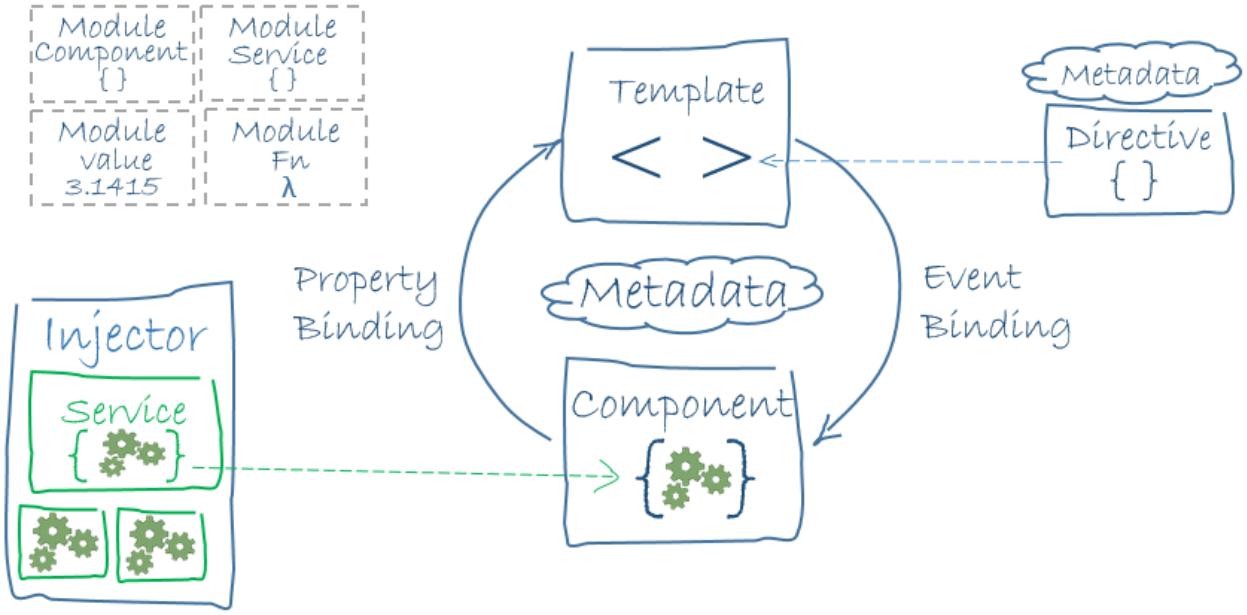
Quels sont les composants clés de Angular?



- **Composant:** Il s'agit des éléments de base d'une application angulaire permettant de contrôler les vues HTML.
- **Modules:** ensemble de blocs de construction de base angulaires tels que composant, directives, services, etc.
- **Templates:** Ceci représente ***les vues*** d'une application angulaire.
- **Services:** Il est utilisé pour créer des composants qui peuvent être partagés sur l'ensemble de l'application.
- **Métadonnées:** Ceci peut être utilisé pour ***ajouter plus de donnée***s à une classe angulaire.

Que sont les directives?

Les directives ajoutent ***un comportement à un élément DOM existant ou à une instance de composant existante.***

```
import { Directive, ElementRef, Input } from '@angular/core';

@Directive({ selector: '[myHighlight]' })
export class HighlightDirective {
   constructor(el: ElementRef) {
      el.nativeElement.style.backgroundColor = 'yellow';
   }
}
```

Now this directive extends HTML element behavior with a yellow background as below

```
< p myHighlight >Highlight me!< /p>
```

What are components?

Components are the most basic UI building block of an Angular app .

`

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

@Component ({
  selector: 'my-app',
  template: ` <div>
 <h1>{{title}}</h1>
 <div>Learn Angular6 with examples</div>
 </div> `,
})

export class AppComponent {
```

```
  title: string = 'Welcome to Angular world';
}
```

Qu'est-ce qu'un template?

Un modèle est une vue HTML dans laquelle vous pouvez afficher des données en liant des contrôles aux propriétés d'un composant angulaire.

Utilisation d'un modèle en ligne avec une syntaxe de modèle

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

@Component ({
  selector: 'my-app',
  template: '
 <div>
      <h1>{{title}}</h1>
      <div>Learn Angular</div>
    </div>
  '
})

export class AppComponent {
  title: string = 'Hello World';
}
```

Utilisation d'un fichier de modèle séparé tel que app.component.html

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

```
@Component ({
  selector: 'my-app',
  templateUrl: 'app/app.component.html'
})

export class AppComponent {
  title: string = 'Hello World';
}
```

Qu'est-ce qu'un module?

Les modules sont des limites logiques dans votre application et l'application est divisée en modules distincts pour séparer les fonctionnalités de votre application.

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

@NgModule ({
  imports:      [ BrowserModule ],
  declarations: [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

Qu'est-ce que HttpClient et ses avantages?

La plupart des applications frontales communiquent avec les services principaux via un protocole HTTP à l'aide de l'interface XMLHttpRequest ou de l'API fetch ().

```
import { HttpClientModule } from '@angular/common/http';
```

Importer HttpClient dans le module racine:

```
import { HttpClientModule } from '@angular/common/http';
@NgModule({
 imports: [
  BrowserModule,
  // import HttpClientModule after BrowserModule.
  HttpClientModule,
 ],
 .....
 })
 export class AppModule {}
```

Injecter le HttpClient dans l'application

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

const userProfileUrl: string = 'assets/data/profile.json';

@Injectable()
export class UserProfileService {
 constructor(private http: HttpClient) { }
}

getUserProfile() {
 return this.http.get(this.userProfileUrl);
}
```

Créer un composant pour le service abonné

```
fetchUserProfile() {
  this.userProfileService.getUserProfile()
    .subscribe((data: User) => this.user = {
      id: data['userId'],
      name: data['firstName'],
      city:  data['city']
    });
}
```

Que sont les pipes?

Un canal prend des données en entrée et les transforme en une sortie souhaitée. Par exemple, prenons un tuyau pour transformer la propriété d'anniversaire d'un composant en une date conviviale à l'aide d'un tuyau de date.

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

@Component({
  selector: 'app-birthday',
  template: `<p>Birthday is {{ birthday | date }}</p>`
})
export class BirthdayComponent {
  birthday = new Date(1987, 6, 18); // June 18, 1987
}
```

Comment catégorisez-vous les types de liaison de données?

Les types de liaison peuvent être regroupés en trois catégories distinguées par la direction du flux de données. Ils sont listés ci-dessous,

De la source à la vue

- {{expression}}
- [target]="expression"
- bind-target="expression"

De la vue à la source

- (target)="statement"
- on-target="statement"

Voir à la source pour voir

- [(target)]="expression"
- bindon-target="expression"

Qu'est-ce que l'interpolation?

L'interpolation est une syntaxe spéciale que Angular **convertit en liaison de propriété.** Il est représenté par des doubles accolades ({{}}). Le texte entre les accolades est souvent le nom d'une propriété de composant.

```
< h3>
  {{title}}
  < img src="{{url}}" style="height:30px">
< /h3>
```

Quel est le but de la directive ngIf?

La directive angular ngIf insère ou supprime un élément basé sur une condition de vérité / falsification. Prenons un exemple pour afficher un message si l'âge de l'utilisateur est supérieur à 18 ans.

```
< p *ngIf="user.age > 18">You are not eligible for student pass!< /p>
```

Quel est le but de la directive ngFor?

Nous utilisons la directive Angular ngFor dans le modèle pour afficher chaque élément de la liste. Par exemple, ici nous parcourons la liste des utilisateurs.

```
< li *ngFor="let user of users"> {{ user }} < /li>
```

Qu'est-ce qu'un service?

Un service est utilisé lorsqu'une fonctionnalité commune doit être fournie à différents modules.

```
import { Injectable } from '@angular/core';
import { Http } from '@angular/http';

@Injectable() // The Injectable decorator is required for dependency injection to work
export class RepoService{
 constructor(private http: Http){
 }

 fetchAll(){
  return this.http.get('https://api.github.com/repositories').map(res => res.json());
 }
```

```
}
```

Quelle est la difference entre constructor et ngOnInit?

Les classes TypeScript ont une méthode par défaut appelée constructeur qui est normalement utilisée aux fins d'initialisation.

Alors que la méthode ngOnInit est spécifique à Angular, elle est notamment utilisée pour définir les liaisons angulaires. Même si le constructeur est appelé en premier, il est préférable de déplacer toutes vos liaisons angulaires vers la méthode ngOnInit.

```
export class App implements OnInit{
  constructor(){
    //called first time before the ngOnInit()
  }

  ngOnInit(){
    //called after the constructor and called  after the first ngOnChanges()
  }
}
```

Qu'est-ce qu'un CLI angular?

Angulaire CLI (Command Line Interface) est une interface de ligne de commande pour échafaudage et construire des modules angulaires des applications utilisant nodejs de style (CommonJS). Vous devez installer en utilisant la commande ci-dessous npm.

1. **Creating New Project:** ng new
2. **Generating Components, Directives & Services:** ng generate/g The different types of commands would be,

- ng generate class my-new-class: add a class to your application
- ng generate component my-new-component: add a component to your application
- ng generate directive my-new-directive: add a directive to your application
- ng generate enum my-new-enum: add an enum to your application
- ng generate module my-new-module: add a module to your application
- ng generate pipe my-new-pipe: add a pipe to your application
- ng generate service my-new-service: add a service to your application

3. **Running the Project:** ng serve

Qu'est-ce que les métadonnées?

Les métadonnées sont utilisées pour décorer une classe afin qu'elle puisse configurer le comportement attendu de la classe. ***Les métadonnées sont représentées par les décorateurs***

Décorateurs de classe, par exemple @Component et @NgModule

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

@Component({
  selector: 'my-component',
  template: '<div>Class decorator</div>',
})
export class MyComponent {
  constructor() {
    console.log('Hey I am a component!');
  }
}
```

```
@NgModule({
  imports: [],
  declarations: [],
})
export class MyModule {
  constructor() {
    console.log('Hey I am a module!');
  }
}
```

Décorateurs de propriétés Utilisés pour les propriétés dans les classes, par exemple. @Input et @Output

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

@Component({
    selector: 'my-component',
    template: '<div>Property decorator</div>'
})

export class MyComponent {
    @Input()
    title: string;
}
```

Décorateurs de méthodes Utilisés pour les méthodes dans les classes, par exemple. @HostListener

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

@Component({
    selector: 'my-component',
    template: '<div>Method decorator</div>'
```

```
})
export class MyComponent {
  @HostListener('click', ['$event'])
  onHostClick(event: Event) {
    // clicked, `event` available
  }
}
```

Décorateurs de paramètres Utilisés pour les paramètres dans les constructeurs de classe, par exemple. @Injecter

```
import { Component, Inject } from '@angular/core';
import { MyService } from './my-service';

@Component({
  selector: 'my-component',
  template: '<div>Parameter decorator</div>'
})
export class MyComponent {
  constructor(@Inject(MyService) myService) {
    console.log(myService); // MyService
  }
}
```

Qu'est-ce qu'une liaison de données?

La liaison de données est un concept fondamental dans Angular et permet de définir la communication entre un composant et le DOM, ce qui facilite grandement la définition d'applications interactives sans se soucier de l'extraction ou de l'extraction de données.

Du composant au DOM: Interpolation: {{valeur}}: Ajoute la valeur d'une propriété du composant.

< li> Nom: {{user.name}} < / li>

< li> Adresse: {{user.address}} < / li>

Liaison de propriété: [propriété] = "valeur": la valeur est transmise du composant à la propriété spécifiée ou à l'attribut HTML simple.

<input = "email" [value] = "user.email">

Du DOM au composant : Association d'événement: (événement) = "fonction" : Lorsqu'un événement DOM spécifique se produit (par exemple: clic, modification, incrustation), appelez la méthode spécifiée dans le composant.

<button (click) = "logout ()"> </ button>

Liaison bidirectionnelle: Liaison de données bidirectionnelle: [(ngModel)]] = "valeur": La liaison de données bidirectionnelle permet de faire circuler les données dans les deux sens. Par exemple, dans l'extrait de code ci-dessous, l'entrée de courrier électronique DOM et la propriété de courrier électronique du composant sont synchronisées.

<input type = "email" [(ngModel)] = "user.email">

Quels sont les hooks de cycle de vie disponibles?

La description de chaque méthode de cycle de vie est la suivante.

- **ngOnChanges:** lorsque la valeur d'une propriété liée aux données change, cette méthode est appelée.
- **ngOnInit:** Ceci est appelé chaque fois que l'initialisation de la directive / composante après que Angular a d'abord affiché les propriétés liées aux données se produit.
- **ngDoCheck:** Ceci est pour la détection et pour agir sur les changements que Angular ne peut pas ou ne détectera pas par lui-même.

- **ngAfterContentInit:** Ceci est appelé après le contenu externe des projets angulaires dans la vue du composant.
- **ngAfterContentChecked:** Ceci est appelé en réponse après que Angular vérifie le contenu projeté dans le composant.
- **ngAfterViewInit:** Ceci est appelé en réponse après qu'Angular initialise les vues et les vues enfants du composant.
- **ngAfterViewChecked:** Ceci est appelé en réponse après que Angular a vérifié les vues et les vues enfants du composant.
- **ngOnDestroy:** Il s'agit de la phase de nettoyage juste avant qu'Annular ne supprime la directive / le composant.