Angular UX Notes

# INSTALLATION/UPGRADE

## NPM Upgrade

### Windows

Run command in power shell

>npm install --global --production npm-windows-upgrade

>Npm-windows-upgrade //in admin mode

# npm install

A package is:

a) a folder containing a program described by a [package.json](https://docs.npmjs.com/files/package.json) file

b) a gzipped tarball containing (a)

c) a url that resolves to (b)

d) a **<name>@<version>** that is published on the registry (see [npm-registry](https://docs.npmjs.com/misc/registry)) with (c)

e) a **<name>@<tag>** (see [npm-dist-tag](https://docs.npmjs.com/cli/dist-tag)) that points to (d)

f) a **<name>** that has a "latest" tag satisfying (e)

g) a **<git remote url>** that resolves to (a)

## Formats

npm install (with no args, in package dir)  
npm install [<@scope>/]<name>  
npm install [<@scope>/]<name>@<tag>  
npm install [<@scope>/]<name>@<version>  
npm install [<@scope>/]<name>@<version range>  
npm install <tarball file>  
npm install <tarball url>  
npm install <folder>

## Private Hosting

**Cmd line**

npm install git+ssh://git@[techjini.repositoryhosting.com/techjini/petrofac\_landing\_portal.git](http://techjini.repositoryhosting.com/techjini/petrofac_landing_portal.git)

npm install git@github.com:dimsemenov/PhotoSwipe.git#v3.0.x

**package.json**

"Dependencies":

{

. . .

"mySpaceApp": "git+ssh://git@[techjini.repositoryhosting.com/techjini/petrofac\_landing\_portal.git](http://techjini.repositoryhosting.com/techjini/petrofac_landing_portal.git)",

}

**package**

npm install ./package.tgz

npm install <https://github.com/indexzero/forever/tarball/v0.5.6>

# Creating Project

## Gruntfile.js

Server default location

connect: {

options: {

base: ‘example’

…

# How To create a Yo Generator

Ref: [Create A Custom Yeoman Generator in 4 Easy Steps](https://scotch.io/tutorials/create-a-custom-yeoman-generator-in-4-easy-steps)

1. npm install -g yo generator-generator
2. yo generator

# BluePrint Generator

## Blueprint-directive-generator

### Prerequisites

Install yo, grunt-cli, bower, generator-angular and generator-karma:

> npm install -g grunt-cli bower yo generator-karma generator-angular

### Install BluePrint Directive Generator

The generator can be installed from Techjini Repo using ssh protocol

> npm install -g git+ssh://git@techjini.repositoryhosting.com/techjini/generator-blueprint-directive.git --force

### Creating Project

(Your inputs are highlighted in yellow)

> yo blueprint-directive

? What is your directive name? (Directive) menubar

? What is your directive name? menubar

? What is your GitHub user name? (someuser) Jatin

? What is your GitHub user name? Jatin

Above steps will create a bower publishable project in folder **blueprint-directive-bpMenubar** with following contents-

blueprint-directive-menubar

│ .bowerrc

│ .editorconfig

│ .gitattributes

│ .jscsrc

│ .jshintrc

│ .npmignore

│ .yo-rc.json

│ bower.json

│ Gruntfile.js

│ package.json

│

├───bower\_components

├───demo

│ demo.css

│ demo.js *//Set you directive configuration here*

│ index.html

│

├───node\_modules

│

**├───src** *// your directive source is here.*

**│ │ bpMenubar.html** *// All new code should be*

**│ │** *// in src folder only*

**│ ├───scripts**

**│ │ bpMenubar.js**

**│ │**

**│ ├───services**

**│ │ bpMenubarServiceProvider.js**

**│ │**

**│ └───styles**

**│ bpMenubar.less**

│

└───test

│ .jshintrc

│ karma.conf.js

│

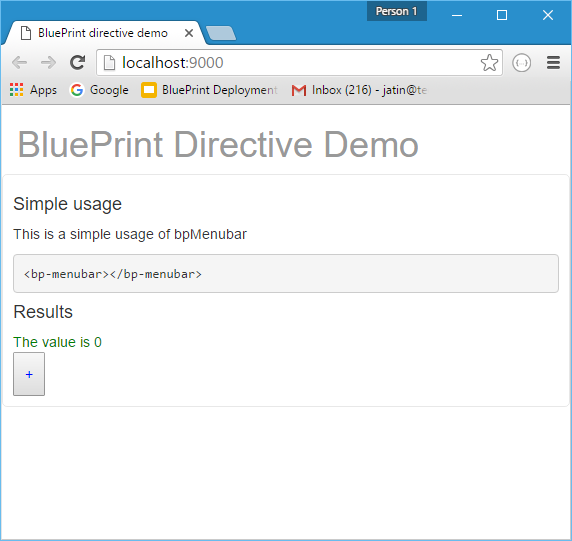
└───spec

└───services

### Run newly created Project

The project is ready to be executed

blueprint-directive-menubar> grunt serve



### Building Project

Execute command

blueprint-directive-menubar> grunt build

dist folder will be created with following files

dist

│ bpMenubar.css

│ bpMenubar.js

│ bpMenubar.min.css

│ bpMenubar.min.js

[note: all the files (**.js** and **html** template) in the **src** folder will **concatenated**, **uglified** and **minified** into above files]

### Check-in in the newly created Project

If we want to commit the project to TJ repository ssh://git@techjini.repositoryhosting.com/techjini/blueprint-directive-menubar.git

Execute following commands

blueprint-directive-menubar> git init

blueprint-directive-menubar> git remote add ssh://git@techjini.repositoryhosting.com/techjini/blueprint-directive-menubar.git

Edit .gitignore

/node\_modules

/.tmp

/.sass-cache

/bower\_components

/.tmp

\*.sw?

[note: do not add dist folder to .gitignore]

blueprint-directive-menubar> git add .

blueprint-directive-menubar> git commit -a

blueprint-directive-menubar> git push -u origin master

### Adding project to your code

your-code> bower install --save git+ssh://git@techjini.repositoryhosting.com/techjini/blueprint-directive-menubar.git

This will add following two lines to you index file-

<link rel="stylesheet" href="**bower\_components/blueprint-directive-menubar/dist/bpMenubar.css**" />

<script src="**bower\_components/blueprint-directive-menubar/dist/bpMenubar.js**"></script>

### Adding project to your code without commit

If you want to test the code before doing a commit to repo, execute following commands

blueprint-directive-menubar> grunt build

blueprint-directive-menubar> cd ../path/to/your/code

your-code> bower install --save ../path/to/folder/blueprint-directive-menubar/

**Restoring user state**

A tricky part of authentication in a single page application (SPA) is retaining or restoring the information of the logged in user when the visitor refreshes the page. Since all of the state is client-side, refreshing will clear user info. In order to fix this I usually implement an API endpoint (e.g. /profile) which returns the user data when logged in. This endpoint is called when starting the AngularJS application (e.g. in the “run” function). The user data is then saved in a Session service or in $rootScope, just as you would after a successful login. Alternatively you can embed the user data in index.html, so you don’t have to do an additional request. A third solution would be to store the user data in a cookie or Local Storage, but that makes it harder to force a logout and/or forget user data.

When trying to directly (by URL) access a route which requires a user to be logged in, it may happen that the user data is not loaded before the page is rendered. This can be avoided by defining a ‘resolve’ property on the route:

$stateProvider.state('protected-route', {

url: '/protected',

resolve: {

auth: function resolveAuthentication(AuthResolver) {

return AuthResolver.resolve();

}

}

});

The AuthResolver is a service which watches the value of ‘currentUser’ on $rootScope, and will only resolve after currentUser has been set:

.factory('AuthResolver', function ($q, $rootScope, $state) {

return {

resolve: function () {

var deferred = $q.defer();

var unwatch = $rootScope.$watch('currentUser', function (currentUser) {

if (angular.isDefined(currentUser)) {

if (currentUser) {

deferred.resolve(currentUser);

} else {

deferred.reject();

$state.go('user-login');

}

unwatch();

}

});

return deferred.promise;

}

};

})

Since this watcher will only be used once, it should be disabled afterwards. This can be done by *unwatching*: $watch returns a function which you can call to remove the watcher. You could extend the example above with additional functionality such as a redirect to the original page after login.

There’s countless variations and other options that you can explore. In the end it will mostly depend on the specifics of your back-end implementation. Keep in mind that the only real security is on the server-side. And remember to always use HTTPS.

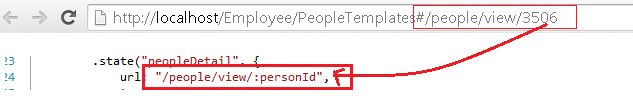
[https://github.com/yeoman/generator-angular#view](https://github.com/yeoman/generator-angular%23view)

**AngularJS-Boilerplate this is alternate of Yo with better modula code structure**

<https://github.com/jbutko/AngularJS-Boilerplate>

# Hiding Route in the URL

[Stackoverflow Qs](http://stackoverflow.com/questions/29062465/how-to-hide-stateprovider-url-displayed-in-address-bar-using-angular-js)?



Is it possible to hide those potential information using Angular JS 1.3.0?

Answer:

The personId is displayed in the url, because its defined as URL parameter.

But we can avoid that, by defining it with params notation:

.state("peopleDetail", { url : "/", params : { personId : null }, ... }

Check the doc:

### [$stateProvider](http://angular-ui.github.io/ui-router/site/%23/api/ui.router.state.$stateProvider)

A map which optionally configures parameters declared in the url, or defines additional non-url parameters. For each parameter being configured, add a configuration object keyed to the name of the parameter.

...

Each parameter configuration object may contain the following properties:

* value
* array
* squash -- (see more in that resource)

For similar stuff and some more details, please check:

[Angular ui router passing data between states without URL](http://stackoverflow.com/q/28248236/1679310)