



# Angular

Vasyl Hladush, Bruno Banaszczyk

---



# Plan prezentacji

## 01 Wprowadzenie

Dlaczego Angular?

## 02 Architektura

Opis architektury i  
kluczowych konceptów  
aplikacji w Angular.

## 03 Narzędzia

Najważniejsze narzędzia  
do pracy z Angular.

## 04 Przegląd kodu

Prezentacja  
przykładowej aplikacji.





01

# Wprowadzenie

---

Dlaczego Angular?

# Historia i rozwój

**2009**

## **Powstanie AngularJS**

Misko Hevery i Adam Abrons tworzą pierwszą wersję Angulara dla Google

**2010**

## **Oficjalne wydanie**

Angular JS trafia do front-end developerów.

**2016**

## **Powstanie Angular 2**

Powstaje nowa wersja z wieloma przełomowymi zmianami.



# Angular **vs** AngularJS



## **TS** TypeScript

Od 2016 roku Angular bazuje na TypeScript zamiast JavaScript.



## Wydajność

Angular osiąga do 5 razy większą wydajność niż jego poprzednik.





## Komponenty

Najbardziej podstawowe "kawałki" kodu, które definiują zachowanie elementów HTML.



## Narzędzia

Nowy Angular posiada własny CLI, co przyspiesza tworzenie aplikacji.



# Popularność

# 19,89 %

W ankiecie przeprowadzonej przez StackOverflow w 2023 roku 19,89% web-developerów odpowiedziało, że dotychczas korzystali głównie z Angulara i planują z niego korzystać w następnym roku.

Link: <https://survey.stackoverflow.co/2023/#most-popular-technologies-webframe-prof>

# Dlaczego **Angular**?



## Komponenty

Minimalistyczne i przejrzyste tworzenie aplikacji.



## Czysty kod

TypeScript zapewnia większą czytelność tworzonego kodu.



## Multiplatform

Angular umożliwia łatwe tworzenie aplikacji na iOS i Android.



## CSR

Client Side Rendering sprawia, że strona szybciej reaguje na interakcje.



## Angular CLI

Narzędzie pozwalające na szybkie i proste tworzenie aplikacji.



## Ekosystem

Mnogość dostępnych modułów, dodatków i narzędzi.



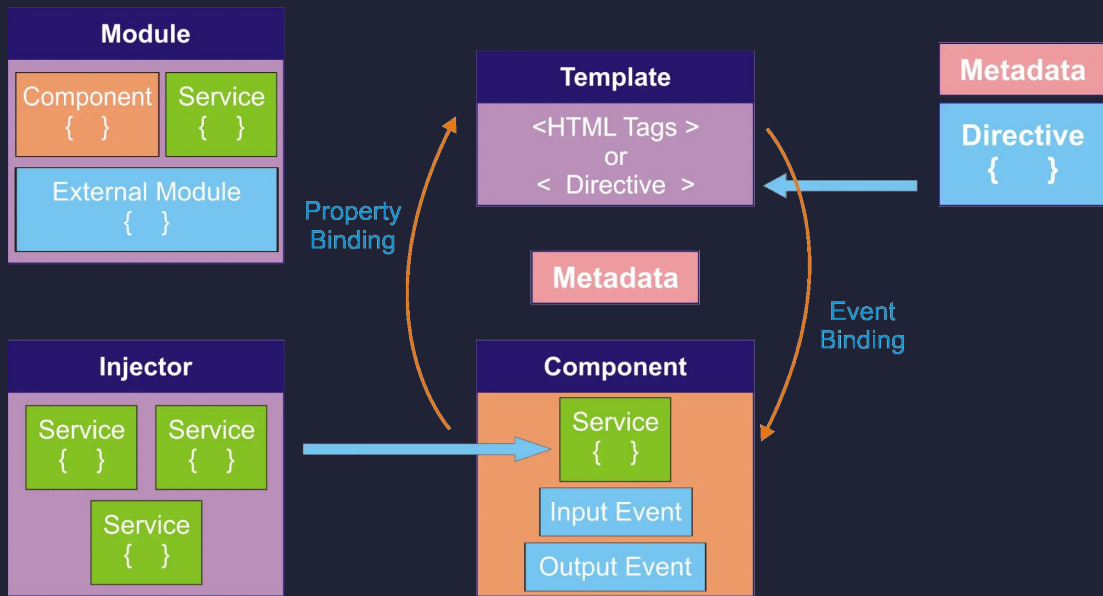
02

# Architektura

---

Opis architektury i kluczowych konceptów aplikacji w Angular.





# Komponenty

## HTML Template

### HTML Template

Jest to zwykły kod HTML, który dodatkowo posiada syntax Angulara, aby komunikować się z komponentem.

## Component Class

### Component Class

Jest to klasa TypeScript, w której są atrybuty przechowujące dane komponentu oraz metody opisujące jego logikę.

## Component Metadata

### Component Metadata

Dodatkowe dane opisujące komponent, które umożliwiają API Angulara jego obsługę.

greet.component.ts

Import → `import { Component, OnInit } from '@angular/core';`

Metadata { `@Component({`  
    `selector: 'app-greet',` ← Component Tag  
    `templateUrl: './greet.component.html',` ← HTML Template File Name and Location  
    `styleUrls: ['./greet.component.css']` ← CSS File Name and Location  
    `})`

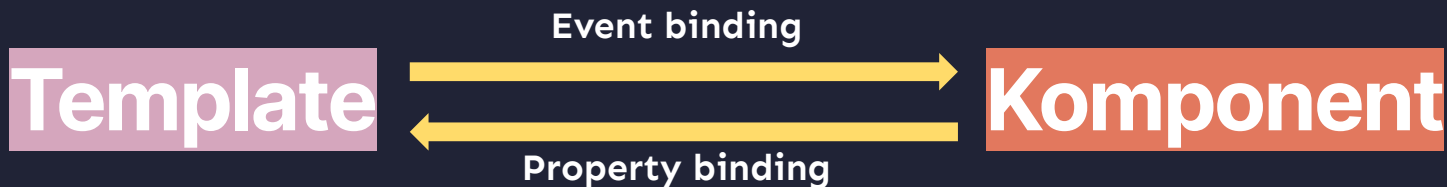
© TutorialsTeacher.com

Component Class { `export class GreetComponent implements OnInit {`  
    `constructor() { }`  
    `ngOnInit(): void {`  
        `}`  
    `}`

Link: <https://www.tutorialsteacher.com/angular/angular-component>

## Event binding

Umożliwia komponentowi reakcję na zdarzenia użytkownika, takie jak kliknięcie, przesunięcie myszy, wciśnięcie klawisza.



## Property binding

Umożliwia przekazanie wartości z komponentu do właściwości elementu HTML.

# Dyrektywy

**Dyrektywy** w Angularze są to specjalne markery na elementach DOM, które mówią frameworkowi Angular, jak ma się zachować lub jak przekształcić dany element i jego dzieci.



## Dyrektywy komponentów

Dyrektywy komponentów łączą logikę bezpośrednio z HTML szablonami, które definiują, jak komponent ma być renderowany.



## Dyrektywy strukturalne

Dyrektywy te zmieniają układ DOM poprzez dodawanie, usuwanie lub zastępowanie elementów.

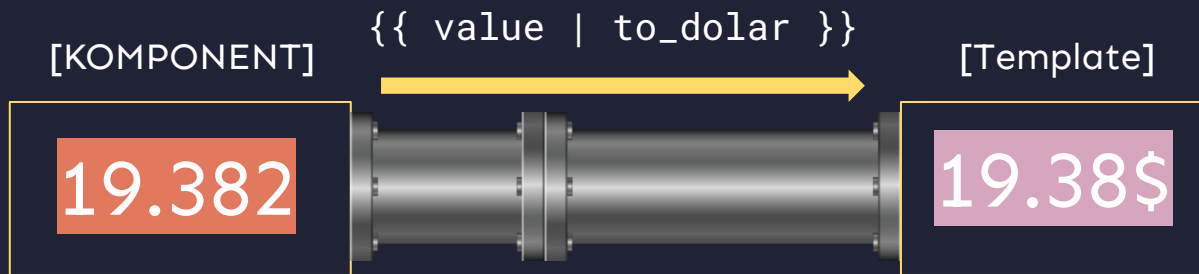


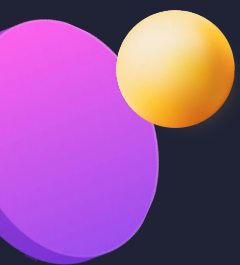
## Dyrektywy atrybutowe

Zmieniają wygląd lub zachowanie istniejących elementów.

# Pipes



**Pipe** to specjalna funkcja, która służy do transformacji danych w szablonach HTML przed ich wyświetleniem. Pipes w Angularze są używane do formatowania danych wyjściowych bez zmiany oryginalnego źródła danych.

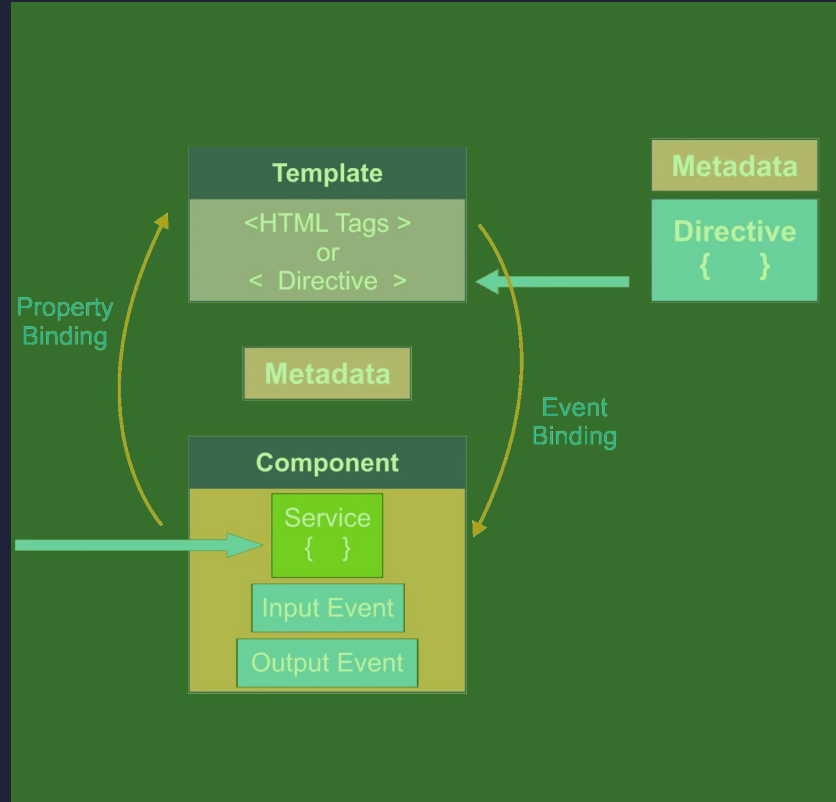
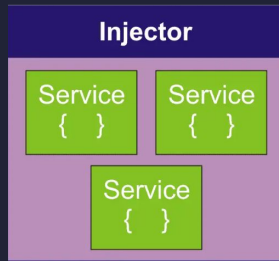
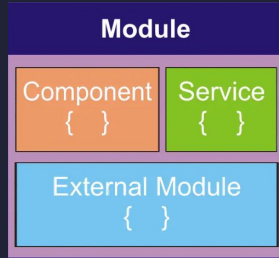




```
import { Pipe, PipeTransform } from '@angular/core';

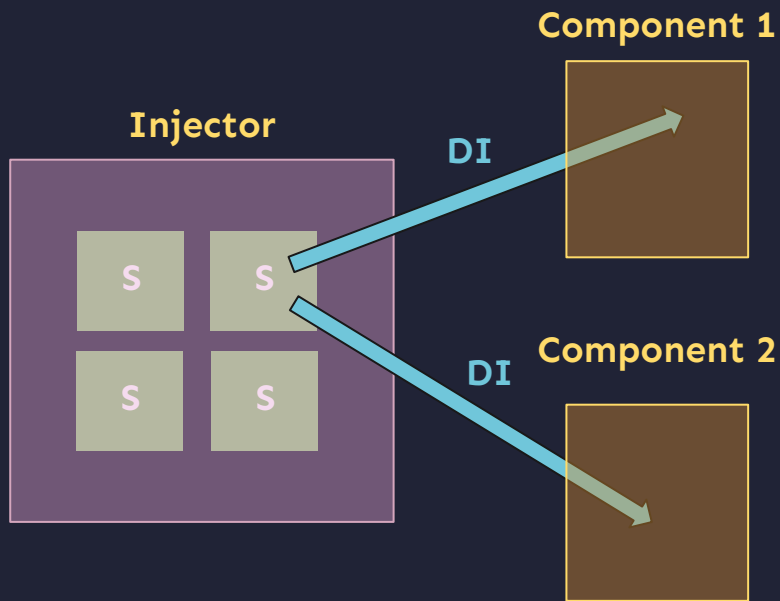
@Pipe({name: 'reverse'})
export class ReversePipe implements PipeTransform {
  transform(value: string): string {
    return value.split('').reverse().join('');
  }
}
```







# Service



## Service

Jest to klasa do tworzenia danych i funkcjonalności niezwiązanych z widokiem i interfejsem użytkownika. Jest ona niezależna od komponentów, a przez to podatna do wielokrotnego wykorzystania.

## Injector

Injector automatycznie tworzy instancje serwisów i dostarcza je do komponentów (lub innych serwisów) w momencie ich potrzeby.

## Dependency Injection

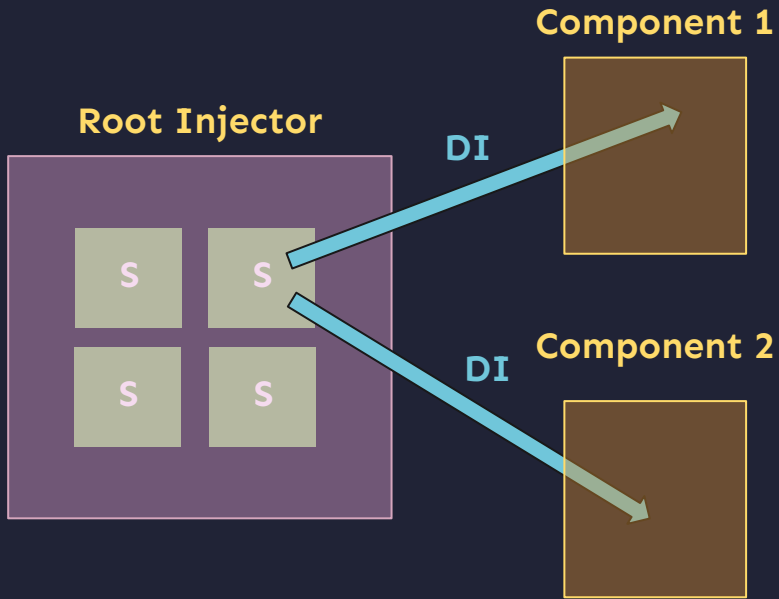
Ogólnie jest to sposób tworzenia klas, w którym wszystkie zależności danej klasy dostarczane są ze źródeł zewnętrznych, a nie tworzone w tej klasie.

# Dependency Injection

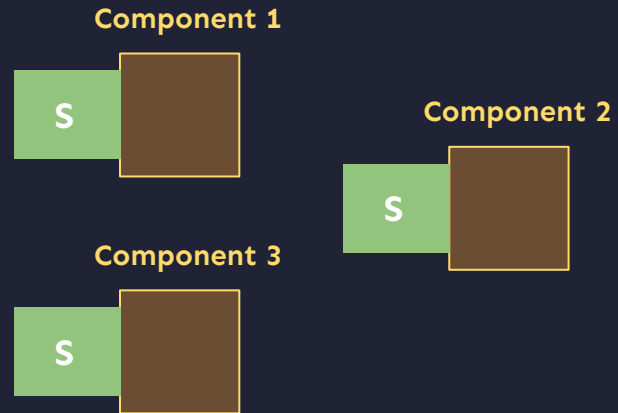
1. Definicja serwisów →  @Injectable
2. Rejestracja serwisów → {providedIn: 'root'}
3. Wstrzykiwanie serwisów →   

```
@Injectable({providedIn: 'root'})  
export class HeroService {
```

# Root vs Component Provider

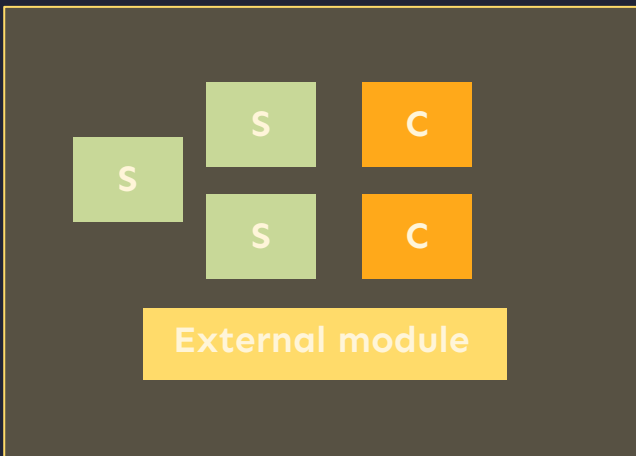


```
@Component({
  standalone: true,
  selector: 'app-hero-list',
  templateUrl: './hero-list.component.html',
  imports: [ NgFor, NgIf, HeroDetailComponent ],
  providers: [ HeroService ]
})
```



# Modules

## Module



## Module

Moduł w Angularze, znany również jako **NgModule**, to kluczowy mechanizm organizacyjny, który pozwala grupować powiązane ze sobą komponenty, serwisy, pipe'y oraz inne moduły w koherentne bloki.

Użycie modułów w Angularze pozwala na łatwe zarządzanie złożonymi aplikacjami, organizację kodu oraz jego ponowne użycie.



03

# Narzędzia

---

Najważniejsze narzędzia do pracy z Angular.

# Angular CLI

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



```
ng new my-first-project  
cd my-first-project  
ng serve
```



```
ng generate <schematic>
```

# Angular DevTools

The screenshot displays the Angular DevTools interface for a web application titled 'todos'. At the top, a form titled 'What needs to be done?' contains two input fields: 'Learn Angular' and 'Build an Angular application', both with 'x' icons to clear them. Below the form, it indicates '2 item left' and a 'Clear completed' button. The main interface is divided into two panels. The left panel, titled 'Components', shows a tree view of the component hierarchy. The right panel, titled 'app-todos', shows the component's metadata, including its outputs and properties. The 'app-todos' component is selected in the tree view.

**Components Panel (Left):**

- app-root
  - router-outlet[RouterOutlet]
  - ng-component
    - router-outlet[RouterOutlet]
    - app-todo-demo
      - a[RouterLinkWithHref]
      - a[RouterLinkWithHref]
      - router-outlet[RouterOutlet]
      - app-todos == \$ng0
        - a[RouterLinkWithHref]
        - a[RouterLinkWithHref]
        - a[RouterLinkWithHref]
        - app-todo[TooltipDirective]
          - div[TooltipDirective]
        - app-todo[TooltipDirective]
          - div[TooltipDirective]

**app-todos Panel (Right):**

Angular version: 12.0.0-next.8+395.sha-eac42e9 | DevTools SHA: d67c0ba

Search components

app-todos

@Outputs

- > add : {...}
- > delete : {...}
- > update : {...}

Properties

- > cdRef : {...}
- hashListener: (...)
- > todos :
  - > 0 :
    - completed: false
    - id: 42
    - label: Learn Angular
  - > 1 :
    - completed: false
    - id: 43

1. Components

2. Profiler

# Angular Language Service

```
{{ cu| }}
```



customer

property



[Angular] Identifier 'orders' is not defined. The component declaration, template variable declarations, and element references do not contain such a member

```
{{ orders }}
```