



Angular + Component

Apa itu Angular Component?

Komponen adalah building block utama atau unit kecil yang membentuk aplikasi angular, class yang menggunakan decorator **@Component()** di dalamnya. Karena terpecah menjadi unit-unit kecil, memudahkan kita untuk maintenance kode.

Setiap komponen di Angular akan terdiri dari:

- Sebuah **template** HTML, yang akan berisi hal yang dirender ke dalam suatu halaman.
- Typescript **class** yang merupakan behaviour.
- A CSS **selector** sebagai penanda penggunaan komponen.
- Optional, CSS **styles** yang diaplikasikan ke template.

Terdapat 2 cara pembuatan komponen:

1. Dengan menggunakan Angular CLI
2. Pembuatan secara manual.

Membuat Component: Dengan Angular CLI

- Di terminal, pindahkan direktori ke aplikasi angular.
- Jalankan perintah `ng generate component <component-name>`,

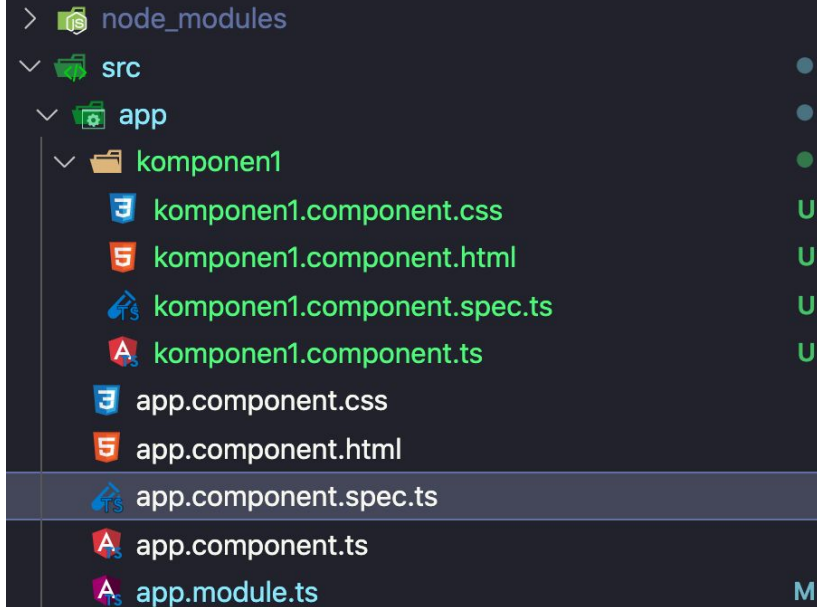
`<component-name>` adalah nama dari komponen baru.

Secara default, perintah ini akan otomatis mengenerate:

- Folder yang bernamakan `<component-name>`. Dimana didalam folder ini terdapat file:
 - File component, `<component-name>.component.ts`
 - File template HTML, `<component-name>.component.html`
 - File CSS, `<component-name>.component.css`
 - File testing, `<component-name>.component.spec.ts`

Membuat Component: Dengan Angular CLI

```
> ng generate component komponen1
CREATE src/app/komponen1/komponen1.component.css (0 bytes)
CREATE src/app/komponen1/komponen1.component.html (24 bytes)
CREATE src/app/komponen1/komponen1.component.spec.ts (647 bytes)
CREATE src/app/komponen1/komponen1.component.ts (287 bytes)
UPDATE src/app/app.module.ts (408 bytes)
```



The screenshot shows a file explorer with the following structure:

- node_modules
- src
 - app
 - komponen1
 - komponen1.component.css
 - komponen1.component.html
 - komponen1.component.spec.ts
 - komponen1.component.ts
 - app.component.css
 - app.component.html
 - app.component.spec.ts
 - app.component.ts
 - app.module.ts

Contoh pembuatan komponen baru,
dengan nama “komponen1”.

Membuat Component: Secara Manual (Inline html & css)

Dalam pembuatan komponen disamping, anotasi @Component merupakan hal yang penting untuk mendefinisikan komponen.

Dimana terdapat beberapa seperti:

1. **selector** untuk identifikasi pemanggilan komponen..
2. **template** html
3. **styles** css
4. **providers** biasanya berisi services (ex: API service) yang digunakan dalam komponen tersebut)

*Contoh disamping pembuatan komponen secara manual bernama “component-overview”

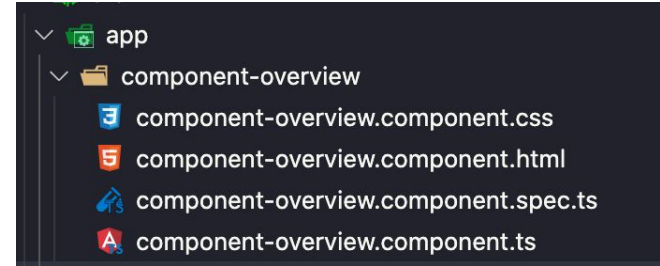
```
TS component-overview.component.3.ts ×
src > app > component-overview > TS component-overview.component.3.ts
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-component-overview',
5    template: '<h1>Hello World!</h1>',
6    styles: ['h1 { font-weight: normal; }']
7  })
8
9  export class ComponentOverviewComponent {
10
11  }
12
```

Membuat Component: Secara Manual

Bila template dan style ingin di file berbeda, bisa dilakukan cara:

1. Buatlah file css, html, spec.ts, dan ts. Dengan contoh penamaan:
`<component-name>.component.<extension>`
2. Di dalam file `<component-name>.component.ts`, masukkan kode disamping ini.
3. Selector, templateUrl dan styleUrls bisa dimasukkan path ke file html, dan css yang sebelumnya dibuat.
4. Penamaan selector dibebaskan, namun harus representatif dan harus unik dari komponen lainnya.

*Contoh disamping pembuatan komponen secara manual bernama "component-overview"



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

Deklarasi Komponen di Module

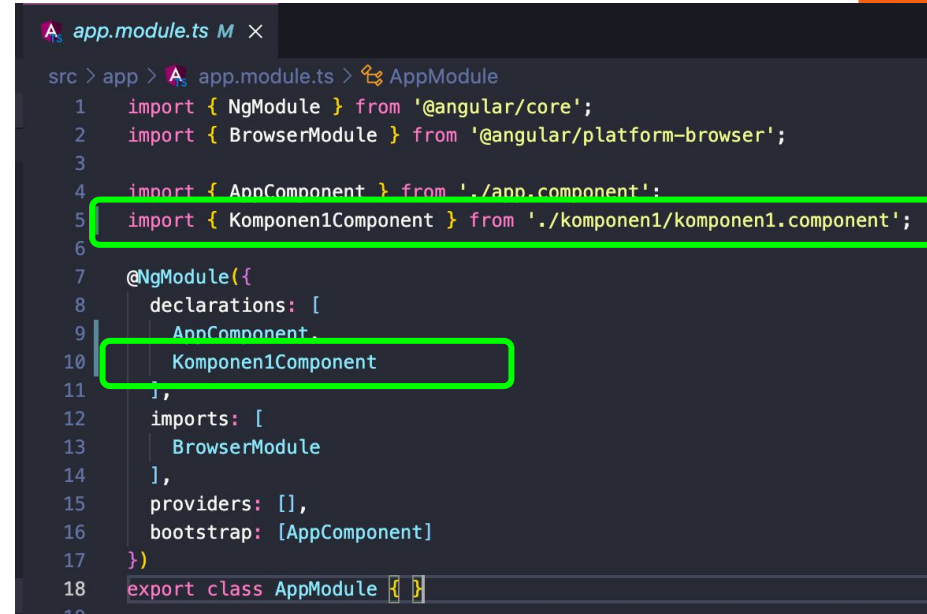
Setiap komponen yang dibuat, **harus dideklarasikan**, agar bisa dikenali oleh aplikasi Angular.

@NgModules berguna untuk mendefine module yang akan dipakai aplikasi, termasuk komponen-komponen.

Contoh disamping mendeklarasikan komponen di file `app.module.ts`.

Importlah class dalam file

`<component-name>.component.ts` dan masukkan kedalam `declarations @NgModule`.



```
src > app > A app.module.ts > AppModule
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  import { AppComponent } from './app.component';
5  import { Komponen1Component } from './komponen1/komponen1.component';
6
7  @NgModule({
8    declarations: [
9      AppComponent,
10     Komponen1Component
11   ],
12   imports: [
13     BrowserModule
14   ],
15   providers: [],
16   bootstrap: [AppComponent]
17 })
18 export class AppModule {}
```

Render Komponen ke Browser

Jika komponen sudah dideklarasikan dengan @NgModule (di slide sebelum ini), Kita dapat menggunakan selector komponen di file html (misal di app.component.html), yang akan merender isi dari file html <component-name>.component.html.

```
komponen1.component.ts U x
src > app > komponen1 > A komponen1.component.ts > ...
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-komponen1',
5    templateUrl: './komponen1.component.html',
6    styleUrls: ['./komponen1.component.css']
7  })
8  export class Komponen1Component implements OnInit {
9
10   constructor() { }
11
12   ngOnInit(): void {
13   }
14
15 }
```

```
app.component.html M x
src > app > app.component.html > ...
1  <app-komponen1></app-komponen1>
```


Render Komponen ke Browser

Ketika aplikasi dijalankan, maka akan terlihat hasil render komponen di halaman browser.

```
app.component.html M x
src > app > app.component.html > ...
1 | <app-komponen1></app-komponen1>
```

```
komponen1.component.html U x
src > app > komponen1 > komponen1
1 | <p>komponen1 works!</p>
```



Menentukan hierarki Component Parent-Child

Untuk menentukan mana saja yang Parent dan Child Component, bisa dilihat contoh komponen template dibawah ini:

app.component merupakan Parent dari **app-add-new-item**. Karena **app-add-new-item** dirender didalam template app.component.

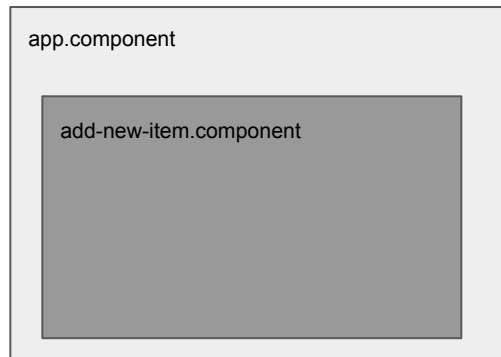
```
app.component.html M x
src > app > app.component.html > ...
1 <ul>
2   <li *ngFor="let item of items">{{item}}</li>
3 </ul>
4 <app-add-new-item (newItemEvent)="addItemInParent($event)"></app-add-new-item >
```

app.component.html (Parent)

```
add-new-item.component.html U x
src > app > add-new-item > add-new-item.component.html
1 <label for="item-input">Add an item:</label>
```

add-new-item.component.html (Child)

Nama selector: app-add-new-item



Hierarki Komponen:

Parent: app.component

Child: app-add-new-item

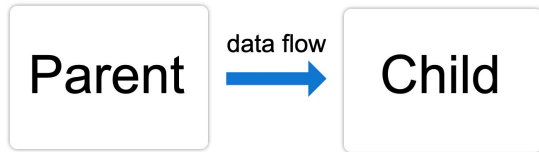
Sharing data Komponen Parent-Child

Cara sharing data antar komponen parent dan child ini menggunakan decorate **@Input** dan **@Output**.

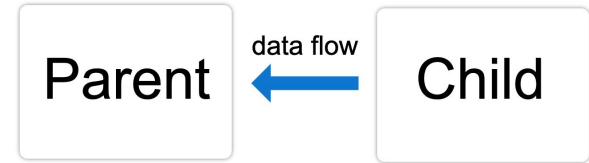
@Input: Memperbolehkan parent component untuk update/mengirim data ke child component

@Output: Memperbolehkan child component untuk mengirimkan data ke parent component

@Input



@Output



Sharing data dari Parent ke Child (@Input)

Mengirimkan data dari Parent ke Child dengan decorator **@Input**

1. Parent Komponen mengirimkan data **parentMessage** ke child component, yang dirender di parent template, dengan property binding di dalam template. (Dalam contoh ini, bernama di-binding bernama **message**).

Note: **app-komponen1** adalah nama selector dari child komponen.

```
app.component.ts M x
src > app > app.component.ts > AppComponent > pa
1  import { Component } from '@angular/core';
2
3  > @Component({ ...
7  })
8  export class AppComponent {
9      parentMessage = "Halo ini dari Parent"
10
11
```

App.component (Parent)

```
app.component.html M x
src > app > app.component.html > ...
1  <app-komponen1 [message]="parentMessage"></app-komponen1>
```

Sharing data dari Parent ke Child (@Input)

2. @Input decorator harus disediakan di class child component, agar data binding (**message**) dari parent dapat diterima oleh Child Component. Nama data binding dan nama variable dalam decorator **@Input** harus sama.
3. Sesudah data diterima oleh child component, maka data **message** dapat dirender ke template child

```
app.component.html M x
src > app > app.component.html > ...
1 | <app-komponen1 [message]="parentMessage"></app-komponen1>
2
```

App.component.html (Parent)

```
1 import { Component, Input } from '@angular/core';
2
3 > @Component({ ...
4   })
5 export class Komponen1Component {
6   @Input() message = '';
7 }
```

komponen1.component (Child)

```
komponen1.component.html U x
src > app > komponen1 > komponen1.component.html > ...
1 <p>komponen1 works!: {{message}}</p>
```

Sharing data Komponen dari Child ke Parent (@Output)

1. Buat binding event click di child template, (contoh method: **addNewItem()**). Dimana didalamnya akan meng-emit event ke Parent.
2. Daftarkan nama_event_emitter di **@Output**.
 - **EventEmitter** merupakan salah satu class, untuk dapat meng-emit event ke Parent.
 - Untuk meng-emit value ke parent, dapat menggunakan `.emit()`:

`this.nama_event_emitter.emit(value_yang_ingin_dikirim_ke_parent)`

add-new-item.component.html U X

src > app > add-new-item > add-new-item.component.html > ...

```
1 <label for="item-input">Add an item:</label>
2 <input type="text" id="item-input" #newItem>
3 <button (click)="addNewItem(newItem.value)">Add to parent's list</button>
```

1

add-new-item.component.ts U X

src > app > add-new-item > add-new-item.component.ts > ...

```
1 import { Component, EventEmitter, Output } from '@angular/core';
2
3 @Component({ ...
4 })
5
6 export class AddNewItemComponent {
7   @Output() newItemEvent = new EventEmitter<string>();
8
9   addNewItem(value: string) {
10     this.newItemEvent.emit(value);
11   }
12 }
13
14 }
```

2

add-new-item.component (Child)



HACKTIV8

Sharing data Komponen dari Child ke Parent (@Output)

3. Bind nama_event_emitter di parent template. (nama event emitter disini **newItemEvent**).

4. Arahkan bind event_emitter ke method dalam parent component. **\$event** harus dipass, agar value dari child dapat diterima kedalam parent.

```
add-new-item.component.ts U x
src > app > add-new-item > add-new-item.component.ts > ...
1 import { Component, EventEmitter, Output } from '@angular/core';
2
3 > @Component({...
7 })
8 export class AddNewItemComponent {
9   @Output() newItemEvent = new EventEmitter<string>();
10
11   addNewItem(value: string) {
12     this.newItemEvent.emit(value);
13   }
14 }
```

add-new-item.component.html (Child)

```
app.component.html M x
src > app > app.component.html > ...
1 <ul>
2   <li *ngFor="let item of items">{{item}}</li>
3 </ul>
4 <app-add-new-item (newItemEvent)="addItemInParent($event)"></app-add-new-item >
```

app.component (parent)

```
app.component.ts M x
src > app > app.component.ts > ...
1 import { Component } from '@angular/core';
2
3 > @Component({...
7 })
8 export class AppComponent {
9   items = ['sepatu', 'sendal', 'tas']
10
11   addItemInParent(newItem: string) {
12     this.items.push(newItem);
13   }
14 }
```

