



Research & Development Team

Angular 8

www.pnpsw.com

sommaik@pnpsw.com

081-754-4663

Lineid : sommaik

Software

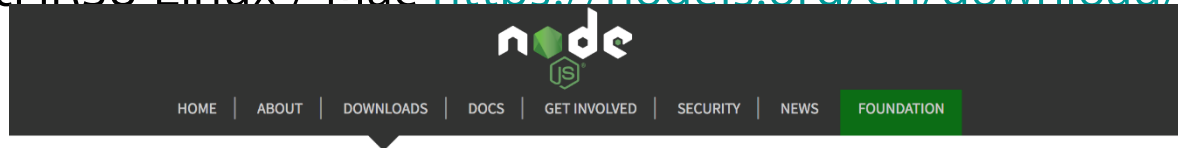
INSTALLATION

การติดตั้ง

NODE.JS

การติดตั้ง Node.js




- สำหรับ Windows <https://nodejs.org/en/download/current/>
- สำหรับ Linux / Mac <https://nodejs.org/en/download/package-manager/>



Downloads

Latest Current Version: 10.1.0 (includes npm 5.6.0)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS Recommended For Most Users	Current Latest Features	
 Windows Installer <small>node-v10.1.0-x86.msi</small>	 macOS Installer <small>node-v10.1.0.pkg</small>	 Source Code <small>node-v10.1.0.tar.gz</small>

Windows Installer (.msi)
 Windows Binary (.zip)
 macOS Installer (.pkg)
 macOS Binary (.tar.gz)
 Linux Binaries (x64)
 Linux Binaries (ARM)
 Source Code

32-bit	64-bit
32-bit	64-bit
64-bit	
64-bit	
64-bit	
ARMv6	ARMv7
ARMv8	
node-v10.1.0.tar.gz	

การตรวจสอบหลังการติดตั้ง node.js

- เปิด command line. ขึ้นมาแล้ว พิมพ์คำสั่งด้านล่างลงไป

```
node --version  
v8.11.2  
Sommais-MacBook-Pro:~ sommaik$ node --version  
v8.11.2  
Sommais-MacBook-Pro:~ sommaik$
```

การติดตั้ง

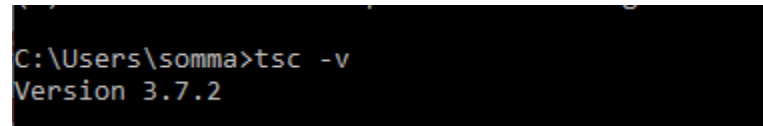
TYPESCRIPT

การติดตั้ง Typescript

เปิด terminal / cmd แล้วพิมพ์คำสั่ง เพื่อทำการติดตั้ง

```
npm install -g typescript
```

ตรวจสอบหลังติดตั้ง ใช้คำสั่งดังนี้



```
C:\Users\somma>tsc -v  
Version 3.7.2
```

การติดตั้ง

ANGULAR CLI

การติดตั้ง Angular CLI

ติดตั้ง Angular CLI โดยการพิมพ์คำสั่งเหล่านี้ใน command line

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

เสร็จแล้วให้ตรวจสอบการติดตั้งด้วยคำสั่งดังนี้

```
ng help
```

```
Sommais-MacBook-Pro:~ sommaik$ ng help
Available Commands:
  add Add support for a library to your project.
  new Creates a new directory and a new Angular app
  generate Generates and/or modifies files based on
  update Updates your application and its dependencies
  build Builds your app and places it into the output directory
  serve Builds and serves your app, rebuilding on file changes
  test Run unit tests in existing project.
  e2e Run e2e tests in existing project.
  lint Lints code in existing project.
```

การติดตั้ง

VISUAL STUDIO CODE

การติดตั้ง VSC

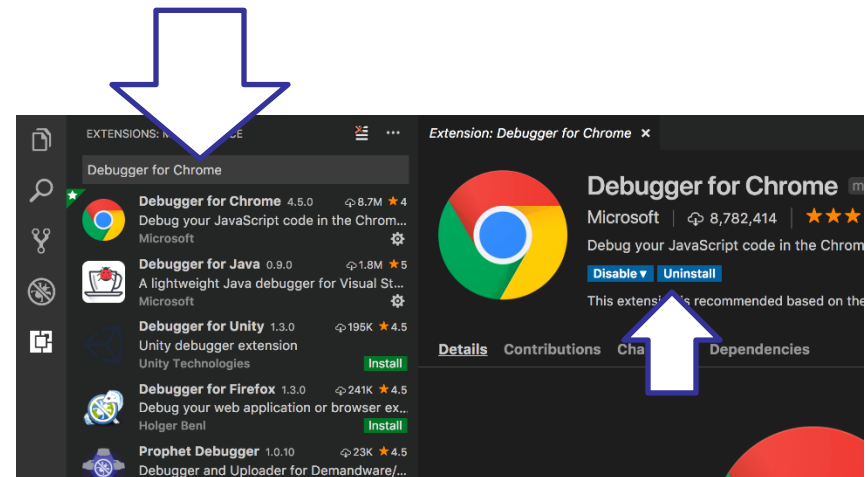
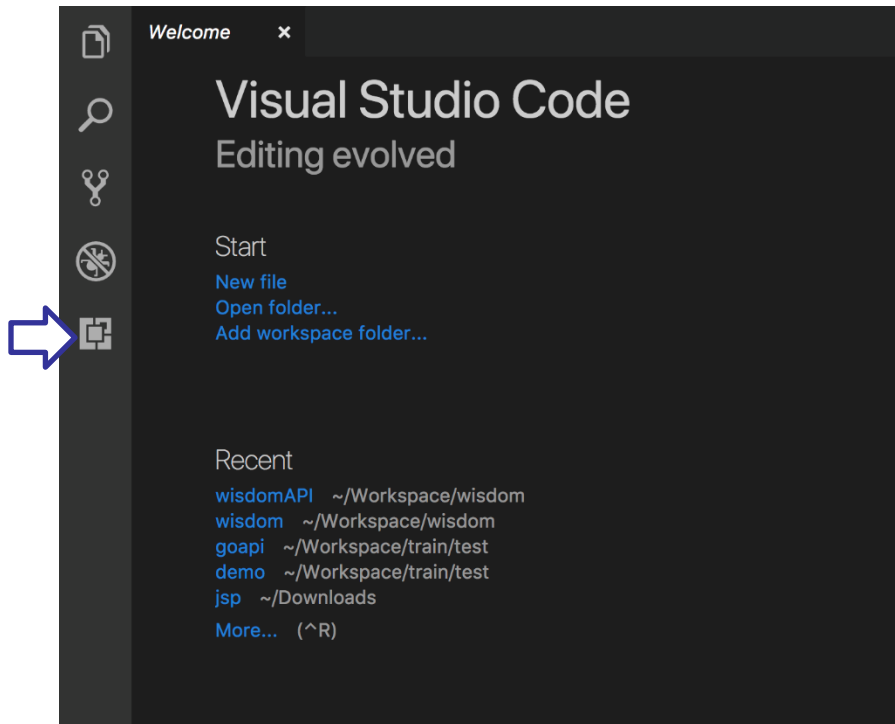
เข้าไปที่ website <https://code.visualstudio.com/>

เลือก download สำหรับ windows (stable)

The screenshot shows the Visual Studio Code website. The main heading is "Code editing. Redefined." with the subtext "Free. Open source. Runs everywhere." Below this, there's a "Download for Mac" section with a dropdown menu. The "Stable Build" is selected. To the right, there's a table showing download links for different operating systems and architectures. The "Windows x64" row is highlighted, showing links for "Installer" and "zip" files. The "Linux x64" row also shows links for ".deb", ".rpm", and ".tar.gz" files. On the right side of the image, there's a preview of the Visual Studio Code interface showing a code editor with a TypeScript file named "www.ts" and an "EXTENSIONS" sidebar on the left.

	Stable	Insiders
macOS	Package ↓	↓
Windows x64 32 bit versions	Installer ↓ zip ↓	↓ ↓
Linux x64 32 bit versions	.deb ↓ .rpm ↓ .tar.gz ↓	↓ ↓ ↓

Install Extensions



Install Extensions

Visual Studio Code Extensions

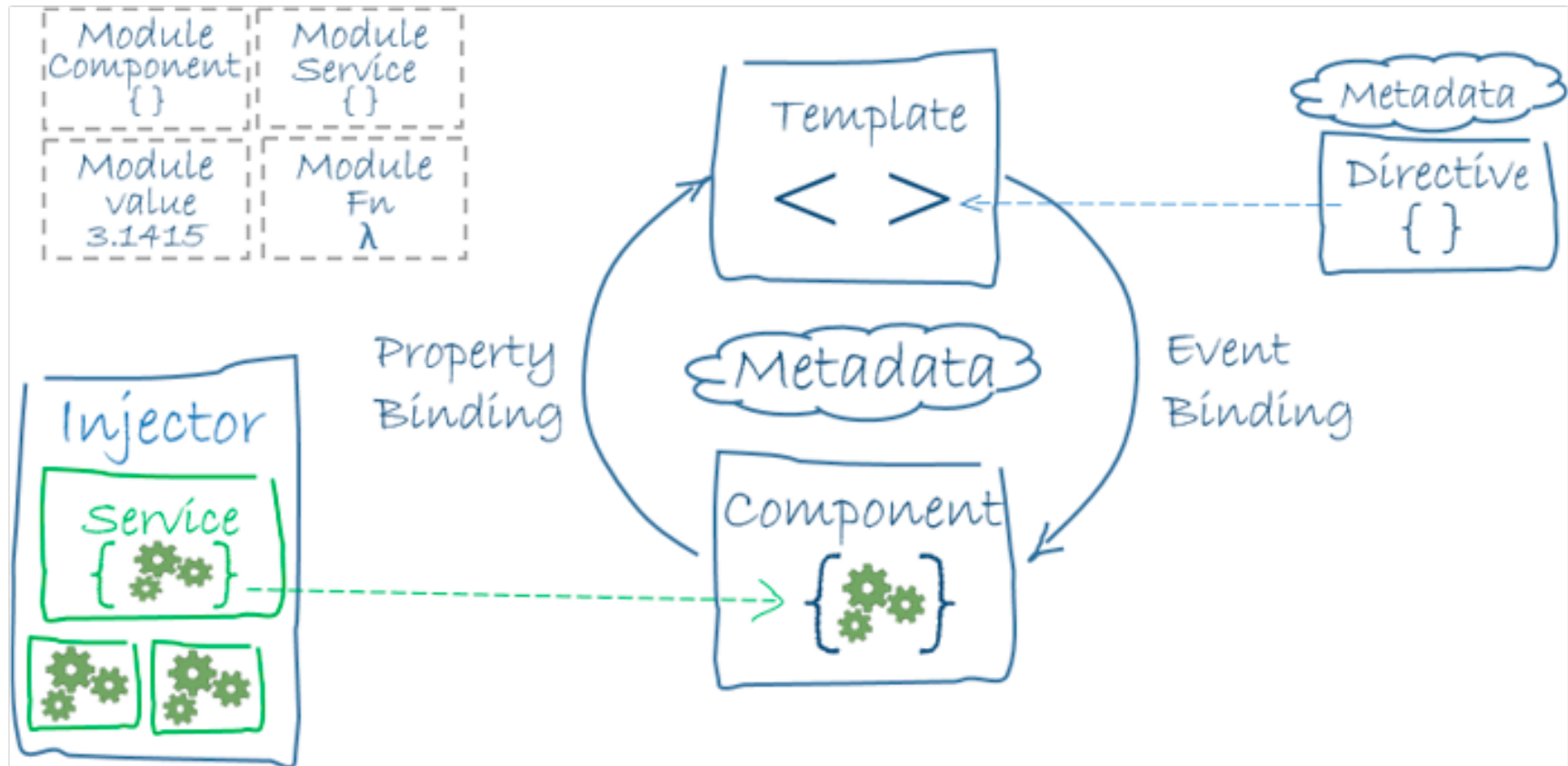
Angular Essentials

Debugger For Chrome

Introduction to

ANGULAR

Angular Architecture Overview



Architecture Overview #2

- Basics of Typescript
- Components, Bootstrap, and the DOM
- Directives and pipes
- Data binding
- Dependency Injection
- Services and other business logic
- Data Persistence
- Routing

พื้นฐาน

TYPESCRIPT

Basics of Typescript

- JavaScript that scale
- Starts and ends with JavaScript
- Strong tools for large apps
- State of the art JavaScript

Basics of Typescript #2

Create file app.ts

```
var message:string = "Hello World"  
console.log(message)
```

Compile

```
tsc app.ts
```

Run

```
node app.js
```

Output

```
Hello World
```

TypeScript — Keywords

break	as	any	switch
case	if	throw	else
var	number	string	get
module	type	instanceof	typeof
public	private	enum	export
finally	for	while	void
null	super	this	new
in	return	true	false
any	extends	static	let
package	implements	interface	function
new	try	yield	const
continue	do	catch	

TypeScript and OOP

```
class Greeting {  
    greet():void {  
        console.log("Hello World!!!")  
    }  
}
```

```
var obj = new Greeting();  
obj.greet();
```

Built-in types

- number
- string
- Boolean
- void
- null
- undefined
- any

Variable Declaration

var [identifier] : [type] = value ;

Ex: var name:string = 'my name is angular.io';

var [identifier] : [type];

Ex: var name:string;

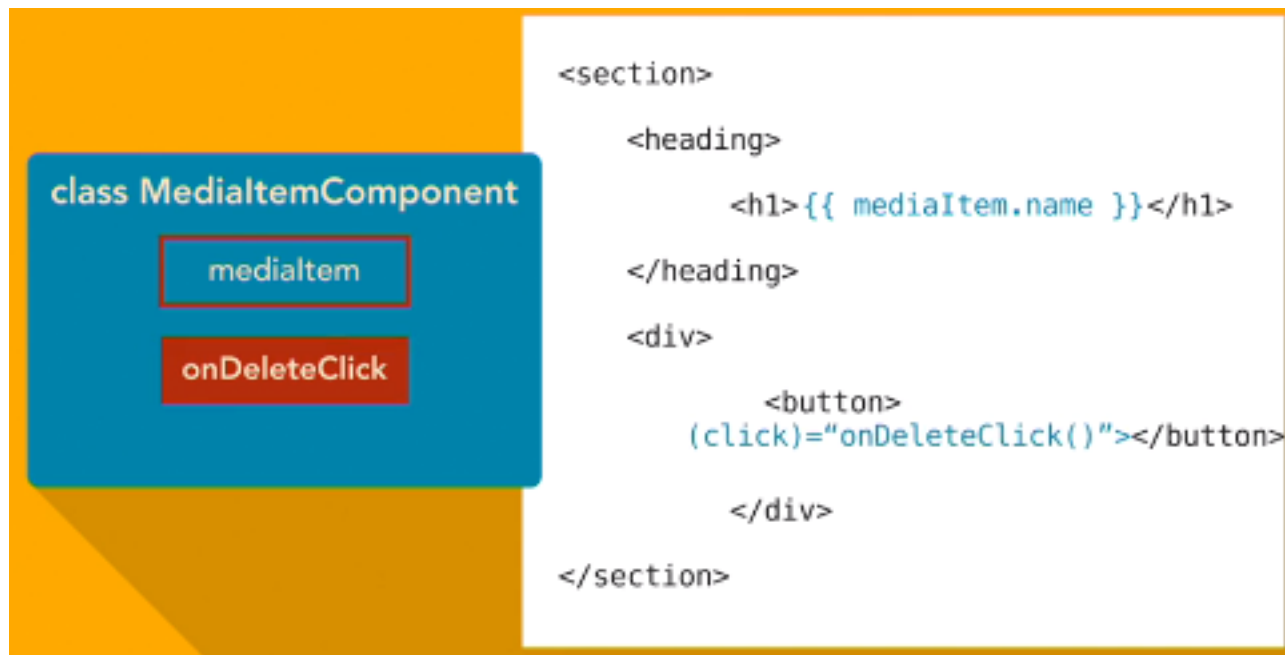
var [identifier] = value ;

Ex: var name = 'my name is angular.io';

var [identifier] ;

Ex: var name;

Components, Bootstrap, and the DOM



The diagram illustrates the relationship between a JavaScript class and its rendered HTML structure. On the left, a blue box represents the `MediaItemComponent` class, which contains two properties: `mediaItem` and `onDeleteClick`. On the right, a white box shows the corresponding HTML structure, which is a `<section>` element containing a heading, a heading element, a `<div>` element, and a button element. The button element has a `click` event that calls the `onDeleteClick` method.

```
class MediaItemComponent
  mediaItem
  onDeleteClick
```

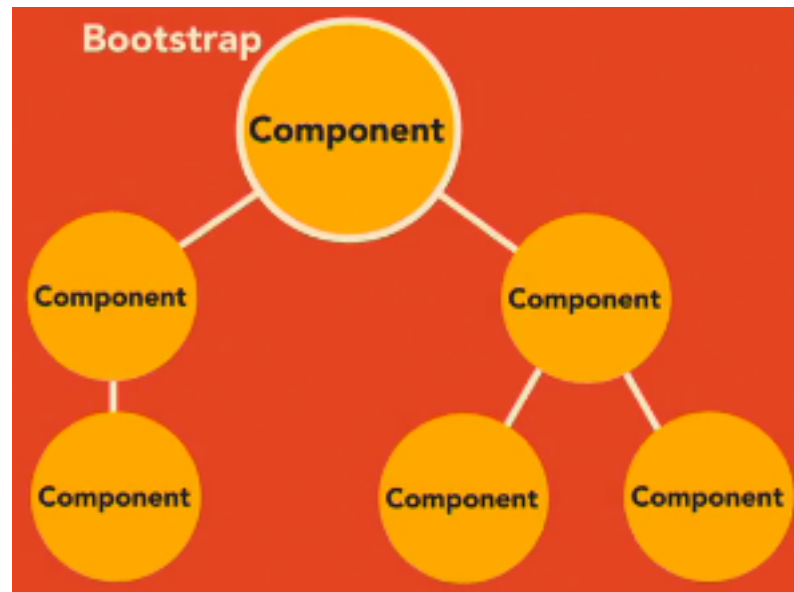
```
<section>
  <heading>
    <h1>{{ mediaItem.name }}</h1>
  </heading>
  <div>
    <button>
      (click)="onDeleteClick()"</button>
    </div>
</section>
```


Components, Bootstrap, and the DOM #2

The diagram illustrates the relationship between component classes and their HTML rendering. On the left, a stack of three colored boxes represents the component hierarchy: a yellow box at the top, an orange box in the middle, and a red box at the bottom. The orange box contains the text 'class MedialtemComponent'. The red box contains the text 'class ViewRatingComponent'. Inside the red box, there is a grey box with the text 'selector: 'view-rating''. To the right of this stack, a code block shows the corresponding HTML structure. The code is as follows:

```
<section>
  <heading>
    <h1>{{ mediaItem.name }}</h1>
  </heading>
  <div>
    <button>
      (click)="onDeleteClick()"</button>
    </div>
    <view-rating></view-rating>
  </section>
```

Components, Bootstrap, and the DOM #3



command line interface with

ANGULAR CLI

New Project

new project

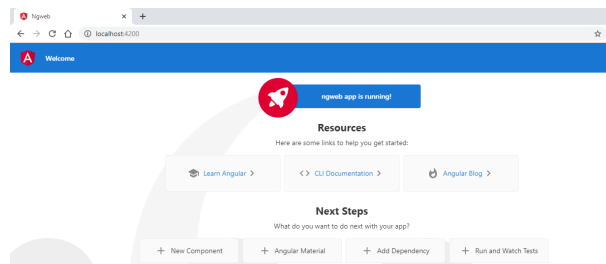
```
ng new ngweb --routing --style=css
```

fixed vulnerabilities (for angular 6)

```
npm i karma@3.0.0 --save
```

Test

เข้าไปที่ folder ชื่อเดียวกับ project ที่สร้างไว้ ด้วยคำสั่ง `cd ngweb`
สั่ง start dev server ด้วยคำสั่ง `ng serve` หรือ `npm start`
เปิด browser แล้วเข้า url <http://localhost:4200>



app.module.ts

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

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

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

app.module.ts

- 1. declarations
 - ประกาศ component ที่ต้องการใช้ในระบบ
- 2. imports
 - ประกาศ module ที่ต้องการใช้ในระบบ
- 3. providers
 - ประกาศ service ที่ต้องการใช้ในระบบ

Environment file

- เอาไว้เก็บค่าตัวแปรที่ต้องการใช้งานที่แปรผันตาม environment เช่น prod, dev ต้องการให้ใช้ api คนละตัวกัน เป็นต้น
- ใน file เก็บข้อมูลอยู่ในรูปแบบของ json จึงสามารถเรียกใช้งานใน program ได้ดังนี้
- import environment เข้าไปใน class ที่ต้องการใช้งาน

```
import {environment} from '../environments/environment';
```

- เรียกใช้ผ่านตัวแปรชื่อ environment เช่น

```
console.log(environment.production);
```

ตัวอย่างการเรียกใช้ environment

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

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

  constructor() {
    console.log(environment.production);
  }
}
```


Assets

- เป็นที่เก็บรวบรวม file ที่เอาไว้ share ร่วมกันใน project เช่น file รูป, css เป็นต้น
- เก็บ file ไว้ภายใต้ folder assets
- ตอนเรียกใช้สามารถเรียกได้ดังนี้
- ตัวอย่างเราเก็บ file logo.jpg ไว้ใน folder app/assets
- เราสามารถเรียกใช้ได้แบบนี้

```

```

Component

เพื่อเอาไว้สร้าง component โดยมีรูปแบบคำสั่งดังนี้

- ng generate component `<component_name>`
- ng g component `<component_name>`
- ng g c `<component_name>`
- npm run ng g c `<component_name>`

ตัวอย่าง

- ng g component home

สามารถระบุ path ได้ดังนี้

- ng g component page/home

Component

- HTML (TEMPLATE)
- CSS
- Javascript / TypeScript (Script)

```
import { Component, OnInit } from '@angular/core';  
  
@Component({  
  selector: 'app-home',  
  templateUrl: './home.component.html',  
  styleUrls: ['./home.component.css']  
})  
  
export class HomeComponent implements OnInit {  
  constructor() {}  
  
  ngOnInit() {  
  }  
}
```

TEMPLATE

- `{{}}` : RENDERING
- `[]`: BINDING PROPERTIES
- `()`: HANDLING EVENTS
- `[()]`: TWO-WAY DATA BINDING
- `*`: THE ASTERISK
- `#` : REFERENCE

{{}} : RENDERING

- ใช้สำหรับนำค่าจาก script มาแสดงผลใน html

Component Script

```
title='appworks!';
```

Template Html

```
<h1>{{title}}</h1>
```

[]: BINDING PROPERTIES

- ใช้สำหรับเชื่อมโยงค่าตัวแปรมาจาก component

Component Script

```
url = "http://www.google.com";
```

Template Html

```
<a [href]="url">goto url</a>
```

@Input

Component Script #ต้นทาง

```
@Input()name;
```

Template Html#ต้นทาง

```
<p>
```

```
homeworks!{{name}}
```

```
</p>
```

Component Script#ปลายทาง

```
appName="HomeApp";
```

Template Html#ปลายทาง

```
<app-home[name]="appName"></
```

```
app-home>
```

- ใช้สำหรับเชื่อมโยงค่าตัวแปรมาจากอีก component

() : HANDLING EVENTS

- ใช้สำหรับจัดการกับ Event ของ user

Component Script

```
onBtnClick(){  
    console.log("Hello World");  
}
```

Template Html

```
<button class="btn" (click)="onBtnClick();">Click</button>
```


[()]: TWO-WAY DATA BINDING

- เป็น directive ที่ทำ two-way databinding
- วิธีการใช้ ngModel มี 2 แบบคือ
 - `<input [(ngModel)]="code" />`
 - `<input name="code" ngModel />`
- ก่อนใช้ ngModel ต้องทำการ import module ดังนี้เข้าไปที่ app.module.ts
 - FormsModule
 - ReactiveFormsModule

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';

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

[()]: TWO-WAY DATA BINDING

- ใช้สำหรับเชื่อมโยงค่าตัวแปรกับ user input

Component Script

```
appName = "Home App";
```

Template Html

```
<input type="text" [(ngModel)]="appName"/>
```

Form [TS]

```
import { NgForm } from '@angular/forms';

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

  ngOnInit() {}

  onSubmit(simpleForm: NgForm) {
    if (simpleForm.valid) {
      console.log('post data to server');
    } else {
      console.log('show error');
    }
  }
}
```

Form [html]

```
<form #simpleForm="ngForm" (submit)="onSubmit(simpleForm)">
  <input name="first" ngModel #first="ngModel" required minlength="5" />
  <input name="last" ngModel #last="ngModel" required />
  <button>Submit</button>
</form>
```

```
<h1>{{ simpleForm.value | json }}</h1>
<h1>{{ first.errors | json }}</h1>
<h1>{{ first.invalid }}</h1>
```

Form Group [TS]

```
export class FormGroupComponent implements OnInit {  
  constructor(private fb: FormBuilder) {}  
  
  simpleForm = this.fb.group({  
    first: ['', [Validators.required, Validators.minLength(5)]],  
    last: ['', [Validators.required]]  
  });  
  
  ngOnInit() {}  
  
  onSubmit() {  
    if (this.simpleForm.valid) {  
      console.log('post data to server');  
    } else {  
      console.log('show error');  
    }  
  }  
}
```

Form Group [html]

```
<form [formGroup]="simpleForm" (submit)="onSubmit()">
  <input formControlName="first" />
  <input formControlName="last" />
  <button>Submit</button>
</form>
```

```
<h1>{{ simpleForm.value | json }}</h1>
<h1>{{ simpleForm.get('first').errors | json }}</h1>
<h1>{{ simpleForm.get('first').invalid | json }}</h1>
```

Control status CSS classes

- .ng-valid
- .ng-invalid
- .ng-pending
- .ng-pristine
- .ng-dirty
- .ng-untouched
- .ng-touched

@Output

Component Script #ດ້ນາກ

```
url = 'www.google.com';
```

```
@Output()
```

```
btnGoToClick: EventEmitter<string> = new  
EventEmitter<string>();
```

```
onGotoClick() {  
  this.btnGoToClick.emit(this.url);  
}
```

Template Html #ດ້ນາກ

```
<input type="text" [(ngModel)]="url">  
<button (click)="onGotoClick();">Goto Url</button>
```

Component Script #ປາຍກາກ

```
onGotoBtnClick(url) {  
  console.log(url);  
}
```

Template Html #ປາຍກາກ

```
<app-home  
(btnGoToClick)="onGotoBtnClick($event)"></app-  
home>
```

Class

เพื่อใช้สร้าง class โดยมีรูปแบบคำสั่งดังนี้

- ng g class <class_name>
- ng generate class <class_name>
- npm run ng g class <class_name>

ตัวอย่าง

- ng g class user

Directive

เพื่อใช้สร้าง directive โดยมีรูปแบบคำสั่งดังนี้

- ng g directive <directive_name>
- ng generate directive <directive_name>

ตัวอย่าง

- ng g directive highlight

Built-in Directives

- *ngIf
- *ngFor
- [ngSwitch], [ngSwitchCase], [ngSwitchDefault]
- [ngClass]

Built-in directives

ngIf

```
<section*ngIf="showSection">
```

ngFor

```
<li*ngFor="let item of list">
```

ngClass

```
<div[ngClass]="{ 'active': isActive,  
'disabled': isDisabled }">
```

Built-in directives #2

```
<div [ngSwitch]="conditionExpression">
```

```
<ng-template [ngSwitchCase]="case1Exp">One</ng-template>
```

```
<ng-template ngSwitchCase="case2LiteralString">Two</ng-template>
```

```
<ng-template ngSwitchDefault>Three</ng-template>
```

```
</div>
```

```
//  
export class AppComponent {  
  title = 'app';  
  conditionExpression = "A";  
  case1Exp = "A";  
}
```

Custom Directive [TS]

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

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {
  constructor(private el: ElementRef) {}

  @Input('appHighlight') color: string;

  @HostListener('mouseover') onmouseover() {
    this.el.nativeElement.style.backgroundColor = this.color;
  }

  @HostListener('mouseleave') onmouseleave() {
    this.el.nativeElement.style.backgroundColor = '';
  }
}
```

Pipe

เพื่อใช้สร้าง pipe โดยมีรูปแบบคำสั่งดังนี้

- `ng g pipe <pipe_name>`
- `ng generate pipe <pipe_name>`

ตัวอย่าง

- `ng g pipe pipe/trim-credit-card`

Built-in PIPES

- Pipes transform displayed values within a template.

Built-In

DecimalPipe = number[:format]

Script

```
price=12300.5;
```

Html

```
{{price|number:'1.2-3'}}
```

Built-in PIPES

Built-In

LowerCasePipe = lowercase

Html

```
{{item|lowercase}}
```

Built-In

UpperCasePipe = uppercase

Html

```
{{item | uppercase}}
```

Built-in PIPES

Built-In

DatePipe = date[:format]

Script

```
currentDate = new Date();
```

Html

```
{{currentDate|date:'dd/MM/yH:mm:ss'}}
```

Service

```
import { HighlightDirective } from '../directive/highlight.directive';
import { TrimCreditCardPipe } from '../pipe/trim-credit-card.pipe';
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    HighlightDirective,
    TrimCreditCardPipe
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
```

Service

เพื่อใช้สร้าง service โดยมีรูปแบบคำสั่งดังนี้

- ng g service **<class_name>**
- ng generate service **<class_name>**

ตัวอย่าง

- ng g service **user**

SERVICE

- ng g service /service/home

```
installing service
create src\app\home.service.spec.ts
create src\app\home.service.ts
WARNING Service is generated but not provided, it must be provided to be used
```

import library

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

Inject http object

```
constructor(private http: HttpClient){}
```

Service get data from server

Service Script

```
loadItem():Observable<any>{  
  returnthis.http.get('change to your url');  
}
```

Service get data from server

Component Script

import

```
import{HomeService}from'../home.service'
```

set providers

```
@Component({
```

```
...
```

```
  providers:[HomeService]
```

```
...
```

```
})
```

inject

```
constructor(privatehomeService:HomeService){}
```


Component call get data from server

```
onBtnClick(){  
  this.homeService.loadItem()  
    .subscribe(  
      datas=>{  
        this.items=datas;  
      },  
      err=>{  
        console.log(err);  
      });  
}
```

POST

```
addData(body: Home): Observable<HomeResp> {  
    const httpOptions = {  
        headers: new HttpHeaders({  
            'Content-Type': 'application/xml'  
        })  
    };  
    return this.http.post(this.dataUrl, body, options);  
}
```

Interface

เพื่อใช้สร้าง interface โดยมีรูปแบบคำสั่งดังนี้

- ng g interface **<class_name>**
- ng generate interface **<class_name>**

ตัวอย่าง

- ng g interface **user**

Guard

เพื่อใช้สร้าง class โดยมีรูปแบบคำสั่งดังนี้

- ng g guard **<class_name>**
- ng generate guard **<class_name>**

ตัวอย่าง

- ng g guard **auth**

Enum

เพื่อใช้สร้าง enum โดยมีรูปแบบคำสั่งดังนี้

- ng g enum **<class_name>**
- ng generate enum **<class_name>**

ตัวอย่าง

- ng g enum **title**

Module

เพื่อใช้สร้าง module โดยมีรูปแบบคำสั่งดังนี้

- `ng g module <class_name>`
- `ng generate module <class_name>`

ตัวอย่าง

- `ng g module user`

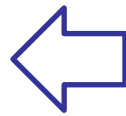
Routing

create file app-routing.module.ts

```
import{HomeComponent}from'./home/home.component';  
exportconstappRoutes=[  
  {path:'home',component:HomeComponent}  
];
```

app.component.html

```
<h1>  
{{title}}  
</h1>  
<router-outlet></router-outlet>
```



ต้องมี tag router-outlet ใน
html ของ root component

Routes

```
const routes: Routes = [  
  {  
    path: 'one',  
    loadChildren: './one/one.module#OneModule' ← Lazy load module  
  },  
  {  
    path: 'two',  
    canActivate: [AuthGuard], ← Guard  
    children: [  
      { ← Child Route  
        path: '',  
        component: FormComponent  
      }  
    ],  
  },  
  {  
    path: 'three',  
    redirectTo: 'one', ← Redirect route  
    pathMatch: 'full'  
  },  
  {  
    path: 'four/:id', ← Path with parameter  
    component: FormComponent  
  }  
];
```


Routing with parameter [TS]

```
constructor(private route: ActivatedRoute) {  
  this.route.params.subscribe( (params) => {  
    console.log(params);  
  });  
}
```

Navigate to other page

Import

```
import {Router} from '@angular/router';
```

Inject

```
constructor(private router: Router) {}
```

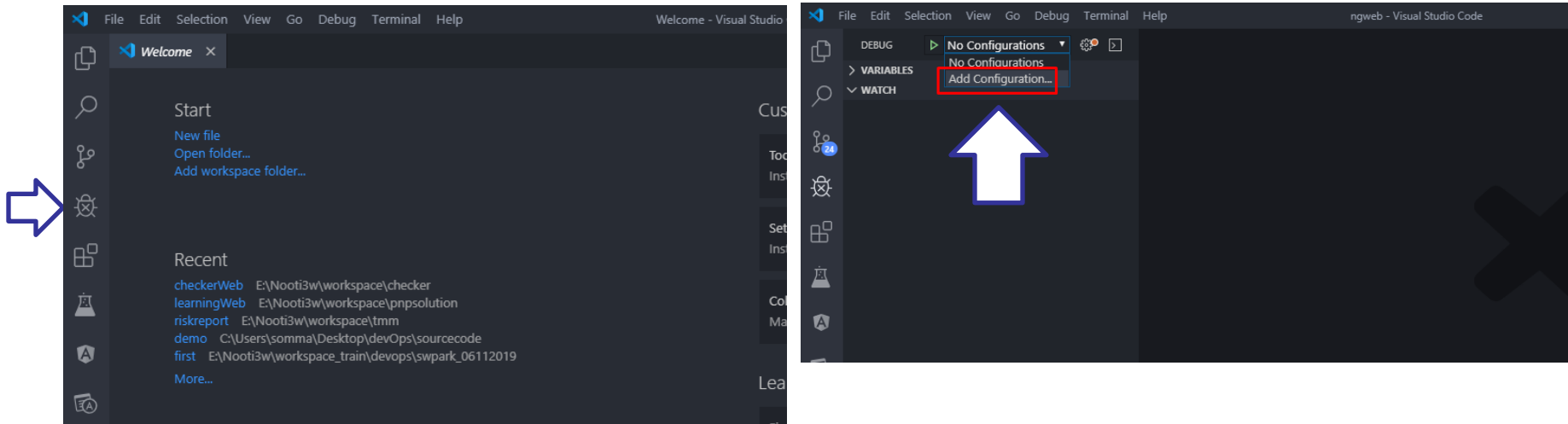
Navigate

```
this.router.navigate(['']);
```

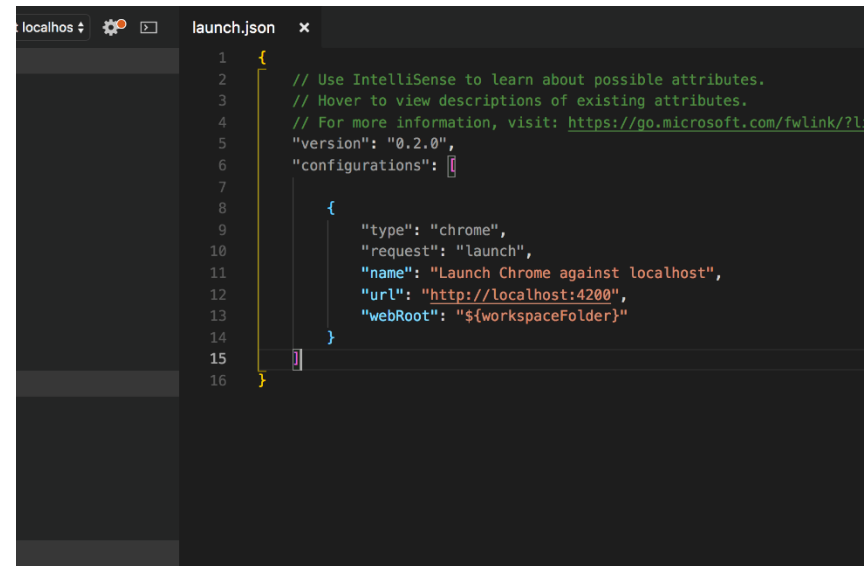
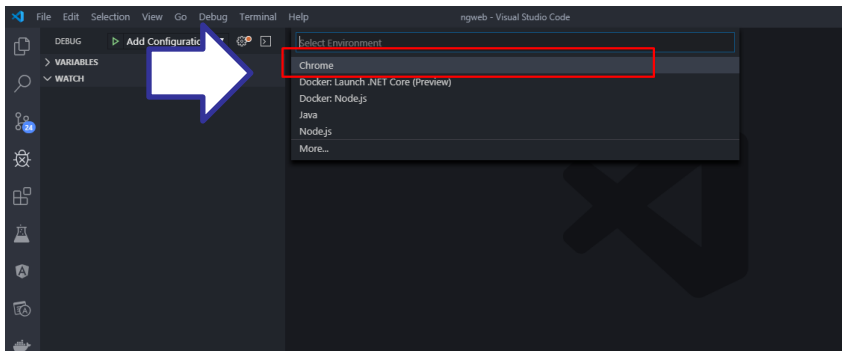
Debug angular with

DEBUGGER FOR CHROME

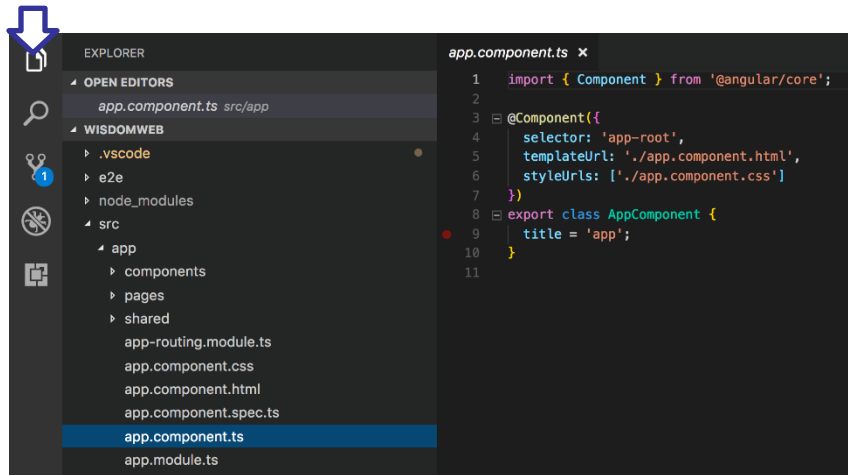
Add new debug configuration



Add new debug configuration

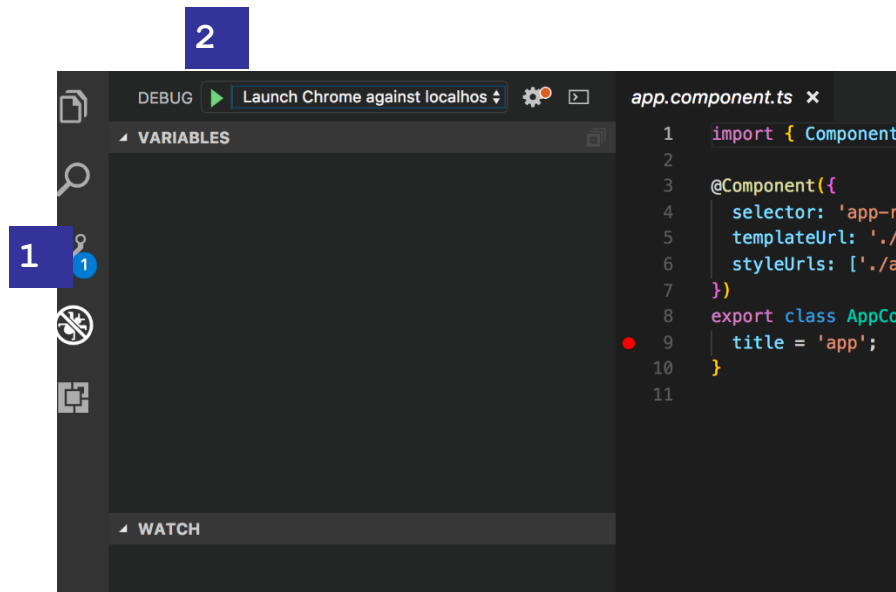


Debug step 1



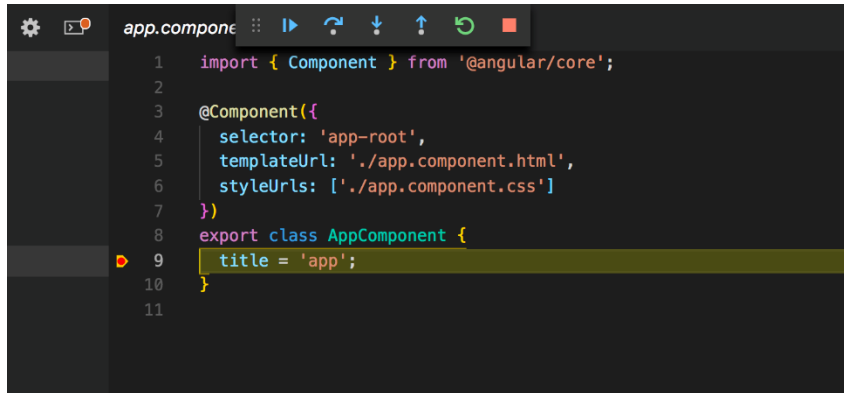
- เข้าสู่หน้าจอ explorer
- เลือก file ที่ต้องการ debug
- กดที่หน้าบรรทัดที่ต้องการให้เป็นจุดสีแดง

Debug step 2

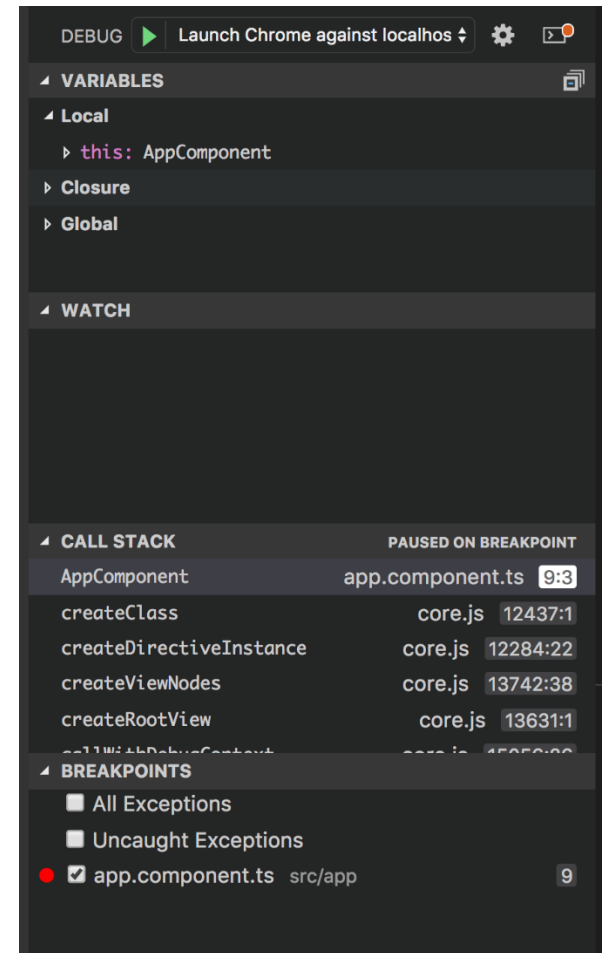


- [1] กดเข้าสู่ mode debug
- [2] กดปุ่มเพื่อเริ่มต้นการ debug
- *หมายเหตุ* ต้อง start angular ก่อนด้วยคำสั่ง ng serve

Debug step 3



```
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 = 'app';
10 }
11
```



DEBUG Launch Chrome against localhos

VARIABLES

- Local
 - this: AppComponent
- Closure
- Global

WATCH

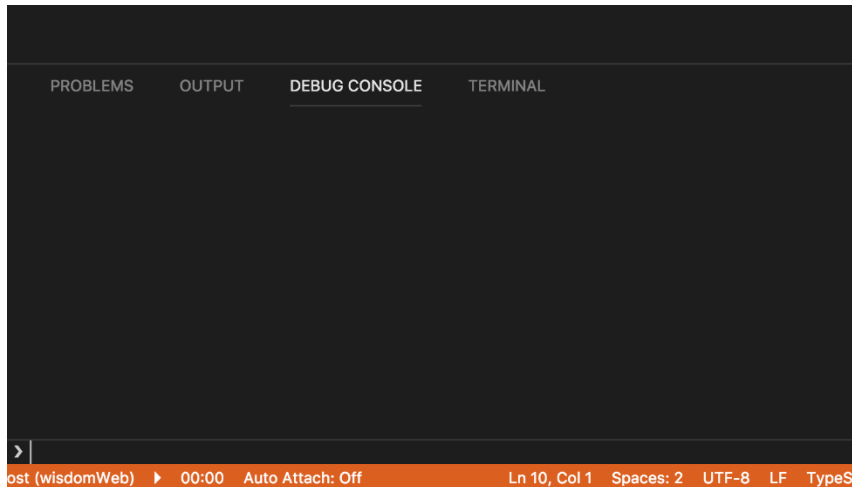
CALL STACK PAUSED ON BREAKPOINT

AppComponent	app.component.ts	9:3
createClass	core.js	12437:1
createDirectiveInstance	core.js	12284:22
createViewNodes	core.js	13742:38
createRootView	core.js	13631:1
callWithDebugContext	core.js	15250:26

BREAKPOINTS

- ☐ All Exceptions
- ☐ Uncaught Exceptions
- ☒ app.component.ts src/app 9

Debug Console



- สามารถส่งรับคำสั่งภายใน component ที่กำลัง debug อยู่ได้
- สามารถแก้ไขข้อมูลใน component. ได้

package project with

BUILD COMMAND

build command

- สั่ง build ด้วย angular cli ใช้คำสั่งดังนี้

`ng build --prod`

`--prod` หมายถึง production mode

- หลังจาก build แล้วจะได้ folder ชื่อ dist เกิดขึ้นมา แล้วในนั้นจะมีเป็น file ที่ทำการ build เป็นที่เรียบร้อยแล้ว เราสามารถนำ file นี้ไป deploy บน webserver ต่างๆ ได้เช่น nginx, apache, tomcat เป็นต้น
- `npm run ng build -- --prod`

Any questions?

