# Dyrektywy w Angular

## Tworzenie serwisów Web 2.0

dr inż. Robert Perliński
rperlinski@icis.pcz.pl

Politechnika Częstochowska
Instytut Informatyki Teoretycznej i Stosowanej

25 maja 2020

# Plan prezentacji

# Dyrektywy w Angular

- Dyrektywa (ang. directive) to klasa, która może modyfikować strukturę drzewa DOM albo modyfikować atrybuty w drzewie DOM i dane w komponentach.
- Definicja klasy dyrektywy jest bezpośrednio poprzedzona dekoratorem `@Directive()`, który dosarcza do niej metadane.
- Angular dostarcza wiele wbudowanych dyrektyw, które rozpoczynaja się prefiksem "ng".
- Można tworzyć nowe dyrektywy, które implementują naszą własną funkcjonalność.

# Dyrektywy w Angular

Angular ma trzy rodzaje dyrektyw:

1. Komponenty
   - dyrektywy z szablonem
   - używają dekoratora `@Component()`, będącego rozszerzeniem dekoratora `@Directive()`, do powiązania szablonu z klasą
2. Dyrektywy strukturalne
   - zmieniają strukturę drzewa DOM
   - poprzez dodawanie albo usuwanie elementów z drzewa DOM
   - zmieniają strukturę widoku
3. Dyrektywy atrybutowe
   - zmieniają wygląd albo zachowanie elementu, komponentu albo innej dyrektywy

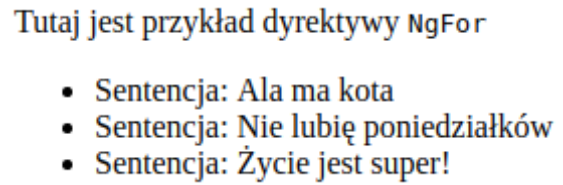
# Dyrektywy strukturalne

# Dyrektywy strukturalne

Dyrektywy strukturalne:

- są odpowiedzialne za strukturę kodu HTML, za jego układ
- tworzą albo przekształcają strukturę drzewa DOM poprzez dodawanie, usuwanie czy modyfikowanie jego elementów
- stosujemy do konkretnego elementu, który taka dyrektywa przekształca razem ze elementami podrzędnymi, potomkami w drzewie DOM
- łatwo rozpoznać, są poprzedzone gwiazdką (*) jeśli występują jako atrybuty, przykład:

```
<p>Tutaj jest przykład dyrektywy <code>NgFor</code></p>
<ul>
  <li *ngFor="let s of sentences">
    Sentencja: {{s}}
  </li>
</ul>
```




https://angular.io/guide/structural-directives

# Dyrektywy strukturalne

- Notacja gwiazdki (*) jest notacją przyjęta dla wygody i ciąg znaków, string jest bardziej mikroskładnią niż zwykłym wyrarzeniem szablonowym.
- Parser mikroskładni tłumaczy ten ciąg znaków w atrybuty znacznika `<ng-template>`, który otacza element gospodarza i jego potomków.
- Różne dyrektwy strukturalne robą co innego z szablonem gospodarza, mają na nim inne działanie.

- Nazwy dyrektywy pisane z dużej litery, np. `NgIf` czy `NgFor` dotyczą nazw klas danej dyrektywy (`UpperCamelCase`). Dotyczy to właściwości dyrektyw i ich funkcjonalności.
- Nazwy dyrektywy pisane z małej litery, np. `ngIf` czy `ngFor` dotyczą nazwy atrybutu danej dyrektywy (`lowerCamelCase`). Dotyczy to stosowania dyrektywy na elemencie HTML w szablonie.
- Do elementu można dodać wiele dyrektyw atrybutowych, ale **tylko jedną dyrektywę strukturalną**.

# Dyrektywy strukturalne wbudowane w Angular

Dyrektywy strukturalne wbudowane w Angular:

1. `NgIf`
2. `NgFor`
3. `NgSwitch`

Z dyrektywami struktualnymi związane są pojęcia:

- `Template Input Variables`
- `ng-template`
- `ng-container`

# Dyrektywa NgIf

Dyrektywa NgIf:

- najprostsza dyrektywa strukturalna
- pozwala na pojawienie się albo usunięcie całego fragmentu drzewa DOM zależnie od wartości logiczne wyrażenia
- mikroskładnia `ngIf`

```
<div *ngIf="hero" class="name">{{hero.name}}</div>
```

- zamieniona na szablon, element `<ng-template>`:

```
<ng-template [ngIf]="hero">
  <div class="name">{{hero.name}}</div>
</ng-template>
```

# Dyrektywa NgIf - przykład, kod szablonu

kod szablonu, plik *.html

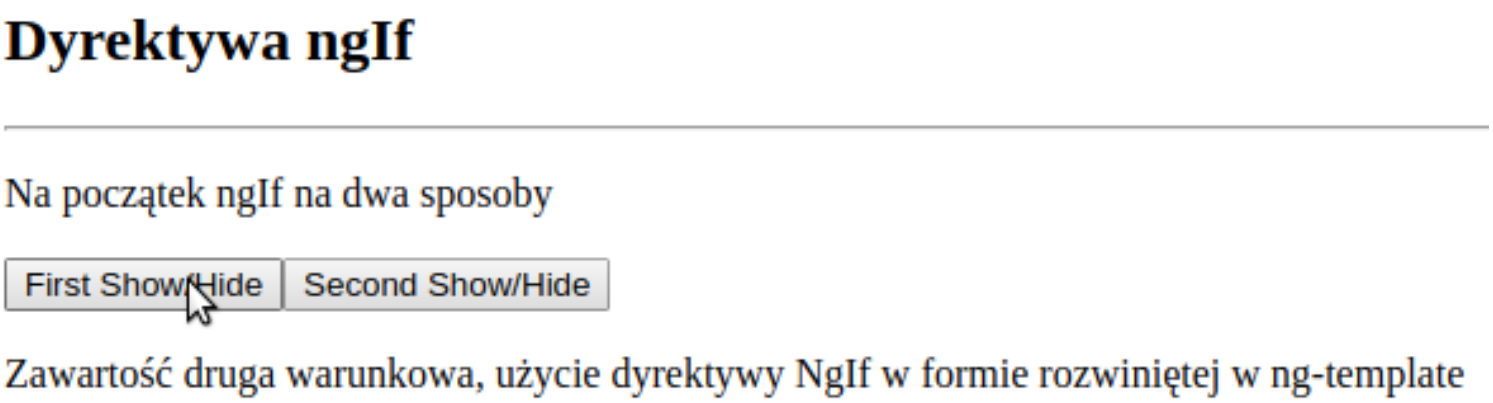
```
<h2>Dyrektywa ngIf</h2> <hr>
<p>Na początek ngIf na dwa sposoby</p>
<input type="button" (click)="toggleFirstIf()" name="sh1" value="First Show/Hide">
<input type="button" (click)="toggleSecondIf()" name="sh2" value="Second Show/Hide">

<div *ngIf="firstIf">
  <p>Zawartość pierwsza warunkowa, klasyczny użycie dyrektywy NgIf</p>
</div>
<ng-template [ngIf]="secondIf">
  <p>Zawartość druga warunkowa, użycie dyrektywy NgIf w formie rozwiniętej w ng-template</p>
</ng-template>
```

kod klasy komponentu, plik *.ts

```
@Component({
  selector: 'app-ng-if-directive',
  templateUrl: './ng-if-directive.component.html',
  styleUrls: ['./ng-if-directive.component.css']
})
export class NgIfDirectiveComponent implements OnInit {
  constructor() { }
  ngOnInit(): void { }
  firstIf = false;
  secondIf = false;

  toggleFirstIf() { this.firstIf = !this.firstIf; }
  toggleSecondIf() { this.secondIf = !this.secondIf; }
}
```

# Dyrektywa NgIf - przykład, wynikowy kod HTML

Wynik w przeglądarce:




Wymikowy kod HTML:

```
▼<app-ng-if-directive _ngcontent-lap-c12 _nghost-lap-c11>
   <h2 _ngcontent-lap-c11>Dyrektywa ngIf</h2>
   <hr _ngcontent-lap-c11>
   <p _ngcontent-lap-c11>Na początek ngIf na dwa sposoby</p>
   <input _ngcontent-lap-c11 type="button" name="sh1" value="First Show/Hide">
   <input _ngcontent-lap-c11 type="button" name="sh2" value="Second Show/Hide">
   <!--bindings={
     "ng-reflect-ng-if": "false"
   }-->
  ▼<p _ngcontent-lap-c11>
     "Zawartość druga warunkowa, użycie dyrektywy NgIf w formie rozwiniętej w ng-template"
   </p>
   <!--bindings={
     "ng-reflect-ng-if": "true"
   }-->
 </app-ng-if-directive>
```

# Dyrektywa NgFor

Dyrektywa NgFor:

- mikroskładnia `ngFor`

```
<div *ngFor="let hero of heroes; let i=index; let odd=odd; trackBy: trackById"
     [class.odd]="odd">
  ({{i}}) {{hero.name}}
</div>
```

- zamieniona na szablon, element `<ng-template>`:

```
<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>
```

# Dyrektywa NgSwitch

## Dyrektywa NgSwitch

- dodaje albo usuwa szablony kodu HTML (pokazuje albo ukrywa widoki) w zależności do tego, która instrukcja case zostanie dopasowana do wartości przekazanej w wyrażeniu `switch`
- jest w rzeczywistości zbiorem trzech współpracujących ze sobą dyrektyw: `NgSwitch`, `NgSwitchCase`, `NgSwitchDefault`

```
<container-element [ngSwitch]="switch_expression">
  <some-element *ngSwitchCase="match_expression_1">...</some-element>
  <some-element *ngSwitchCase="match_expression_2">...</some-element>
  ...
  <some-element *ngSwitchDefault>...</some-element>
</container-element>
```

- przykład dla bohaterów:

app.component.html

```
<div [ngSwitch]="hero?.emotion">
  <app-bohater-dobry    *ngSwitchCase="'dobry'"   [bohater]="bohater"></app-bohater-dobry>
  <app-bohater-zly      *ngSwitchCase="'zly'"     [bohater]="bohater"></app-bohater-zly>
  <app-bohater-zmienny  *ngSwitchCase="'zmienny'" [bohater]="bohater"></app-bohater-zmienny>
  <app-bohater-nieznany *ngSwitchDefault          [bohater]="bohater"></app-bohater-nieznany>
</div>
```

# Dyrektywa NgSwitch

Dyrektywa NgSwitch

- dyrektywa `[ngSwitch]` umieszczona w jakimś znaczniku będącym kontenerem (np. div, ng-container) zawiera wyrażenie, do kórego trzeba dopasować wartość
- renderowany jest każdy widok, który ma określoną wartość pasującą do wyrażenia
- jeśli nie ma żadnej pasującej wartości, renderowany jest widok spod dyrektywy `ngSwitchDefault`
- elementy w instrukcji `[NgSwitch]`, ale poza jakąkolwiek dyrektywą `ngSwitchCase` lub `ngSwitchDefault` są umieszczane w danym miejscu

# Dyrektywa NgSwitch - przykład

Przykład wykorzystania dyrektywy NgSwitch:

app.component.html

```
...
<button type="button" (click)="toggleChoice()">NgSwitch</button>
...
```

app.component.ts

```
export class AppComponent {
  ...
  sentences: Array<string> = [
    "Ala ma kota", "Nie lubię poniedziałków", "Życie jest super!"];
  choice: number = 0;

  toggleChoice() : void {
    this.choice++;
    if(this.choice>=4) this.choice = 0;
  }
}
```

# Dyrektywa NgSwitch - przykład

Przykład wykorzystania dyrektywy NgSwitch:

app.component.html

```
<p>Wartość zmiennej <var>choice</var>, od której zależy wybrany fragment widoku:
  <strong>{{choice}}</strong>
</p>
<article [ngSwitch]="choice">
  <div *ngSwitchCase="0">
    Pierwsza sentencja jest uczelniana, o Ali i jej kocie:<br>
    <textarea rows="3" cols="30">{{sentences[choice]}}</textarea>
  </div>
  <div *ngSwitchCase="1">
    Druga sentencja jest życiowa, taka o realiach:<br>
    <code>{{sentences[choice]}}</code>
  </div>
  <div *ngSwitchCase="2">
    Trzecia sentencja jest z Ducha Świętego, taka pełna radości:<br>
    <h3>{{sentences[choice]}}</h3>
  </div>
  <div *ngSwitchDefault>
    Ta sentencja jest wyświetlana, jeśli żadna inna nie została dopasowana:<br>
    <strong>Lorem ipsum...</strong>
  </div>
</article>
```

# Dyrektywa NgSwitch - przykład

Przykład wykorzystania dyrektywy `NgSwitch`

- wartość `choice` równa 0:

Wartość zmiennej *choice*, od której zależy wybrany fragment widoku: **0**

Pierwsza sentencja jest uczelniana, o Ali i jej kocie:

```
Ala ma kota
```

- wartość `choice` równa 1:

Wartość zmiennej *choice*, od której zależy wybrany fragment widoku: **1**

Druga sentencja jest życiowa, taka o realiach:

```
Nie lubię poniedziałków
```

- wartość `choice` równa 2:

Wartość zmiennej *choice*, od której zależy wybrany fragment widoku: **2**

Trzecia sentencja jest z Ducha Świętego, taka pełna radości:

**Życie jest super!**

- wartość `choice` różna od 0, 1, 2:

Wartość zmiennej *choice*, od której zależy wybrany fragment widoku: **3**

Ta sentencja jest wyświetlana, jeśli żadna inna nie została dopasowana:
**Lorem ipsum...**

# Dyrektywa NgSwitch

- mikroskładnia `ngSwitch`

```
<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>
```

- zamieniona na szablon, element `<ng-template>`:

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

# Element składniowy <ng-container>

# Grupowanie elementów z <ng-container>

- Zwykle dostępny jest jakiś element gospodarza, w którym umieszczamy dyrektywę strukturalną.
- Często jest to element <li> dla dyrektywy *ngFor:

```
<li *ngFor="let hero of heroes">{{hero.name}}</li>
```

- Jeśli nie ma takiego elementu, do możemy utworzyć jakiś inny nadrzędny element, w którym umieścimy dyrektywę strukturalną, np. wewnątrz znacznikia :

```
<div *ngIf="hero" class="name">{{hero.name}}</div>
```

- Zwykle można tak uczynić wykorzystując znacznik  albo  ale nie zawsze!
- Element grupujący może zaburzyć wygląd szablonu poprzez niedopasowanie stylów CSS.
- Element <ng-container> pozwala na grupowanie elementów i nie wpływa na style CSS czy układ elementów w drzewie DOM.

# Grupowanie elementów z <ng-container>


- <ng-container> jest elementem składniowym rozpoznawanym przez Angular
- nie jest dyrektywą, komponentem, klasą czy interfejsem
- odpowiada bardziej nawiasom klamrowym ( {...} ) z języka JavaScript, obejmującym większy fragment kodu

# Grupowanie elementów z <ng-container>, kod I

- Lista zakupów, są produkty z laktozą i glutenem:

```
let listaZakupow = [
  { "nazwa":"chleb", "laktoza":false, "gluten":true,
    "url":"https://upload.wikimedia.org/wikipedia/commons/3/39/Breadindia.jpg" },
  { "nazwa":"mleko", "laktoza":true, "gluten":false,
    "url":"https://upload.wikimedia.org/wikipedia/commons/0/0e/Milk_glass.jpg" },
  { "nazwa":"pomodory", "laktoza":false, "gluten":false,
    "url":"https://upload.wikimedia.org/wikipedia/commons/6/66/Pomidory_-_tomato.jpg"},
  { "nazwa":"jabłka", "laktoza":false, "gluten":false,
    "url":"https://upload.wikimedia.org/wikipedia/commons/1/15/Red_Apple.jpg"},
  { "nazwa":"mąka", "laktoza":false, "gluten":true,
    "url":"https://www.wedrowkipokuchni.com.pl/wp-content/uploads/2016/06/ddd.jpg"},
  { "nazwa":"ser żółty", "laktoza":true, "gluten":false,
    "url":"https://upload.wikimedia.org/wikipedia/commons/8/89/Swiss_cheese_cubes.jpg"},
  { "nazwa":"kasza gryczana", "laktoza":false, "gluten":false,
    "url":"https://zlotysrodek.com.pl/wp-content/uploads/2019/04/kasza-gryczana-palona.jpg"},
  { "nazwa":"majonez", "laktoza":false, "gluten":false,
    "url":"https://upload.wikimedia.org/wikipedia/commons/8/87/Zaanse_mayonaise.jpg"}
];
```

- Chcemy na liście wyboru (znacznik <select>) wyświetlić tylko produkty bez laktozy i glutenu.
- Wybór produktu ma wyświetlać zdjęcie wybranego produktu.

# Grupowanie elementów z <ng-container>, kod II

```
<div class="row">
  <div class="column">
    <p>Lista wszystkich produktów:</p>

    <select (change)="onChange1($event.target.value)">
      <option *ngFor="let towar of lista" value="{{towar.url}}">{{towar.nazwa}}</option>
    </select> <hr>
    <img src="{{wybranyTowarURL1}}" width="200px;" alt="">
  </div>

  <div class="column">
    <p>Lista wszystkich produktów bez glutenu i laktozy:</p>

    <select (change)="onChange2($event.target.value)">
      <div *ngFor="let towar of lista">
        <option *ngIf="!towar.laktoza && !towar.gluten" value="{{towar.url}}">{{towar.nazwa}}</option>
      </div>
    </select> <hr>
    <img src="{{wybranyTowarURL2}}" width="200px;" alt="">
  </div>

  <div class="column">
    <p>Lista wszystkich produktów bez glutenu i laktozy:</p>

    <select (change)="onChange3($event.target.value)">
      <ng-container *ngFor="let towar of lista">
        <option *ngIf="!towar.laktoza && !towar.gluten" value="{{towar.url}}">{{towar.nazwa}}</option>
      </ng-container>
    </select> <hr>
    <img src="{{wybranyTowarURL3}}" width="200px;" alt="">
  </div>
</div>
```

# Grupowanie elementów z <ng-container>, kod III

```
@Component({
  selector: 'app-ng-container',
  templateUrl: './ng-container.component.html',
  styleUrls: ['./ng-container.component.css']
})
export class NgContainerComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
    this.wybranyTowarURL1 = this.lista[0].url;
    this.wybranyTowarURL2 = this.lista[1].url;
    this.wybranyTowarURL3 = this.lista[2].url;
  }

  onChange1(wybranyTowarURL) {
    this.wybranyTowarURL1 = wybranyTowarURL;
  }
  onChange2(wybranyTowarURL) {
    this.wybranyTowarURL2 = wybranyTowarURL;
  }
  onChange3(wybranyTowarURL) {
    this.wybranyTowarURL3 = wybranyTowarURL;
  }

  wybranyTowarURL1 = null;
  wybranyTowarURL2 = null;
  wybranyTowarURL3 = null;

  lista = listaZakupow;
}
```

# Tworzenie własnych dyrektyw strukturalnych

# Tworzenie własnej dyrektywy strukturalnej

Tworzenie własnej dyrektywy strukturalnej:

- importujemy dekorator dyrektywy (`@Directive`)
- importujemy `Input`, `TemplateRef` i `ViewContainerRef` - wszystkie trzy będą potrzebne do każdej dyrektywy
- stosujemy dekorator do klasy, która ma się stać naszą dyrektywą
- określamy atrybut selektora, który identyfikuje dyrektywę stosowaną do danego elementu

```
import {
  Directive, Input, TemplateRef, ViewContainerRef } from '@angular/core';

@Directive({ selector: '[appAbc]'})
export class AbcDirective {
}
```

- wstrzykujmey dwa obiekty do konstruktora dyrektywy:

```
constructor(
  private templateRef: TemplateRef<any>,
  private viewContainer: ViewContainerRef) { }
```

# Tworzenie własnej dyrektywy strukturalnej

Tworzenie własnej dyrektywy strukturalnej:

- `ViewContainerRef` - referencja na kontener, do którego możemy dodawać nasze komponenty, nasz kod HTML. Utworzony kontener możemy czyścić funkcją `clear()`.

```
this.viewContainer.clear();
```

- `ViewContainerRef` będzie umieszczony w widoku naszej aplikacji w każdym miejscu, w którym jest elemnet drzewa DOM z naszą dyrektywą.
- Metoda `createEmbeddedView()` tworzy nam instancję `EmbeddedView` zawierającą szablon elementu, w którym nasza dyrektywa została dodana.

```
const embeddedView = this.viewContainer.createEmbeddedView(this.templateRef);
```

- `TemplateRef` - referencja na szablon kodu HTML. Do konstruktora dyrektywy wstrzykujemy `TemplateRef`. Wykorzystanie naszej dyrektywy w konkretnym elemencie HTML udostępnia zawartość (szablon) tego elementu poprzez `TemplateRef`.
- Instancja klasy `ViewContainerRef` ma jeszcze wiele metod: `get()`, `createComponent()`, `insert()`, `move()`, `indexOf()`, `remove()`, `detach()`

# Przykład 1: dyrektywa appHello bez parametrów

app.component.html

```
<p *appHello >Treść znacznika testującego dyrektywę *appHello</p>
```

Treść znacznika testującego dyrektywę *appHello
Hello World

hello.directive.ts

```
import {Directive, TemplateRef, ViewContainerRef, OnInit} from '@angular/core';

@Directive({
  selector: '[appHello]'
})
export class HelloDirective {
  constructor(
    private templateRef: TemplateRef<any>,
    private viewContainer: ViewContainerRef
  ) {
    const embView = this.viewContainer.createEmbeddedView(this.templateRef);
    embView.rootNodes[0].innerHTML += "<br> Hello World";
    embView.rootNodes[0].setAttribute("style", 'color:red; font-size: 35px;');
  }
}
```

# Manipulacja na elementach DOM z poziomu TS

Manipulacja na elementach DOM:

- z poziomu języka TypeScript:
  `https://www.typescriptlang.org/docs/handbook/dom-manipulation.html`
- Obiekty DOM z poziomu JavaScript:
  `https://www.w3schools.com/js/js_htmldom.asp`
- np. elemety HTML:
  `https://www.w3schools.com/js/js_htmldom_elements.asp`

```
// 1. Select the div element using the id property
const app = document.getElementById("app");

// 2. Create a new <p></p> element programmatically
const p = document.createElement("p");

// 3. Add the text content
p.textContent = "Hello, World!";

// 4. Append the p element to the div element
app?.appendChild(p);
```

# Przykład 2: dyrektywa appHello2, jeden parametr

app.component.html

```
<p *appHello2=8 >Treść znacznika testującego dyrektywę *appHello2 </p>
```

# Nagłówek 1

## Nagłówek 2

### Nagłówek 3

#### Nagłówek 4

##### Nagłówek 5

###### Nagłówek 6

# Przykład 2: dyrektywa appHello2, jeden parametr

hello2.directive.ts

```
import { Directive, Input, TemplateRef, ViewContainerRef, OnInit } from '@angular/core';

@Directive({
  selector: '[appHello2]'
})
export class Hello2Directive {
  @Input() appHello2: number;

  constructor(
    private templateRef: TemplateRef<any>,
    private viewContainer: ViewContainerRef
  ) { }

  ngOnInit() {
    this.viewContainer.clear();
    const embeddedView = this.viewContainer.createEmbeddedView(this.templateRef);
    embeddedView.rootNodes[0].innerHTML = '';

    // let h: HTMLElement[] = [];
    let kolory: string[] = ['red', 'green', 'yellow', 'blue', 'pink', 'silver', 'teal'];
    for(let i=1; i<=this.appHello2; i++) {
      let elem = document.createElement("h"+i);
      let kolor = Math.floor( Math.random()*7 );
      elem.innerHTML = "Nagłówek " + i;
      elem.setAttribute("style", `color:${kolory[kolor]};`)
      embeddedView.rootNodes[0].appendChild(elem);
      if(i>=6) break;
    }
  }
}
```

# Przykład 3: dyrektywa appHello3, trzy parametry

app.component.html

```
<p *appHello3="'Witaj sesjo egzaminacyjna!'; color:'teal'; 'fontSize':'72px';" >
  Treść znacznika testującego dyrektywę *appHello3 z trzema parametrami
</p>
```

Witaj sesjo egzaminacyjna!

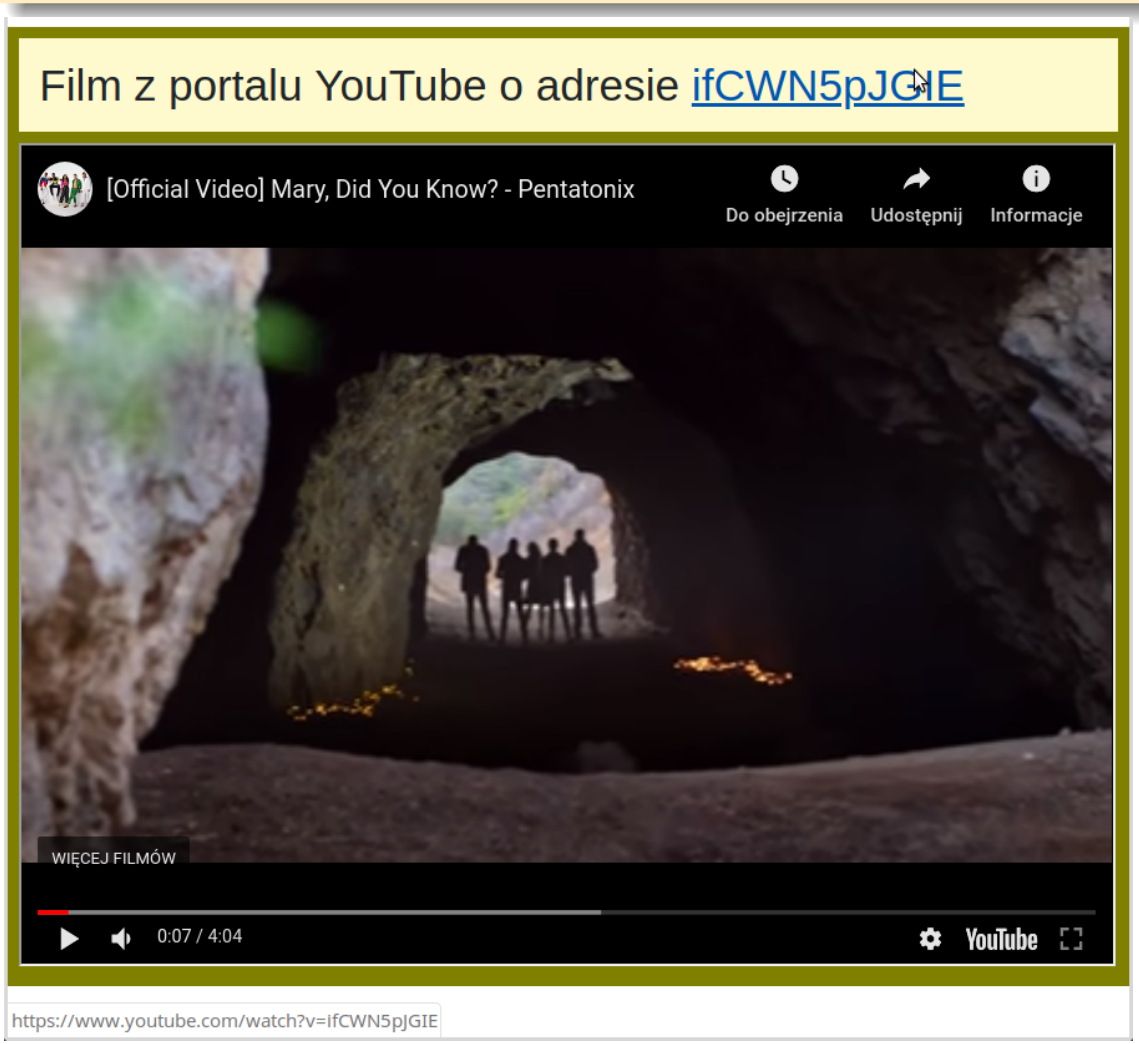
hello3.directive.ts

```
import { Directive, Input, TemplateRef, ViewContainerRef, OnInit } from '@angular/core';

@Directive({
  selector: '[appHello3]'
})
export class Hello3Directive {
  @Input() appHello3: string;
  @Input() appHello3Color: string;
  @Input() appHello3FontSize: string;

  constructor(
    private templateRef: TemplateRef<any>,
    private viewContainer: ViewContainerRef
  ) { }
  ngOnInit() {
    const embeddedView = this.viewContainer.createEmbeddedView(this.templateRef);
    embeddedView.rootNodes[0].innerHTML = this.appHello3;
    embeddedView.rootNodes[0].setAttribute(
      "style", `color:${this.appHello3Color}; font-size: ${this.appHello3FontSize};`);
  }
}
```

# Przykład 4: dyrektywa appYouTube I

app.component.html

```
<div *appYouTube="'ifCWN5pJGIE'; width:800; height:600"
  class="p-2" style="background-color: Olive;">
</div>
```



# Przykład 4: dyrektywa appYouTube II

Tworzenie własnej dyrektywy appYouTube:

- `ng g d you-tube` albo `ng generate directive you-tube`

```
CREATE src/app/you-tube.directive.spec.ts (229 bytes)
CREATE src/app/you-tube.directive.ts (143 bytes)
UPDATE src/app/app.module.ts (1441 bytes)
```

app.module.ts

```
...
import { YouTubeDirective } from './you-tube.directive';

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

# Przykład 4: dyrektywa appYouTube III

Domyślnie utworzony plik dyrektywy `you-tube.directive.ts`:

you-tube.directive.ts

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

@Directive({
  selector: '[appYouTube]'
})
export class YouTubeDirective {

  constructor() { }

}
```

# Przykład 4: dyrektywa appYouTube IV

Do dyrektyw należy:

- zaimportować odpowiednie dekoratory, klasy, uchwyty do szablonów: `Input, OnInit, TemplateRef, ViewContainerRef`
- dodać odpowiednie pola `@Input`: `appYouTube`, `appYouTubeWidth`, `appYouTubeHeight`
- wstrzyknąc odpowiednie uchwyty do szablu widoku: `templateRef`, `viewContainer`
- dodać kod towrzący znacznik `iframe` dla filmu z YouTube we wstawce programowej `ngOnInit`

# Przykład 4: dyrektywa appYouTube V

you-tube.directive.ts - po zmianach, działający przykład

```
import { Directive, Input, OnInit, TemplateRef, ViewContainerRef } from '@angular/core';

@Directive({
  selector: '[appYouTube]'
})
export class YouTubeDirective implements OnInit {
  @Input() appYouTube:string;
  @Input() appYouTubeWidth:number = 400;
  @Input() appYouTubeHeight:number = 300;

  constructor( private templateRef: TemplateRef<any>, private viewContainer: ViewContainerRef ) { }

  ngOnInit() {
    this.viewContainer.clear();
    const embeddedView = this.viewContainer.createEmbeddedView(this.templateRef);

    var h2 = document.createElement("h2");
    h2.innerHTML = `Film z portalu YouTube o adresie
        <a href="https://www.youtube.com/watch?v=${this.appYouTube}">${this.appYouTube}</a>`;
    h2.style['background-color'] = 'LemonChiffon'; h2.style['padding'] = '15px';

    var iframe = document.createElement("iframe");
    iframe.width = String(this.appYouTubeWidth);
    iframe.height = String(this.appYouTubeHeight);
    iframe.src = `https://www.youtube.com/embed/${this.appYouTube}`;

    embeddedView.rootNodes[0].appendChild(h2);
    embeddedView.rootNodes[0].appendChild(iframe);
  }
}
```

```
<div
  *appYouTube="'oOVz-ZLaT0U'; width:400; height:300">
</div>
```

# Dyrektywy atrybutowe

# Dyrektywy atrybutowe wbudowane w Angular

Dyrektywa atrybutowa zmienia wygląd albo zachowanie elementu drzewa DOM.

Najpopularniejsze dyrektywy atrybutowe wbudowane w Angular:

1. `NgClass` - ustawia klasy elementów drzewa DOM
2. `NgStyle` - ustawia style elementów drzewa DOM
3. `NgModel` - dwukierunkowe wiązanie danych

`https://angular.io/guide/attribute-directives`

# Dyrektywa NgClass

Dyrektywa `NgClass` dodaje lub usuwa klasty CSS do/z elementu HTML.
Klasy CSS są aktualizowane w elemencie w następujący sposób, zależnie od typu wyrażenia przekazanego do dyrektywy:

- `string` - klasy CSS umieszczone w typie string (podzielone spacją) zostają dodane
- `Array` - klasy CSS zadeklarowane jako elementy tablicy zostają dodane
- `Object` - kluczami w obiekcie są nazwy klas CSS, które zostają dodane do elementu, jeśli wartość spod danego klucza będzie `true`, w przeciwnym razie (wartość `false`) dana klasa CSS będzie usunięta z elementu HTML

```
<some-element [ngClass]="'first second'">...</some-element>

<some-element [ngClass]="['first', 'second']">...</some-element>

<some-element [ngClass]="{'first': true, 'second': false}">...</some-element>

<some-element [ngClass]="stringExp|arrayExp|objExp">...</some-element>

<some-element [ngClass]="{'class1 class2 class3' : true}">...</some-element>
```

# Dyrektywa NgClass - przykład

Przykład wykorzystania dyrektywy NgClass:

app.component.html

```
...
<button type="button" (click)="toggleChoice()">NgClass</button>
...
```

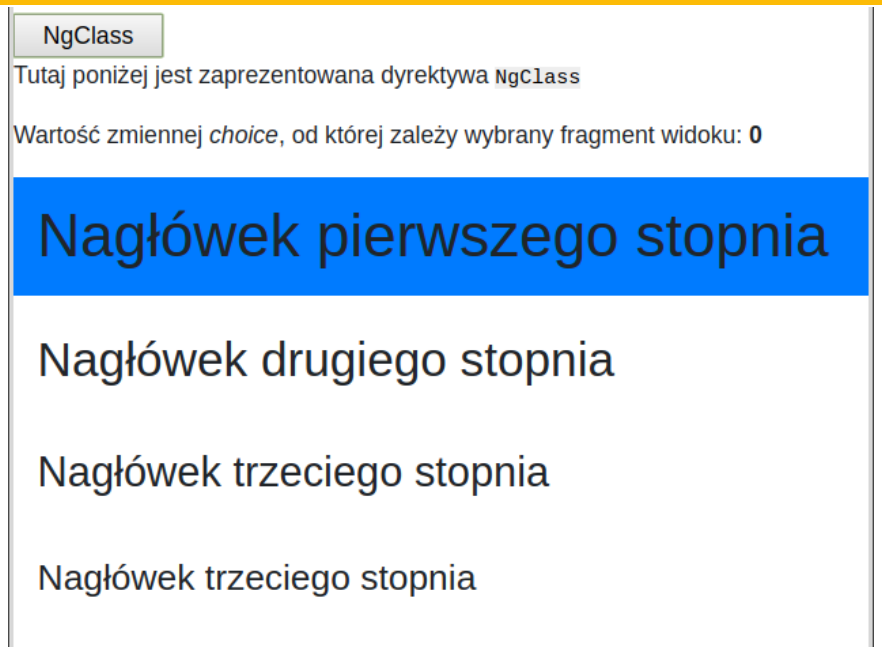
app.component.ts

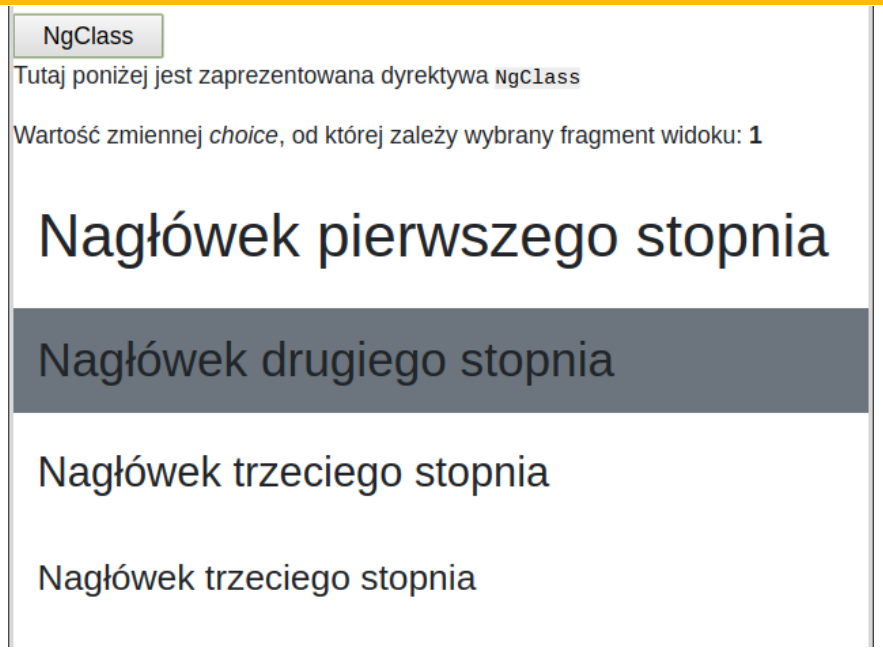
```
export class AppComponent {
  ...
  sentences: Array<string> = [
    "Ala ma kota", "Nie lubię poniedziałków", "Życie jest super!"];
  choice: number = 0;

  toggleChoice() : void {
    this.choice++;
    if(this.choice>=4) this.choice = 0;
  }
}
```

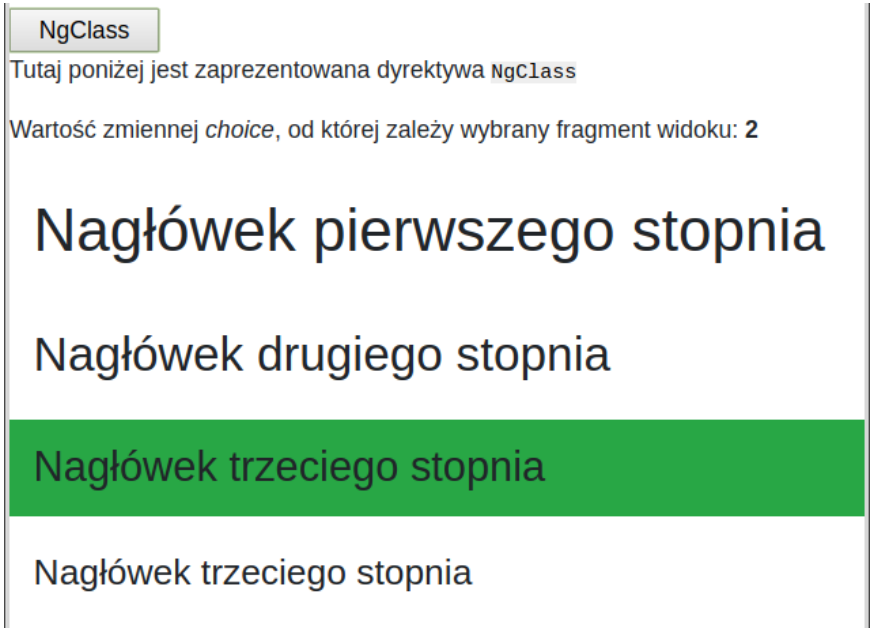
# Dyrektywa NgClass - przykład

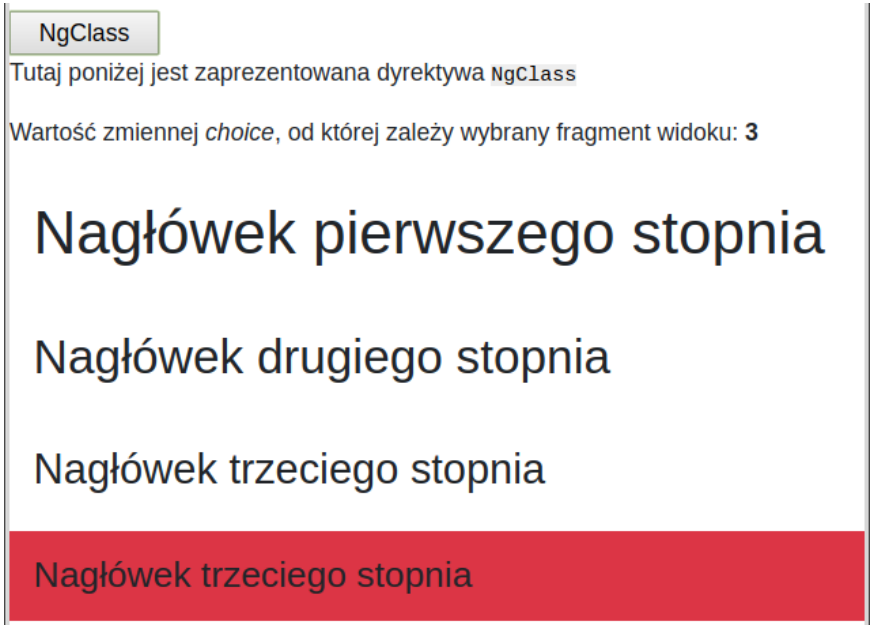
```
<div>
  <button type="button" (click)="toggleChoice()">NgClass</button>
</div>

<p>Tutaj poniżej jest zaprezentowana dyrektywa <code>NgClass</code></p>
<p>Wartość zmiennej <var>choice</var>, od której zależy wybrany fragment widoku:
  <strong>{{choice}}</strong>
</p>

<h1 [ngClass]="'p-3 mb-2 ' + (choice==0?'bg-primary':'bg-white')">
  Nagłówek pierwszego stopnia
</h1>
<h2 [ngClass]="'p-3 mb-2 ' + (choice==1?'bg-secondary':'bg-white')">
  Nagłówek drugiego stopnia
</h2>
<h3 [ngClass]="'p-3 mb-2 ' + (choice==2?'bg-success':'bg-white')">
  Nagłówek trzeciego stopnia
</h3>
<h4 [ngClass]="'p-3 mb-2 ' + (choice==3?'bg-danger':'bg-white')">
  Nagłówek trzeciego stopnia
</h4>
```

# Dyrektywa NgClass - przykład



# Dyrektywa NgStyle

Dyrektywa `NgStyle`:

- aktualizuje wartość stylów elementu HTML, który ją zawiera
- uswawia jedną lub więcej właściwości stylu określonego jako pary klucz-wartość oddzielone dwukropkiem
- klucz jest nazwą stylu, z opcjonalnym dopiskiem `.<jednostka>` (np. '`top.px`', '`font.em`')
- wartość należy wyznaczyć z wyrażenia, które jest przypisane do danego klucza
- wynik wyrażenia, inny niż `null` oznacza wartość donej właściwości stylu w określonych w kluczu jednostkach miary
- wynik wyrażenia równy `null` oznacza usunięcie danego stylu z elementu HTML

# Dyrektywa NgStyle

Dyrektywa `NgStyle`

Ustawienie czcionki pewnego elemetu (`some-element`) na wartość wyniku wyrażenia `styleExp`:

```
<some-element [ngStyle]="{'font-style': styleExp}">...</some-element>
```

Ustawienie szerokości pewnego elementu (jednostka to piksele) na wartość zwróconą przez wyrażenie `styleExp`:

```
<some-element [ngStyle]="{'max-width.px': widthExp}">...</some-element>
```

Ustawienie kolekcji stylów pewnego elementu używając wyrażenia (`objExp`), które zwraca pary klucz-wartość:

```
<some-element [ngStyle]="objExp">...</some-element>
```

# Dyrektywa NgStyle - przykład

Przykład wykorzystania dyrektywy NgStyle:

app.component.css

```
h2 {
  color : white;
  padding: 12pt;
}
```

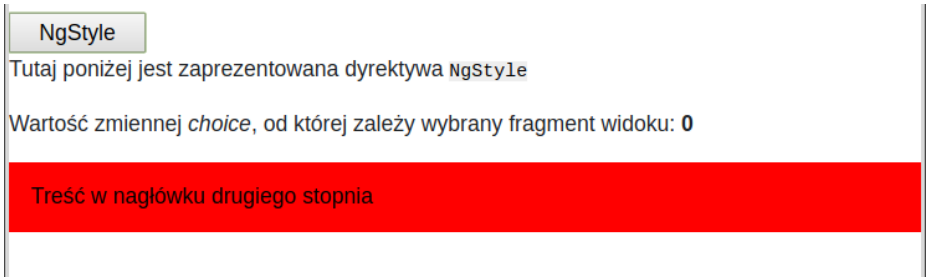
app.component.ts

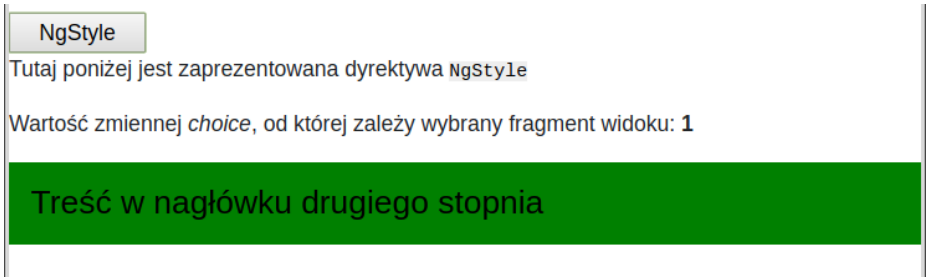
```
export class AppComponent {
  ...

  colors: Array<string> = ["red", "green", "blue", "purple"];
  choice: number = 0;

  toggleChoice() : void {
    this.choice++;
    if(this.choice>=4) this.choice = 0;
  }
}
```

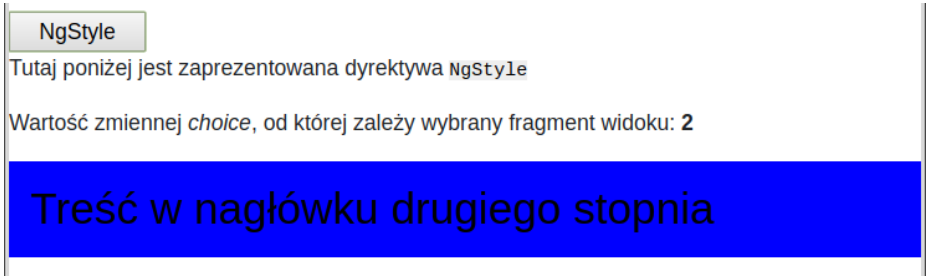
# Dyrektywa NgStyle - przykład

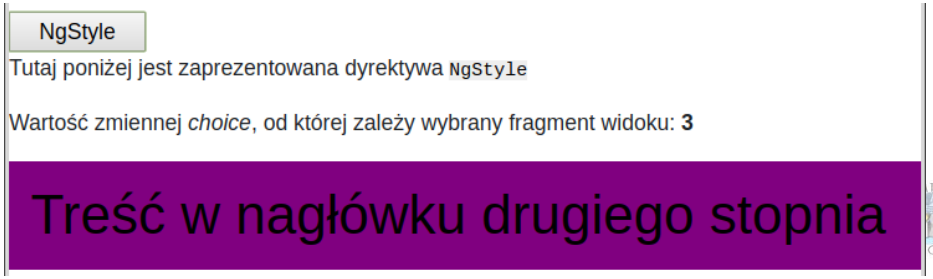
app.component.html

```
...
<button type="button" (click)="toggleChoice()">NgStyle</button>
...
<h2 [ngStyle]=
  "{'font-size.%': choice*50+100,
    'color': 'black',
    'background-color': colors[choice]
  }"
> Treść w nagłówku drugiego stopnia</h2>
<!-- [style.color]="null" -->
```



# Źródła

- https://angular.io/
- https://www.typescriptlang.org/
- http://www.angular.love/2017/10/03/angular-dyrektywy-strukturalne/s