower.json” file you will see the knockout library and version number currently being used. 

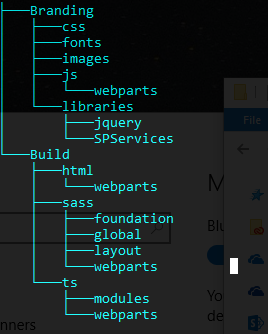
You may notice that urls are being used for pulling the jquery and SPServices library. This is a more efficient but manual way of downloading dependencies because it will not pull all the project dependency files. You cannot run these from the command line and must place the url inside this file and run, “bower install” if you don’t want to include knockout’s project dependency files.

Inside your “typings.json” file, you will also see a reference to knockout’s definitely typed source code.

Both of these package managers are very good at keeping track of your project’s dependencies and will make it trivial to move projects from one computer to another. If you were to run either “bower install” or “typings install” without specifying a library name, you will install everything referenced inside each .json file respectively.

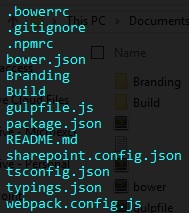
# Framework Architecture – Files and Folders

SP2013Framework is your launching point for all SharePoint 2013 branding projects. It ships with a simple folder hierarchy that ensures the right files get saved to the same place on your SharePoint site. This section goes over the features that help keep your architecture organized on your local machine and in SharePoint as well as how gulp works to push and pull files from SharePoint. Below is an image of all the directories without their contents.



### About the Project Root Files

All files that set up the configuration of the framework sit in the project’s root directory. This includes configuration files for tools like gulp, typescript, and webpack. It also contains an encrypted copy of your user credentials for logging in and out of sharepoint which gulp will handle for you when you run gulp commands.



#### Project Root Files

**.bowerrc:** Bower configuration file.

**.gitignore:** Contains a basic list of files for git to ignore.

**.npmrc:** NPM configuration file.

**bower.json**: Bower’s project file, keeps track of front end libraries that the project depends on.

**gulpfile.js:** Contains all gulp commands that the framework requires to move and compile files.

**package.json:** NPM’s project file, keeps track of build tool dependencies SP2013Framework depends on.

**README.md:** Project’s README file that can be used for developer notes.

**sharepoint.config.json:** File that contains all user credentials, the site url, and a list of directories that gulp will point to when saving files.

**tsconfig.json:** Typescript configuration file.

**typings.json:** Typing’s project file, keeps track of all definitely typed files typescript depends on when working with libraries.

**webpack.config.js:** Webpack configuration file (webpack is required for compiling typescript into javascript and is responsible for generating minified code)

#### The ‘Build’ Directory

Contains uncompiled source code for the project. All typescript, sass, and html files go into these directories. Think of it as the place you will be writing your code and the framework will refer to this directory before compiling things and moving them to your sharepoint site.

**Build/html:** Contains all master page html files. By default, the name of the master page is “custom.html”. This is where you will add sharepoint snippets and put the references to script and stylesheets.

**Build/html/webparts:** Contains any custom webparts you want to add to the sharepoint site. By default, it contains an example of a webpart that points to a javascript file that is only used by this webpart. It is important to keep webparts in this directory so SP2013Framework compiles them correctly.

**Build/sass:** Contains all sass files. ‘style.scss’ is the entry file which pulls in all of the other sass modules and is used by ‘custom.html’ as the main stylesheet for the site. These can be organized anyway the developer wants to organize them but the framework does come packed with default styles ready to be used. It doesn’t matter where you put styles related to webparts (unlike the .html and .ts files).

**Build/ts:** Contains all typescript files. “main.ts” is by default the entry point for all external typescript modules but again, these can be organized however the developer wants.

**Build/ts/webparts:** Contains entry points for webpart specific javascript files. It is important to keep webpart entry typescript files in this directory so that SP2013Framework will handle them correctly.

**Build/ts/modules:** Contains reusable framework components. As the framework gets used, overtime these modules may come in handy for future projects. You can import modules into either “main.ts” or into any webpart of your choice.

#### The ‘Branding’ Directory

Stores a copy of all compiled and static files which are sent to the sharepoint site by SP2013Framework. This folder contains the files that sharepoint will use to brand its master page. You will have to add files here like images and fonts but other than that, unless you need to fix a javascript library source code, this directory is mainly used as a place holder for static files before they get sent to SharePoint

**Branding/css:** Contains css code compiled from sass.

**Branding/fonts:** Contains all font files.

**Branding/images:** Contains all static images.

**Branding/js:** Contains all js files compiled from ts

**Branding/libraries:** Contains all 3rd party javascript libraries. Bower will automatically save 3rd party javascript libraries here (see the section on package management for details).