

Application Introduction à Angular



Site officiel : <https://angular.io/>

Installation

Pour débiter notre projet voici un résumé de ce que nous allons faire :

- Node.js sera notre plateforme de développement JavaScript.
- Visual studio Code sera notre éditeur de code.
- Git sera notre gestionnaire de logiciel.
- Angular CLI sera notre outil. Voir : <https://angular.io/cli>

Nous installons Angular CLI à l'aide de la commande npm suivante :

```
npm install -g @angular/cli@latest
```

Installez d'abord une nouvelle application Angular à l'aide de la commande ci-dessous, exécutons la commande ci-dessous :

```
ng new my-intro-app --routing
```

```
> Which stylesheet format would you like to use? (Use arrow keys)
> CSS
SCSS [ https://sass-lang.com/documentation/syntax#scss ]
Sass [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
Less [ http://lesscss.org ]
```

Cliquer sur Entrée pour valider **CSS**

```

? Which stylesheet format would you like to use? CSS
CREATE my-intro-app/angular.json (2738 bytes)
CREATE my-intro-app/package.json (1043 bytes)
CREATE my-intro-app/README.md (1064 bytes)
CREATE my-intro-app/tsconfig.json (981 bytes)
CREATE my-intro-app/.editorconfig (274 bytes)
CREATE my-intro-app/.gitignore (548 bytes)
CREATE my-intro-app/tsconfig.app.json (263 bytes)
CREATE my-intro-app/tsconfig.spec.json (273 bytes)
CREATE my-intro-app/.vscode/extensions.json (130 bytes)
CREATE my-intro-app/.vscode/launch.json (470 bytes)
CREATE my-intro-app/.vscode/tasks.json (938 bytes)
CREATE my-intro-app/src/main.ts (214 bytes)
CREATE my-intro-app/src/favicon.ico (948 bytes)
CREATE my-intro-app/src/index.html (296 bytes)
CREATE my-intro-app/src/styles.css (88 bytes)
CREATE my-intro-app/src/app/app-routing.module.ts (245 bytes)
CREATE my-intro-app/src/app/app.module.ts (393 bytes)
CREATE my-intro-app/src/app/app.component.html (23115 bytes)
CREATE my-intro-app/src/app/app.component.spec.ts (1089 bytes)
CREATE my-intro-app/src/app/app.component.ts (216 bytes)
CREATE my-intro-app/src/app/app.component.css (0 bytes)
CREATE my-intro-app/src/assets/.gitkeep (0 bytes)
- Installing packages (npm)...

```

```

✓ Packages installed successfully.

```

```

cd my-intro-app

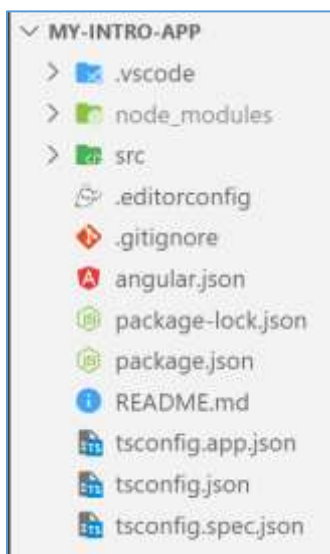
```

Ouvrir le projet dans Visual Studio code

```

code .

```



Exécuter le projet :

```

ng serve --open

```

```

? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. (y/N) n

```

Répondre n pour non.

```

details and how to change this setting, see https://angular.io/analytics. No
Global setting: enabled
Local setting: disabled
Effective status: disabled
✓ Browser application bundle generation complete.

Initial Chunk Files | Names | Raw Size
vendor.js | vendor | 2.34 MB
polyfills.js | polyfills | 333.19 kB
styles.css, styles.js | styles | 230.46 kB
main.js | main | 48.12 kB
runtime.js | runtime | 6.52 kB
| Initial Total | 2.95 MB

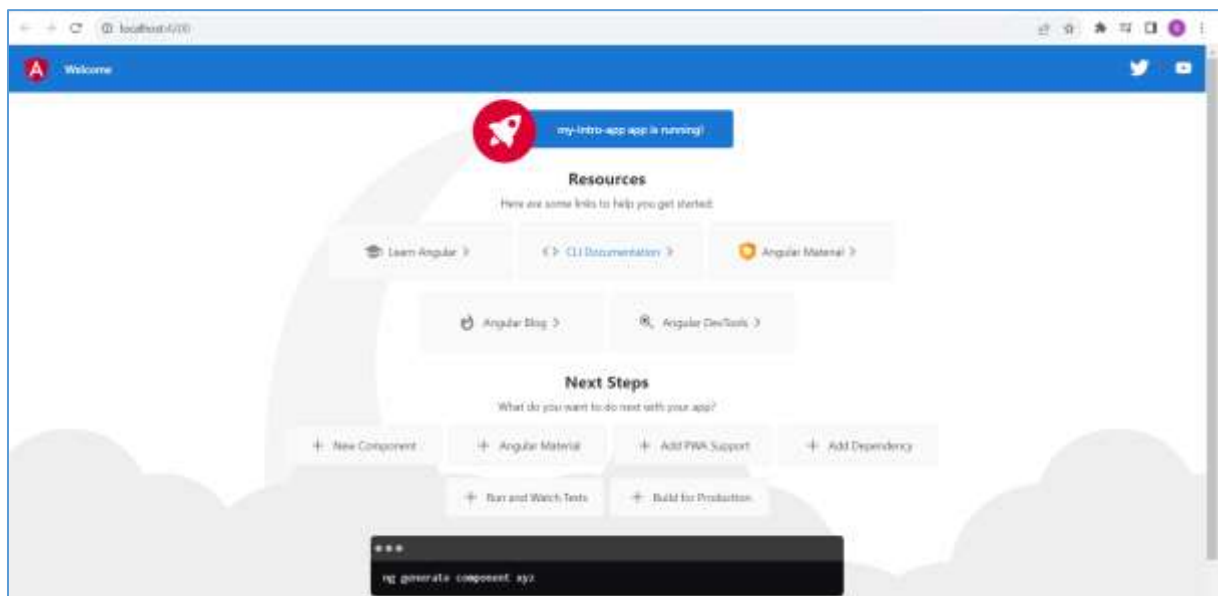
Build at: 2023-09-11T07:58:09.851Z - Hash: ba75f4d83b1498a4 - Time: 24752ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

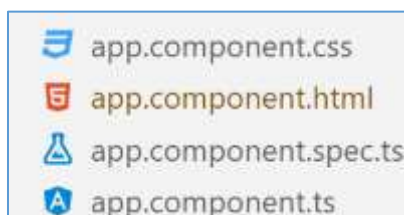
```

Page ouverte dans le navigateur avec l'adresse :

<http://localhost:4200/>



Le composant principal est app.component



Un composant Angular est un objet. Il définit une zone d'interaction de l'interface utilisateur (UI pour User Interface) et permet la réutilisation.

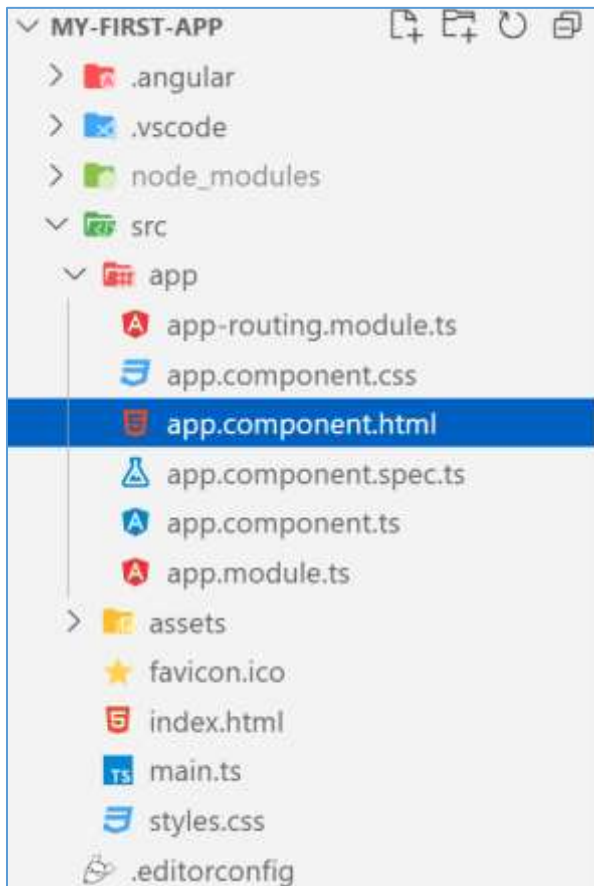
Dans Angular un composant est constitué de 3 éléments :

- Une classe de composant .ts (component class). Elle gère les données et les fonctionnalités
- Un template HTML (HTML template). Il détermine ce qui est présenté visuellement à l'utilisateur
- Les styles spécifiques au composant (component-specific styles). Ils définissent le look du composant.

Un autre composant .spec.ts qui sert pour les tests.

Ouvrir le contenu de :

src > app >  app.component.html >

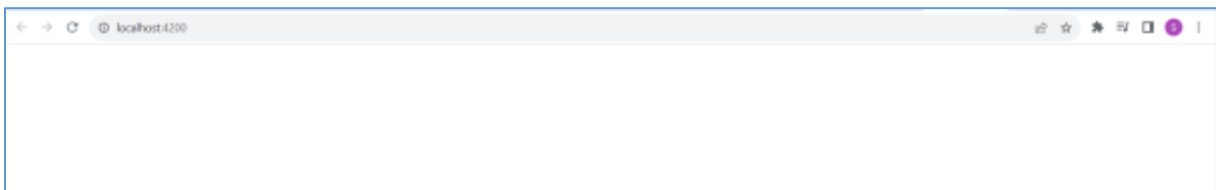


Supprimer tout le contenu sauf cette dernière ligne :

```
<router-outlet></router-outlet>
```

Cette balise permet d'injecter le bon template (la bonne Vue) directement là où est la balise.

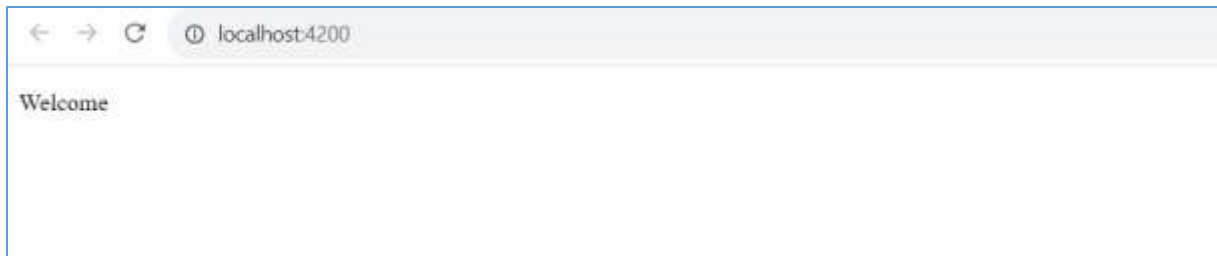
En supprimant le contenu on obtient une page vide :



src > app >  app.component.html >

On peut ajouter le code suivant :

```
<p>Welcome</p>
<router-outlet></router-outlet>
```

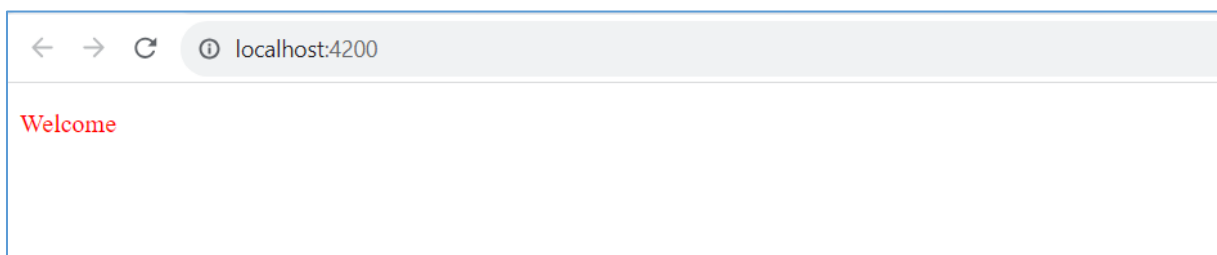


```
src > app > app.component.css >
```

```
.myP{  
  color: red;  
}
```

```
src > app > app.component.html >
```

```
<p class="myP">Welcome</p>
```



Composant Angular

Dans Angular en plus du composant principal, on peut créer d'autres composants.

La commande pour créer un nouveau composant est ***ng generate component nom_composant***

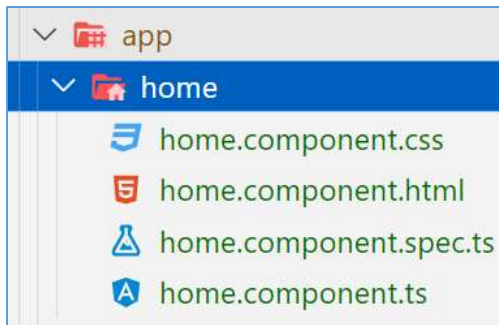
Un raccourci existe avec ***ng g c nom_composant***

On va créer le composant home avec la commande cli.

```
ng generate component home
```

On obtient :

```
CREATE src/app/home/home.component.html (19 bytes)  
CREATE src/app/home/home.component.spec.ts (545 bytes)  
CREATE src/app/home/home.component.ts (194 bytes)  
CREATE src/app/home/home.component.css (0 bytes)  
UPDATE src/app/app.module.ts (467 bytes)
```



Par défaut on obtient ce code :

```
src > app > home > home.component.html >  
1 <p>home works!</p>  
2
```

```
src > app > home > home.component.ts > ...  
1 import { Component } from '@angular/core';  
2  
3 @Component({  
4   selector: 'app-home',  
5   templateUrl: './home.component.html',  
6   styleUrls: ['./home.component.css']  
7 })  
8 export class HomeComponent {  
9  
10 }  
11
```

Si on ouvre app.module.ts

```
src > app > app.module.ts >
```

On y trouve :

```
src > app > app.module.ts > ...
You, 1 second ago | 1 author (You)
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { HomeComponent } from './home/home.component';
7
8 @NgModule({
9   declarations: [
10     AppComponent,
11     HomeComponent
12   ],
13   imports: [
14     BrowserModule,
15     AppRoutingModule
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
21
```

Si nous regardons en détail le fichier app.module.ts, nous verrons que c'est une classe qui comporte un décorateur NgModule. Et le NgModule, un peu comme les modules JavaScript, il sert à importer et exporter certains éléments au sein de notre application. En fait, c'est lui qui va regrouper à la fois tous les imports, tous les éléments de notre code source, les composants que l'on va créer, par exemple, etc. De regrouper tous ces éléments qui vont être utiles dans notre application pour les transmettre au compilateur. Et ainsi, notre compilateur pourra effectuer les bundles, qui est représenté par les gros fichiers JavaScript qui seront envoyés à notre navigateur.

Dans

```
src > app > app.component.html >
```

Ajoutons le code :

```
<p class="myP">Welcome</p>
<app-home></app-home>
<router-outlet></router-outlet>
```

app-home car dans home.component.ts on a le selector correspondant :

```
src > app > home > home.component.ts > HomeComponent
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-home',
5    templateUrl: './home.component.html',
6    styleUrls: ['./home.component.css']
7  })
8  export class HomeComponent {
9
10 }
11
```

← → ↻ localhost:4200

Welcome

home works!

Data Biding et interpolation

String interpolation

L'interpolation fait référence à l'intégration d'expressions dans du texte balisé. Par défaut, l'interpolation utilise les doubles accolades `{{` et `}}` comme délimiteurs.

Pour illustrer le fonctionnement de l'interpolation, considérons la variable `name` dans `home.component.ts` :

src > app > home > home.component.ts >

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent {
  name = "ABC corporation"
}
```


src > app > home >  home.component.html >

```
<h3> Our company : {{ name }} </h3>
```

← → ↻ ⓘ localhost:4200

Welcome

Our company : ABC corporation

src > app > home >  home.component.ts >

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent {
  name = "ABC corporation";
  imageURL="https://static.vecteezy.com/system/resources/previews/008/214/517/non_2x/abstract-geometric-logo-or-infinity-line-logo-for-your-company-free-vector.jpg";
}
```

src > app > home >  home.component.html >

```
<h3>
<img
  src={{imageURL}}

  width="200"
  alt="logo"
/>
Our company : {{ name }}
</h3>
```



Property binding

Autre méthode :

```
<h3>
<img
[src]="imageUrl"
width="200"
alt="logo"
/>
Our company : {{ name }}
</h3>
```

Utilisez `[]` pour lier la source à l'affichage. Il s'agit de property binding.

src > app > home >  home.component.ts >

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent {
  name = "ABC corporation";
  imageUrl="https://static.vecteezy.com/system/resources/previews/008/214/517/no
n_2x/abstract-geometric-logo-or-infinity-line-logo-for-your-company-free-
vector.jpg";
alternative="logo"
}
```

```
src > app > home >  home.component.html >
```

```
<h3>
<img
  [src]="imageUrl"
  width="200"
  [alt]="alternative"
/>
Our company : {{ name }}
</h3>
```

Event binding

Il permet d'écouter et de répondre aux actions des utilisateurs telles que les frappes au clavier, les mouvements de souris, les clics et les touchers.

```
src > app > home >  home.component.html >
```

```
<h3>
<img
  [src]="imageUrl"
  width="200"
  [alt]="alternative"
/>
Our company : {{ name }}
</h3>
<button (click)="change()">Change name</button>
```

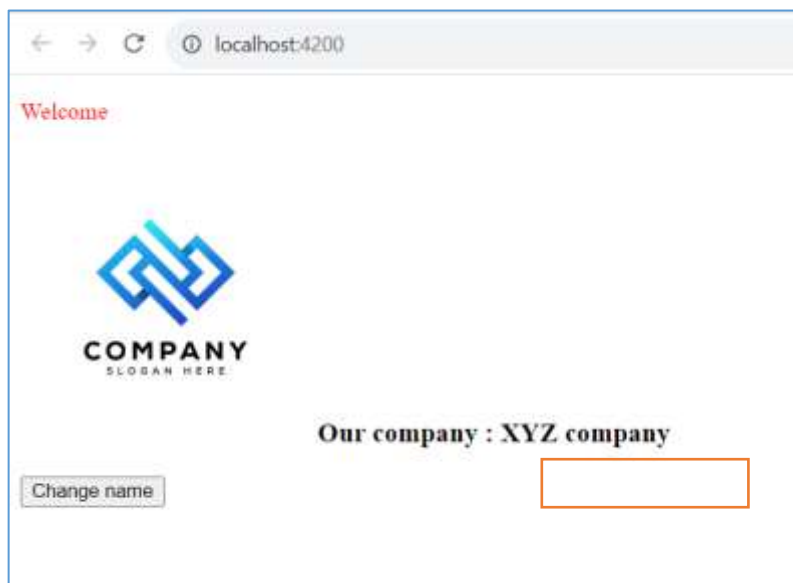
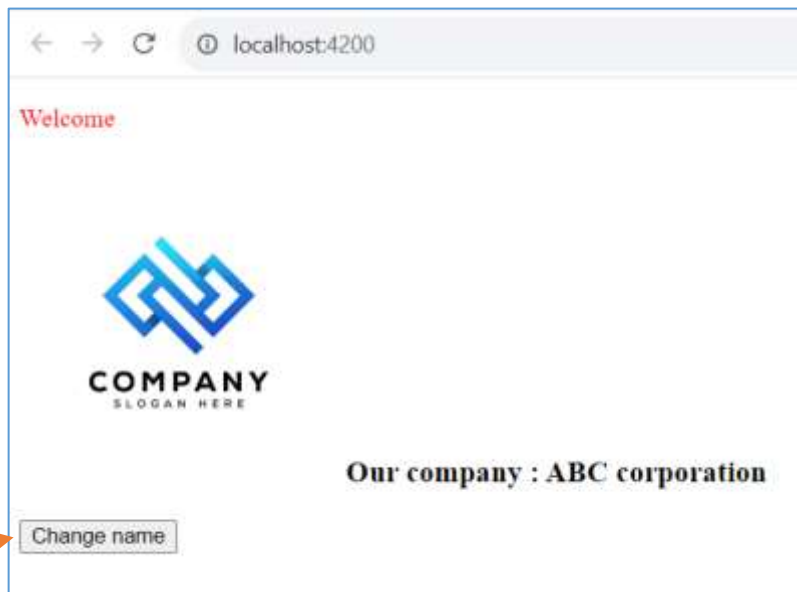
```
src > app > home >  home.component.ts >
```

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent {
  name = "ABC corporation";
  imageUrl="https://static.vecteezy.com/system/resources/previews/008/214/517/no_n_2x/abstract-geometric-logo-or-infinity-line-logo-for-your-company-free-vector.jpg";
  alternative="logo";

  change(){
    this.name="XYZ company";
  }
}
```

```
}  
}
```



two way data binding

La liaison de données bidirectionnelle dans Angular permet aux données de circuler du composant vers la vue et inversement. Il est utilisé pour afficher des informations à l'utilisateur final et lui permet d'apporter des modifications aux données sous-jacentes à l'aide de l'interface utilisateur.

Pour utiliser des champs de formulaire, il faut le déclarer dans app.module.ts

```
src > app >  app.module.ts >
```

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

```

import { AppRoutingModuleModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HomeComponent } from './home/home.component';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModuleModule,
    FormsModule,
    ReactiveFormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

Dans les applications Angular, si vous souhaitez utiliser la liaison de données bidirectionnelle pour les entrées de formulaire (**ngModel**), nous devons importer le FormsModule depuis @angular/core.

Pour ne pas avoir l'erreur « Can't bind to 'ngModel' since it isn't a known property of 'input' error »

Le module ReactiveFormsModule implémente également un service FormBuilder qui permet de simplifier la création des FormGroup, FormArray ou FormControl.

Pour ne pas avoir l'erreur "Can't bind to 'formGroup' since it isn't a known property of 'form' Cette erreur se produit lorsque vous utilisez formGroup sur un élément form sans avoir importé ReactiveFormsModule dans votre module.

src > app > home >  home.component.html >

```

<h3>
<img
  [src]="imageUrl"
  width="200"
  [alt]="alternative"
/>
Our company : {{ name }}
</h3>
New name : <input type="text" [(ngModel)]="newName"/>
<button (click)="change()">Change name</button>

```

src > app > home >  home.component.ts >

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

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

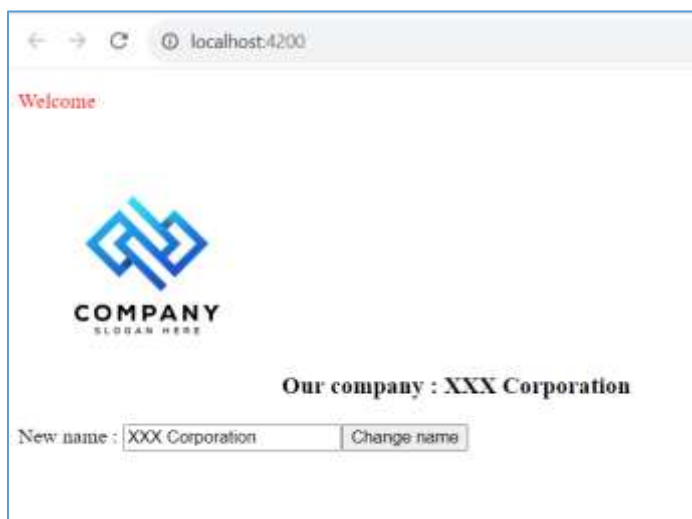
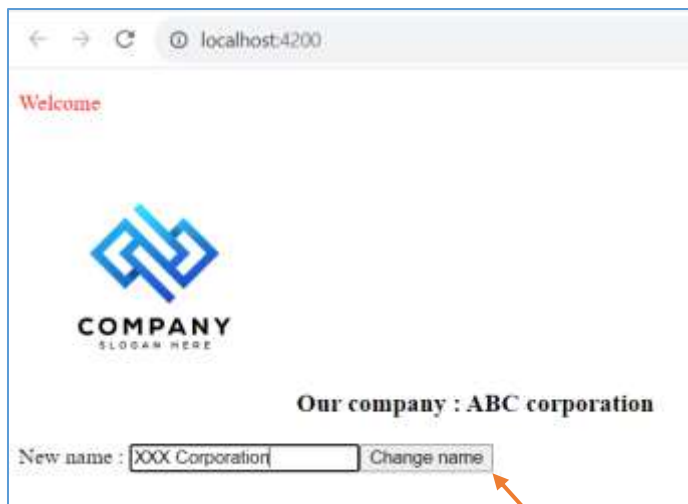
  name = "ABC corporation";
  imageURL="https://static.vecteezy.com/system/resources/previews/008/214/517/non_2x/abstract-geometric-logo-or-infinity-line-logo-for-your-company-free-vector.jpg";
  alternative="logo";

  newName="";

  change(){
    this.name=this.newName;
  }

}
```





ngModel est une directive d'Angular qui permet de lier une propriété d'un composant à un élément de formulaire HTML. Cela signifie qu'elle permet de synchroniser les données saisies dans un formulaire avec les propriétés de votre composant, et inversement.

On peut changer le code en supprimant le bouton et en utilisant une seule variable :

src > app > home >  home.component.html >

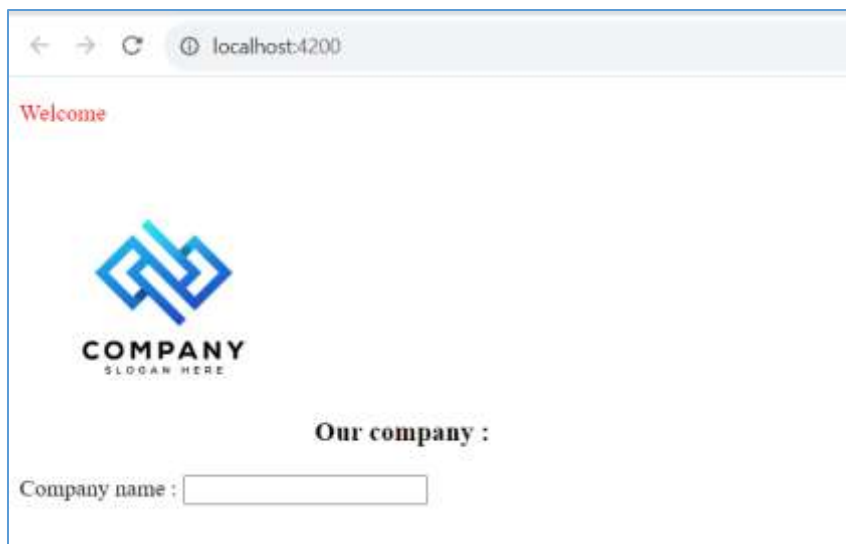
```
<h3>
<img
  [src]="imageUrl"
  width="200"
  [alt]="alternative"
/>
Our company : {{ name }}
</h3>
Company name : <input type="text" [(ngModel)]="name"/>
```

```
src > app > home >  home.component.ts >
```

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

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

  name = "";
  imageURL="https://static.vecteezy.com/system/resources/previews/008/214/517/non_2x/abstract-geometric-logo-or-infinity-line-logo-for-your-company-free-vector.jpg";
  alternative="logo";
}
```



Affichage instantané de la saisie.



Les directives

Les directives d'Angular sont des classes qui ajoutent un comportement supplémentaire aux éléments du DOM.

*ngFor

src > app > home >  home.component.ts >

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

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

  name = "";
  imageURL="https://static.vecteezy.com/system/resources/previews/008/214/517/no
n_2x/abstract-geometric-logo-or-infinity-line-logo-for-your-company-free-
vector.jpg";
  alternative="logo";

  todos = [{title: 'Learn angular', done: false},{title: 'Create project
angular', done: false}];

  change(){
    this.todos.map(value => {
      value.done = true;
    });
  }
}
```

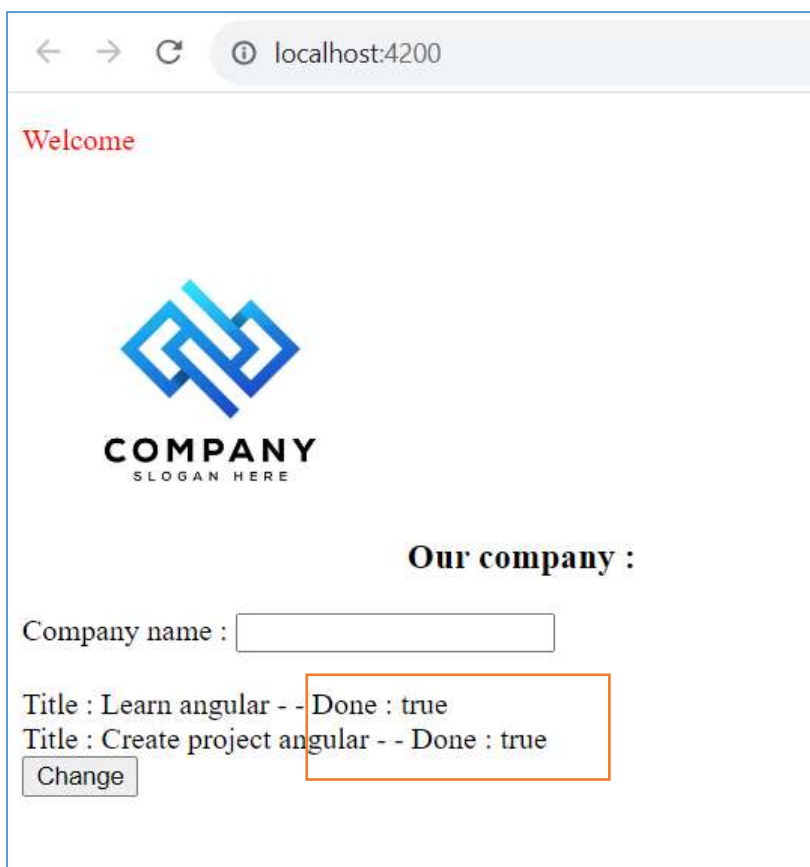
```
}
```

```
}
```

src > app > home >  home.component.html >

```
<div>
<h3>
<img
  [src]="imageUrl"
  width="200"
  [alt]="alternative"
/>
Our company : {{ name }}
</h3>
Company name : <input type="text" [(ngModel)]="name"/>
</div>
<div>
  <br/>
</div>
<div *ngFor="let todo of todos">
  <div>
    Title : {{todo.title}}
    - -
    Done : {{todo.done}}
  </div>
</div>

<button (click)="change()" >Change</button>
```

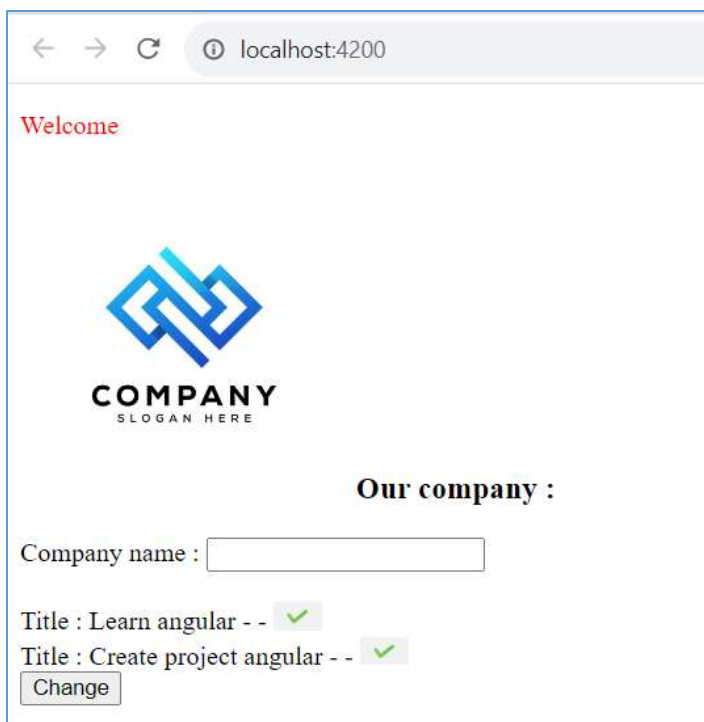
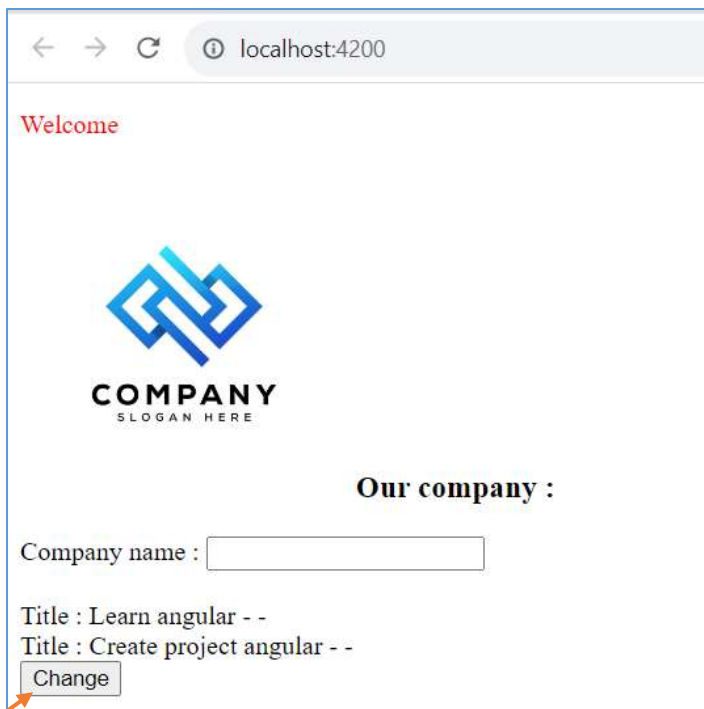


*ngIf

src > app > home >  home.component.html >

```
<div>
<h3>
<img
  [src]="imageUrl"
  width="200"
  [alt]="alternative"
/>
Our company : {{ name }}
</h3>
Company name : <input type="text" [(ngModel)]="name"/>
</div>
<div>
  <br/>
</div>
<div *ngFor="let todo of todos">
  <div>
    Title : {{todo.title}}
    - -
    <span *ngIf="todo.done "> </span>
  </div>
</div>

<button (click)="change()" >Change</button>
```



ng-template

src > app > home >  home.component.html >

```
<div>  
<h3>
```

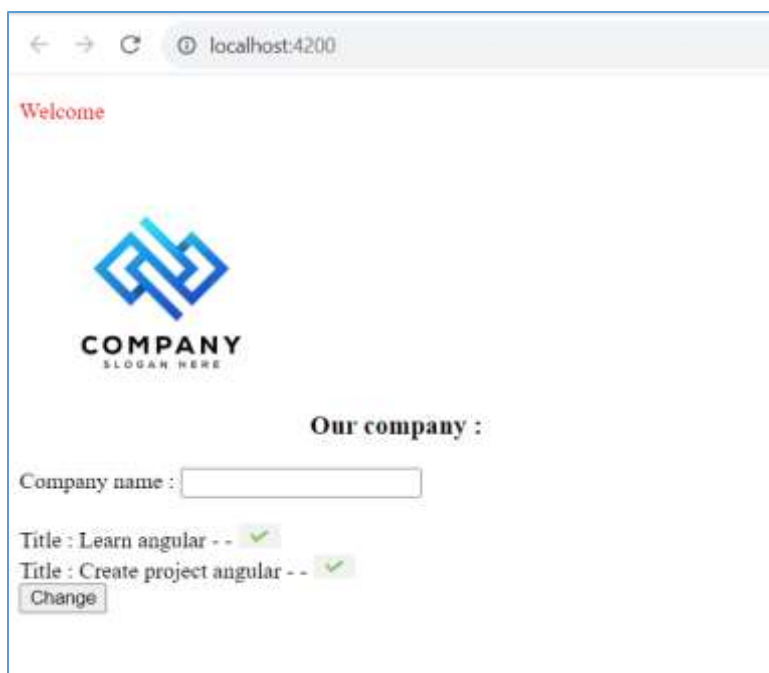
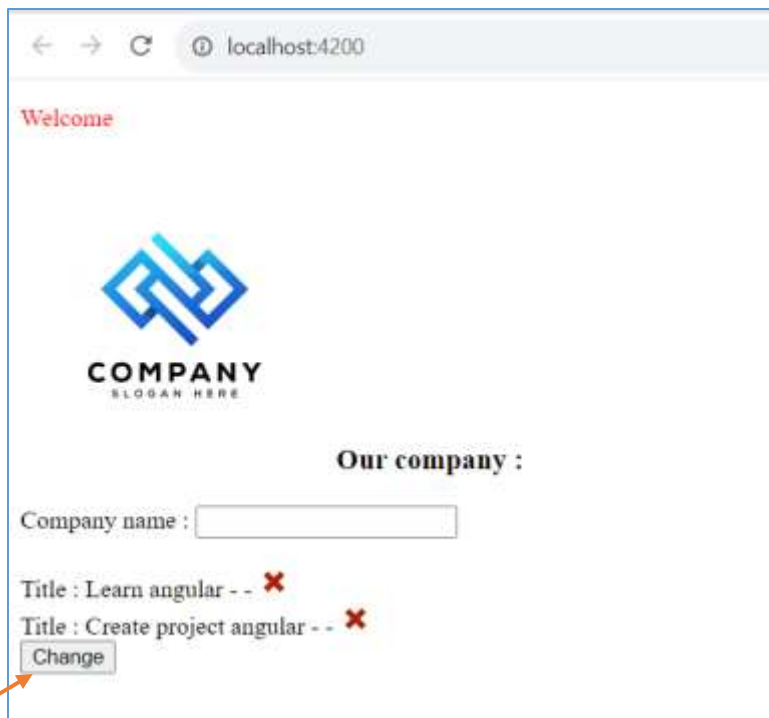
```

<img
[src]="imageUrl"
width="200"
[alt]="alternative"
/>
Our company : {{ name }}
</h3>
Company name : <input type="text" [(ngModel)]="name"/>
</div>
<div>
    <br/>
</div>
<div *ngFor="let todo of todos">
    <div>
        Title : {{todo.title}}
        - -
        <span *ngIf="todo.done ; else crossImage"> </span>
    </div>
</div>

<button (click)="change()" >Change</button>

<ng-template #crossImage>
    
</ng-template>

```




ngSwitch

```
src > app > home >  home.component.ts >
```

Ajouter l'attribut :

```
theme="Default";
```

```
src > app > home >  home.component.html >
```

Ajouter le code :

```
<div>
Theme : <select [(ngModel)]="theme">
  <option>Default</option>
  <option>Light</option>
  <option>Dark</option>
</select>
</div>
<div [ngSwitch]="theme">
  <div *ngSwitchCase="'Default'"> background : White</div>
  <div *ngSwitchCase="'Light'"> background : Grey</div>
  <div *ngSwitchCase="'Dark'"> background : Black </div>
</div>
```

Theme : ▼
background : White

Theme : ▼
background : Grey

ngStyle


```
src > app > home >  home.component.html >
```

```
<div [ngStyle]="{backgroundColor: theme}">
<div>
Theme : <select [(ngModel)]="theme">
  <option value="white">Default</option>
  <option value="grey">Light</option>
  <option value="black">Dark</option>
</select>
</div>
<div [ngSwitch]="theme">
  <div *ngSwitchCase="'white'"> background : White</div>
  <div *ngSwitchCase="'grey'"> background : Grey</div>
  <div *ngSwitchCase="'black'"> background : Black </div>
```



```
</div>
```

```
</div>
```

Theme : 
background : Grey

ngClass

src > app > home >  home.component.css >

```
.bg-dark{  
  color :white;  
}  
  
.bg-not-dark  
{  
  color :black;  
}
```

src > app > home >  home.component.html >

```
<div [ngStyle]="{backgroundColor: theme}" [ngClass]="{'bg-dark': theme  
=== 'black', 'bg-not-dark' : theme !== 'black'}">  
<div>  
Theme : <select [(ngModel)]="theme">  
  <option value="white">Default</option>  
  <option value="grey">Light</option>  
  <option value="black">Dark</option>  
</select>  
</div>  
<div [ngSwitch]="theme">  
  <div *ngSwitchCase="'white'"> background : White</div>  
  <div *ngSwitchCase="'grey'"> background : Grey</div>  
  <div *ngSwitchCase="'black'"> background : Black </div>  
</div>  
</div>
```

Theme :
background : White

Theme :
background : Black

Les pipes

Les pipes sont un moyen puissant de transformer des données dans des applications Angular. Ils peuvent être utilisés pour formater des données, les convertir en différentes unités ou même effectuer des calculs complexes.

Les pipes sont très utiles lorsqu'il s'agit de gérer des données au sein de l'interpolation « {{ | }} ». Afin de transformer les données, nous utilisons le caractère |.

On va créer le composant formations

```
ng g c formations
```

```
CREATE src/app/formations/formations.component.html (25 bytes)
CREATE src/app/formations/formations.component.spec.ts (587 bytes)
CREATE src/app/formations/formations.component.ts (218 bytes)
CREATE src/app/formations/formations.component.css (0 bytes)
UPDATE src/app/app.module.ts (675 bytes)
```

```
src > app > formations > A formations.component.ts >
```

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

@Component({
  selector: 'app-formations',
  templateUrl: './formations.component.html',
  styleUrls: ['./formations.component.css']
})
export class FormationsComponent {
  d= new Date();
  planning=[
    {sujet:"html5/css3",volume:25,prix:250,lieu:"rue ankara ariana tunis
tunisie",date:this.d,ouverte:true },
    {sujet:"react",volume:30,prix:400,lieu:"rue de la bourse 34 les berges du
lac 2 tunis tunisie",date:this.d,ouverte:false},
    {sujet:"angular",volume:20,prix:350,lieu:"rue ankara ariana tunis
tunisie",date:this.d,ouverte:true}
  ]
}
```

```
}
```

```
src > app > formations >  formations.component.html >
```

```
<div>
  <table border="1">
    <thead>
      <tr>
        <th> Sujet </th>
        <th> Volume </th>
        <th> Prix </th>
        <th> Lieu </th>
        <th> Date </th>
        <th> Inscription ouverte </th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let p of planning">
        <td>{{ p.sujet | lowercase }}</td>
        <td>{{ p.volume | number:'1.1-2' }}</td>
        <td>{{ p.prix | currency:'TND' }}</td>
        <td>{{ p.lieu | slice: -13 | titlecase }}</td>
        <td>{{ p.date | date:'fullDate' }}</td>
        <td>{{ (p.ouverte ? 'En cours' : 'Fermée') | uppercase }}</td>
      </tr>
    </tbody>
  </table>
</div>
```

Pour le pipe number nous avons utilisé l'option :

```
{minIntegerDigits}.{minFractionDigits}-{maxFractionDigits}
```

- minIntegerDigits: Le nombre minimum de chiffres entiers avant la virgule décimale. La valeur par défaut est 1.
- minFractionDigits: Le nombre minimum de chiffres après la virgule. La valeur par défaut est 0.
- maxFractionDigits: Le nombre maximum de chiffres après la virgule. La valeur par défaut est 3.

```
src > app >  app.component.html >
```

```
<p class="myP">Welcome</p>
<app-home></app-home>
<app-formations></app-formations>
<router-outlet></router-outlet>
```

Sujet	Volume	Prix	Lieu	Date	Inscription ouverte
html5/css3	25.0	TND250.000	Tunis Tunisie	Tuesday, September 12, 2023	EN COURS
react	30.0	TND400.000	Tunis Tunisie	Tuesday, September 12, 2023	FERMÉE
angular	20.0	TND350.000	Tunis Tunisie	Tuesday, September 12, 2023	EN COURS

ng generate pipe datePipe

```
CREATE src/app/date-pipe.pipe.spec.ts (196 bytes)
CREATE src/app/date-pipe.pipe.ts (221 bytes)
UPDATE src/app/app.module.ts (742 bytes)
```

src > app >  date-pipe.pipe.ts >

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'datePipe'
})
export class DatePipePipe implements PipeTransform {

  transform(value: unknown, ...args: unknown[]): unknown {
    return null;
  }
}
```

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'datePipe'
})
export class DatePipePipe implements PipeTransform {

  transform(date: Date): string {
    return date.toLocaleString();
  }
}
```

src > app > formations >  formations.component.html >

```
<tr *ngFor="let p of planning">
  <td>{{ p.sujet | lowercase }}</td>
  <td>{{ p.volume | number:'1.1-2' }}</td>
  <td>{{ p.prix | currency:'TND' }}</td>
  <td>{{ p.lieu | slice: -13 | titlecase }}</td>
  <td>{{ p.date | datePipe }}</td>
  <td>{{ (p.ouverte ? 'En cours' : 'Fermée') | uppercase }}</td>
```

</tr>

Sujet	Volume	Prix	Lieu	Date	Inscription ouverte
html5/css3	25.0	TND250.000	Tunis Tunisie	12/09/2023 16:11:01	EN COURS
react	30.0	TND400.000	Tunis Tunisie	12/09/2023 16:11:01	FERMÉE
angular	20.0	TND350.000	Tunis Tunisie	12/09/2023 16:11:01	EN COURS