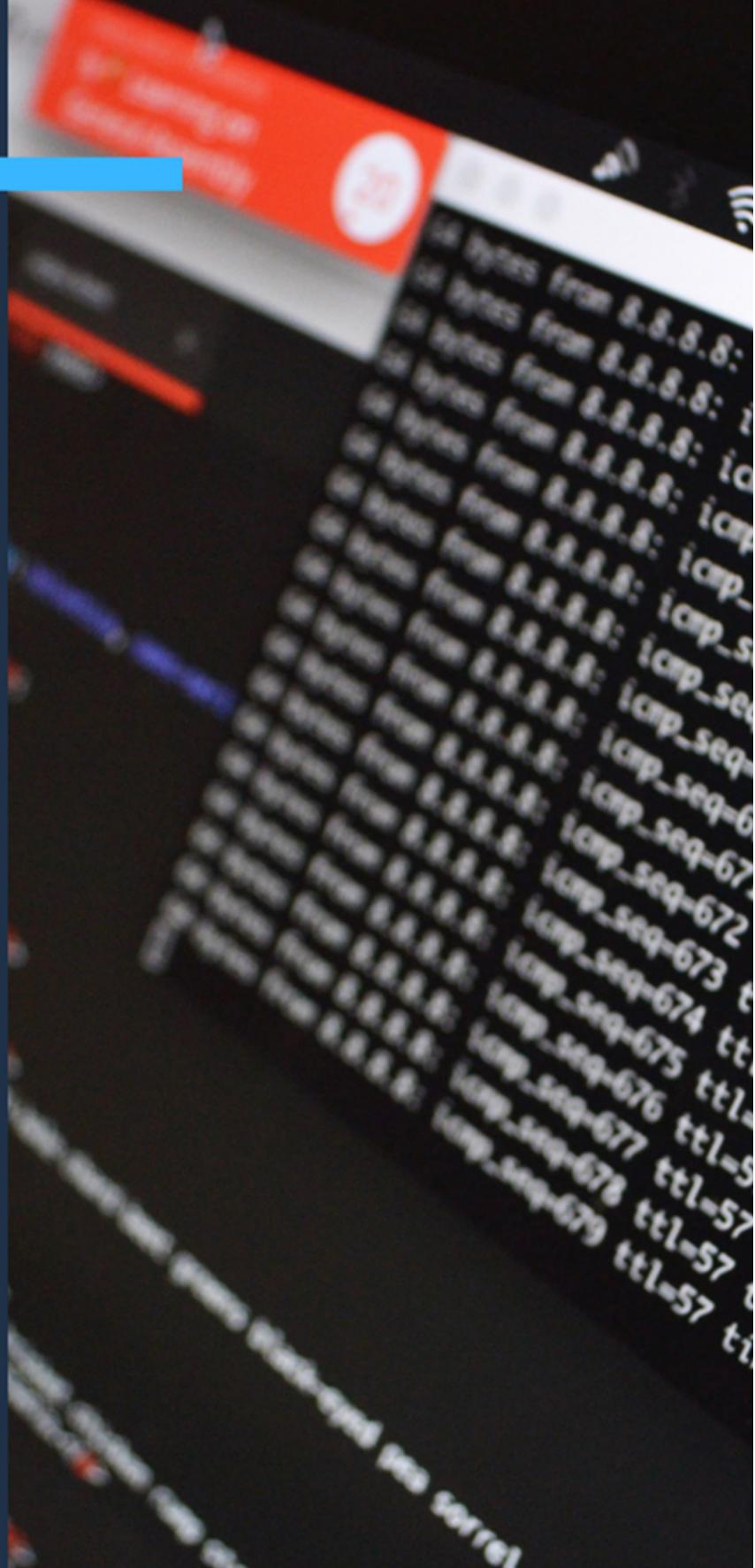


DAY 1

Web Development

Basic Angular



Cascading Training Cyber Security Programme

	I
CHAPTER 1	1
PENGENALAN	1
KEMAMPUAN AKHIR YANG DIHARAPKAN	1
PENGENALAN ANGULAR	1
ANGULAR VS HTML	1
KONSEP PEMBUATAN FRONT END DENGAN ANGULAR	2
KONSEP DASAR ANGULAR	3
MODULES	3
COMPONENT	3
SERVICES	4
DIRECTIVES	4
ROUTING	4
LATIHAN	5
TUGAS	5
CHAPTER 2	6
INSTALASI	6
KEMAMPUAN AKHIR YANG DIHARAPKAN	6
INSTALL NODEJS	6
INSTALL NODEJS WINDOWS	6
INSTALL NODEJS LINUX	6
INSTALL NODEJS MACOS	6
INSTALL ANGULAR CLI	6
BUAT PROJECT ANGULAR DENGAN ANGULAR CLI	6
CHAPTER 3	8
STRUKTUR FILE PADA PROJECT ANGULAR	8
LANGKAH PERCOBAAN	8
CHAPTER 4	12
COMPONENT BASIC	12
LANGKAH PERCOBAAN	12
CHAPTER 5	16
ANGULAR MODULES	16
CHAPTER 6	18
ALUR STARTING APLIKASI ANGULAR	18
CHAPTER 7	20

INTERPOLASI	20
LANGKAH PRAKTIKUM	20
CHAPTER 8	23
PROPERTY BINDING	23
LANGKAH PERCOBAAN	23
CHAPTER 9	27
EVENT BINDING	27
LANGKAH PRAKTIKUM	27
CHAPTER 10	35
TEMPLATE REFERENCE	35
LANGKAH PERCOBAAN	35
CHAPTER 11	38
PERCABANGAN (NGIF)	38
LANGKAH PERCOBAAN	38
CHALLENGE TIME	40
NGIF , ELSE , THEN, NGIFELSE	40
LANGKAH PERCOBAAN	40
CHALLENGE TIME	44
CHAPTER 12	46
PERULANGAN	46
LANGKAH PERCOBAAN	46
CHAPTER 13	50
STYLE BINDING	50
NGSTYLE	50
NGCLASS	50
LANGKAH PERCOBAAN	50
CHAPTER 14	55
PIPES	55
LANGKAH PERCOBAAN	55

CHAPTER 1

Pengenalan

Kemampuan Akhir Yang Diharapkan

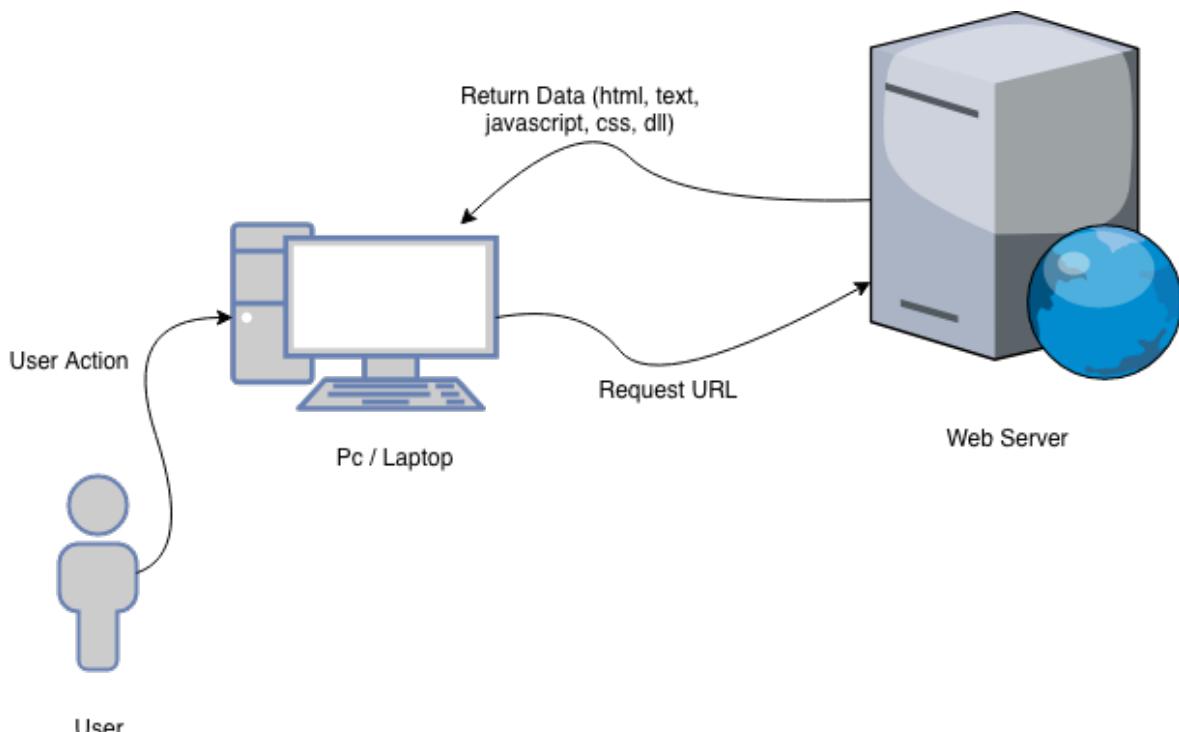
1. Peserta memahami perbedaan antara cara pembuatan frontend dengan angular dan html biasa.
2. Peserta memahami konsep pembuatan frontend dengan angular.
3. Peserta mampu memahami konsep "component" dalam pembuatan frontend.

Pengenalan Angular

Angular adalah salah satu framework front end yang sedang populer saat ini, angular dikembangkan dengan bahasa pemrograman typescript.

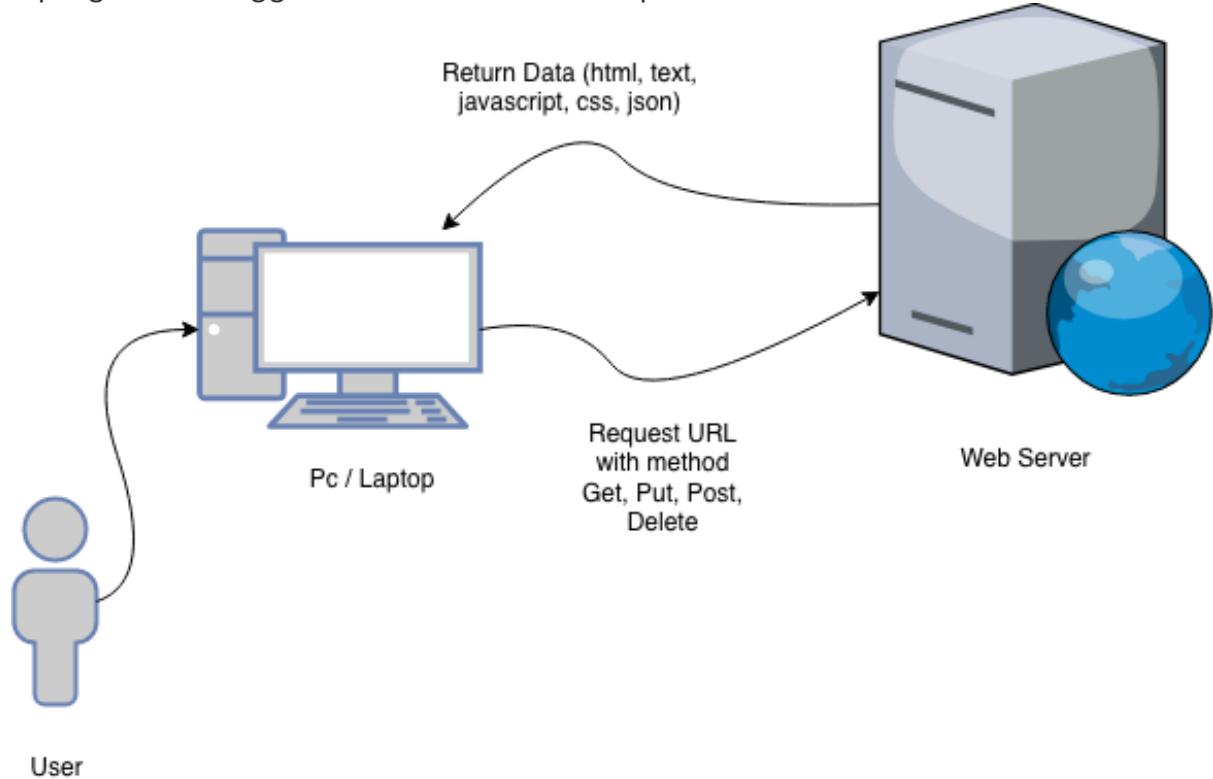
Angular vs Html

Teknik pembuatan frontend untuk saat ini sudah berkembang dengan sangat pesat, pada beberapa tahun yang lalu pembuatan front end masih menggunakan view yang di render oleh server. Teknik ini menggunakan bahasa pemrograman html untuk tampilan front end dan bahasa pemrograman server side untuk mengatur tampilan dan data.



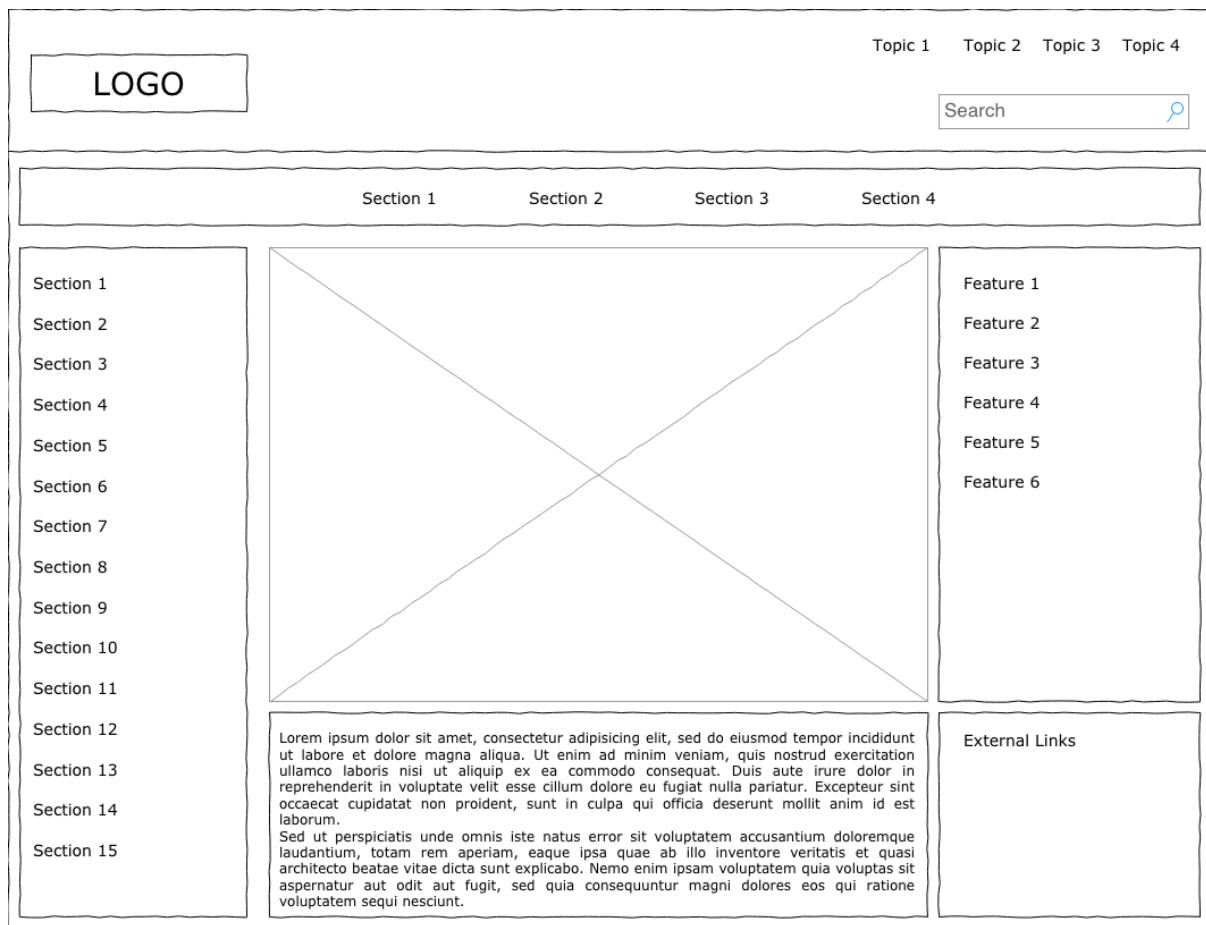
Kemudian pada dua sampai tiga tahun terakhir pembuatan aplikasi web sudah mengarah kepada pembuatan sistem yang terbagi menjadi dua bagian yaitu back

end dan front end. Angular berfokus di pengembangan front end sehingga front end yang pada beberapa tahun yang lalu hanya menampilkan data dari server sekarang sudah berubah tidak sekedar menampilkan data saja namun dapat diprogram sehingga lebih interaktif dan responsif.



Konsep Pembuatan Front end Dengan Angular

Konsep pembuatan front end dengan angular sangat berbeda dengan pembuatan front end dengan html atau bahasa pemrograman server side biasa. Pada pembuatan front end dengan angular konsep yang sangat penting untuk dipahami adalah “Segala sesuatu yang dapat dilihat oleh user dibagi sebagai sebuah komponen”. Komponen ini dapat digunakan kembali dan dapat dikombinasikan dengan komponen lain sehingga dapat menjadi sebuah halaman yang dapat digunakan oleh user.



Konsep Dasar Angular

Angular dibuat berdasarkan konsep component, namun tidak hanya komponen dalam sebuah project angular kita harus memahami konsep konsep dasar yang lain seperti Modules, Component, Services, Directives dan Routing. Kelima konsep inilah yang menjadi dasar pengembangan Angular.

Modules

Setiap project angular memiliki minimal satu modules. Modules ini lah yang menjadi wadah aplikasi yang menampung component, route, service, directive dan lain lain. Project angular baru minimal memiliki satu module, modul defaultnya adalah AppModule.

Sebuah Project dapat memiliki lebih dari satu module, contohnya ketika project membutuhkan routing maka dapat ditambahkan routing module.

Component

Component adalah bagian dari angular yang berisi template, data dan logic.

Kode program component terdiri dari @Component annotation yang berisi selector, template, dan stylesheets untuk tampilan component ini. Setelah itu pada

kode program component juga berisi class typescript yang berisi logika untuk component tersebut.

Data pada component bisa didapat dari service dan dari component ini sendiri. Logic pada component dapat diisi pada kode typescript nya.

Sebuah Component dapat menjadi anak dari component lain atau menjadi parent dari child component nya. Sebuah component juga dapat berinteraksi ke child atau parent nya melalui input dan event.

Services

Service pada aplikasi angular merupakan sebuah class yang bertugas melakukan sesuatu yang spesifik untuk membantu jalannya aplikasi, biasanya service digunakan oleh component melalui dependency injections.

Pada aplikasi angular component dan service dipisahkan agar component berfokus pada user sedangkan service berfokus pada "urusan dapur" dari component".

Contohnya jika terdapat sebuah component "search" maka proses menampilkan hasil searching itu dilakukan oleh component sedangkan proses query ke database atau ke REST API dilakukan oleh service.

Perlu dipahami karena service berdiri sendiri maka sebuah service dapat digunakan berulang kali pada component yang berbeda jadi service tidak terikat pada satu component saja.

Directives

Pada dasarnya directives adalah sebuah fungsi yang akan dieksekusi oleh angular jika fungsi tersebut ditemukan di DOM. Fungsi fungsi ini digunakan untuk menambah kemampuan halaman html dengan memberikan sintaks baru yang berasal dari directives. Sintaks baru ini dapat mengubah sifat html yang awalnya statis menjadi lebih dinamis. Directives ini sudah ada dan disiapkan oleh angular berupa default directives dan pengembang aplikasi dapat menambahkan directives baru.

Routing

Routing pada angular berfungsi untuk menampilkan suatu spesific component berdasarkan url yang dibuka, routing merupakan sebuah module angular tersendiri yang bernama RoutingModule.

Module routing ini memberikan kemudahan dalam melakukan navigasi user.

Latihan

Tugas

1. Cari lah sebuah website, dan bagilah website tersebut menjadi component component kecil.
2. Dari komponen komponen tersebut susunlah mana yang menjadi parent component dan child component.
3. Desain lah sebuah component tree yang lengkap sehingga website yang anda buka tersebut dapat terbagi menjadi module, component, service, directive dan routing.

CHAPTER 2

Instalasi

Kemampuan Akhir Yang Diharapkan

1. Peserta dapat melakukan installasi angular.
2. Peserta dapat menjalankan server developement angular.

Install NodeJs

Versi node js yang dibutuhkan untuk dapat menjalankan aplikasi angular minimal adalah versi 8.x atau 10.x.

Install nodejs Windows

Berikut ini langkah installasi nodejs untuk sistem operasi windows :

Install nodejs Linux

Berikut ini langkah installasi nodejs untuk sistem operasi linux :

Install nodejs MacOs

Berikut ini langkah installasi nodejs untuk sistem operasi macOs :

Install Angular CLI

Angular CLI digunakan untuk membuat project baru dan kode program angular, oleh karena itu angular CLI harus di install secara global melalui node package manager anda, untuk melakukan installasi ketikkan perintah berikut ini pada command line :

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

Lamanya proses installasi tergantung pada kecepatan koneksi internet anda silahkan tunggu sampai semua proses selesai.

Perlu diperhatikan jangan melakukan perintah installasi angular CLI saat terminal / command line anda login sebagai root atau administrator.

Buat Project Angular dengan Angular CLI

Untuk membuat project baru menggunakan angular cli dapat dilakukan dengan mengetik perintah berikut ini pada command line / terminal.

```
ng new my-app  
ng serve -open
```



CHAPTER 3

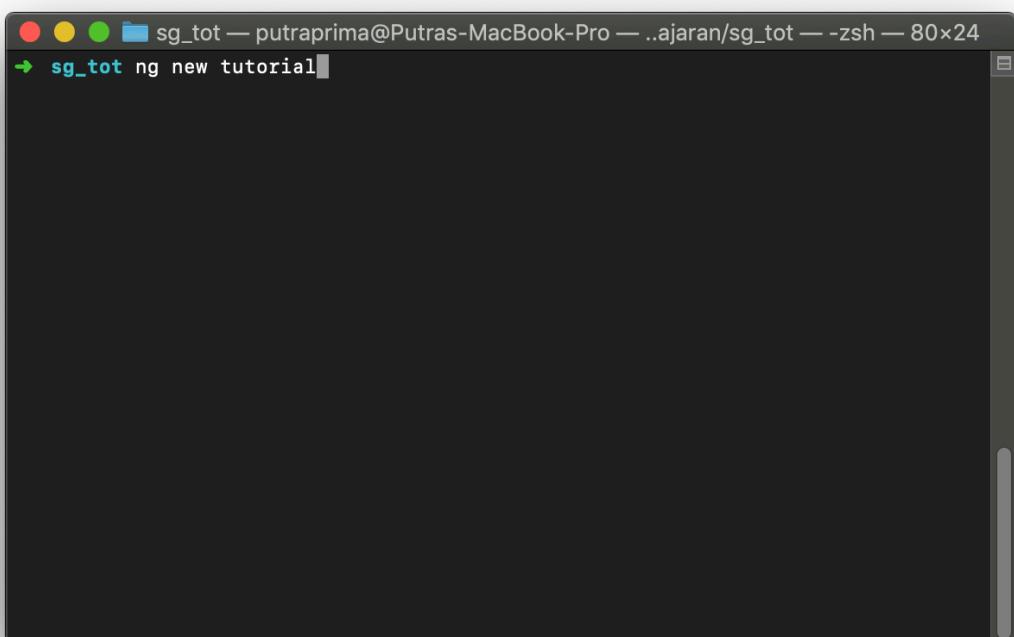
Struktur File Pada Project Angular

Untuk memulai membuat project angular dapat dilakukan dengan membuat sebuah project baru menggunakan angular cli, untuk membuat project dengan angular cli ikutilah langkah langkah percobaan dibawah ini

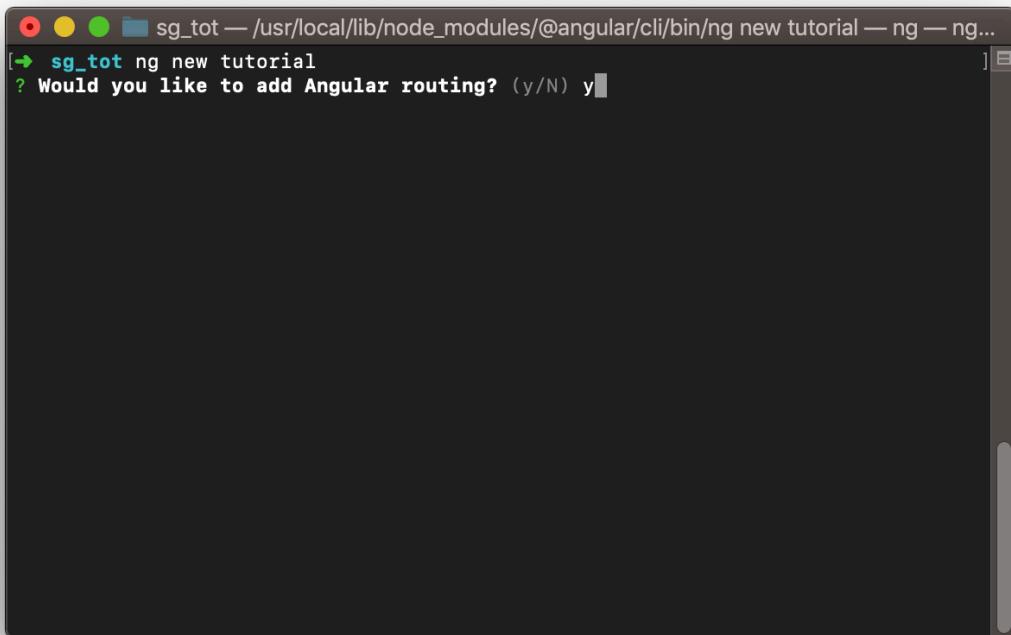
Langkah percobaan

Ketikkan perintah berikut ini pada terminal anda

```
ng new tutorial
```

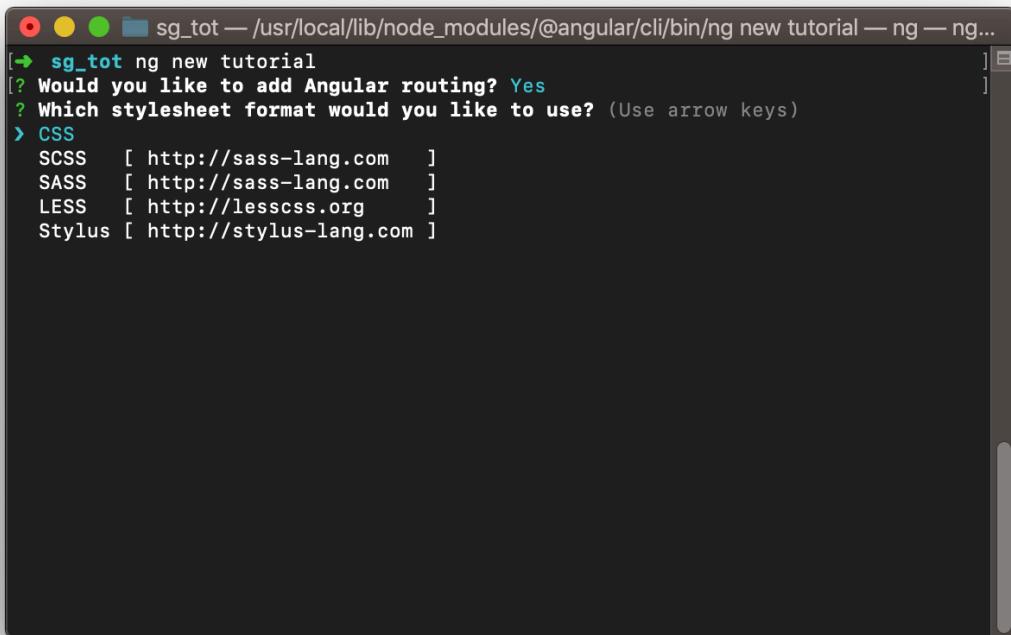


Setelah ng new tutorial dijalankan angular cli akan meminta anda memilih untuk menambahkan routing module atau tidak, untuk saat ini pilih Y untuk menambahkan routing module



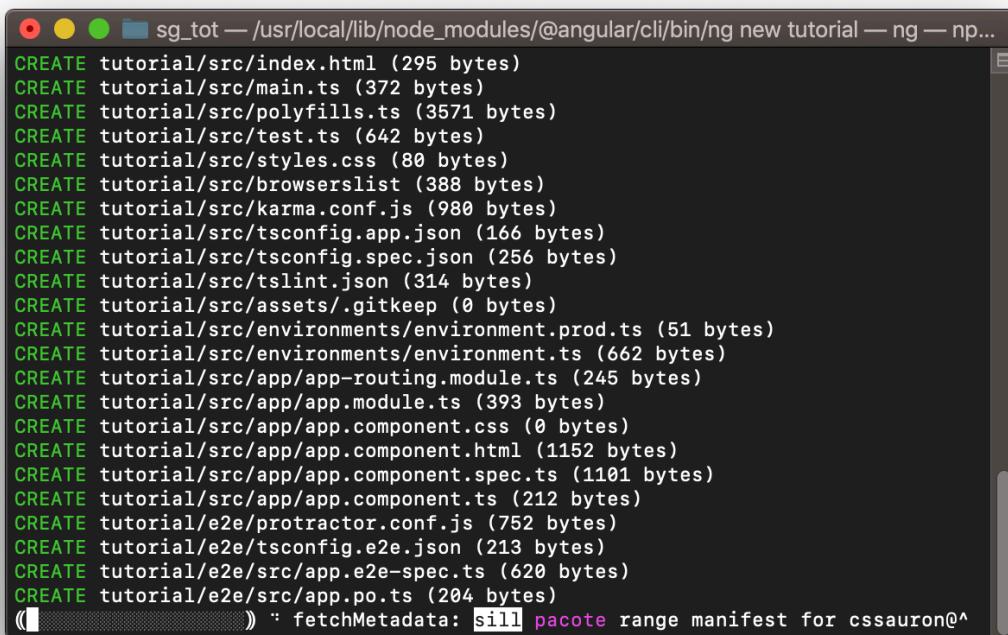
```
[→ sg_tot — /usr/local/lib/node_modules/@angular/cli/bin/ng new tutorial — ng — ng...
[?] Would you like to add Angular routing? (y/N) y
```

Selanjutnya angular cli akan meminta anda untuk memilih format stylesheet yang akan digunakan pilihlah css sebagai stylesheetnya.



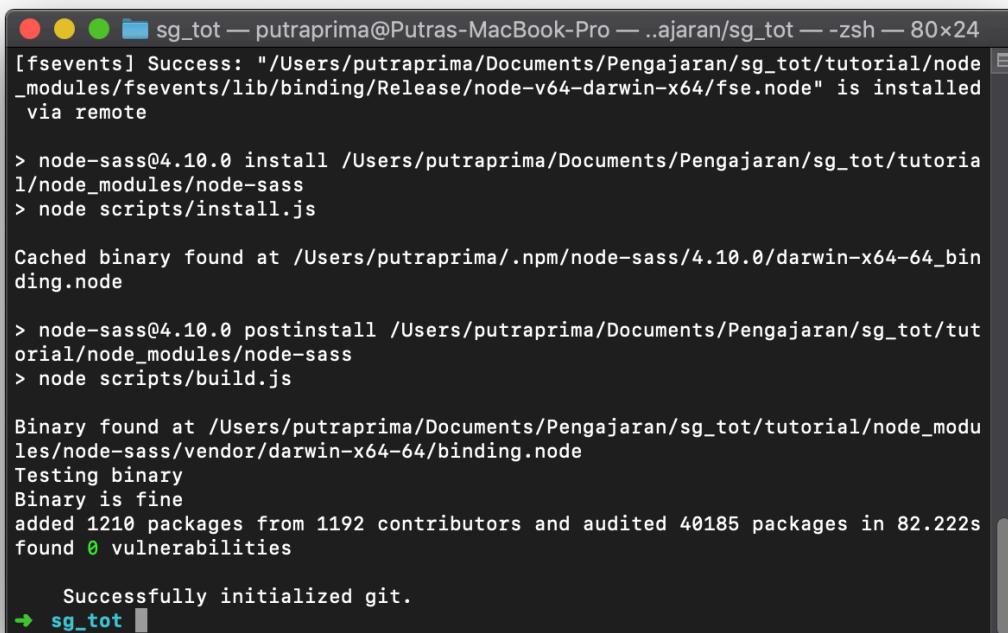
```
[→ sg_tot — /usr/local/lib/node_modules/@angular/cli/bin/ng new tutorial — ng — ng...
[?] Would you like to add Angular routing? Yes
[?] Which stylesheet format would you like to use? (Use arrow keys)
> CSS
  SCSS [ http://sass-lang.com ]
  SASS [ http://sass-lang.com ]
  LESS [ http://lesscss.org ]
  Stylus [ http://stylus-lang.com ]
```

Proses selanjutnya angular cli akan mendownload dan menginstall dependency yang dibutuhkan. Tunggu lah proses ini sampai selesai



```
CREATE tutorial/src/index.html (295 bytes)
CREATE tutorial/src/main.ts (372 bytes)
CREATE tutorial/src/polyfills.ts (3571 bytes)
CREATE tutorial/src/test.ts (642 bytes)
CREATE tutorial/src/styles.css (80 bytes)
CREATE tutorial/src/browserslist (388 bytes)
CREATE tutorial/src/karma.conf.js (980 bytes)
CREATE tutorial/src/tsconfig.app.json (166 bytes)
CREATE tutorial/src/tsconfig.spec.json (256 bytes)
CREATE tutorial/src/tslint.json (314 bytes)
CREATE tutorial/src/assets/.gitkeep (0 bytes)
CREATE tutorial/src/environments/environment.prod.ts (51 bytes)
CREATE tutorial/src/environments/environment.ts (662 bytes)
CREATE tutorial/src/app/app-routing.module.ts (245 bytes)
CREATE tutorial/src/app/app.module.ts (393 bytes)
CREATE tutorial/src/app/app.component.css (0 bytes)
CREATE tutorial/src/app/app.component.html (1152 bytes)
CREATE tutorial/src/app/app.component.spec.ts (1101 bytes)
CREATE tutorial/src/app/app.component.ts (212 bytes)
CREATE tutorial/e2e/protractor.conf.js (752 bytes)
CREATE tutorial/e2e/tsconfig.e2e.json (213 bytes)
CREATE tutorial/e2e/src/app.e2e-spec.ts (620 bytes)
CREATE tutorial/e2e/src/app.po.ts (204 bytes)
) " fetchMetadata: sill pacote range manifest for cssauron@^
```

Berikut ini tampilan terminal ketika proses install sudah selesai



```
[fsevents] Success: "/Users/putraprima/Documents/Pengajaran/sg_tot/tutorial/node_modules/fsevents/lib/binding/Release/node-v64-darwin-x64/fse.node" is installed via remote

> node-sass@4.10.0 install /Users/putraprima/Documents/Pengajaran/sg_tot/tutorial/node_modules/node-sass
> node scripts/install.js

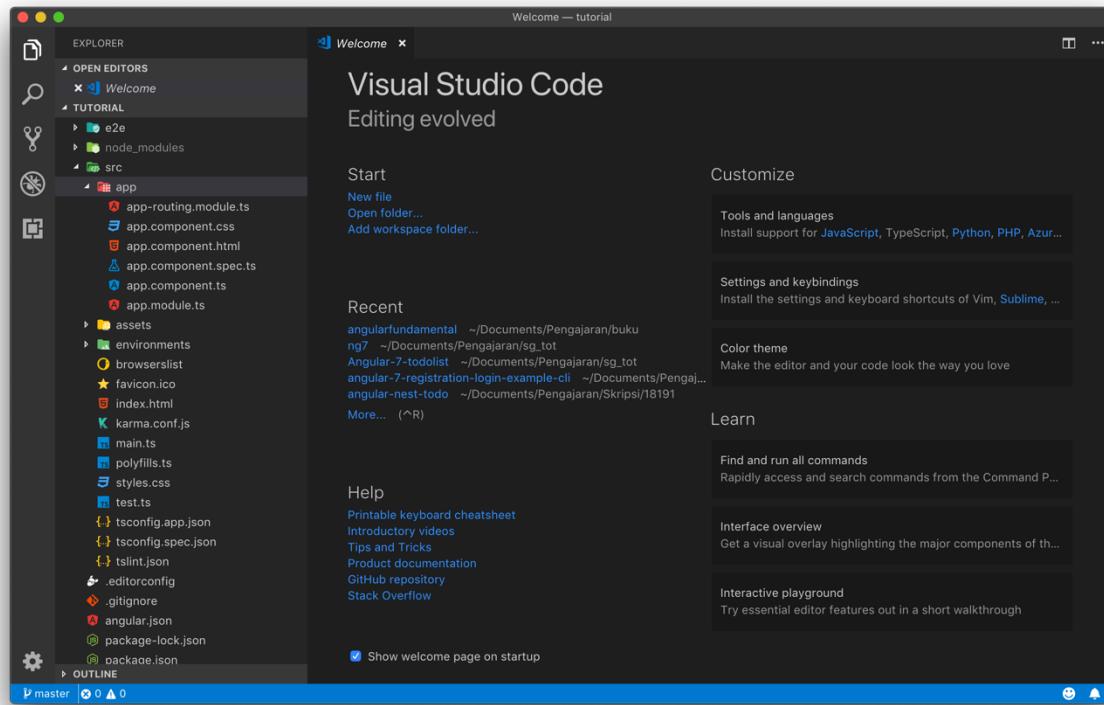
Cached binary found at /Users/putraprima/.npm/node-sass/4.10.0/darwin-x64-64_binding.node

> node-sass@4.10.0 postinstall /Users/putraprima/Documents/Pengajaran/sg_tot/tutorial/node_modules/node-sass
> node scripts/build.js

Binary found at /Users/putraprima/Documents/Pengajaran/sg_tot/tutorial/node_modules/node-sass/vendor/darwin-x64-64/binding.node
Testing binary
Binary is fine
added 1210 packages from 1192 contributors and audited 40185 packages in 82.222s
found 0 vulnerabilities

  Successfully initialized git.
→ sg_tot
```

Untuk langkah selanjutnya silahkan buka folder tutorial yang sudah dibuat dengan menggunakan visual studio code atau editor yang anda kuasai.



Di editor ini anda dapat melihat struktur file yang dimiliki oleh angular v.7 pada dasarnya struktur file ini masih sama dengan yang dimiliki oleh angular v6.

Didalam folder src terdapat folder app yang berisi module dan component, nanti pada pembelajaran selanjutnya kita akan mengubah dan menambah component serta module di dalam folder ini. Untuk tahap awal pembelajaran angular kita akan lebih sering bekerja di folder app.

CHAPTER 4

Component Basic

Component adalah bagian penting dari sebuah aplikasi angular, sebuah aplikasi yang dikembangkan dengan menggunakan angular terdiri dari beberapa component yang bekerja sama.

Langkah Percobaan

Untuk memulai membuat component bukalah file app.component.ts pada folder app. Perhatikan kode program yang ada :

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

Kode program diatas adalah kode program minimal untuk membuat sebuah komponen, minimal terdiri dari sebuah class yang memiliki decorator @Component.

Untuk melatih proses pembuatan component hapuslah semua kode program pada app.component.ts

Langkah pertama tambahkan statement import

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

statement import ini berfungsi untuk menginclude Class Component dari library angular, kedepannya kita juga akan membutuhkan class lain namun untuk sebuah componen sederhana cukup mengimport class Component saja.

selanjutnya setelah melakukan import tambahkan class yang ingin dijadikan component

```
export class AppComponent {  
  title = "tutorial";  
}
```

pada kode program di atas kita melakukan export terhadap sebuah class dengan nama AppComponent, rule dan konvensi untuk penamaan class adalah nama class dibuat CamelCase dan nama file menyesuaikan nama class jika nama class AppComponent maka nama file app.component.ts.

pada class ini dilakukan export agar component ini dapat di import di component lain, jika tidak ada statement export maka componen / module lain.

Selanjutnya untuk dapat dijadikan sebuah component class ini harus ditambahkan decorator @Component, dimana dekorator ini membutuhkan 3 parameter, parameternya adalah selector, templateUrl atau template, dan styleUrls.

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

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

Pada intinya sebuah component harus memiliki class, template html, style html nya dan selector yang unik untuk memanggil componen tersebut.

Template html nya dapat berada dalam satu file dengan file component atau berada di file lain.

Untuk langkah selanjutnya jika sudah membuat component gantilah variabel title pada AppComponent nilainya menjadi nama anda masing masing, contoh :

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

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

bukalah terminal pada visual studi code anda, arahkan lokasi terminal pada root project kemudian ketikkan

```
ng serve --open
```

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

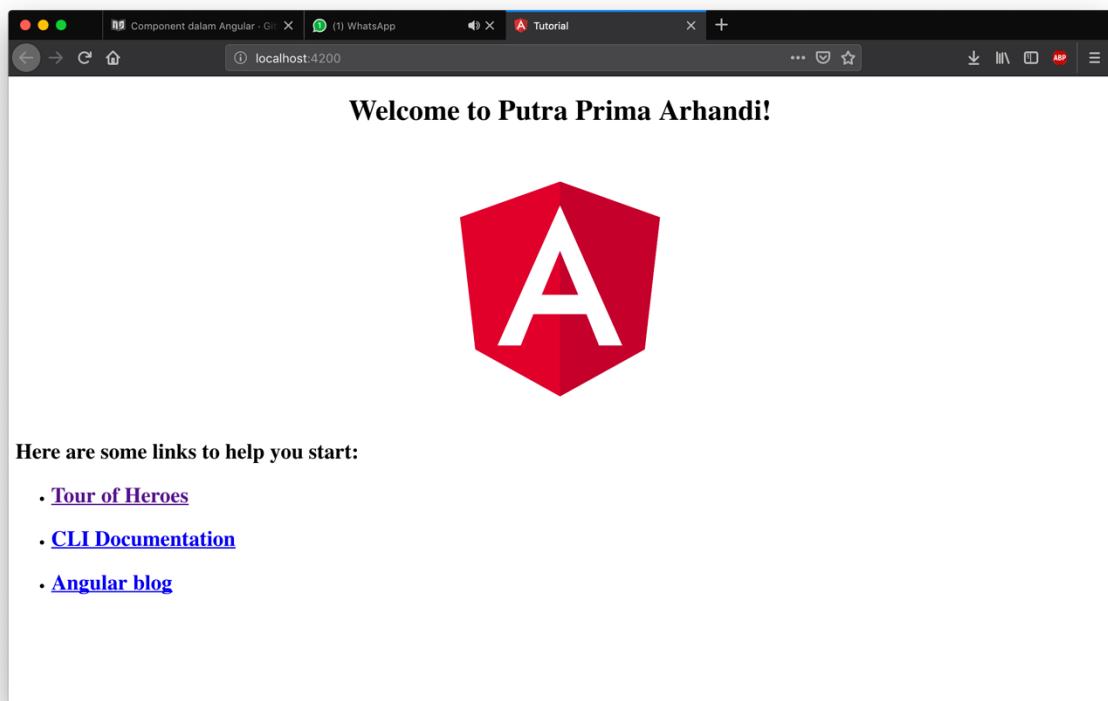
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

tutorial git:(master) ✘ ng serve --open

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

tutorial git:(master) ✘ ng serve --open



CHAPTER 5

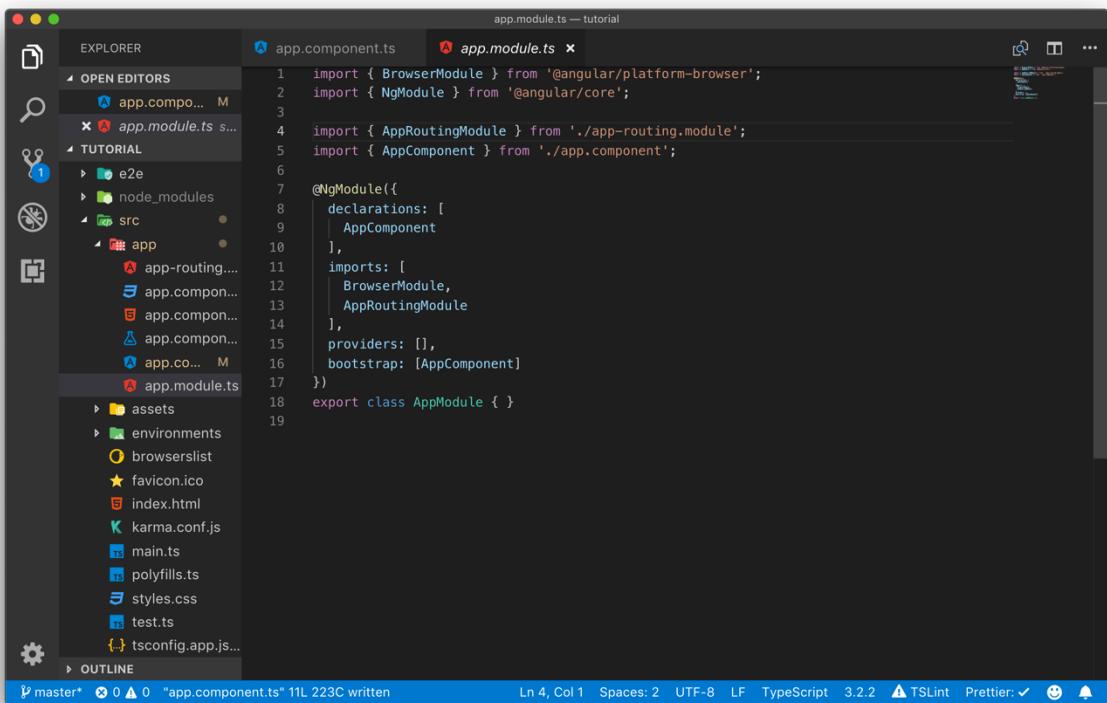
Angular Modules

Angular memiliki sistem untuk membagi aplikasi menjadi bagian-bagian kecil yang disebut module. Decorator yang digunakan untuk membuat sebuah module adalah `NgModule`. Module pada Angular merupakan sebuah wadah yang menampung sekumpulan kode program baik component, service atau kode program lain yang bertugas menyelesaikan satu fitur atau satu domain aplikasi.

Setiap aplikasi Angular minimal memiliki satu module yang diberi nama `AppModule`, pada `AppModule` ini lah dilakukan bootstrapping untuk memulai aplikasi Angular.

Sebuah aplikasi boleh memiliki lebih dari satu module penambahan module ini tergantung pada desain dan cara pengembangan aplikasi, dengan syarat semua module baru harus diincludekan pada `AppModule` sebagai "child" dari `AppModule`.

Berikut ini contoh kode program `AppModule` :



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

Langkah Percobaan

Perhatikan pada file `AppModule.ts` tetap ditambahkan beberapa statement import, yaitu `BrowserModule`, `NgModule`, `AppRoutingModule` dan `AppComponent`.

```
import { BrowserModule } from "@angular/platform-browser";
```

statement diatas digunakan untuk mengimport BrowserModule, module ini diperlukan untuk pengembangan aplikasi angular yang berbasis web.

```
import { NgModule } from "@angular/core";
```

Statement import diatas digunakan untuk mengimport Decorator NgModule, ini diperlukan untuk membuat sebuah class sebagai Module.

```
import { AppRoutingModule } from "./app-routing.module";
```

Statement ini digunakan untuk mengimport Routing Module ini digenerate otomatis oleh angular cli ketika kita memilih opsi Y ketika membuat project di chapter awal.

```
import { AppComponent } from "./app.component";
```

Statement ini digunakan untuk megimport component AppComponent, jika nanti kita menambahkan component baru maka kita harus menambahkan statement import terhadap component tersebut agar dapat digunakan oleh angular.

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

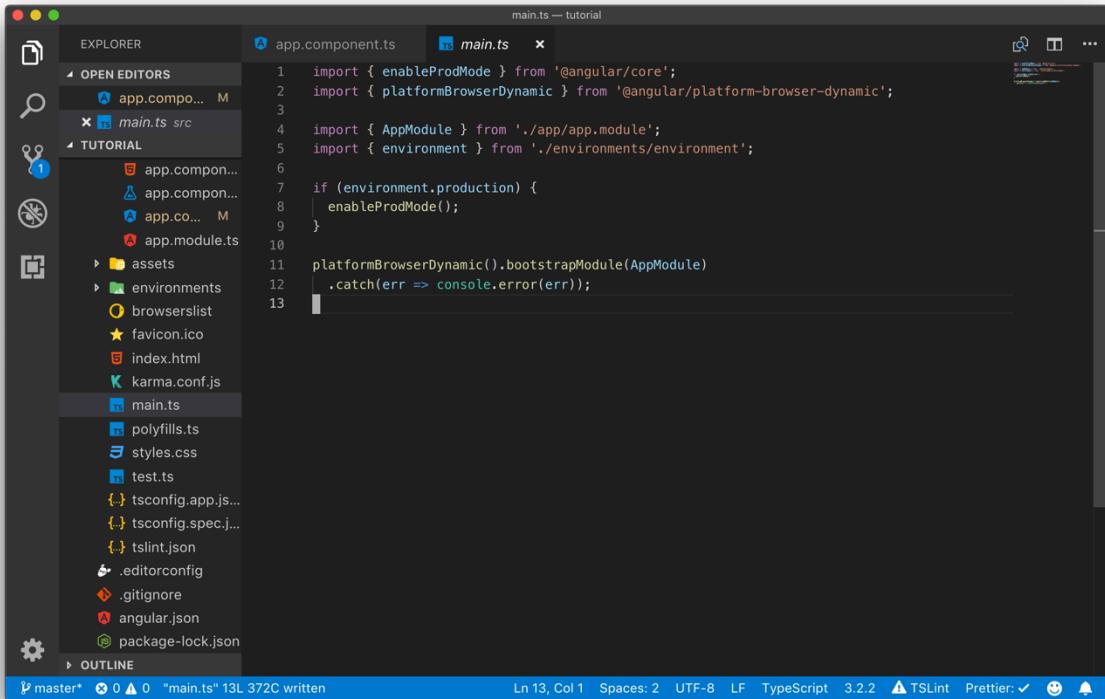
Statement diatas adalah Decorator @NgModule yang didapat dari import NgModule dari @angular/core. Decorator ini memiliki beberapa meta data yaitu declarations, exports, imports, providers dan bootstraps.

1. Metadata declarations berisi semua component, directives ,dan pipes yang dimiliki oleh module bersangkutan. Setiap kali kita menambahkan component directive ataupun pipes maka semuanya harus di import dan di tambahkan ke declarations.
2. Metadata exports berisi nama module yang dapat digunakan oleh ngModule lain.
3. Metadata imports berisi module lain yang dibutuhkan oleh component yang ada pada module ini.
4. Metadata providers biasanya berisi service yang dibutuhkan oleh module bersangkutan.
5. Metadata bootstrap metadata khusus yang hanya dimiliki oleh root module.

CHAPTER 6

Alur Starting Aplikasi Angular

Aplikasi angular dimulai dari file main.ts, pada file inilah dilakukan bootstrapping angular. Untuk mempermudah pemahaman terhadap alur ini bukalah file main.ts pada project anda.



```
main.ts — tutorial
EXPLORER app.component.ts main.ts
OPEN EDITORS app.compo... M main.ts src
TUTORIAL app.compon... app.co... M app.module.ts
assets environments browserslist favicon.ico index.html karma.conf.js
main.ts polyfills.ts styles.css test.ts
tsconfig.app.json tsconfig.spec.j... tslint.json
.editorconfig .gitignore angular.json package-lock.json
OUTLINE
Ln 13, Col 1 Spaces: 2 UTF-8 LF TypeScript 3.2.2 TSLint Prettier: ✓ 😊 🚙
master* 0 0 0 "main.ts" 13L 372C written
```

The screenshot shows the Visual Studio Code interface with the main.ts file open in the editor. The file contains the following code:

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

Pada kode program di atas dapat dilihat pada file main.ts hanya terdapat statement import dan fungsi untuk memulai bootstrapping.

```
import { enableProdMode } from "@angular/core";
import { platformBrowserDynamic } from "@angular/platform-browser-dynamic";

import { AppModule } from "./app/app.module";
import { environment } from "./environments/environment";
```

Pada statement import diatas dimulai dengan mengimport enableProdMode, enableProdMode ini adalah library untuk mengaktifkan production mode artinya aplikasi angular yang anda jalankan sudah pada tahap production dimana pada tahap ini proses checking yang biasanya digunakan pada development mode sudah dimatikan.

Selanjutnya yang di import adalah platformBrowserDynamic, library ini digunakan untuk melakukan kompilasi kode program angular di browser. Perhatikan bahwa di main.ts yang di import adalah platformBrowserDynamic bukan platformBrowser.

Selanjutnya yang di import adalah module utama atau AppModule, khusus pada file ini hanya di load root module nya saja jadi pada file ini tidak boleh ada module lain. Jika anda membagi aplikasi angular ke beberapa module maka module lain ini harus menjadi anak /child dari root module (AppModule)

```
if (environment.production) {  
  enableProdMode();  
}  
  
platformBrowserDynamic()  
  .bootstrapModule(AppModule)  
  .catch(err => console.error(err));
```

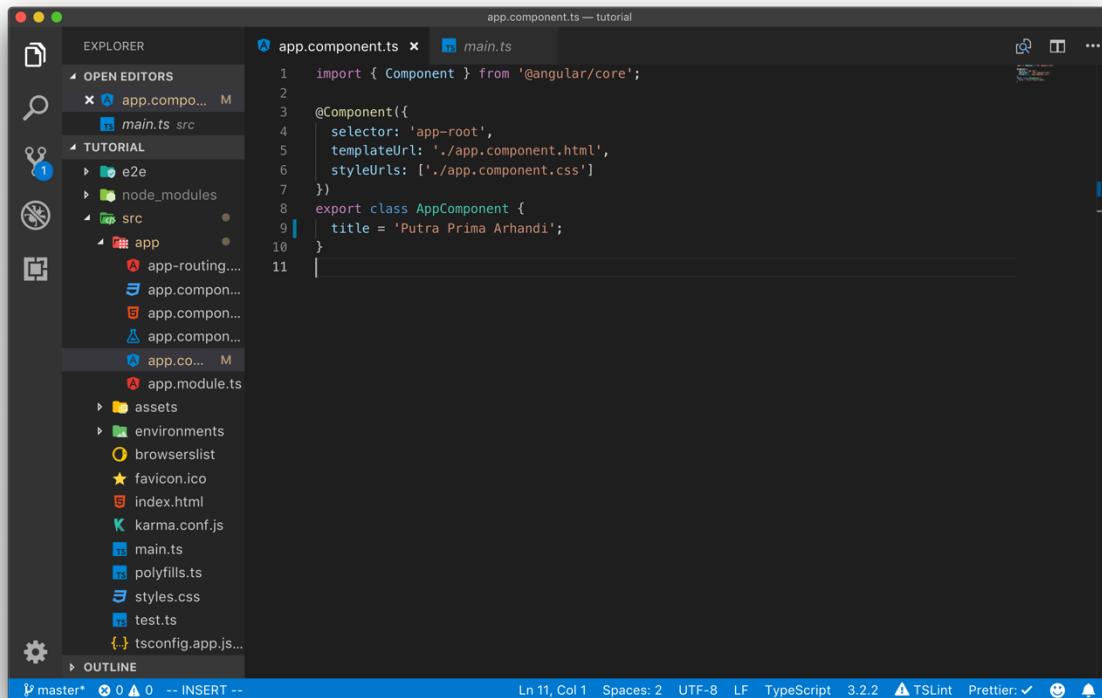
Selanjutnya dilakukan checking environment dan yang paling utama adalah dipanggil method platformBrowserDynamic yang melakukan bootstrap module AppModule. Dari AppModule bootstrapping dilanjutkan ke AppComponent.

CHAPTER 7

Interpolasi

Angular menghubungkan antara data di class component dengan template html melalui sebuah proses interpolasi. Proses interpolasi ini dilakukan dengan menambahkan syntaks `{{ . }}` pada template html.

Contoh sederhana interpolasi yang sudah dilakukan pada percobaan sebelumnya adalah menghubungkan antara variabel `title` pada class dengan `{{title}}` pada template html.



```
app.component.ts — tutorial
EXPLORER
OPEN EDITORS
app.component.ts M
main.ts src
TUTORIAL
e2e
node_modules
src
app
app-routing...
app.compon...
app.compon...
app.compon...
app.co... M
app.module.ts
assets
environments
browserslist
favicon.ico
index.html
karma.conf.js
main.ts
polyfills.ts
styles.css
test.ts
tsconfig.app.js...
OUTLINE
Ln 11, Col 1  Spaces: 2  UTF-8  LF  TypeScript  3.2.2  TSLint  Prettier: ✓  ☺  🔔
```

Semua variabel yang telah di hubungkan (binding) antara class dengan templatenya dapat diperlakukan seperti sebuah variabel javascript contohnya dapat dilakukan penjumlahan atau proses lain yang dapat dilakukan pada variabel javascript.

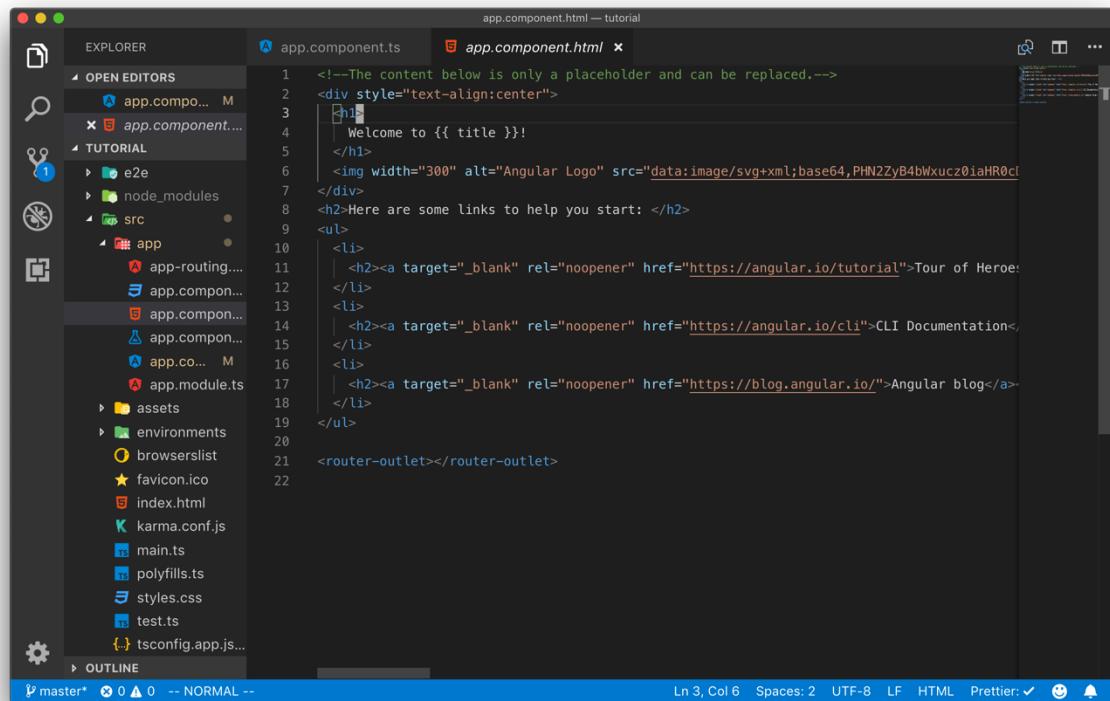
Langkah Praktikum

Pada file AppComponent.ts buatlah beberapa variabel baru dengan memperhatikan tipe datanya karena pada angular digunakan typescript.

```
export class AppComponent {
  title = "Putra Prima Arhandi";
  panjang: Number = 10;
  lebar: Number = 10;
  isTeaBreakTime: Boolean = true;
}
```

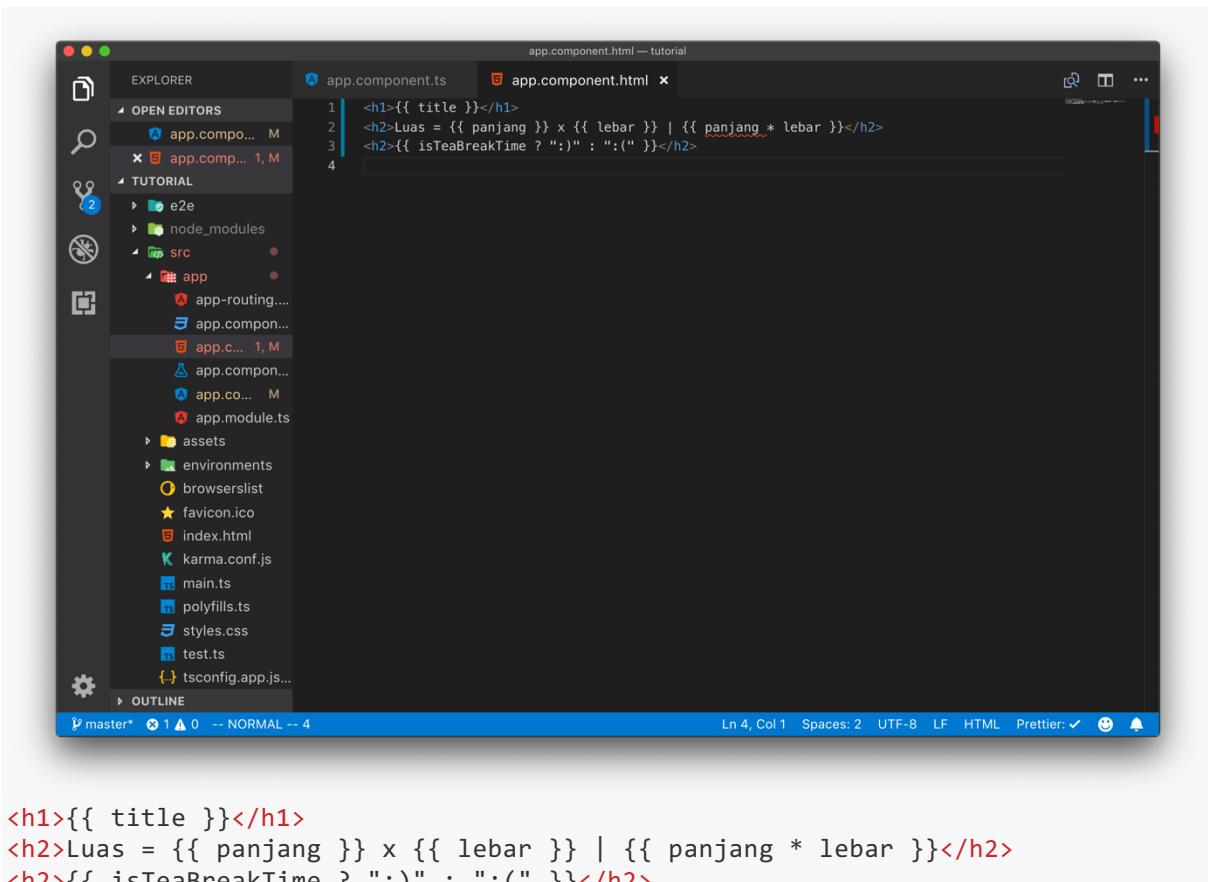
Pada kode program diatas terdapat 4 variabel title, panjang, lebar, dan isTeaBreakTime. pada variabel title dibuat tanpa tipe data dan di isi data berformat string, untuk variabel panjang dan lebar di beri tipedata Number dan pada variabel isTeaBreakTime diberi tipe data Boolean.

Selanjutnya silahkan buka file app.component.html kemudian hapuslah semua isinya berikut ini tampilan awal file app.component.html



```
<!--The content below is only a placeholder and can be replaced.-->
<div style="text-align:center">
  <h1>
    Welcome to {{ title }}!
  </h1>
  Tour of Heroes</a></h2>
    </li>
    <li>
      | <h2><a target="_blank" rel="noopener" href="https://angular.io/cli">CLI Documentation</a></h2>
    </li>
    <li>
      | <h2><a target="_blank" rel="noopener" href="https://blog.angular.io/">Angular blog</a></h2>
    </li>
  </ul>
</div>
<router-outlet></router-outlet>
```

Hapuslah kode program diatas dan tambahkan kode program berikut ini :



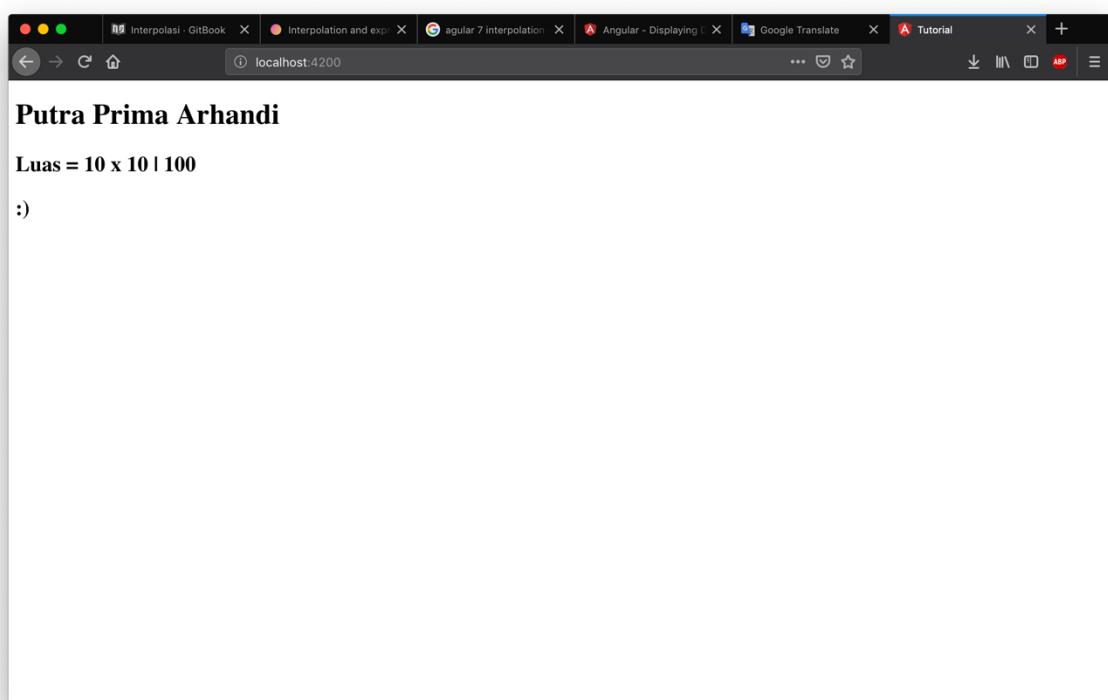
The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: Shows the project structure with files like app.component.ts, app.component.html, app-routing.module.ts, app.module.ts, assets, environments, browserslist, favicon.ico, index.html, karma.conf.js, main.ts, polyfills.ts, styles.css, and test.ts.
- OPEN EDITORS**:
 - app.component.ts
 - app.component.html — tutorial (active tab)
- app.component.html — tutorial** content:

```
1 <h1>{{ title }}</h1>
2 <h2>Luas = {{ panjang }} x {{ lebar }} | {{ panjang * lebar }}</h2>
3 <h2>{{ isTeaBreakTime ? ":" : "(" }}</h2>
4
```
- STATUS BAR**: master*, 1 ▲ 0 -- NORMAL -- 4, Ln 4, Col 1, Spaces: 2, UTF-8, LF, HTML, Prettier: ✓, smiley face, bell icon.

```
<h1>{{ title }}</h1>
<h2>Luas = {{ panjang }} x {{ lebar }} | {{ panjang * lebar }}</h2>
<h2>{{ isTeaBreakTime ? ":" : "(" }}</h2>
```

Berikut ini hasil jika anda menjalankan dengan benar :



CHAPTER 8

Property Binding

Property binding adalah cara angular untuk menghubungkan antara tag attribute pada template html dan variabel class component. Untuk mempermudah pemahaman terhadap property binding ada beberapa konsep yang harus dipahami yaitu :

1. Semua tag html memiliki attribut.
2. Semua attribut pada tag html tersebut dapat di akses melalui Document Object Model (Dom).
3. Semua Dom memiliki properties yang sesuai dengan attribut pada tag html tersebut.

Untuk yang sudah berpengalaman dengan akses DOM menggunakan jquery atau javascript mungkin dapat lebih mudah memahami konsep ini

Contoh :

```
<!-- kode program html biasa: -->
<input type="text" value="hello world" />
<!-- kode program property binding pada atribut html: -->
<input type="text" [value]="namaVariabelDiComponent" />
<!-- kode program property binding pada interpolasi pada html: -->
<input type="text" value="{{namaVariabelDiComponent}}" />
```

Dengan potongan kode program di atas kita melakukan property binding ke attribut value dari input text. Dengan cara ini kita dapat menghubungkan nilai variabel dengan nama "namaVariabelDiComponent" ke isi dari input text.

Selain dengan menggunakan [] (square bracket) property binding juga bisa dilakukan dengan menggunakan interpolasi seperti pada kode program di atas, namun cara yang umum dan sering digunakan adalah cara pada kode program baris ke dua.

Property binding dengan cara ini merupakan model "one way data binding" dimana data mengalir satu arah dari component ke template, artinya data hanya akan berubah jika component mengubah nilai dari data tersebut template html hanya menjadi presenter atau alat untuk menampilkan data.

Langkah Percobaan

Dari percobaan sebelumnya hapuslah isi dari file app.component.html, kemudian berilah sebuah header dengan judul "Property Binding"

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** view: Shows the project structure under the `TUTORIAL` folder, including `e2e`, `node_modules`, `src` (with `app`), `assets`, `environments`, `browserslist`, `favicon.ico`, `index.html`, and `karma.conf.js`.
- OPEN EDITORS** view: Shows two files: `app.component.ts` and `app.component.html`.
- app.component.ts** content:

```
1 <h1>Property Binding</h1>
```
- app.component.html** content:

```
1 <h1>Property Binding</h1>
2 |
```
- Bottom Status Bar**: Shows the file name "app.component.html — tutorial", line count "Ln 2, Col 1", spaces setting "Spaces: 2", encoding "UTF-8", line separator "LF", file type "HTML", and Prettier status "Prettier: ✓".

Selanjutnya lakukanlah binding terhadap sebuah tag html h2 yang berisi nilai dari variabel title pada component dengan cara property binding dan interpolasi.

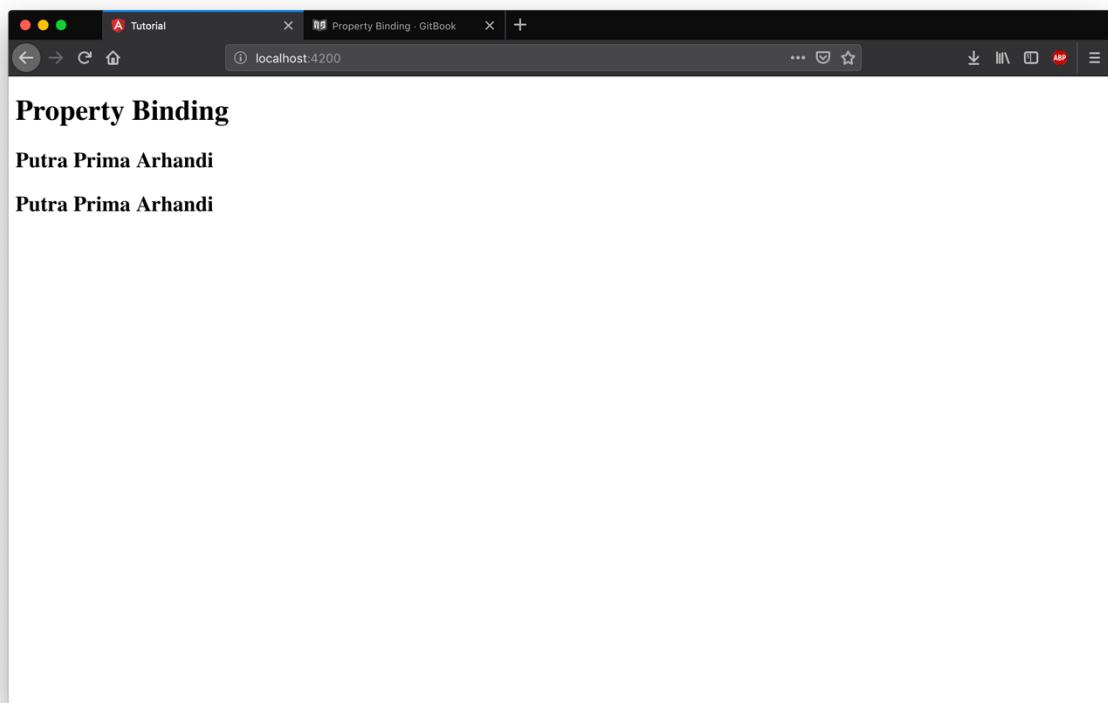
The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** view: Shows the project structure under the `TUTORIAL` folder, including `e2e`, `node_modules`, `src` (with `app`), `assets`, `environments`, `browserslist`, `favicon.ico`, `index.html`, and `karma.conf.js`.
- OPEN EDITORS** view: Shows two files: `app.component.ts` and `app.component.html`.
- app.component.ts** content:

```
1 <h1>Property Binding</h1>
```
- app.component.html** content:

```
1 <h1>Property Binding</h1>
2 |
3 <h2 [innerHTML]="title"></h2>
4 <h2>{{ title }}</h2>
5 |
```
- Bottom Status Bar**: Shows the file name "app.component.html" 5L 78C written, line count "Ln 5, Col 1", spaces setting "Spaces: 2", encoding "UTF-8", line separator "LF", file type "HTML", and Prettier status "Prettier: ✓".

Hasil dari kode program diatas



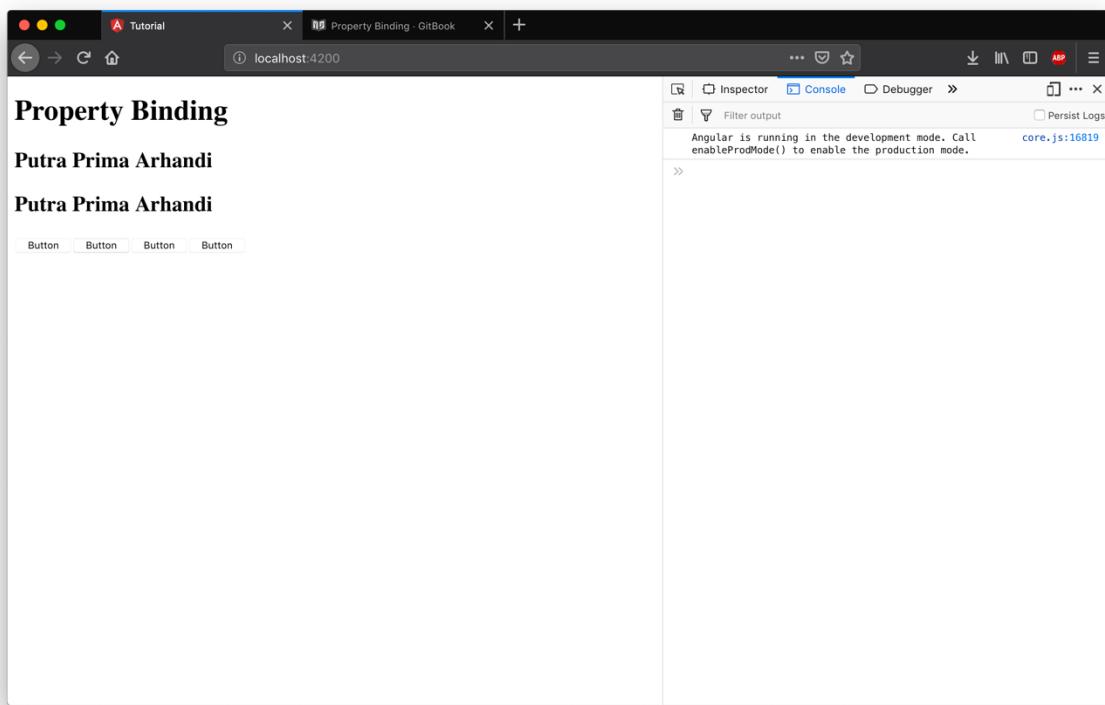
Dari hasil percobaan diatas terlihat binding dengan cara property binding dan interpolasi memberikan hasil yang sama.

Selanjutnya buatlah sebuah tombol dengan binding attribut disabled nya ke variabel `isTeaBreakTime` seperti pada kode program dibawah ini.

```
<h1>Property Binding</h1>
<h2 [innerHTML]="title"></h2>
<h2>{{ title }}</h2>

<button [disabled]="isTeaBreakTime">Button</button>
<button [disabled]!="isTeaBreakTime">Button</button>
<button disabled="{{ isTeaBreakTime }}">Button</button>
<button disabled="{{ !isTeaBreakTime }}">Button</button>
```

Jalankan server angular anda dan lihat hasilnya pada web browser.



Perhatikan bahwa hasil binding menggunakan interpolasi pada button selalu membuat button di disable.

Interpolasi dapat digunakan sebagai property binding karena interpolasi merupakan syntactic sugar untuk mempersingkat penulisan binding, namun tidak dapat digunakan untuk nilai boolean jadi sebaiknya jika tipe data yang di binding bukan string di binding menggunakan property binding saja.

CHAPTER 9

Event Binding

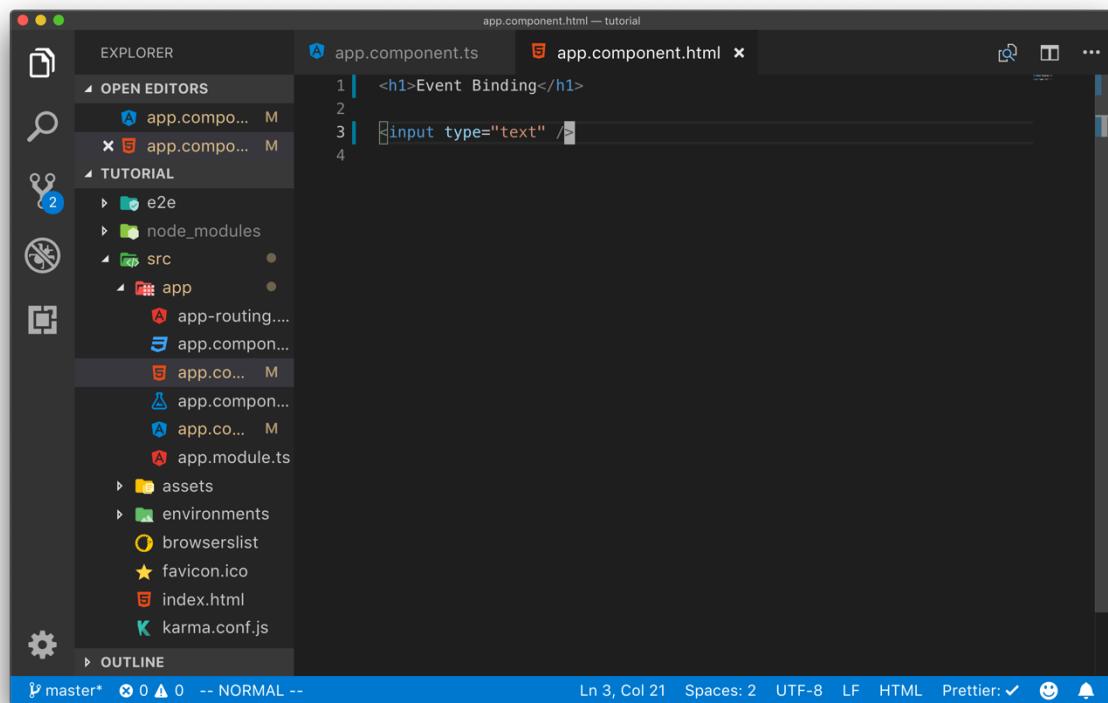
Property binding bisa digunakan untuk menampilkan data dari component ke template html, untuk menerima input / action dari user sebuah component dapat menggunakan Event Binding.

Event binding adalah binding yang khusus untuk event yang ada pada tag html yang ada pada template, contohnya jika pada template terdapat sebuah tag input maka event binding dapat dibind ke event onBlur, onInput, onChange, onFocus dan lain lain.

Untuk membuat event binding digunakan sintaks () yang di berikan sebuah fungsi pada class component.

Langkah Praktikum

Hapuslah kode program pada file app.component.html yang ada dari percobaan sebelumnya, kemudian tambahkan kode program berikut ini.

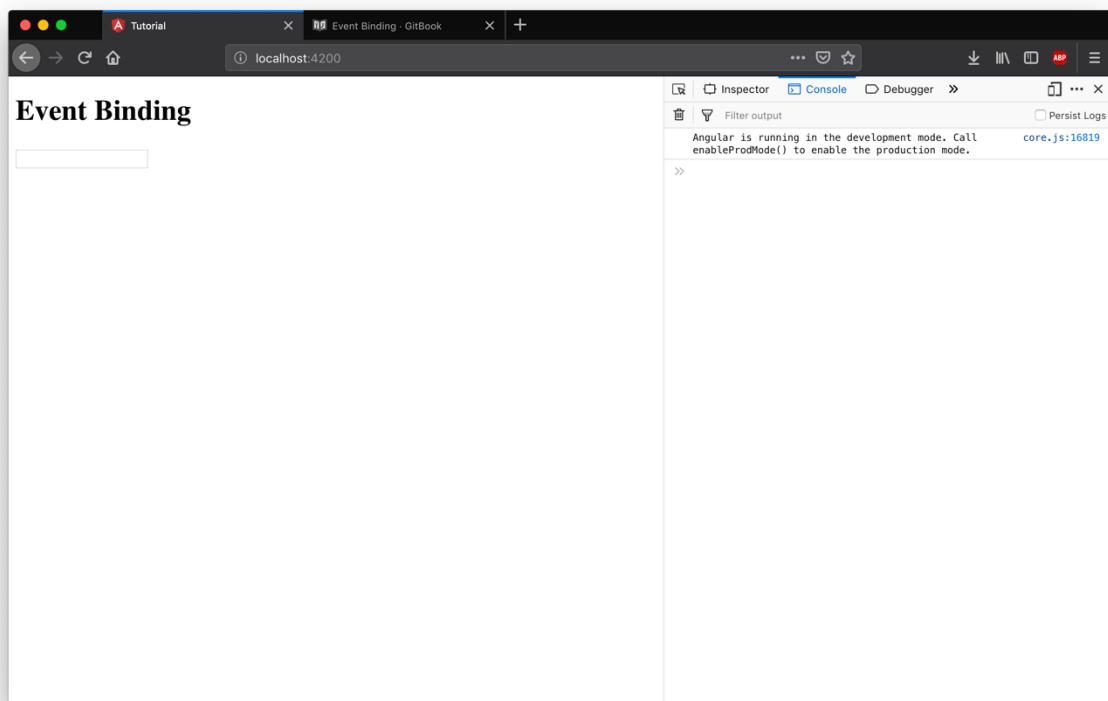


The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a project structure with folders like 'OPEN EDITORS', 'TUTORIAL', and 'src'. Inside 'src/app', there are files such as 'app-routing.module.ts', 'app.component.ts', 'app.component.html', 'app.module.ts', 'assets', 'environments', 'browserslist', 'favicon.ico', 'index.html', and 'karma.conf.js'. The 'app.component.html' tab is active in the top bar, showing the following code:

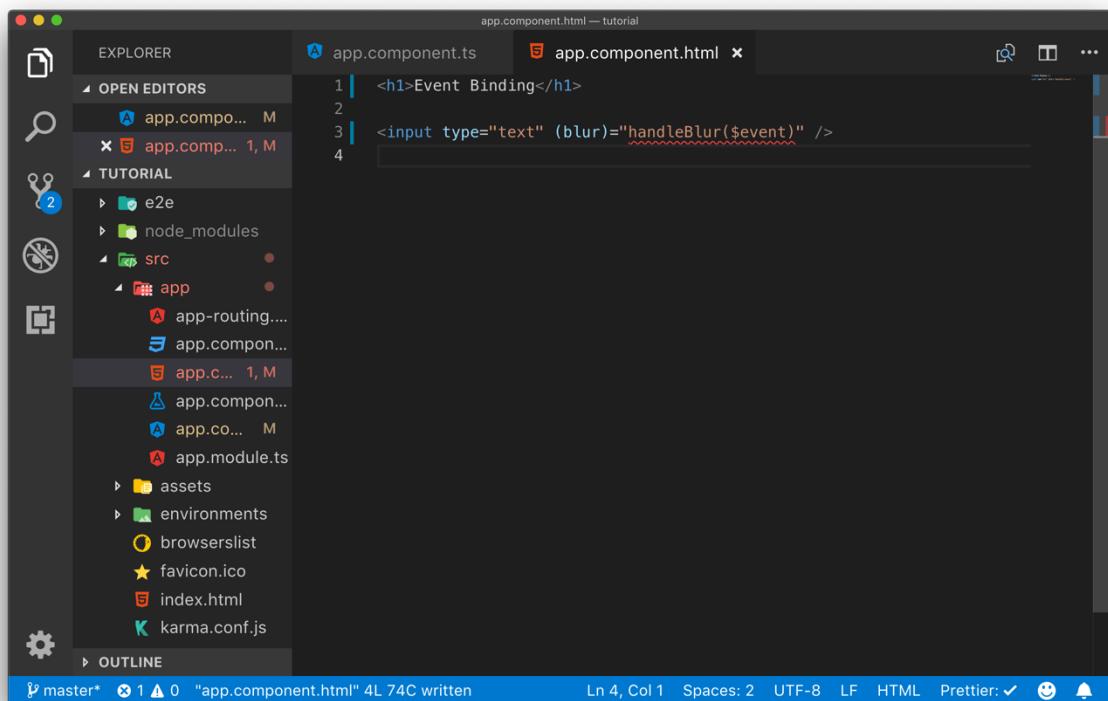
```
<h1>Event Binding</h1>
<input type="text" />
```

The bottom status bar indicates the file is in 'NORMAL' mode, with line 3, column 21, and other details like 'Spaces: 2', 'UTF-8', 'LF', 'HTML', and 'Prettier' status.

Kemudian jalankan server angular dengan perintah `ng serve --open` hasilnya akan seperti gambar dibawah ini.

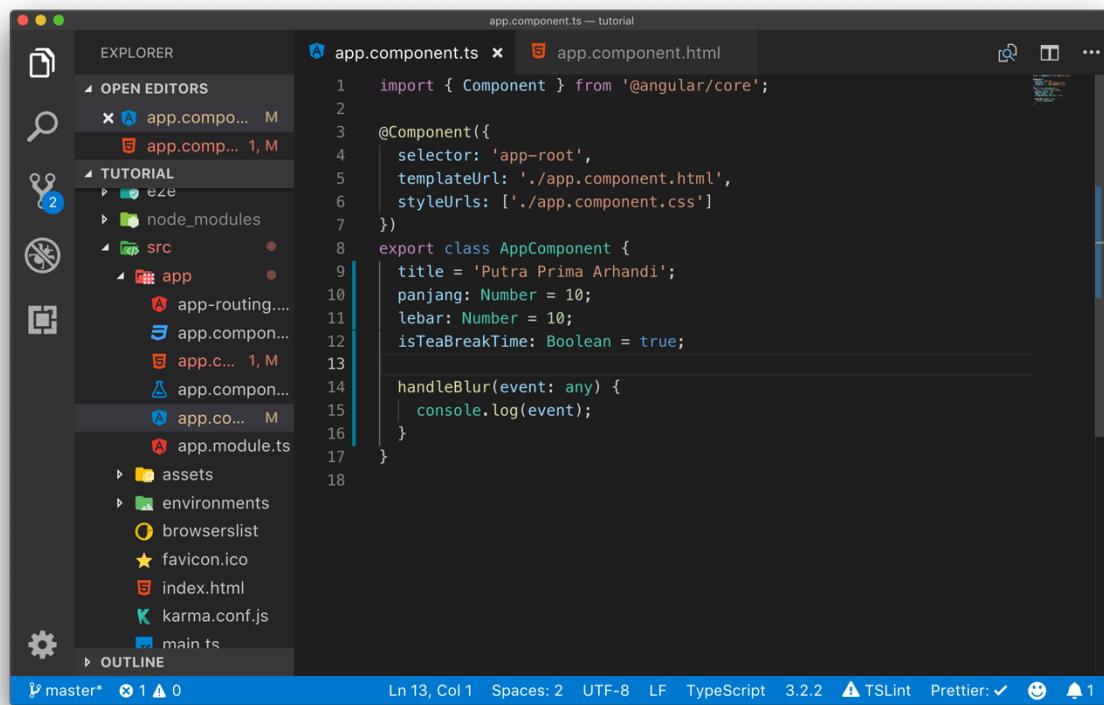


Selanjutnya tambahkan eventBinding Angular onBlur pada tag input dengan cara menambahkan `(blur)=handleBlur($event)` pada input text di app.component.html.



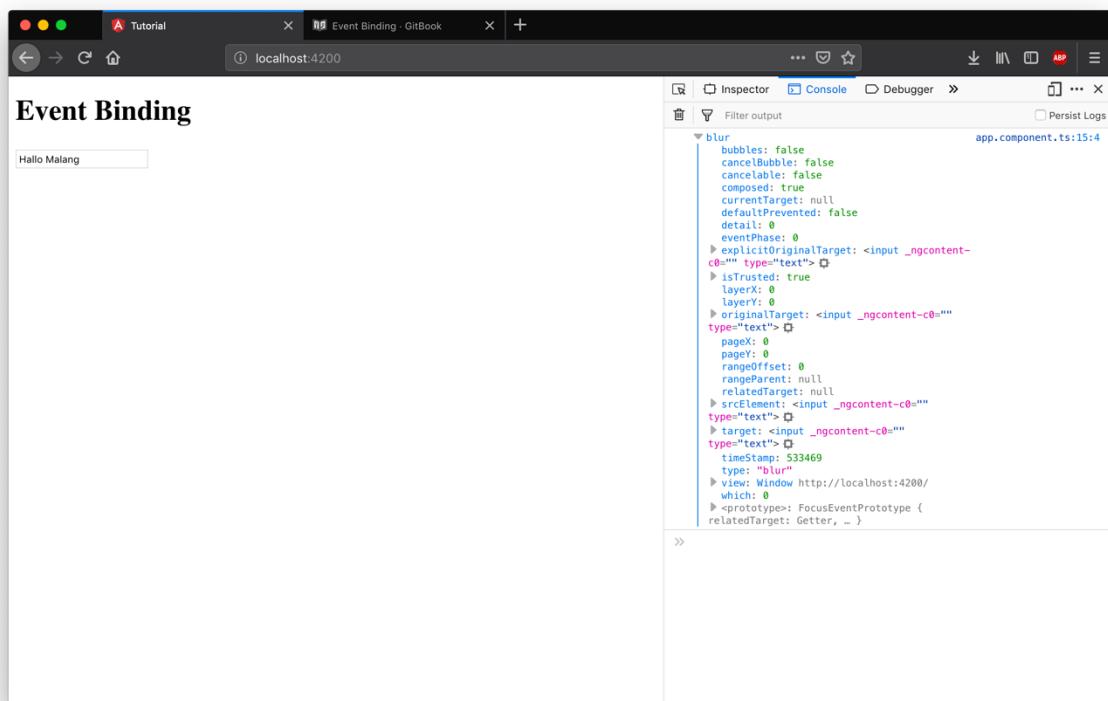
Visual studio code akan menunjukkan error pada script di app.component.html itu dikarenakan kita belum membuat function handleBlur pada component class di

app.component.ts. Selanjutnya buatlah fungsi handleBlur yang akan menghandle event tersebut pada class component.



```
app.component.ts — tutorial
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'Putra Prima Arhandi';
10  panjang: Number = 10;
11  lebar: Number = 10;
12  isteaBreakTime: Boolean = true;
13
14  handleBlur(event: any) {
15    console.log(event);
16  }
17
18 }
```

Pada kode program di atas kita membuat sebuah function untuk menampilkan log ke console web browser setiap kali event onBlur dipanggil. Event onBlur / blur akan dipanggil setelah kita mengetik dan beranjak dari element input. Untuk menguji nya isikanlah sebuah text ke input text pada web browser kemudian klik di area diluar input text. Maka browser akan menampilkan log berisi nilai event yang di trigger oleh onBlur.



Didalam object event ini terdapat banyak properties, properties yang kita inginkan adalah event.target.value dimana properties tersebut merupakan text dari input yang kita isi. Untuk melihat apakah event.target.value merupakan properties yang kita inginkan ubahlah console.log pada app.component.ts menjadi seperti dibawah ini.

```

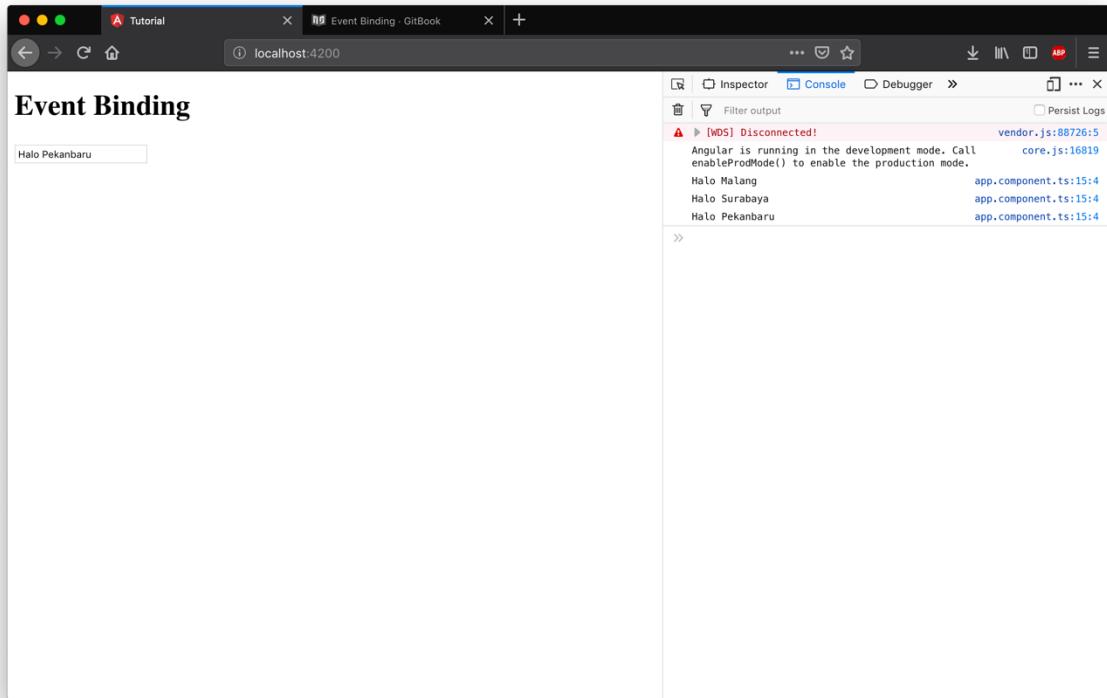
app.component.ts — tutorial
EXPLORER
OPEN EDITORS
app.compo... M
app.compo... 1, M
TUTORIAL
eze
node_modules
src
app
app-routing...
app.compon...
app.c... 1, M
app.compon...
app.co... M
app.module.ts
assets
environments
browserslist
favicon.ico
index.html
karma.conf.js
main ts
OUTLINE
Ln 15, Col 34  Spaces: 2  UTF-8  LF  TypeScript  3.2.2  TSLint  Prettier: ✓  1  1  0
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Putra Prima Arhandi';
  panjang: Number = 10;
  lebar: Number = 10;
  isTeaBreakTime: Boolean = true;

  handleBlur(event: any) {
    console.log(event.target.value);
  }
}

```

Hasil dari kode program di atas akan seperti gambar dibawah ini dimana setiap

kali kita keluar dari input text maka pada console akan di log kan sebuah value berupa text yang kita inputkan.



Tentu saja melakukan sebuah log ke console tidak banyak manfaatnya untuk user, untuk menampilkan perubahan nilai dari input text ke user anda dapat menggunakan interpolasi atau property binding terhadap sebuah tag html di app.component.html untuk menampilkan text tersebut. Untuk mencoba hal ini kita menggunakan variabel title yang sudah ada pada component, kemudian mengubah nilainya sesuai dengan input dari user setiap kali user melakukan event blur. Sesuaikanlah isi dari file app.component.ts dan app.component.html anda seperti gambar dibawah ini.

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** sidebar: Shows the project structure with files like app.component.ts, app.component.html, and app.module.ts.
- EDITOR**:
 - File: `app.component.ts`
 - Content:

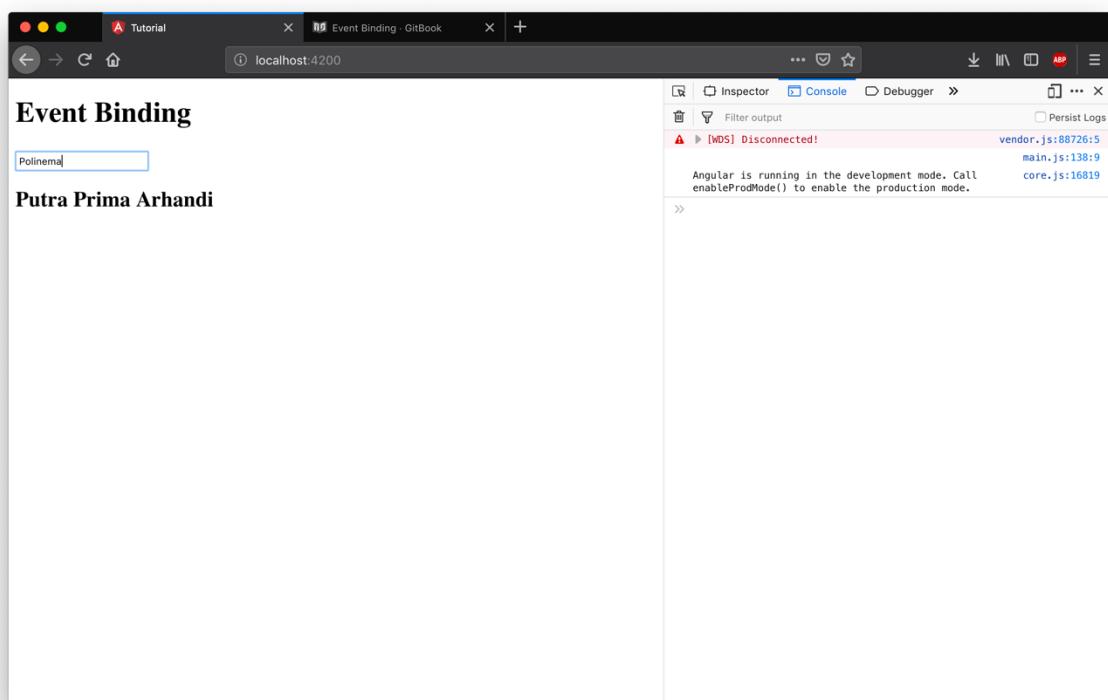
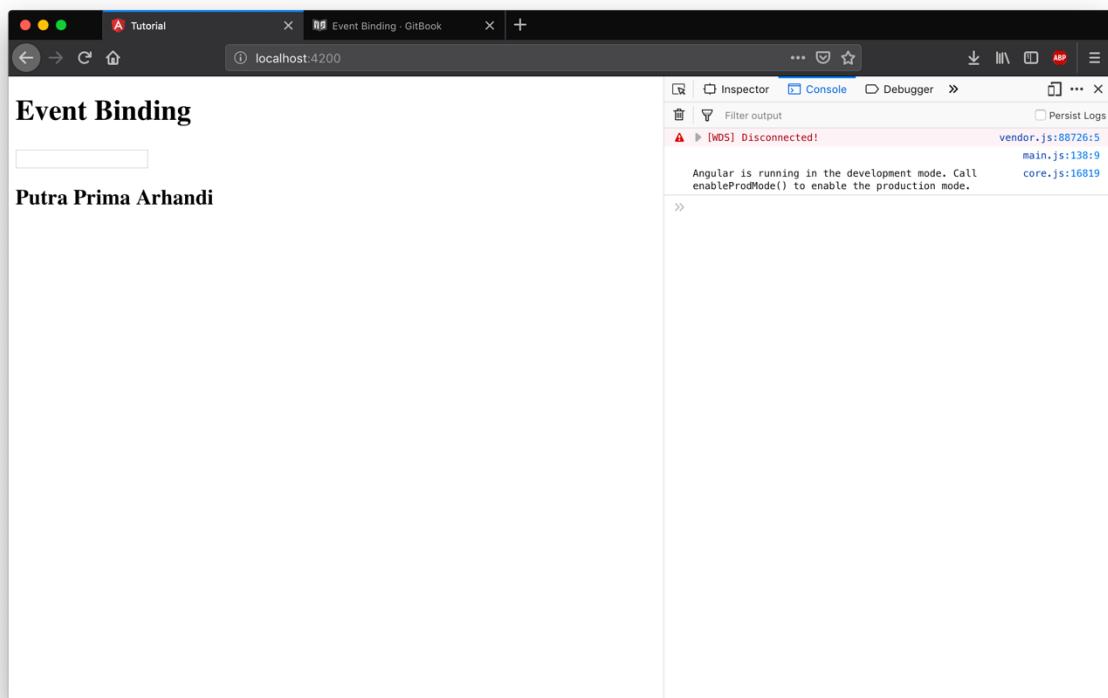
```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'Putra Prima Arhandi';
10  panjang: Number = 10;
11  lebar: Number = 10;
12  isTeaBreakTime: Boolean = true;
13
14  handleBlur(event: any) {
15    this.title = event.target.value;
16    console.log(event.target.value);
17 }
```
- STATUS BAR**:
 - Ln 8, Col 27
 - Spaces: 2
 - UTF-8
 - LF
 - TypeScript 3.2.2
 - TSLint
 - Prettier: ✓
 - 1 notification

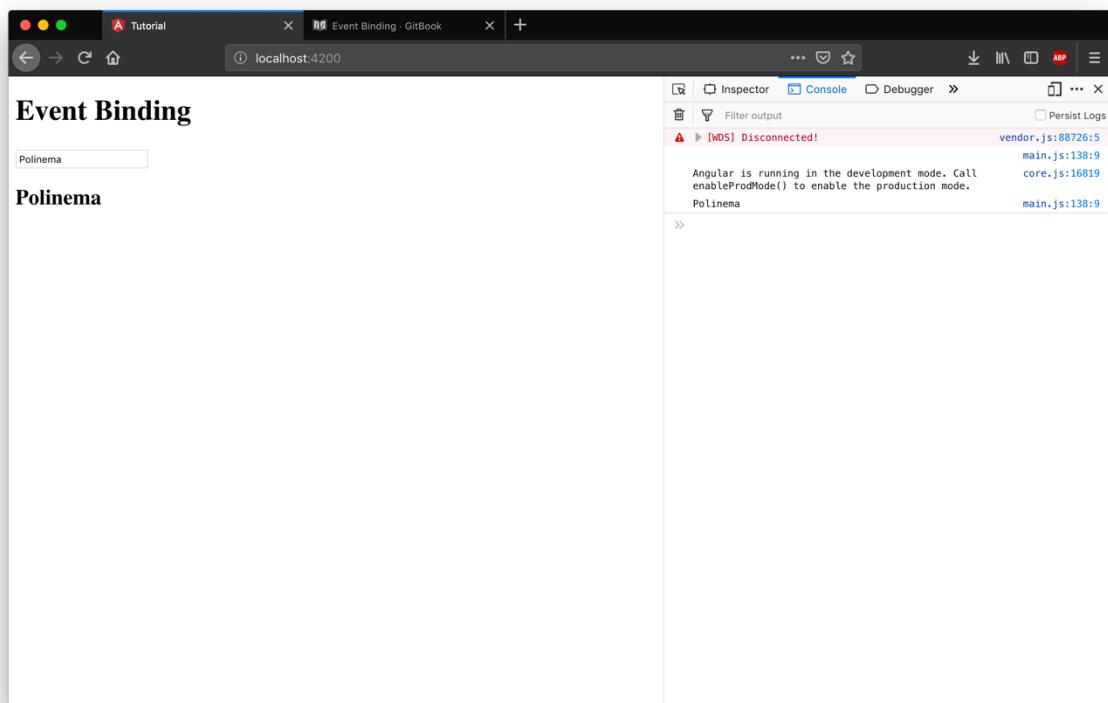
The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** sidebar: Shows the project structure with files like app.component.ts, app.component.html, and app.module.ts.
- EDITOR**:
 - File: `app.component.html`
 - Content:

```
1 <h1>Event Binding</h1>
2
3 <input type="text" (blur)="handleBlur($event)" />
4 <h2>{{ title }}</h2>
```
- STATUS BAR**:
 - Ln 4, Col 11
 - Spaces: 2
 - UTF-8
 - LF
 - HTML
 - Prettier: ✓
 - 1 notification

Hasil dari kode program di atas adalah merubah nilai h2 pada app.component.html setiap kali kita melakukan event blur pada input text.





Challenge Time

Modifikasilah kode program di atas sehingga nilai title berubah setiap kali user mengetikkan text pada input.

Modifikasi lah kode program di atas dengan membuatkan sebuah tombol untuk kembali me reset nilai variabel title kembali seperti semula (hints : button pada html mempunyai event onClick yang dapat di trigger menggunakan eventBinding (click))

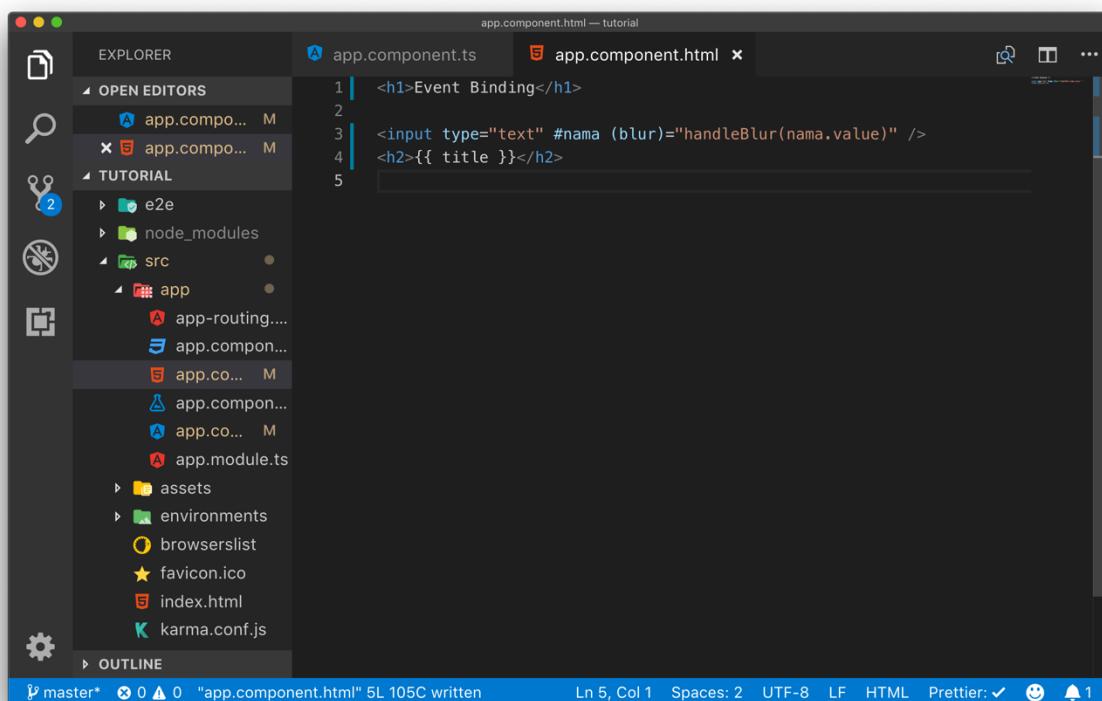
CHAPTER 10

Template Reference

Template Reference adalah cara untuk mempermudah akses terhadap binding kepada suatu element.

Langkah Percobaan

Ubahlah kode program pada file app.component.html dengan menambahkan template ref dengan nama #nama pada input text. Dan ganti lah parameter pada fungsi handleBlur().



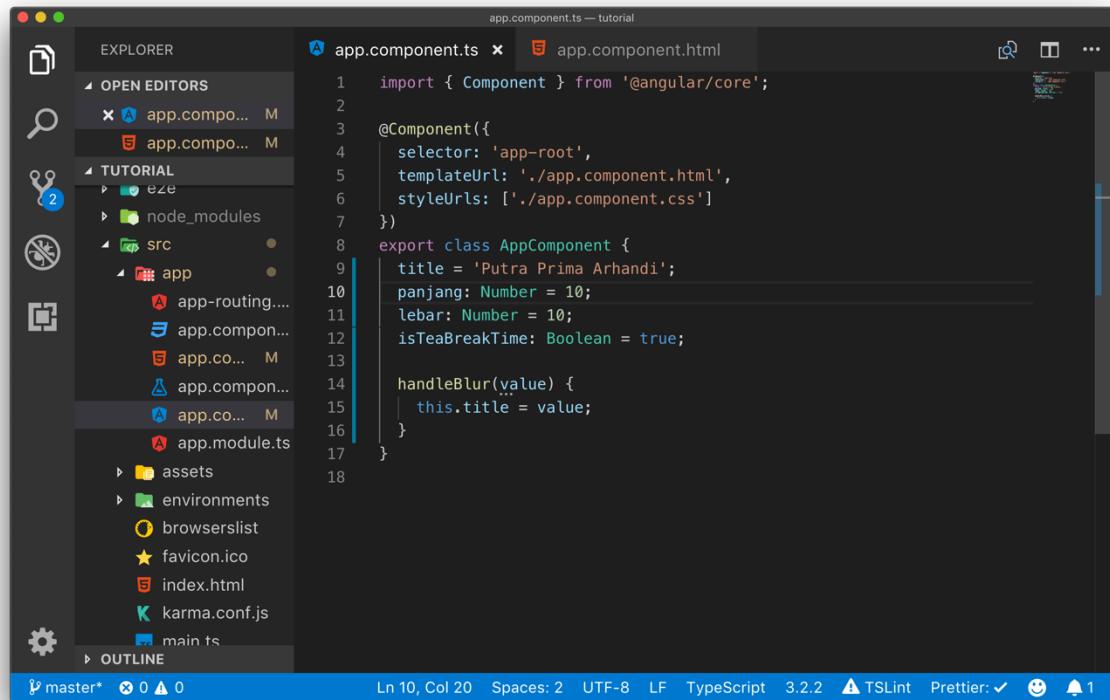
```
app.component.html — tutorial
EXPLORER      app.component.ts    app.component.html x
OPEN EDITORS   app.compo... M
TUTORIAL      x app.compo... M
               ▾ OPEN EDITORS
               ▾ TUTORIAL
               ▾ e2e
               ▾ node_modules
               ▾ src
               ▾ app
               ▾ app-routing.... M
               ▾ app.compon...
               ▾ app.co... M
               ▾ app.co... M
               ▾ app.module.ts
               ▾ assets
               ▾ environments
               ▾ browserslist
               ▾ favicon.ico
               ▾ index.html
               ▾ karma.conf.js
               ▾ OUTLINE
Ln 5, Col 1  Spaces: 2  UTF-8  LF  HTML  Prettier: ✓  1
master*  0 ▲ 0 "app.component.html" 5L 105C written
```

The screenshot shows the VS Code interface with the 'app.component.html' file open in the editor. The code is as follows:

```
<h1>Event Binding</h1>
<input type="text" #nama (blur)="handleBlur(nama.value)" />
<h2>{{ title }}</h2>
```

Selanjutnya ubah lagi kode program pada app.component.ts dengan mengganti fungsi handleBlur.





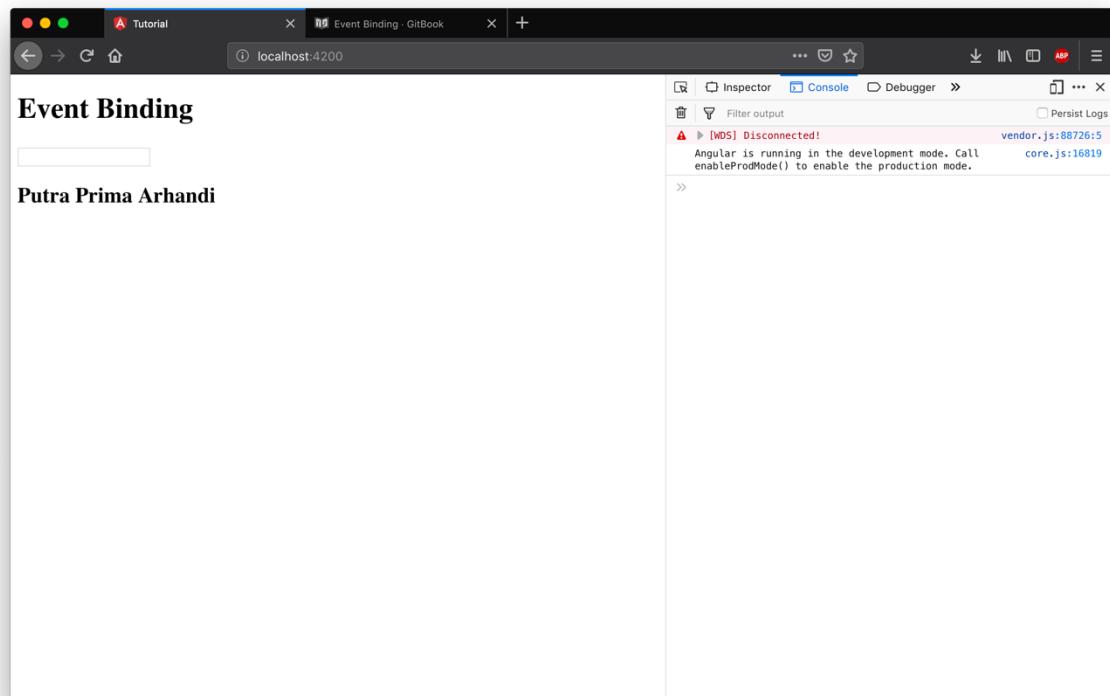
The screenshot shows the Visual Studio Code interface with the following details:

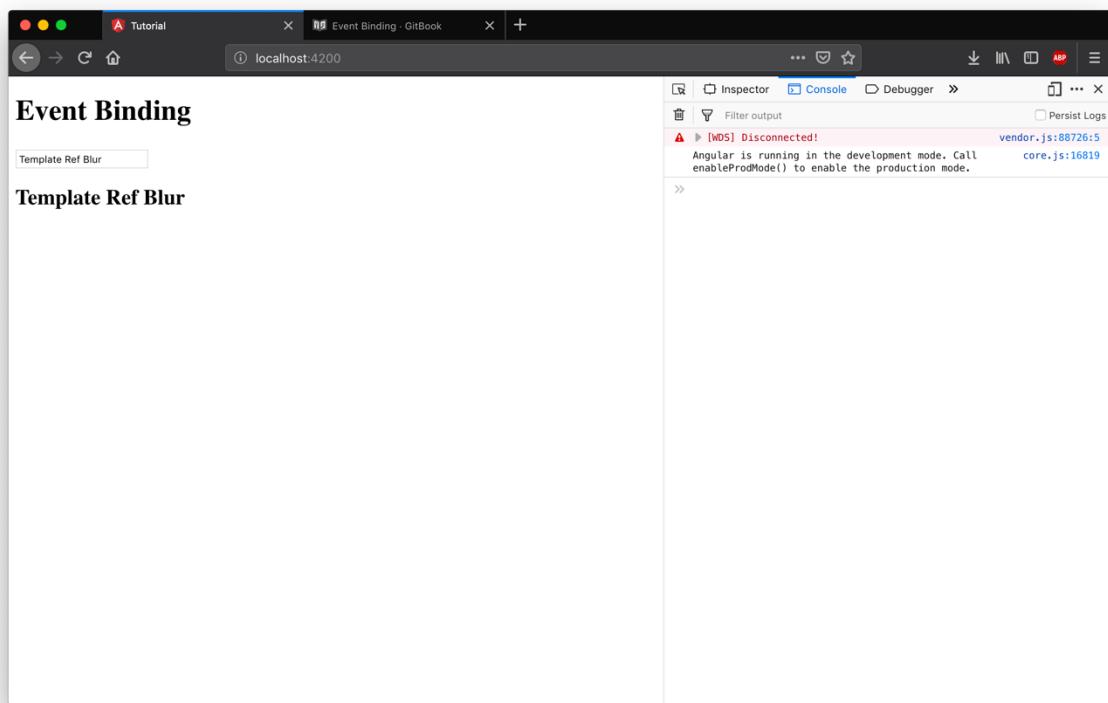
- EXPLORER** sidebar: Shows the project structure with files like app.component.ts, app.component.html, app-routing.module.ts, app.component.css, app.module.ts, assets, environments, browserslist, favicon.ico, index.html, karma.conf.js, and main.ts.
- EDITOR**: The active file is `app.component.ts`, containing the following TypeScript code:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8
9 export class AppComponent {
10   title = 'Putra Prima Arhandi';
11   panjang: Number = 10;
12   lebar: Number = 10;
13   isTeaBreakTime: Boolean = true;
14
15   handleBlur(value) {
16     this.title = value;
17   }
18 }
```

Bottom status bar: Ln 10, Col 20 Spaces: 2 UTF-8 LF TypeScript 3.2.2 TSLint Prettier: ✓

Dengan kode program yang lebih rapi menggunakan template ref kita mendapatkan hasil yang sama pada halaman web.





CHAPTER 11

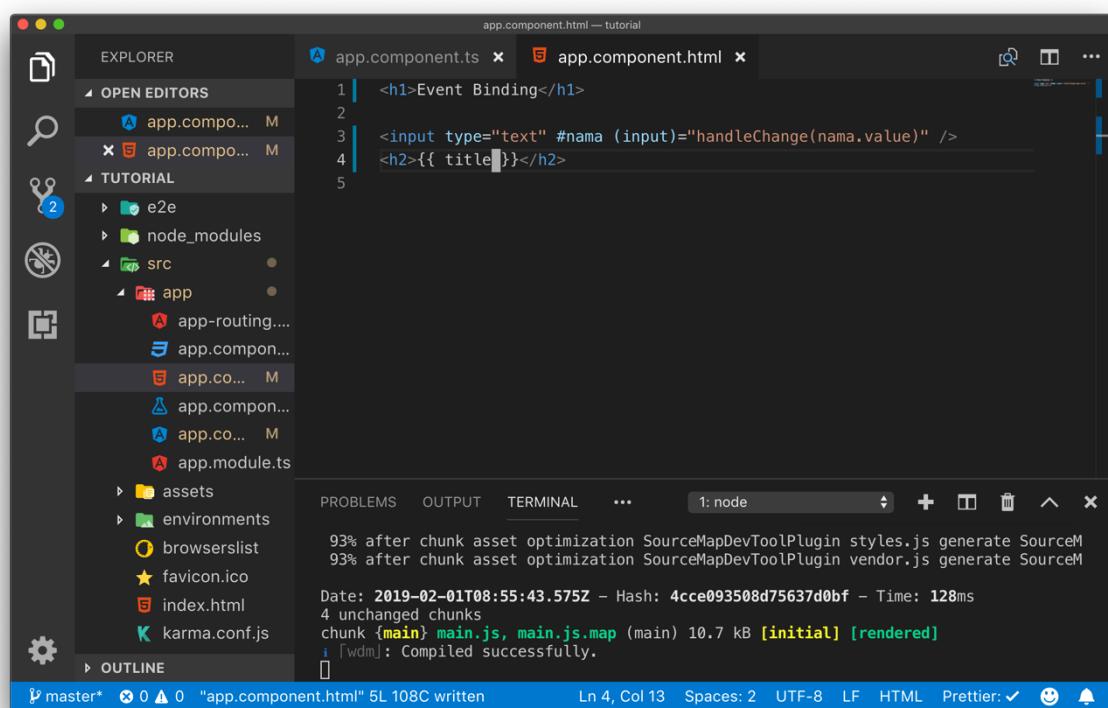
Percabangan (ngIf)

Seperti bahasa pemrograman pada umumnya angular juga memiliki sintaks untuk melakukan percabangan. Nglf pada angular sama seperti percabangan pada bahasa pemrograman lain namun dengan template dan sintaks khusus pada template htmlnya. Untuk percabangan menggunakan nglf khusus untuk percabangan pada file template html, untuk file class component percabangannya menggunakan sintaks if seperti biasa.

Langkah Percobaan

Untuk mempermudah pemahaman terhadap nglf silahkan melakukan langkah percobaan berikut ini.

Dengan file yang ada dari percobaan sebelumnya modifikasi lah sehingga input text yang di isi oleh user memiliki event (`input`), hal ini dapat anda lakukan dengan mengubah event binding yang ada pada `app.component.html` yang sebelumnya (`blur`) menjadi (`input`) kemudian ubahlah handler pada event binding tersebut menjadi `handleChange()`



```
<h1>Event Binding</h1>
<input type="text" #nama (input)="handleChange(nama.value)" />
<h2>{{ title }}</h2>
```

PROBLEMS OUTPUT TERMINAL ... 1: node

93% after chunk asset optimization SourceMapDevToolPlugin styles.js generate SourceM
93% after chunk asset optimization SourceMapDevToolPlugin vendor.js generate SourceM

Date: 2019-02-01T08:55:43.575Z - Hash: 4cce093508d75637d0bf - Time: 128ms

4 unchanged chunks

chunk {main} main.js, main.js.map (main) 10.7 kB [initial] [rendered]

i [wdm]: Compiled successfully.

master* 0 0 "app.component.html" 5L 108C written Ln 4, Col 13 Spaces: 2 UTF-8 LF HTML Prettier: ✓

Selanjutnya kita sesuaikan nama fungsi di `app.component.ts` dengan perubahan yang kita lakukan di `app.component.html`.

```

app.component.ts — tutorial
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'Putra Prima Arhandi';
10   panjang: Number = 10;
11   lebar: Number = 10;
12   isTeaBreakTime: Boolean = true;
13
14   handleChange(value) {
15     this.title = value;
16   }
17 }

PROBLEMS OUTPUT TERMINAL ...
Date: 2019-02-01T08:56:05.564Z - Hash: 8344b8d7811d6dc78927 - Time: 135ms
5 unchanged chunks
[wdm]: Compiled successfully.

```

Pada percobaan ini kita akan menampilkan sebuah elemen dom baru yang isinya merupakan input dari user di input text jika input tersebut sudah mencapai minimal 3 karakter. Hal ini dapat dilakukan dengan menggunakan directive nglf.

Langkah selanjutnya adalah menambahkan directive nglf ke template app.component.html.

```

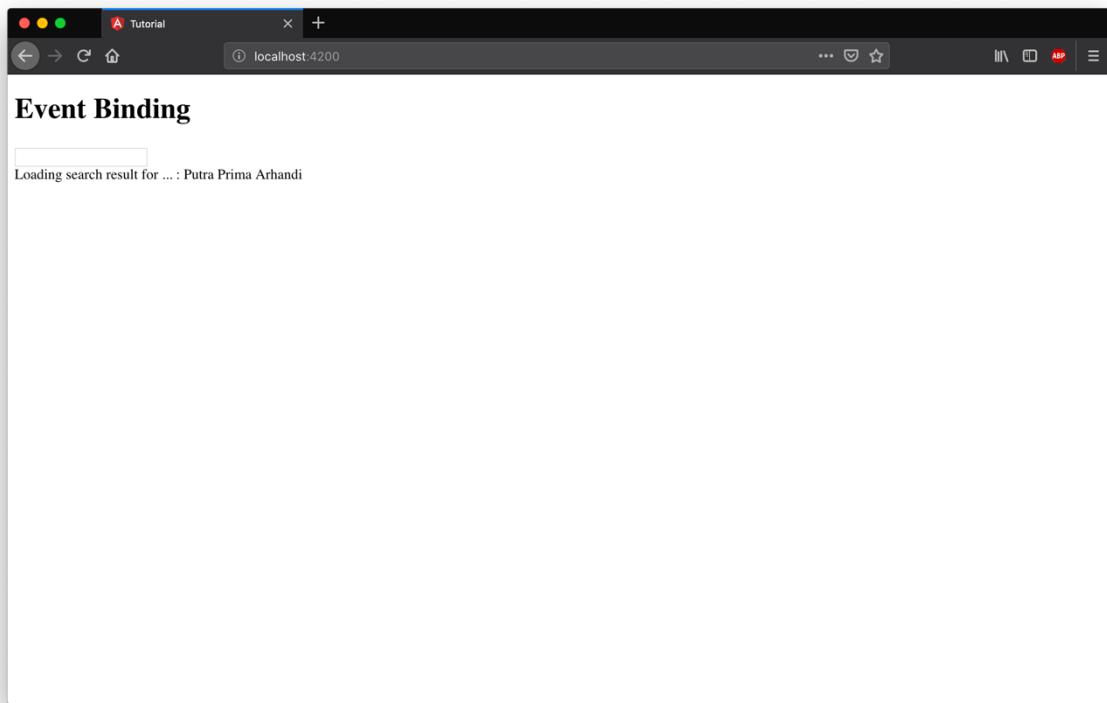
app.component.html — tutorial
1 <h1>Event Binding</h1>
2
3 <input type="text" #nama (input)="handleChange(nama.value)" />
4 <div *ngIf="title.length">
5   | Loading search result for ... : {{ title }}
6 </div>
7

PROBLEMS OUTPUT TERMINAL ...
Date: 2019-02-01T08:55:56.100Z - Hash: 8344b8d7811d6dc78927 - Time: 173ms
4 unchanged chunks
chunk {main} main.js, main.js.map (main) 10.7 kB [initial] [rendered]
[wdm]: Compiled successfully.

```

Dengan menggunakan kode program di atas kita dapat membatasi div yang berisi informasi search result akan tampil jika variabel title memiliki nilai length.

Selanjutnya jalankanlah server angular dengan perintah `ng-serve` kemudian bukalah dengan web browser.



Cobalah untuk mengisi input kemudian perhatikan hasilnya.

Challenge Time

Pada saat anda menjalankan server pada awal start sudah muncul kata `Loading search result for ...` : modifikasi app.component.ts sehingga pada awal start kata tersebut tidak muncul. (hint: variabel title pada awal start server panjangnya harus 0)

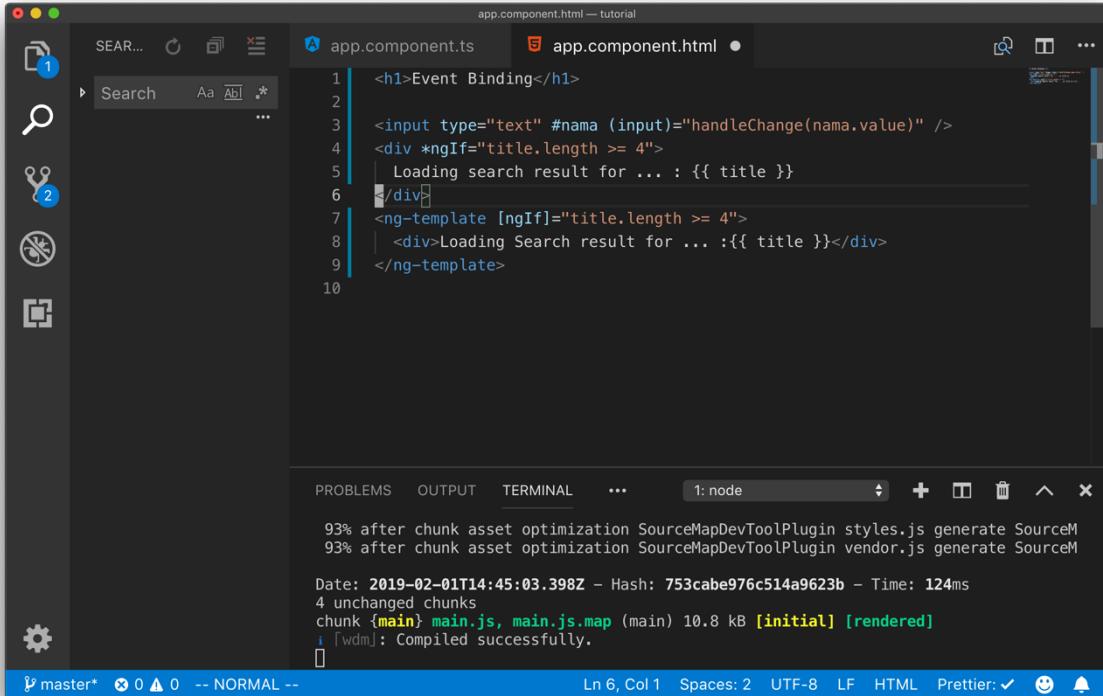
Challenge selanjutnya adalah membatasi munculnya kata `Loading search result for ...` hanya setelah user menginputkan 4 kata. (hint: `ngIf` bisa memiliki operator perbandingan di dalam argumen nya)

`ngIf` , `else` , `then`, `ngIfElse`

Tidak hanya if statement yang dapat dipakai di template html angular, selain itu kita juga dapat menggunakan else dan then. Untuk memahaminya lakukanlah langkah percobaan dibawah ini

Langkah Percobaan

Saat menggunakan `*ngIf` perhatikan bahwa ada tanda `*` pada dasarnya `*ngIf` adalah sugar syntax untuk mempermudah penulisan kode program.
Ubahlah kode program pada `app.component.html` menjadi seperti dibawah ini :

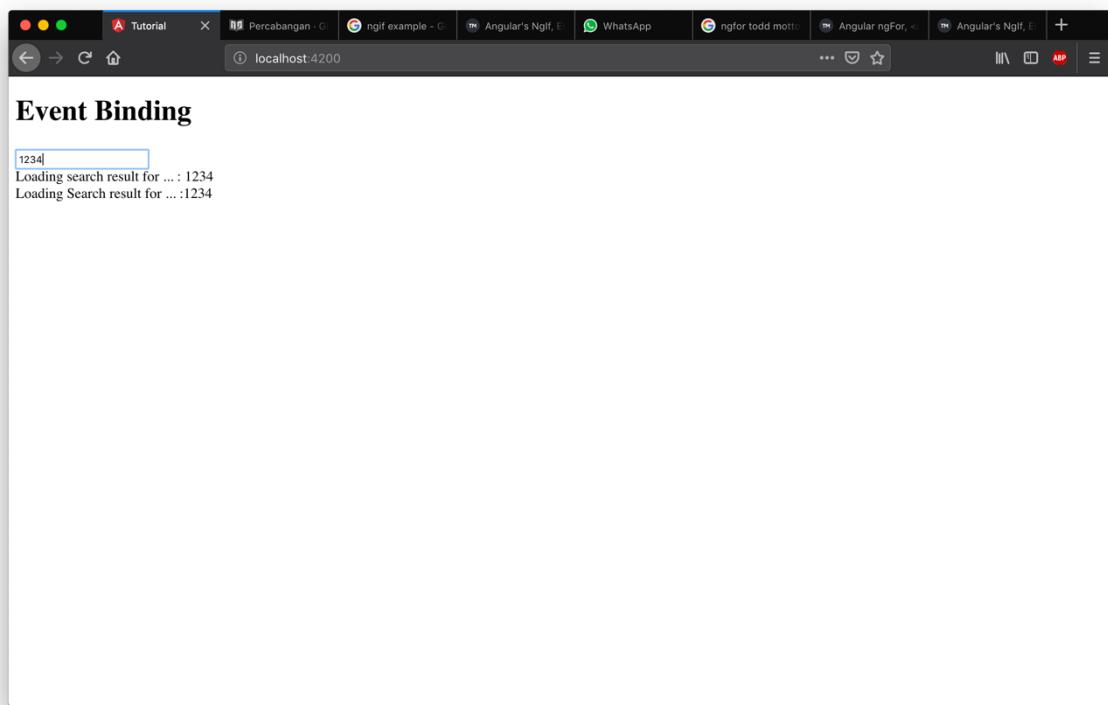


```
app.component.html — tutorial
app.component.ts app.component.html ●
1 <h1>Event Binding</h1>
2
3 <input type="text" #nama (input)="handleChange(nama.value)" />
4 <div *ngIf="title.length >= 4">
5   Loading search result for ... : {{ title }}
6 </div>
7 <ng-template [ngIf]="title.length >= 4">
8   <div>Loading Search result for ... :{{ title }}</div>
9 </ng-template>
10

PROBLEMS OUTPUT TERMINAL ...
1: node
Date: 2019-02-01T14:45:03.398Z - Hash: 753cabef976c514a9623b - Time: 124ms
93% after chunk asset optimization SourceMapDevToolPlugin styles.js generate SourceM
93% after chunk asset optimization SourceMapDevToolPlugin vendor.js generate SourceM
4 unchanged chunks
chunk {main} main.js, main.js.map (main) 10.8 KB [initial] [rendered]
  i [wdm]: Compiled successfully.

Ln 6, Col 1  Spaces: 2  UTF-8  LF  HTML  Prettier: ✓  ☺  🔔
```

Jalankanlah server angular kemudian perhatikan bahwa kedua kode program `*ngIf` dan `<ng-template [ngIf]>` memberikan hasil yang sama

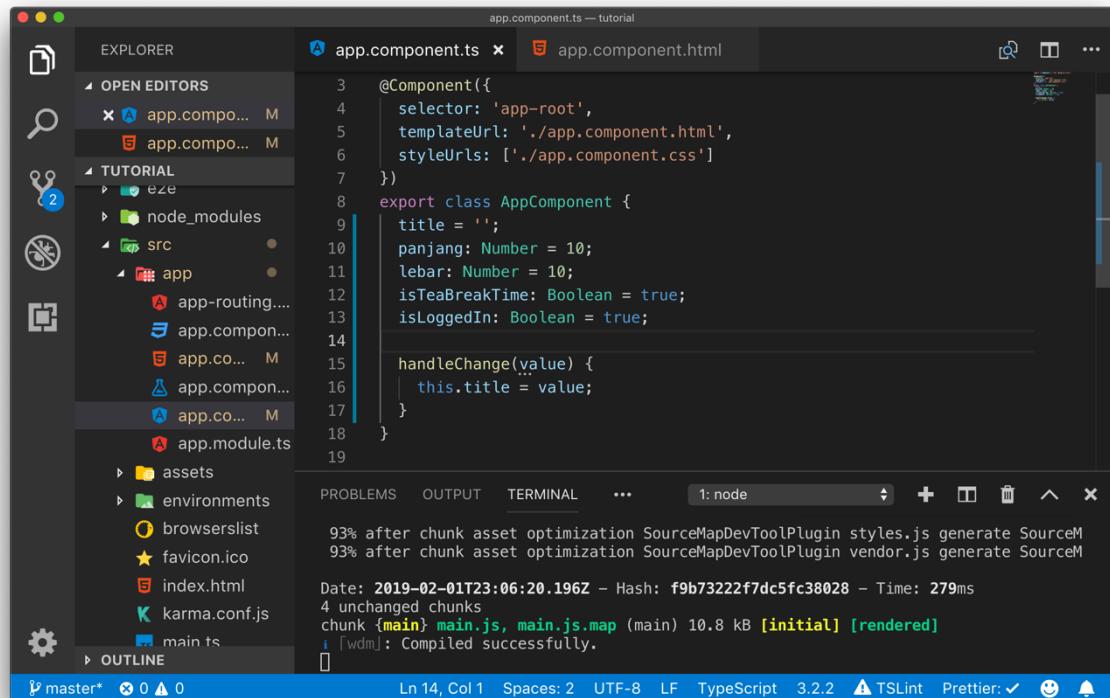


```
<div *ngIf="title.length >= 4">Loading search result for ... : {{ title }}</div>
<ng-template [ngIf]="title.length >= 4">
  <div>Loading Search result for ... :{{ title }}</div>
</ng-template>
```

Dari percobaan diatas dapat dilihat bahwa penggunaan `*ngIf` adalah sugar syntax untuk `ng-template`

Dengan menggunakan `ng-template` kita dapat memperluas penggunaan `ngIf` dengan tambahan `ngIfElse`, `then` dan `else`.

Untuk mencobanya lakukan perubahan pada file `app.component.ts` dengan menambahkan satu variabel baru yaitu `isLoggedIn`

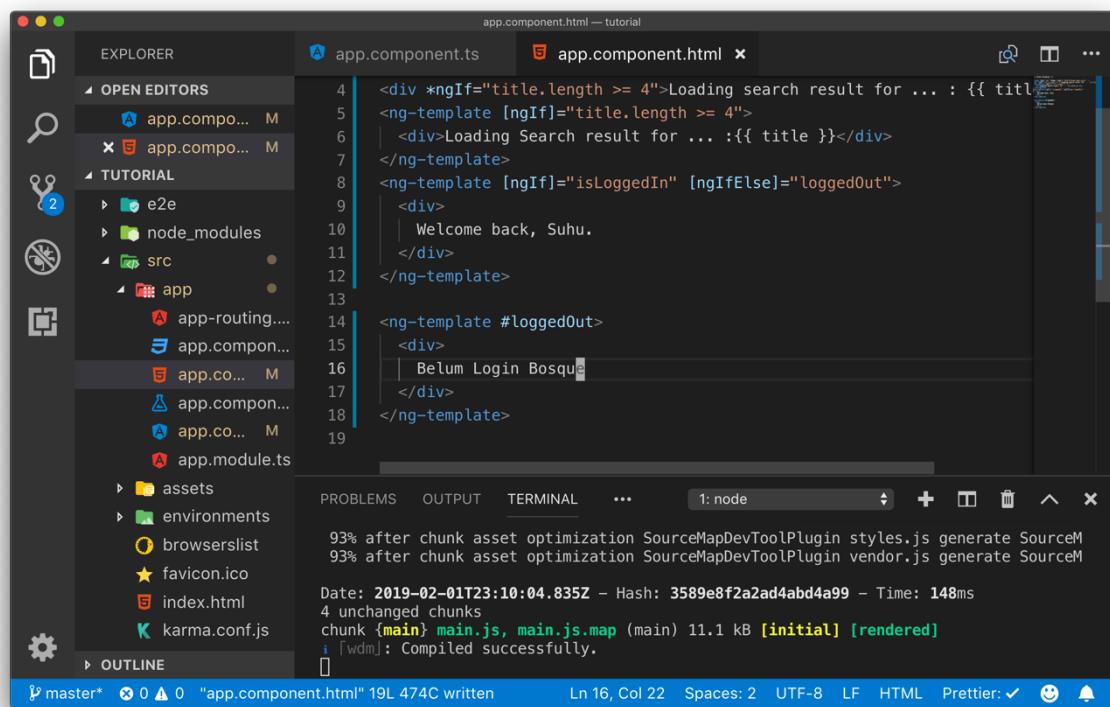


```
app.component.ts — tutorial
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = '';
10   panjang: Number = 10;
11   lebar: Number = 10;
12   isTeaBreakTime: Boolean = true;
13   isLoggedIn: Boolean = true;
14
15   handleChange(value) {
16     this.title = value;
17   }
18 }

PROBLEMS OUTPUT TERMINAL ...
93% after chunk asset optimization SourceMapDevToolPlugin styles.js generate SourceM
93% after chunk asset optimization SourceMapDevToolPlugin vendor.js generate SourceM
Date: 2019-02-01T23:06:20.196Z - Hash: f9b73222f7dc5fc38028 - Time: 279ms
4 unchanged chunks
chunk {main} main.js, main.js.map (main) 10.8 kB [initial] [rendered]
i [wdm]: Compiled successfully.

Ln 14, Col 1  Spaces: 2  UTF-8  LF  TypeScript  3.2.2  ▲ TSLint  Prettier:✓  ☺  🔔
```

Selanjutnya tambahkan kode html berikut ini pada app.component.html.

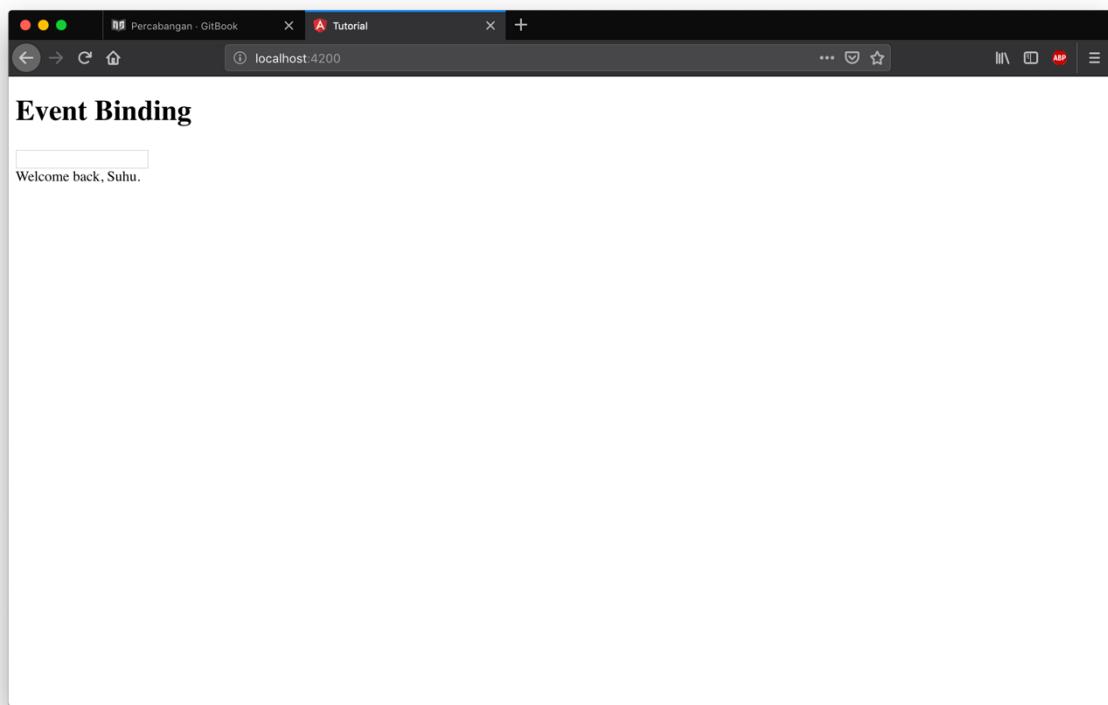


```
app.component.html — tutorial
4  <div *ngIf="title.length >= 4">Loading search result for ... : {{ title }}
5  <ng-template [ngIf]="title.length >= 4">
6    <div>Loading Search result for ... :{{ title }}</div>
7  </ng-template>
8  <ng-template [ngIf]="isLoggedIn" [ngIfElse]="loggedOut">
9    <div>
10      Welcome back, Suhu.
11    </div>
12  </ng-template>
13
14  <ng-template #loggedOut>
15    <div>
16      Belum Login Bosque
17    </div>
18  </ng-template>

PROBLEMS OUTPUT TERMINAL ...
93% after chunk asset optimization SourceMapDevToolPlugin styles.js generate SourceM
93% after chunk asset optimization SourceMapDevToolPlugin vendor.js generate SourceM
Date: 2019-02-01T23:10:04.835Z - Hash: 3589e8f2a2ad4abd4a99 - Time: 148ms
4 unchanged chunks
chunk {main} main.js, main.js.map (main) 11.1 kB [initial] [rendered]
i [wdm]: Compiled successfully.

Ln 16, Col 22  Spaces: 2  UTF-8  LF  HTML  Prettier:✓  ☺  🔔
```

Jalankan server angular kemudian perhatikan output yang dihasilkan.



Perhatikan pada tulisan `Welcome back, Suhu.` ini menunjukkan `ng-template` sudah dapat bekerja dengan baik karena output yang dihasilkan sesuai dengan nilai variabel `isLoggedIn` pada `app.component.ts`

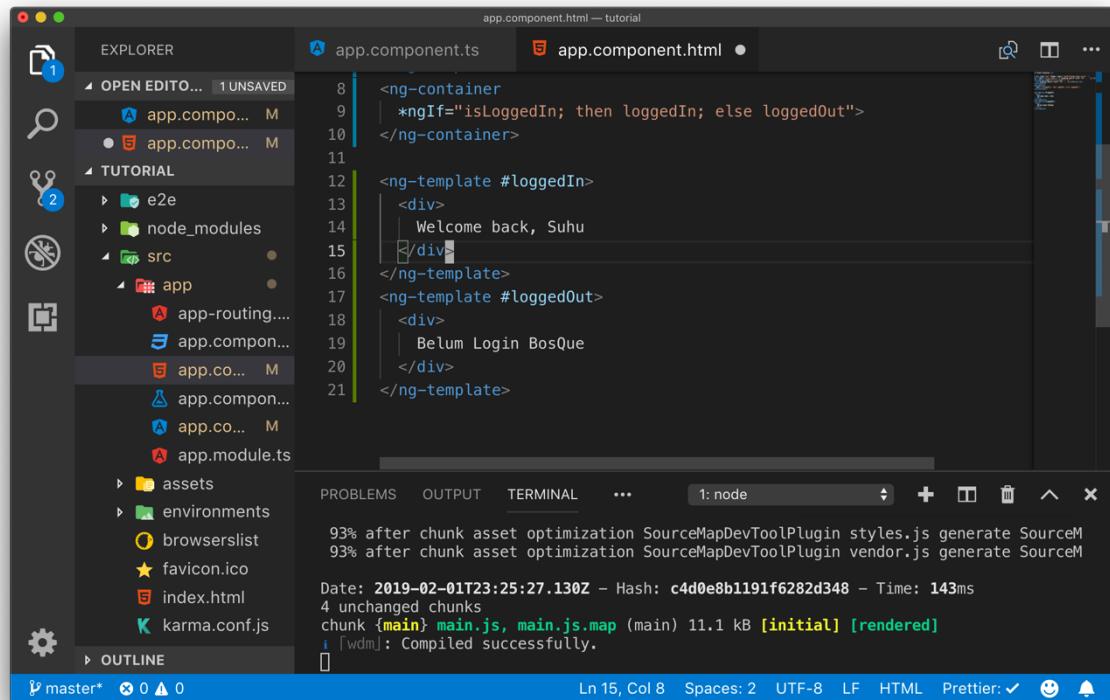
Challenge Time

Cobalah merubah nilai `isLoggedIn` menjadi `false` kemudian perhatikan perubahan apa yang terjadi

Ubah lah tulisan `Welcome back, Suhu.` menjadi sesuai dengan data yang dituliskan user pada input text.

Selain itu `ng-template` juga dapat digunakan dengan menggunakan sintaks `else` dan `then`

Untuk mencobanya silahkan ubah `app.component.html` menjadi seperti gambar dibawah ini.



```
app.component.ts
8  <ng-container
9    *ngIf="isLoggedIn; then loggedIn; else loggedOut">
10   </ng-container>
11
12   <ng-template #loggedIn>
13     <div>
14       Welcome back, Suhu
15     </div>
16   </ng-template>
17   <ng-template #loggedOut>
18     <div>
19       Belum Login Bosque
20     </div>
21   </ng-template>
```

PROBLEMS OUTPUT TERMINAL ... 1: node

93% after chunk asset optimization SourceMapDevToolPlugin styles.js generate SourceM
93% after chunk asset optimization SourceMapDevToolPlugin vendor.js generate SourceM

Date: 2019-02-01T23:25:27.130Z - Hash: c4d0e8b1191f6282d348 - Time: 143ms

4 unchanged chunks

chunk {main} main.js, main.js.map (main) 11.1 kB [initial] [rendered]

i [wdm]: Compiled successfully.

Ln 15, Col 8 Spaces: 2 UTF-8 LF HTML Prettier: ✓

Perhatikan pada kode program diatas ditambahkan tag `ng-container` dengan sugar syntax `*ngIf` dan tetap menggunakan `ng-template` untuk tampilan `then` dan `else` nya.

CHAPTER 12

Perulangan

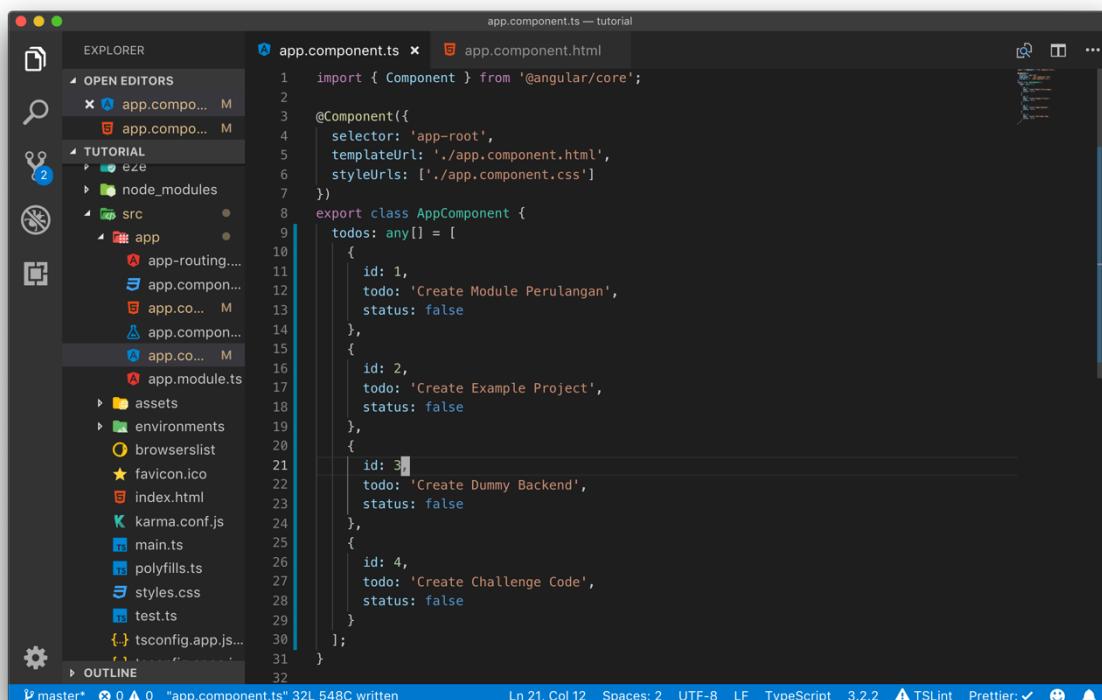
Perulangan pada angular menggunakan sugar syntax `*ngFor`, perulangan ini dapat dilakukan dengan menggunakan dataset yang didapatkan dari component.

Sintax perulangan dengan `*ngFor` dilakukan pada file template html dari sebuah component, perulangan dapat dilakukan terhadap semua tag html ataupun tag baru dari component lain.

Untuk memahami perulangan dengan `*ngFor` lakukanlah langkah percobaan dibawah ini.

Langkah Percobaan

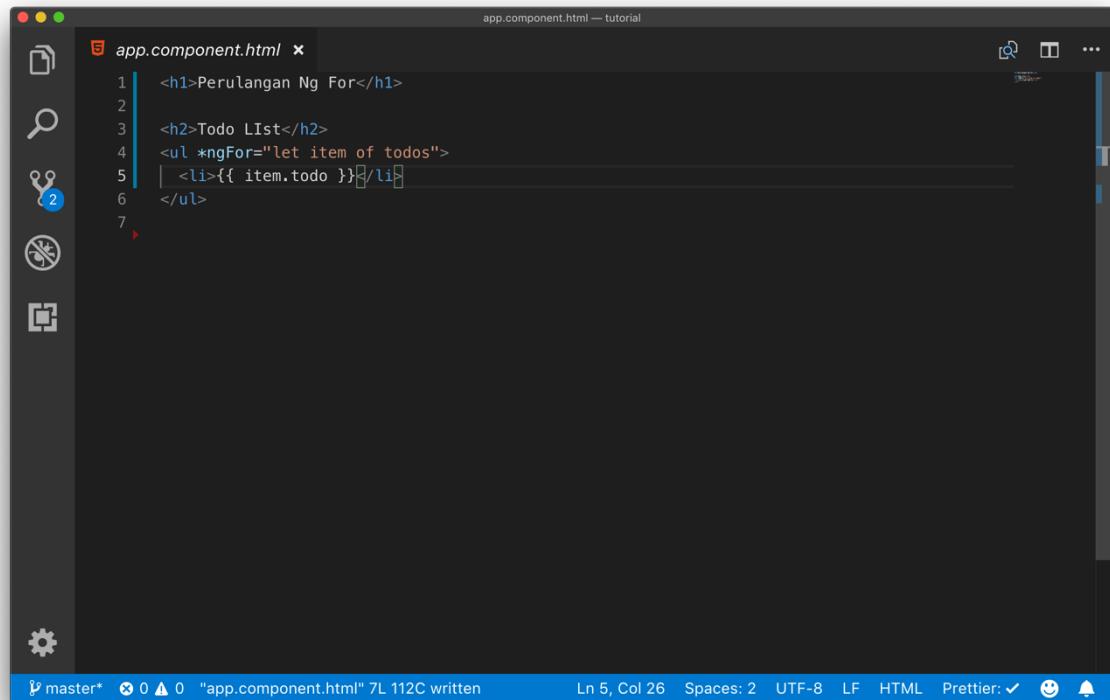
Hapuslah kode program dari percobaan sebelumnya pada file `app.component.ts` kemudian ubah menjadi seperti gambar dibawah ini.



```
app.component.ts — tutorial
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   todos: any[] = [
10     {
11       id: 1,
12       todo: 'Create Module Perulangan',
13       status: false
14     },
15     {
16       id: 2,
17       todo: 'Create Example Project',
18       status: false
19     },
20     {
21       id: 3,
22       todo: 'Create Dummy Backend',
23       status: false
24     },
25     {
26       id: 4,
27       todo: 'Create Challenge Code',
28       status: false
29     }
30   ];
31 }
```

Pada kode program di atas semua kode program sebelumnya di dalam class `AppComponent` dihapus kemudian diisikan sebuah variabel baru dengan nama `todos` yang berupa sebuah array of objects dengan tipe data `any`. Tipe data `any` adalah tipe data yang dapat menerima semua tipe data. `any` digunakan untuk mempermudah langkah percobaan pada tahap ini, untuk production code sebaiknya kita menggunakan variabel yang memiliki tipe data.

Setelah mempunyai variabel baru, selanjutnya ubahlah file `app.component.html` dari kode program sebelumnya di ganti menjadi seperti gambar di bawah ini.



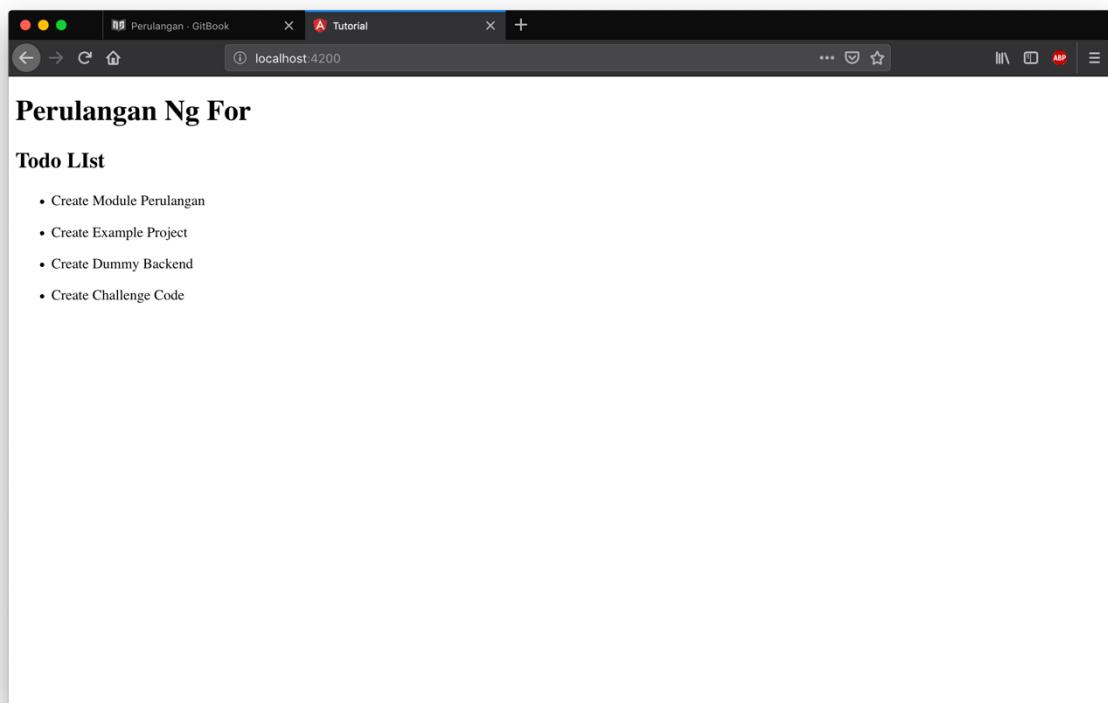
The screenshot shows a code editor window with the file `app.component.html` open. The code is as follows:

```
<h1>Perulangan Ng For</h1>
<h2>Todo LIst</h2>
<ul *ngFor="let item of todos">
| <li>{{ item.todo }}</li>
</ul>
```

The editor has a dark theme with light-colored syntax highlighting. The status bar at the bottom shows the file is in the `master*` branch, has 0 changes, and was written 7L 112C ago. It also shows the current line (Ln 5, Col 26), spaces (Spaces: 2), encoding (UTF-8), line endings (LF), file type (HTML), and Prettier status (✓).

Pada kode program diatas kita melakukan perulangan dengan menampilkan `li` sebanyak jumlah data pada variabel `todos`, `*ngFor` dipasangkan pada tag yang melingkupi element yang akan di ulang, dalam hal ini di pasang di tag `ul`

Untuk melihat hasil kode program diatas jalankan server angularnya dengan perintah `ng serve`, berikut ini hasil dari kode program diatas :



Dalam perulangan tentu saja memiliki index, index pada perulangan sama seperti index pada array yang dimulai dari nol.

Untuk mencoba index pada perulangan ubahlah kode program pada app.component.html menjadi seperti pada gambar dibawah ini

A screenshot of a code editor showing the file "app.component.html" with the following content:

```
<h1>Perulangan Ng For</h1>
<h2>Todo LIst</h2>
<ul *ngFor="let item of todos; let i = index">
| <li>{{ i }}:{{ item.todo }}</li>
</ul>
```

The code editor interface includes a sidebar with icons for file operations like save, search, and copy/paste, and a status bar at the bottom showing file details and a Prettier status.

setelah itu bukalah web browser dan liat perubahan apa yang terjadi.

The screenshot shows a web browser window with two tabs: 'Perulangan - GitBook' and 'Tutorial'. The active tab is 'Tutorial' at 'localhost:4200'. The page title is 'Perulangan Ng For'. Below the title, there is a section titled 'Todo List' containing the following bullet points:

- 0:Create Module Perulangan
- 1:Create Example Project
- 2:Create Dummy Backend
- 3:Create Challenge Code

Challenge Time

NgFor dan Nglf dapat digabung dalam template html, untuk memahaminya ubahlah tampilan Todo List untuk hanya menampilkan todo yang memiliki id genap atau yang ganjil saja.

CHAPTER 13

Style Binding

Pada angular untuk menerapkan suatu style kepada sebuah tag html dapat digunakan beberapa opsi antara lain :

1. ngStyle
2. ngClass

ngStyle

ngStyle digunakan untuk melakukan binding sebuah style ke tag html dengan format css, memanfaatkan javascript properties yang sesuai dengan style css yang bisa dipasang pada sebuah tag. Dengan menggunakan ngStyle kita menulis style nya di template html penulisan style dengan ngStyle ini mirip dengan proses inline style pada html.

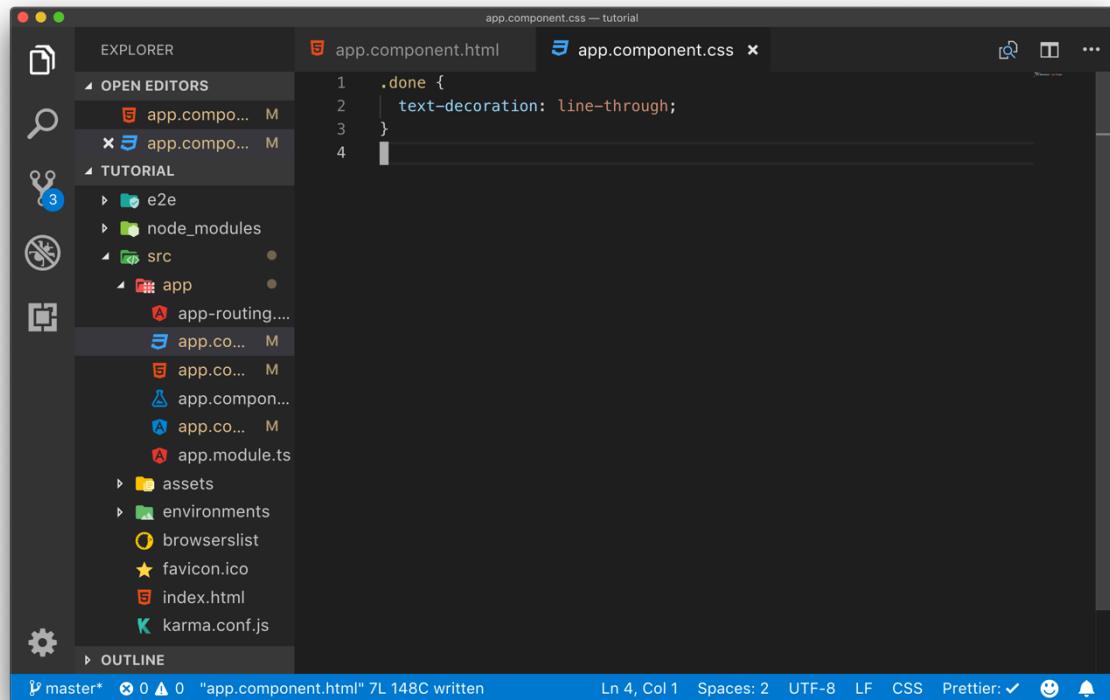
ngClass

ngClass digunakan untuk melakukan binding sebuah class css ke sebuah tag html, dalam menggunakan ngClass kita menulis class nya di file style dari sebuah component kemudian membinding nya ke tag html

Langkah Percobaan

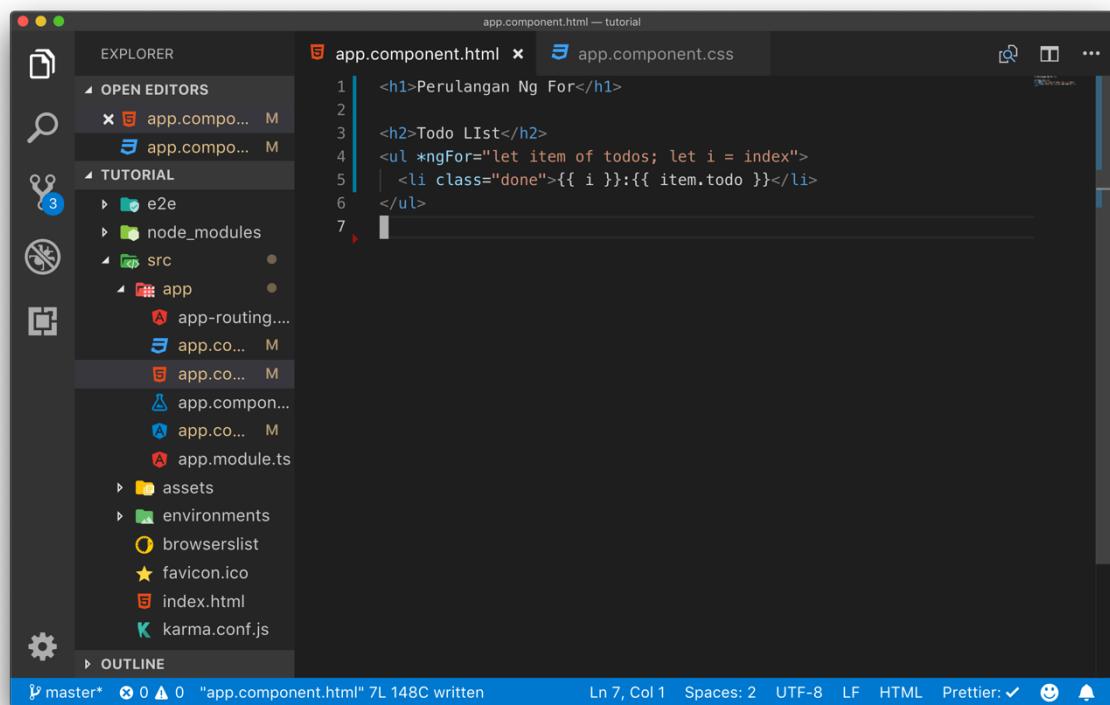
Untuk memahami penggunaan `ngStyle` dan `ngClass` lakukanlah langkah percobaan dibawah ini :

Bukalah file app.component.css kemudian tambahkan class css seperti pada gambar berikut ini :



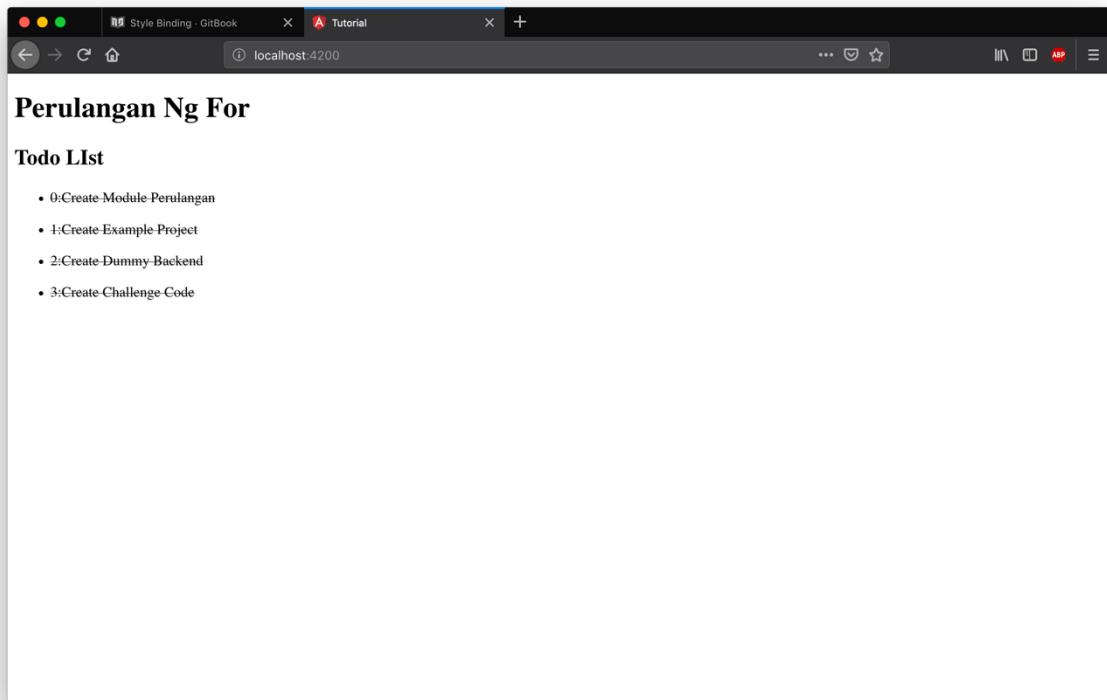
```
.done {
  text-decoration: line-through;
}
```

Kemudian bukalah file app.component.html lalu tambahkan class .done pada tag li



```
<h1>Perulangan Ng For</h1>
<h2>Todo LIst</h2>
<ul *ngFor="let item of todos; let i = index">
  <li class="done">{{ i }}:{{ item.todo }}</li>
</ul>
```

Kemudian buka lah aplikasi angular melalui web, pada saat ini anda akan melihat semua task dalam todo list sudah berubah menjadi text yang tercoret, karena dipasang sebuah css text-decoration: line-through



- 0:Create Module-Perulangan
- 1:Create Example Project
- 2:Create Dummy Backend
- 3:Create Challenge Code

Selanjutnya modifikasi file app.component.html untuk menampilkan `text-decoration:line-through` hanya jika todo memiliki status true hal ini dapat dilakukan dengan menambahkan binding `ngClass` pada tag `li` seperti pada gambar dibawah ini.

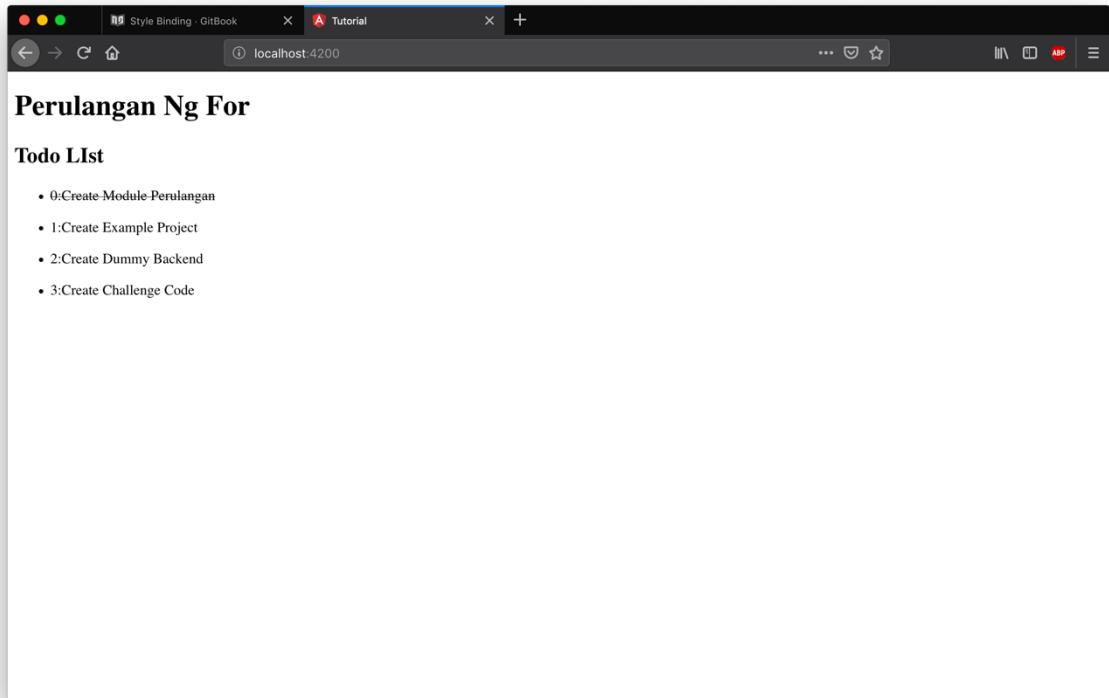
The screenshot shows the Visual Studio Code interface. The left sidebar (EXPLORER) shows a project structure with files like app.component.html, app.component.css, and app.component.ts. The main editor area displays the following code for `app.component.html`:

```
<h1>Perulangan Ng For</h1>
<h2>Todo List</h2>
<ul *ngFor="let item of todos; let i = index">
  <li
    [ngClass]="{
      | done: item.status
    }"
  >
    {{ i }}:{{ item.todo }}
  </li>
</ul>
```

At the bottom, the status bar shows: 'master*' '0 0 0 "app.component.html" 13L 194C written' 'Ln 4, Col 23' 'Spaces: 2' 'UTF-8' 'LF' 'HTML' 'Prettier:✓' with some icons.

Pada kode program di atas kita menambahkan binding ke `ngClass` dimana

argumen dari untuk `ngClass` ini berupa sebuah object, maka kita memberikan properties done sesuai dengan value dari `todo.status`



Hal yang sama dapat dilakukan dengan menggunakan sintaks `ngStyle` untuk mencoba tambahkan kode program ini ke `app.component.html` anda.

A screenshot of a code editor (VS Code) showing the file `app.component.html`. The code contains two sections of `` elements using `*ngFor` and `ngClass` or `ngStyle` bindings.

```
<h1>Perulangan Ng For</h1>
<h2>Todo LIst</h2>
<ul *ngFor="let item of todos; let i = index">
  <li
    [ngClass]="{
      | done: item.status
    }"
  >
    {{ i }}:{{ item.todo }}
  </li>
</ul>
<h2>Todo LIst</h2>
<ul *ngFor="let item of todos; let i = index">
  <li
    [ngStyle]="{
      | textDecoration: item.status ? 'line-through' : 'none'
    }"
  >
    {{ i }}:{{ item.todo }}
  </li>
</ul>
```

pada kode program di atas kita melakukan binding dengan menggunakan `ngStyle` khusus binding dengan `ngStyle` ini kita menggunakan

object yang properties nya adalah properties css dan nilai nya adalah nilai dari css nya bukan class css yang kita buat sebelumnya.

CHAPTER 14

Pipes

Pipes adalah sebuah fungsi sederhana yang bertugas melakukan transformasi data / input menjadi data yang terformat sesuai dengan yang dibutuhkan.

Contoh Pipes sederhana adalah pipes untuk memformat tanggal. Pada umumnya format tanggal pada database atau rest api disimpan dalam bentuk sebuah date time format. Untuk menampilkan ke user format ini harus diubah ke format yang sesuai dengan keinginan user.

Untuk memahami cara penggunaan pipes ikutilah langkah percobaan dibawah ini

Langkah Percobaan

Ubahlah variabel todos pada app.component.ts menjadi seperti pada gambar dibawah ini.

```
app.component.ts — tutorial
OPEN EDITORS
  app.compo... M
  app.compo... M
  ✘ app.compo... M
  TUTORIAL
    node_modules
      src
        app
          app-routing.... M
          app.com... M
          app.com... M
          app.compon... M
          app.co... M
          app.module.ts
        assets
        environments
        browserslist
        favicon.ico
        index.html
        karma.conf.js
OUTLINE
Ln 32, Col 36  Spaces: 2  UTF-8  LF  TypeScript  3.2.2  TSLint  Prettier: ✓  ⚡  🎉  🔔
```

```
export class AppComponent {
  todos: any[] = [
    {
      id: 1,
      todo: 'Create Module Perulangan',
      status: true,
      createdAt: new Date(2019, 2, 1)
    },
    {
      id: 2,
      todo: 'Create Example Project',
      status: false,
      createdAt: new Date(2019, 2, 1)
    },
    {
      id: 3,
      todo: 'Create Dummy Backend',
      status: false,
      createdAt: new Date(2019, 2, 2)
    },
    {
      id: 4,
      todo: 'Create Challenge Code',
      status: false,
      createdAt: new Date(2019, 2, 2)
    }
  ]
}
```

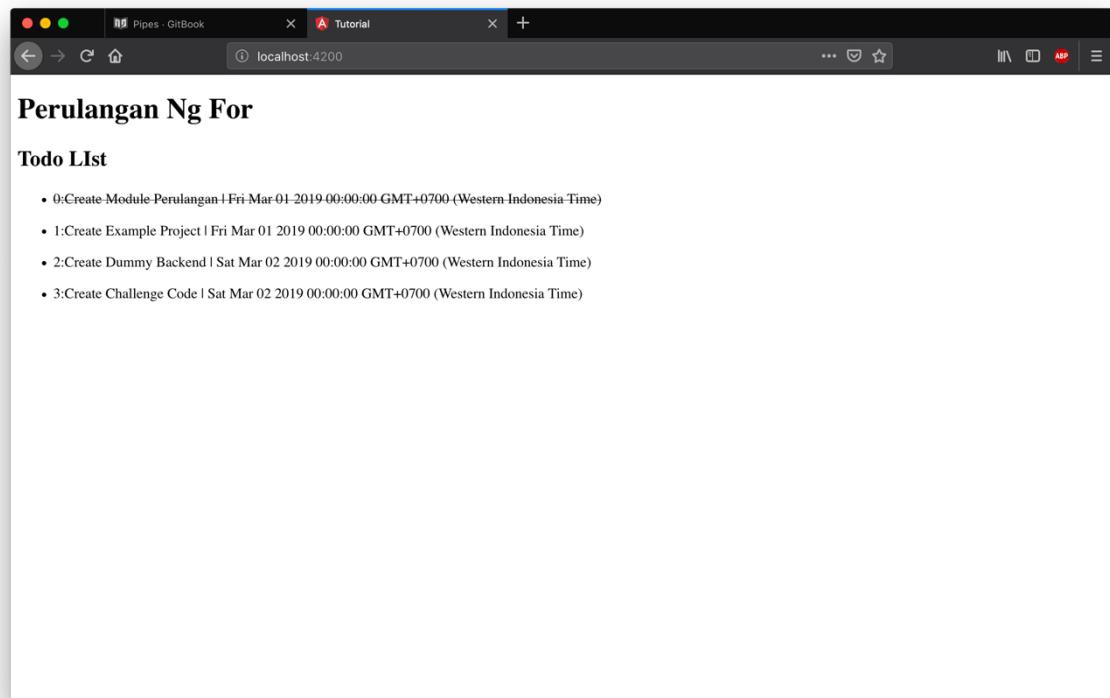
Pada kode program diatas kita melakukan perubahan pada object Todos menjadi memiliki properties `createdAt` yang memiliki tipe data `date`. Selanjutnya lakukan perubahan pada file template `app.component.html` menjadi seperti pada gambar dibawah ini :

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays the project structure, including files like app.component.html, app.component.css, and app.component.ts. The main editor area shows the following code:

```
<h1>Perulangan Ng For</h1>
<h2>Todo LIst</h2>
<ul *ngFor="let item of todos; let i = index">
  <li
    [ngClass]="{
      | done: item.status
    }"
  >
    {{ i }}:{{ item.todo }} | {{ item.createdAt }}
  </li>
</ul>
```

The status bar at the bottom indicates the file is "master*", has 0 changes, and was written 217C ago. It also shows the current line (Ln 10, Col 19), spaces (Spaces: 2), encoding (UTF-8), line endings (LF), file type (HTML), and Prettier status.

Pada kode program diatas kita menambahkan tanggal dibuatnya todo list dengan menampilkan `item.createdAt`
jalankan server angular dan lihatlah hasilnya pada web browser.



Pada gambar di atas kita sudah bisa menampilkan tanggal dibuatnya todo, namun formatnya masih tidak familiar bagi user. Untuk memperbaiki hal ini tambahkan pipes pada `app.component.html`

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** sidebar: Shows the project structure with a folder named "TUTORIAL" containing "e2e", "node_modules", and "src". Inside "src", there is an "app" folder which contains "app-routing.module.ts", "app.component.css", "app.component.html", "app.component.ts", "app.module.ts", "assets", "environments", "browserslist", "favicon.ico", and "index.html".
- OPEN EDITORS**: Three tabs are open: "app.component.html", "app.component.css", and "app.component.ts".
- app.component.html** content (Line numbers 1-13):

```
<h1>Perulangan Ng For</h1>
<h2>Todo LIst</h2>
<ul *ngFor="let item of todos; let i = index">
  <li
    [ngClass]="{
      | done: item.status
    }"
  >
    {{ i }}:{{ item.todo }} | {{ item.createdAt | date: "dd/MM/yy" }}
  </li>
</ul>
```
- Bottom status bar**: Shows "Ln 13, Col 1" and other settings like "Spaces: 2", "UTF-8", "LF", "HTML", "Prettier: ✓".

Pada kode program diatas kita menambahkan pipes dengan menambahkan sintaks | date : 'dd/MM/yy'

Selanjutnya silahkan buka browser dan hasilnya tanggal yang sebelumnya tidak terformat rapi sudah berubah menjadi format d/m/y yang sesuai dengan user di indonesia.

