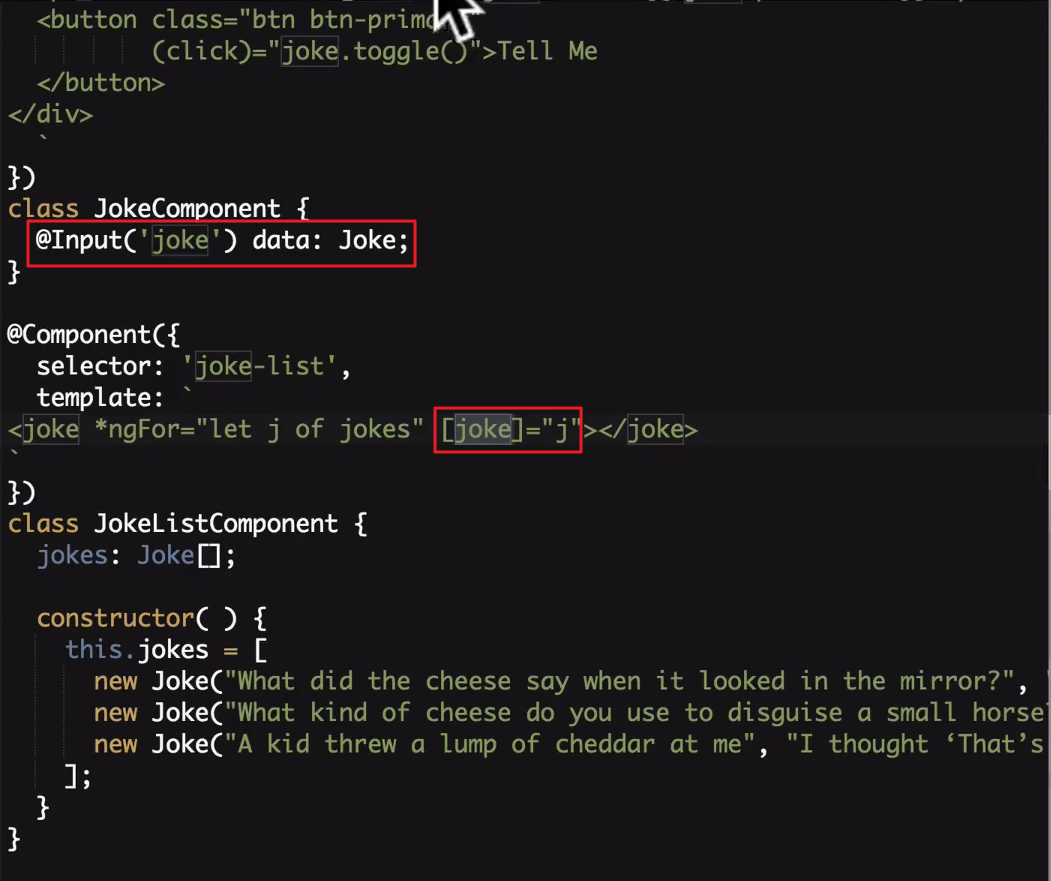
<https://codecraft.tv/courses/angular/>

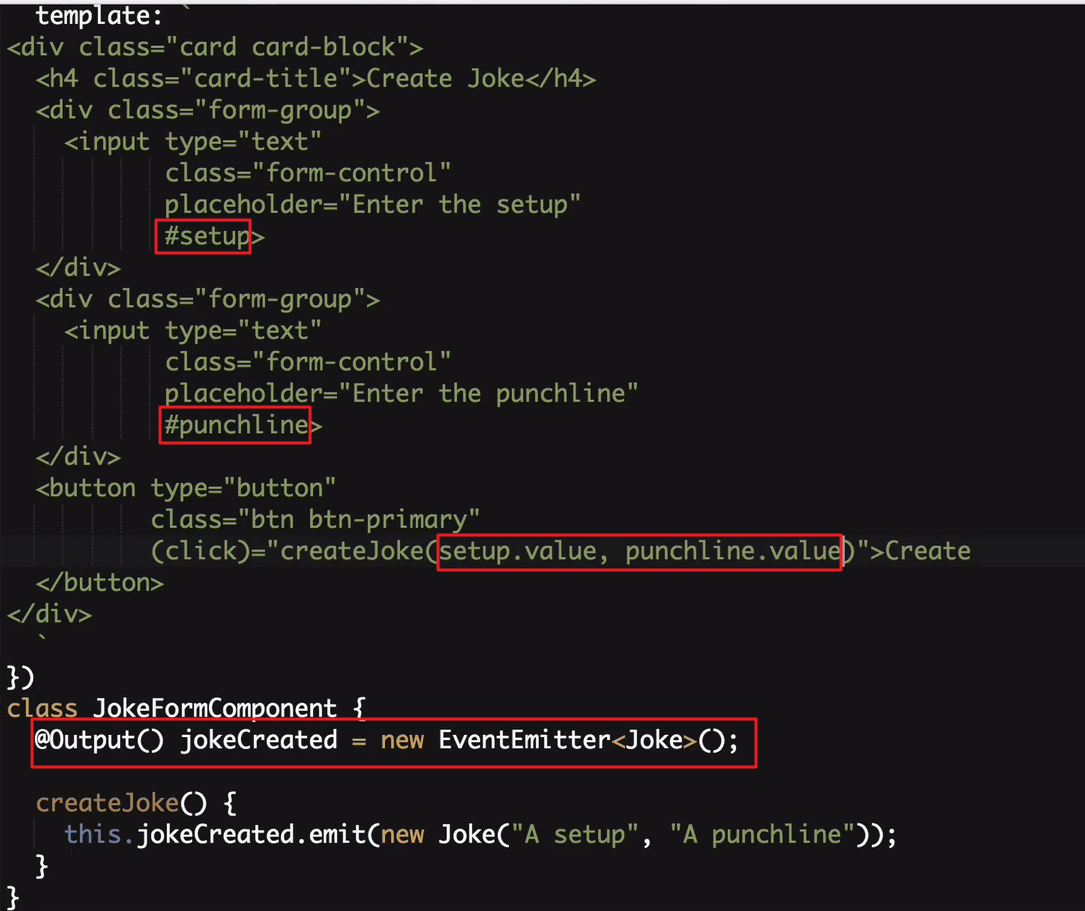
[https://github.com/codecraftpro/](https://github.com/codecraftpro/angular2-sample-code/blob/master/2.es6-typescript/10.class-interface/script.ts)

**QuickStart**

@Input

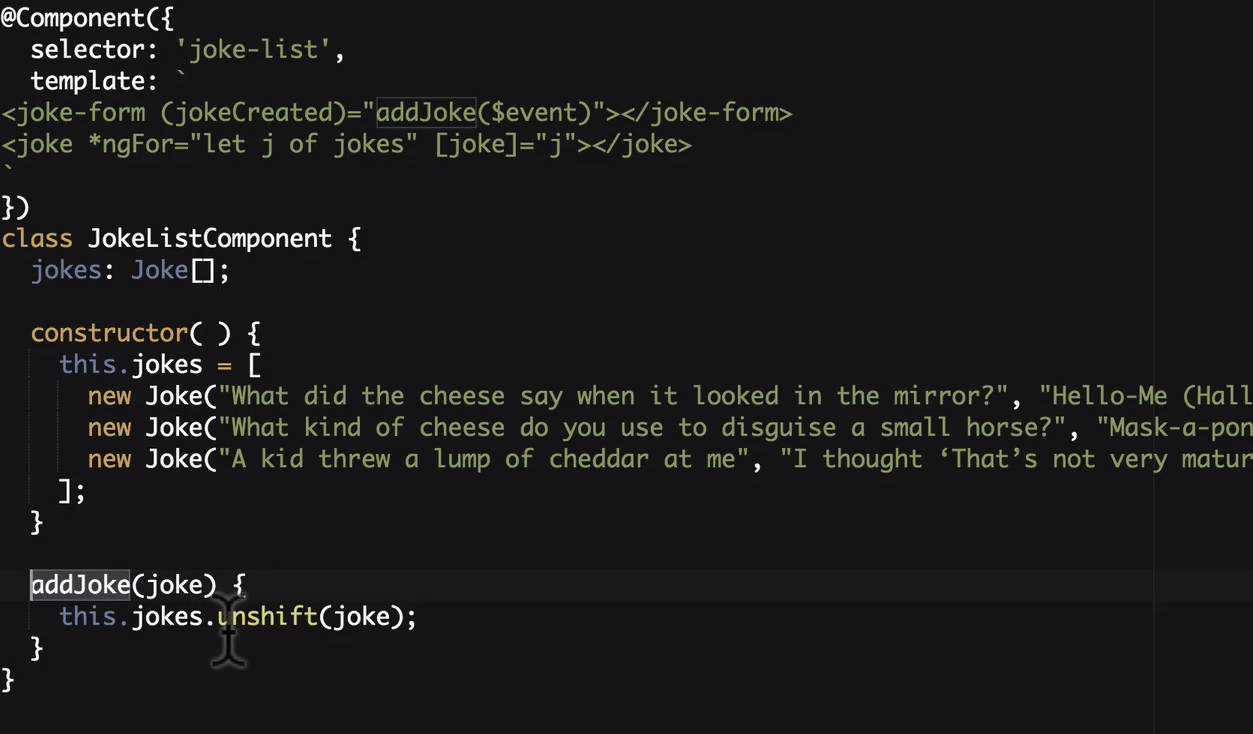


템플릿 안쪽에서 참조



EventEmitter로 컴포넌트간의 데이터 전송



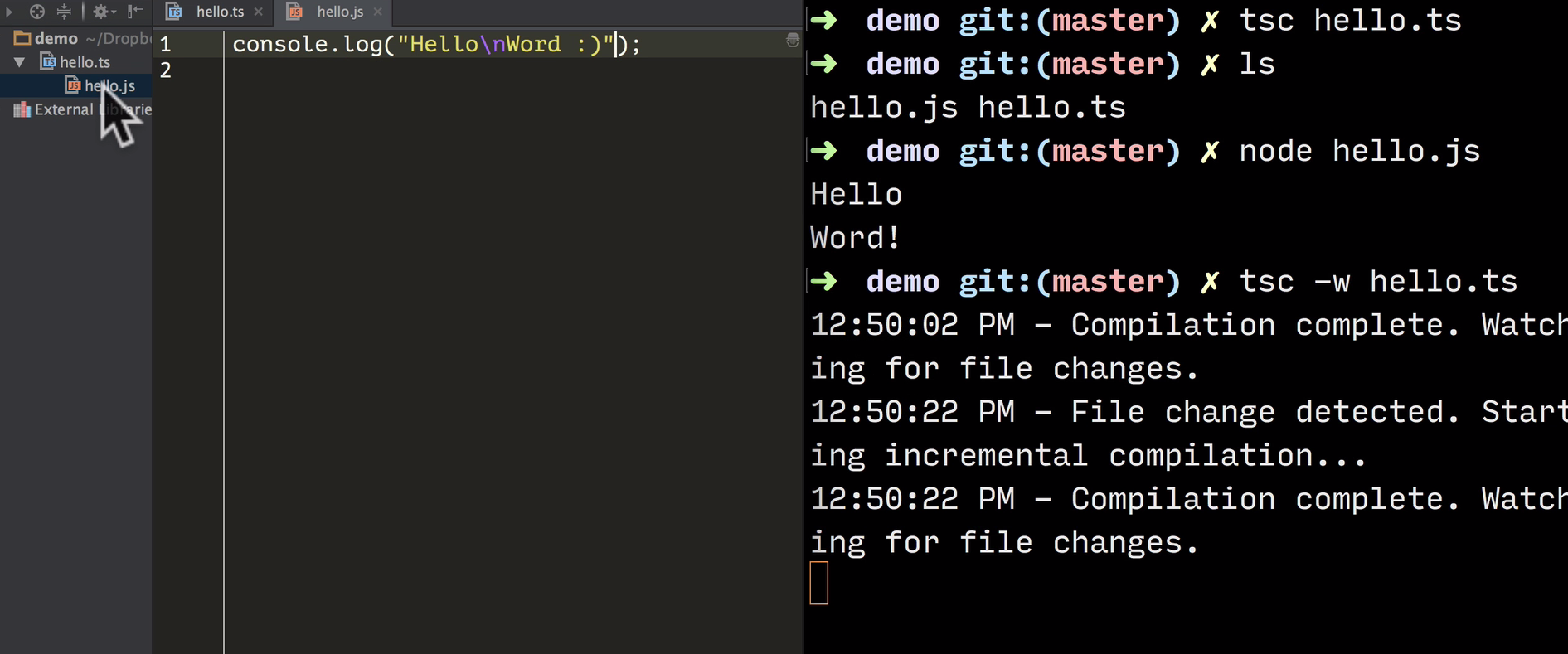


<http://plnkr.co/edit/b0F6Dhb40Hm5zfiamAix?p=preview>

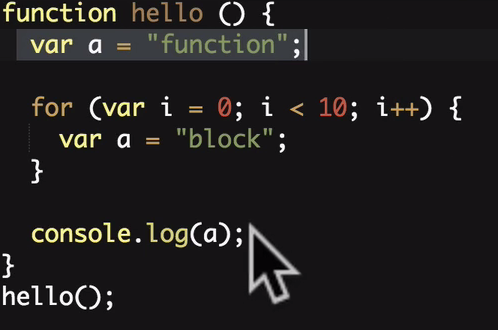
|  |
| --- |
| **import** {platformBrowserDynamic} from '@angular/platform-browser-dynamic'; **import** {Component, NgModule, Input, Output, EventEmitter} from '@angular/core'; **import** {BrowserModule} from '@angular/platform-browser';  **class** Joke {  **public** setup: string;  **public** punchline: string;  **public** hide: boolean;   constructor(setup: string, punchline: string) {  **this**.setup = setup;  **this**.punchline = punchline;  **this**.hide = **true**;  }   toggle() {  **this**.hide = !**this**.hide;  } }   @Component({  selector: 'joke-form',  template: ` <div class="card card-block">  <h4 class="card-title">Create Joke</h4>  <div class="form-group">  <input type="text"  class="form-control"  placeholder="Enter the setup"  #setup>  </div>  <div class="form-group">  <input type="text"  class="form-control"  placeholder="Enter the punchline"  #punchline>  </div>  <button type="button"  class="btn btn-primary"  (click)="createJoke(setup.value, punchline.value)">Create  </button> </div>  ` }) **class** JokeFormComponent {  @Output() jokeCreated = **new** EventEmitter<Joke>();   createJoke(setup: string, punchline: string) {  **this**.jokeCreated.emit(**new** Joke(setup, punchline));  } }  @Component({  selector: 'joke',  template: ` <div class="card card-block">  <h4 class="card-title">  {{data.setup}}  </h4>  <p class="card-text"  [hidden]="data.hide">{{data.punchline}}</p>  <a (click)="data.toggle()"  class="btn btn-warning">Tell Me  </a>  <a (click)="deleteItem()"  class="btn btn-danger">Delete  </a>  </div>  ` }) **class** JokeComponent {  @Input('joke') data: Joke;  @Output() jokeDeleted = **new** EventEmitter<Joke>();   deleteItem() {  **this**.jokeDeleted.emit(**this**.data);  } }  @Component({  selector: 'joke-list',  template: ` <joke-form (jokeCreated)="addJoke($event)"></joke-form> <joke \*ngFor="let j of jokes" [joke]="j" (jokeDeleted)="deleteJoke($event)"></joke>  ` }) **class** JokeListComponent {  jokes: Joke[];   constructor() {  **this**.jokes = [  **new** Joke("What did the cheese say when it looked in the mirror?", "Hello-me (Halloumi)"),  **new** Joke("What kind of cheese do you use to disguise a small horse?", "Mask-a-pony (Mascarpone)"),  **new** Joke("A kid threw a lump of cheddar at me", "I thought ‘That’s not very mature’"),  ];  }   addJoke(joke) {  **this**.jokes.unshift(joke);  }   deleteJoke(joke) {  **let** indexToDelete = **this**.jokes.indexOf(joke);  **if** (indexToDelete !== -1) {  **this**.jokes.splice(indexToDelete,1);  }  } }  @Component({  selector: 'app',  template: ` <joke-list></joke-list>  ` }) **class** AppComponent { }  @NgModule({  imports: [BrowserModule],  declarations: [  AppComponent,  JokeComponent,  JokeListComponent,  JokeFormComponent  ],  bootstrap: [AppComponent] }) **export class** AppModule { }  platformBrowserDynamic().bootstrapModule(AppModule); |

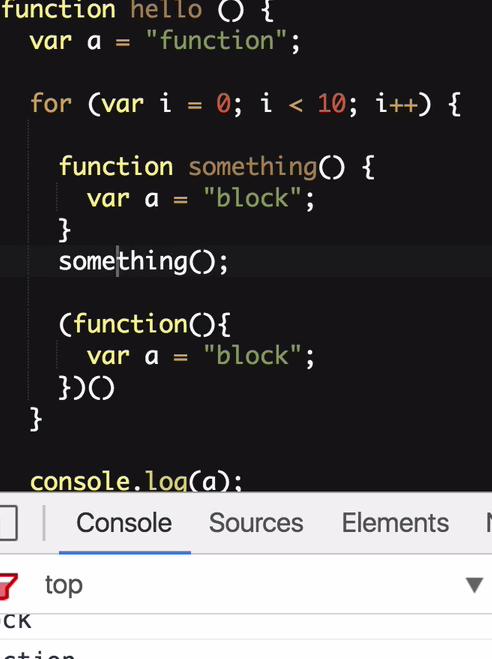
**E6 & TypeScript**

타입스크립트 컴파일 실행



**var**

되기때문에

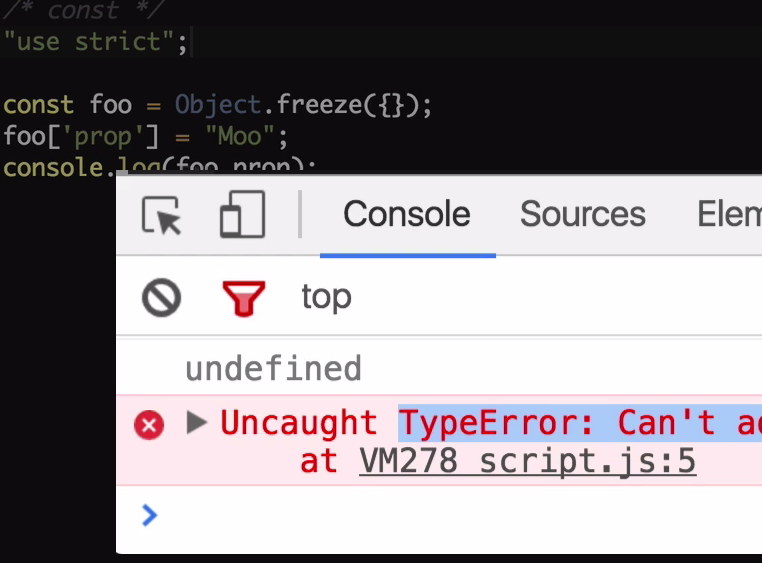
처럼 처리를 해야됐다.

let 은 안쪾에 세이프하게 해준다.

**const**

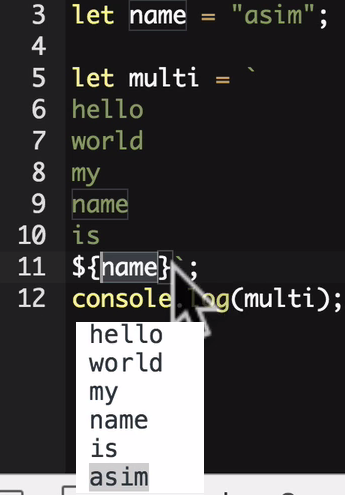
무조건 초기값이 있어야하며 값을 변경할수 없다.

블럭안에서만 효율성이 있다.



오브젝트 안쪽내용까지 다 수정못하게 하려면 Object.freeze로 초기값을 해도된다.

**template string**



**Fat Arrow Function**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | | |

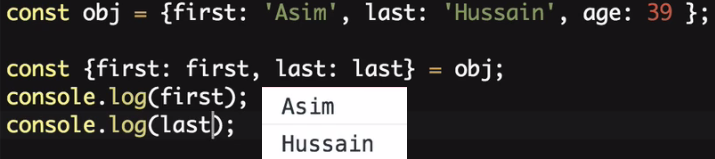
|  |  |
| --- | --- |
|  |  |

호출자의 기준으로 this가 지정되는것을 => 형식으로 쓴다면 그것또한 해제시켜준다 예를보자

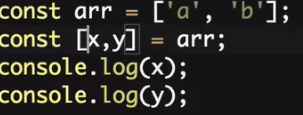
|  |  |
| --- | --- |
| 문제발생 | 기존처럼 문제해결 |
| => 로 문제해결 | |

**Destructuring**

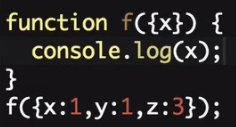
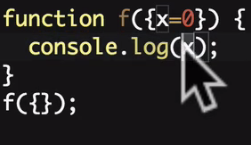
오브젝트 형을 바로 변수로받기







파라미터로 받기

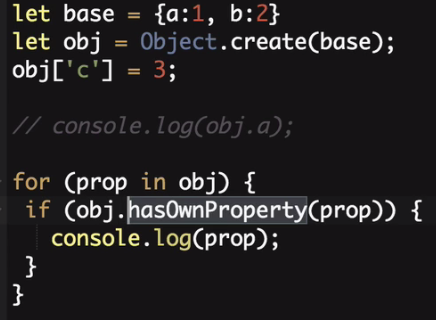
 

**For of**

기존

|  |  |
| --- | --- |
|  |  |
| of |  |

다른 Object와 프로퍼티 비교하기

 c 하나 나옴

**Map & Set**

|  |  |
| --- | --- |
|  |  |
| map.clear() , map.keys(), map.values(), map.entries() |  |

|  |  |
| --- | --- |
|  |  |
| set.values(), set.size, set.clear(), set.add() |  |

**Promises**

|  |  |
| --- | --- |
| 기존 |  |
|  |  |

**Class & Interface**

|  |  |
| --- | --- |
|  |  |

|  |
| --- |
| **interface** Human {  firstName: **string**;  lastName: **string**;  name?: Function;  isLate?(time: Date): Function; }  **class** Person **implements** Human {  **constructor**(**public** firstName, **public** lastName) {  }   **public** name() {  **return** `${**this**.firstName} ${**this**.lastName}`;  }   **protected** whoAreYou() {  **return** `Hi i'm ${**this**.name()}`;  } }  **class** Student **extends** Person {  **constructor**(**public** firstName, **public** lastName, **public** course) {  **super**(firstName, lastName);  }   whoAreYou() {  **return** `${**super**.whoAreYou()} and i'm studying ${**this**.course}`;  } }  **let** asim = **new** Student("Asim", "Hussain", "typescript"); console.log(asim.whoAreYou()); |

**Decorator**

선언시에 가로채서 변환을 해준다.

|  |
| --- |
| **function** Student(config) {  **return function** (target) {  Object.defineProperty(target.prototype, 'course', {value: () => config.course})  } }    @Student({  course: "angular3" }) **class** Person {  **constructor**(**private** firstName, **private** lastName) {  }   **public** name() {  **return** `${**this**.firstName} ${**this**.lastName}`;  }   **protected** whoAreYou() {  **return** `Hi i'm ${**this**.name()}`;  } }  **let** asim = **new** Person("Asim", "Hussain"); //noinspection TypeScriptUnresolvedFunction console.log(asim.course());  tsc --experimentalDecorators decorator.ts  node decorator.js |

**Module**

export 로 다른 모듈에서 내자신을 import가능하도록한다.

|  |  |
| --- | --- |
| script.ts | utils.ts |
| **import** \* **as** utils **from** './utils'; console.log(utils.square(4)); utils.cow();  import {square, cow} from "./utils"  import square from "./utils"  import {square as sqr, cow} from "./utils" | **export function** square(x) {  **return** Math.pow(x, 2) }  **export function** cow() {  console.log("Mooooo!!!") }  export {square: square, cow: cow} 처럼해도됨 |

**Types**

|  |
| --- |
| "use strict";  // Core **let** decimal: **number** = 6; **let** done: **boolean** = **false**; **let** color: **string** = "blue"; **let** list: **number**[] = [1, 2, 3]; **let** list2: Array<**number**> = [1, 2, 3];  // Function **let** fun: Function = () => console.log("Hello"); **function** returnNumber(): **number** {  **return** 1; }  // Void **function** returnNothing(): **void** {  console.log("Moo"); }  // Enum **enum** Direction {  *Up*,  *Down*,  *Left*,  *Right* } **let** go: Direction; go = Direction.*Up*;   // Class **class** Person { } **let** person: Person; **let** people: Person[];  // Any **let** notsure: **any** = 1; notsure = "hello"; // This is fine since we don't do type checking with any  // Type Assertion **let** value: **any** = "Asim Hussain"; **let** length: **number** = (<**string**>value).length;   // Generics **class** Audio { } **class** Video { }  **class** Post<T> {  content: T; }  **let** audioPost: Post<Audio>; **let** videoPost: Post<Video>; |

npm install -g typings

typings install jquery --save --source dt --global

typings install ionic --save --source dt --global

**Angular CLI**

|  |  |
| --- | --- |
| npm install -g angular-cli  ng -v  ng new newProject  ng server  -----  ng generate component Header    ng g component LoginButton  ng g directive My  ng g pipe My  ng g servcice My  ng g class MyClass  ng g interface MyInterface  ng g enum MyEnum  -----  ng build    ng build --dev  또는  ./node\_modules/@angular/cli/bin/ng build -w -op ../webapp --dev  운영시  ng build --prod |  |

npm install moment --save

npm install @types/moment --save

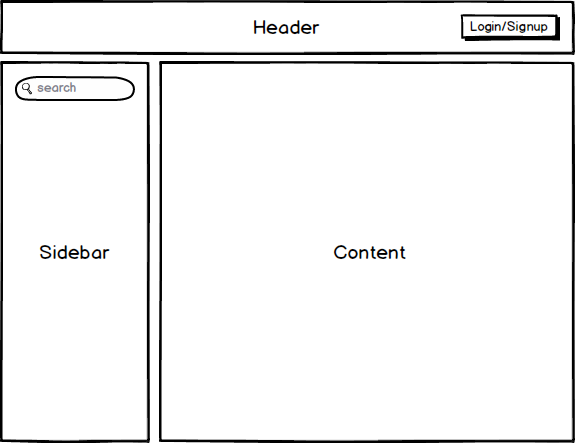
npm install bootstrap@next

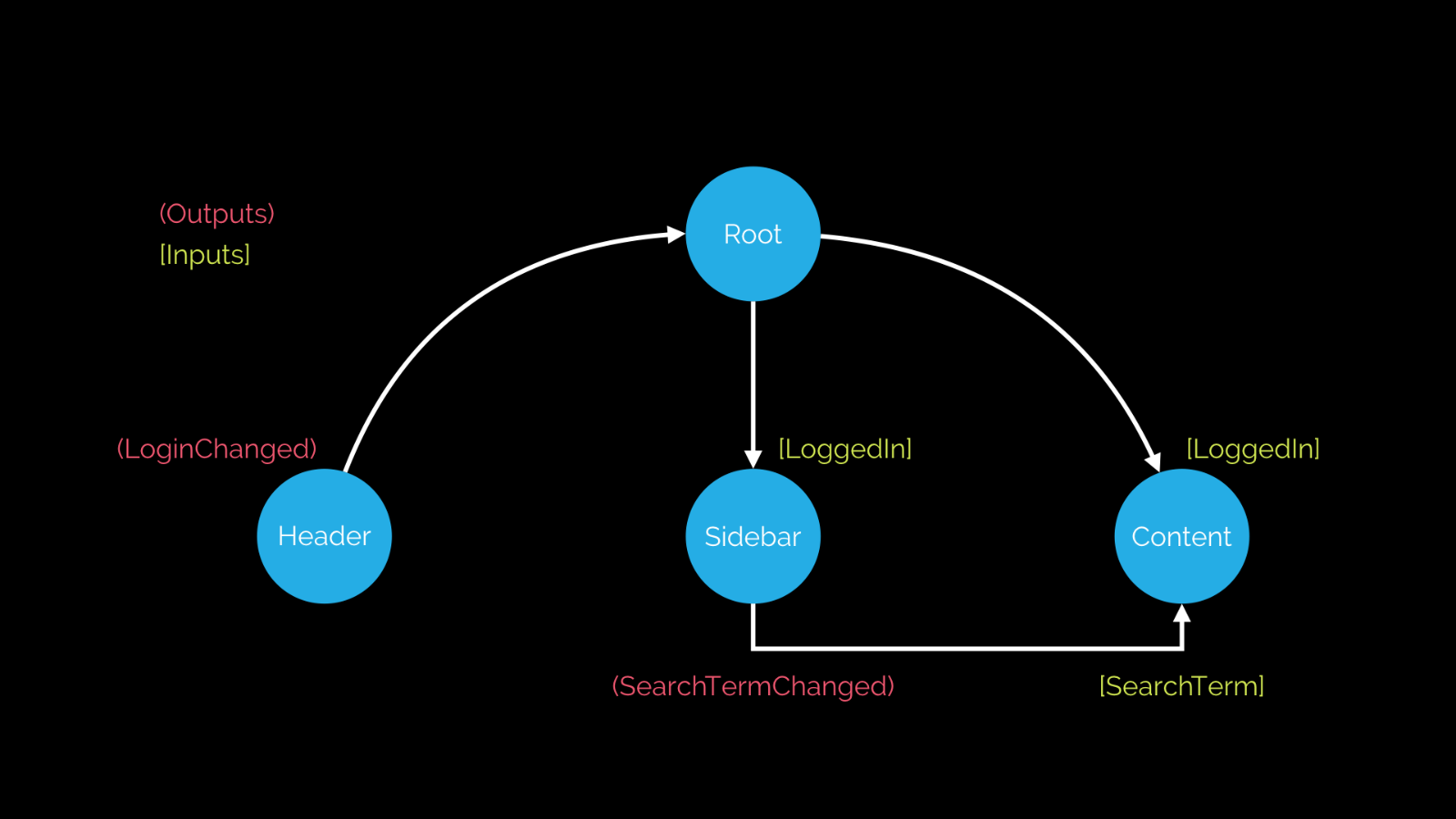
ng test

ng help

**Component**

Architecting with Components





<header (loginChanged)="loggedIn = $event"></header>

<sidebar (searchTermChanged)="searchTerm = $event"></sidebar>

<content [searchTerm]="searchTerm"></content>

**Templates, Styles & View Encapsulation**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| @Component({  selector: 'joke-form',  templateUrl: 'joke-form-component.html',  styleUrls: [  'joke-form-component.css'  ],  encapsulation: ViewEncapsulation.Emulated  // encapsulation: ViewEncapsulation.Native  // encapsulation: ViewEncapsulation.None  }) **class** JokeFormComponent {  @Output() jokeCreated = **new** EventEmitter<Joke>();   createJoke(setup: **string**, punchline: **string**) {  **this**.jokeCreated.emit(**new** Joke(setup, punchline));  } }  encapsulation: ViewEncapsulation.Emulated // encapsulation: ViewEncapsulation.Native // encapsulation: ViewEncapsulation.None   |  |  | | --- | --- | | ViewEncapsulation.None | encapsulated none screen  encapsulated none html | | ViewEncapsulation.Native |  | | ViewEncapsulation.Emulated | 기본값  encapsulation emulated html  encapsulation emulated css | |  |  | |

**Content Projection**

|  |
| --- |
| @Component({  selector: 'joke',  template: ` <div class="card card-block">  <h4 class="card-title">  <ng-content select=".setup"></ng-content>  </h4>  <p class="card-text"  [hidden]="data.hide">  <ng-content select=".punchline"></ng-content>  </p>  <a class="btn btn-primary"  (click)="data.toggle()">Tell Me  </a> </div> ` }) **class** JokeComponent {  @Input('joke') data: Joke; }  @Component({  selector: 'joke-list',  template: ` <joke-form (jokeCreated)="addJoke($event)"></joke-form> <joke \*ngFor="let j of jokes" [joke]="j">  <span class="setup">{{ j.setup }}?</span>  <h1 class="punchline">{{ j.punchline }}</h1> </joke>  ` }) **class** JokeListComponent {  jokes: Joke[];   **constructor**() {  **this**.jokes = [  **new** Joke("What did the cheese say when it looked in the mirror", "Hello-me (Halloumi)"),  **new** Joke("What kind of cheese do you use to disguise a small horse", "Mask-a-pony (Mascarpone)"),  **new** Joke("A kid threw a lump of cheddar at me", "I thought ‘That’s not very mature’"),  ];  }   addJoke(joke) {  **this**.jokes.unshift(joke);  } } |

**Lifecycle Hooks 라이프 사이클**

|  |
| --- |
| @Component({  selector: 'joke',  template: ` <div class="card card-block">  <h4 class="card-title">  <ng-content select=".setup"></ng-content>  </h4>  <p class="card-text"  [hidden]="data.hide">  <ng-content select=".punchline"></ng-content>  </p>  <a class="btn btn-primary"  (click)="data.toggle()">Tell Me  </a> </div> ` }) **class** JokeComponent **implements** OnChanges,  OnInit,  DoCheck,  AfterContentInit,  AfterContentChecked,  AfterViewInit,  AfterViewChecked,  OnDestroy {  @Input('joke') data: Joke;   **constructor**() {  console.log(`new - data is ${**this**.data}`);  }   ngOnChanges(changes: SimpleChanges) {  console.log(`ngOnChanges - data is ${**this**.data}`);   **for** (**let** key **in** changes) {  console.log(`${key} changed.  Current: ${changes[key].currentValue}.  Previous: ${changes[key].previousValue}`);  }  }  ngOnInit() {  console.log(`ngOnInit - data is ${**this**.data}`);  }  ngDoCheck() {  console.log("ngDoCheck")  }  ngAfterContentInit() {  console.log("ngAfterContentInit");  }  ngAfterContentChecked() {  console.log("ngAfterContentChecked");  }  ngAfterViewInit() {  console.log("ngAfterViewInit");  }  ngAfterViewChecked() {  console.log("ngAfterViewChecked");  }  ngOnDestroy() {  console.log("ngOnDestroy");  } }  Lifecycle Hooks |

**ViewChildren & ContentChildren**

|  |
| --- |
| @Component({  selector: 'joke',  template: ` <div class="card card-block">  <h4 class="card-title">  <ng-content select=".setup"></ng-content>  </h4>  <p class="card-text"  [hidden]="data.hide">  <ng-content select=".punchline"></ng-content>  </p>  <a class="btn btn-primary"  (click)="data.toggle()">Tell Me  </a> </div> ` }) **class** JokeComponent {   @Input('joke') data: Joke; }  @Component({  selector: 'joke-list',  template: ` <h4 #header>View Jokes</h4> <joke \*ngFor="let j of jokes" [joke]="j">  <span class="setup">{{ j.setup }}?</span>  <h1 class="punchline">{{ j.punchline }}</h1> </joke> <h4>Content Jokes</h4> <ng-content></ng-content> ` }) **class** JokeListComponent **implements** OnInit,  AfterContentInit,  AfterViewInit {   jokes: Joke[] = [  **new** Joke("What did the cheese say when it looked in the mirror", "Hello-me (Halloumi)"),  **new** Joke("What kind of cheese do you use to disguise a small horse", "Mask-a-pony (Mascarpone)")  ];   @ViewChild(JokeComponent) jokeViewChild: JokeComponent;  @ViewChildren(JokeComponent) jokeViewChildren: QueryList<JokeComponent>;  @ViewChild("header") headerEl: ElementRef;  @ContentChild(JokeComponent) jokeContentChild: JokeComponent;   **constructor**() {  console.log(`new - jokeViewChild is ${**this**.jokeViewChild}`);  console.log(`new - jokeContentChild is ${**this**.jokeContentChild}`);  }   ngAfterContentInit() {  console.log(`ngAfterContentInit - jokeContentChild is ${**this**.jokeContentChild}`);  }   ngAfterViewInit() {  console.log(`ngAfterViewInit - jokeViewChild is ${**this**.jokeViewChild}`);   **let** jokes: JokeComponent[] = **this**.jokeViewChildren.toArray();  console.log(jokes);   console.log(`ngAfterViewInit - headerEl is ${**this**.headerEl}`);  **this**.headerEl.nativeElement.textContent = "Best Joke Machine";  } } |

**Built-in**

**NgFor**

|  |
| --- |
| **import** {NgModule, Component} **from** '@angular/core'; **import** {BrowserModule} **from** '@angular/platform-browser'; **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic';   @Component({  selector: 'ngfor-example',  template: `<h4>NgFor</h4> <ul>  <li \*ngFor="let person of people; let i = index">  {{ i + 1 }} - {{ person.name }}  </li> </ul>  ` }) **class** NgForExampleComponent {  people: **any**[] = [  {  "name": "Douglas Pace"  },  {  "name": "Mcleod Mueller"  },  {  "name": "Day Meyers"  },  {  "name": "Aguirre Ellis"  },  {  "name": "Cook Tyson"  }  ]; }   @Component({  selector: 'ngfor-grouped-example',  template: `<h4>NgFor (grouped)</h4> <ul \*ngFor="let group of peopleByCountry">  <li>{{ group.country }}</li>  <ul>  <li \*ngFor="let person of group.people">  {{ person.name }}  </li>  </ul> </ul>  ` }) **class** NgForGroupedExampleComponent {   peopleByCountry: **any**[] = [  {  'country': 'UK',  'people': [  {  "name": "Douglas Pace"  },  {  "name": "Mcleod Mueller"  },  ]  },  {  'country': 'US',  'people': [  {  "name": "Day Meyers"  },  {  "name": "Aguirre Ellis"  },  {  "name": "Cook Tyson"  }  ]  }  ]; }  @Component({  selector: 'directives-app',  template: `  <ngfor-grouped-example></ngfor-grouped-example>  <ngfor-example></ngfor-example>  ` }) **class** DirectivesAppComponent { }   @NgModule({  imports: [BrowserModule],  declarations: [  NgForExampleComponent,  NgForGroupedExampleComponent,  DirectivesAppComponent],  bootstrap: [DirectivesAppComponent], }) **class** AppModule {  }  platformBrowserDynamic().bootstrapModule(AppModule);  NgFor Index  NgFor Grouped |

**NgIf & NgSwitch**

|  |
| --- |
| **import** {NgModule, Component} **from** '@angular/core'; **import** {BrowserModule} **from** '@angular/platform-browser'; **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic';  @Component({  selector: 'ngif-example',  template: ` <h4>NgIf</h4> <ul \*ngFor="let person of people">  <li \*ngIf="person.age < 30">  {{ person.name }} ({{ person.age }})  </li> </ul> ` }) **class** NgIfExampleComponent {   people: **any**[] = [  {  "name": "Douglas Pace",  "age": 35  },  {  "name": "Mcleod Mueller",  "age": 32  },  {  "name": "Day Meyers",  "age": 21  },  {  "name": "Aguirre Ellis",  "age": 34  },  {  "name": "Cook Tyson",  "age": 32  }  ]; }   @Component({  selector: 'ngswitch-example',  template: `<h4>NgSwitch</h4> <ul \*ngFor="let person of people"  [ngSwitch]="person.country">   <li \*ngSwitchCase="'UK'"  class="text-success">  {{ person.name }} ({{ person.country }})  </li>  <li \*ngSwitchCase="'USA'"  class="text-primary">  {{ person.name }} ({{ person.country }})  </li>  <li \*ngSwitchCase="'HK'"  class="text-danger">  {{ person.name }} ({{ person.country }})  </li>  <li \*ngSwitchDefault  class="text-warning">  {{ person.name }} ({{ person.country }})  </li> </ul>` }) **class** NgSwitchExampleComponent {   people: **any**[] = [  {  "name": "Douglas Pace",  "age": 35,  "country": 'MARS'  },  {  "name": "Mcleod Mueller",  "age": 32,  "country": 'USA'  },  {  "name": "Day Meyers",  "age": 21,  "country": 'HK'  },  {  "name": "Aguirre Ellis",  "age": 34,  "country": 'UK'  },  {  "name": "Cook Tyson",  "age": 32,  "country": 'USA'  }  ]; }   @Component({  selector: 'directives-app',  template: `  <ngswitch-example></ngswitch-example>  <ngif-example></ngif-example>  ` }) **class** DirectivesAppComponent { }   @NgModule({  imports: [BrowserModule],  declarations: [  NgIfExampleComponent,  NgSwitchExampleComponent,  DirectivesAppComponent],  bootstrap: [DirectivesAppComponent] }) **class** AppModule {  }  platformBrowserDynamic().bootstrapModule(AppModule); |

**NgStyle & NgClass**

|  |
| --- |
| **import** {NgModule, Component} **from** '@angular/core'; **import** {BrowserModule} **from** '@angular/platform-browser'; **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic';  @Component({  selector: 'ngstyle-example',  template: `<h4>NgStyle</h4> <ul \*ngFor="let person of people">  <li [ngStyle]="{'font-size.px':24}"  [style.color]="getColor(person.country)">  {{ person.name }} ({{ person.country }})  </li> </ul>  ` }) **class** NgStyleExampleComponent {   getColor(country) {  **switch** (country) {  **case** 'UK':  **return** 'green';  **case** 'USA':  **return** 'blue';  **case** 'HK':  **return** 'red';  }  }   people: **any**[] = [  {  "name": "Douglas Pace",  "country": 'UK'  },  {  "name": "Mcleod Mueller",  "country": 'USA'  },  {  "name": "Day Meyers",  "country": 'HK'  },  {  "name": "Aguirre Ellis",  "country": 'UK'  },  {  "name": "Cook Tyson",  "country": 'USA'  }  ]; }   @Component({  selector: 'ngclass-example',  template: `<h4>NgClass</h4> <ul \*ngFor="let person of people">  <li [ngClass]="{  'text-success':person.country === 'UK',  'text-primary':person.country === 'USA',  'text-danger':person.country === 'HK'  }">  {{ person.name }} ({{ person.country }})  </li> </ul>  <!-- <ul \*ngFor="let person of people">  <li [class.text-success]="person.country === 'UK'"  [class.text-primary]="person.country === 'USA'"  [class.text-danger]="person.country === 'HK'">  {{ person.name }} ({{ person.country }})  </li> </ul> -->  ` }) **class** NgClassExampleComponent {   people: **any**[] = [  {  "name": "Douglas Pace",  "age": 35,  "country": 'UK'  },  {  "name": "Mcleod Mueller",  "age": 32,  "country": 'USA'  },  {  "name": "Day Meyers",  "age": 21,  "country": 'HK'  },  {  "name": "Aguirre Ellis",  "age": 34,  "country": 'UK'  },  {  "name": "Cook Tyson",  "age": 32,  "country": 'USA'  }  ]; }   @Component({  selector: 'directives-app',  template: ` <ngclass-example></ngclass-example> <ngstyle-example></ngstyle-example>` }) **class** DirectivesAppComponent { }   @NgModule({  imports: [BrowserModule],  declarations: [  NgClassExampleComponent,  NgStyleExampleComponent,  DirectivesAppComponent],  bootstrap: [DirectivesAppComponent] }) **class** AppModule {  }  platformBrowserDynamic().bootstrapModule(AppModule); |

**NgNonBindable**

|  |
| --- |
| **import** {NgModule, Component} **from** '@angular/core'; **import** {BrowserModule} **from** '@angular/platform-browser'; **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic';  @Component({  selector: 'ngnonbindable-example',  template: `<h4>NgNonBindable</h4> <div>  To render the name variable we use this syntax  <pre ngNonBindable>{{ name }}</pre> </div>   ` }) **class** NgNonBindableExampleComponent { }  @Component({  selector: 'directives-app',  template: `<ngnonbindable-example></ngnonbindable-example>` }) **class** DirectivesAppComponent { }   @NgModule({  imports: [BrowserModule],  declarations: [NgNonBindableExampleComponent, DirectivesAppComponent],  bootstrap: [DirectivesAppComponent], }) **class** AppModule {  }  platformBrowserDynamic().bootstrapModule(AppModule);  NgNonBindable |

**Structural Directives**

|  |
| --- |
| **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic'; **import** {  Component,  Directive,  NgModule,  Input,  Output,  EventEmitter,  TemplateRef,  ViewContainerRef } **from** '@angular/core'; **import** {BrowserModule} **from** '@angular/platform-browser'; **import** {Directive, Input} **from** '@angular/core';  // // Domain Model //  **class** Joke {  **public** hide: **boolean**;   **constructor**(**public** setup: **string**, **public** punchline: **string**) {  **this**.hide = **true**;  }   toggle() {  **this**.hide = !**this**.hide;  } }  // // Structural Directives //  @Directive({  selector: '[ccIf]' }) **export class** CodeCraftIfDirective {  **constructor**(**private** templateRef: TemplateRef<**any**>,  **private** viewContainer: ViewContainerRef) {  }   @Input() **set** ccIf(condition: **boolean**) {  **if** (condition) {  **this**.viewContainer.createEmbeddedView(**this**.templateRef);  } **else** {  **this**.viewContainer.clear();  }  } }   @Directive({  selector: '[ccFor]' }) **export class** CodeCraftForOfDirective {  **constructor**(**private** templateRef: TemplateRef<**any**>,  **private** viewContainer: ViewContainerRef) {  }   @Input() **set** ccForOf(collection: **any**) {  **if** (condition) {  **this**.viewContainer.createEmbeddedView(**this**.templateRef);  } **else** {  **this**.viewContainer.clear();  }  } }   // // Components // //  @Component({  selector: 'joke',  template: ` <div class="card card-block">  <h4 class="card-title">  {{ data.setup }}  </h4>  <ng-template [ngIf]="!data.hide">  <p class="card-text">  {{ data.punchline }}  </p>   </ng-template>  <button class="btn btn-primary"  (click)="data.toggle()">Tell Me  </button> </div> ` }) **class** JokeComponent **implements** OnInit {  @Input('joke') data: Joke; }  @Component({  selector: 'joke-list',  template: ` <ng-template ngFor  let-j  [ngForOf]="jokes">  <joke [joke]="j"></joke> </ng-template> ` }) **class** JokeListComponent {  jokes: Joke[] = [];   **constructor**() {  **this**.jokes = [  **new** Joke("What did the cheese say when it looked in the mirror?", "Hello-me (Halloumi)"),  **new** Joke("What kind of cheese do you use to disguise a small horse?", "Mask-a-pony (Mascarpone)"),  **new** Joke("A kid threw a lump of cheddar at me", "I thought ‘That’s not very mature’"),  ];  } }   @Component({  selector: 'app',  template: ` <joke-list></joke-list> ` }) **class** AppComponent { }  // // Bootstrap //  @NgModule({  imports: [BrowserModule],  declarations: [  AppComponent,  JokeComponent,  JokeListComponent,  CodeCraftIfDirective  ],  bootstrap: [AppComponent] }) **export class** AppModule { }  platformBrowserDynamic().bootstrapModule(AppModule); |

**Custom Directives**

**Creating a custom directive**

|  |
| --- |
| **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic'; **import** {  Component,  Directive,  Renderer,  ElementRef,  NgModule,  Input,  Output,  EventEmitter } **from** '@angular/core'; **import** {BrowserModule} **from** '@angular/platform-browser';  **class** Joke {  **public** setup: **string**;  **public** punchline: **string**;  **public** hide: **boolean**;   **constructor**(setup: **string**, punchline: **string**) {  **this**.setup = setup;  **this**.punchline = punchline;  **this**.hide = **true**;  }   toggle() {  **this**.hide = !**this**.hide;  } }   @Directive({  selector: "[ccCardHover]" }) **class** CardHoverDirective {  **constructor**(**private** el: ElementRef,  **private** renderer: Renderer) {  //noinspection TypeScriptUnresolvedVariable,TypeScriptUnresolvedFunction  renderer.setElementStyle(el.nativeElement, 'backgroundColor', 'gray');  } }  @Component({  selector: 'joke',  template: ` <div class="card card-block" ccCardHover>  <h4 class="card-title">{{data.setup}}</h4>  <p class="card-text"  [hidden]="data.hide">{{data.punchline}}</p>  <button (click)="data.toggle()"  class="btn btn-primary">Tell Me  </button> </div>  ` }) **class** JokeComponent {  @Input('joke') data: Joke; }  @Component({  selector: 'joke-list',  template: ` <joke \*ngFor="let j of jokes" [joke]="j"></joke>  ` }) **class** JokeListComponent {  jokes: Joke[];   **constructor**() {  **this**.jokes = [  **new** Joke("What did the cheese say when it looked in the mirror?", "Hello-me (Halloumi)"),  **new** Joke("What kind of cheese do you use to disguise a small horse?", "Mask-a-pony (Mascarpone)"),  **new** Joke("A kid threw a lump of cheddar at me", "I thought ‘That’s not very mature’"),  ];  } }   @Component({  selector: 'app',  template: ` <joke-list></joke-list>  ` }) **class** AppComponent { }  @NgModule({  imports: [BrowserModule],  declarations: [  AppComponent,  JokeComponent,  JokeListComponent,  CardHoverDirective  ],  bootstrap: [AppComponent] }) **export class** AppModule { }  platformBrowserDynamic().bootstrapModule(AppModule);  Directive Gray |

**HostListener & HostBinding**

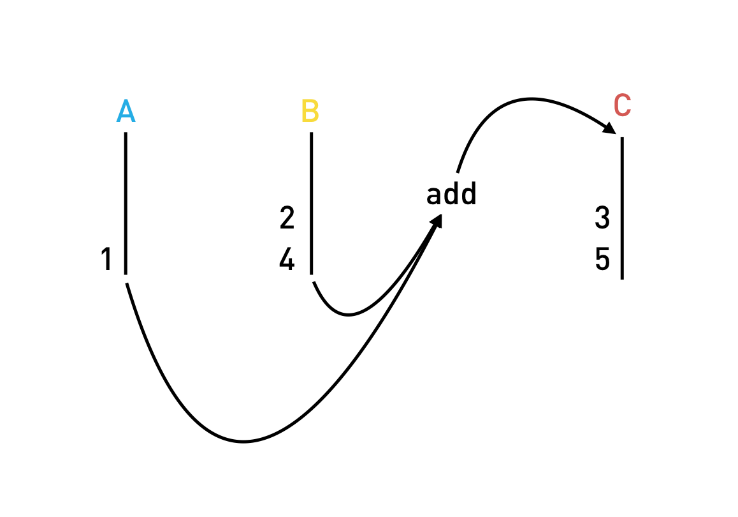
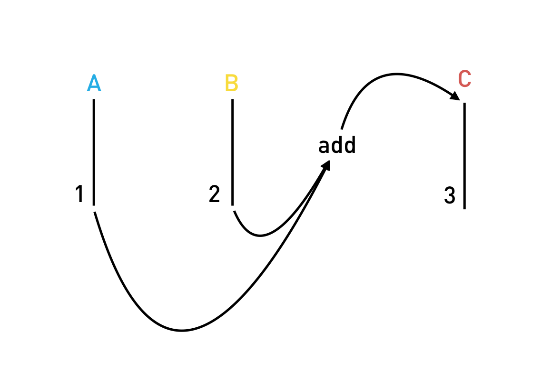
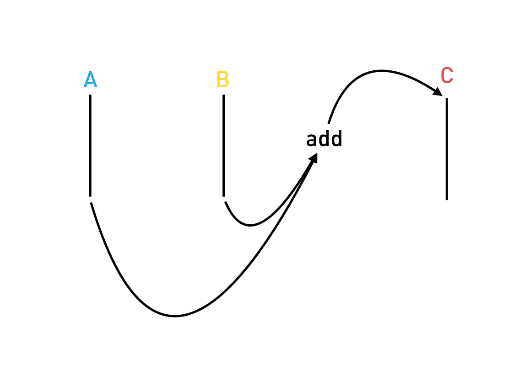
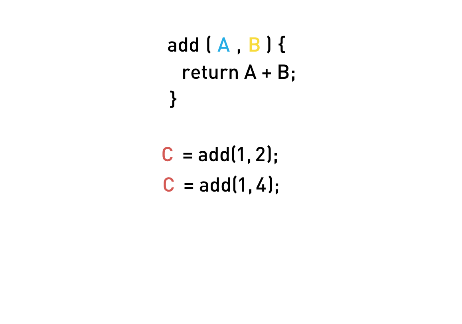
|  |
| --- |
| **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic'; **import** {  Component,  Directive,  Renderer,  HostListener,  HostBinding,  ElementRef,  NgModule,  Input,  Output,  EventEmitter } **from** '@angular/core'; **import** {BrowserModule} **from** '@angular/platform-browser';  **class** Joke {  **public** setup: **string**;  **public** punchline: **string**;  **public** hide: **boolean**;   **constructor**(setup: **string**, punchline: **string**) {  **this**.setup = setup;  **this**.punchline = punchline;  **this**.hide = **true**;  }   toggle() {  **this**.hide = !**this**.hide;  } }   @Directive({  selector: "[ccCardHover]" }) **class** CardHoverDirective {  @HostBinding('class.card-outline-primary')**private** ishovering: **boolean**;   **constructor**(**private** el: ElementRef,  **private** renderer: Renderer) {  // renderer.setElementStyle(el.nativeElement, 'backgroundColor', 'gray');  }   @HostListener('mouseover') onMouseOver() {  **let** part = **this**.el.nativeElement.querySelector('.card-text');  **this**.renderer.setElementStyle(part, 'display', 'block');  **this**.ishovering = **true**;  }   @HostListener('mouseout') onMouseOut() {  **let** part = **this**.el.nativeElement.querySelector('.card-text');  **this**.renderer.setElementStyle(part, 'display', 'none');  **this**.ishovering = **false**;  } }  @Component({  selector: 'joke',  template: ` <div class="card card-block" ccCardHover>  <h4 class="card-title">{{data.setup}}</h4>  <p class="card-text"  [style.display]="'none'">{{data.punchline}}</p> </div>  ` }) **class** JokeComponent {  @Input('joke') data: Joke; }  @Component({  selector: 'joke-list',  template: ` <joke \*ngFor="let j of jokes" [joke]="j"></joke>  ` }) **class** JokeListComponent {  jokes: Joke[];   **constructor**() {  **this**.jokes = [  **new** Joke("What did the cheese say when it looked in the mirror?", "Hello-me (Halloumi)"),  **new** Joke("What kind of cheese do you use to disguise a small horse?", "Mask-a-pony (Mascarpone)"),  **new** Joke("A kid threw a lump of cheddar at me", "I thought ‘That’s not very mature’"),  ];  } }   @Component({  selector: 'app',  template: ` <joke-list></joke-list>  ` }) **class** AppComponent { }  @NgModule({  imports: [BrowserModule],  declarations: [  AppComponent,  JokeComponent,  JokeListComponent,  CardHoverDirective  ],  bootstrap: [AppComponent] }) **export class** AppModule { }  platformBrowserDynamic().bootstrapModule(AppModule);  Directive HostBinding |

**Inputs & Configuration**

|  |
| --- |
| **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic'; **import** {  Component,  Directive,  Renderer,  HostListener,  HostBinding,  ElementRef,  NgModule,  Input,  Output,  EventEmitter } **from** '@angular/core'; **import** {BrowserModule} **from** '@angular/platform-browser';  **class** Joke {  **public** setup: **string**;  **public** punchline: **string**;  **public** hide: **boolean**;   **constructor**(setup: **string**, punchline: **string**) {  **this**.setup = setup;  **this**.punchline = punchline;  **this**.hide = **true**;  }   toggle() {  **this**.hide = !**this**.hide;  } }   @Directive({  selector: "[ccCardHover]" }) **class** CardHoverDirective {  @HostBinding('class.card-outline-primary') **private** ishovering: **boolean**;   @Input('ccCardHover') config: Object = {  querySelector: '.card-text'  };   **constructor**(**private** el: ElementRef,  **private** renderer: Renderer) {  // renderer.setElementStyle(el.nativeElement, 'backgroundColor', 'gray');  }   @HostListener('mouseover') onMouseOver() {  **let** part = **this**.el.nativeElement.querySelector(**this**.config.querySelector);  **this**.renderer.setElementStyle(part, 'display', 'block');  **this**.ishovering = **true**;  }   @HostListener('mouseout') onMouseOut() {  **let** part = **this**.el.nativeElement.querySelector(**this**.config.querySelector);  **this**.renderer.setElementStyle(part, 'display', 'none');  **this**.ishovering = **false**;  } }  @Component({  selector: 'joke',  template: ` <div class="card card-block"  [ccCardHover]="{querySelector:'.card-text'}">  <h4 class="card-title">{{data.setup}}</h4>  <p class="card-text"  [style.display]="'none'">{{data.punchline}}</p> </div>  ` }) **class** JokeComponent {  @Input('joke') data: Joke; }  @Component({  selector: 'joke-list',  template: ` <joke \*ngFor="let j of jokes" [joke]="j"></joke>  ` }) **class** JokeListComponent {  jokes: Joke[];   **constructor**() {  **this**.jokes = [  **new** Joke("What did the cheese say when it looked in the mirror?", "Hello-me (Halloumi)"),  **new** Joke("What kind of cheese do you use to disguise a small horse?", "Mask-a-pony (Mascarpone)"),  **new** Joke("A kid threw a lump of cheddar at me", "I thought ‘That’s not very mature’"),  ];  } }   @Component({  selector: 'app',  template: ` <joke-list></joke-list>  ` }) **class** AppComponent { }  @NgModule({  imports: [BrowserModule],  declarations: [  AppComponent,  JokeComponent,  JokeListComponent,  CardHoverDirective  ],  bootstrap: [AppComponent] }) **export class** AppModule { }  platformBrowserDynamic().bootstrapModule(AppModule); |

**Reactive Programming with RxJS**

**Streams & Reactive Programming**



**Observables & RxJS**

|  |
| --- |
| let obs = Rx.Observable  .interval(1000)  .take(3)  .map((v) => Date.now());  obs.subscribe(value => console.log("Subscriber: " + value)); |
| <!DOCTYPE html> <html>  <head>  <script src="https://cdnjs.cloudflare.com/ajax/libs/rxjs/4.1.0/rx.all.js"></script>  <script src="main.js"></script> </head>  <body> </body>  </html> |

|  |  |
| --- | --- |
| Subscriber: 0  Subscriber: 1  Subscriber: 2  Subscriber: 3  Subscriber: 4  Subscriber: 5  Subscriber: 6  Subscriber: 7  Subscriber: 8  Subscriber: 9  Subscriber: 10 | Subscriber: 0  Subscriber: 1  Subscriber: 2 |
| Subscriber: 1475506794287  Subscriber: 1475506795286  Subscriber: 1475506796285 | Map Marble |
|  |  |

[**http://reactivex.io**](http://reactivex.io)

**RxJS & Angular**

|  |
| --- |
| **import** {NgModule, Component} **from** '@angular/core'; **import** {BrowserModule} **from** '@angular/platform-browser'; **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic'; **import** {ReactiveFormsModule, FormGroup, FormControl, Validators, FormBuilder} **from** "@angular/forms"; **import** 'rxjs/Rx';  @Component({  selector: 'form-app',  template: `<form [formGroup]="form"  (ngSubmit)="onSubmit()">    <!-- Output comment -->  <div class="card card-block">  <pre class="card-text">{{ form.value.comment }}</pre>  </div>  <p class="small">{{ form.value.lastUpdateTS }}</p>   <!-- Comment text area -->  <div class="form-group">  <label for="comment">Comment</label>  <textarea class="form-control"  formControlName="comment"  rows="3"></textarea>  <small class="form-text text-muted">  <span>{{ 100 - form.value.comment.length }}</span> characters left  </small>  </div>   <!-- Name input -->  <div class="form-group">  <label for="name">Name</label>  <input type="text"  class="form-control"  formControlName="name"  placeholder="Enter name">  </div>   <!-- Email input -->  <div class="form-group">  <label for="email">Email address</label>  <input type="email"  class="form-control"  formControlName="email"  placeholder="Enter email">  <small class="form-text text-muted">  We'll never share your email with anyone else.  </small>  </div>   <button type="submit"  class="btn btn-primary"  [disabled]="!form.valid">Submit  </button> </form>   ` }) **class** FormAppComponent {  form: FormGroup;  comment = **new** FormControl("", Validators.required);  name = **new** FormControl("", Validators.required);  email = **new** FormControl("", [  Validators.required,  Validators.pattern("[^ @]\*@[^ @]\*")  ]);   /\* Observable Solution \*/  **constructor**(fb: FormBuilder) {  **this**.form = fb.group({  "comment": **this**.comment,  "name": **this**.name,  "email": **this**.email  });  this.form.valueChanges  .filter(data => this.form.valid)  .map(data => {  data.comment = data.comment.replace(/<(?:.|\n)\*?>/gm, '');  return data  })  .map(data => {  data.lastUpdateTS = new Date();  return data  })  .subscribe( data => console.log(JSON.stringify(data)));  }   /\* None Observable Solution \*/  // constructor(fb: FormBuilder) {  // this.form = fb.group({  // "comment": this.comment,  // "name": this.name,  // "email": this.email  // });  // this.form.valueChanges  // .subscribe( data => {  // if (this.form.valid) {  // data.comment = data.comment.replace(/<(?:.|\n)\*?>/gm, '');  // data.lastUpdateTS = new Date();  // console.log(JSON.stringify(data))  // }  // });  // }   onSubmit() {  console.log("Form submitted!");  } }  @Component({  selector: 'app',  template: ` <form-app></form-app>  ` }) **class** AppComponent { }   @NgModule({  imports: [BrowserModule, ReactiveFormsModule],  declarations: [AppComponent, FormAppComponent],  bootstrap: [AppComponent], }) **class** AppModule {  }  platformBrowserDynamic().bootstrapModule(AppModule);  form example |

**Pipes**

**Built-in Pipes**

|  |
| --- |
| **import** {NgModule, Component} **from** '@angular/core'; **import** {BrowserModule} **from** '@angular/platform-browser'; **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic';  @Component({  selector: 'pipe-builtins',  template: `<div class="card card-block">  <h4 class="card-title">Currency</h4>  <div class="card-text">  <p ngNonBindable>{{ 1234.56 | currency:'CAD' }}</p>  <p>{{ 1234.56 | currency:"CAD" }}</p>    <p ngNonBindable>{{ 1234.56 | currency:'CAD':'code' }}</p>  <p>{{ 1234.56 | currency:'CAD':'code'}}</p>    <p ngNonBindable>{{ 1234.56 | currency:'CAD':'symbol' }}</p>  <p>{{ 1234.56 | currency:'CAD':'symbol'}}</p>    <p ngNonBindable>{{ 1234.56 | currency:'CAD':'symbol-narrow' }}</p>  <p>{{ 1234.56 | currency:'CAD':'symbol-narrow'}}</p>   </div> </div>  <div class="card card-block">  <h4 class="card-title">Date</h4>  <div class="card-text">  <p ngNonBindable>{{ dateVal | date: 'shortTime' }}</p>  <p>{{ dateVal | date: 'shortTime' }}</p>   <p ngNonBindable>{{ dateVal | date:'fullDate' }}</p>  <p>{{ dateVal | date: 'fullDate' }}</p>   <p ngNonBindable>{{ dateVal | date: 'shortTime' }}</p>  <p>{{ dateVal | date: 'shortTime' }}</p>   <p ngNonBindable>{{ dateVal | date: 'd/M/y' }}</p>  <p>{{ dateVal | date: 'd/M/y' }}</p>  </div> </div>  <div class="card card-block">  <div class="card-text">  <h4 class="card-title">DecimalPipe</h4>  <p ngNonBindable>{{ 3.14159265 | number: '3.1-2' }}</p>  <p>{{ 3.14159265 | number: '3.1-2' }}</p>   <p ngNonBindable>{{ 3.14159265 | number: '1.4-4' }}</p>  <p>{{ 3.14159265 | number: '1.4-4' }}</p>  </div> </div>  <div class="card card-block">  <h4 class="card-title">JsonPipe</h4>  <div class="card-text">  <p ngNonBindable>{{ jsonVal }}</p>  <p>{{ jsonVal }}</p>    <p ngNonBindable>{{ jsonVal | json }}</p>  <p>{{ jsonVal | json }}</p>  </div> </div>   <div class="card card-block">  <h4 class="card-title">LowerCasePipe</h4>  <div class="card-text">  <p ngNonBindable>{{ 'ASIM' | lowercase }}</p>  <p>{{ 'ASIM' | lowercase }}</p>  </div> </div>  <div class="card card-block">  <h4 class="card-title">UpperCasePipe</h4>  <div class="card-text">  <p ngNonBindable>{{ 'asim' | uppercase }}</p>  <p>{{ 'asim' | uppercase }}</p>  </div> </div>  <div class="card card-block">  <h4 class="card-title">PercentPipe</h4>  <div class="card-text">  <p ngNonBindable>{{ 0.123456 | percent }}</p>  <p>{{ 0.123456 | percent }}</p>   <p ngNonBindable>{{ 0.123456 | percent: '2.1-2' }}</p>  <p>{{ 0.123456 | percent: '2.1-2' }}</p>   <p ngNonBindable>{{ 42 | percent: '10.4-4' }}</p>  <p>{{ 0.123456 | percent : "10.4-4" }}</p>  </div> </div>  <div class="card card-block">  <h4 class="card-title">SlicePipe</h4>  <div class="card-text">  <p ngNonBindable>{{ [1,2,3,4,5,6] | slice:1:3 }}</p>  <p>{{ [1,2,3,4,5,6] | slice:1:3 }}</p>   <p ngNonBindable>{{ [1,2,3,4,5,6] | slice:2 }}</p>  <p>{{ [1,2,3,4,5,6] | slice:2 }}</p>   <p ngNonBindable>{{ [1,2,3,4,5,6] | slice:2:-1 }}</p>  <p>{{ [1,2,3,4,5,6] | slice:2:-1 }}</p>   <pre ngNonBindable>  &lt;ul&gt;  &lt;li \*ngFor=&quot;let v of [1,2,3,4,5,6] | slice:2:-1&quot;&gt;  {{v}}  &lt;/li&gt; &lt;/ul&gt;  </pre>    <ul>  <li \*ngFor="let v of [1,2,3,4,5,6] | slice:2:-1">  {{v}}  </li>  </ul>  </div> </div>   ` }) **class** PipeBuiltinsComponent {  **private** dateVal: Date = **new** Date();  **private** jsonVal: Object = {moo: 'foo', goo: {too: 'new'}};  }  @Component({  selector: 'app',  template: ` <pipe-builtins></pipe-builtins>  ` }) **class** AppComponent { }   @NgModule({  imports: [BrowserModule],  declarations: [AppComponent,  PipeBuiltinsComponent  ],  bootstrap: [AppComponent], }) **class** AppModule {  }  platformBrowserDynamic().bootstrapModule(AppModule);  decimal pipe  date pipe  currency pipe  json pipe  lowercase pipe  uppercase pipe  percent pipe  slice pipe  <https://angular.io/docs/ts/latest/api/common/index/DatePipe-pipe.html> |

**Async Pipe**

|  |
| --- |
| **import** {NgModule, Component, OnDestroy} **from** '@angular/core'; **import** {BrowserModule} **from** '@angular/platform-browser'; **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic'; **import** { Observable } **from** 'rxjs/Rx'; @Component({  selector: 'async-pipe',  template: `  <div class="card card-block">  <h4 class="card-title">AsyncPipe</h4>    <p class="card-text" ngNonBindable>{{ promise | async }} </p>  <p class="card-text">{{ promise | async }} </p>   <p class="card-text" ngNonBindable>{{ observable | async }} </p>  <p class="card-text">{{ observable | async }}</p>    <p class="card-text" ngNonBindable>{{ observableData }} </p>  <p class="card-text">{{ observableData }}</p>   </div> ` }) **class** AsyncPipeComponent **implements** OnDestroy {  promise: Promise<**string**>;  observable: Observable<**number**>;  subscription: Object = **null**;  observableData: **number**;  **constructor**() {  **this**.promise = **this**.getPromise();  **this**.observable = **this**.getObservable();  **this**.subscribeObservable();  }  getObservable() {  **return** Observable  .interval(1000)  .take(10)  .map((v) => v \* v);  }  subscribeObservable() {// AsyncPipe subscribes to the observable automatically  **this**.subscription = **this**.getObservable()  .subscribe((v) => **this**.observableData = v);  }  getPromise() {  **return new** Promise((resolve, reject) => {  setTimeout(() => resolve("Promise complete!"), 3000);  });  }  ngOnDestroy() {// AsyncPipe unsubscribes from the observable automatically  **if** (**this**.subscription) {  **this**.subscription.unsubscribe();  }  } } @Component({  selector: 'app',  template: ` <async-pipe></async-pipe> ` }) **class** AppComponent {  imageUrl: **string** = ""; } @NgModule({  imports: [BrowserModule],  declarations: [AppComponent,  AsyncPipeComponent  ],  bootstrap: [AppComponent], }) **class** AppModule { } platformBrowserDynamic().bootstrapModule(AppModule); |

**Custom Pipes**

|  |
| --- |
| **import** {NgModule, Component, Pipe} from '@angular/core'; **import** {BrowserModule} from '@angular/platform-browser'; **import** {platformBrowserDynamic} from '@angular/platform-browser-dynamic'; **import** {Observable} from 'rxjs/Rx';   @Pipe({  name: "default" }) **class** DefaultPipe {  transform(value: string, fallback: string, forceHttps: boolean = **false**): string {  **let** image = "";  **if** (value) {  image = value;  } **else** {  image = fallback;  }   **if** (forceHttps) {  **if** (image.indexOf("https") == -1) {  image = image.replace("http", "https");  }  }   **return** image;  } }  @Component({  selector: 'app',  template: `  <img [src]="imageUrl | default:'http://s3.amazonaws.com/uifaces/faces/twitter/sillyleo/128.jpg':true"/>  ` }) **class** AppComponent {  imageUrl: string = ""; }   @NgModule({  imports: [BrowserModule],  declarations: [AppComponent,  DefaultPipe  ],  bootstrap: [AppComponent], }) **class** AppModule {  }  platformBrowserDynamic().bootstrapModule(AppModule);  default pipe part1 |

**Form**

**Model Driven Forms**

|  |
| --- |
| **import** {  NgModule,  Component,  Pipe,  OnInit } **from** '@angular/core'; **import** {  ReactiveFormsModule,  FormsModule,  FormGroup,  FormControl,  Validators,  FormBuilder } **from** '@angular/forms'; **import** {BrowserModule} **from** '@angular/platform-browser'; **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic';   @Component({  selector: 'model-form',  template: `<!--suppress ALL --> <form novalidate  [formGroup]="myform">   <fieldset formGroupName="name">  <div class="form-group">  <label>First Name</label>  <input type="text"  class="form-control"  formControlName="firstName">  </div>   <div class="form-group">  <label>Last Name</label>  <input type="text"  class="form-control"  formControlName="lastName">  </div>  </fieldset>    <div class="form-group">  <label>Email</label>  <input type="email"  class="form-control"  formControlName="email">  </div>   <div class="form-group">  <label>Password</label>  <input type="password"  class="form-control"  formControlName="password">  </div>   <div class="form-group">  <label>Language</label>  <select class="form-control"  formControlName="language">  <option value="">Please select a language</option>  <option \*ngFor="let lang of langs"  [value]="lang">{{lang}}  </option>  </select>  </div>   <pre>{{myform.value | json}}</pre> </form>  ` }) **class** ModelFormComponent **implements** OnInit {  langs: **string**[] = [  'English',  'French',  'German',  ];  myform: FormGroup;    ngOnInit() {  **this**.myform = **new** FormGroup({  name: **new** FormGroup({  firstName: **new** FormControl('', Validators.required),  lastName: **new** FormControl('', Validators.required),  }),  email: **new** FormControl('', [  Validators.required,  Validators.pattern("[^ @]\*@[^ @]\*")  ]),  password: **new** FormControl('', [  Validators.required,  Validators.minLength(8)  ]),  language: **new** FormControl()  });  } }   @Component({  selector: 'app',  template: `<model-form></model-form>` }) **class** AppComponent { }   @NgModule({  imports: [  BrowserModule,  FormsModule,  ReactiveFormsModule],  declarations: [  AppComponent,  ModelFormComponent  ],  bootstrap: [  AppComponent  ], }) **class** AppModule { }  platformBrowserDynamic().bootstrapModule(AppModule);  form setup |

**Model Driven Form Validation**

|  |
| --- |
| **import** {  NgModule,  Component,  Pipe,  OnInit } **from** '@angular/core'; **import** {  ReactiveFormsModule,  FormsModule,  FormGroup,  FormControl,  Validators,  FormBuilder } **from** '@angular/forms'; **import** {BrowserModule} **from** '@angular/platform-browser'; **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic';   @Component({  selector: 'model-form',  template: `<form novalidate  [formGroup]="myform">   <fieldset formGroupName="name">  <div class="form-group"  [ngClass]="{  'has-danger': firstName.invalid && (firstName.dirty || firstName.touched),  'has-success': firstName.valid && (firstName.dirty || firstName.touched)  }">  <label>First Name</label>  <input type="text"  class="form-control"  formControlName="firstName"  required>  <div class="form-control-feedback"  \*ngIf="firstName.errors && (firstName.dirty || firstName.touched)">  <p \*ngIf="firstName.errors.required">First Name is required</p>  </div>   <!--  <pre>Valid? {{ myform.controls.name.controls.firstName.valid }}</pre>  <pre>Dirty? {{ myform.controls.name.controls.firstName.dirty }}</pre>  -->  </div>   <div class="form-group"  [ngClass]="{  'has-danger': lastName.invalid && (lastName.dirty || lastName.touched),  'has-success': lastName.valid && (lastName.dirty || lastName.touched)  }">  <label>Last Name</label>  <input type="text"  class="form-control"  formControlName="lastName"  required>  <div class="form-control-feedback"  \*ngIf="lastName.errors && (lastName.dirty || lastName.touched)">  <p \*ngIf="lastName.errors.required">Last Name is required</p>  </div>  </div>  </fieldset>    <div class="form-group"  [ngClass]="{  'has-danger': email.invalid && (email.dirty || email.touched),  'has-success': email.valid && (email.dirty || email.touched)  }">  <label>Email</label>  <input type="email"  class="form-control"  formControlName="email"  required>  <div class="form-control-feedback"  \*ngIf="email.errors && (email.dirty || email.touched)">  <p \*ngIf="email.errors.required">Email is required</p>  <p \*ngIf="password.errors.pattern">The email address must contain at least the @ character</p>  </div>   <!--  <pre>Valid? {{ myform.controls.email.valid }}</pre>  <pre>Dirty? {{ myform.controls.email.dirty }}</pre>  -->   </div>   <div class="form-group"  [ngClass]="{  'has-danger': password.invalid && (password.dirty || password.touched),  'has-success': password.valid && (password.dirty || password.touched)  }">  <label>Password</label>  <input type="password"  class="form-control"  formControlName="password"  required>  <div class="form-control-feedback"  \*ngIf="password.errors && (password.dirty || password.touched)">  <p \*ngIf="password.errors.required">Password is required</p>  <p \*ngIf="password.errors.minlength">Password must be 8 characters long, we need another {{password.errors.minlength.requiredLength - password.errors.minlength.actualLength}} characters </p>  </div>  </div>   <!--  <pre>{{ password.errors | json }}</pre>  -->   <div class="form-group"  [ngClass]="{  'has-danger': language.invalid && (language.dirty || language.touched),  'has-success': language.valid && (language.dirty || language.touched)  }">  <label>Language</label>  <select class="form-control"  formControlName="language">  <option value="">Please select a language</option>  <option \*ngFor="let lang of langs"  [value]="lang">{{lang}}  </option>  </select>  </div>   <pre>{{myform.value | json}}</pre> </form>` }) **class** ModelFormComponent **implements** OnInit {  langs: **string**[] = [  'English',  'French',  'German',  ];  myform: FormGroup;  firstName: FormControl;  lastName: FormControl;  email: FormControl;  password: FormControl;  language: FormControl;    ngOnInit() {  **this**.createFormControls();  **this**.createForm();  }   createFormControls() {  **this**.firstName = **new** FormControl('', Validators.required);  **this**.lastName = **new** FormControl('', Validators.required);  **this**.email = **new** FormControl('', [  Validators.required,  Validators.pattern("[^ @]\*@[^ @]\*")  ]);  **this**.password = **new** FormControl('', [  Validators.required,  Validators.minLength(8)  ]);  **this**.language = **new** FormControl('');  }   createForm() {  **this**.myform = **new** FormGroup({  name: **new** FormGroup({  firstName: **this**.firstName,  lastName: **this**.lastName,  }),  email: **this**.email,  password: **this**.password,  language: **this**.language  });  } }   @Component({  selector: 'app',  template: `<model-form></model-form>` }) **class** AppComponent { }   @NgModule({  imports: [  BrowserModule,  FormsModule,  ReactiveFormsModule],  declarations: [  AppComponent,  ModelFormComponent  ],  bootstrap: [  AppComponent  ], }) **class** AppModule { }  platformBrowserDynamic().bootstrapModule(AppModule); |

**Submitting & Resetting**

|  |
| --- |
| **import** {  NgModule,  Component,  Pipe,  OnInit } **from** '@angular/core'; **import** {  ReactiveFormsModule,  FormsModule,  FormGroup,  FormControl,  Validators,  FormBuilder } **from** '@angular/forms'; **import** {BrowserModule} **from** '@angular/platform-browser'; **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic';   @Component({  selector: 'model-form',  template: `<form novalidate  [formGroup]="myform"  (ngSubmit)="onSubmit()">   <fieldset formGroupName="name">  <div class="form-group"  [ngClass]="{  'has-danger': firstName.invalid && (firstName.dirty || firstName.touched),  'has-success': firstName.valid && (firstName.dirty || firstName.touched)  }">  <label>First Name</label>  <input type="text"  class="form-control"  formControlName="firstName"  required>  <div class="form-control-feedback"  \*ngIf="firstName.errors && (firstName.dirty || firstName.touched)">  <p \*ngIf="firstName.errors.required">Last Name is required</p>  </div>   <!--  <pre>Valid? {{ myform.controls.name.controls.firstName.valid }}</pre>  <pre>Dirty? {{ myform.controls.name.controls.firstName.dirty }}</pre>  -->  </div>   <div class="form-group"  [ngClass]="{  'has-danger': lastName.invalid && (lastName.dirty || lastName.touched),  'has-success': lastName.valid && (lastName.dirty || lastName.touched)  }">  <label>Last Name</label>  <input type="text"  class="form-control"  formControlName="lastName"  required>  <div class="form-control-feedback"  \*ngIf="lastName.errors && (lastName.dirty || lastName.touched)">  <p \*ngIf="lastName.errors.required">Last Name is required</p>  </div>  </div>  </fieldset>    <div class="form-group"  [ngClass]="{  'has-danger': email.invalid && (email.dirty || email.touched),  'has-success': email.valid && (email.dirty || email.touched)  }">  <label>Email</label>  <input type="email"  class="form-control"  formControlName="email"  required>  <div class="form-control-feedback"  \*ngIf="email.errors && (email.dirty || email.touched)">  <p \*ngIf="email.errors.required">Email is required</p>  <p \*ngIf="password.errors.pattern">The email address must contain at least the @ character</p>  </div>   <!--  <pre>Valid? {{ myform.controls.email.valid }}</pre>  <pre>Dirty? {{ myform.controls.email.dirty }}</pre>  -->   </div>   <div class="form-group"  [ngClass]="{  'has-danger': password.invalid && (password.dirty || password.touched),  'has-success': password.valid && (password.dirty || password.touched)  }">  <label>Password</label>  <input type="password"  class="form-control"  formControlName="password"  required>  <div class="form-control-feedback"  \*ngIf="password.errors && (password.dirty || password.touched)">  <p \*ngIf="password.errors.required">Password is required</p>  <p \*ngIf="password.errors.minlength">Password must be 8 characters long, we need another {{password.errors.minlength.requiredLength - password.errors.minlength.actualLength}} characters </p>  </div>  </div>   <!--   <pre>{{ language.errors | json }}</pre>  -->   <div class="form-group"  [ngClass]="{  'has-danger': language.invalid && (language.dirty || language.touched),  'has-success': language.valid && (language.dirty || language.touched)  }">  <label>Language</label>  <select class="form-control"  formControlName="language">  <option value="">Please select a language</option>  <option \*ngFor="let lang of langs"  [value]="lang">{{lang}}  </option>  </select>  </div>   <button type="submit"  class="btn btn-primary">Submit  </button>   <pre>{{myform.value | json}}</pre> </form>` }) **class** ModelFormComponent **implements** OnInit {  langs: **string**[] = [  'English',  'French',  'German',  ];  myform: FormGroup;  firstName: FormControl;  lastName: FormControl;  email: FormControl;  password: FormControl;  language: FormControl;    ngOnInit() {  **this**.createFormControls();  **this**.createForm();  }   createFormControls() {  **this**.firstName = **new** FormControl('', Validators.required);  **this**.lastName = **new** FormControl('', Validators.required);  **this**.email = **new** FormControl('', [  Validators.required,  Validators.pattern("[^ @]\*@[^ @]\*")  ]);  **this**.password = **new** FormControl('', [  Validators.required,  Validators.minLength(8)  ]);  **this**.language = **new** FormControl('');  }   createForm() {  **this**.myform = **new** FormGroup({  name: **new** FormGroup({  firstName: **this**.firstName,  lastName: **this**.lastName,  }),  email: **this**.email,  password: **this**.password,  language: **this**.language  });  }   onSubmit() {  **if** (**this**.myform.valid) {  console.log("Form Submitted!");  **this**.myform.reset();  }  } }  @Component({  selector: 'app',  template: `<model-form></model-form>` }) **class** AppComponent { } @NgModule({  imports: [  BrowserModule,  FormsModule,  ReactiveFormsModule],  declarations: [  AppComponent,  ModelFormComponent  ],  bootstrap: [  AppComponent  ], }) **class** AppModule { }  platformBrowserDynamic().bootstrapModule(AppModule); |

**Reactive Model Form**

|  |
| --- |
| **import** {  NgModule,  Component,  OnInit } from '@angular/core'; **import** {  ReactiveFormsModule,  FormControl } from '@angular/forms'; **import** {BrowserModule} from '@angular/platform-browser'; **import** {platformBrowserDynamic} from '@angular/platform-browser-dynamic'; **import** 'rxjs/Rx';  @Component({  selector: 'reactive-model-form',  template: `<input type="search"  class="form-control"  placeholder="Please enter search term"  [formControl]="searchField"> <hr/> <ul>  <li \*ngFor="let search of searches">{{ search }}</li> </ul> ` }) **class** ReactiveModelFormComponent **implements** OnInit {   searchField: FormControl;  searches: string[] = [];   ngOnInit() {  **this**.searchField = **new** FormControl();  **this**.searchField.valueChanges  .debounceTime(400)  .distinctUntilChanged()  .subscribe(term => {  **this**.searches.push(term);  });  } }  @Component({  selector: 'app',  template: `<reactive-model-form></reactive-model-form>` }) **class** AppComponent { }   @NgModule({  imports: [  BrowserModule,  ReactiveFormsModule  ],  declarations: [  AppComponent,  ReactiveModelFormComponent  ],  bootstrap: [  AppComponent  ], }) **class** AppModule { }  platformBrowserDynamic().bootstrapModule(AppModule); |

**Template Driven Forms**

|  |
| --- |
| **import** {  NgModule,  Component,  OnInit,  ViewChild } **from** '@angular/core'; **import** {  FormsModule,  FormGroup,  FormControl } **from** '@angular/forms'; **import** {BrowserModule} **from** '@angular/platform-browser'; **import** {platformBrowserDynamic} **from** '@angular/platform-browser-dynamic';  **class** Signup {  **constructor**(**public** firstName: **string** = '',  **public** lastName: **string** = '',  **public** email: **string** = '',  **public** password: **string** = '',  **public** language: **string** = '') {  } }   @Component({  selector: 'template-form',  template: `<!--suppress ALL --> <form novalidate  (ngSubmit)="onSubmit()"  #f="ngForm">   <fieldset ngModelGroup="name">  <div class="form-group"  [ngClass]="{  'has-danger': firstName.invalid && (firstName.dirty || firstName.touched),  'has-success': firstName.valid && (firstName.dirty || firstName.touched)  }">  <label>First Name</label>  <input type="text"  class="form-control"  name="firstName"  [(ngModel)]="model.firstName"  required  #firstName="ngModel">  <div class="form-control-feedback"  \*ngIf="firstName.errors && (firstName.dirty || firstName.touched)">  <p \*ngIf="firstName.errors.required">First name is required</p>  </div>  </div>   <div class="form-group"  [ngClass]="{  'has-danger': lastName.invalid && (lastName.dirty || lastName.touched),  'has-success': lastName.valid && (lastName.dirty || lastName.touched)  }">  <label>Last Name</label>  <input type="text"  class="form-control"  name="lastName"  [(ngModel)]="model.lastName"  required  #lastName="ngModel">  <div class="form-control-feedback"  \*ngIf="lastName.errors && (lastName.dirty || lastName.touched)">  <p \*ngIf="lastName.errors.required">Last name is required</p>  </div>  </div>  </fieldset>    <div class="form-group"  [ngClass]="{  'has-danger': email.invalid && (email.dirty || email.touched),  'has-success': email.valid && (email.dirty || email.touched)  }">  <label>Email</label>  <input type="email"  class="form-control"  name="email"  [(ngModel)]="model.email"  required  pattern="[^ @]\*@[^ @]\*"  #email="ngModel">  <div class="form-control-feedback"  \*ngIf="email.errors && (email.dirty || email.touched)">  <p \*ngIf="email.errors.required">Email is required</p>  <p \*ngIf="email.errors.pattern">Email must contain at least the @ character</p>  </div>  </div>    <div class="form-group"  [ngClass]="{  'has-danger': password.invalid && (password.dirty || password.touched),  'has-success': password.valid && (password.dirty || password.touched)  }">  <label>Password</label>  <input type="password"  class="form-control"  name="password"  [(ngModel)]="model.password"  required  minlength="8"  #password="ngModel">  <div class="form-control-feedback"  \*ngIf="password.errors && (password.dirty || password.touched)">  <p \*ngIf="password.errors.required">Password is required</p>  <p \*ngIf="password.errors.minlength">Password must be at least 8 characters long</p>  </div>  </div>   <div class="form-group">  <label>Language</label>  <select class="form-control"  name="language"  [(ngModel)]="model.language">  <option value="">Please select a language</option>  <option \*ngFor="let lang of langs"  [value]="lang">{{lang}}  </option>  </select>  </div>   <button type="submit"  class="btn btn-primary"  [disabled]="f.invalid">Submit  </button>   <pre>{{f.value | json}}</pre> </form>  ` }) **class** TemplateFormComponent {   model: Signup = **new** Signup();  @ViewChild('f') form: **any**;   langs: **string**[] = [  'English',  'French',  'German',  ];   onSubmit() {  **if** (**this**.form.valid) {  console.log("Form Submitted!");  **this**.form.reset();  }  } }  @Component({  selector: 'app',  template: `<template-form></template-form>` }) **class** AppComponent { }   @NgModule({  imports: [  BrowserModule,  FormsModule  ],  declarations: [  AppComponent,  TemplateFormComponent  ],  bootstrap: [  AppComponent  ], }) **class** AppModule { }  platformBrowserDynamic().bootstrapModule(AppModule);  **이렇게 쓸수 있는것을** **할수있다** |