Angular | class 14

Testing Tools →

Karma init → to initialize karma
Karma start → to start karma (to test) while start.

1. npm init
2. npm install jasmine --save-dev
3. npm install karma --save-dev
4. npm install karma -cli -g
5. npm install karma-jasmine --save-dev
6. npm install karma-chrome-launcher --save-dev
7. npm install karma-firefox-launcher --save-dev
8.

- Or from bower

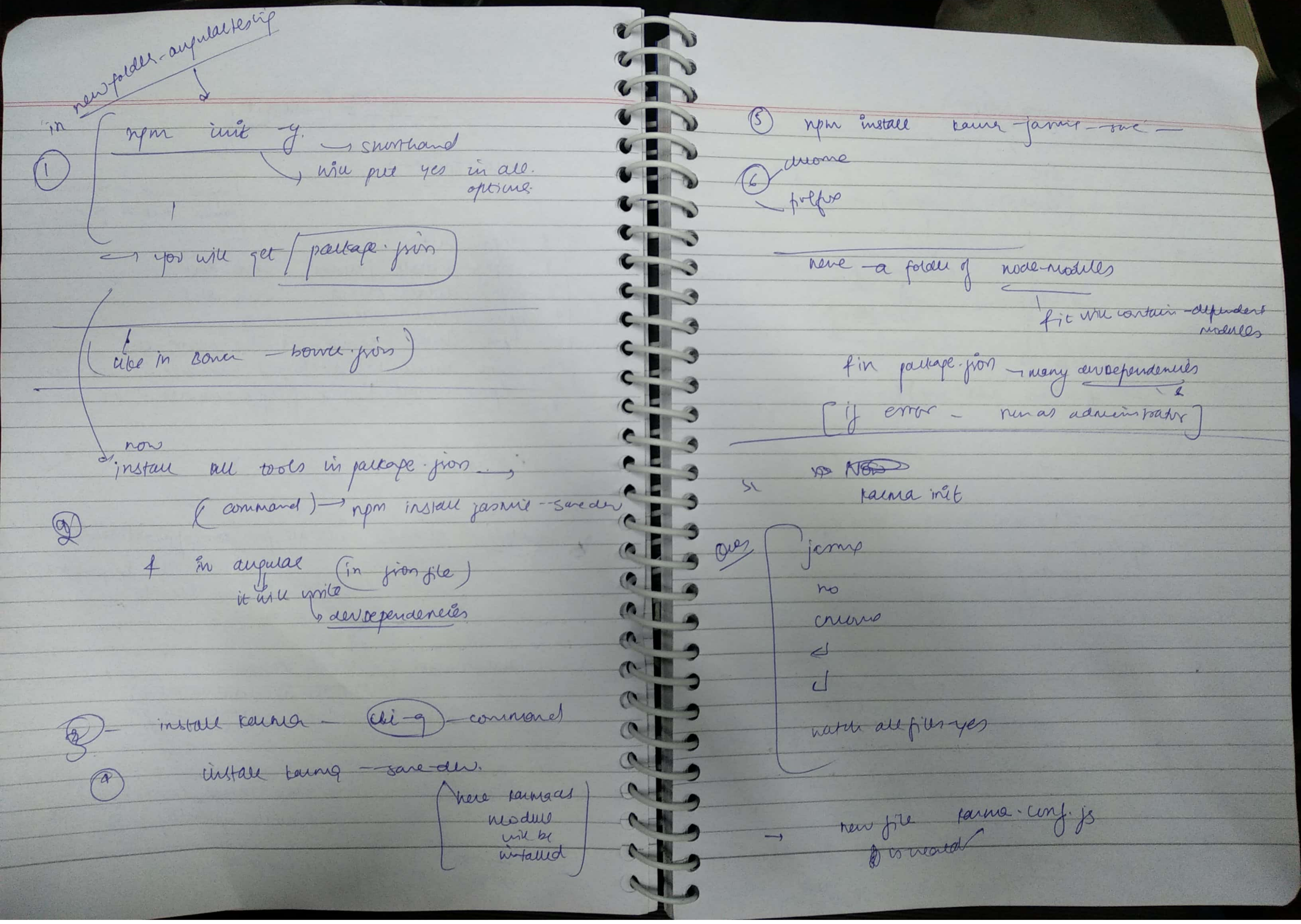
» -dev → means - all are for development mode & will not be pushed to server.

» → Jasmine — for unit testing (also do integration testing)

→ karma — cross browser — run on all browsers chrome, firefox. etc.

→ karma-cli — to run karma commands ∴ install globally

→ ~~jasmine~~ chrome launcher & firefox launcher.



new folder - angulartestip

in

① npm init -y. → shorthand
→ will put yes in all. options.

→ you will get package.json

(like in bower - bower.json)

now install all tools in package.json →

(command) → npm install jasmine --save-dev

② f in angular (in json file)
it will write
→ devDependencies

③ - install karma - (cli -g) - command

④ install karma --save-dev.

(here karma's module will be installed)

⑤ npm install karma-jasmine --save-

⑥ chrome
firefox

here - a folder of node-modules
it will contain - dependent modules

in package.json → many devDependencies

[if error - run as administrator]

~~NOW~~

karma init

Ques
jasmine
no
chrome
↵
↵
watch all files - yes

→ new file karma.conf.js is created

make a new ~~file~~ folder

src
(codeing)

spec
(testing)

make new file
addspec.js

add.js

~~in addspec.js~~ in

```
function add(x,y){

    }
```

in spec.js
'code for test'

>> unit Testing — testcase — for a particular case of a func^n
test suite
all testcases together

karma — five test suite
& each suite manages 1 suite for 1 func^n
if 50 suite — 50 func^n

coding for testing (JS)

predefined func^n of jasmine
name of suite
func^n of suite

```
describe("add test suite", function(){
   testcase ← it ("should add 2 nos", function(){   signature
        var result = add(100,200);
        expect(result).toBe(300);   jasmine func^n
   })
});
```

if cond^n passes then Test case(passed)

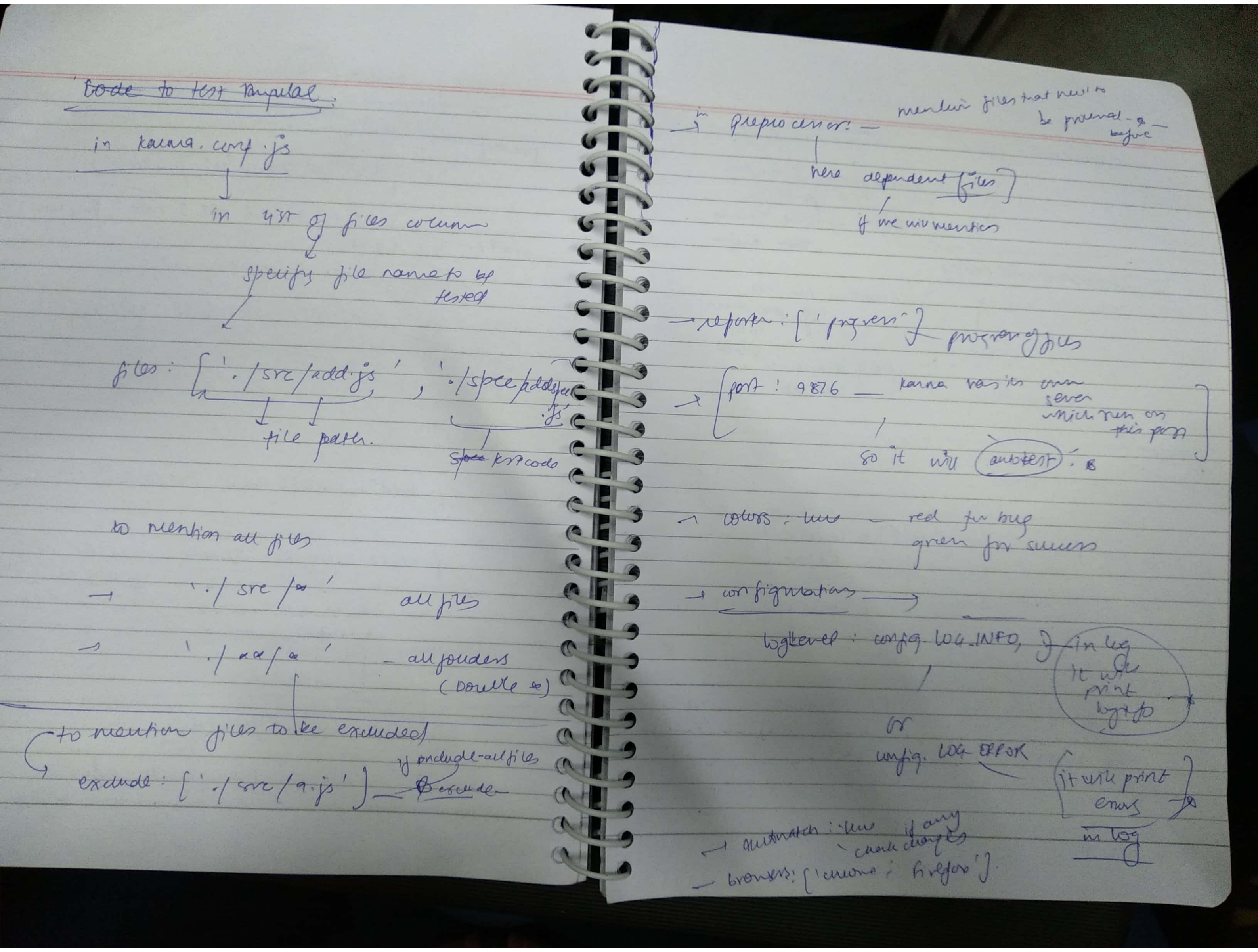
& make copy of 'it' code to make copy of test cases.*

100 - 200
- 300

100 - 200
- 100

TDT (Test driven technology)

first test cases then write code



~~Code to test Angular~~:

in karma.conf.js

in list of files column

specify file name to be tested

files: ['./src/add.js' , './spec/addspec.js']

file path.

~~Spec~~ test code

to mention all files

→ './src/*' all files

→ './**/*' - all folders (Double *)

to mention files to be excluded

exclude: ['./src/a.js'] — if include-all files ~~Exclude~~

→ in preprocessor: — mention files that need to be processed before

here dependent files

if we will mention

→ reporter: ['progress'] progress of files

→ [port : 9876 — karma has its own server which run on this port]

so it will autotest : 8

→ colors : true — red for bug
green for success

→ configurations →

logLevel : config.LOG_INFO, } in log it will print loginfo

or

config.LOG_ERROR

{ it will print errors } in log

→ autowatch : true if any change changes

→ browsers: ['chrome', 'firefox']

singleRun : false

6-7

It will check both Browsers parallely

if true → check chrome then firefox

→ concurrency: Infinity

→ mention the no. of processors to be used

if infinity it will use all.

Tunnel

karma start

6 cases successfull (3 in chrome
3 in firefox)

Simulation Env.

>) fake injections

as we are testing 'single unit'

∴ Other should be fake.

to create a environment called simulation env

to make this

→ install angular-mocks

→ install angular

→ we can use 'bower' for this.

- bower install angular --save
- bower install angular-mocks --save

Now in ~~controller~~ in src folder

make controller.js
& ~~fact~~ service.js

& reuse some prev code.

Scanned by CamScanner

in spec folder

- controllerspec.js
- servicespec.js

in controllerspec.js

describe (" name " , func () { }

~~init~~

beforeEach (module ("myapp"));

only difference is injections

- inject ~~angular~~ ng-app
- inject — $controller

of these injections using ("Mock")

Spec — Jasmine (describe init)
- injection — Mock
- Execution karma

to run code before testcases

beforeEach (module ("myapp")) — ~~ng-app~~ myapp

of angular mock

function equivalent ngapp

to inject "myapp"

var $controller;

beforeEach(inject (function (_$controller_){

ng-controller

- underscore — reserved word
- / always put '_' before & after.

underscore makes it equivalent to ng-controller.

$controller = _$controller_; //inject controller

myctrl

$scope

var controller = $controller(' myctrl', {$scope : myscope }]

origine object

for controller need scope also *

var myscope = {};

myscope. firstnumber = 100

myscope. second number = '200';

Protractor — will do these simulation for us

only code — by user
firstname
secondname.

karma.config.js
↓
files: ['./src/*.js' , './bower_components/angular/angular.min.js',
'./bower_components/angular-mocks/angular-mocks.js'

karma start

Grunt
↳ how to push on server

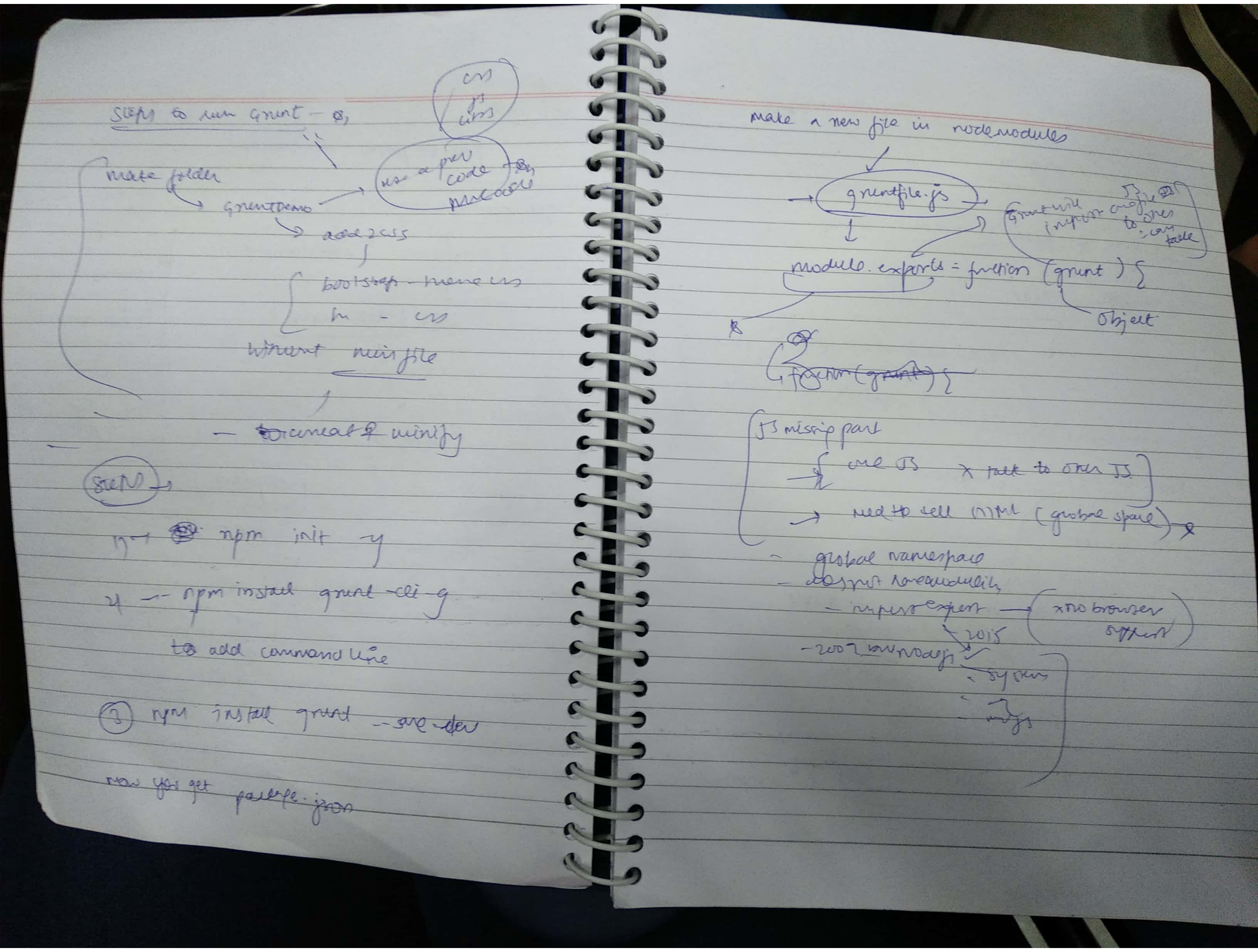
goto gruntjs.com → 6000-7000 plugins
↓
JS task runner
↓
task → to compress time
to concat
compressors
minification
→ linting → code standard

online tools
JS minifiers
to automate this task
↓
write code

~~karma~~
↓
Test runner

Jasmine
to write
testcases

→ plugins — coffeescript



Steps to run Grunt — 8,

css
js
imgs

use a prev code for prev code

make folder
→ GruntDemo
→ add 2 css
{ bootstrap — theme css
{ [illegible] — css

without main file

— ~~t~~concat & minify

Steps →

1) → ~~[illegible]~~ npm init -y

2) → npm install grunt-cli -g

to add command line

3) npm install grunt --save-dev

now you get package.json

make a new file in node modules

→ gruntfile.js →

grunt will import config to others JS can take

module.exports = function (grunt) {

Object

~~function (grunt)~~ {

JS missing part
→ { one JS × talk to other JS }
→ need to tell HTML (global space) ×

- global namespace
- doesn't have modularity
- import export → (× no browser support)
- 2015
- 2009 now node js
- sys [illegible]
- [illegible] js

Scanned by CamScanner

~~7:36~~

(week) 2
grunt plugins

→ Star ← plugin — made by grunt.

(1st plugin)
→ contrib-clean
↓
[when we do changes if make new folders / to delete prev one]

contrib-uglify — compress

contrib-concat

↓
all plugin are in node
..npm

[1) Steps ~~to~~
(install uglify) using npm]

[~~package.json~~ entry]

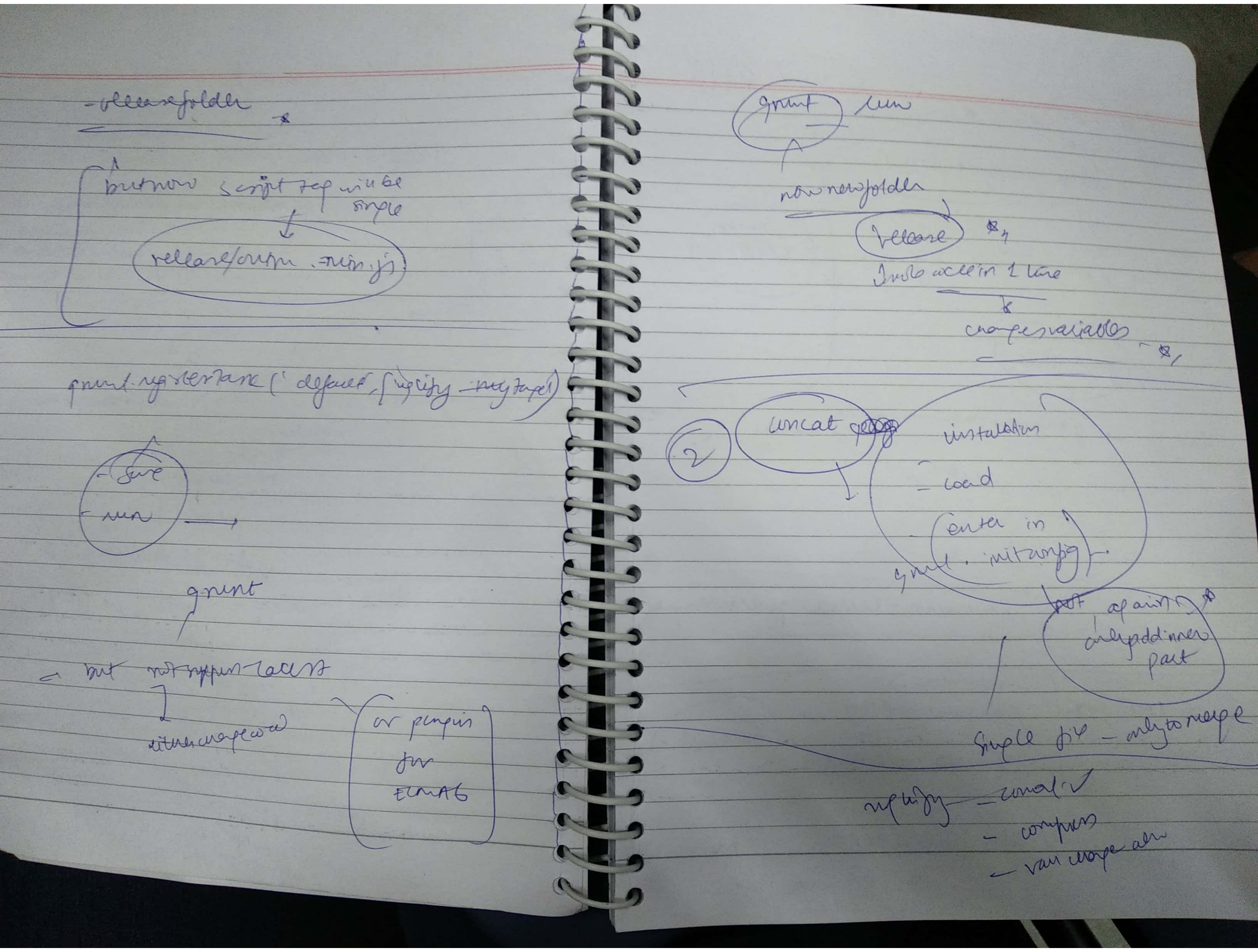
(load) steps

[grunt.loadNpmTasks('grunt-contrib-uglify');
use ~~[illegible]~~ uglify]

```
grunt.initConfig({ — obj an argument.
    uglify : {
        my_target : {
            files : {
            'dest/output.min.js' :

                ['js/*.js']
                  all files
        }
      }
   }
```

[illegible] plugin   pkg : grunt
grunt.loadNpmTasks
grunt.registerTask('default', ['uglify'])



- release folder *

but now script tag will be simple

release/custom.min.js

grunt.registerTask('default', [uglify - my task])

- save
- run

grunt

- but not support latest

either change code

or plugins for ECMA6

grunt run

now new folder

release *

Invoke all in 1 line

&

change variables *

2

concat

installation

- load
- enter in grunt.initConfig

but again only add inner part *

Simple file - only to merge

uglify - concat ✓
- compress
- var change also

Yeoman

web - scaffolding Tool

→ npm install -g yo

run on CMD

→ add generators

→ if angular then — generator (of angular)

→ 9000 generators

→ angular

[generator - karma generator-angular

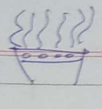
go to CMD → " new folder
, angularsample,

yo angular

on Terminal

ask would you like to use gulp NO

—— sass (with compass) — no
Bootstr — yes



view

route
sanitize -

Angular JS — fail

ionic - 1

heavy project

chief for mobile

folder
APP

→ del files

if anything new
we use commands

yo angular: controller
route

we
not

gitmuds/yeoman/
generators

not use new js

scaffold
etc

npm start

Scanned by CamScanner