Angular

Jalal Hejazi

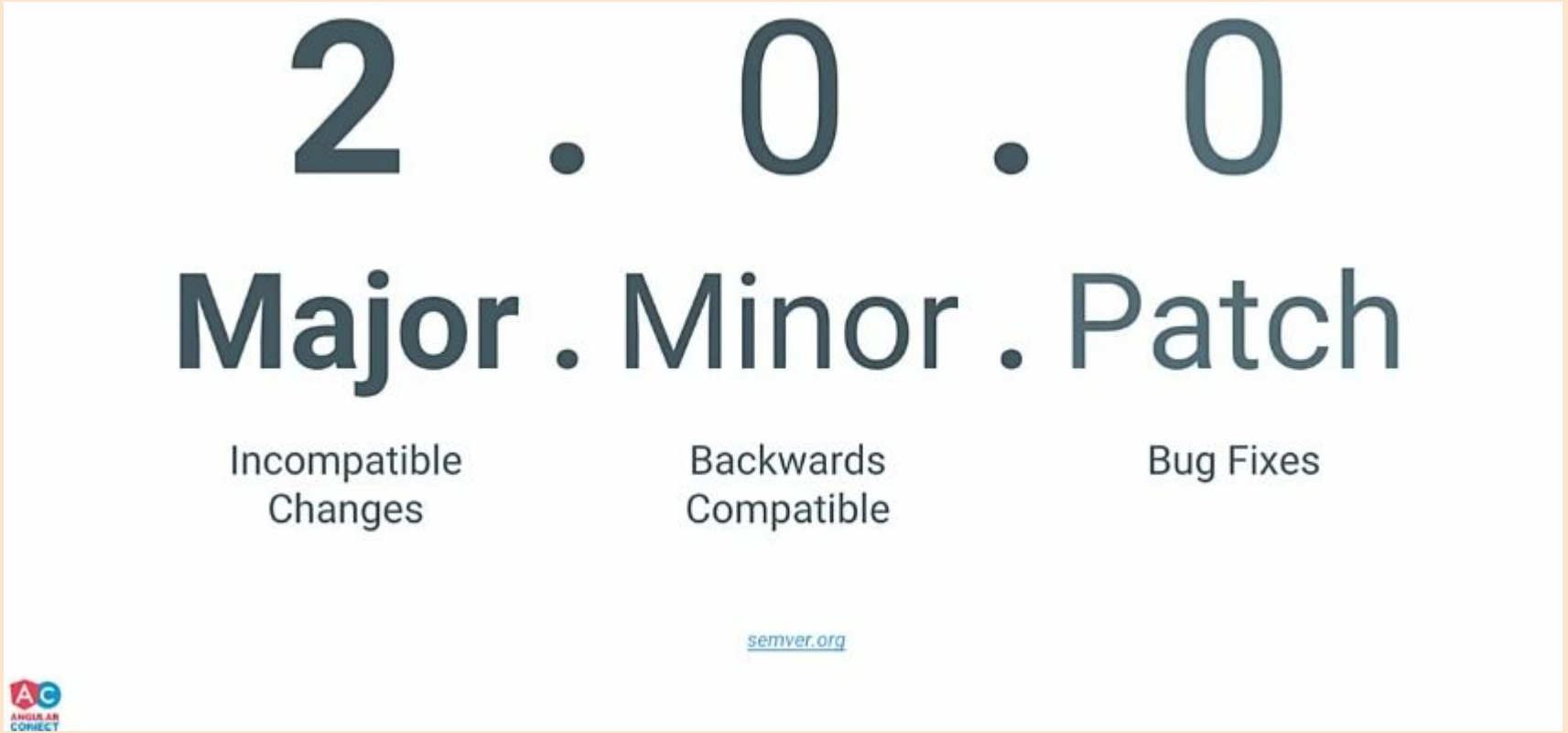
Indhold

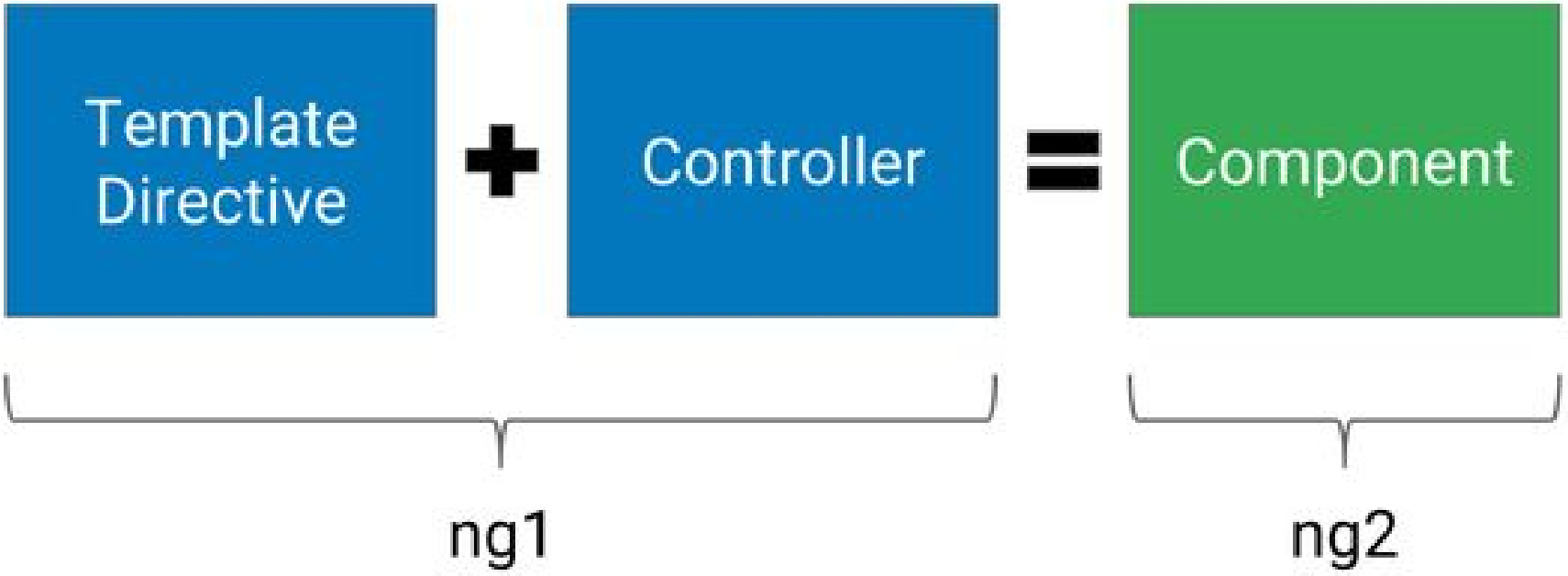
- About Angular 2 (v1 vs. v2 vs. V4)
- Training Path
- Angular Architecture
- angular-cli
- Workshops

About Angular 2

https://angularjs.org/

Angular Version 1 !== Angular version 2 (Incompatible Changes)





Google team har brugt erfaring fra ng1 for at kunne redesigne ng2

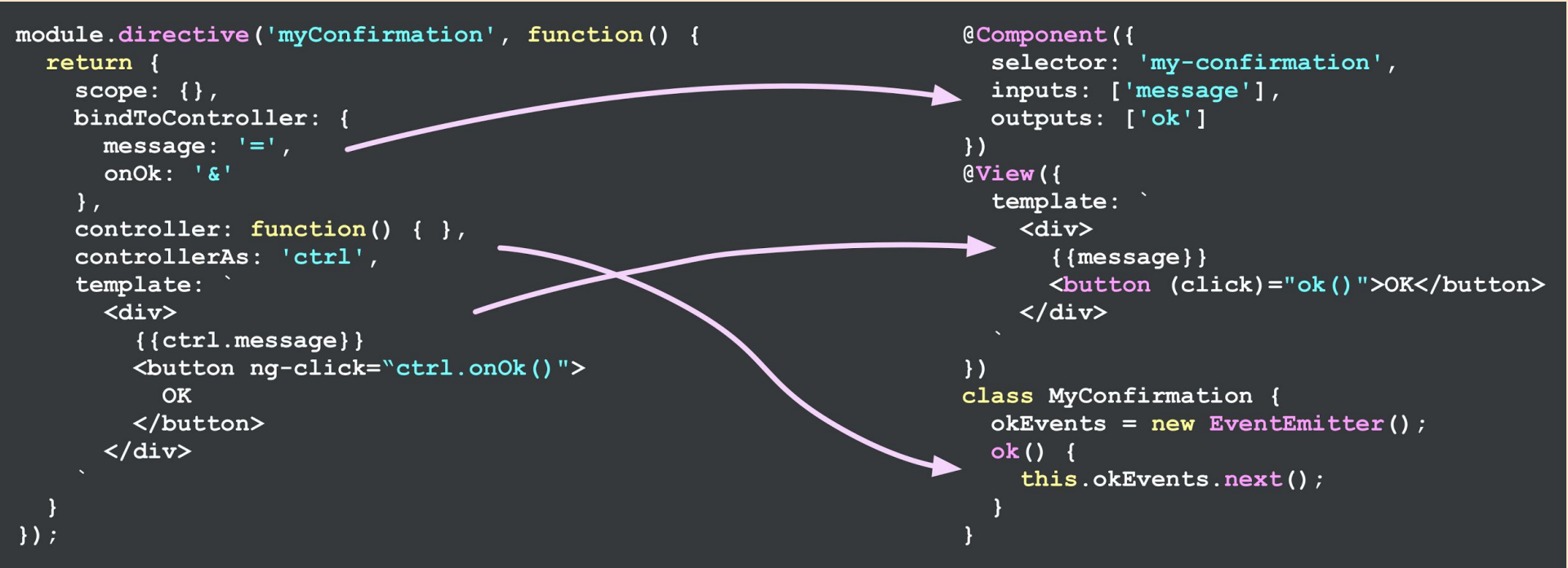
ng2 gøre det nemmere for webudviklere at bygge apps med komponenter :-)

Angular 2 is a framework for building client applications in HTML and either JavaScript or a language (like Dart or TypeScript) that compiles to JavaScript.

You write Angular applications by composing HTML templates with Angularized markup, writing component classes to manage those templates, adding application logic in services, and boxing components and services in modules.

Start learning Angular 1.x and move to Angular 2.x

EcmaScript 5 (JS) → EcmaScript 6 (JS) → TypeScript (JS)



```
module.directive('myConfirmation', function() {
  return {
    scope: {},
    bindToController: {
      message: '=',
      onOk: '&'
    },
    controller: function() { },
    controllerAs: 'ctrl',
    template: `
      <div>
        {{ctrl.message}}
        <button ng-click="ctrl.onOk()">
          OK
        </button>
      </div>
    `
  }
});
```

```
@Component({
  selector: 'my-confirmation',
  inputs: ['message'],
  outputs: ['ok']
})
@View({
  template: `
    <div>
      {{message}}
      <button (click)="ok()">OK</button>
    </div>
  `
})
class MyConfirmation {
  okEvents = new EventEmitter();
  ok() {
    this.okEvents.next();
  }
}
```

Hvorfor Angular 2

- Bedre support for Web Components
- Du bestemmer sproget
  - Plain old javaScript → ES5 → legacy sprog
  - Future javaScript → ES6/ES7 → moderne sprog
  - Strongly Typed javaScript → TypeScript → moderne sprog
  -
- Hvorfor ES6 og ikke ES5 ?
  - Class Oriented → Reusability → Components
  - Module loaders → Performance

- Hvorfor Module loaders ?
  - Cross platform support → angular universal
    - Mobile
    - Desktop
    - Web

Professional application requirement 2017

- a sensible project structure
- data binding
- master/detail
- services
- dependency injection
- navigation
- remote data access
- component design
- component reusability

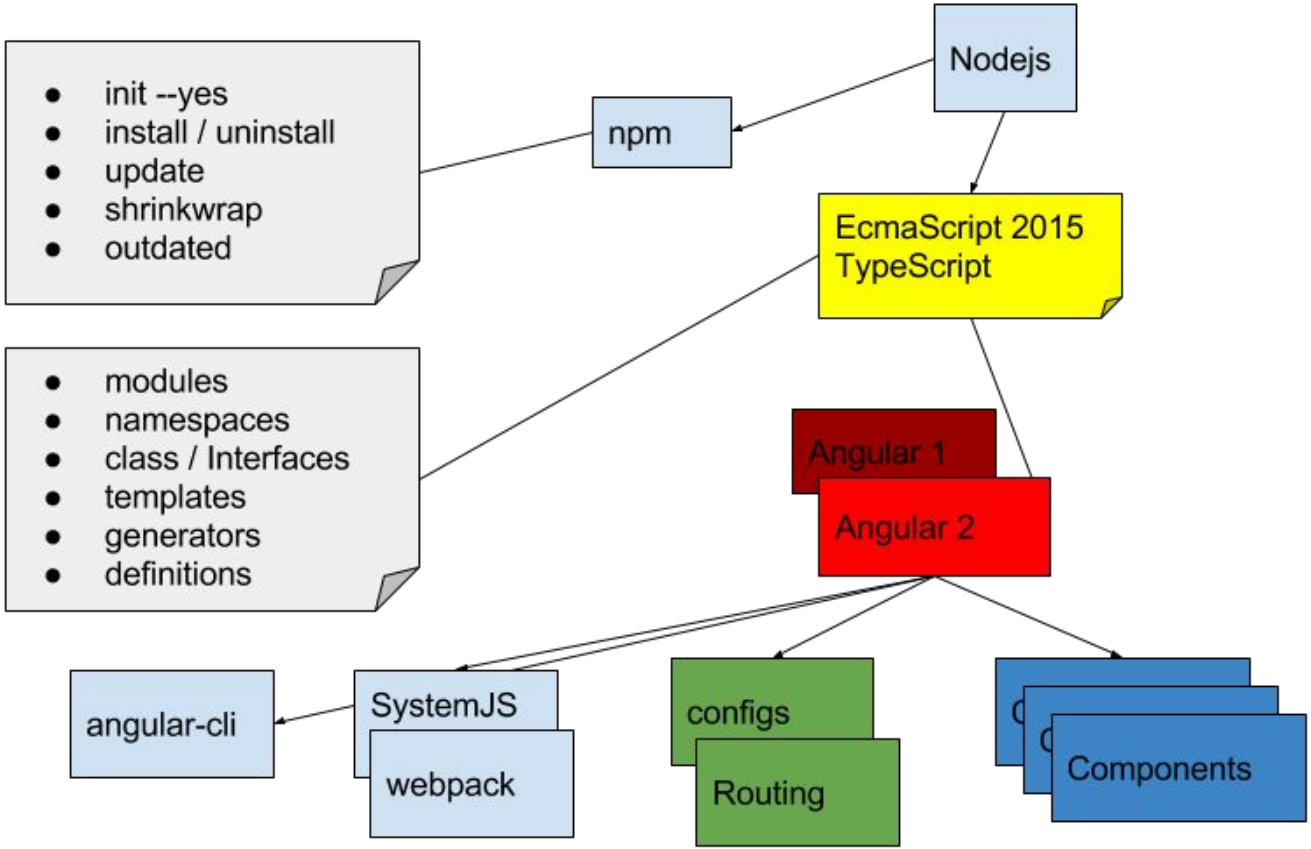
- Maintainability
- Testability
- Readability / easy to understand
- Performance optimized
- SEO optimized
- Cross platform
- Google Developers (trust)
- Community supported
- One Framework / One Language

Training Path

Learning Path

Learning Curve

1. Setup Dev Tools
2. javaScript on server side
3. node.js / npm → det er denne vej udvikling går i dag
   a. start learning node.js before it is too late :-)
   b. npm install packages → client and server dependency
   c. npm init → package.json
   d. npm link → the same as npm install --global
4. EcmaScript 6 → compile to ES5
5. TypeScript → compile to ES6/ES5
6. SystemJS / WebPack (System loaders)
7. Angular 1 → MVC + Directives
8. Angular 2 → Angular 1 + Components + [6 + 5 + 4 + 3 + 2 + 1]
9. Release-Build and Deployment (git, Nginx, PM2)



Learning Curve

Jalal Hejazi 2016

Hvor kan man læse om Angular 2 ?

http://book_ng.itacademy.dk/

http://book_ng.itacademy.dk/book.pdf

TODO: Setup Dev Tools

Learning Curve

http://kursus_js.itacademy.dk/SETUP.html

EcmaScript 5 vs. EcmaScript 6

Learning Curve

http://kursus_js.itacademy.dk/ecmascript.html

Learning TypeScript language

http://kursus_js.itacademy.dk/typescript.html

Learning Curve

Teori: **Webpack** (System loaders)

Hvorfor webpack ?

- Module loader
  - optimize loadtime
- Dependency Bundling
  - Performance
- Compilers
  - TypeScript → es5
  - Babel → es5
  - SASS/LESS → css

Learning Curve

TODO:

Opgaven webpack med es6

http://kursus_js.itacademy.dk/ecmascript_modules.html

Workshop: angular-v1-webpack

http://git.itacademy.dk/ecmascript/es6_democode/tree/master/21-angular-v1-webpack

- Webpack bundling
- ES6 modules
- ES6 class
- Angular.services
- Angular.directives
- Angular.routings
- Angular.configs
- Angular.controllers

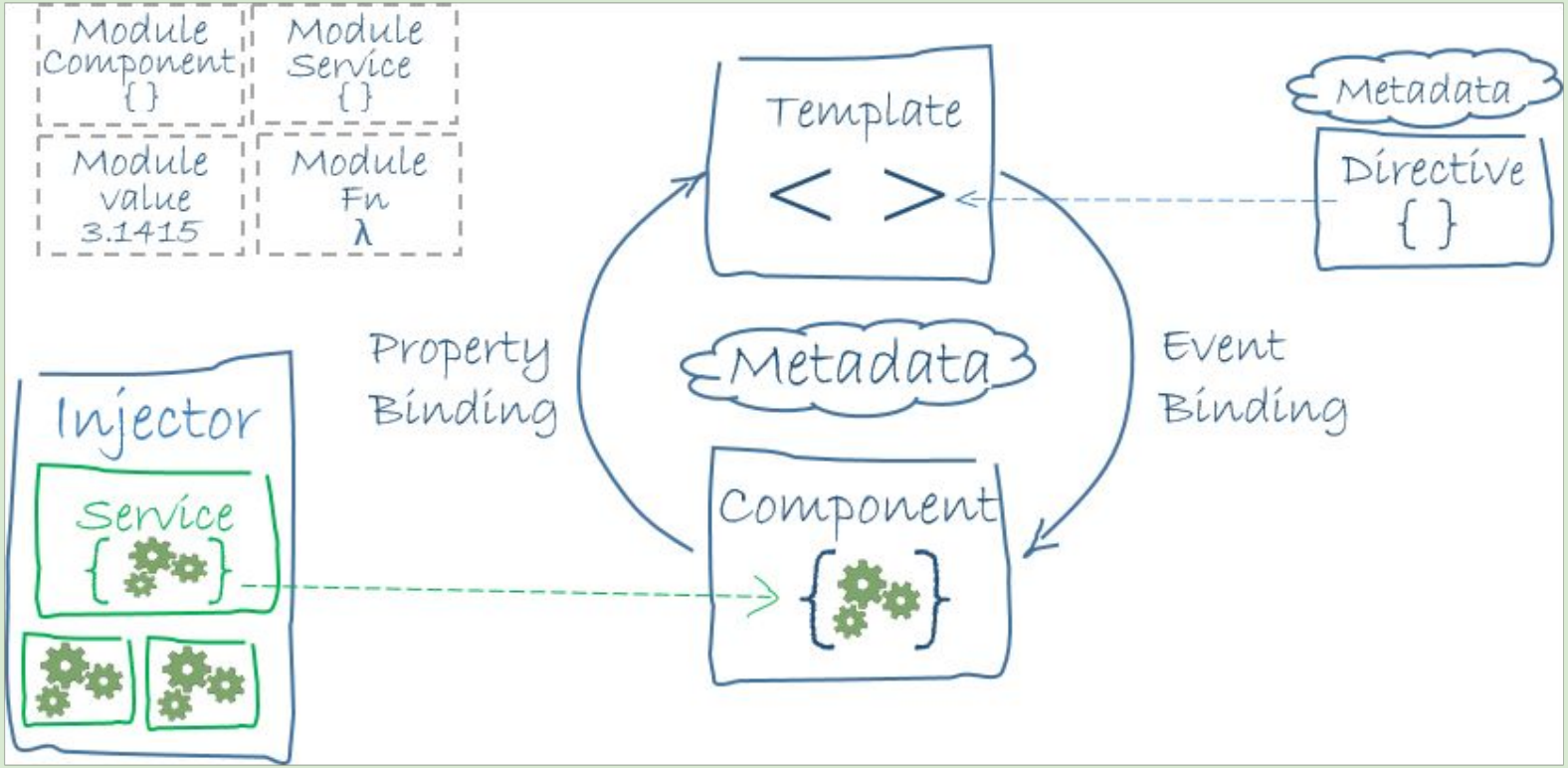
Teori: angular-v2 med TypeScript

http://kursus_js.itacademy.dk/angular_2_ts.html

Workshop: ng2 + typescript

http://git.itacademy.dk/ecmascript/angular-v2-using-components.git

Angular Architecture



http://devdocs.io/angular~2_typescript/guide/architecture

http://devdocs.io/angular~2_typescript/guide/architecture#modules

ANGULARJS

app/app.module.ts

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
  imports:      [ BrowserModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

Module
Component
{ }

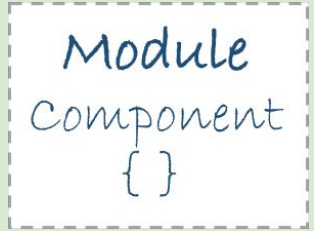
@NgModule is a component declared using export class AppModule { }

Every Angular app has at least one module, the *root module*, conventionally named `AppModule`
Apps can have one ngModule (root) or more modules for features

http://devdocs.io/angular~2_typescript/guide/architecture#modules

app/app.module.ts

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
  imports:      [ BrowserModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```



`NgModule` is a decorator function that takes a single metadata object whose properties describe the module. The most important properties are:

- `declarations` - the *view classes* that belong to this module. Angular has three kinds of view classes: components, directives, and pipes.
- `exports` - the subset of declarations that should be visible and usable in the component templates of other modules.
- `imports` - other modules whose exported classes are needed by component templates declared in *this* module.
- `providers` - creators of services that this module contributes to the global collection of services; they become accessible in all parts of the app.
- `bootstrap` - the main application view, called the *root component*, that hosts all other app views. Only the *root module* should set this `bootstrap` property.

http://exploringjs.com/es6/ch_modules.html

ANGULARJS

JavaScript has had modules for a long time. However, they were implemented via libraries, not built into the language.
ES6 is the first time that JavaScript has built-in modules.

There can be multiple *named exports*:

```
//------ lib.js ------
export const sqrt = Math.sqrt;
export function square(x) {
    return x * x;
}
export function diag(x, y) {
    return sqrt(square(x) + square(y));
}

//------ main.js ------
import { square, diag } from 'lib';
console.log(square(11)); // 121
console.log(diag(4, 3)); // 5
```

You can also import the complete module:

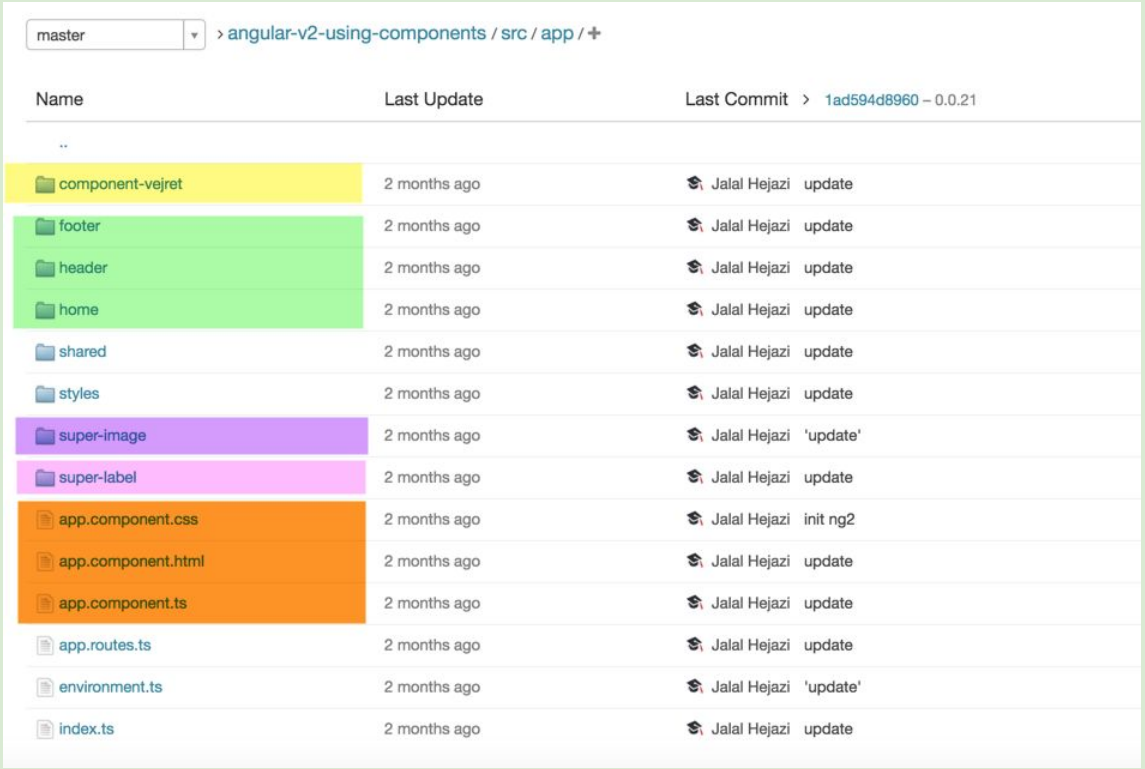
```
//------ main.js ------
import * as lib from 'lib';
console.log(lib.square(11)); // 121
console.log(lib.diag(4, 3)); // 5
```

Module
Component
{ }

ANGULARJS

Features:

- @Components
- @Input
- @output
- Bindings
- `[(ngModel)]`
- `@angular/router`
- `ngOnInit()`
- `ngOnDestroy()`
- `onClickEvent()`
- `onMouseOverEvent()`
- ……



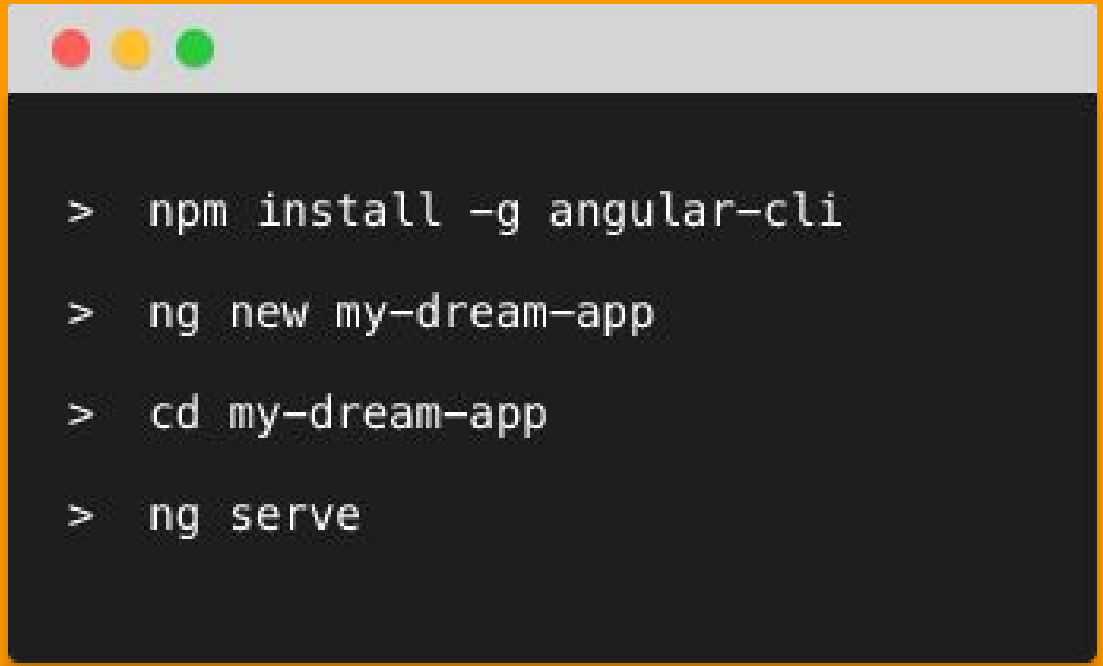
angular-cli

npm install

For Angular version 1.x → **npm install --global angcli**

For Angular version 2.x → **npm install --global angular-cli**

npm home angular-cli



ANGULARJS

You can find all possible blueprints in the table below:

Scaffold	Usage
Component	`ng g component my-new-component`
Directive	`ng g directive my-new-directive`
Pipe	`ng g pipe my-new-pipe`
Service	`ng g service my-new-service`
Class	`ng g class my-new-class`
Interface	`ng g interface my-new-interface`
Enum	`ng g enum my-new-enum`
Module	`ng g module my-module`

angular-cli@version

npm i -g angular-cli@1.0.0-beta.11-webpack.9-4

Development Hints for hacking on angular-cli

Working with master

```
git clone https://github.com/angular/angular-cli.git
cd angular-cli
npm link
```

`npm link` is very similar to `npm install -g` except that instead of downloading the package from the repo, the just cloned `angular-cli/` folder becomes the global package. Any changes to the files in the `angular-cli/` folder will immediately affect the global `angular-cli` package, allowing you to quickly test any changes you make to the cli project.

Now you can use `angular-cli` via the command line:

```
ng new foo
cd foo
npm link angular-cli
ng serve
```

`npm link angular-cli` is needed because by default the globally installed `angular-cli` just loads the local `angular-cli` from the project which was fetched remotely from npm. `npm link angular-cli` symlinks the global `angular-cli` package to the local `angular-cli` package. Now the `angular-cli` you cloned before is in three places: The folder you cloned it into, npm's folder where it stores global packages and the `angular-cli` project you just created.

You can also use `ng new foo --link-cli` to automatically link the `angular-cli` package.

Please read the official npm-link documentation and the npm-link cheatsheet for more information.

package.json

How to load dependency from npm

unpkg is a fast, global content delivery network for stuff that is published to npm. Use it to quickly and easily load files using a simple URL like:

```
https://unpkg.com/package@version/file
```

A few examples:

- https://unpkg.com/react@15.3.1/dist/react.min.js
- https://unpkg.com/react-dom@15.3.1/dist/react-dom.min.js
- https://unpkg.com/history@4.2.0/umd/history.min.js

Use npm package.json

package.json is your data for “About page”

http://eval.superusers.dk/about

Angular 1.x

- https://unpkg.com/angular@latest/package.json
- https://unpkg.com/angular@1.5.8/package.json

Angular 2.x

- https://unpkg.com/@angular/core@2.1.0/package.json
- https://unpkg.com/@angular/http@2.1.0/package.json
- https://unpkg.com/@angular/common@2.1.2/package.json
- https://unpkg.com/@angular/platform-browser-dynamic@2.1.2/package.json
-

Others:

- https://unpkg.com/jquery@latest/package.json
- https://unpkg.com/moment@latest/package.json
- https://unpkg.com/bootstrap@latest/package.json
- https://unpkg.com/lodash@latest/package.json

DEMO: http://git.itacademy.dk/angularjs/ng_version2_formbuilder.git


```
const angularVersion = '2.0.0-rc.6';

System.config({
  baseUrl: '/',
  paths: {
    'unpkg:*': 'https://unpkg.com/*'
  }
});

System.config({
  transpiler: 'typescript',
  typescriptOptions: { emitDecoratorMetadata: true },

  meta: {
    '*': {
      deps: [ 'zone.js', 'reflect-metadata' ]
    }
  }
});

System.config({
  packageConfigPaths: [
    "unpkg:@*/*/package.json"
  ],

  map: {
    '@angular/core': 'unpkg:@angular/core@'+angularVersion,
    '@angular/compiler': 'unpkg:@angular/compiler@'+angularVersion,
    '@angular/common': 'unpkg:@angular/common@'+angularVersion,
    '@angular/forms': 'unpkg:@angular/forms@'+angularVersion,
    '@angular/platform-browser': 'unpkg:@angular/platform-browser@'+angularVersion,
    '@angular/platform-browser-dynamic': 'unpkg:@angular/platform-browser-dynamic@'+angularVersion,
    '@angular/http': 'unpkg:@angular/http@'+angularVersion,
    'rxjs': 'unpkg:rxjs@5.0.0-beta.11',
    'zone.js': 'unpkg:zone.js@0.6.17',
    'reflect-metadata': 'unpkg:reflect-metadata@0.1.3',
    "crypto": "@empty"
  },

  packages: {
    'app': {
      defaultExtension: 'ts',
      main: './index.ts'
    }
  }
});
```