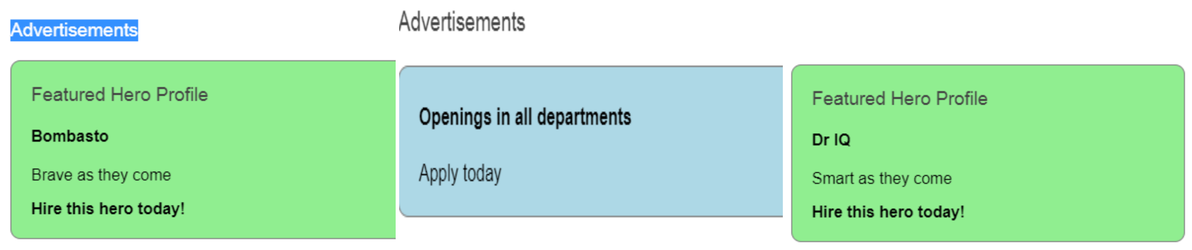
Advertisements

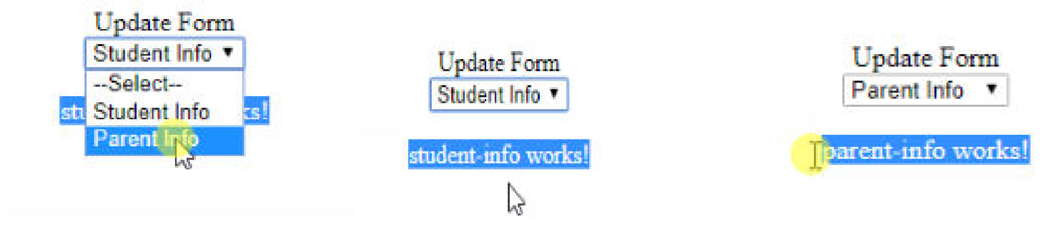


****

http://www.sahosofttutorials.com/Course/Angular7/200/

Dynamic component loader

If we give like this <app-comp> all will view but Requirement like load component when click or select



Ap.module.ts

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

import { AppComponent } from './app.component';
import { StudentInfoComponent } from './student-info/student-info.component';
import { ParentInfoComponent } from './parent-info/parent-info.component';

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

app.component.html

```
<div class="box-header with-border">
  <div class="form-group voffset row">
      <label for="" class="vertical-label col-sm-4 col-md-8 text-right"> Update Form</label>
      <div class="col-sm-8 col-md-4">
          <select (change)="selectName($event.target.value);">
                  <option value="0">--Select--</option>
               <option [value]="obj.Id" *ngFor="let obj of data">
                   {{obj.Name}}
            </option>
       </select>
      </div>
  </div>
  <div class="form-group voffset row">
  <template #loadComponent>
  </template>
  </div>
</div>
```

App.component.ts

```
import {
  Component,
  ViewChild,
   ViewContainerRef,
   ComponentFactoryResolver,
   ComponentRef,
   ComponentFactory
} from '@angular/core';
import { StudentInfoComponent } from './student-info/student-info.component';
import { ParentInfoComponent } from './parent-info/parent-info.component';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app';
  componentRef: any;
  @ViewChild('loadComponent', { read: ViewContainerRef }) entry: ViewContainerRef;
  constructor(private resolver: ComponentFactoryResolver) { }
  createComponent(Id: number) {
    this.entry.clear();
    if (Id == 1) {
        const factory = this.resolver.resolveComponentFactory(StudentInfoComponent);
        this.componentRef = this.entry.createComponent(factory);
    } else if (Id == 2) {
      const factory = this.resolver.resolveComponentFactory(ParentInfoComponent);
       this.componentRef = this.entry.createComponent(factory);
    }
    this.componentRef.instance.message = "Called by appComponent";
  }
  destroyComponent() {
    this.componentRef.destroy();
  }
  data = [
    {
      "Id": 1,
      "Name": "Student Info"
    },
    {
      "Id": 2,
      "Name": "Parent Info"
    }
  ]
  selectName(id : number) {
    this.createComponent(id);
  }
}
```

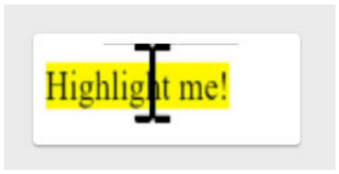
https://stackblitz.com/angular/vvmmveykakg?file=src%2Fapp%2Fhighlight.directive.ts

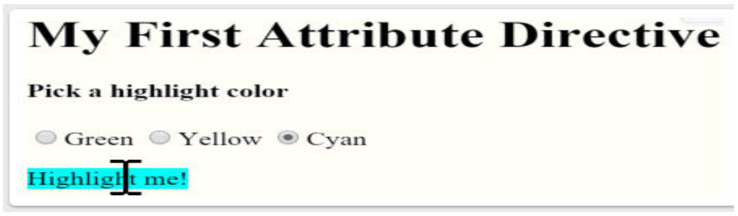
Attribute directives

mouse enters or leaves Text color has to change

color appears when the pointer hovers over the paragraph element and disappears as the pointer moves out.

on radio button selection color has to change





Step1

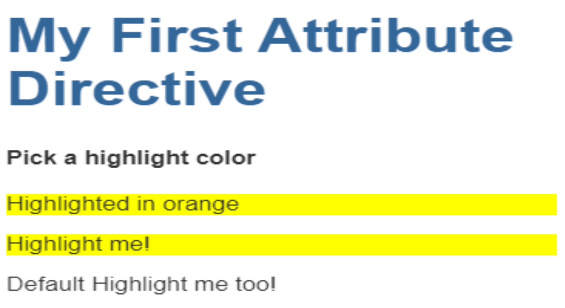
```
import { Directive, ElementRef } from '@angular/core';

@Directive({ selector: '[appHighlight]' })

export class HighlightDirective {

constructor(el: ElementRef) {

 el.nativeElement.style.backgroundColor = 'yellow';

 }

 };

App.comp.html

<p appHighlight>Highlight me!</p>
```

Step2

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

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {

  constructor(private el: ElementRef) {
    this.highlight('yellow')
   }
   @Input() defaultColor: string;

  @Input('appHighlight') highlightColor: string;

@HostListener('mouseenter') OnMouseEnter(){
  this.highlight(this.highlightColor = 'red' )
}
@HostListener('mouseleave') OnMouseLeave(){
  this.highlight(null)
}
  private highlight(color: string) {
    this.el.nativeElement.style.backgroundColor = color;
  }
}
```

Html

```
<h4>Pick a highlight color</h4>

<p appHighlight="orange">Highlighted in orange</p>
<p [appHighlight]="color">Highlight me!</p>
<p [appHighlight]="color" defaultColor="violet">
  Default Highlight me too!
</p>
```



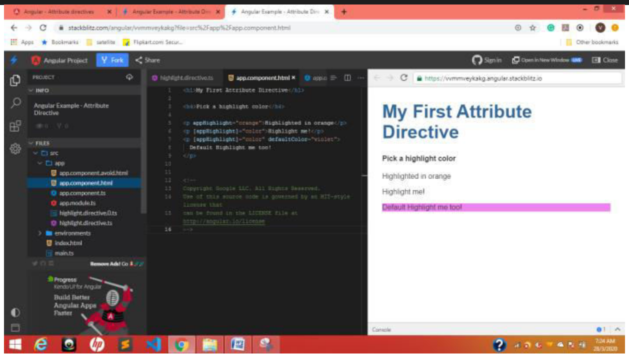
My First Attribute Directive

Pick a highlight color

Highlighted in orange

Highlight me!

Default Highlight me too!



```
<h1>My First Attribute Directive</h1>

<h4>Pick a highlight color</h4>

<p appHighlight="orange">Highlighted in orange</p>
<p [appHighlight]="color">Highlight me!</p>
<p [appHighlight]="color" defaultColor="violet">
 Default Highlight me too!
</p>


<!--
Copyright Google LLC. All Rights Reserved.
Use of this source code is governed by an MIT-style
license that
can be found in the LICENSE file at
http://angular.io/license
-->
```

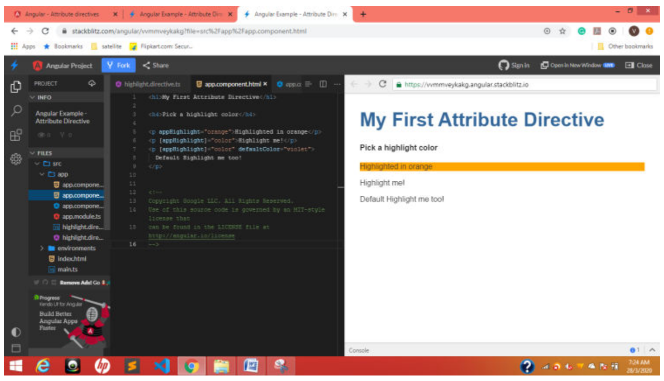
My First Attribute Directive

Pick a highlight color

Highlighted in orange

Highlight me!

Default Highlight me too!



```
<h1>My First Attribute Directive</h1>

<h4>Pick a highlight color</h4>

<p appHighlight="orange">Highlighted in orange</p>
<p [appHighlight]="color">Highlight me!</p>
<p [appHighlight]="color" defaultColor="violet">
 Default Highlight me too!
</p>


<!--
Copyright Google LLC. All Rights Reserved.
Use of this source code is governed by an MIT-style
license that
can be found in the LICENSE file at
http://angular.io/license
-->
```

My First Attribute Directive

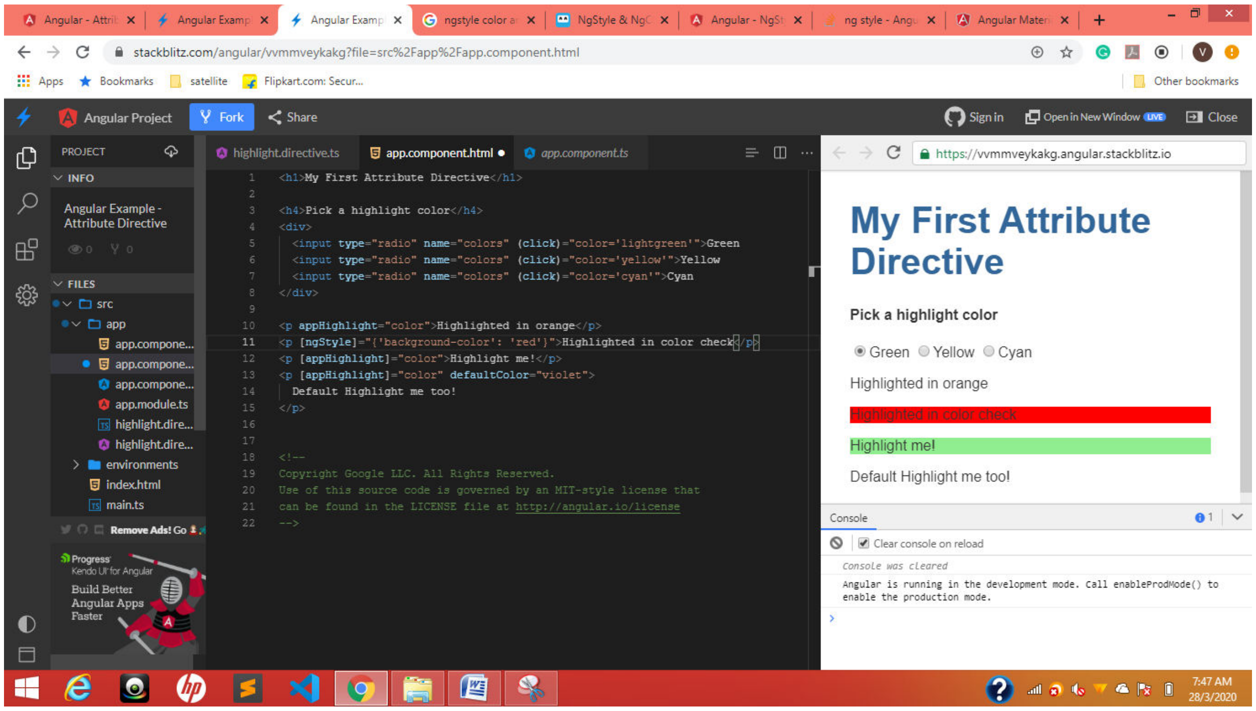
Pick a highlight color

Highlighted in orange

Highlight me!

Default Highlight me too!

Three different colors



```html
<h1>My First Attribute Directive</h1>

<h4>Pick a highlight color</h4>
<div>
  <input type="radio" name="colors" (click)="color='lightgreen'">Green
  <input type="radio" name="colors" (click)="color='yellow'">Yellow
  <input type="radio" name="colors" (click)="color='cyan'">Cyan
</div>

<p appHighlight="color">Highlighted in orange</p>
<p [ngStyle]="{'background-color': 'red'}">Highlighted in color check</p>
<p [appHighlight]="color">Highlight me!</p>
<p [appHighlight]="color" defaultColor="violet">
  Default Highlight me too!
</p>


<!--
Copyright Google LLC. All Rights Reserved.
Use of this source code is governed by an MIT-style license that
can be found in the LICENSE file at http://angular.io/license
-->
```

My First Attribute Directive

Pick a highlight color

● Green ○ Yellow ○ Cyan

Highlighted in orange

Highlighted in color check

Highlight me!

Default Highlight me too!

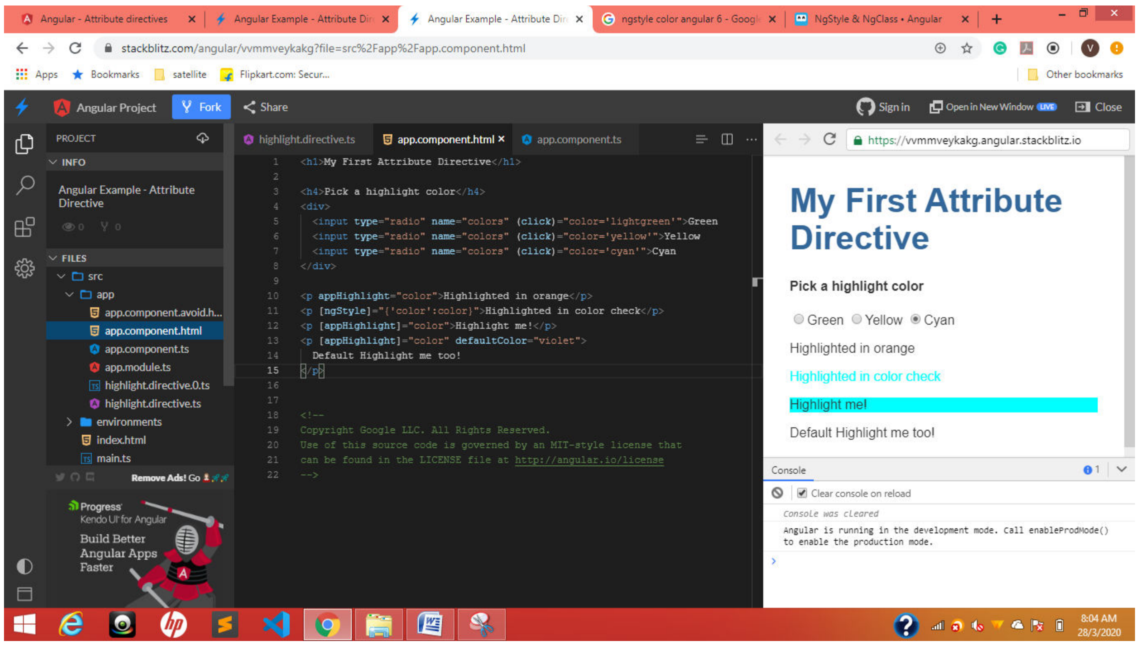
```
Console was cleared
Angular is running in the development mode. Call enableProdMode() to enable the production mode.
```

[ngStyle]="{'background-color': 'red'}"

[ngStyle]="{'key': 'value'}"

ngStyle is objective

```html
<p appHighlight="color">Highlighted in orange</p>
<p [ngStyle]="{'color':color}">Highlighted in color check</p>
<p [appHighlight]="color">Highlight me!</p>
<p [appHighlight]="color" defaultColor="violet">
  Default Highlight me too!
</p>
```



```html
<h1>My First Attribute Directive</h1>

<h4>Pick a highlight color</h4>
<div>
  <input type="radio" name="colors" (click)="color='lightgreen'">Green
  <input type="radio" name="colors" (click)="color='yellow'">Yellow
  <input type="radio" name="colors" (click)="color='cyan'">Cyan
</div>

<p appHighlight="color">Highlighted in orange</p>
<p [ngStyle]="{'color':color}">Highlighted in color check</p>
<p [appHighlight]="color">Highlight me!</p>
<p [appHighlight]="color" defaultColor="violet">
  Default Highlight me too!
</p>


<!--
Copyright Google LLC. All Rights Reserved.
Use of this source code is governed by an MIT-style license that
can be found in the LICENSE file at http://angular.io/license
-->
```

My First Attribute Directive

Pick a highlight color

Green Yellow [✓ Cyan]

Highlighted in orange

Highlighted in color check

Highlight me!

Default Highlight me too!

```
Console was cleared
Angular is running in the development mode. Call enableProdMode()
to enable the production mode.
```

Structural directives

https://stackblitz.com/angular/ygjxdrnerky?file=src%2Fapp%2Fapp.component.html

ngIf

`*ngIf="true"`   `[style.display]="'block'"`   `[style.display]="'none'"`

```html
<p *ngIf="true">
  Expression is true and ngIf is true.
  This paragraph is in the DOM.
</p>
<p *ngIf="false">
  Expression is false and ngIf is false.
  This paragraph is not in the DOM.
</p>

<p [style.display]="'block'">
  Expression sets display to "block".
  This paragraph is visible.
</p>
<p [style.display]="'none'">
  Expression sets display to "none".
  This paragraph is hidden but still in the DOM.
</p>
```

Toggle

Now conditionally exclude a *select* `<option>` with `<ng-container>`.

The *<ng-template>*

The <ng-template> is an Angular element for rendering HTML. It is never displayed directly. In fact, before rendering the view, Angular *replaces* the `<ng-template>` and its contents with a comment.

If there is no structural directive and you **merely wrap some elements** in a `<ng-template>`, those elements disappear. That's the fate of the middle "Hip!" in the phrase "Hip! Hip! Hooray!".

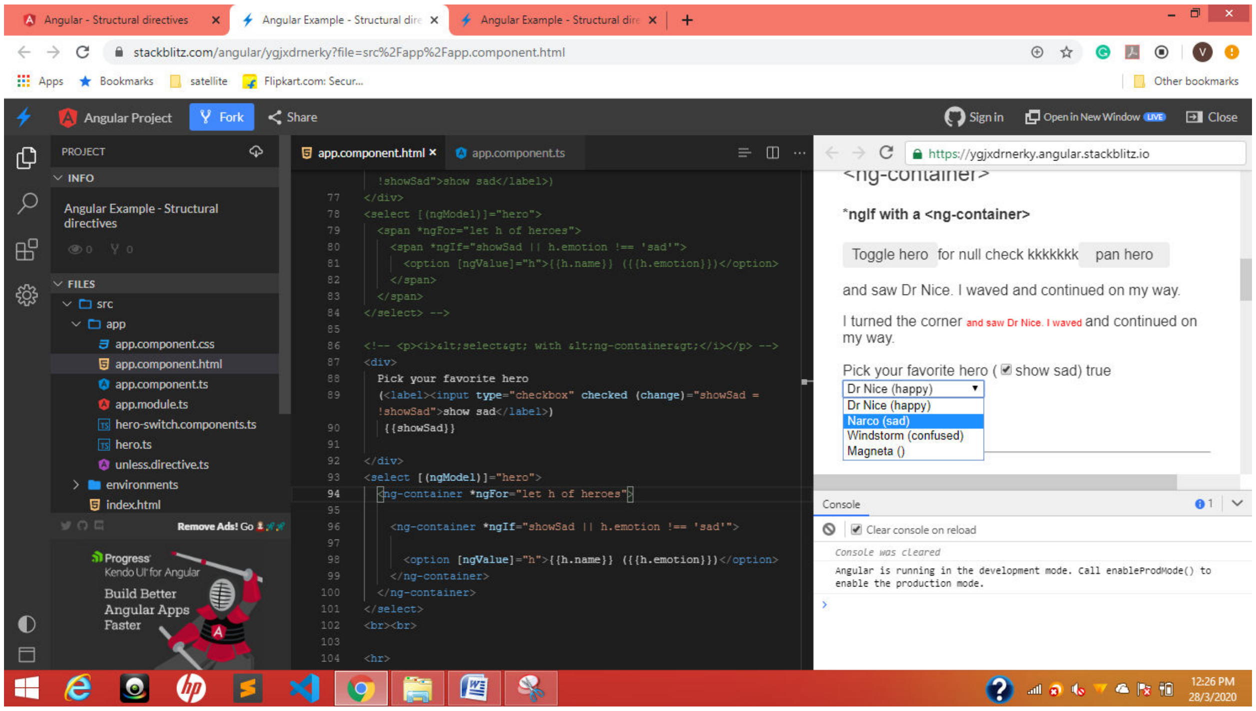
```
*ngIf="hero"
*ngIf="!hero"
```

```
<button (click)="hero = hero ? kiran : heroes[0]">Toggle hero</button>
<span *ngIf='kiran? kiran : hero'>for null check kkkkkkk</span>
```

Ts.file

```
kiran:boolean;
```

if selected only then show in dropdown <ng-container>



```
  !showSad">show sad</label>)
</div>
<select [(ngModel)]="hero">
  <span *ngFor="let h of heroes">
    <span *ngIf="showSad || h.emotion !== 'sad'">
      <option [ngValue]="h">{{h.name}} ({{h.emotion}})</option>
    </span>
  </span>
</select> -->

<!-- <p><i>&lt;select&gt; with &lt;ng-container&gt;</i></p> -->
<div>
  Pick your favorite hero
  (<label><input type="checkbox" checked (change)="showSad =
  !showSad">show sad</label>)
   {{showSad}}

</div>
<select [(ngModel)]="hero">
  <ng-container *ngFor="let h of heroes">

    <ng-container *ngIf="showSad || h.emotion !== 'sad'">

      <option [ngValue]="h">{{h.name}} ({{h.emotion}})</option>
    </ng-container>
  </ng-container>
</select>
<br><br>

<hr>
```

<ng-container>

*ngIf with a <ng-container>

Toggle hero for null check kkkkkkk pan hero

and saw Dr Nice. I waved and continued on my way.

I turned the corner and saw Dr Nice. I waved and continued on my way.

Pick your favorite hero (✓ show sad) true

Dr Nice (happy)
Dr Nice (happy)
Narco (sad)
Windstorm (confused)
Magneta ()

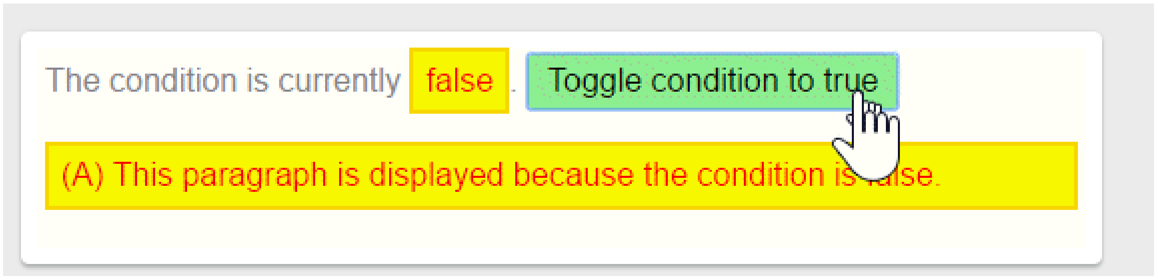
```
Console was cleared
Angular is running in the development mode. Call enableProdMode() to
enable the production mode.
```

```
<select [(ngModel)]="hero">
  <ng-container *ngFor="let h of heroes">
    <ng-container *ngIf="showSad || h.emotion !== 'sad'">
      <option [ngValue]="h">{{h.name}} ({{h.emotion}})</option>
    </ng-container>
  </ng-container>
</select>
```

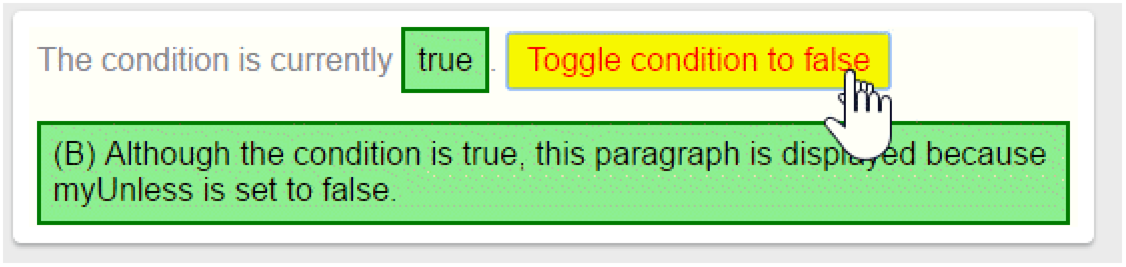
TemplateRef and *ViewContainerRef*

The *appUnless* property

UnlessDirective


The condition is currently false. Toggle condition to true

(A) This paragraph is displayed because the condition is false.


The condition is currently true. Toggle condition to false

(B) Although the condition is true, this paragraph is displayed because myUnless is set to false.

Just toggle *appUnless

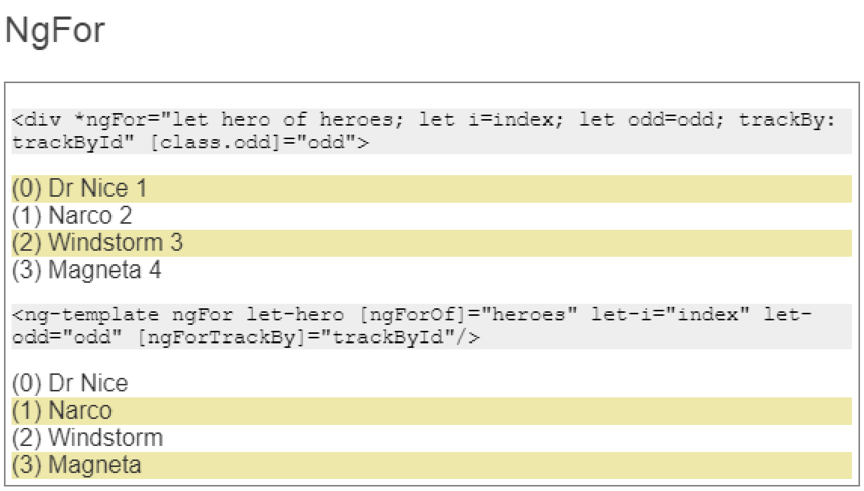
```
<!-- start -->
<button (click)="condition = !condition"  > Toggle condition to
condition ? 'false' : 'true'}} </button>
 <p *appUnless="condition"> (A) This paragraph is displayed because the condition is false. </p>
<p *appUnless="!condition" > (B) Although the condition is true, this paragraph is displayed because appUnless
is set to false. </p>
<p *appUnless="condition">Show this sentence unless the condition is true.</p>

<p *appUnless="condition" class="code unless">
  (A) &lt;p *appUnless="condition" class="code unless"&gt;
</p>
```

```
<!-- end -->
```

Html

```
<h2 id="appUnless">UnlessDirective</h2>

 <p> The condition is currently <span [ngClass]="{ 'a': !condition, 'b': condition, 'unless': true }">{{condition}}</span>.

<button (click)="condition = !condition" [ngClass] = "{ 'a': condition, 'b': !condition }" > Toggle condition to

condition ? 'false' : 'true'}} </button> </p>

 <p *appUnless="condition" class="unless a"> (A) This paragraph is displayed because the condition is false.
</p>

<p *appUnless="!condition" class="unless b"> (B) Although the condition is true, this paragraph is displayed
because appUnless is set to false. </p>
```

ngFor odd even text highlet



NgFor

```
<div *ngFor="let hero of heroes; let i=index; let odd=odd; trackBy:
trackById" [class.odd]="odd">
```

(0) Dr Nice 1
(1) Narco 2
(2) Windstorm 3
(3) Magneta 4

```
<ng-template ngFor let-hero [ngForOf]="heroes" let-i="index" let-
odd="odd" [ngForTrackBy]="trackById"/>
```

(0) Dr Nice
(1) Narco
(2) Windstorm
(3) Magneta

```
[class.odd]="even">
[class.odd]="odd"
```

```
<p class="code">&lt;div *ngFor="let hero of heroes; let i=index; let odd=odd; trackBy: trackById" [class.odd]="
odd"&gt;</p>
<div
*ngFor="let hero of heroes;
let i=index;
let odd=odd;
let even=even;
```

```
trackBy: trackById"
[class.odd]="even">
({{i}}) {{hero.name}} {{hero.id}}
</div>
<p class="code">&lt;ng-template ngFor let-hero [ngForOf]="heroes" let-i="index" let-
odd="odd" [ngForTrackBy]="trackById"/&gt;</p>
<ng-template ngFor let-hero [ngForOf]="heroes" let-i="index" let-odd="odd" [ngForTrackBy]="trackById">
  <div [class.odd]="odd">({{i}}) {{hero.name}}</div>
</ng-template>
```

NgSwitch

Many components in component.ts file

```
import { Component, Input } from '@angular/core';
import { Hero } from './hero';

@Component({
  selector: 'app-happy-hero',
  template: `Wow. You like {{hero.name}}. What a happy hero ... just like you.`
})
export class HappyHeroComponent {
  @Input() hero: Hero;
}

@Component({
  selector: 'app-sad-hero',
  template: `You like {{hero.name}}? Such a sad hero. Are you sad too?`
})
export class SadHeroComponent {
  @Input() hero: Hero;
}

@Component({
  selector: 'app-confused-hero',
  template: `Are you as confused as {{hero.name}}?`
})
export class ConfusedHeroComponent {
  @Input() hero: Hero;
}

@Component({
  selector: 'app-unknown-hero',
  template: `{{message}}`
})
export class UnknownHeroComponent {
  @Input() hero: Hero;
  get message() {
    return this.hero && this.hero.name ?
      `${this.hero.name} is strange and mysterious.` :
      'Are you feeling indecisive?';
  }
}

export const heroSwitchComponents =
  [ HappyHeroComponent, SadHeroComponent, ConfusedHeroComponent, UnknownHeroComponent ];
```

```
/*
Copyright Google LLC. All Rights Reserved.
Use of this source code is governed by an MIT-style license that
can be found in the LICENSE file at http://angular.io/license
*/
```

```
<p>
  <label *ngFor="let h of heroes">
    <input type="radio" name="heroes" [(ngModel)]="hero" [value]="h">{{h.name}}
  </label>
  <label><input type="radio" name="heroes" (click)="hero = null">None of the above</label>
</p>

<h4>NgSwitch</h4>

<div [ngSwitch]="hero?.emotion">
  <app-happy-hero    *ngSwitchCase="'happy'"    [hero]="hero"></app-happy-hero>
  <app-sad-hero      *ngSwitchCase="'sad'"      [hero]="hero"></app-sad-hero>
  <app-confused-hero *ngSwitchCase="'confused'" [hero]="hero"></app-confused-hero>
  <app-unknown-hero  *ngSwitchDefault           [hero]="hero"></app-unknown-hero>
</div>
```

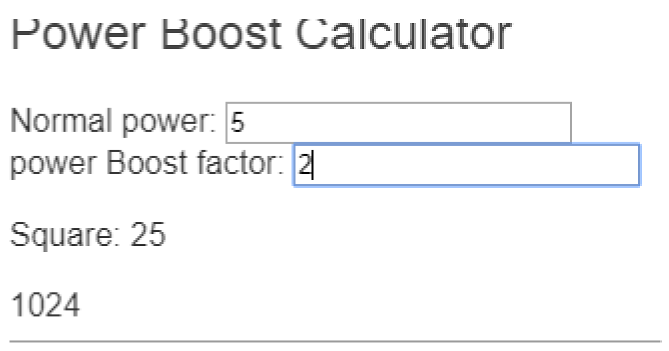
Pipes

Power Booster

2^10=1024

```
 <p>Super power boost: {{2 | exponentialStrength: 10}}</p>
```

https://stackblitz.com/angular/yordgxmxrgo?file=src%2Fapp%2Fpower-boost-calculator.component.ts


Power Boost Calculator

Normal power: 5

power Boost factor: 2

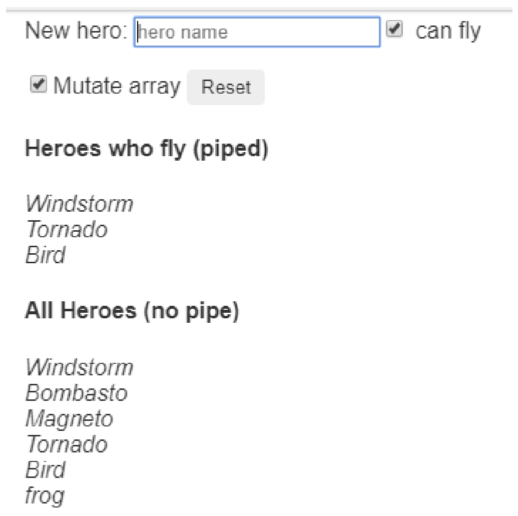
Square: 25

1024

```

@Component({
  selector: 'app-power-boost-calculator',
  template: `
    <h2>Power Boost Calculator</h2>
    <div>Normal power: <input [(ngModel)]="power"></div>
    <div>Boost factor: <input [(ngModel)]="factor"></div>
    <p>
      Super Hero Power: {{power | exponentialStrength: factor}}
    </p>
  `
})
export class PowerBoostCalculatorComponent {
  power = 5;
  factor = 1;
}
```

```
enterValue =Math.pow(2, 10);;
```

https://stackblitz.com/angular/yordgxmxrgo?file=src%2Fapp%2Fflying-heroes.component.html



Heroes who fly (piped)

Windstorm
Tornado
Bird

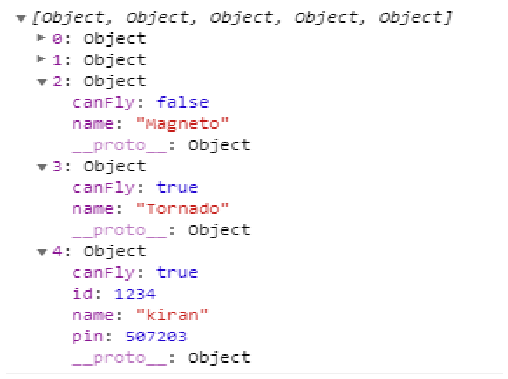
All Heroes (no pipe)

Windstorm
Bombasto
Magneto
Tornado
Bird
frog

On click of check box entered value should go to respective category

```
<h2>{{title}}</h2>
<p>
New hero:
  <input type="text" #box
          (keyup.enter)="addHero(box.value); box.value=''"
          placeholder="hero name">
  <input id="can-fly" type="checkbox" [(ngModel)]="canFly"> can fly
</p>
<p>
  <input id="mutate" type="checkbox" [(ngModel)]="mutate">Mutate array
  <button (click)="reset()">Reset</button>
</p>

<h4>Heroes who fly (piped)</h4>
<div id="flyers">
  <div *ngFor="let hero of (heroes | flyingHeroes)">
    {{hero.name}}
  </div>
</div>

<h4>All Heroes (no pipe)</h4>
<div id="all">
  <div *ngFor="let hero of heroes">
    {{hero.name}}
  </div>
```

```
</div>
```



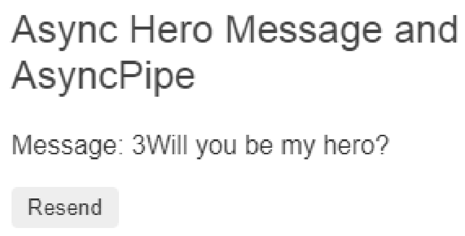
```
▼ [Object, Object, Object, Object, Object]
  ► 0: Object
  ► 1: Object
  ▼ 2: Object
      canFly: false
      name: "Magneto"
      __proto__: Object
  ▼ 3: Object
      canFly: true
      name: "Tornado"
      __proto__: Object
  ▼ 4: Object
      canFly: true
      id: 1234
      name: "kiran"
      pin: 507203
      __proto__: Object
```

ID, pin of key and values added dynamically

```
import { Component }              from '@angular/core';
import { HEROES }                 from './heroes';

@Component({
  selector: 'app-flying-heroes',
  templateUrl: './flying-heroes.component.html',
  styles: ['#flyers, #all {font-style: italic}']
})
export class FlyingHeroesComponent {
  heroes: any[] = [];
  canFly = true;
  mutate = true;
  title = 'Flying Heroes (pure pipe)';

  constructor() { this.reset(); }

  addHero(name: string) {

    name = name.trim();
    if (!name) { return; }
      let hero = {name, canFly: this.canFly,  id:1234, pin:507203};
    if (this.mutate) {
    // Pure pipe won't update display because heroes array reference is unchanged
    // Impure pipe will display
    this.heroes.push(hero);
    } else {
      // Pipe updates display because heroes array is a new object
      // this.heroes.push(hero);
      this.heroes = this.heroes.concat(hero);

    }
  }

  reset() { this.heroes = HEROES.slice(); }
}
```

Async Hero Message and AsyncPipe

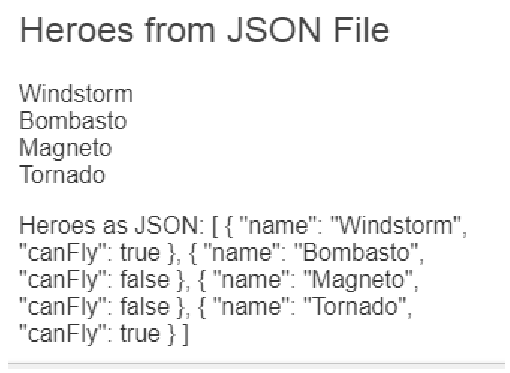
Send message with async pipe


Async Hero Message and AsyncPipe

Message: 3Will you be my hero?

```
import { Component } from '@angular/core';
import { Observable, interval } from 'rxjs';
import { map, take } from 'rxjs/operators';

@Component({
  selector: 'app-hero-message',
  template: `
    <h2>Async Hero Message and AsyncPipe</h2>
    <p>Message: {{ message$ | async }}</p>
    <button (click)="resend()">Resend</button>`,
})
export class HeroAsyncMessageComponent {
  message$: Observable<string>;

  private messages = [
    '1You are my hero!',
    '2You are the best hero!',
    '3Will you be my hero?'
  ];
  constructor() { this.resend(); }
  resend() {
    this.message$ = interval(1000).pipe(
      map(i => this.messages[i]),
      take(this.messages.length)
    );
  }
}
```

How to convert data type into json format


Heroes from JSON File

Windstorm
Bombasto
Magneto
Tornado

Heroes as JSON: [{ "name": "Windstorm", "canFly": true }, { "name": "Bombasto", "canFly": false }, { "name": "Magneto", "canFly": false }, { "name": "Tornado", "canFly": true }]

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

@Component({
  selector: 'app-hero-list',
  template: `
    <h2>Heroes from JSON File</h2>

    <div *ngFor="let hero of ('assets/heroes.json' | fetch) ">
      {{hero.name}}
    </div>

    <p>Heroes as JSON:
      {{'assets/heroes.json' | fetch | json}}
    </p>`
})
export class HeroListComponent { }
```