Cheatography

Yeoman Generator Development (CoffeeScript) Cheat Sheet

by Arun N Kumar (gettoarun) via cheatography.com/26194/cs/7280/

Start Up - Initialize your project

Create an empty node project	**npm init .**
Make the folder structure as below	<see structure below>
Add / register generator folders in package.json	<see sample below>
Add yeoman-generator deps	npm i --save yeoman-generator

folder structure

```
generator-goofy
├──package.json
└──generators/
       ├──app/
       └──index.js
```

package.json

```
{
  "name": "generator-goofy",
  "version": "1.0.0",
  "files": [
    "generators/app",
```

A Generator Class

Import required deps	generators = require 'yeoman-generator'
Extend from generators.Base	class GoofyGen **extends generators.Base**
Ensure CTOR initializes yeoman env	constructor : () -> **generators.Base.apply this, arguments**
Add whatever methods you need	...
Finally export the class	**module.exports** = GoofyGen

Example

```
generators = require 'yeoman-generator'

class GoofyGen extends generators.Base
  constructor : () ->
    generators.Base.apply this, arguments

    @log 'Initializing option...'

  method1 : () ->
    @log 'processing...'

module.exports = GoofyGen
```

Yeoman Run Priorities

Yeoman Run Priorities (cont)

end	Called last, cleanup, say good bye, etc

Developers get to control the execution flow and composability, yeoman implements a Grouped-queue based Run loop and priorities.

Priorities are defined in your code as special prototype method names. When present they execute it accordingly.

```
generators.Base.extend({
  priorityName: {
    method: function () {},
    method2: function () {}
  }
});
```

initializing	Your initialization methods (checking current project state, getting configs, etc)
prompting	Where you prompt users for options (where you'd call this.prompt())
config-uring	Saving configurations and configure the project (creating .editorconfig files and other metadata files)
default	If the method name doesn't match a priority, it will be pushed to this group.
writing	Where you write the generator specific files (routes, controllers, etc)
conflicts	Where conflicts are handled (used internally)
install	Where installation are run (npm, bower)

By **Arun N Kumar** (gettoarun)
cheatography.com/gettoarun/
www.arunkumar.io

Not published yet.
Last updated 13th May, 2016.