# TypeScript

## For

For..in: lấy chỉ số của từng phần tử

var someArray = [9, 2, 5];

for (var item in someArray) {

console.log(item); // 0,1,2

}

For..of: lấy giá trị của từng phần tử

var someArray = [9, 2, 5];

for (var item of someArray) {

console.log(item); // 9,2,5

}

## Class

class Point {

x: number; // default public

y: number;

constructor(x: number, y: number) {

this.x = x;

this.y = y;

}

add(point: Point) {

return new Point(this.x + point.x, this.y + point.y);

}

}

var p1 = new Point(0, 10);

var p2 = new Point(10, 20);

var p3 = p1.add(p2); // {x:10,y:30}

### abstract class

abstract class FooCommand {

abstract execute(): string;

}

class BarErrorCommand extends FooCommand {} //lỗi vl

class BarCommand extends FooCommand {

execute() {

return `Command Bar executed`;

}

}

const barCommand = new BarCommand();

barCommand.execute(); // Command Bar executed

### contructor

class Foo {

constructor(public x:number) {

}

}

Hàm tạo và dùng luôn

function Person(age) {

this.age = age;

this.growOld = () => {

this.age++;

}

}

## Tham số còn lại

Các tham số còn lại (được biểu thị bằng đối số cuối cùng) cho phép bạn nhanh chóng chấp nhận nhiều đối số trong hàm của mình và nhận chúng dưới dạng một mảng. Điều này được chứng minh trong ví dụ dưới đây....argumentName

* Đơn gianrL truyền thừa thì nó sẽ nhận hét vào param cuối và nó là 1 mảng( nếu không khai báo cụ thể kiểu dữ liệu)

function iTakeItAll ( first , second , ... allOthers ) {

bàn điều khiển . nhật ký ( allOthers ) ;

}

iTakeItAll ( 'foo' , 'bar' ) ; // []

iTakeItAll ( 'foo' , 'bar' , 'bas' , 'qux' ) ; // ['bas', 'qux']

## Cấu trúc mảng

Một câu hỏi lập trình phổ biến: "Làm thế nào để hoán đổi hai biến mà không sử dụng biến thứ ba?". Giải pháp TypeScript:

var x = 1 , y = 2 ;

[ x , y ] = [ y , x ] ;

bàn điều khiển . log ( x , y ) ; // 2,1

## Gán mảng

Toán tử spread cho phép bạn dễ dàng đặt *phiên bản mở rộng* của một mảng vào một mảng khác. Điều này được chứng minh trong ví dụ dưới đây:

var list = [ 1 , 2 ] ;

list = [ ... list , 3 , 4 ] ;

bàn điều khiển . log ( danh sách ) ; // [1,2,3,4]

Bạn có thể đặt mảng đã mở rộng ở bất kỳ vị trí nào và nhận được hiệu quả như mong đợi:

var list = [ 1 , 2 ] ;

list = [ 0 , ... list , 4 ] ;

bàn điều khiển . log ( danh sách ) ; // [0,1,2,4]

## Gán object

Vd1:

const point2D = {x: 1, y: 2};

/\*\* Create a new object by using all the point2D props along with z \*/

const point3D = {...point2D, z: 3};

Vd2:

const point2D = {x: 1, y: 2};

const anotherPoint3D = {x: 5, z: 4, ...point2D};

console.log(anotherPoint3D); // {x: 1, y: 2, z: 4}

const yetAnotherPoint3D = {...point2D, x: 5, z: 4}

console.log(yetAnotherPoint3D); // {x: 5, y: 2, z: 4}

Vd3:

const foo = {a: 1, b: 2, c: 0};

const bar = {c: 1, d: 2};

/\*\* Merge foo and bar \*/

const fooBar = {...foo, ...bar};

// fooBar is now {a: 1, b: 2, c: 1, d: 2}

# Angular

## Class vs style

(V-bind:””)

Set điều kiện = có ngặc{}

Không điều kiện = [] hoặc khôn cần dấu

<p [className]="'red'">students works is real!</p>

<p [ngClass]="{'red': true}">{{name}}</p>

-với nhiều class

< div [ngClass] = " ['example-class', 'other-class'] " > </ div >

<p [ngClass]="{'red': true, 'bg-cl': false}">{{name}}</p>

-với biến điều kiện

<p [ngClass]="{'red': true, 'bg-cl': isBackground}">{{name}}</p>

Or

<p [ngClass]="{'red': true}" [class.bg-cl] ="isBackground">{{name}}</p>

-style

<p [style.color]="'#cccccc'">attribute</p>

## Attribute

<img src='{{avatar}}' alt="">

Or

<p [style.color]="'#cccccc'">attribute</p>

<img

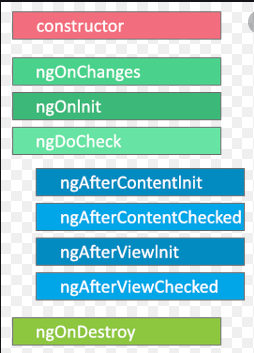
[src]='avatar' alt="">

* avatar là 1 biến

### hidden

<input [hidden]="isBackground"/>

## LifeCycle



Vd về vòng đơig angular

Angular is running in development mode. Call enableProdMode() to enable production mode.

Method A ngOnInit. value:

Method A ngAfterContentInit. value:

Method A ngAfterContentChecked. value:

Method Child ngOnChanges. value:

Method Child ngOnInit.

Method Child ngAfterContentInit. value:

Method Child ngAfterContentChecked. value:

Method Child ngAfterViewInit. value:

Method Child ngAfterViewChecked. value:

Method A ngAfterViewInit. value:

Method A ngAfterViewChecked. value:

Method A ngAfterContentChecked. value:

Method Child ngAfterContentChecked. value:

Method Child ngAfterViewChecked. value:

Method A ngAfterViewChecked. value:

# Event

(v-on: or @)

<button (click)="isBackground = !isBackground">ẩn/tắt</button>

## Directive

Attribute directives

Tạo mới 1 directive

ng generate directive highlight

Ta có cấu trúc bên trong:

content\_copyimport { [Directive](https://angular.io/api/core/Directive), [ElementRef](https://angular.io/api/core/ElementRef) } from '@angular/core';

@[Directive](https://angular.io/api/core/Directive)({

selector: '[appHighlight]' // tên chỉ thị để add vào html tag mong muốn

})

export class HighlightDirective {

constructor(el: [ElementRef](https://angular.io/api/core/ElementRef)) { el.nativeElement.style.backgroundColor = 'yellow'; }

}

Và đây là khi sử dụng:

<p appHighlight>Highlight me!</p>

# Pipe

Giống filter

Nó cung cấp các hàm như sau

## Chung

* sử dụng nhiều pipe 1 lúc

<p>Today is {{today | date | lowercase}}</p>

## Tạo pipe riêng

Tạo 1 file ts riêng nhé

Temp-converter.pipe.ts

import { Pipe } from '@angular/core';

@Pipe({

name: 'tempConverter'

})

export class ChangeStrPipe implements PipeTransform {

transform(value: string): string {

let newStr: string = 'htm' + value;

return newStr;

}

}

Thêm vào trong ngModul

**import** { TempConverterPipe } **from** './pipes/temp-converter.pipe';

@NgModule({

declarations: [

*// các Component, Directive, Pipe khác*

TempConverterPipe

],

imports: [...],

providers: [...],

*//...*

})

**export** **class** AppModule { }

## [AsyncPipe](https://angular.io/docs/ts/latest/api/common/index/AsyncPipe-pipe.html)

<span>Wait for it... {{ greeting | [async](https://angular.io/api/common/AsyncPipe) }}</span>

Đợi khi nào có greeting thì mới hiển thị với greeting là 1 lời hứa

this.greeting = new Promise<string>((resolve, reject) => { this.resolve = resolve; });

## [CurrencyPipe](https://angular.io/docs/ts/latest/api/common/index/CurrencyPipe-pipe.html)

## [DatePipe](https://angular.io/docs/ts/latest/api/common/index/DatePipe-pipe.html)

Pre-defined format options

| **Option** | **Equivalent to** | **Examples (given in en-US locale)** |
| --- | --- | --- |
| 'short' | 'M/d/yy, h:mm a' | 6/15/15, 9:03 AM |
| 'medium' | 'MMM d, y, h:mm:ss a' | Jun 15, 2015, 9:03:01 AM |
| 'long' | 'MMMM d, y, h:mm:ss a z' | June 15, 2015 at 9:03:01 AM GMT+1 |
| 'full' | 'EEEE, MMMM d, y, h:mm:ss a zzzz' | Monday, June 15, 2015 at 9:03:01 AM GMT+01:00 |
| 'shortDate' | 'M/d/yy' | 6/15/15 |
| 'mediumDate' | 'MMM d, y' | Jun 15, 2015 |
| 'longDate' | 'MMMM d, y' | June 15, 2015 |
| 'fullDate' | 'EEEE, MMMM d, y' | Monday, June 15, 2015 |
| 'shortTime' | 'h:mm a' | 9:03 AM |
| 'mediumTime' | 'h:mm:ss a' | 9:03:01 AM |
| 'longTime' | 'h:mm:ss a z' | 9:03:01 AM GMT+1 |
| 'fullTime' | 'h:mm:ss a zzzz' | 9:03:01 AM GMT+01:00 |

## [DecimalPipe](https://angular.io/docs/ts/latest/api/common/index/DecimalPipe-pipe.html)

<p>B: {{b | [currency](https://angular.io/api/common/CurrencyPipe):'CAD':'symbol':'4.2-2'}}</p>

4: trước dấy ,

2-2: sau dấu , từ 2 đến 2 ký tựu

<p>B: {{b | [currency](https://angular.io/api/common/CurrencyPipe):'CAD':'symbol':'4.1-5'}}</p>

4: trước dấy ,

1-5: sau dấu , từ 1 đến 5 ký tựu

## [JsonPipe](https://angular.io/docs/ts/latest/api/common/index/JsonPipe-pipe.html)

## [LowerCasePipe](https://angular.io/docs/ts/latest/api/common/index/LowerCasePipe-pipe.html)

## [PercentPipe](https://angular.io/docs/ts/latest/api/common/index/PercentPipe-pipe.html)

## [SlicePipe](https://angular.io/docs/ts/latest/api/common/index/SlicePipe-pipe.html)

## [TitleCasePipe](https://angular.io/docs/ts/latest/api/common/index/TitleCasePipe-pipe.html)

## [UpperCasePipe](https://angular.io/docs/ts/latest/api/common/index/UpperCasePipe-pipe.html)

# Selectors

## \*ng-if

### Kiểu 1:

<span \*ngIf="massageChild == undefined">{{massageChild | uppercase}}</span>

Thỏa mãn thì show

### Kiểu 2:

<p>skill: <span \*ngIf="messageSkill == '';else showMessageSkill">không có dữ liệu</span></p>

<ng-template #showMessageSkill>messageSkill</ng-template>

Else thì show #showMessageSkill đúng thì hiện không có dữ liệu

* # #nullMessageSkill Đặt nó ở đâu thì đặt

Kiểu 3: là kiểu 2 dạng viết tắt

<p>skill: <span \*ngIf="messageSkill == '';then nonMessageSkill else showMessageSkill"></span></p>

<ng-template #nonMessageSkill>không có dữ liệu</ng-template>

<ng-template #showMessageSkill>messageSkill</ng-template>

<ng-template> Được truyền vào bên trong <span>

Lúc này không thể thêm bất cứ thứ gì khác vào bên trong <span> được

# Lắng nghe sự kiện resize

## Sử dụng @HostListener

@HostListener("window:resize", [])

  private onResize() {

    setTimeout(function(){

      $(".size-screen").css("background-color","red");

    }, 300);

    $(".size-screen").css("background-color","green");

    this.screen.sizeX = $(window).width();

  }

## ResizeObserver

Resize đối với các loại tag

Install cái này đã: npm i @juggle/resize-observer

Vd:

  ngAfterViewInit() {

    const me= this;

    const resizeObserver = new ResizeObserver((entries) =>{

      for (let entry of entries) {

        me.screen.sizeX = entry.contentRect.width;

        me.screen.sizeY = entry.contentRect.height;

      }

    });

    const els = document.querySelectorAll("body");

    resizeObserver.observe(els[0]);

  }

Nhớ phải dùng querySelectorAll để truyền els[0]

# @HostListenter

Mẫu:

@[HostListener](https://angular.io/api/core/HostListener)(DomEvent, ['$event.target']) functionName(btn) { …}

Vd:  
@[HostListener](https://angular.io/api/core/HostListener)('window:keydown', ['$event'])

handleKeyDown(event: KeyboardEvent) { …}

- DomEvent: thẻ dom:sự kiện sử lý

- functionName: hàm thực thi function . (thường là hàm rỗng và được định nghĩa lại trong ngOnInit())

This.handleKeyDown = function(event: KeyboardEvent){…};

\*Có thể chỉ định lắng nghe phím từ bàn phím

vd: chỉ lắng nghe sự kiện Ctrl + S

@[HostListener](https://angular.io/api/core/HostListener)('window:keydown.control.s', ['$event'])

handleKeyDown(event: KeyboardEvent) { …}

or: @HostListener('keydown.shift.tab', ['$event'])

# View Child

Vd 1: dùng như 1 componet để gọi hàm hoặc lấy dữ liệu

@ViewChild('primaryColorSample')

sample: ColorSampleComponent;

vd2:

# ngZone

tổng quan: đối với vue hoặc react khi thay đổi bất cứ thứ gì dữ liệu sẽ được render vào dom ảo trước khi render lên dom thật. Sẽ có 1 cơ chế so sánh giữa dom ảo và dom thật để render, khi này sự ưu việt được thể hiện so với angular vì bản chất angular lắng nghe và render liên tục dù là 1 thay đổi nhỏ.

Lợi dụng điều này ta có thể chạy thay đổi 1 số dữ liệu là không

Khi Sử dụng **jquery** với thường là không gây ảnh hưởng đến dữ liệu bên trong các component. Lợi dụng điều này ta có thể đặt các câu lệnh **jquery** bên trong runOutsideAngular() .Khi đó cho dù có thay đổi dữ liệu angular cũng sẽ không tự động cập nhật lên html cho đến khi kết thúc khối lệnh trong runOutsideAngular

* Khi ta dùng j**query** hoặc **setTimeout** ,**setInterval CHU**

zone.run

example:

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

@Component({

  selector: 'my-app',

  template: `

  In Angular zone: {{timestampNgZone}}<br>

  Outside Angular zone: {{timestampOutsideNgZone}}<br><br>

  <br />

  Change detection only runs for the user click, but not for the async callback. So, the second binding is out-of-date because it has the last value when re-evaluated after the next user click.

  `

})

export class AppComponent  {

  timestampNgZone: string | Date =  'never';

  timestampOutsideNgZone: string | Date =  'never';

  constructor(private ngZone: NgZone) {

  }

  @HostListener('pointerdown')

  onPointerdown() {

    const timestamp = new Date();

    this.timestampNgZone = timestamp

    this.ngZone.runOutsideAngular(() => {

      setTimeout(() => this.timestampOutsideNgZone = timestamp);

    }

    )

  }

}

=>

In Angular zone: Tue May 11 2021 00:03:33 GMT+0700 (Indochina Time)  
Outside Angular zone: Tue May 11 2021 00:03:29 GMT+0700 (Indochina Time)

Khi click: cả 2 biến cùng được cập nhật. nhưng timestampOutsideNgZone được chạy trong runOutsideAngular nên không được cập nhật trên html. Mặc dù trong file .ts giá trị của nó tương đương timestampNgZone .Giá trị hiển thị runOutsideAngular chính là giá trị của lần click trước. Ngay khi click. zone đã so sánh value có sự khác nhau đối với runOutsideAngular Nó cập nhật lên html giá trị đang có. Nhưng giá trị mới của lần ấn này lại bị sẽ được hiển thị ở lần ấn kế tiếp

Lưu ý: khi thực hiện click là đã thông báo có sự thay đổi. vì thời gian thông báo tương đương với thời gian thay đổi trong runOutsideAngular Nên phải dùng setTimeOut( hoặc setInterval) đẻ **zone** không nhận ra sự thay đổi này

1 example khác: https://stackblitz.com/edit/ngzone-demo?file=src%2Fapp%2Fng-zone.component.ts

// Loop outside of the Angular zone

  // so the UI DOES NOT refresh after each setTimeout cycle

  processOutsideOfAngularZone() {

    this.label = 'outside';

    this.progress = 0;

    this.\_ngZone.runOutsideAngular(() => {

      this.\_increaseProgress(() => {

        // reenter the Angular zone and display done

        this.\_ngZone.run(() => { console.log('Outside Done!'); });

      });

    });

  }

  \_increaseProgress(doneCallback: () => void) {

    this.progress += 1;

    console.log(`Current progress: ${this.progress}%`);

    if (this.progress < 100) {

      window.setTimeout(() => this.\_increaseProgress(doneCallback), 10);

    } else {

      doneCallback();

    }

  }

increaseProgress được gọi lần đầu tiên tại thời điển tương đương 1 sự kiện gọi đến processOutsideOfAngularZone Nên htmlsẽ hiện 1%. Từ đệ quy hàm đó trở đi trong setTimeOut thì html sẽ không hiện process nữa và html sẽ hiện 100% sau khi chạy ra khỏi runOutsideAngular

# detechange

# select

tương tự như slot trong vuejs

**link tham khảo: https://scotch.io/tutorials/angular-2-transclusion-using-ng-content**