Par NIDHAL JELASSI
nidhal.jelassi@fsegt.utm.tn

PROGRAMMATION WEB AVANCÉE

ANGULAR

Chapitre 1 : Introduction et Environnement

PRÉ-REQUIS

- HTML 5 / CSS 3

- JavaScript

- Programmation Orientée Objet

POO

PLAN DU COURS

Partie 1 :

- Présentation de Angular.
- ECMAScript 6.
- Découvrir TypeScript.

Partie 2 :

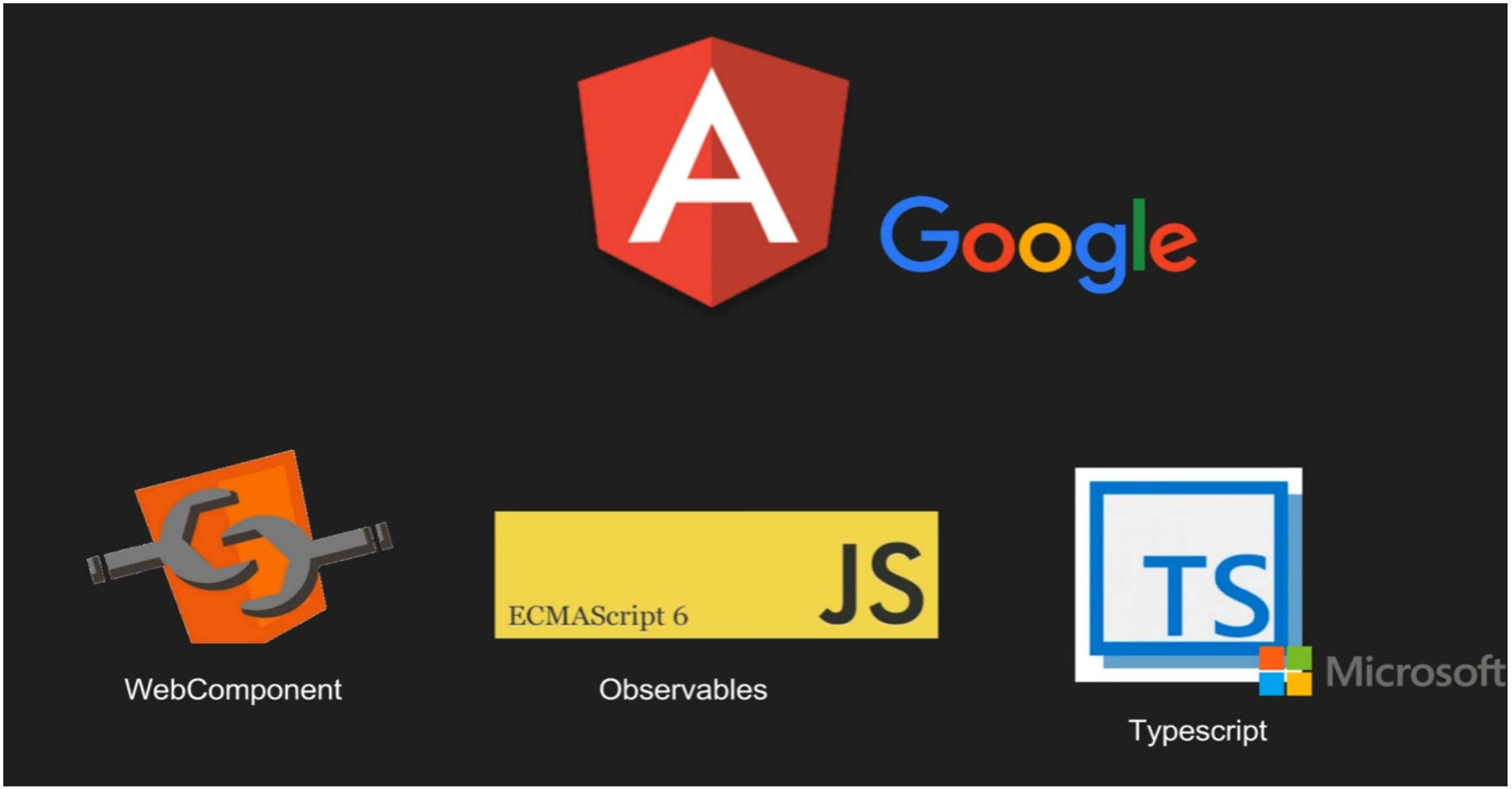
- Les composants.
- Les templates.
- Les directives.
- Les pipes.
- Les routes.
- Les services et l'injection de dépendances.

PLAN DU COURS

Partie 3 :

- Formulaires pilotés par le template.
- Formulaires pilotés par le modèle.
- La programmation réactive.
- Effectuer des requêtes HTTP standards.
- Effectuer des traitements asynchrones avec RxJS.
- Authentification.
- Backend avec Loopback
- Déployer une application.

PRÉSENTATION D'ANGULAR



PRÉSENTATION D'ANGULAR

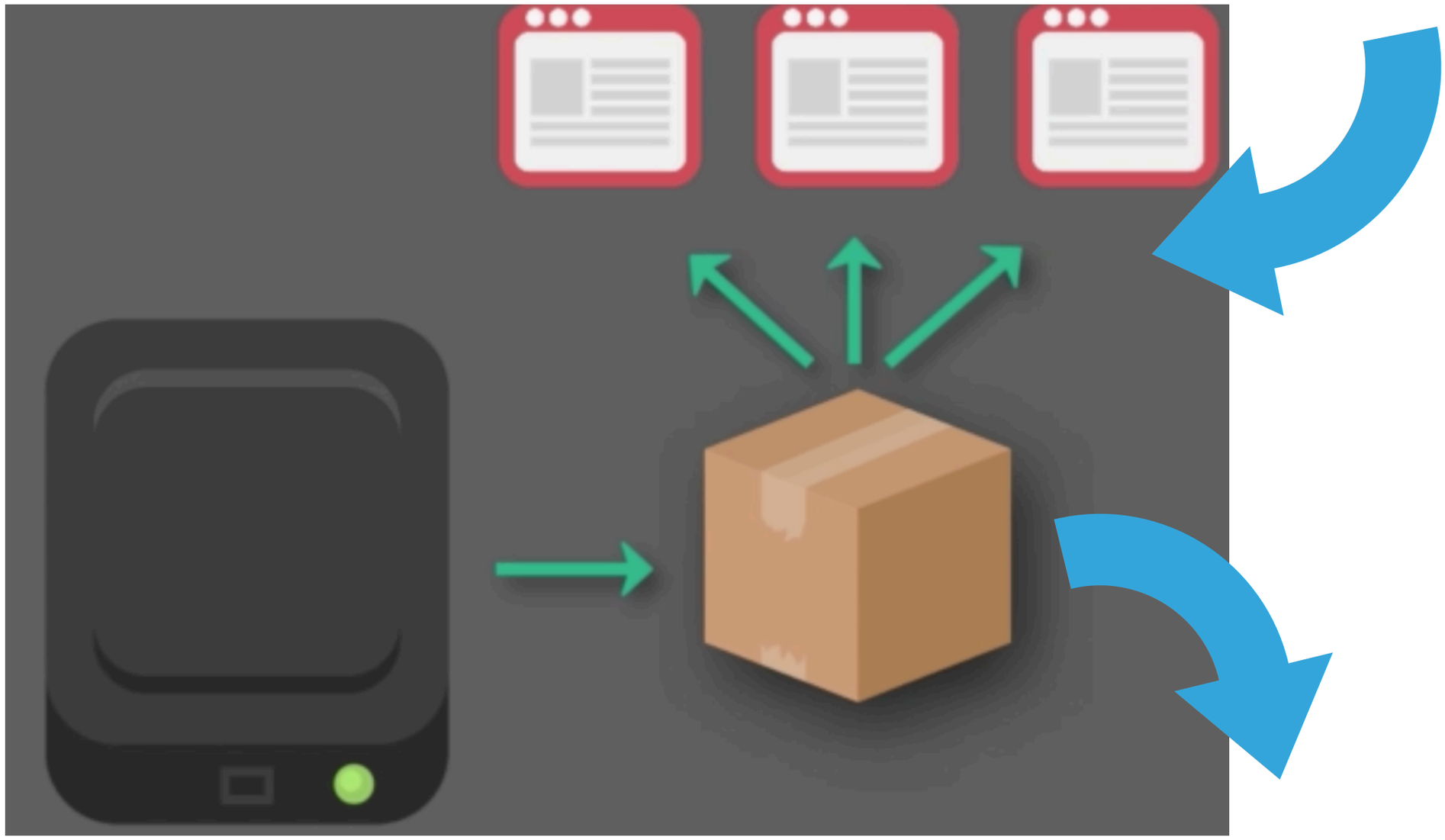
- Angular est un framework (cadre de travail)
- Permet de développer des sites web, des applications web, des applications mobiles hybrides, de manière robuste et efficace.
- Supporte plusieurs langages : ES5, TypeScript, Dart
- Développé par Google en 2009
- AngularJS vs Angular
- Principaux concurrents : Vue.Js et ReactJS

SITE WEB

Serveur

APPLICATION WEB

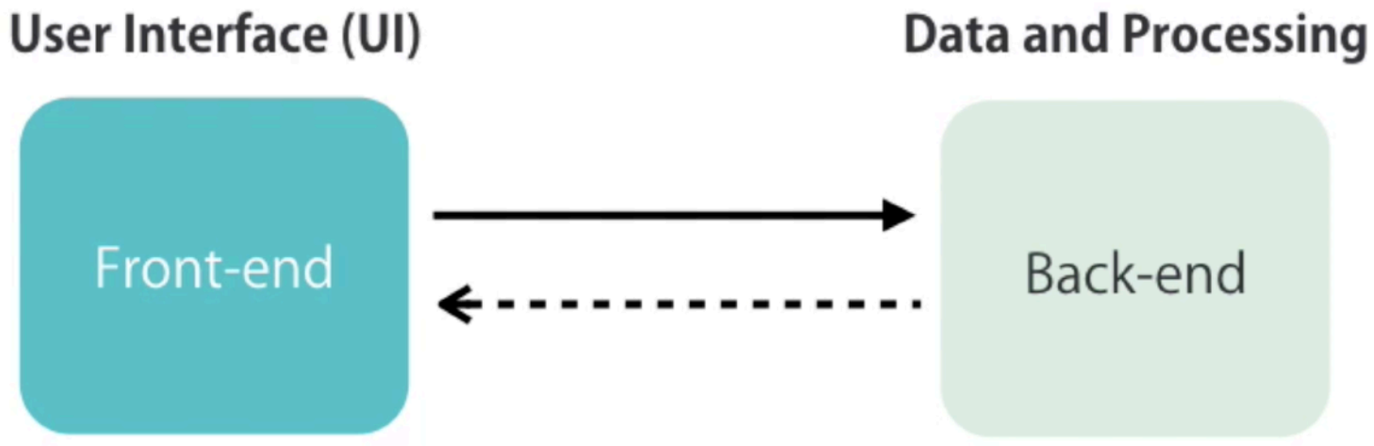
Intervention du JS



Serveur

Application web en SPA
(Single Page Application)

FRONT VS BACK



HTML
CSS
TypeScript
Angular

Front-end Development

HTML
CSS
JavaScript
Angular

Back-end Development

APIs
Databases

PRINCIPALES NOUVEAUTÉS

A partir de la version 5 :

- Les contrôleurs : L'architecture de base MVC est remplacé par une architecture réactive à base de composants web (MVW).
- Les directives : 3 nouveaux types de directives : les composants, les directives d'attributs, les directives structurelles
- Scope : Simplification d'utilisation
- Modules : Modules natives d'ES 6
- JQlite (version plus légère de JQuery) retirée
- 2-ways data-binding de base n'est plus disponible de base (mais reste implémentable)

PHILOSOPHIE

- Angular est un langage ***orienté composants***
- Une application Angular est donc un ensemble de petits composants assemblés entre-eux.
- Un composant est l'assemblage d'un morceau de code HTML et d'une classe JavaScript dédiée à une tache particulière.
  - **Classe… JavaScript ?!?**
- Ces composants reposent sur le standard des **web components** (pas encore supporté par tous les navigateurs). Ce standard a été pensé pour découper une page web en fonction de leurs rôles (navbar, discussion, etc…).
- Un composant est censé être une partie qui fonctionne de manière autonome dans une application.

ECMASCRIPT 6 .. ECMASCRIPT 2015 .. ES6 .. JAVASCRIPT 6

- ECMAScript 6 est le nom de la dernière version standardisé de JavaScript.
- Cette standardisation a été approuvé par l'organisme de normalisation en Juin 2015
- Le besoin d'une nouvelle standardisation s'est fait sentir pour fournir à JavaScript les moyens de développer des applications web robustes.
- ES6, via sa nouvelle syntaxe, apporte plusieurs nouvelles évolutions à JavaScript.

NOUVEAUTÉS DE ES6

- Les **classes**
- Le nouveau mot-clef **let** : Le mot-clé let permet de déclarer une variable locale, dans le contexte où elle a été assignée.
- Le nouveau mot-clef **const**
- Les **fonctions fléchées** : Les fonctions fléchées, ou arrow functions, ne sont pas des fonctions classiques, parce qu'elles ne définissent pas un nouveau contexte comme les fonctions traditionnelles. Nous les utiliserons beaucoup dans le cadre de la programmation asynchrone qui nécessite beaucoup de fonctions anonymes.
- Les **paramètres** de fonctions **par défaut**
- La syntaxe template **string** : on peut désormais utiliser des templates strings, qui commencent et se terminent par un backtick `.

TYPESCRIPT

- *« TypeScript est un langage de programmation libre et open-source, développé par Microsoft, qui a pour but d'améliorer et de sécuriser la production de code JavaScript. C'est un sur-ensemble de JavaScript (c'est-à-dire que tout code Javascript correct peut être utilisé avec TypeScript). Le code TypeScript est transcompilé en JavaScript, pouvant ainsi être interprété par n'importe quel navigateur web ou moteur JavaScript ».*

- Il est chaudement recommandé d'utiliser TypeScript pour développer en Angular. C'est ce que Google recommande explicitement dans la documentation officielle d'Angular.

- En bonus, Angular lui-même est développer avec TypeScript. C'est pour cela que nous utiliserons TypeScript dans ce cours.

TYPESCRIPT : PRINCIPALES CARACTÉRISTIQUES

- **Le typage** : Voici comment typer nos variables avec TypeScript:

```
var nb: number = 100;
var nom: string = 'Nidhal Jelassi';
```

- **TypeScript et les fonctions** : On peut aussi typer les paramètres et la valeur de retour d'une fonction, toujours avec la syntaxe des deux points:

```
function creerCours(nb: number, nom: string): Cours {
      var cours = new Cours();
      heros.nb = nb;
      heros.nom = nom;
      return Cours; }
```

- **Les décorateurs** : Les annotations TypeScript permettent d'ajouter des informations sur nos classes, pour indiquer par exemple que telle classe est un composant de l'application, ou telle autre un service. On utilise @ comme syntaxe:

```
@Component({
      selector: 'mon-composant',
      template: 'mon-template.html'
    }) export class MonComposant {}
```

ANGULAR

- Angular est modulaire
  - Chaque application va définir Angular Modules or NgModules
    - Chaque module Angular est une classe avec une annotation @NgModule
    - Chaque application a au moins un module, c'est le module principale.
- Rapide
- Orienté Composant

ANGULAR

- Le module principal est le module qui permet de lancer l'application.
- Le nom par convention est **AppModule**.
- L'annotation (decorator) **@NgModule** identifie AppModule comme un module Angular.
- L'annotation prend en paramètre un objet spécifiant à Angular comment compiler et lancer l'application.
  - **imports** : tableau contenant les modules utilisés.
  - **declarations** : tableau contenant les composants, directives et pipes de l'application.
  - **bootstrap** : indique le composant exécuter au lancement de l'application.
- Il peut y avoir aussi d'autres attributs dans cet objet comme provider

LES LIBRAIRIES D'ANGULAR

- Ensemble de modules JS
- Des librairies qui contiennent un ensemble de fonctionnalités.
- Toutes les librairies d'Angular sont préfixées par @angular
- Récupérable à travers un import JavaScript.
- Exemple pour récupérer l'annotation component :

```
import { Component } from '@angular/core';
```

COMPOSANTS – TEMPLATES

- Le composant est la partie principale d'Angular.
- Un composant s'occupe d'une partie de la vue.
- L'interaction entre le composant et la vue se fait à travers une API.

- Un Template est le complément du composant.
- C'est la vue associée au composant.
- Elle représente le code HTMl géré par le composant.

META DATA

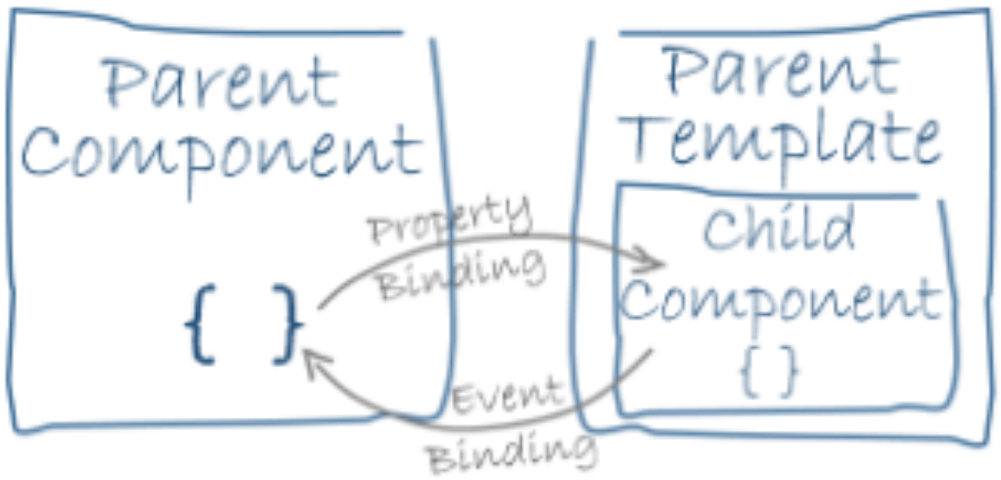
- Appelé aussi « decorator », ce sont des informations permettant de décrire les classes.

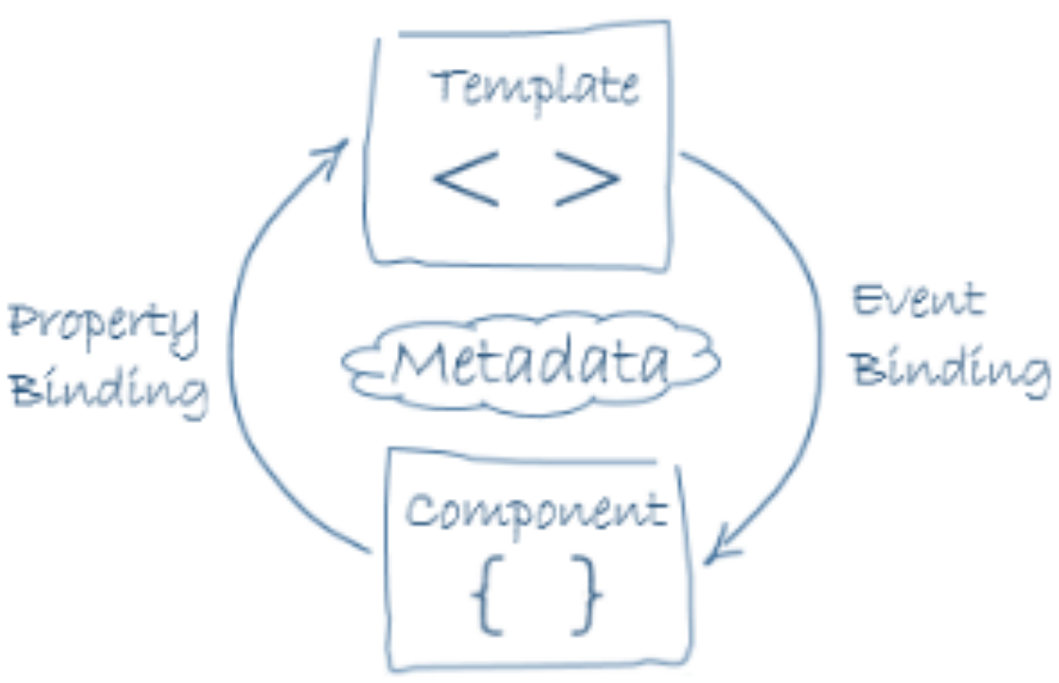
- @Component permet d'identifier la classe comme étant un composant angular.

DATA BINDING

- Le Data Binding est le mécanisme qui permet de mapper des éléments du DOM avec des propriétés et des méthodes du composant.

- Le Data Binding permettra aussi de faire communiquer les composants.





DIRECTIVES – SERVICES

- Les directives Angular sont des classes avec la métadata @Directive. Elle permettent de modifier le DOM et de rendre les Template dynamiques.

- Apparaissent dans des éléments HTML comme les attributs.

- Un composant est une directive à laquelle Angular a associé un Template.

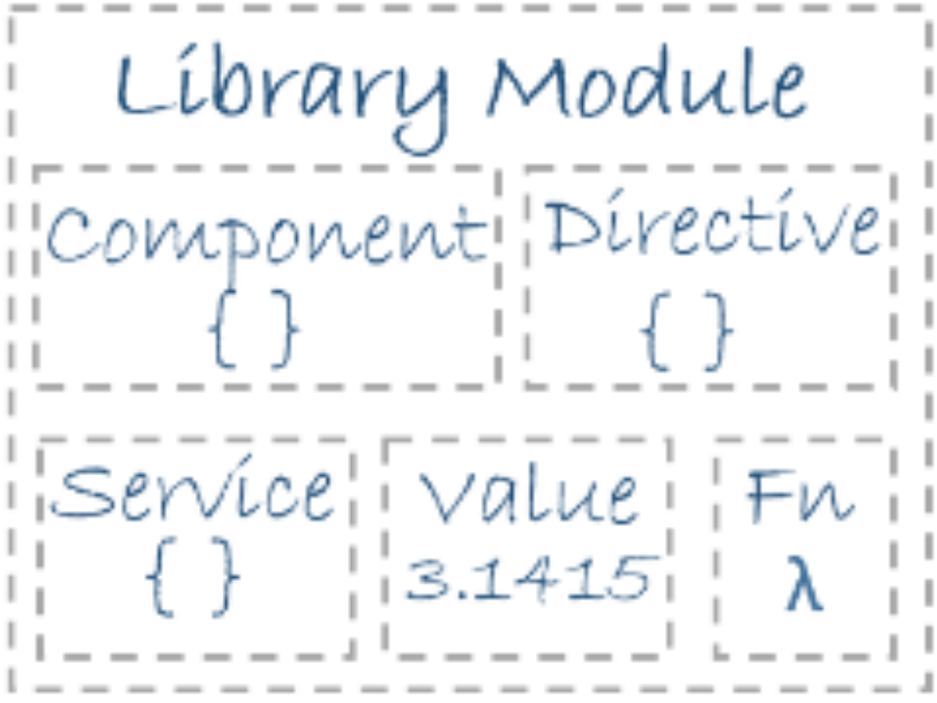
- Ils existe deux autres types de directives : Directives structurelles, Directive d'attributs.

- Classes permettant d'encapsuler des traitements métiers. Doivent être légers.

- Associées aux composants et autres classes par injection de dépendances.

LIBRAIRIES

- Ensemble de modules JS
- Des librairies qui contiennent un ensemble de fonctionnalités.
- Toutes les librairies d'Angular sont préfixées par @angular
- Récupérable à travers un import JavaScript.
- Exemple pour récupérer l'annotation component :
- import { Component } from '@angular/core';



ENVIRONNEMENT DE DÉVELOPPEMENT

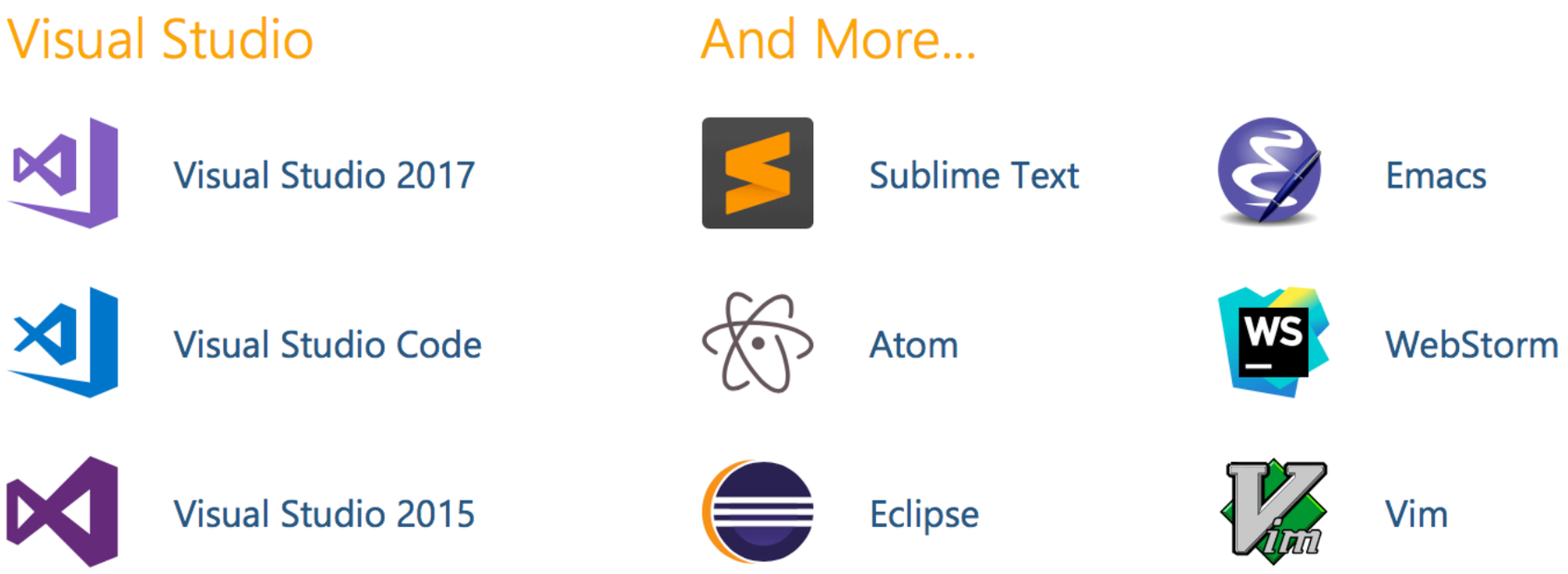
- Commençons par installer la dernière version de Node depuis **https://nodejs.org/en/**
  - Node : Environnement d'exécution pour programmes en JS à l'extérieur du navigateur.
- Vérifier l'installation en tapant : $node --version
- Ceci nous permettra d'utiliser Node Package Manager (**NPM**) pour pouvoir utiliser des librairies tierces telle que **Angular CLI (Command Line Interface)**
- **$(sudo)* npm install -g @angular/cli**
- **$ng —-version**

QUELQUES COMMANDES DU CLI

Commande	Utilisation
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
Module	ng g module my-module

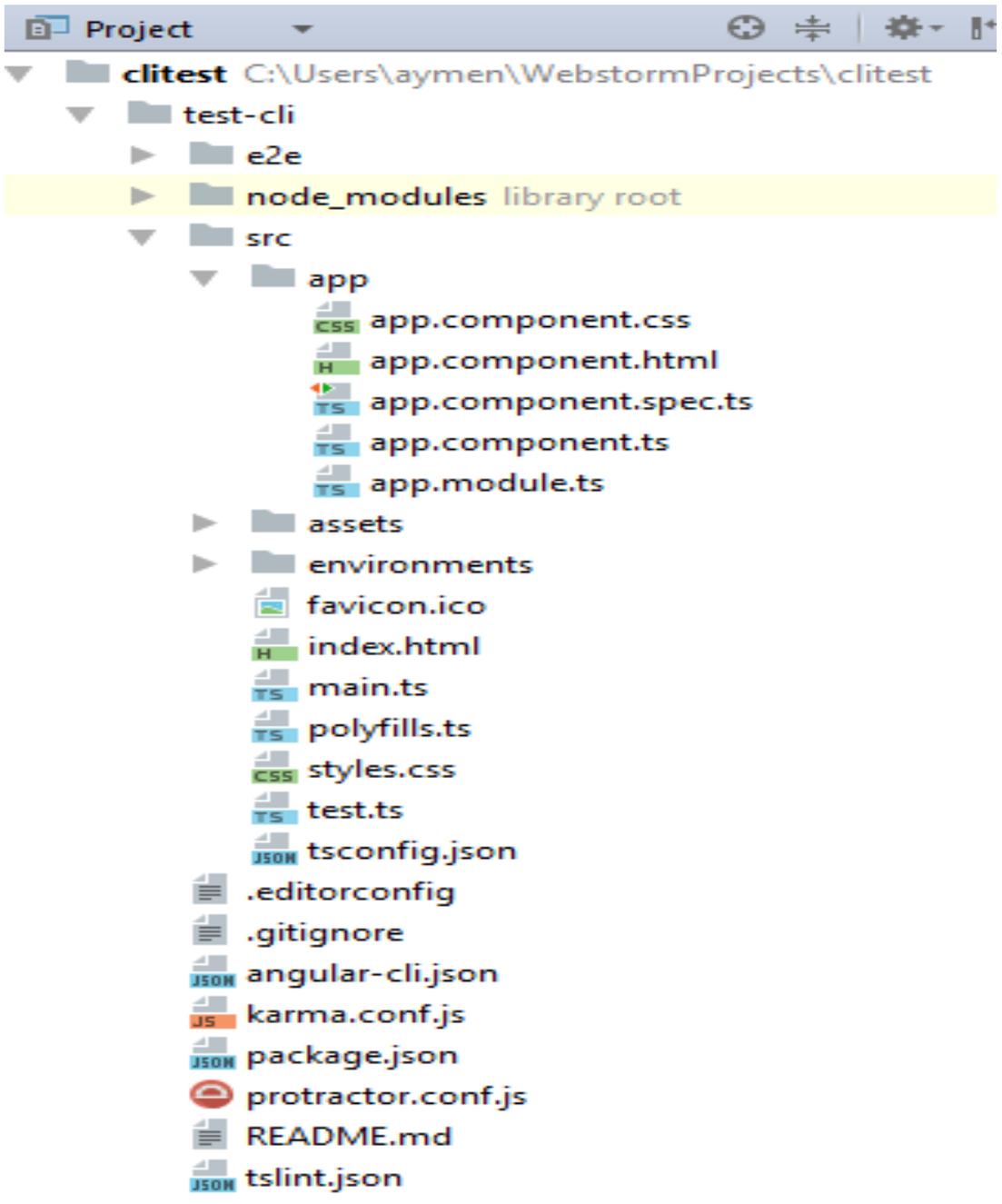
ENVIRONNEMENT DE DÉVELOPPEMENT

- Tapons dans le terminal : **$ng new FirstApp**
- Vérifions que notre premier projet s'exécute avec **$ng serve**
- Il nous reste à choisir un éditeur. Faites votre choix…



- … Néanmoins, je vous recommande Visual Code

ARBORESENCE D'UN PROJET



- **E2e** : end to end pour réaliser des tests d'intégration
- **Node_modules** : les dépendences
- **Src** : dossier contenant l'index le code source les styles, main.ts qui est le bootstrap d'angular ainsi que d'autres fichiers
- **App** : Les sources du projet par défaut ainsi que le module appModule, le fichier app.component.ts ainsi que son template (app.component.html), son style (app.component.css) et app.component.spec.ts pour les tests unitaires.

ARBORESENCE D'UN PROJET

- On a besoin au minimum d'un module racine et d'un composant racine par application.
- Le module racine se nomme par convention AppModule.
- Le composant racine se nomme par convention AppComponent.
- L'ordre de chargement de l'application est le suivant : index.html > main.ts > app.module.ts > app.component.ts.
- Le fichier package.json initial est fourni avec des commandes prêtes à l'emploi comme la commande npm start (ou ng serve), qui nous permet de démarrer notre application web.

WEBPACK

- Terminal

```
Date: 2019-02-10T12:56:11.435Z
Hash: 8e7863a6c7ec4def8f79
Time: 9294ms
chunk {es2015-polyfills} es2015-polyfills.js, es2015-polyfills.js.map (es2015-polyfills) 2
83 kB [initial] [rendered]
chunk {main} main.js, main.js.map (main) 11.5 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 235 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.08 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 16.3 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.75 MB [initial] [rendered]
i ｢wdm｣: Compiled successfully.
```

- Navigateur

```
▼ <body>
  ▶ <app-root _nghost-c0 ng-version="7.1.4">…</app-root>
    <script type="text/javascript" src="runtime.js"></script>
    <script type="text/javascript" src="polyfills.js"></script>
    <script type="text/javascript" src="styles.js"></script>
    <script type="text/javascript" src="vendor.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </body>
```

AJOUTER BOOTSTRAP À VOTRE PROJET

- On peut ajouter Bootstrap de plusieurs façons. La plus recommandée est la suivante :

1- Via npm avec la commande $npm install bootstrap --save

2- Ensuite, on ajoute dans le chemin des dépendances dans les tableaux styles et scripts dans le fichier angular.json:

```
"styles": [
"../node_modules/bootstrap/dist/css/bootstrap.min.css",
"styles.css",
],
"scripts": [
"../node_modules/bootstrap/dist/js/bootstrap.min.js"
],
```

AJOUTER BOOTSTRAP À VOTRE PROJET

- 2ème méthode : Vous pouvez aussi ajouter dans le fichier src/style.css un import de vos bibliothèques.

- @import "~bootstrap/dist/css/bootstrap.min.css";

- Essayer la même chose avec font-awesome qu'on pourra également utiliser dans notre projet.