



# Présentation



- Framework
  - Différence avec jquery, js etc
  - Contrainte
  - Basé sur le modèle MVC
  - Plus de développement local possible sans serveur
  - La logique est différente d'HTML qui héberge du JS
  - Développement via node
  - Angular == client exclusivement

# Présentation



- Framework
- 2 versions
  - Angular 1 (**angularjs** périmé mais à maintenir)
  - Angular 2 (nouvelle organisation)
    - Angular 3 (mort-né)
    - Angular 4 (très proche d'angular 2)
    - ....
  - Angular 9
  - Retrocompatibilité -> **Angular 4**
- Souvent associé à Bootstrap (+ jQuery)

# Présentation



- Logique particulière à base de composants
- Composant :
  - Associé à une balise (sélecteur CSS)
    - L'implantation HTML
  - Associé à du code HTML (avec patterns)
    - La vue
  - Associé à une feuille CSS
    - Paramétrage de la vue
- Module :
  - Ensemble de composants

# Présentation



- Projet Angular
- Hiérarchie de répertoires
  - Contient les descriptions de composants, modules
  - Les vues
  - Le code
- Le code est Javascript
- TypeScript

# typeScript



- Origine Microsoft
- Support Ecma6
- Javascript typé
  - Syntaxe proche de Scala
  - Attention, transpiler systématique
  - Pas encore une norme navigateurs
- Exemple de code



```
// Création d'une variable contenant une valeur booléenne.  
var maValeurBooleenne: boolean = false;  
  
// Création d'une variable contenant une chaîne de caractère.  
var maChaineDeCaractere: string = "Hello World";  
  
// Création d'une variable contenant un nombre.  
var monNombre: number = 1;  
  
// Création d'une fonction retournant une chaîne de caractère.  
function maFonction(): string {  
    return "Ma valeur de retour";  
}
```



```
function maFonction<T>(parametre: T) {  
    // Contenu de la fonction.  
}
```

```
class MaClasse<T> {  
    maVariable : T;  
    // Contenu de la classe.  
}
```

```
// Création d'une instance de la classe "MaClasse" en définissant un type.  
var monInstance = new MaClasse<string>();  
monInstance.maVariable = "Hello World";
```



```
interface MonInterface {  
    // Création d'une signature de variable.  
    maVariable: string;  
    // Création d'une signature de méthode.  
    maMethode(parametre: string): void;  
}  
  
class MaClasse implements MonInterface {  
    maVariable: string;  
    maMethode(parametre: string): void {  
        // Contenu de la méthode.  
    }  
}  
  
// Précision du type de la variable en utilisant l'interface.  
var instance: MonInterface = new MaClasse();
```



```
class MaClasseDeBase {
    private _firstname;
    private _lastname;

    public constructor(firstname: string, lastname: string) {
        this._firstname = firstname;
        this._lastname = lastname;
    }

    public direBonjour(): string {
        return "Bonjour " + this._firstname + ", " + this._lastname;
    }
}

// La classe hérite de "MaClasseDeBase".
class MaClasse extends MaClasseDeBase {
    public constructor(firstname: string, lastname: string) {
        // Accède au constructeur de "MaClasseDeBase".
        super(firstname, lastname);
    }
}

// Création d'une instance de "MaClasse" et
// appel de la méthode: "direBonjour" de la classe parente : "MaClasseDeBase".
var monInstance: MaClasse = new MaClasse("Jean", "Dupond");
monInstance.direBonjour();
```

```
module mon.espace.de.nom {  
    // Contenu du module: classe, fonction, etc.  
}
```



```
function maFonction(monParametre?: string) {  
    // On teste si le paramètre "monParametre" a une valeur.  
    if (monParametre) {  
        return monParametre;  
    } else {  
        // Dans le cas contraire, une valeur par défaut est retournée.  
        return "Hello World";  
    }  
}  
  
// La valeur retournée sera : "Hello World" sans avoir un message d'avertissement lors de  
// la compilation.  
var resultat: string = maFonction();  
// La valeur retournée sera : "Ma valeur".  
var resultat: string = maFonction("Ma valeur");
```

# typeScript



- Exemple de développement
- Classique + extension
- Environnement :
  - Node.js > 6.9 (en théorie mais attention à Angular)
  - 10.3 ou > 12
  - npm
  - Navigateur Web

# typeScript

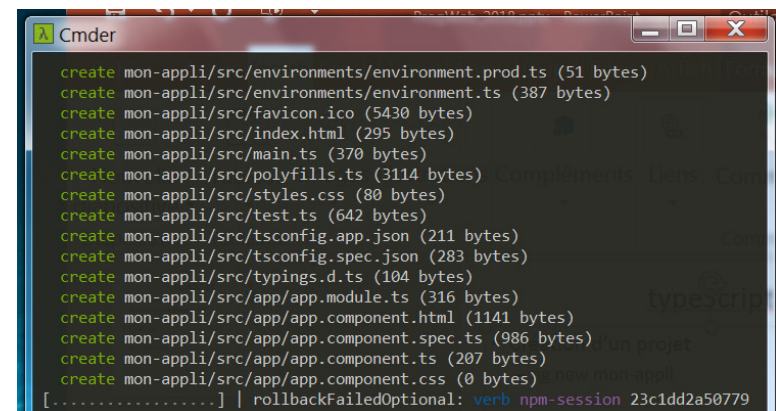


- Installation
  - `npm install -g @angular/cli`
- En global
  - Installation de scripts
  - Cf prefix etc..

# typeScript



- Création d'un projet
  - `ng new mon-appli`
  - Node.js : 10.13 ou 12.
  - Préférez les version LTS
- ng pour angular
  - Fabrication d'une hiérarchie
  - Composant de base
  - Prépare node\_modules

A screenshot of a Windows Command Prompt window titled 'Cmder'. It displays a series of 'create' commands for a project named 'mon-appli'. Each line shows a file path followed by its size in bytes. The files created include environment files, a favicon, index.html, main.ts, polyfills.ts, styles.css, test.ts, tsconfig files, typings.d.ts, app.module.ts, app.component.html, app.component.spec.ts, app.component.ts, and app.component.css. The last line shows a failed optional rollback command.

```
Cmder
create mon-appli/src/environments/environment.prod.ts (51 bytes)
create mon-appli/src/environments/environment.ts (387 bytes)
create mon-appli/src/favicon.ico (5430 bytes)
create mon-appli/src/index.html (295 bytes)
create mon-appli/src/main.ts (370 bytes)
create mon-appli/src/polyfills.ts (3114 bytes)
create mon-appli/src/styles.css (80 bytes)
create mon-appli/src/test.ts (642 bytes)
create mon-appli/src/tsconfig.app.json (211 bytes)
create mon-appli/src/tsconfig.spec.json (283 bytes)
create mon-appli/src/typings.d.ts (104 bytes)
create mon-appli/src/app/app.module.ts (316 bytes)
create mon-appli/src/app/app.component.html (1141 bytes)
create mon-appli/src/app/app.component.spec.ts (986 bytes)
create mon-appli/src/app/app.component.ts (207 bytes)
create mon-appli/src/app/app.component.css (0 bytes)
[.....] | rollbackFailedOptional: verb npm-session 23c1dd2a50779
```

```
C:\Users\menier\angular>ng new mon-appli
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE mon-appli/angular.json (3590 bytes)
CREATE mon-appli/package.json (1285 bytes)
CREATE mon-appli/README.md (1025 bytes)
CREATE mon-appli/tsconfig.json (489 bytes)
CREATE mon-appli/tslint.json (3125 bytes)
CREATE mon-appli/.editorconfig (274 bytes)
CREATE mon-appli/.gitignore (631 bytes)
CREATE mon-appli/browserslist (429 bytes)
CREATE mon-appli/karma.conf.js (1021 bytes)
CREATE mon-appli/tsconfig.app.json (210 bytes)
CREATE mon-appli/tsconfig.spec.json (270 bytes)
CREATE mon-appli/src/favicon.ico (948 bytes)
CREATE mon-appli/src/index.html (294 bytes)
CREATE mon-appli/src/main.ts (372 bytes)
CREATE mon-appli/src/polyfills.ts (2835 bytes)
CREATE mon-appli/src/styles.css (80 bytes)
CREATE mon-appli/src/test.ts (753 bytes)
CREATE mon-appli/src/assets/.gitkeep (0 bytes)
CREATE mon-appli/src/environments/environment.prod.ts (51 bytes)
CREATE mon-appli/src/environments/environment.ts (662 bytes)
CREATE mon-appli/src/app/app-routing.module.ts (246 bytes)
CREATE mon-appli/src/app/app.module.ts (393 bytes)
CREATE mon-appli/src/app/app.component.html (25757 bytes)
CREATE mon-appli/src/app/app.component.spec.ts (1068 bytes)
CREATE mon-appli/src/app/app.component.ts (213 bytes)
```

+ packages...

# typeScript



- Bien comprendre :
  - On ne fait pas ce qu'on veut
  - On est obligé de suivre les conventions d'angular
  - De patcher les fichiers d'angular
  - Donc 😞
    - De très bien comprendre comment angular fonctionne
    - Framework = moins de liberté
    - Bricolage impossible (!)

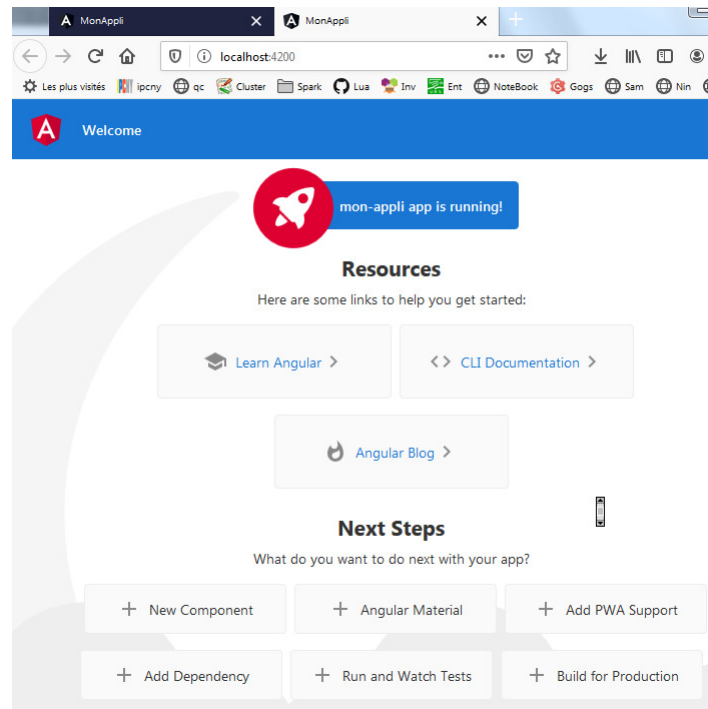


# typeScript



- Cycle de développement
  - Modifier les sources
  - Recompiler TS
  - Lancer un serveur
  - Utiliser la page
- Utilitaire intégré (genre de pm2 –watch + serveur)
- `cd mon-appli`
- `ng serve --open`

## Serveur local en 4200



Compilation des sources,  
Création de l'application Angular  
Lancement d'un serveur  
Accès à l'application Angular via le serveur

Ensuite, le serveur observe les mises  
à jour locales du code,  
recompile si besoin et donne accès au projet















**Attention** : impossible d'ouvrir un index.html  
dans un navigateur : communication angular  
via Ajax exclusivement (donc il faut un  
Serveur HTTP)

Relisez après **Attention**

**Encore une fois (!)**

# Angular



 .git	A votre avis 😊 ?
 e2e	Pour les tests du projet (librairie spéciale)
 node_modules	
 src	
 .angular-cli.json	Configuration ligne commande
 .editorconfig	Configurations IDE
 .gitignore	
 karma.conf.js	Tests unitaires Karma ng test
 package.json	A votre avis 😊 ?
 package-lock.json	
 protractor.conf.js	Tests e2e ng e2e
 README.md	
 tsconfig.json	Config pour le transpiler TypeScript
 tslint.json	

# Angular



DANS src/



app



assets Les images du projet etc..



environments Infos sur les envrt de déploiement



favicon.ico



index.html

Point d'entrée



main.ts

Code de lancement



polyfills.ts

Librairie multi navigateurs



styles.css

Déclarations globales CSS pour tout le projet



test.ts



tsconfig.app.json



tsconfig.spec.json

Configuration pour le transpiler, test etc..



typings.d.ts

# Angular



Dans APP/

Feuille de style pour le composant principal



app.component.css



app.component.html

La vue



app.component.spec.ts



app.component.ts

Le code / modèle + contrôleur



app.module.ts

Un module est une collection de composants

# Angular



- Cycle
  - Index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>MonAppli</title>
  <base href="/">

  <meta name="viewport" content='
  <link rel="icon" type="image/x-
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Allez voir dans

- app.component.css
- app.component.html
- app.component.spec.ts
- app.component.ts
- app.module.ts

# Angular

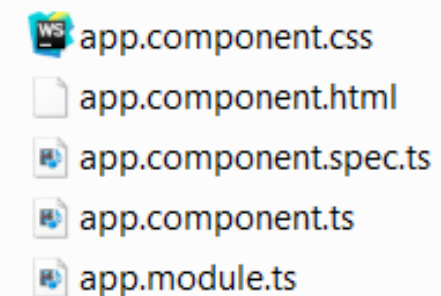


- Cycle
  - app.component.ts

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

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

```
export class AppComponent {  
  title = 'app';  
}
```



# Angular

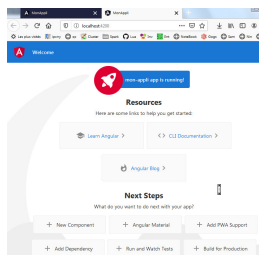


- Cycle
  - app.component.html

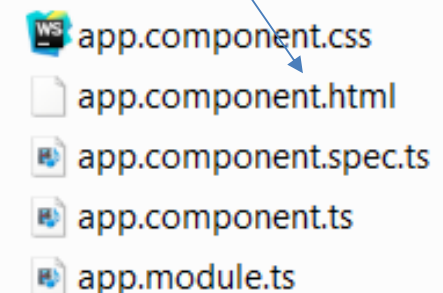
```
</g>  
</svg>
```

```
<span>{{ title }} app is running!</span>
```

```
<svg id="rocket-smoke" alt="Rocket Ship Smoke"  
  <path id="Path_40" data-name="Path 40" d="M  
</svg>
```



```
import { Component } from '@angular/core';  
  
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})  
export class AppComponent {  
  title = 'mon-appli';  
}
```





# Angular



- Application Angular :
  - Collection de composants
  - Imbrication
  - Communiquent ensemble
  - Mécanisme d'injection de code (fonctionnel)
- Bien comprendre Modèle MVC
- Découpage **OBLIGATOIRE**

# Angular



- Création d'une application Angular
  - Complète
  - Exemple classique
  - Étendue
  - 2 parties
    - Interface
    - Communication avec serveur / MongoDB
  - À faire pas à pas avant de se lancer



# Indestructible Team

## Liste des Heros

- Frozone (75 kg )
- GazerBeam (82 kg )
- Mirage (65 kg )
- Snug (14 kg )
- Elastigirl (63 kg )
- Doc Sunbright (56 kg )
- Super pédaleur (89 kg )
- Omnidroid RFC (72 kg )
- Bomb Voyage (89 kg )
- The Mole (69 kg )

Rajouter une nouvelle fiche ?



# Indestructible Team

## MIRAGE

Nom:

Poids:

Identité:

2. Le formulaire apparait

## Liste des Héros

1. On clique sur un des noms

- Frozone (75 kg )
- GazerBeam (82 kg )

### • **Mirage (65 kg )**

- Snug (14 kg )
- Elastigirl (63 kg )
- Doc Sunbright (56 kg )
- Super pédaleur (89 kg )
- Omnidroid RFC (72 kg )
- Bomb Voyage (89 kg )
- The Mole (69 kg )



# Indestructible Team

## MIRAGE GIRL

Nom:

Poids:

Identité:

On peut changer dans le formulaire  
Et l'écran se met à jour automatiquement

## Liste des Heros

- Frozone (75 kg )
- GazerBeam (82 kg )

### ● Mirage Girl (65 kg)

- Snug (14 kg )
- Elastigirl (63 kg )
- Doc Sunbright (56 kg )
- Super pédaleur (89 kg )

On modifie les fichiers dans src / app

Le serveur lancé recompile automatiquement

La page est mise à jour dans le navigateur

**NE PAS STOPPER LE SERVEUR**

Relisez la phrase précédente ;-)

Le serveur joue le rôle de make + de transpilateur

# Angular



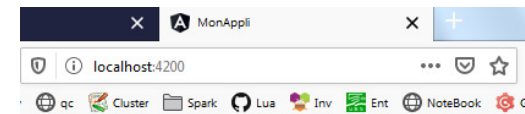
- Nettoyage de l'application principale

- app.component.html

- <h1> <img src='assets/images/i.jpg'/> {{ title }} </h1>

- Placer image dans assets/images/

- Modifier le modèle :



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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'incredible team';
}
```



```

    }

    svg#rocket-smoke {
      right: 120px;
      transform: rotate(-5deg);
    }
  }

  @media screen and (max-width: 575px) {
    svg#rocket-smoke {
      display: none;
      visibility: hidden;
    }
  }
</style>

<div class="content" role="main">

  <h1>    <img src='assets/images/i.jpg' /> {{ title }}  </h1> >

</div>

<router-outlet></router-outlet>

```



# Angular



- Rajouter un composant pour l'application
  - Générer le code d'un nouveau composant
  - **ng generate component heroes**
  - Création d'un nouveau répertoire

```
C:\Users\menier\angular\mon-appli>ng generate component heroes
CREATE src/app/heroes/heroes.component.html (21 bytes)
CREATE src/app/heroes/heroes.component.spec.ts (628 bytes)
CREATE src/app/heroes/heroes.component.ts (275 bytes)
CREATE src/app/heroes/heroes.component.css (0 bytes)
UPDATE src/app/app.module.ts (475 bytes)
```

- Le composant fonctionne de la même manière

# Angular



- Insérer le nouveau composant dans l'application

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

@Component({
  selector: 'app-heroes',
  templateUrl: './heroes.component.html',
  styleUrls: ['./heroes.component.css']
})
export class HeroesComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

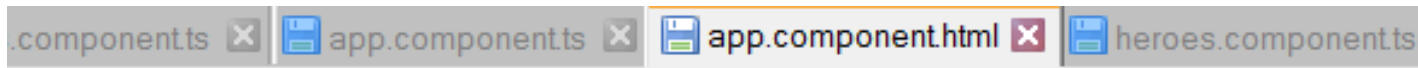
}
```

Le composant est représenté par `<app-heroes></app-heroes>`  
(le selector)

# Angular



- Insérer le nouveau composant dans l'application



```
<h1>
  <img src='assets/images/i.jpg' /> {{ title }}
</h1>
<app-heroes></app-heroes>
```

- On veut que :
- Le composant affiche Hello batman
- Batman est une variable modèle
- Comment : ...

# Angular



- Modifier le **modèle** du composant Heroes

```
etails.componentts x app.componentts x app.component.html x heroes.componentts x
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-heroes',
  templateUrl: './heroes.component.html',
  styleUrls: ['./heroes.component.css']
})
export class HeroesComponent implements OnInit {
  hero = "batman"

  constructor() { }

  ngOnInit() {
  }
}
```

# Angular



- Modifier la **vue** du composant Heroes

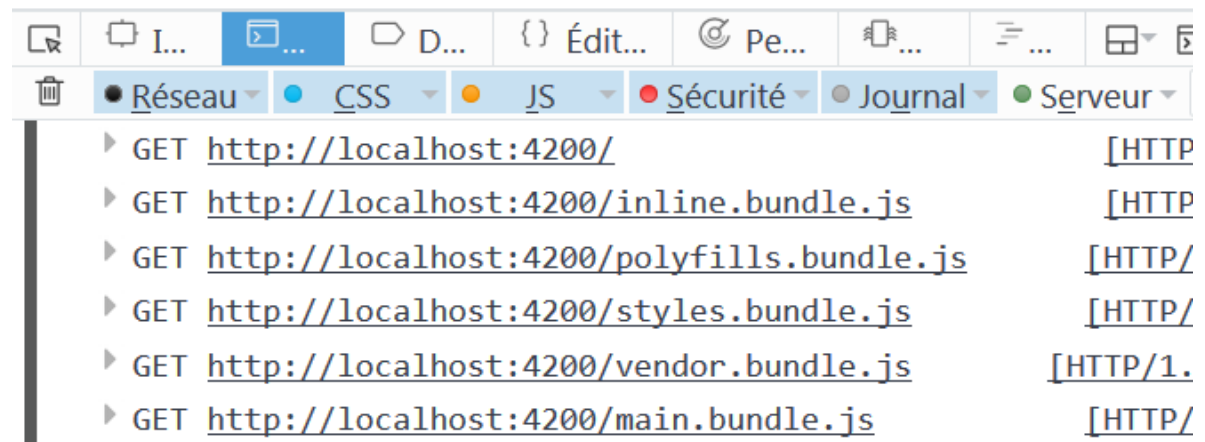
```
component.ts | app.component.html | heroes.component.ts | heroes.component.html
<p>
  Bonjour {{hero}}
</p>
```

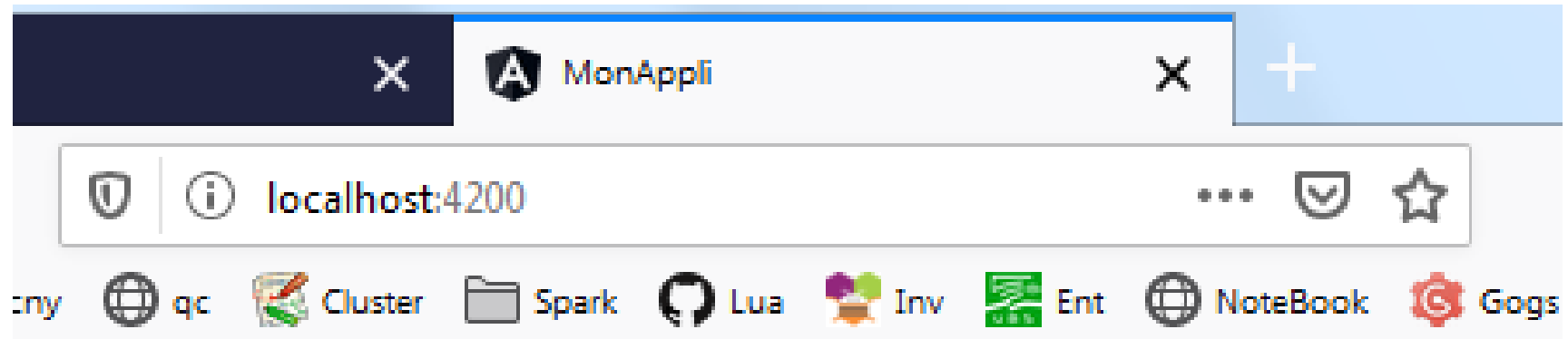


**incredible team**

Bonjour batman

Observez les Appels Ajax





**incredible team**

Bonjour batman

```

<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>MonAppli</title>
  <base href="/">

  <meta name="viewport" c
  <link rel="icon" type="
</head>
<body>
  <app-root></app-root>
</body>
</html>

```

Index.html

```

import { Component } from '@angular/core'

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'incredible team';
}

```

app.component.ts

```

<h1>
  <img src='assets/images/i.jpg' /> {{ title }}
</h1>
<app-heroes></app-heroes>

```

app.component.html

```

import { Component, OnInit }

@Component({
  selector: 'app-heroes',
  templateUrl: './heroes.c
  styleUrls: ['./heroes.cc
})
export class HeroesCompone

  hero = "batman"

  constructor() { }

  ngOnInit() {
  }
}

```

heroes.component.ts

```

<p>
  Bonjour {{hero}}
</p>

```

heroes.component.html

# Angular



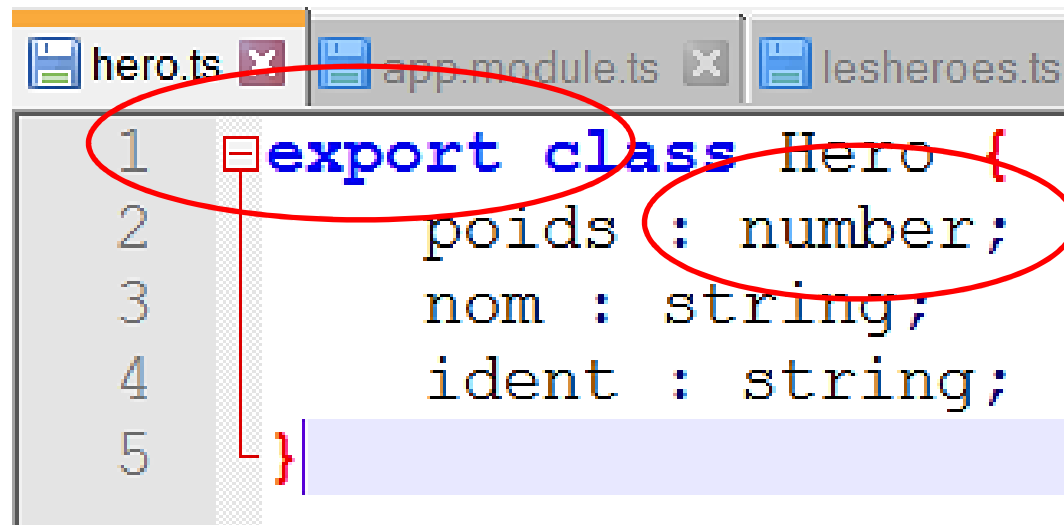
- Structure de données
  - Le nom du héro n'est pas suffisant
  - On rajoute le poids
  - On rajoute la vraie identité
  - Structure de données partagée par tous les composants
  - Définition d'une classe Hero
  - src/app : Hero.ts
  - Attention : **TypeScript**



# Angular



- src/app/Hero.ts

A screenshot of a code editor window with three tabs: 'hero.ts', 'app.module.ts', and 'lesheroes.ts'. The 'hero.ts' tab is active and shows a TypeScript class definition. The code is as follows:

```
1 export class Hero {  
2     poids : number;  
3     nom : string;  
4     ident : string;  
5 }
```

Red circles highlight the 'export class' keyword and the 'poids : number;' property. A red line points from the 'export' keyword to the 'poids' property.

- Accès à la classe dans le composant Heroes
- Attention, c'est forcément du TypeScript

# Angular



- heroes.component.ts

– Rajouter `import { Hero } from '../hero';`

```
import { Component, OnInit } from '@angular/core';
import { Hero } from '../hero';
```

```
@Component({
  selector: 'app-heroes',
  templateUrl: './heroes.component.html',
  styleUrls: ['./heroes.component.css']
})
```

```
export class HeroesComponent implements OnInit {
```

```
  hero: Hero = {
    poids : 85,
    nom : "batman",
    ident : "B Wayne"
  }
```

```
  constructor() { }
```

```
  ngOnInit() {
  }
```

```
}
```



**in**

Bonjour [object Object]

# Angular



- heroes.component.html

```
heroes.component.ts x  heroes.component.html x
<p>
  Bonjour {{hero.nom}} <br/>
  vous pesez {{hero.poids}} <br/>
  vous etes {{hero.ident}}
</p>
```



**incredible team**

Bonjour batman  
vous pesez 85  
vous etes B Wayne

# Angular



- Pipe Angular

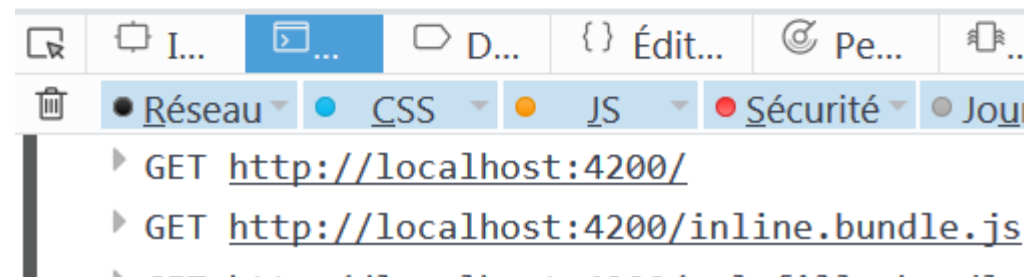
```
s.component.ts x heroes.component.html x  
<p>  
  Bonjour {{hero.nom | uppercase}} <br/>  
  vous pesez {{hero.poids}} <br/>  
  vous etes {{hero.ident}}  
</p>
```



**incredible team**

Un pipe | permet de modifier  
la valeur passée à la vue  
(en cascade)

Bonjour BATMAN  
vous pesez 85  
vous etes B Wayne




# Angular



- Modèle -> vue et **vue -> modèle ?**
- **On souhaite pouvoir éditer le nom du hero**

```
<p>
  Bonjour {{ hero.nom | uppercase }} <br/>
  vous pesez {{ hero.poids }} <br/>
  vous etes {{ hero.ident }} <br/>
</p>
<div>
  <label>nom :
  | <input [(ngModel)] = "hero.nom" >
  </label>
</div>
```

angular **model**

- ▶ GET <http://localhost:4200/polyfills.bundle.js> [HTTP/1.1 200 OK]
- ▶ GET <http://localhost:4200/styles.bundle.js> [HTTP/1.1 200 OK]
- ▶ GET <http://localhost:4200/vendor.bundle.js> [HTTP/1.1 200 OK]
- ▶ GET <http://localhost:4200/main.bundle.js> [HTTP/1.1 200 OK]
- ✖ ▶ Error: Template parse errors: Can't bind to 'ngModel' since it isn't a known property of 'input'. (" <div> <label>name: <input [ERROR ->][[(ngModel)]]="hero.nom"> </label> </div>"): ng:///AppModule/HeroesComponent.html@7:13
- ▶ GET  <http://localhost:4200/sockjs-node/info> [HTTP/1.1 200 OK]
- ▶ GET <http://localhost:4200/sockjs-...> [HTTP/1.1 101 Switching Protocols]

ngModel se trouve dans le module FormsModule d'angular  
Il faut le rajouter au projet + en donner l'accès

- Rajouter le module au niveau de l'application
- **src./app/app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HeroesComponent } from './heroes/heroes.component';

@NgModule({
  declarations: [
    AppComponent,
    HeroesComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    AppRoutingModule
  ],
  providers: []
})
```



## incredible team

Bonjour BATMANNE  
vous pesez 85  
vous etes B Wayne

name:

Mise à jour PENDANT la frappe  
la vue (input) modifie le modèle  
la nouvelle valeur du modèle est injectée  
automatiquement dans la vue

Ce n'est pas votre code qui met à jour : angular a créé un lien et la mise à jour  
Se fait toute seule (cf excel)



# Angular



- Rappel du projet application
  - On veut afficher une liste éditable



**Indestructible Team**

1. Collection d'objets
2. Création d'une liste HTML

## Liste des Heros

- Frozone (75 kg )
- GazerBeam (82 kg )
- Mirage (65 kg )
- Snug (14 kg )
- Elastigirl (63 kg )
- Doc Sunbright (56 kg )
- Super pédaleur (89 kg )
- Omnidroid RFC (72 kg )
- Bomb Voyage (89 kg )
- The Mole (69 kg )

Rajouter une nouvelle fiche ?

# Angular



- Création d'une collection
  - Liste bidon (reçue d'un serveur)
  - Création d'une classe conteneur : /src/app/lesheroes.ts

```
import { Hero } from './hero';  
  
export const HEROES: Hero[] = [  
  { poids: 75, nom: 'Frozone' , ident: 'Lucius Best' },  
  { poids: 82, nom: 'GazerBeam' , ident: 'J Paladino' },  
  { poids: 65, nom: 'Mirage' , ident: 'Linda Johns' },  
  { poids: 14, nom: 'Snug' , ident: '?' },  
  { poids: 63, nom: 'Elastigirl' , ident: 'Helen Parr' },  
  { poids: 56, nom: 'Doc Sunbright', ident: 'Tom Bullit' },  
  { poids: 89, nom: 'Super pédaleur', ident: 'Luc Courtrai' },  
  { poids: 72, nom: 'Omnidroid RFC', ident: 'Fred Guidec' },  
  { poids: 89, nom: 'Bomb Voyage' , ident: 'Bill Murray' },  
  { poids: 69, nom: 'The Mole' , ident: 'Jean-marc Poussin' }  
];
```

# Angular



```
heroes.component.ts | lesheroes.ts | app.component.html | heroes.component.html
1  import { Component, OnInit } from '@angular/core';
2  import { Hero } from '../hero'
3  import { HEROES } from '../lesheroes';
4
5  @Component({
6    selector: 'app-heroes',
7    templateUrl: './heroes.component.html',
8    styleUrls: ['./heroes.component.css']
9  })
10 export class HeroesComponent implements OnInit {
11
12    heroes: Hero[]
13
14    constructor() { }
15
16    ngOnInit() {
17        this.heroes = HEROES
18    }
19
20 }
```

# Angular



```
es.component.ts x | lesheroes.ts x | app.component.html x | heroes.component.html x
<ul class="heroes">
  <li *ngFor="let hero of heroes">
    {{hero.nom}}
  </li>
</ul>
```

Attention : `*ngFor` est important  
Instruction dans la chaîne (pas du HTML)

# Angular



**incredible team**

- Frozone
- GazerBeam
- Mirage
- Snug
- ElastiGirl
- Doc Sunbright
- SuperPédaleur
- Omnidroid RFC
- Bomb Voyage
- The Mole

# Angular



- On souhaite rajouter une interaction
  - Gestion des évènements angular
  - `<li *ngFor="let hero of heroes" (click)="onSelection(hero)">`
  - Pour chaque ligne, rajoute un evt click
  - On veut afficher le poids du héros sélectionné
  - On modifie la vue et le code ts

# Angular



```
heroes.component.ts | lesheroes.ts | app.component.html | heroes.component.html
1  import { Component, OnInit } from '@angular/core';
2  import { Hero } from '../hero'
3  import { HEROES } from '../lesheroes';
4
5  @Component({
6    selector: 'app-heroes',
7    templateUrl: './heroes.component.html',
8    styleUrls: ['./heroes.component.css']
9  })
10 export class HeroesComponent implements OnInit {
11
12     heroes:Hero[]
13     heroSelection:Hero
14
15     constructor() { }
16
17     ngOnInit() {
18         this.heroes = HEROES
19     }
20
21     onSelection(h: Hero) {
22         this.heroSelection = h
23     }
24
25 }
```

# Angular



```
.components.ts x | lesheroes.ts x | app.component.html x | heroes.component.html x
<ul class="heroes">
  <li *ngFor="let hero of heroes" (click)="onSelection(hero)">
    {{hero.nom}}
  </li>
</ul>
son poids est {{heroSelection.poids}}
```



# Angular



**incr**

MAIS :

- Frozone
- GazerBeam
- Mirage
- Snug
- Elastigirl
- Doc Sunbright
- Super pédaleur
- Omnidroid RFC
- Bomb Voyage
- The Mole

1. erreur quand pas de sélection :

```
x ▶ ERROR TypeError: _co.heroSelection is undefined
Trace de la pile :
View_HeroesComponent_0/<@ng:///AppModule
/HeroesComponent.ngfactory.js:33:13
debugUpdateRenderer@webpack-internal:///./node_modules
/@angular/core/esm5/core.js:14909:12
```

2. Quelle ligne est sélectionnée ?

son poids est 82

# Angular



Conditionnelle angular pour la génération de code :

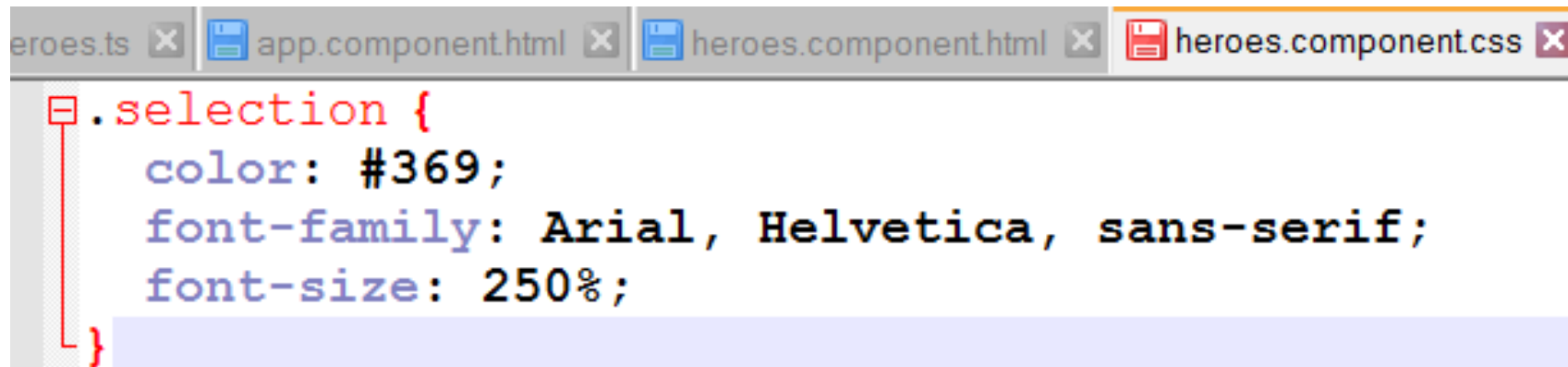
```
s.component.ts x | lesheroes.ts x | app.component.html x | heroes.component.html x
<ul class="heroes">
  <li *ngFor="let hero of heroes" (click)="onSelection(hero)">
    {{hero.nom}}
  </li>
</ul>
<div *ngIf="heroSelection">
  son poids est {{heroSelection.poids}}
</div>
```

\*ngIf seulement si heroSelection est défini

# Angular



Idée pour marquer la ligne sélectionnée : changer la classe CSS  
Modification de la feuille CSS du composant

A screenshot of a code editor window. The top bar shows four tabs: 'heroes.ts', 'app.component.html', 'heroes.component.html', and 'heroes.component.css'. The 'heroes.component.css' tab is active. The code in the editor is:

```
.selection {  
  color: #369;  
  font-family: Arial, Helvetica, sans-serif;  
  font-size: 250%;  
}
```

The code is color-coded: the class name is red, the opening curly brace is red, the property names are blue, the values are black, and the closing curly brace is red. A light blue horizontal bar is visible at the bottom of the code block.

Ensuite, il faut modifier la vue pour que la classe change quand l'objet est Sélectionné :

# Angular



```
s.component.ts x | lesheroes.ts x | app.component.html x | heroes.component.html x | heroes.
<ul class="heroes">
  <li *ngFor="let hero of heroes"
      [class.selection] = "hero == heroSelection"
      (click)="onSelection(hero)"
  >
    {{hero.nom}}
  </li>
</ul>
<div *ngIf="heroSelection">
  son poids est {{heroSelection.poids}}
</div>
```

Rajoute ou enlève une classe CSS en fonction de la condition

# Angular



**incredible team**

- Frozone
- **GazerBeam**
  - Mirage
  - Snug
  - Elastigirl
  - Doc Sunbright
  - Super pédaleur
  - Omnidroid RFC
  - Bomb Voyage
  - The Mole

son poids est 82

# Angular



## Projet d'application



## Indestructible Team

### MIRAGE

Nom:

Poids:

Identité:

### Liste des Heros

- Frozone (75 kg )
- GazerBeam (82 kg )
- **Mirage (65 kg )**
- Snug (14 kg )
- Elastigirl (63 kg )
- Doc Sunbright (56 kg )
- Super pédaleur (89 kg )
- Omnidroid RFC (72 kg )
- Bomb Voyage (89 kg )
- The Mole (69 kg )



# Indestructible Team

Composant hero-details

## MIRAGE

Nom:

Poids:

Identité:

Composant heroes

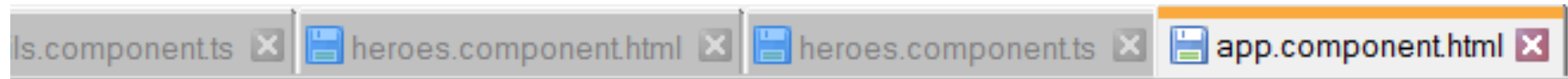
## Liste des Heros

- Frozone (75 kg )
- GazerBeam (82 kg )
- **Mirage (65 kg )**
- Snug (14 kg )
- Elastigirl (63 kg )
- Doc Sunbright (56 kg )
- Super pédaleur (89 kg )
- Omnidroid RFC (72 kg )
- Bomb Voyage (89 kg )
- The Mole (69 kg )

# Angular



app.component.html



```
<h1>  
  <img src='assets/images/i.jpg' /> {{ title }}  
</h1>
```

```
<app-heroes> </app-heroes>
```

---




# Angular



## heroes.component.html : découpage en 2 div

```
<div style="width: 70%; float:right">
  <h2>Liste des Heros</h2>
  <ul >
    <li *ngFor="let hero of heroes | orderBy:['nom']"
      (click)="onSelect(hero)"
      [class.selection]="hero === courant">
      <span>{{hero.nom}} ({{hero.poids}} kg )</span>
      <span *ngIf="courant">
        <button *ngIf="courant === hero"
          (click)="onEfface(hero)">
            effacer {{hero.nom}} ?
          </button>
        </span>
      </li>
    </ul>
    <div *ngIf="heroes.length < 15">
      <button (click)="onRajoute()">Rajouter une nouvelle fiche ?</button>
    </div>
  </div>
```



The screenshot shows the application's UI. On the left, the 'Indestructible Team' logo is displayed. Below it, the 'MIRAGE' hero is selected, showing its details: Nom: Mirage, Poids: 65, Identité: Linda Johns, and a button 'effacer Mirage ?'. On the right, a list of heroes is shown, with 'Mirage (65 kg)' highlighted. The list includes: Frozone (75 kg), GazerBeam (82 kg), Mirage (65 kg), Snug (14 kg), Elastigirl (63 kg), Doc Sunbright (66 kg), Super pédaleur (89 kg), Omnidroid RFC (72 kg), Bomb Voyage (89 kg), and The Mole (69 kg). A button 'Rajouter une nouvelle fiche ?' is at the bottom of the list. A red box highlights the list of heroes and the 'Rajouter une nouvelle fiche ?' button.

# Angular



```
<div style="width: 30%; float:left">
  <app-hero-details [monhero]="courant"></app-hero-details>
  <div *ngIf="courant">
    <button (click)="onEfface(courant)">
      effacer {{courant.nom}} ?
    </button>
  </div>
</div>
```



Indestructible Team

**MIRAGE**

Nom: Mirage

Poids: 65

Identité: Linda Johns

effacer Mirage ?

Liste des Heros

- Frozone (75 kg )
- GazerBeam (82 kg )
- **Mirage (65 kg )**
- Snug (14 kg )
- Elastigirl (63 kg )
- Doc Sunbright (66 kg )
- Super pédaleur (89 kg )
- Omnidroid RFC (72 kg )
- Bomb Voyage (89 kg )
- The Mole (69 kg )

effacer Mirage ?

Rajouter une nouvelle fiche ?

Le composant **app-hero-details** a besoin de connaître le hero courant (**courant**) pour pouvoir le modifier  
Dans le composant **app-hero-details**, il y a un modèle **monhero** : la ligne **[monhero] = "courant"** permet de lier les deux modèles

# Angular



```
1
2 <div *ngIf="monhero">
3   <H2>
4     {{monhero.nom | uppercase}}
5   </H2>
6   <div>
7     <label>Nom:
8       <input [(ngModel)]="monhero.nom" >
9     </label>
10  </div>
11  <div>
12    <label>Poids:
13      <input [(ngModel)]="monhero.poids" >
14    </label>
15  </div>
16  <div>
17    <label>Identité:
18      <input [(ngModel)]="monhero.ident" >
19    </label>
20  </div>
21 </div>
22
```



Indestructible Team

## MIRAGE

Nom: Mirage  
Poids: 65  
Identité: Linda Johns

[effacer Mirage ?](#)

## Liste des Heros

- Frozone (75 kg )
- GazerBeam (82 kg )

• **Mirage (65 kg )**

[effacer Mirage ?](#)

- Snug (14 kg )
- Elastigirl (63 kg )
- Doc Sunbright (66 kg )
- Super pédaleur (89 kg )
- Omnidroid RFC (72 kg )
- Bomb Voyage (89 kg )
- The Mole (69 kg )

[Ajouter une nouvelle fiche ?](#)

# Angular



```
details.component.html x hero-details.component.ts x heroes.component.html x heroes.component.ts x
import { Component, OnInit, Input } from '@angular/core';
import { Hero } from '../hero';

@Component({
  selector: 'app-hero-details',
  templateUrl: './hero-details.component.html',
  styleUrls: ['./hero-details.component.css']
})

export class HeroDetailsComponent implements OnInit {

  @Input() monhero: Hero;

  constructor() { }

  ngOnInit() {
  }
}
```



Indestructible Team

## MIRAGE

Nom: Mirage  
Poids: 65  
Identité: Linda Johns

[effacer Mirage ?](#)

## Liste des Heros

- Frozone (75 kg)
- GazerBeam (82 kg)
- Snug (14 kg)
- Elastigirl (63 kg)
- Doc Sunbright (56 kg)
- Super pédaleur (89 kg)
- Omnidroid RFC (72 kg)
- Bomb Voyage (89 kg)
- The Mole (69 kg)

[Rajouter une nouvelle fiche ?](#)

• **Mirage (65 kg)**

[effacer Mirage ?](#)

# Angular



```
details.component.html x hero-details.component.ts x heroes.component.html x heroes.component.ts x
import { Component, OnInit } from '@angular/core';
import { Hero } from '../hero'
import { HEROES } from '../lesheroes'

@Component({
  selector: 'app-heroes',
  templateUrl: './heroes.component.html',
  styleUrls: ['./heroes.component.css']
})

export class HeroesComponent implements OnInit {
  heroes: Hero[]
  courant : Hero

  constructor() {
    this.heroes = HEROES
  }

  ngOnInit() {
  }

  onSelect(hero: Hero) {
    this.courant = hero
  }

  onEfface(qui: Hero) {
    var a = this.heroes.filter( e => {return (e.nom != qui.nom)} )
    this.heroes = a
    this.courant = undefined
  }

  onRajoute() {
    let nouv : Hero = {poids:50, nom:"petit nouveau", ident:"?"}
    this.heroes.push(nouv)
    this.courant = nouv
  }
}
```



Indestructible Team

## MIRAGE

Nom:   
Poids:   
Identité:

## Liste des Heros

- Frozone (75 kg )
- GazerBeam (82 kg )

### • Mirage (65 kg )

- Snug (14 kg )
- Elastigirl (63 kg )
- Doc Sunbright (66 kg )
- Super pédaleur (89 kg )
- Omnidroid RFC (72 kg )
- Bomb Voyage (89 kg )
- The Mole (69 kg )

# Angular



Indestructible Team

<b>MIRAGE</b>	<b>Liste des Heros</b>
Nom: Mirage	<ul style="list-style-type: none"><li>Frozone (75 kg)</li><li>GazerBeam (82 kg)</li><li><b>Mirage (65 kg)</b></li><li>Snug (14 kg)</li><li>Elastigirl (63 kg)</li><li>Doc Sunbright (66 kg)</li><li>Super pôleleur (89 kg)</li><li>Omnirope RHC (72 kg)</li><li>Bomb Voyage (89 kg)</li><li>The Mole (69 kg)</li></ul>
Poids: 65	
Identité: Linda Johns	
<a href="#">officer Mirage ?</a>	<a href="#">officer Mirage ?</a>
<a href="#">Rapporter une nouvelle fiche ?</a>	

```
details.component.html x hero-details.component.ts x heroes.component.html x heroes.component.ts x
import { Component, OnInit } from '@angular/core';
import { Hero } from '../hero'
import { HEROES } from '../lesheroes'

@Component({
  selector: 'app-heroes',
  templateUrl: './heroes.component.html',
  styleUrls: ['./heroes.component.css']
})

export class HeroesComponent implements OnInit {
  heroes: Hero[]
  courant : Hero

  constructor() {
    this.heroes = HEROES
  }
}
```

# Angular



Indestructible Team

**MIRAGE**  
Nom:   
Poids:   
Identité:

**Liste des Heros**

- Frozone (75 kg)
- GazerBeam (82 kg)
- **Mirage (65 kg)**
- Snugg (14 kg)
- Elastigirl (63 kg)
- Doc Sunbright (66 kg)
- Super pèdaleur (89 kg)
- Omnidroid RFX (72 kg)
- Bomb Voyage (89 kg)
- The Mole (69 kg)

```
onSelect(hero: Hero) {  
    this.courant = hero  
}  
  
onEfface(qui: Hero) {  
    var a = this.heroes.filter( e => {return (e.nom != qui.nom)} )  
    this.heroes = a  
    this.courant = undefined  
}  
  
onRajoute() {  
    let nouv : Hero = {poids:50, nom:"petit nouveau", ident:"?"}  
    this.heroes.push(nouv)  
    this.courant = nouv  
}  
}
```

# Angular



Remarque : le tri de la liste se fait en utilisant un pipe *orderBy*

*La liste est triée même pendant l'édition des noms*

```
<h2>Liste des Heros</h2>
<ul >
  <li *ngFor="let hero of heroes | orderBy:['nom']"
      (click)="onSelect(hero) "
      [class.selection]="hero === courant">
```

Angular ne possède pas de pipe *orderBy*

*Il est possible de programmer soi même des pipes, ou bien de récupérer des librairies via npm :*

*Par exemple :*        npm install angular-pipes --save



# Angular : déploiement



L'application angular ne peut fonctionner que sur un serveur  
PAS dans un navigateur local

*ng build*

Copier le contenu du répertoire *build/* sur le serveur

**Attention** si ce n'est pas à la racine, indiquer le chemin à la compilation

*ng build --base-href=/mon/application/*

**Voir les outils d'optimisation de code et de visualisation de paquets**

# Angular : SPA



## Single Page Application : SPA

Jusqu'à présent :

*serveur*      *Client web*

*chemins* -> *pages html*

*chemins* -> *applications angular*

*une application par chemin*

*communication / http / cookies etc*

*plein de petites pages / applications angular*

*beaucoup de communications serveur*

*une seule app angular*

-> *le chemin est capturé en local*  
*l'application génère la bonne page*  
*moins de communication serveur*  
*une grosse application angular*  
*programmation simplifiée*

# Angular : routage



## Single Page Application : SPA

Application = fichier js (main.bundle.js)

Création des pages à la volée

Système de navigation

HTML : `<a href= > </a>`

Requête HTTP : communication avec le serveur

Angular : 2 mécanismes

a. Insertion de # dans les chemins <http://host.com/#/mapage>

b. Manipulation de l'historique

Javascript : push / etc...

PAS DE COMMUNICATION SERVEUR

# Angular : routage



## lien attribut / chemin

```
<a [routerLink] = ['/'] > home page </a>  
<a [routerLink] = ['/editeur']> Editeur </a>  
...  
<router-outlet></router-outlet>
```

## lien chemin / composant

```
const routes : Routes = [  
  { path : "", component : HomeComponent },  
  { path : 'editeur', component: EditeurComponent }  
]
```

Vue de HomeComponent  
Vue de EditeurComponent

# Angular : routage

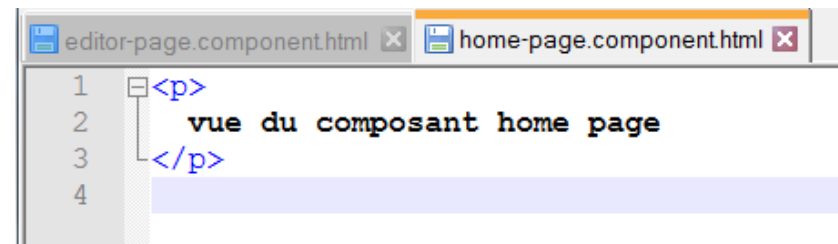


## Exemple simple :

ng new spa-app      nouvelle application  
cd spa-app  
ng serve --open      serveur de développement

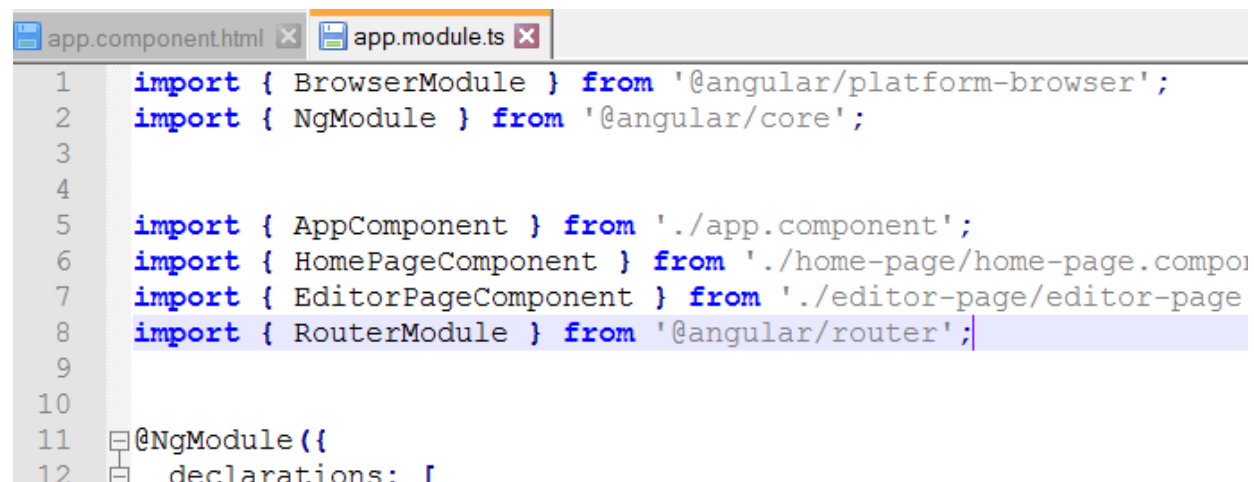
ng generate component home-page  
ng generate component editor-page

changer les vues :



```
1 <p>  
2     vue du composant home page  
3 </p>  
4
```

Rajouter RouterModule dans app.module.ts



```
1 import { BrowserModule } from '@angular/platform-browser';  
2 import { NgModule } from '@angular/core';  
3  
4  
5 import { AppComponent } from './app.component';  
6 import { HomePageComponent } from './home-page/home-page.component';  
7 import { EditorPageComponent } from './editor-page/editor-page.component';  
8 import { RouterModule } from '@angular/router';  
9  
10  
11 @NgModule({  
12     declarations: [  
13         AppComponent,  
14         HomePageComponent,  
15         EditorPageComponent,  
16     ],  
17     imports: [  
18         BrowserModule,  
19         RouterModule,  
20     ],  
21     providers: [],  
22     bootstrap: [AppComponent]  
23 })  
24 export class AppModule {}
```

# Angular : routage



rajouter les correspondances chemin dans app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { HomeComponent } from './home-page/home-page.component';
import { EditorPageComponent } from './editor-page/editor-page.component';
import { RouterModule } from '@angular/router';
```

```
@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    EditorPageComponent
  ],
  imports: [
    BrowserModule,
    RouterModule.forRoot([
      { path: '', component: HomeComponent },
      { path: 'editeur', component: EditorPageComponent }
    ])
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

# Angular : routage



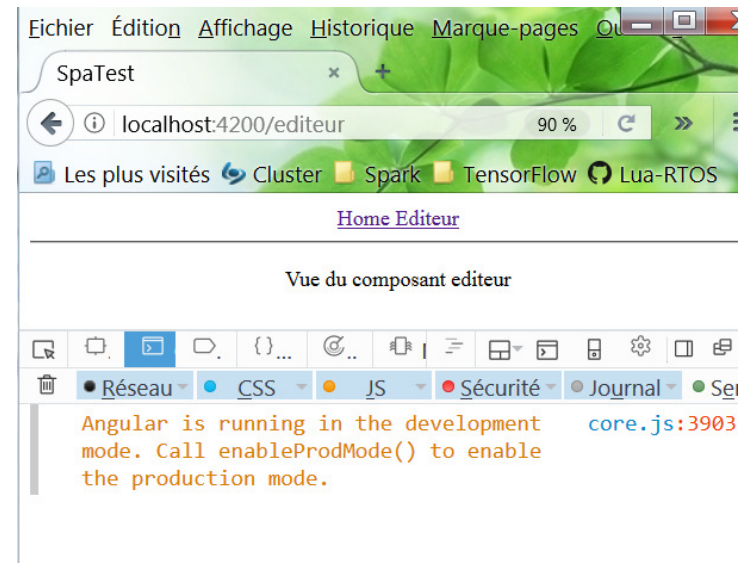
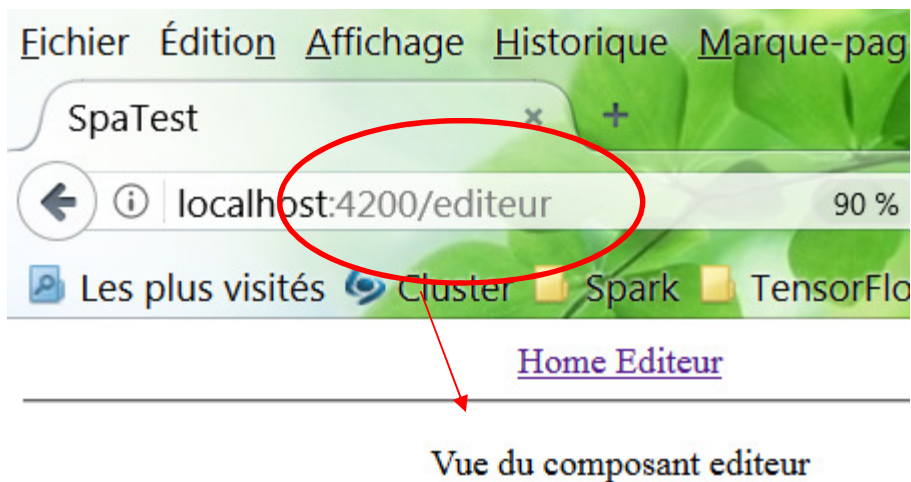
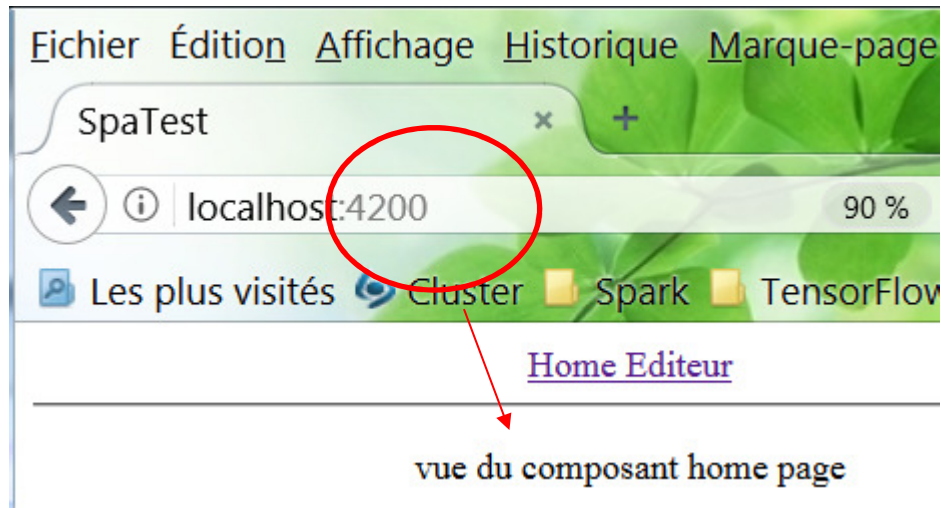
Modifier la page principale (partie navigation / partie contenu)

A screenshot of a code editor window with two tabs: 'app.component.html' and 'app.module.ts'. The 'app.component.html' tab is active, showing a list of line numbers from 1 to 10 on the left. The code in the editor is as follows:

```
1
2 <div style="text-align:center">
3   <a routerLink="/"> Home </a>
4   <a routerLink="/editeur"> Editeur </a>
5   <hr/>
6   <router-outlet></router-outlet>
7
8
9 </div>
10
```

The code is color-coded: HTML tags are blue, attributes and values are red, and text is black. A red bracket on the left side of the code block groups lines 2 through 9.

# Angular : routage



Aucun appel HTTP / serveur  
L'application angular construit les pages



# Angular : SPA



## Single Page Application : SPA

Logique de site Web différente :

*le site Web angular est conçu comme une application  
les pages web représentent l'interface de l'application*

*un site web classique : interface éclatée  
le serveur doit faire le lien  
sessions etc...  
compliqué à développer*

*Angular : les sources des modèles ts peuvent partager  
les variables*

*+*

*le client fait la majorité du travail  
le serveur sert de sauvegarde*

# Angular : Services



Accès à un serveur distant

**Jquery**

**load, get etc..**

**Pas d'intégration Angular / Modèle MVC**

**Service**

**Stream asynchrone**

**Observables**

**ReactiveX**



## RxJS + Services Angular

