# LE FRAMEWORK ANGULAR

Initiation par la pratique

# 1

# ENVIRONNEMENT DE TRAVAIL

# Outils nécéssaires

- **NodeJS:** Toolbox pour dev et réseaux
  - https://nodejs.org/en/download/ (LTS)


- **VSCode:** IDE multi langage gratuit
  - https://code.visualstudio.com/download
  - Plugin Angular Essential
  - Config settings (Auto Save + onFocusChange)
  - Config settings (Icons): material-icon-theme

# GIT

- **Github/Gitlab :** Création d'un compte + Repository
- **Git Bash :** Récupération du projet (*git clone* )
- **Commandes principales GIT :**
  - git clone: Clonage du repository distant
  - git status: Visualisation des fichiers modifiés
  - git add (-A ou .): Ajout des fichiers à l'index
  - git commit (-m '...'): Stockage des fichiers ajoutés (local)
  - git push: Envoie des modications locales au serveur
  - git pull: Récupération des données du serveur distant
  - git config --global user.name "*name*"
  - git config --global user.email "*mail* "

# 2

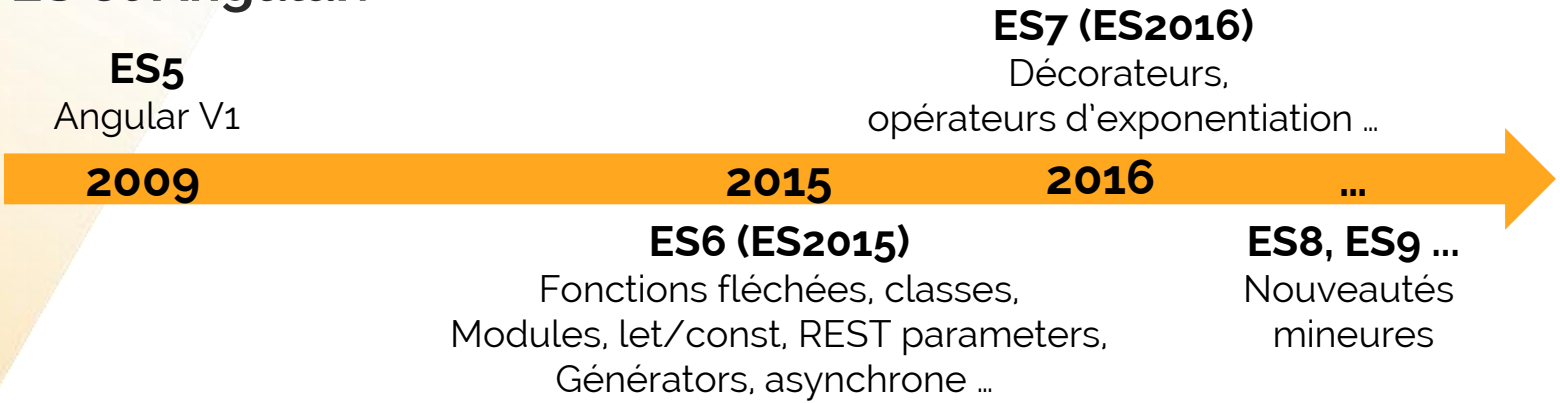
# INTRODUCTION

# Normes ES

## ECMASCRIPT ?

- Standards / Normes JavaScript
- Permet de définir la **syntaxe**, les **type** de variables …

## ES et Angular:



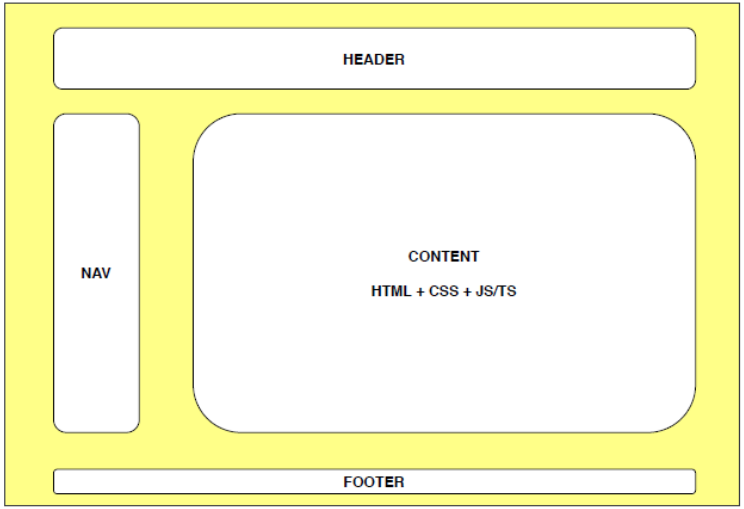
# Historique Angular

- Angular, un framework ?

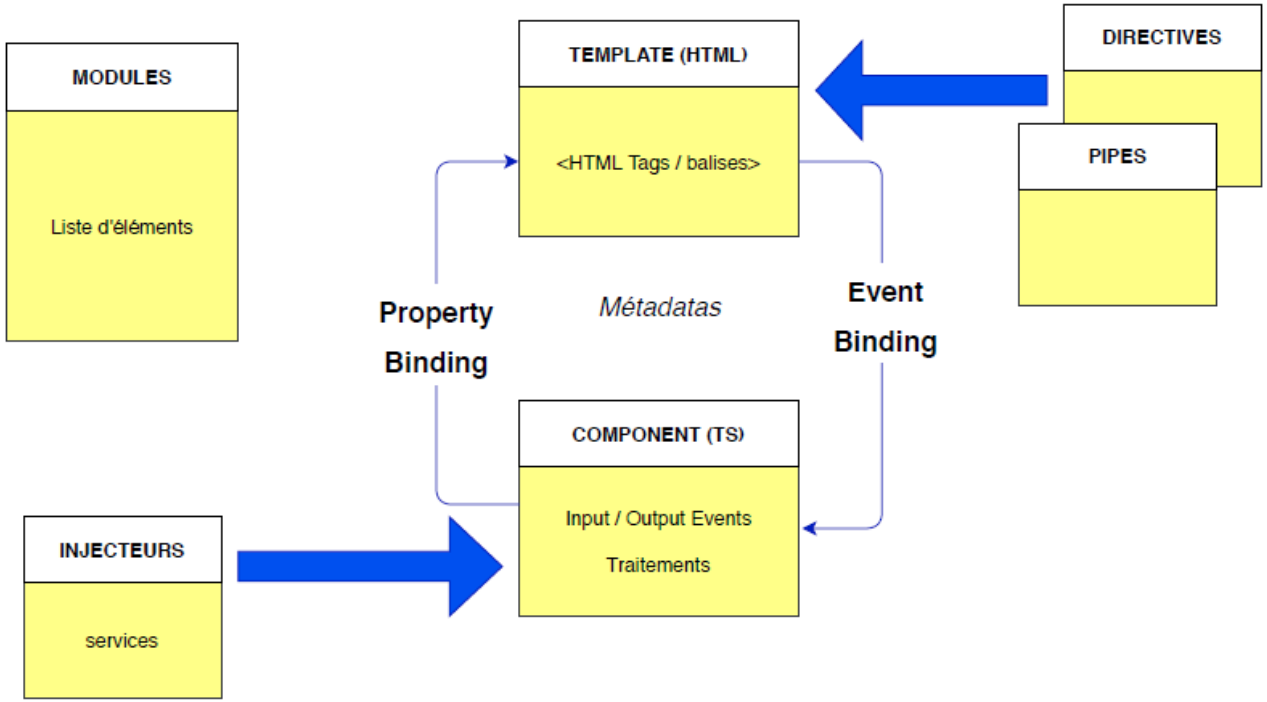
- AngularJS (2010) à Angular 2 (2016)

- Termes Clés :
  - TypeScript
  - Décorateurs
  - WebComponents

**WebComponents**

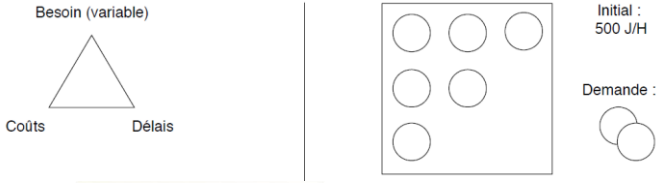


# L'architecture Angular



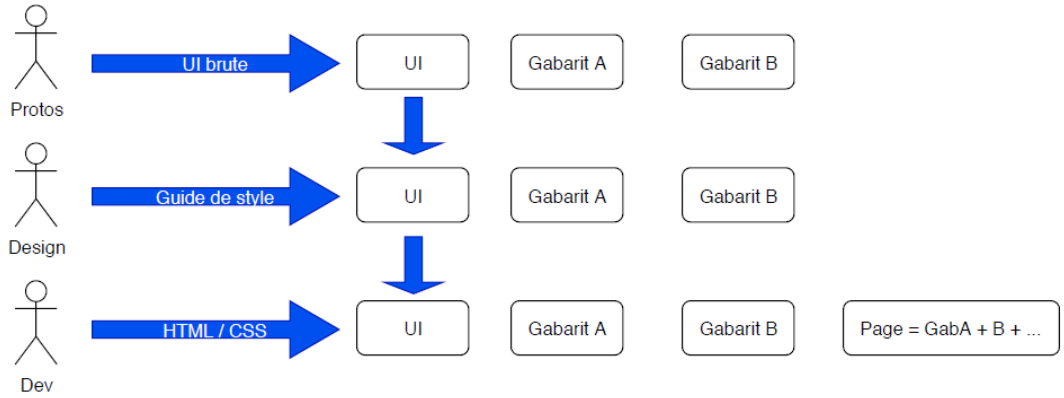
# De l'agilité au logiciel (1/2)

## Principes de l'agilité :



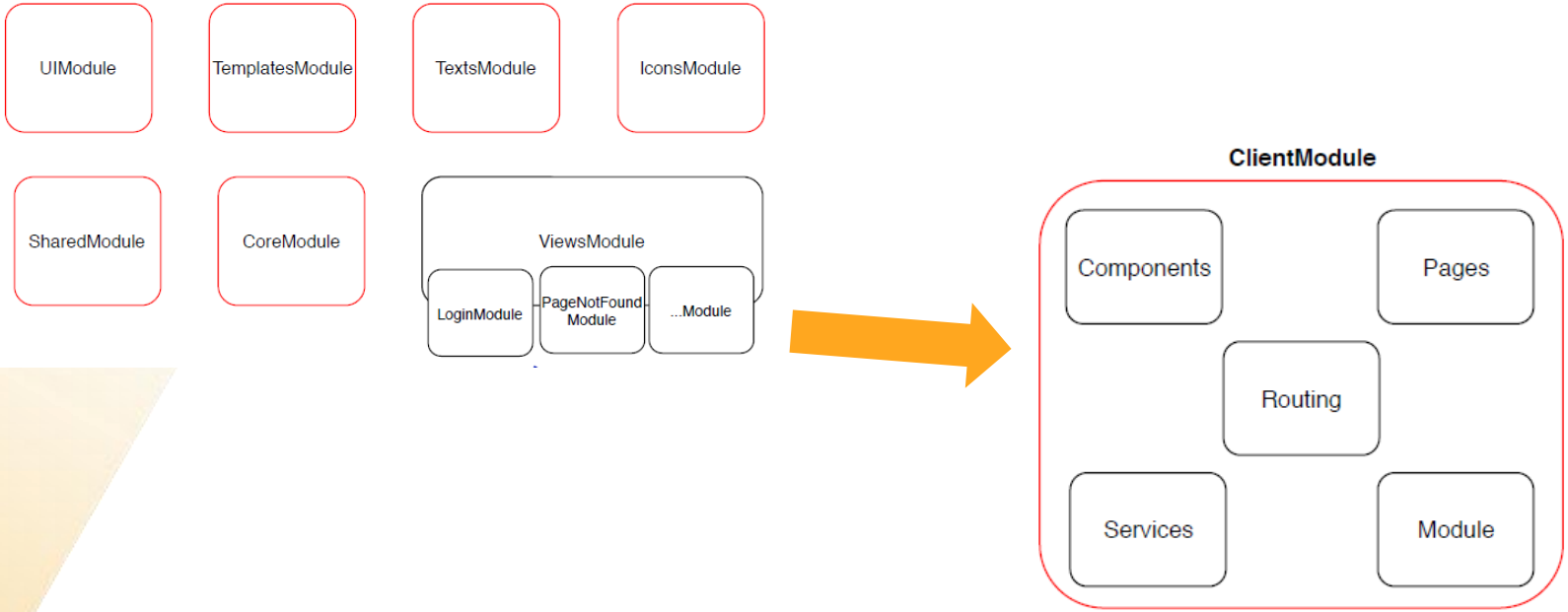

## Développement traditionnel :


## Développement parallèle :



# De l'agilité au logiciel (2/2)

- **Découpage technique :**



# 3

# UN PEU DE PRATIQUE

# Démarrage d'un projet

- **Les Schématics**
- **Angular CLI**
- **Premières commandes :**
  - ng new *<app_name>* : Génération d'un projet
  - ng serve : Serveur de developpement
  - ng build : Génération des sources
  - npm start : Script de démarrage
  - npm run *<script_name>* : Exécuter un script
  - ng test : Exécuter les tests

# Configuration du projet

- **Fichier de configuration / dépendances d'un projet Angular :**
  - angular.json
  - package.json

- **Ajout de librairie :**
  - npm install *<lib_name@lib_version>*
  - npm add *<lib_name@lib_version>*

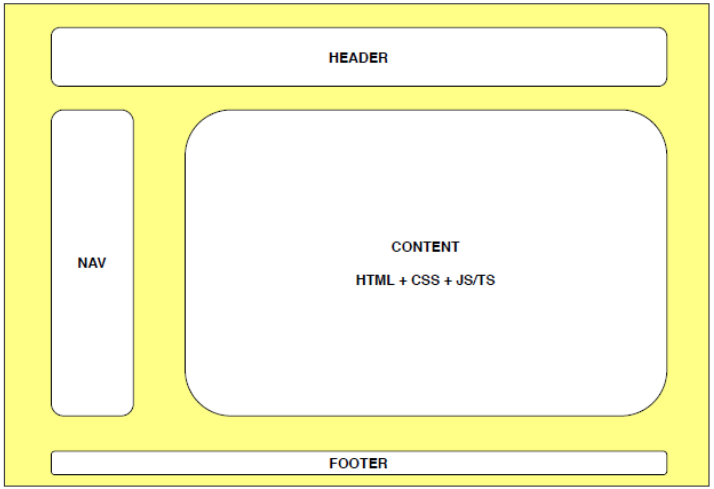
- **Génération avec les schematics :**
  - ng generate *<component_name> --<argument1> --<argumentN> …*

# Objectifs: Init projet

1. Intégrer les librairies necessaires au projet
2. Mettre en place l'architecture
3. Configurer les styles
4. Créer les premiers composants

# Objectifs: Modules Core et UI

1. Création des composants Header, Navbar et Footer
2. Projection de composant
3. Intégration du routage
4. Intégration du routage (Lazy loading)



# LE FRAMEWORK ANGULAR

Journée 2

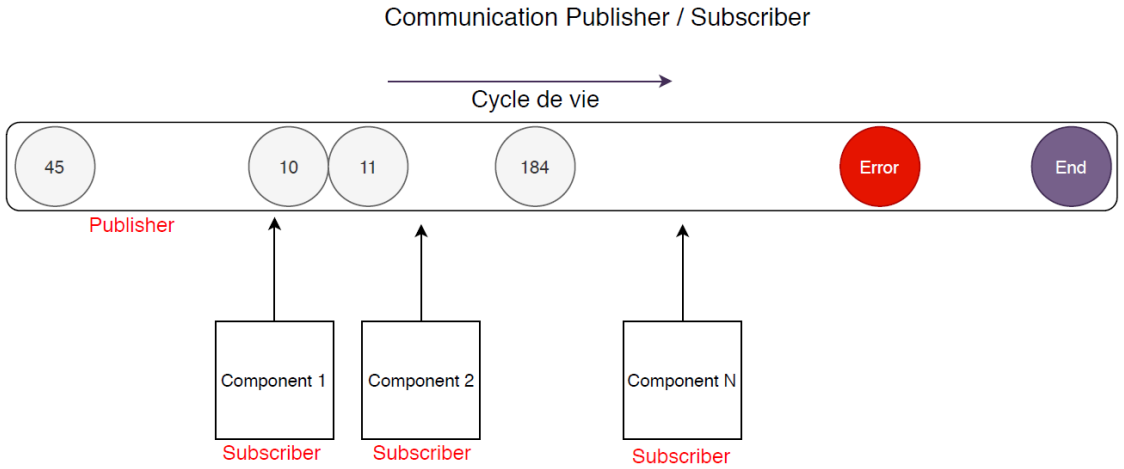
# 4

# LES OBSERVABLES

# Qu'est ce qu'un observable ?

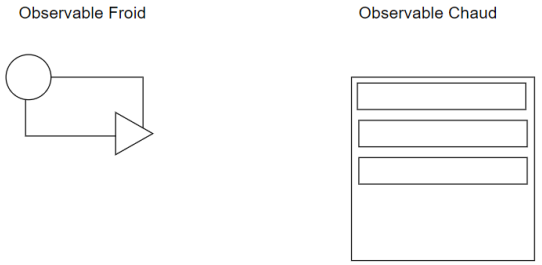
- **CallBack ? Promise ? Observable ?**

- **Exemple d'observable :**



# Observables et RxJS

- **Un observable peut être**
  - **Synchrone (*Chaud*)**
  - **Asynchrone (*Froid*)**

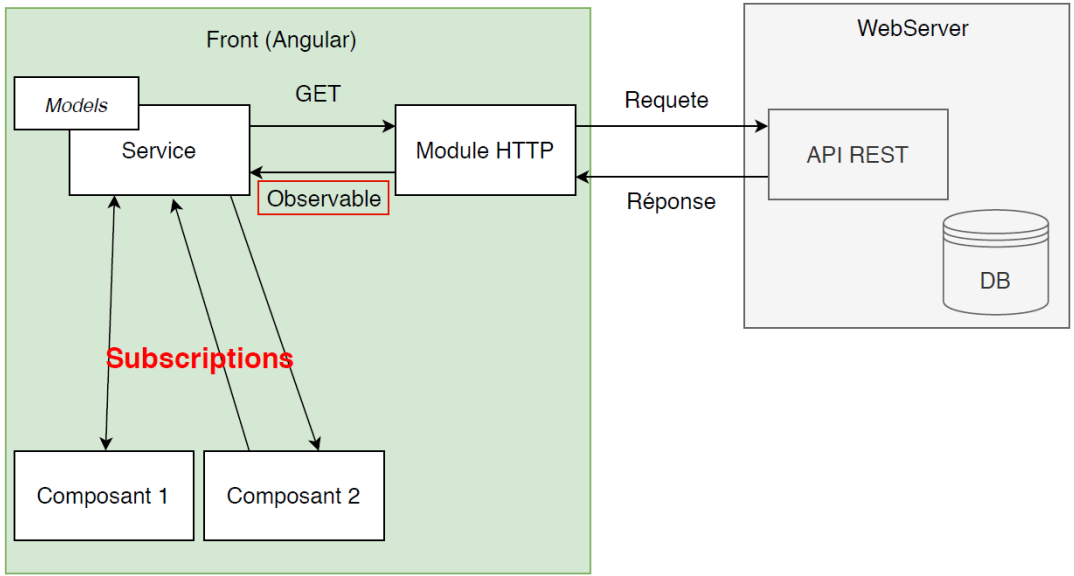



- **Librairie RxJS:**
  - **Traitement sur des flux de données asynchrones**
  - **Traitement sur des événements**
  - **Utilisation d'opérateurs sur les observables**

- **Et les Subjects ?**

# Objectifs: Architecture C/S

Archi C/S classique avec observables



# 5

# LES DECORATEURS

# Qu'est ce qu'un décorateur ?

- Annotation declarative (@)
- Implémenté par TypeScript
- Décorateurs générés avec les Schématics
- Exemples: @Component, @Injectable, @Pipe ...

# Objectifs: Utilisation des décorateurs

1. **Création d'un template de tableau**

2. **Modification d'affichage des données**

   - ✓ **Utilisation des décorateurs:**
     - ✓ **@Input()**
     - ✓ **@Pipe()**
     - ✓ **@Directive()**
     - ✓ **@Output()**

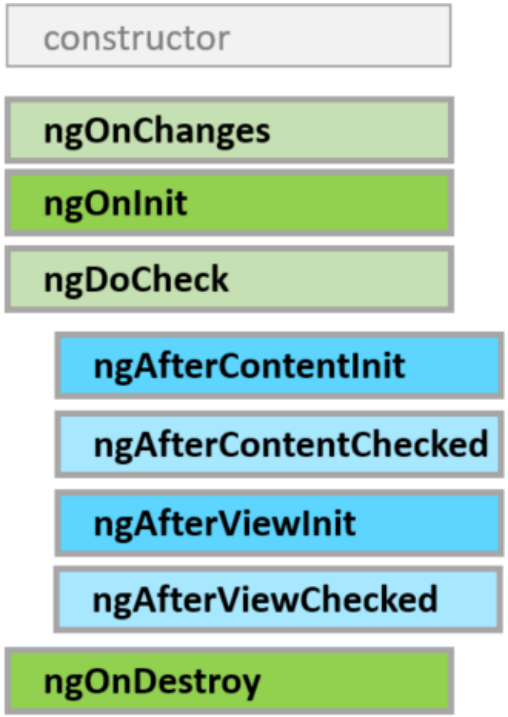
# LE FRAMEWORK ANGULAR

Journée 3

# 6

# QUELQUES CONCEPTS CLÉS

# Cycle de vie des composants



# Sélécteurs et Projection

- **Sélécteurs :**
  - **Identification unique d'un componsant (*selector*)**
  - **Permet d'identifier/récupérer un composant**

- **Projection :**
  - **Mécanisme permettant de projeter du contenu**
  - **Emeteur / Recepteur (contenu / contenant)**
  - **Projection d'un composant: *<balise_name></balise_name>***
  - **Projection de contenu non connu: *<ng-content></ng-content>***

# L'encapsulation

- Comprendre l'encapsulation :
  - Le DOM
  - Le ShadowDOM

- ViewEncapsulation propose par Angular :
  - Native
  - None
  - Emulated

# Manipulation des données

## Echange de données :

- **Interpolation**

  {{ varExample }}

- **Property Binging**

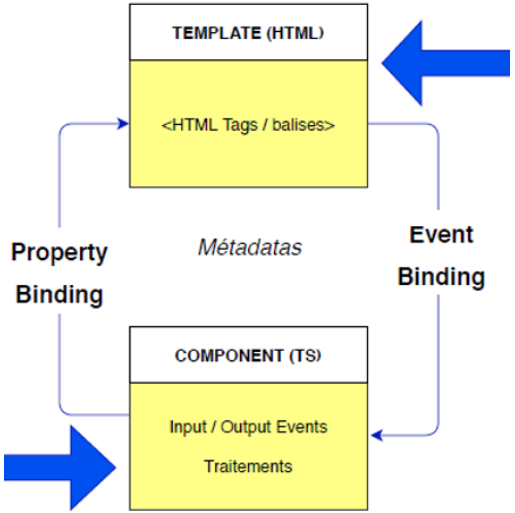
  <button type="bouton" [dsabled]="var">

- **Event Binding**

  <button type="bouton" (click)="test()">

- **“Two-way” Data Binding**

  <input [field]="varField" (change)="varField=$event">



# Objectifs:

1. Rendre le template de tableau générique (réutilisable)
2. Appliquer des modification de données sur le template
3. Transmettre des evenements du template au composant
4. Créer nos propres évènements (Customisation d'un bouton)

# LE FRAMEWORK ANGULAR

Journée 4

# 7

# LES FORMULAIRES

# Angular et Formulaires

- Utilité d'un formulaire ?

- Reactive Forms : Tooblox complète
  - Classe FormControl ➔ Controles
  - Classe FormGroup ➔ Regroupements de contrôles
  - Classe FormArray ➔ Création d'une liste de contrôles
  - Classe FormBuilder ➔ Constructeur de fomulaire

- Les Validators

# Objectifs:

1. Utiliser les fonctionnalités ReactivForms
2. Créer des templates de formulaires
3. Faire les enregistrements en DB
4. Ajouter des validators