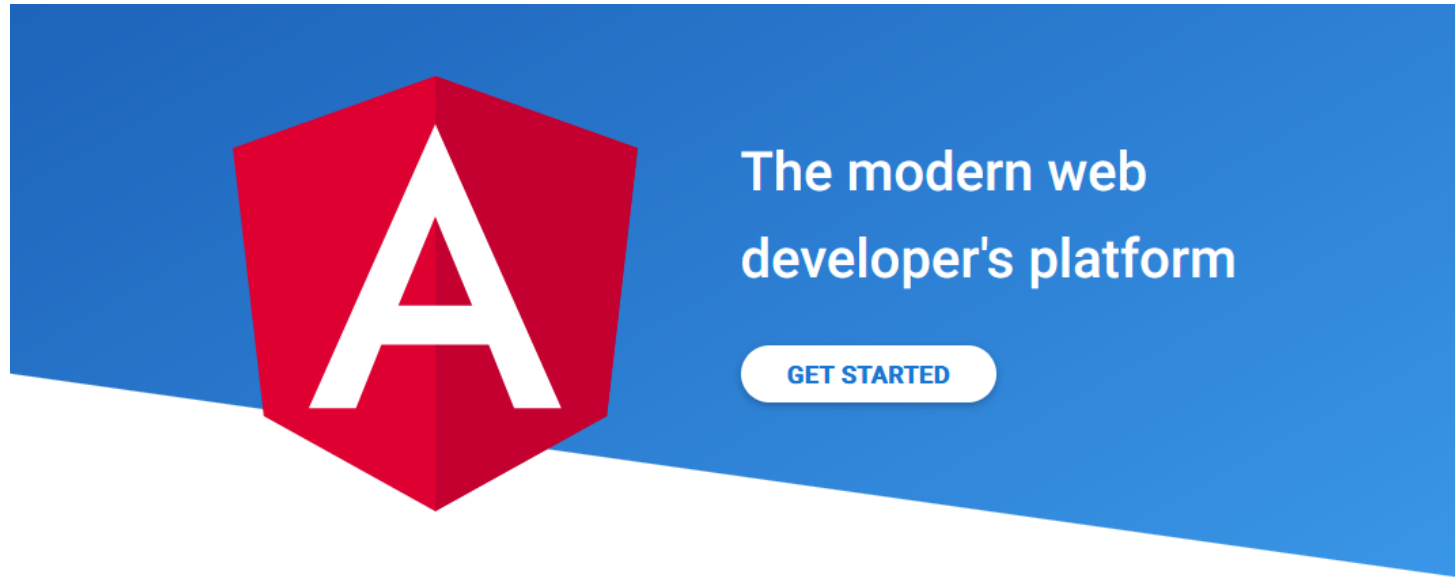


Angular
2.3

Angular

wspierana przez Google otwarta biblioteka JS napisana w języku TypeScript

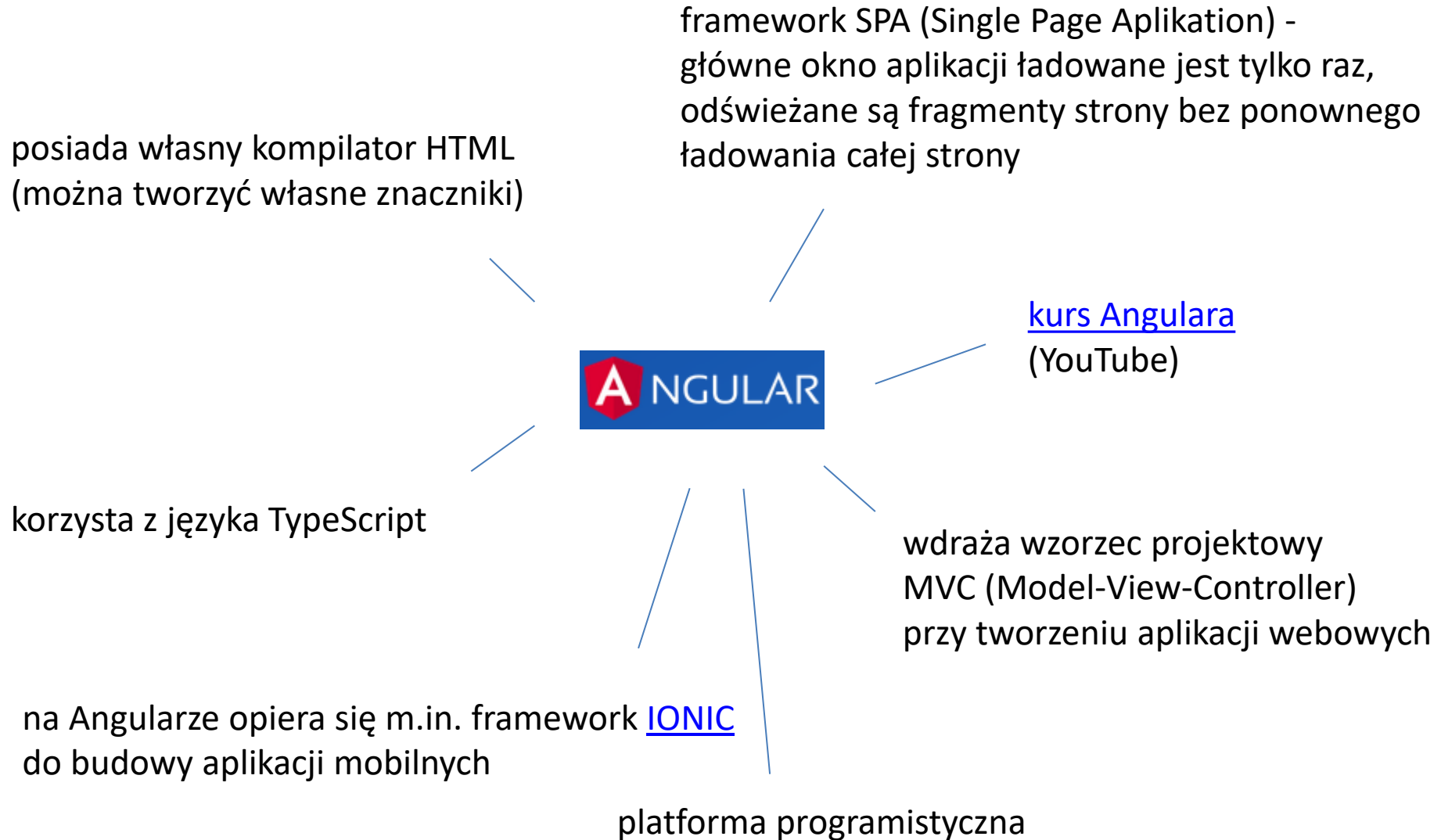


DEVELOP ACROSS ALL PLATFORMS

Learn one way to build applications with Angular and reuse your code and abilities to build apps for any deployment target. For web, mobile web, native mobile and native desktop.

<https://angular.io/>

Angular



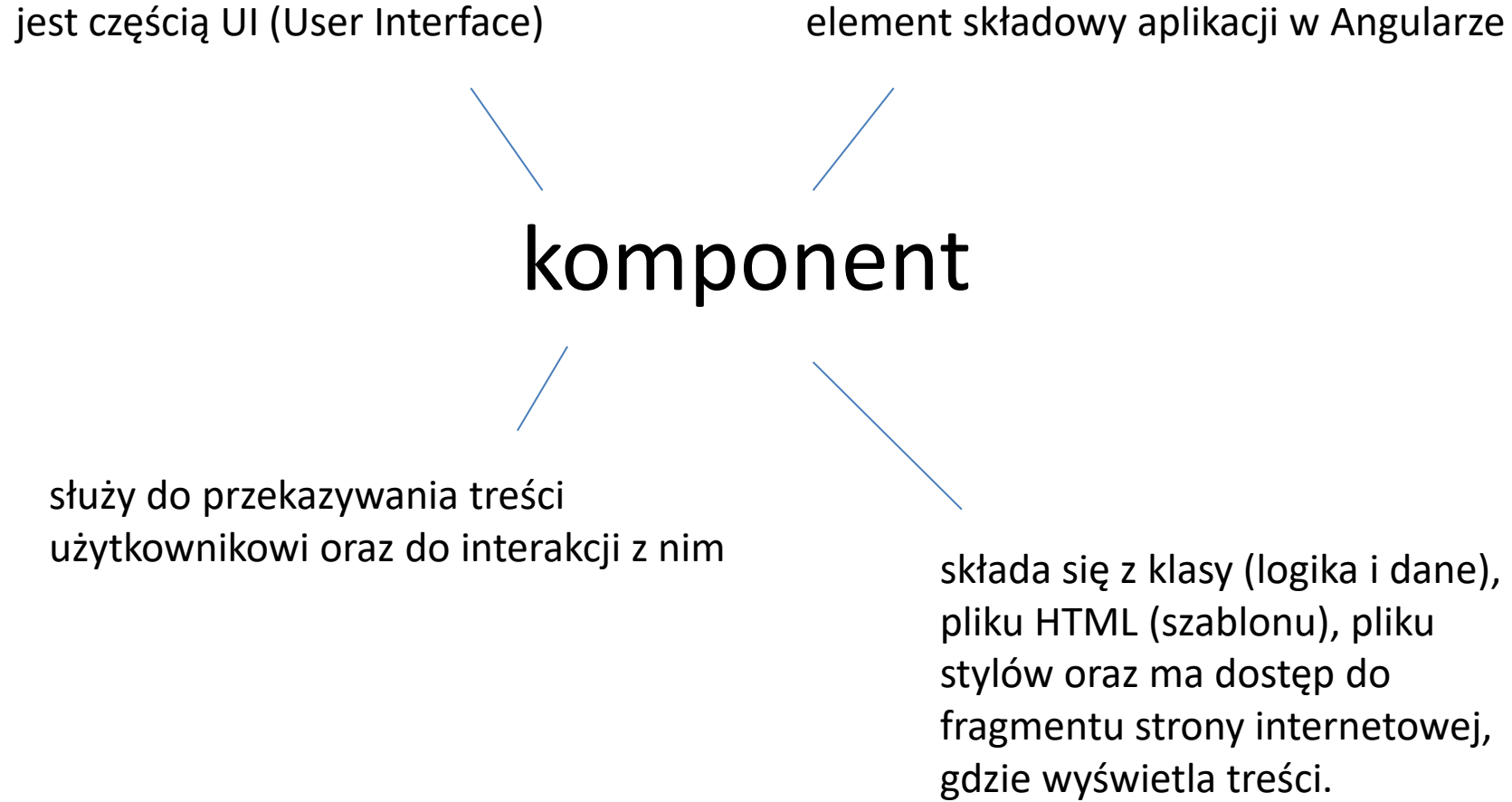
[Getting started with Angular](#)

Component

jest częścią UI (User Interface)

element składowy aplikacji w Angularze

komponent

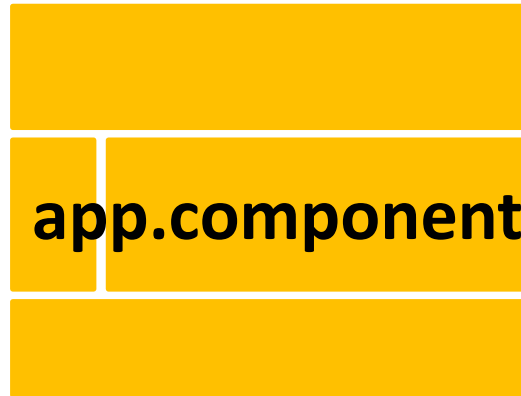


służy do przekazywania treści
użytkownikowi oraz do interakcji z nim

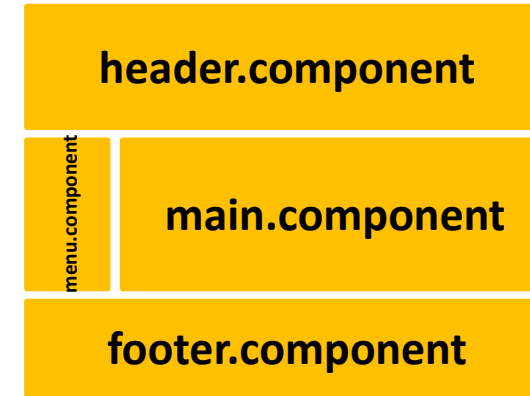
składa się z klasy (logika i dane),
pliku HTML (szablonu), pliku
stylów oraz ma dostęp do
fragmentu strony internetowej,
gdzie wyświetla treści.

Component

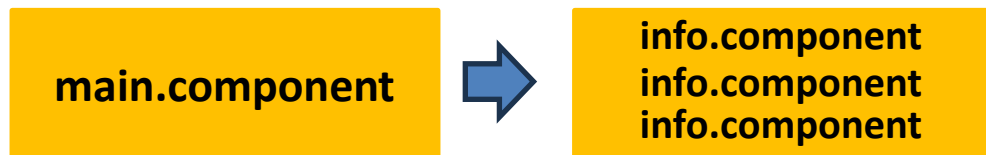
jeden komponent



kilka komponentów



kilka komponentów

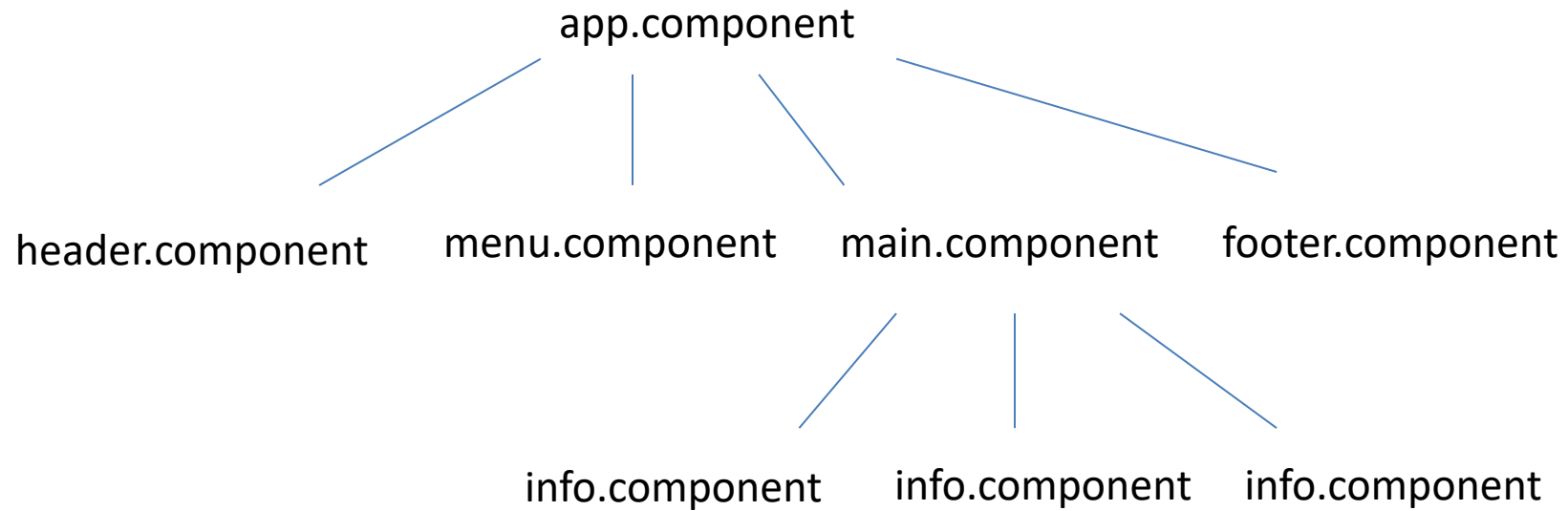


aplikacja składa się z komponentów, można różnie ją zaprojektować

[Understanding Modules with Angular](#)

Component

aplikacja posiada co najmniej jeden
komponent główny app.component



komponenty aplikacji tworzą drzewo komponentów

Komponenty samodzielne (standalone)

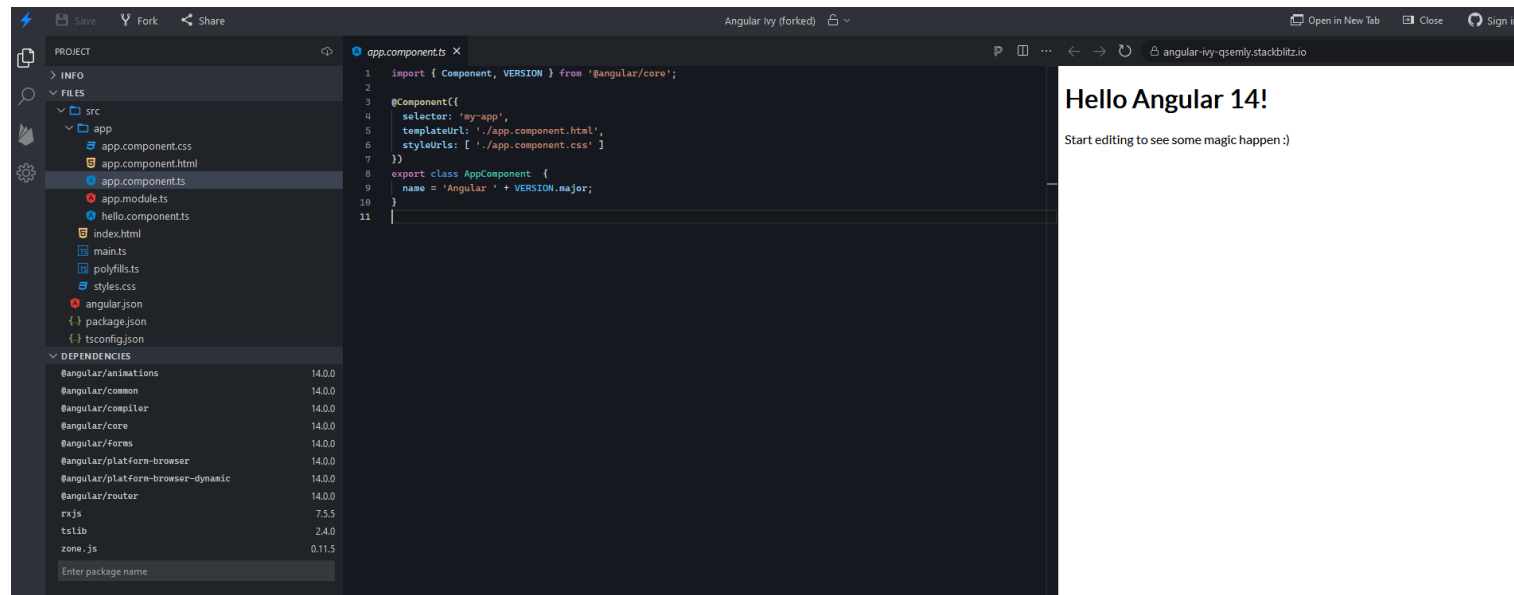
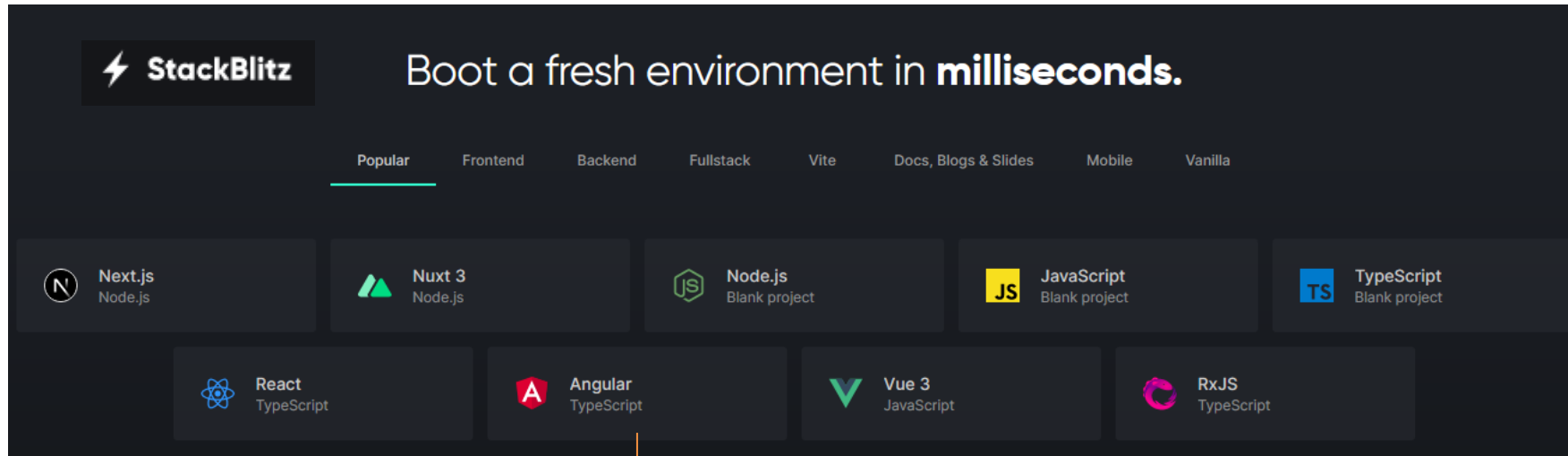
Komponenty, dyrektywy i potoki można oznaczać jako ***samodzielne (standalone: true)***. Klasy Angular oznaczone jako samodzielne nie muszą być deklarowane w NgModule (kompilator Angular zgłosi błąd, jeśli je spróbujemy zadeklarować w ten sposób).

Samodzielne komponenty określają swoje zależności bezpośrednio, zamiast pobierać je za pośrednictwem NgModules.

Na przykład, jeśli X jest samodzielnym komponentem, może bezpośrednio zaimportować inny samodzielny komponent Y

<https://angular.io/guide/standalone-components>

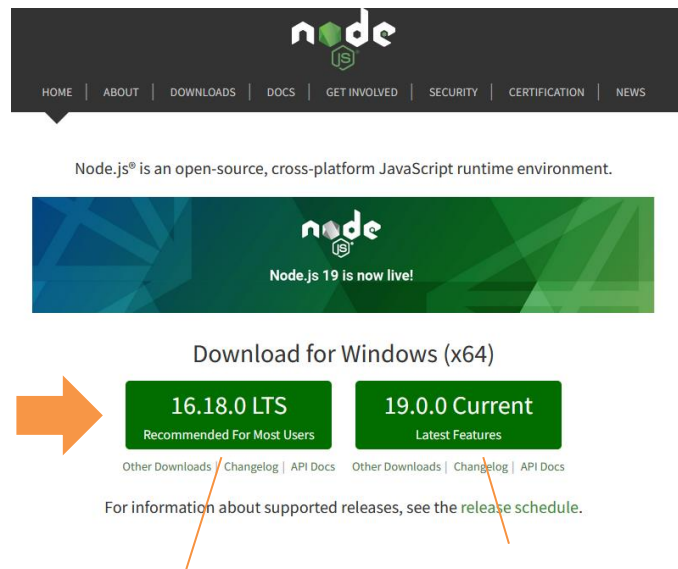
StackBlitz – Angular on-line



Instalacja Node.js

Node.js oraz menadżer pakietów npm

<https://nodejs.org/en/>



Long Term Support
długi okres wsparcia technicznego

wersja bieżąca

sprawdzenie zainstalowanej wersji

```
C:\>node -v  
v16.18.0  
  
C:\>npm -v  
8.19.2
```

Instalacja Angular CLI

Angular CLI to narzędzie interfejsu wiersza poleceń do inicjalizowania, rozwijania i utrzymywania aplikacji.

CMD

```
C:\>npm install -g @angular/cli
```

Angular instalowany globalnie (dodanie wpisu do PATH)

Instalacja Angular CLI

Introduction

Getting started

- What is Angular?
- Try it
- Setup

Understanding Angular

Developer guides

Best practices

Angular tools

Tutorials

Updates and releases

Reference

Documentation contributors guide

Install the Angular CLI

You can use the Angular CLI to create projects, generate application and library code, and perform a variety of ongoing development tasks such as testing, bundling, and deployment.

To install the Angular CLI, open a terminal window and run the following command:

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

On Windows client computers, the execution of PowerShell scripts is disabled by default. To allow the execution of PowerShell scripts, which is needed for npm global binaries, you must set the following [execution policy](#):

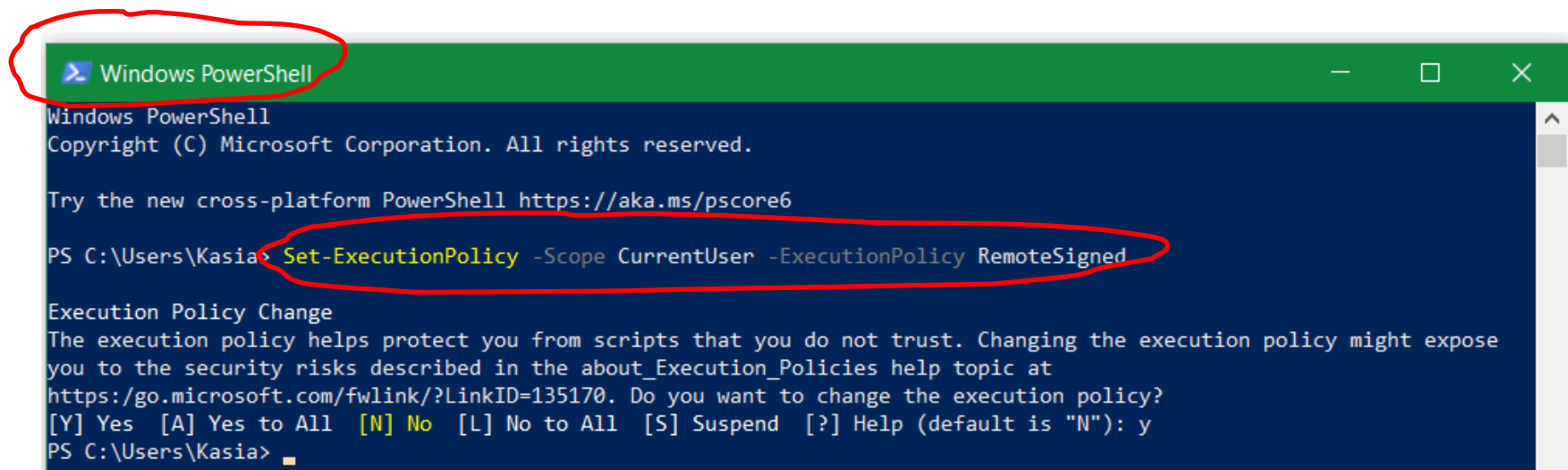
```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

Carefully read the message displayed after executing the command and follow the instructions. Make sure you understand the implications of setting an execution policy.

<https://angular.io/guide/setup-local>

w systemie Windows należy włączyć możliwość wykonywania skryptów w PowerShellu

Instalacja Angular CLI



The screenshot shows a Windows PowerShell window with a green title bar. The terminal text is as follows:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Users\Kasia> Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

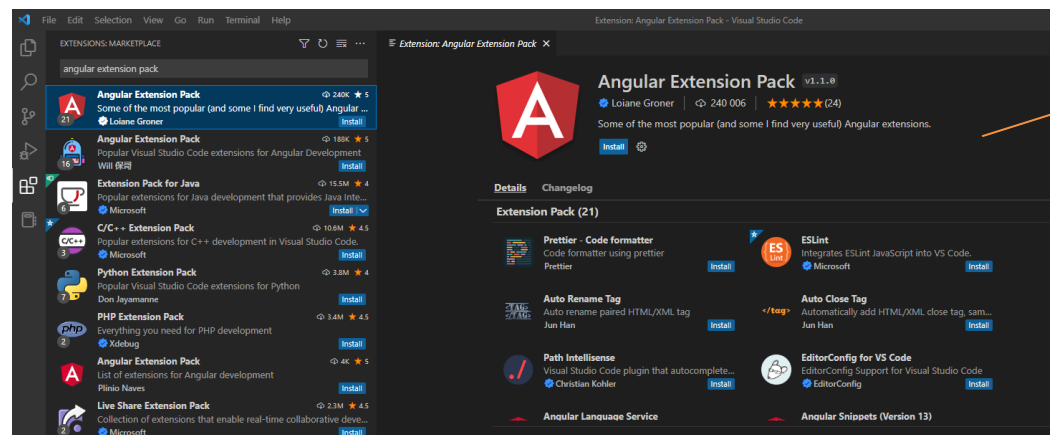
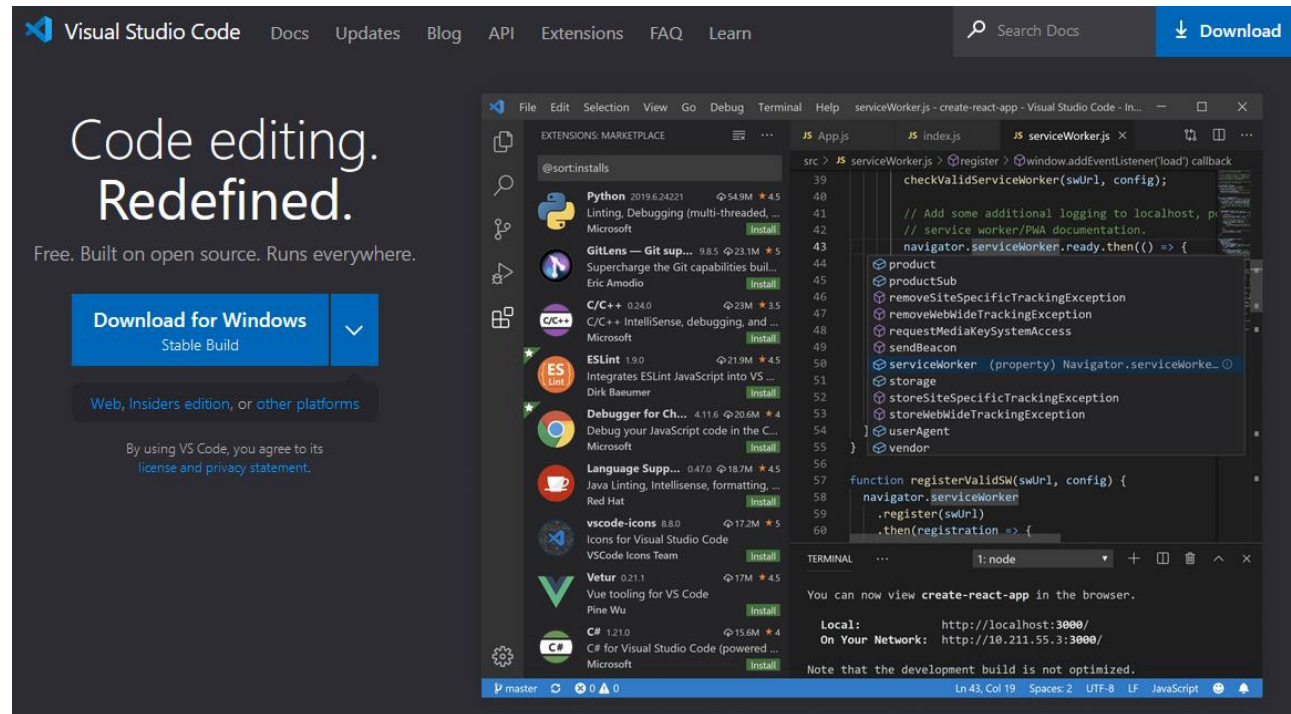
The command `Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned` is highlighted with a red oval. Below the command, the following text is displayed:

```
Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): y
PS C:\Users\Kasia>
```

komendę wykonujemy w PowerShellu, a nie w cmd

Instalacja Visual Studio Code

<https://code.visualstudio.com/>



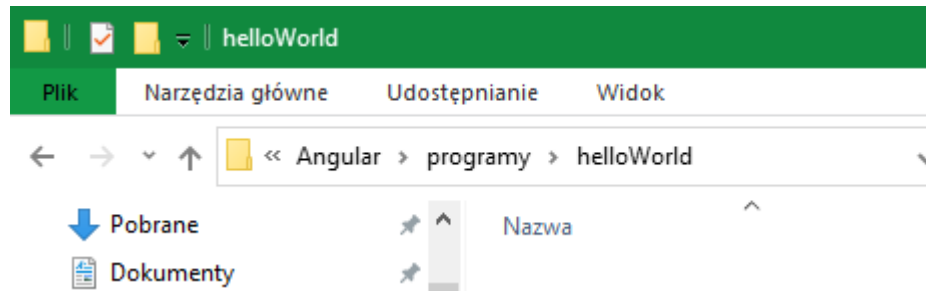
dodatek Angular Extension Pack

Tworzenie nowego projektu

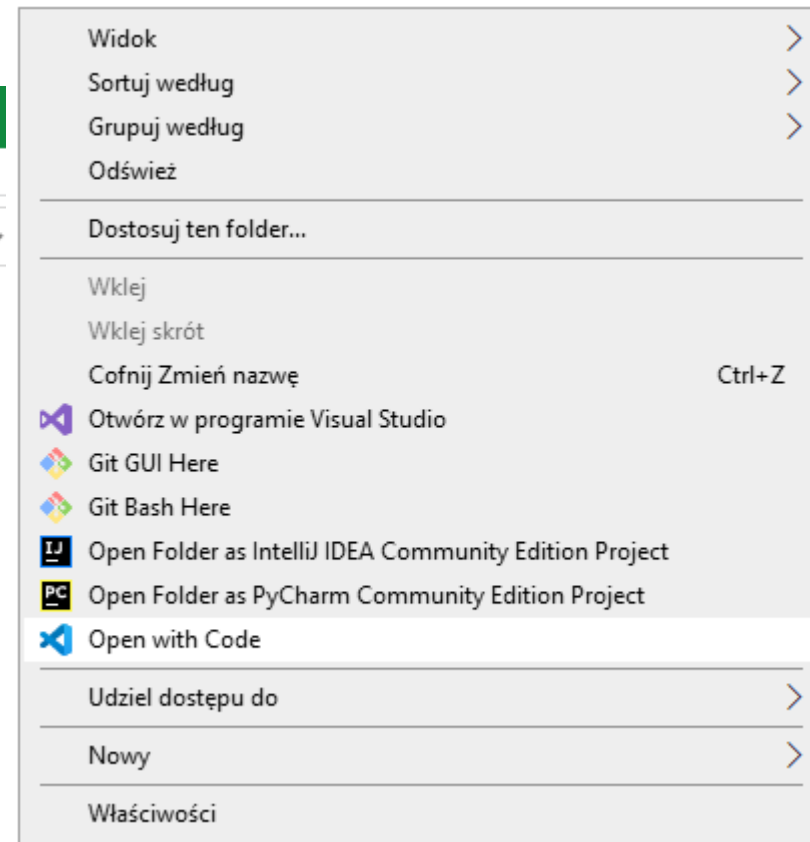
tworzymy folder helloWorld (nie jest to konieczne, nowy projekt utworzy nowy folder)

helloWorld

otwieramy folder helloWorld

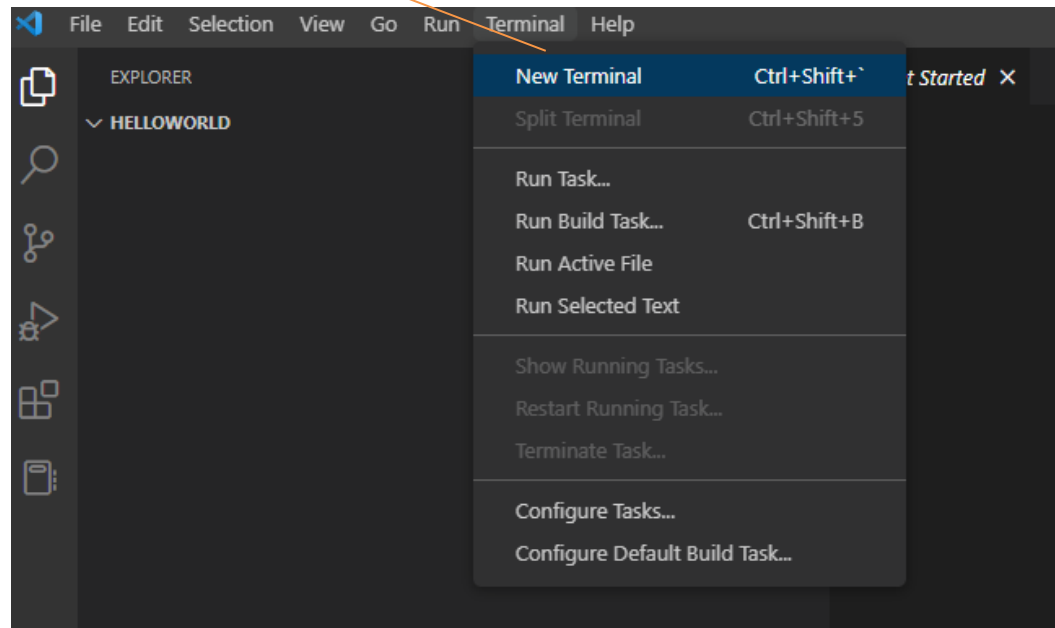


wewnątrz folderu helloWorld
prawy przycisk myszy/Open with Code



Tworzenie nowego projektu

w Visual Studio Code w katalogu projektu otwieramy nowy terminal



Tworzenie nowego projektu - komendy

w terminalu wpisujemy polecenie utworzenia aplikacji o nazwie **myHelloWorld** (z komponentami typu standalone - samodzielne)

```
ng new myHelloWorld
```

wybieramy CSS

```
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
  SCSS [ https://sass-lang.com/documentation/syntax#scss ]
  Sass [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less [ http://lesscss.org ]
```

wybieramy y

```
? Which stylesheet format would you like to use? CSS
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? (y/N)
```

Server-Side Rendering – renderowanie po stronie serwera

Static Site Generation – generowanie stycznych stron w trakcie budowania projektu

obie technologie przyspieszają działanie strony

[ng new](#)

Tworzenie nowego projektu

```
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. No
Global setting: disabled
Local setting: No local workspace configuration file.
Effective status: disabled
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
CREATE myHelloWorld/angular.json (2954 bytes)
CREATE myHelloWorld/package.json (1045 bytes)
CREATE myHelloWorld/README.md (1066 bytes)
CREATE myHelloWorld/tsconfig.json (863 bytes)
CREATE myHelloWorld/.editorconfig (274 bytes)
CREATE myHelloWorld/.gitignore (548 bytes)
CREATE myHelloWorld/.browserslistrc (600 bytes)
CREATE myHelloWorld/karma.conf.js (1431 bytes)
CREATE myHelloWorld/tsconfig.app.json (287 bytes)
CREATE myHelloWorld/tsconfig.spec.json (333 bytes)
CREATE myHelloWorld/.vscode/extensions.json (130 bytes)
CREATE myHelloWorld/.vscode/launch.json (474 bytes)
CREATE myHelloWorld/.vscode/tasks.json (938 bytes)
CREATE myHelloWorld/src/favicon.ico (948 bytes)
CREATE myHelloWorld/src/index.html (298 bytes)
CREATE myHelloWorld/src/main.ts (372 bytes)
CREATE myHelloWorld/src/polyfills.ts (2338 bytes)
CREATE myHelloWorld/src/styles.css (80 bytes)
CREATE myHelloWorld/src/test.ts (749 bytes)
CREATE myHelloWorld/src/assets/.gitkeep (0 bytes)
* Installing packages (npm)...
✓ Packages installed successfully.
```

instalowanie pakietów

Tworzenie nowego projektu

po pomyślnym zainstalowaniu pakietów przechodzimy do nowo utworzonego katalogu projektu **myHelloWorld** oraz uruchamiamy serwer:

`cd .\myHelloWorld\`
`ng serve`

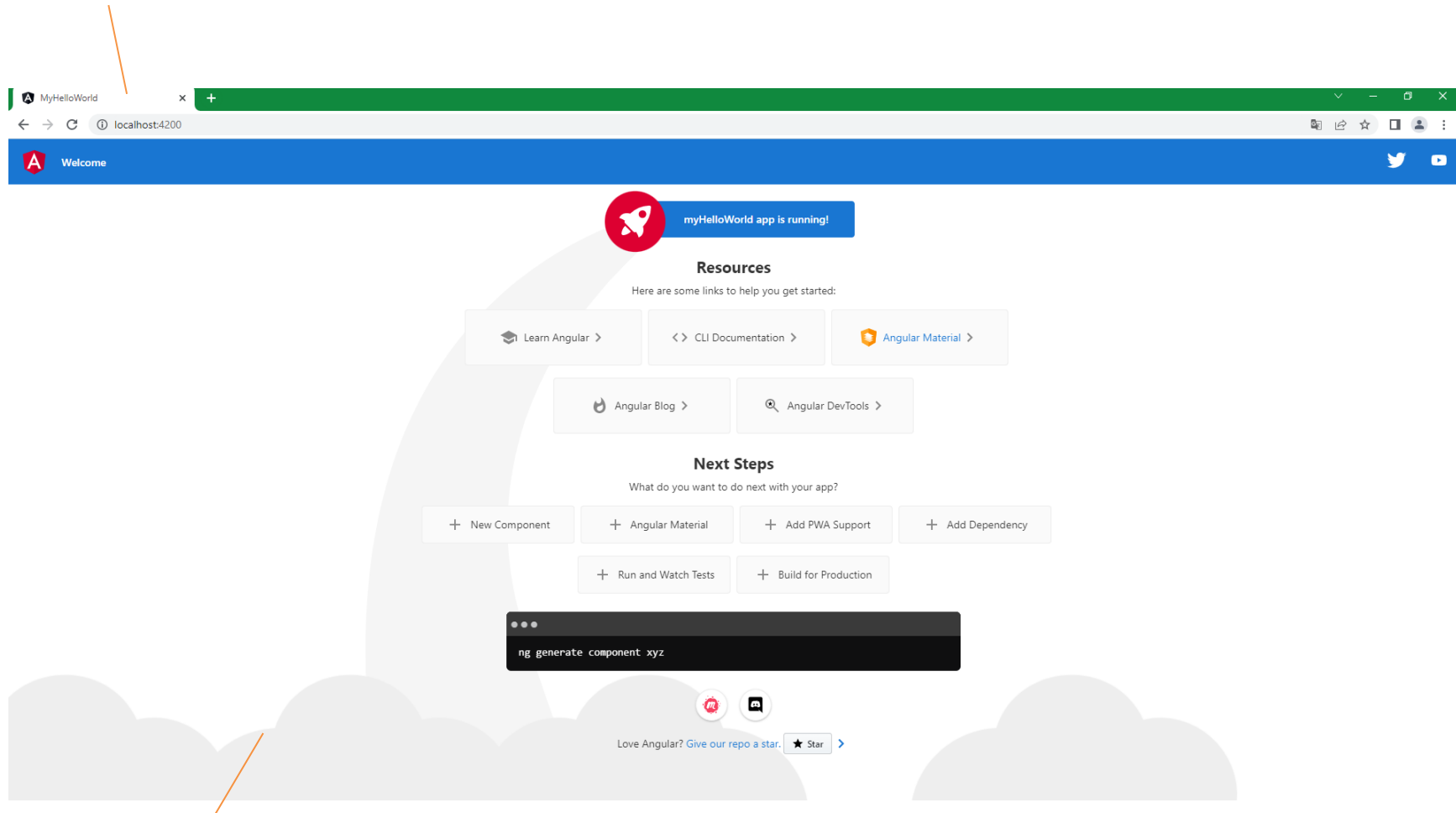
// przejście do katalogu projektu
// uruchomienie serwera

```
PS D:\helloworld> cd .\myHelloWorld\  
PS D:\helloworld\myHelloWorld> ng serve  
✓ Browser application bundle generation complete.  
  
Initial Chunk Files | Names | Raw Size |  
vendor.js | vendor | 1.77 MB |  
polyfills.js | polyfills | 318.06 kB |  
styles.css, styles.js | styles | 210.09 kB |  
main.js | main | 48.05 kB |  
runtime.js | runtime | 6.52 kB |  
  
| Initial Total | 2.34 MB |  
  
Build at: 2022-10-24T12:44:09.270Z - Hash: 33e6ac1e37b11bad - Time: 80208ms  
  
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **  
  
✓ Compiled successfully.  
█
```

został uruchomiony Live Development Server nasłuchujący na adresie localhost:4200

Tworzenie nowego projektu

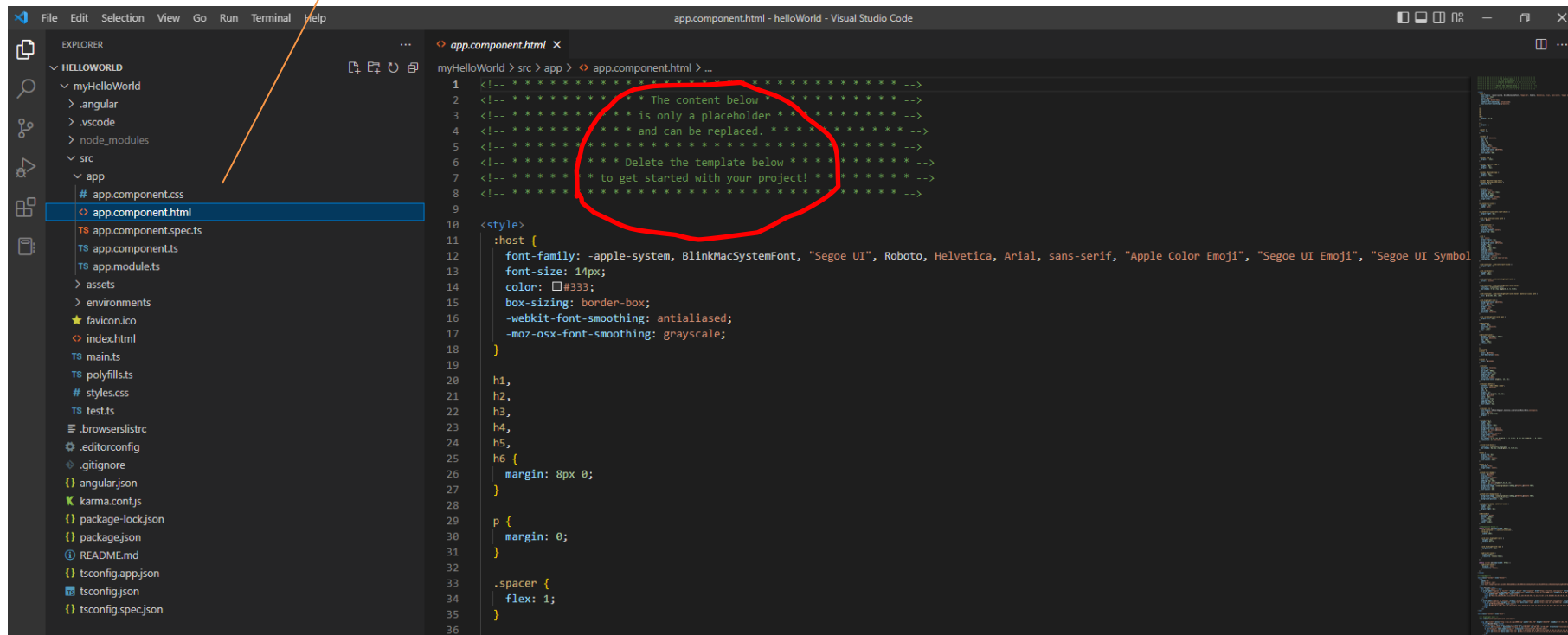
w pasku adresu przeglądarki wpisujemy:
localhost:4200



strona nowego projektu

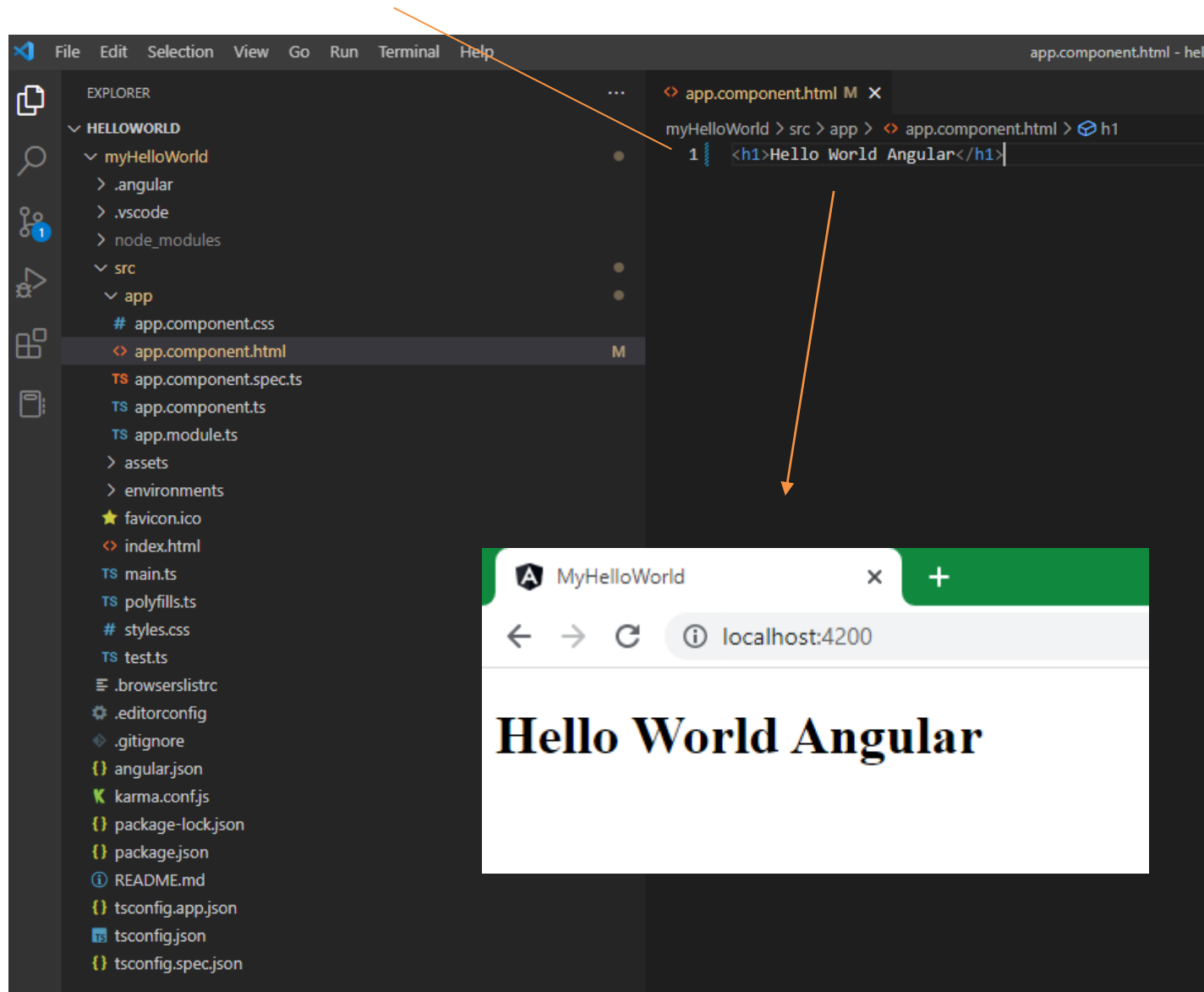
Tworzenie nowego projektu

usuwamy zawartość pliku app.component.html (zawartość startowej strony)

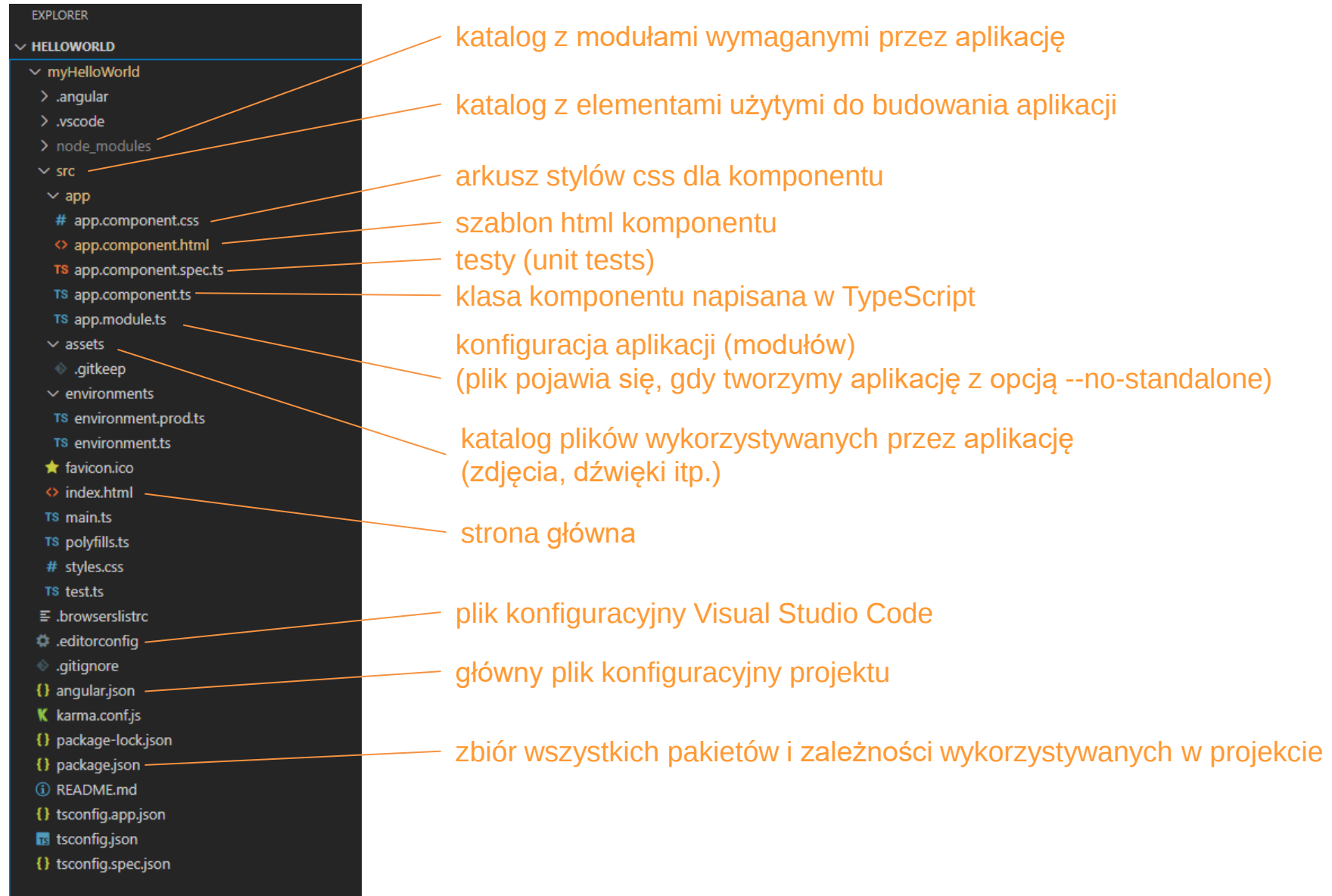


Tworzenie nowego projektu

w pliku `app.component.html` wpisujemy nasz kod



Struktura projektu



app.component.ts

definicja komponentu AppComponent w postaci klasy z
zaimportowanymi modułami (bibliotekami)
(główny komponent aplikacji)

dekorator
(konfiguracja
komponentu)

szablon HTML
powiązany
z komponentem

```
TS app.component.ts X
myHelloWorld > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { RouterOutlet } from '@angular/router';
4
5  @Component({
6    selector: 'app-root',
7    standalone: true,
8    imports: [CommonModule, RouterOutlet],
9    templateUrl: './app.component.html',
10   styleUrls: ['./app.component.css']
11 })
12 export class AppComponent {
13   title = 'myHelloWorld';
14 }
```

wczytanie modułów (dodatkowych bibliotek):
@angular/core - zarządza pracą komponentów
@angular/common - eksportuje wszystkie podstawowe dyrektywy i potoki Angulara
@angular/router - umożliwia nawigację z jednego widoku do drugiego, gdy użytkownicy wykonują zadania aplikacji.

w selektorze <app-root>
w pliku index.html
Angular umieści zawartość
szablону HTML

```
<body>
| <app-root></app-root>
</body>
```

```
<h1>Hello World Angular</h1>
```

szablon app.component.html zawiera
stronę wyświetlaną przez aplikację

pole klasy o nazwie title

arkusz stylów
powiązany
z szablonem

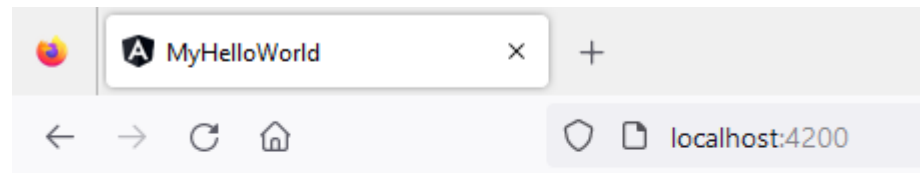
Hello World Angular

app.component.ts

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

@Component({
  selector: 'app-root',
  template: '<p> to jest template </p>',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'myHelloWorld';
}
```

template można umieścić
bezpośrednio w pliku



to jest template

interpolacja

app.component.ts

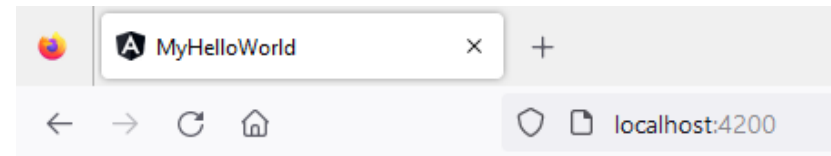
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `
    <style>
    p {
      background-color: yellow;
      font-size: 30px;
      color:red;
    }
    </style>
    <p>{{welcome}}</p>
  `,
  //templateUrl: './app.component.html',
  //styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'myHelloWorld';
  welcome: string = "Powitać!"
}
```

template wielowierszowy znajduje się pomiędzy znakami backtick



interpolacja – nazwę pola wstawiamy między dwa nawiasy klamrowe (string interpolation)



Powitać!

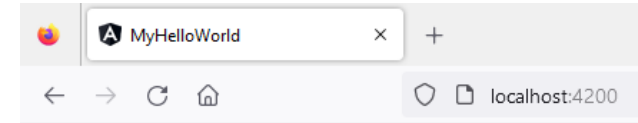
nowe pole o nazwie welcome o wartości typu string „powitać”

interpolacja - wstawianie wyrażenia, wywoływanie metody

app.component.ts

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

@Component({
  selector: 'app-root',
  template: `
    <style>
    p {
      background-color: yellow;
      font-size: 30px;
      color:red;
    }
    </style>
    <p>{{welcome}}</p>
    <h2> 2*3={{2*3}}</h2>
    <h2>{{welcome.toUpperCase()}}
  `,
  //templateUrl: './app.component.html',
  //styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'myHelloWorld';
  welcome: string = "Powitać!"
}
```



wyrażenie

Powitać!

2*3=6

wywołanie
metody

POWITAĆ!

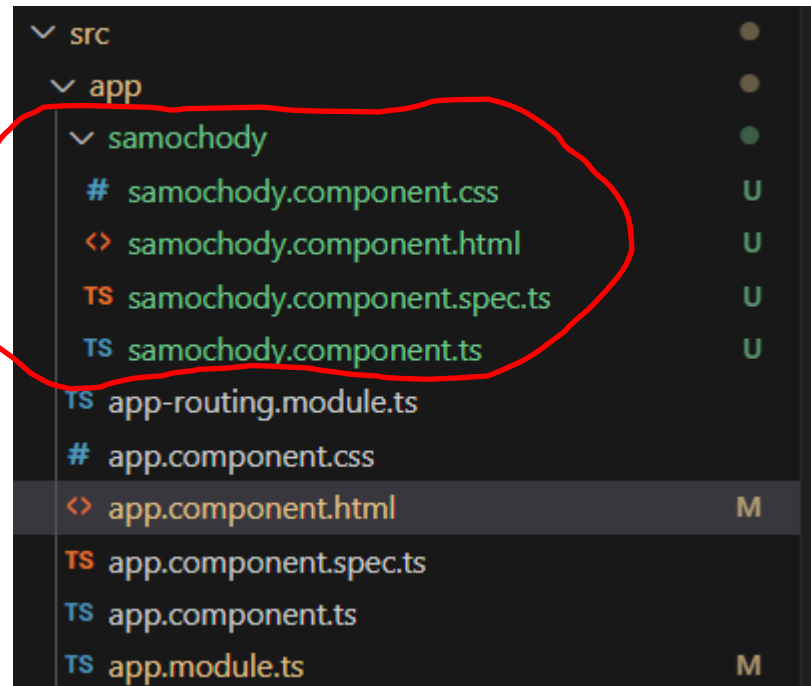
utworzenie nowego komponentu samochodu (aplikacja z komponentami typu no-standalone)

```
ng g c samochody
```

polecenie utworzenia komponentu samochodu:
g: generate
c: component
(wydajemy je z wnętrza katalogu projektu!)

```
CREATE src/app/samochody/samochody.component.html (25 bytes)  
CREATE src/app/samochody/samochody.component.spec.ts (645 bytes)  
CREATE src/app/samochody/samochody.component.ts (221 bytes)  
CREATE src/app/samochody/samochody.component.css (0 bytes)  
UPDATE src/app/app.module.ts (487 bytes)
```

rejestracja nowego komponentu w
app.module.ts (w NgModule)



wygenerowane pliki

utworzenie nowego komponentu samochodu (aplikacja z komponentami typu no-standalone)

```
TS samochody.component.ts U X
```

```
_hello > my-app > src > app > samochody > TS samochody.component.ts > ...  
1   import { Component } from '@angular/core';  
2  
3   @Component({  
4     selector: 'app-samochody',  
5     templateUrl: './samochody.component.html',  
6     styleUrls: ['./samochody.component.css']  
7   })  
8   export class SamochodyComponent {  
9     auta: string = "tu pojawia się najnowsze samochody";  
10  }
```

```
<> samochody.component.html U X
_hello > my-app > src > app > samochody >
Go to component
1 <p>{{auta}}</p>
2
```



The screenshot shows a web browser window. The top bar contains a tab labeled 'MyApp' with a close button (X) and a plus sign (+) for new tabs. Below the tab bar, the address bar shows navigation icons (back, forward, refresh, home) and the URL 'localhost:4200' preceded by a shield icon representing the security status.

tu pojawią się najnowsze samochody

```

<> app.component.html M X
_hello > my-app > src > app > <> app.component.html >  app-samochody
Go to component
1  <app-samochody></app-samochody>

```

utworzenie nowego komponentu samochodu (aplikacja z komponentami typu standalone)

```
ng g c samochody
```

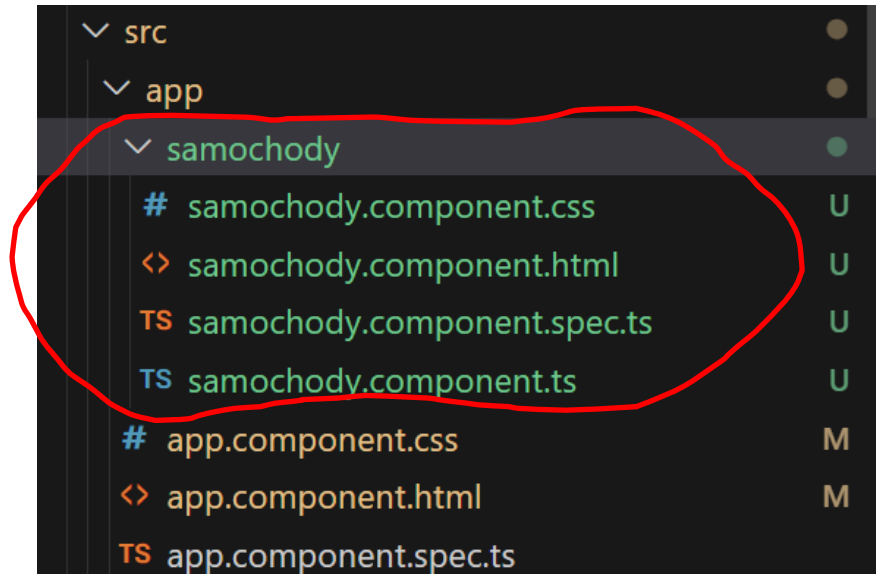
polecenie utworzenia komponentu samochodu:

g: generate

c: component

(wydajemy je z wnętrza katalogu projektu!)

```
CREATE src/app/samochody/samochody.component.html (25 bytes)
CREATE src/app/samochody/samochody.component.spec.ts (640 bytes)
CREATE src/app/samochody/samochody.component.ts (258 bytes)
CREATE src/app/samochody/samochody.component.css (0 bytes)
```



wygenerowane składniki komponentu
(brak rejestracji nowego komponentu
– komponent jest typu standalone)

utworzenie nowego komponentu samochodu (aplikacja z komponentami typu standalone)

```
TS app.component.ts M X
_tmp > tmp > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { RouterOutlet } from '@angular/router';
4  import { SamochodyComponent } from './samochody/samochody.component';
5
6  @Component({
7    selector: 'app-root',
8    standalone: true,
9    imports: [CommonModule, RouterOutlet, SamochodyComponent],
10   templateUrl: './app.component.html',
11   styleUrls: ['./app.component.css']
12 })
13 export class AppComponent {
14   title = 'tmp';
15 }
```

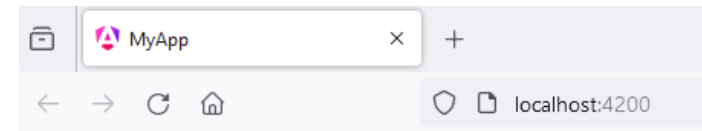
import samodzielnego komponentu SamochodyComponent
do komponentu App.Component w pliku app.component.ts

utworzenie nowego komponentu samochodu (aplikacja z komponentami typu standalone)

```
TS samochody.component.ts U X
_tmp > tmp > src > app > samochody > TS samochody.component.ts > ...
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-samochody',
5    standalone: true,
6    imports: [],
7    templateUrl: './samochody.component.html',
8    styleUrls: ['./samochody.component.css']
9  })
10 export class SamochodyComponent {
11    auta: string = "tu pojawia się najnowsze samochody";
12 }
```

```
<> samochody.component.html U X
_tmp > tmp > src > app > samochody > <> samochody.component.html >
Go to component
1  <p>{{auta}}</p>
```

```
<> app.component.html M X
_tmp > tmp > src > app > <> app.component.html > ...
Go to component
1  <app-samochody></app-samochody>
```



tu pojawia się najnowsze samochody

wiązanie właściwości (property binding)

Galeria Pienińska



wiązanie właściwości (property binding)

powiązanie tworzy aktywne połączenie pomiędzy częścią interfejsu użytkownika utworzoną na podstawie szablonu (elementem DOM, dyrektywą lub komponentem) a modelem (instancją komponentu, do którego należy szablon).

app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  galleryTitle: string = "Galeria Pienińska";
  zdj2: string = "assets/pictures/zdj2.jpg";
  zdj3: string = "assets/pictures/zdj3.jpg";
  zdj4: string = "assets/pictures/zdj4.jpg";
  aktywuj: boolean = true;
  wysrodkuj: string = "center";
}
```

nowe pola w klasie
AppComponent

style

app.component.css

```
.image{
  padding:10px;
  border: solid 1px green;
  width:22%;
}
#container{
  background-color: rgb(175, 251, 108);
  width:740px;
  margin-left: auto;
  margin-right: auto;
}
```

pola w klasie
AppComponent

zmiana stylu app.component.html

```
<div id="container">
  <h1 [style.text-align]="wysrodkuj">{{galleryTitle}}</h1>
  
  <img src={{zdj2}} [class.image] = "aktywuj">
  <img [src] = zdj3 [class.image] = "aktywuj">
  <img [src] = zdj4 [class.image] = "aktywuj">
</div>
```

wiązanie
właściwości

pola w klasie
AppComponent

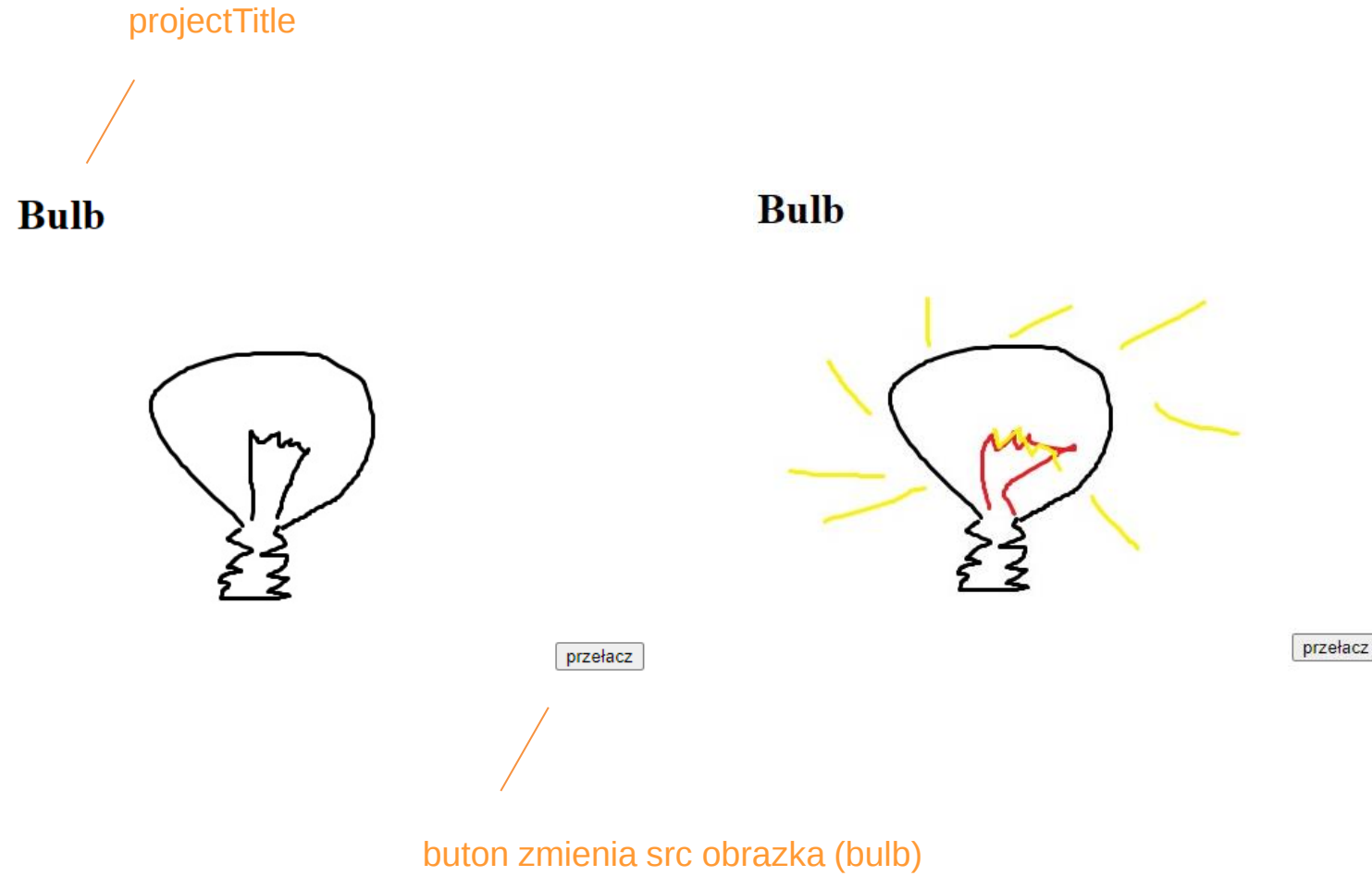
pole w klasie
AppComponent

klasa image

aktywacja
klasy image

<https://angular.io/guide/binding-overview>

wiązanie zdarzeń (event binding)



wiązanie zdarzeń (event binding)

```
<> app.component.html M X  
bulbEvent > src > app > <> app.component.html > ...  
1 <h1>{{projectTitle}}</h1>  
2 <img [src]="bulb">  
3 <button (click)="pstryk()">przełącz</button>
```

interpolacja tytułu strony

związanie właściwości src
obrazka do pola bulb

```
TS app.component.ts M X  
bulbEvent > src > app > TS app.component.ts > ...  
1 import { Component } from '@angular/core';  
2  
3 @Component({  
4   selector: 'app-root',  
5   templateUrl: './app.component.html',  
6   styleUrls: ['./app.component.css']  
7 })  
8 export class AppComponent {  
9   projectName: string = "Bulb";  
10  bulb: string = "assets/pictures/turnedOff.jpg";  
11  
12  pstryk(){  
13    this.bulb = this.bulb == "assets/pictures/turnedOff.jpg" ? "assets/pictures/turnedOn.jpg": "assets/pictures/turnedOff.jpg";  
14  }  
15 }
```

pole tytułu strony

pole bulb (src obrazka)

zdarzenie click przycisku jest
związane z metodą pstryk()
podmieniającą src obrazka

wiązanie dwukierunkowe (two-way binding)

zmiany w szablonie
są odzwierciedlone
w kodzie i na odwrót

Zagadka

$2 * 3 = 23$

to co wpisujemy w pole input pokazuje
się jako wynik

dane z widoku (np. zmiana w
szablonie) są przekazywane do
komponentu (kodu TypeScript) i na
odwrót

wiązanie dwukierunkowe (two-way binding)

Zagadka

$$2 * 3 = 23$$

nok

rezultat sprawdzenia
wyniku

przycisk sprawdzający wynik

Zagadka

$$2 * 3 = 6$$

ok

zawartość formularza jest
wpisywana do pola iloczyn

zawartość pola iloczyn
jest wpisywana do wyniku

wiązanie dwukierunkowe (two-way binding)

```
<> app.component.html M X
zagadka > src > app > <> app.component.html > ...
1 <h1>Zagadka</h1>
2 <h2>2 * 3 = {{iloczyn}}</h2>
3 <input type="text" [(ngModel)] = "iloczyn">
4 <button (click)="check()">check</button>
5 <p>{{rezultat}}</p>
```

interpolacja pola iloczyn

wiązanie dwukierunkowe pola iloczyn, dzięki niemu gdy zmieni się input (np. użytkownik wpisze liczbę) w kodzie ta zmiana jest odwzorowana w polu iloczyn

pole rezultat w szablonie

metoda sprawdzająca wartość pola iloczyn i wypisująca rezultat

```
TS app.component.ts M X
zagadka > src > app > TS app.component.ts > ...
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   iloczyn: string="";
10  rezultat: string="";
11
12  check(){
13    if(this.iloczyn=="6"){
14      this.rezultat="ok";
15    }else{
16      this.rezultat="nok";
17    }
18  }
19 }
```

pole iloczyn

pole rezultat

wiązanie dwukierunkowe (two-way binding)

```
TS app.module.ts M X
zagadka > src > app > TS app.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { FormsModule } from '@angular/forms';
4  import { AppComponent } from './app.component';
5
6  @NgModule({
7    declarations: [
8      AppComponent
9    ],
10   imports: [
11     BrowserModule,
12     FormsModule
13   ],
14   providers: [],
15   bootstrap: [AppComponent]
16 })
17 export class AppModule { }
```

import modułu odpowiedzialnego za obsługę formularza

wiązanie dwukierunkowe (two-way binding)

rozwiązanie zagadki

Zagadka

$$7 + 9 = 16$$

ok



Zagadka

$$6 - 8 = -2$$

ok

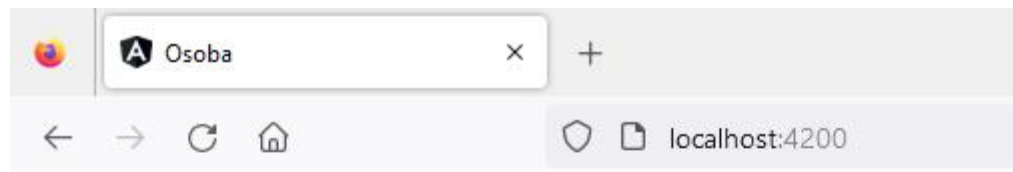


Zagadka

$$5 * 4 =$$

każdorazowo po podaniu
prawidłowego wyniku losowana jest
kolejna zagadka (liczby oraz operator)

Sluchacz



Kwestionariusz osobowy słuchacza



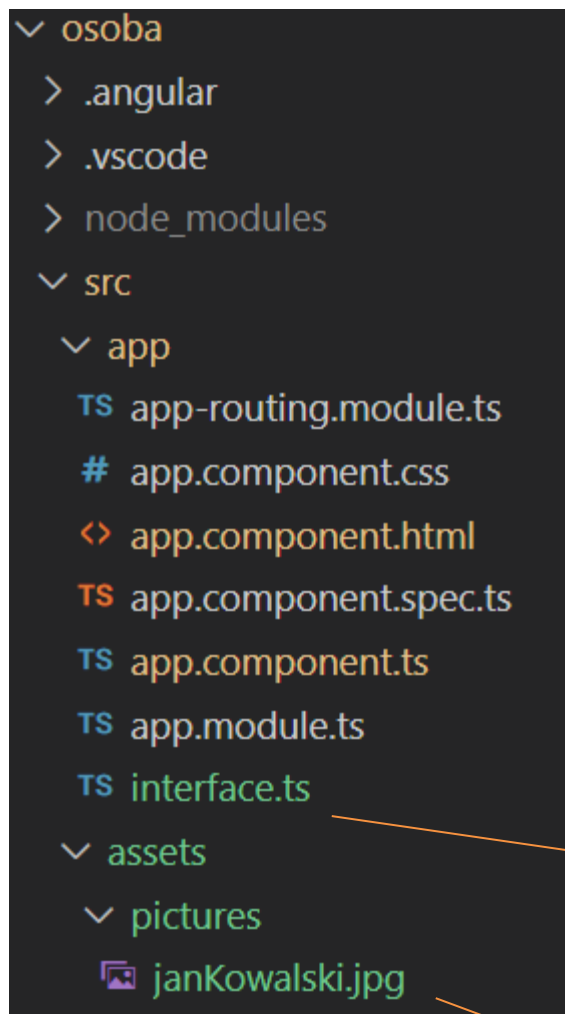
dane słuchacza
pobieramy z obiektu
sluchacz

imię: Jan

nazwisko: Kowalski

kurs kwalifikacyjny: technik programista INF.04

interface.ts



```
TS interface.ts U X
osoba > src > app > TS interface.ts > ...
1  export interface Sluchacz{
2      imie: string;
3      nazwisko: string;
4      kursKwalifikacyjny: string;
5      zdjecie: string;
6  }
```

w katalogu assets
umieszczamy zdjęcie
słuchacza

w pliku interface.ts
(nazwa jest nasza)
umieszczamy interfejs
Sluchacz oraz go
eksportujemy

app.component.ts

TS app.component.ts M X

osoba > src > app > TS app.component.ts > ...

```
1  import { Component } from '@angular/core';
2  import { Sluchacz } from './interface'
3
4  @Component({
5    selector: 'app-root',
6    templateUrl: './app.component.html',
7    styleUrls: ['./app.component.css']
8  })
9  export class AppComponent {
10    sluchacz: Sluchacz = {
11      imie: "Jan",
12      nazwisko: "Kowalski",
13      kursKwalifikacyjny: "technik programista INF.04",
14      zdjecie: "assets/pictures/janKowalski.jpg"
15    }
16
17  }
```

import interfejsu
Sluchacz

obiekt sluchacz jest stworzony na podstawie
zaimportowanego interfejsu

app.component.html

```
osoba > src > app > <> app.component.html > ...
1 <div id="container">
2   <h1>Kwestionariusz osobowy słuchacza</h1>
3   <img src={{sluchacz.zdjecie}}>
4   <p>imię: {{sluchacz.imie}}</p>
5   <p>nazwisko: {{sluchacz.nazwisko}}</p>
6   <p>kurs kwalifikacyjny: {{sluchacz.kursKwalifikacyjny}}</p>
7 </div>
```



wykorzystujemy interpolację

Dyrektywy



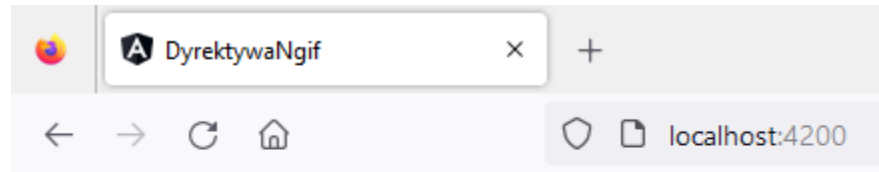
umożliwiają dostęp do struktury dokumentu, mogą zmieniać wygląd i zachowanie danego elementu strony

- komponenty (*Component Directives*)
- dyrektywy strukturalne (*Structural Directives*)
- dyrektywy atrybutowe (*Attribute Directives*)

Dyrektywa strukturalna ngIf

służy do dodawania lub usuwania elementów HTML na stronie (jeśli ma wartość true – element jest uwzględniony w drzewie dokumentów, jeśli false – nie jest uwzględniany).

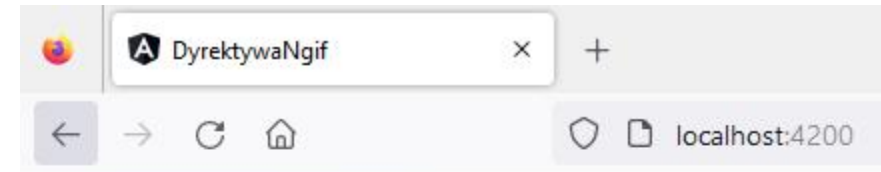
z pomocą dyrektywy ngIf obrazek pojawia się i znika ze strony



toggle

brak obrazka na stronie

przycisk przełącza wartość dyrektywy ngIf z false na true (i na odwrót, dyrektywa jest umieszczona w znaczniku img)



obrazek pojawia się na stronie

toggle

Dyrektywa strukturalna ngIf

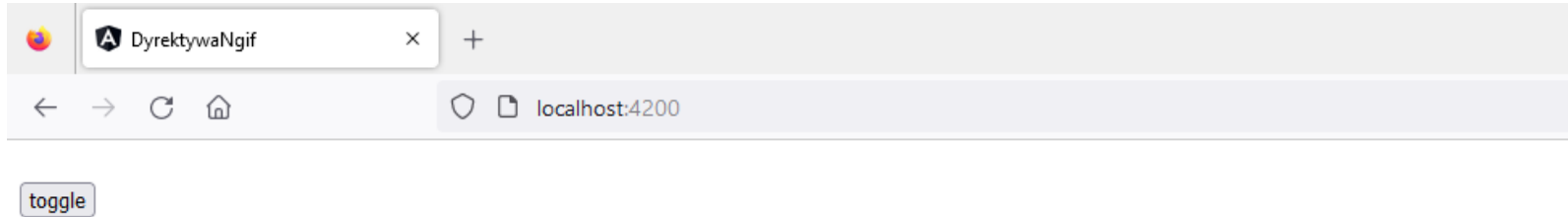
```
TS app.component.ts M X
dyrektywaNgif > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    template: `
6      
7      <br>
8      <button (click)="toggle()">toggle</button>
9    `,
10   styleUrls: ['./app.component.css']
11 })
12 export class AppComponent {
13   showImage: boolean = false;
14   toggle(){
15     this.showImage = !this.showImage;
16   }
17 }
```

do dyrektywy ngIf jest przypisana wartość pola showImage

pole showImage jest zainicjowane wartością false

metoda toggle() (triggerowana przez przycisk) przełącza wartość dyrektywy z false na true i na odwrót

sposób działania nglf



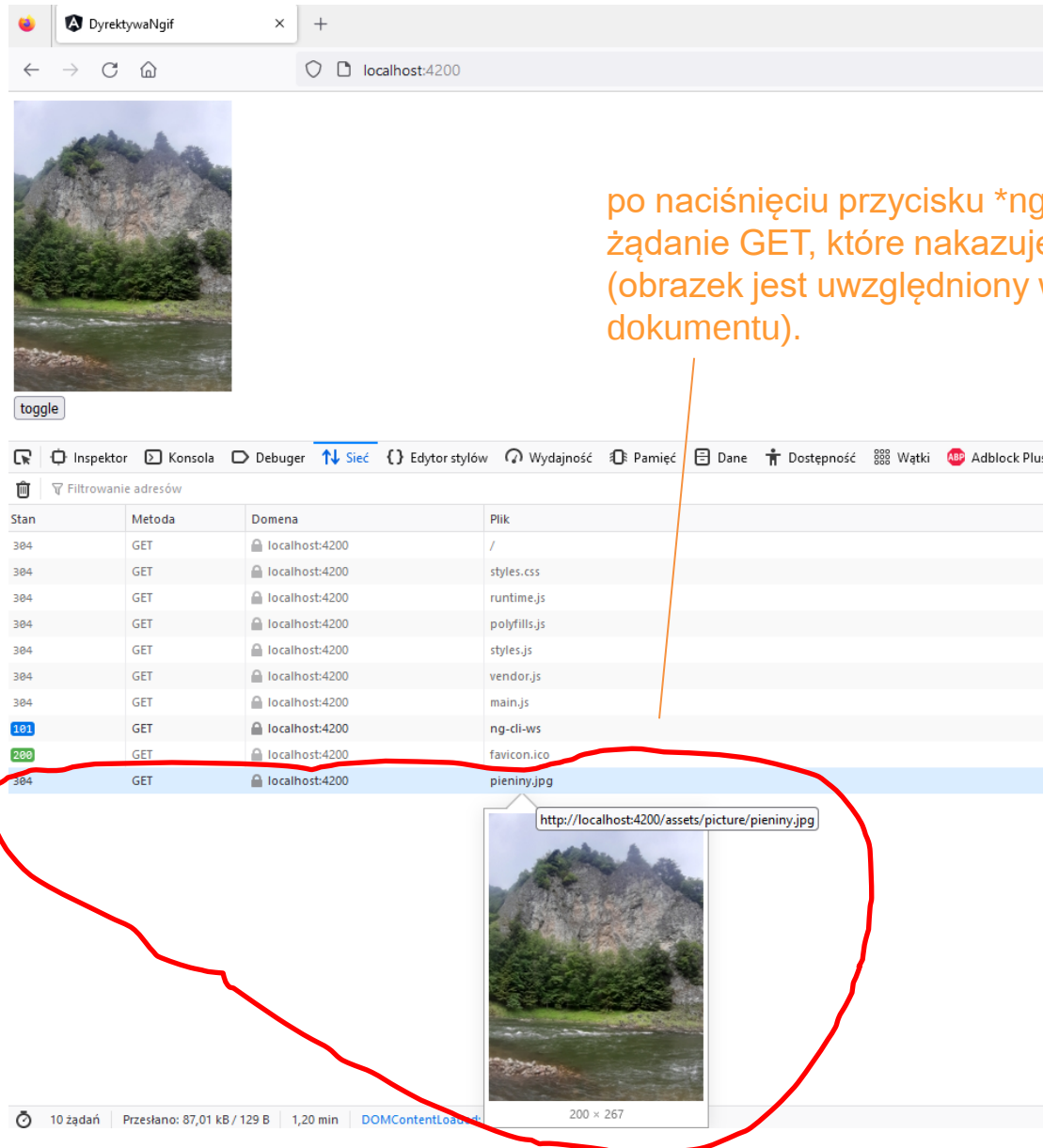
po początkowym załadowaniu strony *nglf = false, stąd brak żądania GET, które nakazuje pobranie obrazka (obrazek nie jest uwzględniony w drzewie dokumentu).

F12 – narzędzia deweloperskie

Stan	Metoda	Domena	Plik
304	GET	localhost:4200	/
304	GET	localhost:4200	styles.css
304	GET	localhost:4200	runtime.js
304	GET	localhost:4200	polyfills.js
304	GET	localhost:4200	styles.js
304	GET	localhost:4200	vendor.js
304	GET	localhost:4200	main.js
101	GET	localhost:4200	ng-cli-ws
200	GET	localhost:4200	favicon.ico

sposób działania ngIf

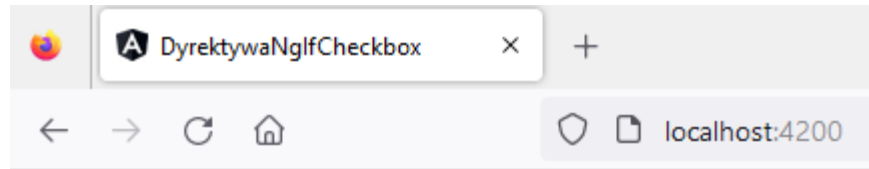
po naciśnięciu przycisku *ngIf = true, stąd
żądanie GET, które nakazuje pobranie obrazka
(obrazek jest uwzględniony w drzewie
dokumentu).



Stan	Metoda	Domena	Plik
304	GET	localhost:4200	/
304	GET	localhost:4200	styles.css
304	GET	localhost:4200	runtime.js
304	GET	localhost:4200	polyfills.js
304	GET	localhost:4200	styles.js
304	GET	localhost:4200	vendor.js
304	GET	localhost:4200	main.js
101	GET	localhost:4200	ng-cli-ws
200	GET	localhost:4200	favicon.ico
304	GET	localhost:4200	pieniny.jpg

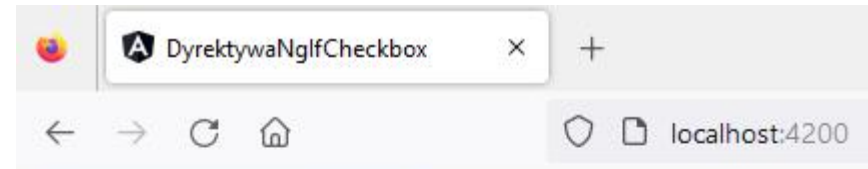
10 żądań | Przesłano: 87,01 kB / 129 B | 1,20 min | DOMContentLoaded

Dyrektywa strukturalna nglf z checkbox



☐ show image

zamiast przycisku checkbox



☒ show image

Dyrektywa strukturalna ngIf z checkbox

TS app.component.ts M X

dyrektywaNgIfCheckbox > src > app > TS app.component.ts > ...

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    template: `
6      
7      <br>
8      <!--<button (click)="toggle()">toggle</button>-->
9      <label><input type="checkbox" [(ngModel)] = showImage > show image </label>
10   `,
11    styleUrls: ['./app.component.css']
12  })
13  export class AppComponent {
14    showImage: boolean = false;
15    //toggle(){
16    //  this.showImage = !this.showImage;
17    //}
18  }
```

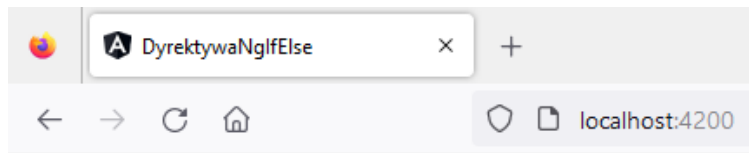
checkbox powiązany z polem
showImage dyrektywą ngModel

Dyrektywa strukturalna ngIf z checkbox

```
TS app.module.ts M X
zagadka > src > app > TS app.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { FormsModule } from '@angular/forms';
4  import { AppComponent } from './app.component';
5
6  @NgModule({
7    declarations: [
8      AppComponent
9    ],
10   imports: [
11     BrowserModule,
12     FormsModule
13   ],
14   providers: [],
15   bootstrap: [AppComponent]
16 })
17 export class AppModule { }
```

import modułu odpowiedzialnego za obsługę formularza

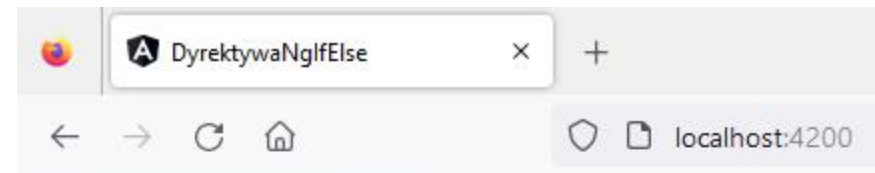
Dyrektywa strukturalna ngIf z else



Nic nie wyświetlam

toggle

efekt działania else przy dyrektywie ngIf



toggle

Dyrektywa strukturalna ngIf z else

```
TS app.component.ts M X
dyrektywaNgIfElse > src > app > TS app.component.ts > AppComponent
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    template: `
6      
7      <br>
8      <button (click)="toggle()">toggle</button>
9      <ng-template #showNothing>
10        <p>{{information}}</p>
11      </ng-template>
12    `,
13    styleUrls: ['./app.component.css']
14  })
15  export class AppComponent {
16    showImage: boolean = false;
17    information: string = "Nic nie wyświetlam";
18    toggle(){
19      this.showImage = !this.showImage;
20    }
21  }
```

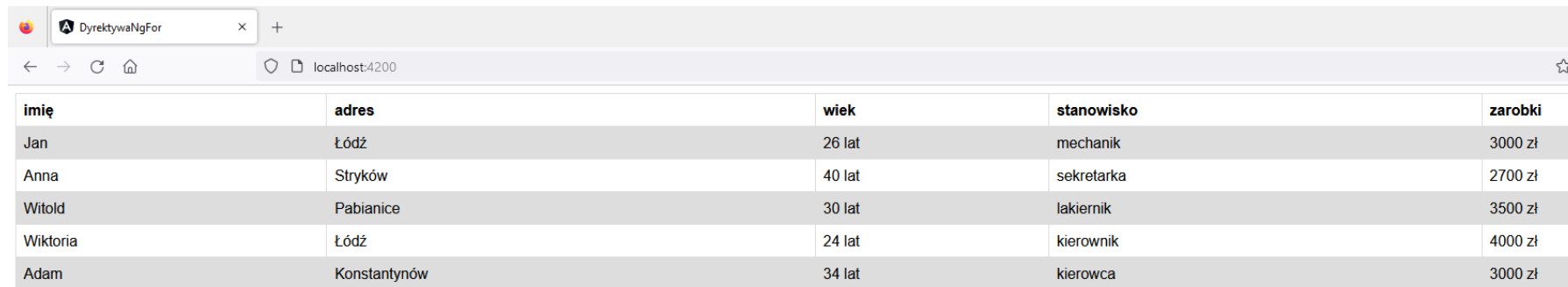
przy dyrektywie ngIf jest else, który zamiast zdjęcia wyświetla zawartość znacznika <ng-template #showNothing>

Dyrektywa strukturalna ngFor

zmienia strukturę drzewa DOM poprzez iterowanie przez tablicę obiektów

*ngFor = "let element of kolekcja"

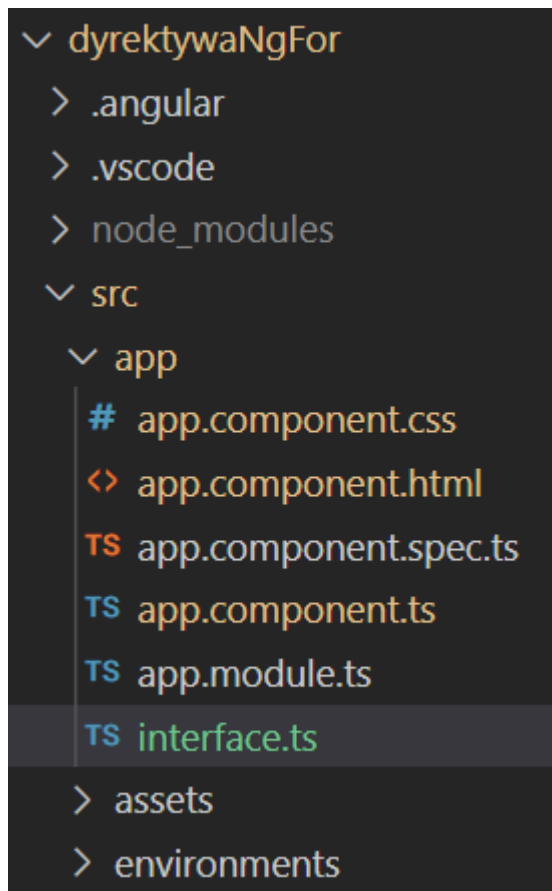
składnia dyrektywy



A screenshot of a web browser window with the title 'DyrektywaNgFor'. The address bar shows 'localhost:4200'. The main content area displays a table with five columns: 'imię', 'adres', 'wiek', 'stanowisko', and 'zarobki'. The table contains five rows of data representing employees.

imię	adres	wiek	stanowisko	zarobki
Jan	Łódź	26 lat	mechanik	3000 zł
Anna	Stryków	40 lat	sekretarka	2700 zł
Witold	Pabianice	30 lat	lakiernik	3500 zł
Wiktoria	Łódź	24 lat	kierownik	4000 zł
Adam	Konstantynów	34 lat	kierowca	3000 zł

Dyrektywa strukturalna ngFor



TS interface.ts U X

dyrektywaNgFor > src > app > TS interface.ts > ...

```
1  export interface Pracownik{
2      imie: string;
3      adres:string;
4      wiek: number;
5      stanowisko:string;
6      zarobki:number;
7  }
```

interjejs Pracownik

nowy plik (nazwa jest nasza)

Dyrektywa strukturalna ngFor

```
TS app.component.ts M X
dyrektywaNgFor > src > app > TS app.component.ts > ...
1 import { Component } from '@angular/core';
2 import { Pracownik } from './interface';
3
4 @Component({
5   selector: 'app-root',
6   templateUrl: './app.component.html',
7   styleUrls: ['./app.component.css']
8 })
9 export class AppComponent {
10   tableHeader: string[] = ['imię', 'adres', 'wiek', 'stanowisko', 'zarobki'];
11   pracownicy: Pracownik[] = [
12     {imię: 'Jan', adres: 'Łódź', wiek: 26, stanowisko: 'mechanik', zarobki: 3000},
13     {imię: 'Anna', adres: 'Stryków', wiek: 40, stanowisko: 'sekretarka', zarobki: 2700},
14     {imię: 'Witold', adres: 'Pabianice', wiek: 30, stanowisko: 'lakiernik', zarobki: 3500},
15     {imię: 'Wiktoria', adres: 'Łódź', wiek: 24, stanowisko: 'kierownik', zarobki: 4000},
16     {imię: 'Adam', adres: 'Konstantynów', wiek: 34, stanowisko: 'kierowca', zarobki: 3000}
17   ];
18 }
```

import interfejsu Pracownik

pole tableHeader - nagłówek
wyświetlanej tabeli

pole pracownicy (tabela typu Pracownik)
- treść wyświetlanej tabeli

Dyrektywa strukturalna ngFor

dyrektywa
iteruje po
tablicy
pracownicy
przypisując
kolejne
obiekty typu
pracownik do
zmiennej
osoba,
poszczególne
pola tej
zmiennej
(obiekту)
wstawiane są
do znaczników
<td>

```
dyrektywaNgFor > src > app > <> app.component.html > ...
1  <table>
2    <tr>
3      <th *ngFor="let header of tableHeader">{{ header }}</th>
4    </tr>
5    <tr *ngFor="let osoba of pracownicy">
6      <td>{{ osoba.imie }}</td>
7      <td>{{ osoba.adres }}</td>
8      <td>{{ osoba.wiek }} lat</td>
9      <td>{{ osoba.stanowisko }}</td>
10     <td>{{ osoba.zarobki }} zł</td>
11   </tr>
12 </table>
```

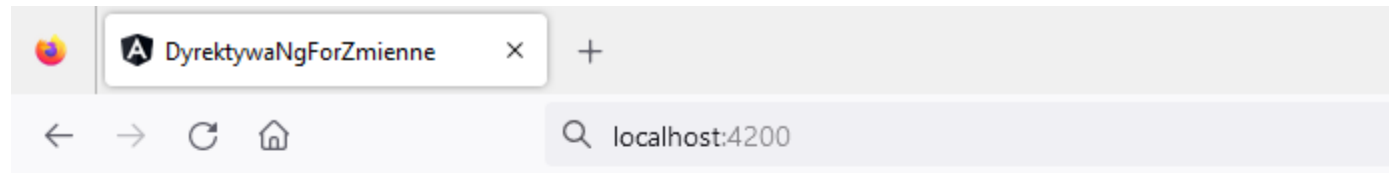
dyrektywa iteruje po tablicy
stringów tableHeader
wstawiając w miejsce
header poszczególne
nagłówki tablicy

imię	adres	wiek	stanowisko	zarobki
Jan	Łódź	26 lat	mechanik	3000 zł
Anna	Stryków	40 lat	sekretarka	2700 zł
Witold	Pabianice	30 lat	lakiernik	3500 zł
Wiktoria	Łódź	24 lat	kierownik	4000 zł
Adam	Konstantynów	34 lat	kierowca	3000 zł

Dyrektywa strukturalna ngFor, zmienne

Nazwa zmiennej	opis
index: number	indeks bieżącego elementu w kolekcji
odd: boolean	true – bieżący element przyjmuje nieparzysty indeks w iterowanym obiekcie
even: boolean	true – bieżący element przyjmuje parzysty indeks w iterowanym obiekcie
first: boolean	true – bieżący element jest pierwszym elementem w iterowanym obiekcie
last: boolean	true – bieżący element jest ostatnim elementem w iterowanym obiekcie
count: number	ilość iterowanych elementów

Dyrektywa strukturalna ngFor, zmienne



nr	imię	adres	wiek	stanowisko	zarobki
1/5	Jan	Łódź	26 lat	mechanik	3000 zł
2/5	Anna	Stryków	40 lat	sekretarka	2700 zł
3/5	Witold	Pabianice	30 lat	lakiernik	3500 zł
4/5	Wiktoria	Łódź	24 lat	kierownik	4000 zł
5/5	Adam	Konstantynów	34 lat	kierowca	3000 zł

za pomocą zmiennych dyrektywy ngFor można zmienić wygląd tabeli z poprzedniego przykładu

Dyrektywa strukturalna ngFor, zmienne

```
<> app.component.html M X
dyrektywaNgForZmienne > src > app > <> app.component.html > ...
1 <table>
2 <tr>
3   <th *ngFor="let header of tableHeader">{{ header }}</th>
4 </tr>
5 <tr *ngFor="let osoba of pracownicy;
6   let i = index;
7   let evenRow = even;
8   let oddRow = odd;
9   let firstRow = first;
10  let lastRow = last;
11  let countRow = count"
12   [class.evenRow] = "evenRow"
13   [class.oddRow] = "oddRow"
14   [class.firstRow] = "firstRow"
15   [class.lastRow] = "lastRow"
16 >
17   <td>{{i + 1}}/{{countRow}}</td>
18   <td>{{ osoba.imie }}</td>
19   <td>{{ osoba.adres }}</td>
20   <td>{{ osoba.wiek }} lat</td>
21   <td>{{ osoba.stanowisko }}</td>
22   <td>{{ osoba.zarobki }} zł</td>
23 </tr>
24 </table>
```

zmienne lokalne

zmienne wykorzystane do zmiany klasy

Dyrektywa strukturalna ngFor, zmienne

```
TS app.component.ts M X
dyrektywaNgForZmienne > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2  import { Pracownik } from './interface';
3
4  @Component({
5    selector: 'app-root',
6    templateUrl: './app.component.html',
7    styleUrls: ['./app.component.css']
8  })
9  export class AppComponent {
10    // nagłówek tabeli
11    tableHeader: string[] = ['nr', 'imię', 'adres', 'wiek', 'stanowisko', 'zarobki'];
12    // tabela z pracownikami
13    pracownicy: Pracownik[] = [
14      {imie: 'Jan', adres: 'Łódź', wiek: 26, stanowisko: 'mechanik', zarobki: 3000},
15      {imie: 'Anna', adres: 'Stryków', wiek: 40, stanowisko: 'sekretarka', zarobki: 2700},
16      {imie: 'Witold', adres: 'Pabianice', wiek: 30, stanowisko: 'lakiernik', zarobki: 3500},
17      {imie: 'Wiktoria', adres: 'Łódź', wiek: 24, stanowisko: 'kierownik', zarobki: 4000},
18      {imie: 'Adam', adres: 'Konstantynów', wiek: 34, stanowisko: 'kierowca', zarobki: 3000}
19    ];
20  }
```

doszła nowa kolumna na indeks

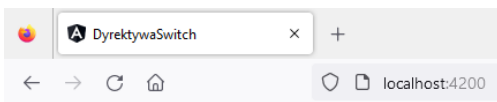
Dyrektywa strukturalna ngFor, zmienne

```
# app.component.css M X
dyrektywaNgForZmienne > src > app > # app.component.css >
1  .evenRow{
2      background-color: ■ DodgerBlue;
3  }
4  .oddRow{
5      background-color: ■ Tomato;
6  }
7  .firstRow{
8      color: ■ red;
9  }
10 .lastRow{
11     color: ■ yellow;
12 }
13 .countRow{
14     text-align: right;
15 }
```

style tabeli załączane zmiennymi
dyrektywy ngFor

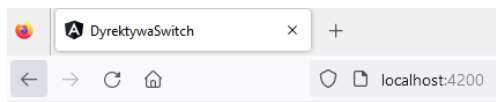
Dyrektywa strukturalna ngSwitch

przełącznik



2

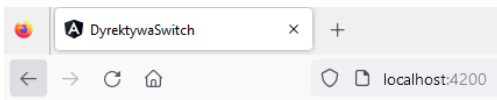
dwa



1

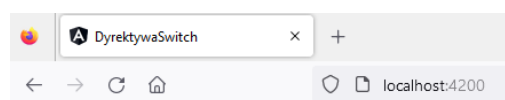
jeden

po przeładowaniu strony wyświetlają
się losowe liczby w zakresie od 1 do 6



4

cztery



6

default

dzięki dyrektywie ngSwitch
wyświetlana jest literowa nazwa liczby

Dyrektywa strukturalna ngSwitch

```
TS app.component.ts M X
dyrektywaSwitch > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    // liczba losowa z zakresu od 1 do 6
10   random: number = Math.floor(Math.random()*6+1);
11 }
```

po przeładowaniu strony do pola
random przypisują się losowe liczby w
zakresie od 1 do 6
pole jest przypisane do dyrektywy
ngSwitch

Dyrektywa strukturalna ngSwitch

```
<> app.component.html M X
dyrektywaSwitch > src > app > <> app.component.html > ...
1  <h1>{{random}}</h1>
2
3  <ng-container [ngSwitch] = "random">
4      <p *ngSwitchCase="1">jeden</p>
5      <p *ngSwitchCase="2">dwa</p>
6      <p *ngSwitchCase="3">trzy</p>
7      <p *ngSwitchCase="4">cztery</p>
8      <p *ngSwitchCase="5">pięć</p>
9      <p *ngSwitchDefault>default</p> <!-- gdy wypadnie 6-->
10 </ng-container>
```

wypisanie losowej
liczby

pole random jest
przypisane do
dyrektywy ngSwitch

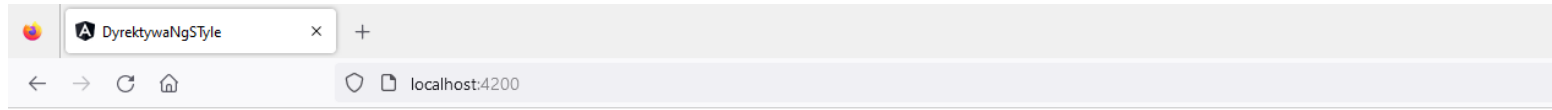
wartość domyślna

kontener
wyświetlający jeden
z paragrafów

dyrektywa ngSwitch przełącza literową nazwę w
zależności od wylosowanej liczby w polu random

Dyrektywa atrybutowa ngStyle

zmiana stylu elementu



napis zmieniający swój styl

zmiana stylu



napis zmieniający swój styl

zmiana stylu

zmiana stylu napisu za pomocą ngStyle

Dyrektywa atrybutowa ngStyle

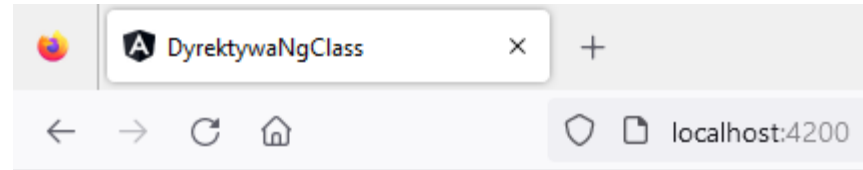
```
TS app.component.ts M X
dyrektywaNgStyle > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2
3  @Component({                                kolory i czcionki wybieramy z tablic
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    index: number = 0;
10   colors: string[] = ["red", "yellow", "green", "blue", "orange"];
11   fonts: string[] = ["Times New Roman", "Arial", "Lucida Handwriting", "Courier New"];
12   increment(){
13     this.index = Math.floor(Math.random() * 6); ——— indeksem w tablicy jest liczba losowa
14   }                                           zmieniana przyciskiem
15 }
```

```
<> app.component.html M X
dyrektywaNgStyle > src > app > <> app.component.html > ...
1  <h1 [ngStyle]="{'color' : colors[index],
2    | 'font-family' : fonts[index],
3    | 'font-size.%' : index*200 + 100
4    | }">
5    napis zmieniający swój styl</h1>
6  <button type="button" (click)="increment()">zmiana stylu</button>
7
```

przycisk zmieniający styl

Dyrektywa atrybutowa ngClass

ustawia klasę elementu



zakupy:

- chleb
- jabłka
- mąka
- ziemniaki

klasa listy jest ustawiona
za pomocą ngStyle

Dyrektywa atrybutowa ngClass

style

```
# app.component.css M X
dyrektywaNgClass > src > app > # app.component.c
1  .red{
2    color: red;
3  }
4  .blue{
5    background-color: blue;
6  }
7  .green{
8    border: 10px solid green;
9  }
```

```
TS app.component.ts M X
dyrektywaNgClass > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    zakupy: string[] = ["chleb", "jabłko", "mąka", "ziemniaki"];
10 }
```

tablica z zakupami

```
<> app.component.html M X
dyrektywaNgClass > src > app > <> app.component.html > ...
1  <h1>zakupy:</h1>
2  <ul>
3    <li *ngFor="let item of zakupy" [ngClass] = "'red green blue'">{{item}}</li>
4  </ul>
```

iteracja przez tablicę zakupów

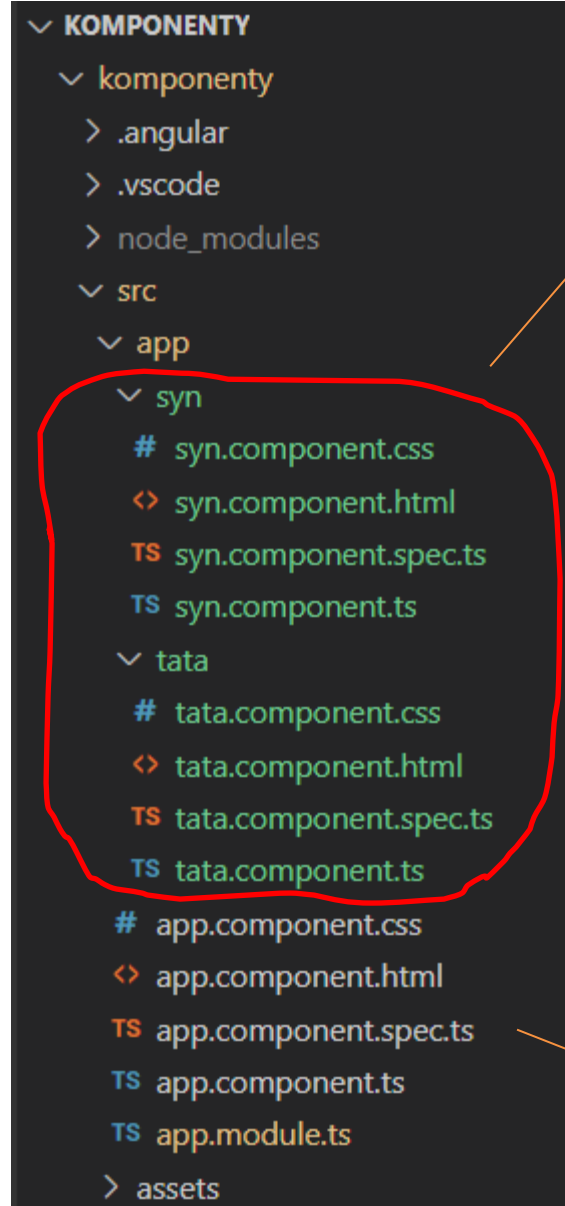
ustawienie klasy znacznika

Komponenty

budują aplikację

każdy nowy
komponent
posiada pliki: css,
html, type script
oraz testowy

generujemy nowe
komponenty tata i syn



ng g c syn

ng g c tata

generate component nazwa komponentu

pliki komponentu głównego, korzenia drzewa
komponentów

Komponenty

```
komponenty > src > app > TS app.module.ts M
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  import { AppComponent } from './app.component';
5  import { TataComponent } from './tata/tata.component';
6  import { SynComponent } from './syn/syn.component';
7
8  @NgModule({
9    declarations: [
10     AppComponent,
11     TataComponent,
12     SynComponent
13   ],
14   imports: [
15     BrowserModule
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
```

w głównym komponencie
automatycznie importują się
nowe komponenty tata i syn

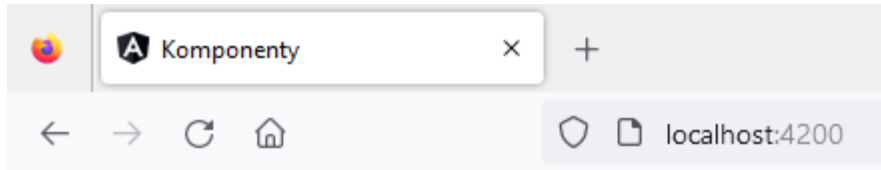
Komponenty

```
<> app.component.html M X
komponenty > src > app > <> app.component.html
1 | <app-tata></app-tata>
```

w głównym
komponencie
zagnieźdźamy
komponent tata
(komponent główny jest
nadrzędny, tata jest
podrzędny)

```
<> tata.component.html U X
komponenty > src > app > tata > <> tata.component.html > ...
1 | <p>tata works!</p>
2 | <app-syn></app-syn>
```

w komponencie tata
zagnieźdźamy
komponent syn
(komponent tata jest
nadrzędny, syn jest
podrzędny)



tata works!

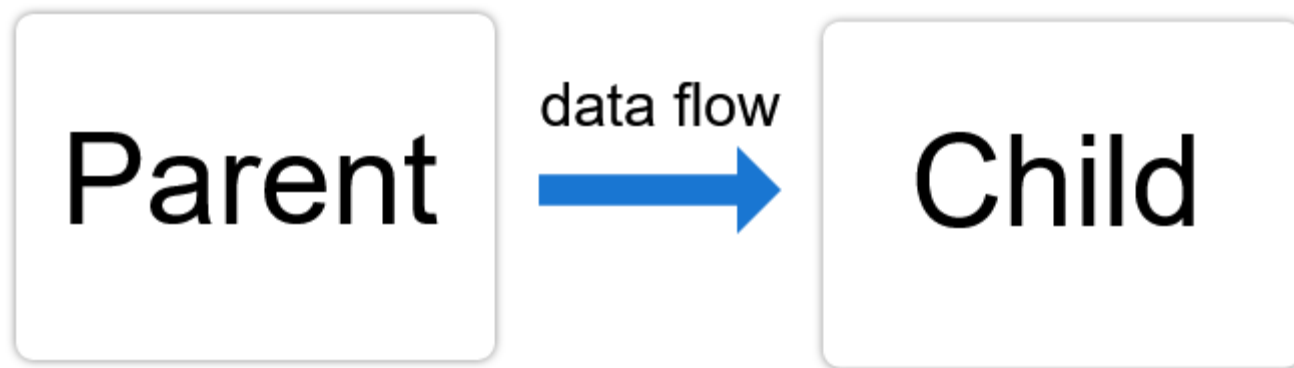
syn works!

komponenty tata i syn
(oraz główny) działają

Komponenty – przesyłanie danych

z komponentu nadrzędnego (tata) do podrzędnego (syn)

@Input



Komponenty – przesyłanie danych

z komponentu nadrzędnego (tata) do podrzędnego (syn)

powiązanie pól komponentu nadrzędnego (tata) z podrzędnym (syn)

tata.component.html

selektor komponentu podrzędnego

target

source

```
<app-syn [infoOdTaty] = "bierzaceInfo"></app-syn>
```

pole @Input komponentu podrzędnego (syna)

pole komponentu nadrzędnego (taty)

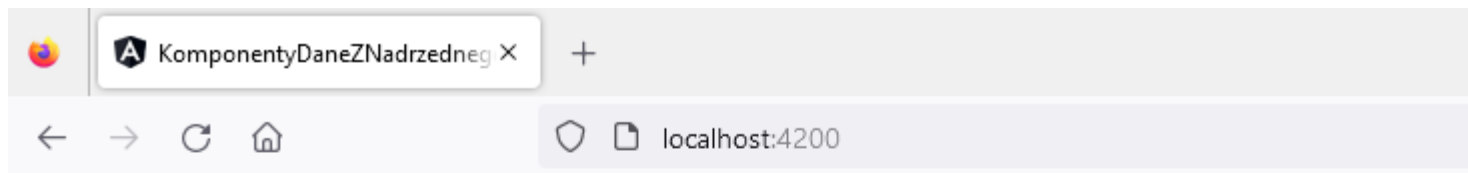
The diagram illustrates an Angular template snippet: `<app-syn [infoOdTaty] = "bierzaceInfo"></app-syn>`. Annotations identify the parts of the snippet:

- `<app-syn` is labeled as the 'selektor komponentu podrzędnego' (child component selector).
- `[infoOdTaty]` is labeled as the 'pole @Input komponentu podrzędnego (syna)' (input field of the child component).
- `=` is labeled as 'target'.
- `"bierzaceInfo"` is labeled as the 'pole komponentu nadrzędnego (taty)' (field of the parent component).
- The entire binding expression is labeled as 'source'.

 A line from the top text 'powiązanie pól komponentu nadrzędnego (tata) z podrzędnym (syn)' points to the binding expression. A line from the file name 'tata.component.html' points to the snippet.

Komponenty – przesyłanie danych

z komponentu nadrzędnego (tata) do podrzędnego (syn)



tata works!

syn works!

Info od taty:

Najważniejsze co się czuje, słuchaj zawsze głosu serca, hej

informacja przesłana z komponentu nadrzędnego (tata) do komponentu podrzędnego (syn)

Komponenty – przesyłanie danych

z komponentu nadrzędnego (tata) do podrzędnego (syn)

1.konfiguracja komponentu podrzędnego (syn)

```
TS syn.component.ts U X
komponentyDaneZNadrzednegoDoPodrzednego > src > app > syn > TS syn.component.ts > ...
1  import { Component, OnInit, Input } from '@angular/core';
2
3  @Component({
4    selector: 'app-syn',
5    templateUrl: './syn.component.html',
6    styleUrls: ['./syn.component.css']
7  })
8  export class SynComponent implements OnInit {
9
10   @Input() infoOdTaty: String = "" ;
11
12   constructor() { }
13
14   ngOnInit(): void {
15   }
```

import komponentu Input

u syna tworzymy pole infoOdTaty z dekoratorem @Input()

```
<> syn.component.html U X
komponentyDaneZNadrzednegoDoPodrzednego > src > app > syn > syn.component.html > ...
1  <p>syn works!</p>
2  <p>Info od taty:</p>
3  <p>{{infoOdTaty}}</p>
```

w pliku html syna posługujemy się np. interpolacją

<https://angular.io/guide/inputs-outputs#configuring-the-child-component>

Komponenty – przesyłanie danych z komponentu nadrzędnego (tata) do podrzędnego (syn)

2.konfiguracja komponentu nadrzędnego (tata)

```
TS tata.component.ts U X
komponentyDaneZNadrzednegoDoPodrzednego > src > app > tata > TS tata.component.ts > ...
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-tata',
5    templateUrl: './tata.component.html',
6    styleUrls: ['./tata.component.css']
7  })
8  export class TataComponent implements OnInit {
9
10     bierzaceInfo: string = "Najważniejsze co się czuje, słuchaj zawsze głosu serca, hej";
11
12     constructor() { }
13
14     ngOnInit(): void {
15     }
16
17 }
```

u taty tworzymy pole bierzaceInfo

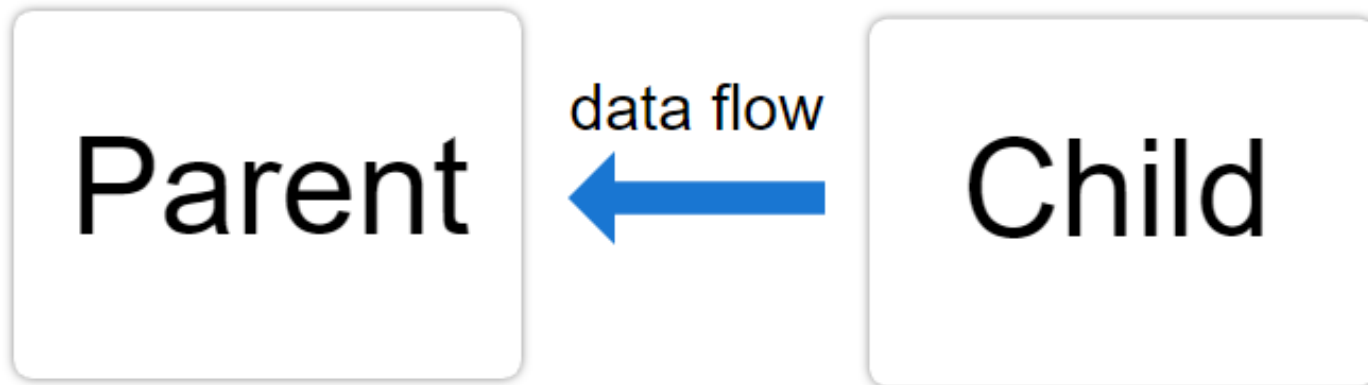
```
<> tata.component.html U X
komponentyDaneZNadrzednegoDoPodrzednego > src > app > tata > <> tata
1  <p>tata works!</p>
2  <app-syn [infoOdTaty] = "bierzaceInfo"></app-syn>
```

w pliku html taty umieszczamy
wiązanie pola syna infoOdTaty
z polem taty bierzaceInfo

Komponenty – przesyłanie danych

z komponentu podrzędnego (syn) do nadrzędnego (tata)

@Output

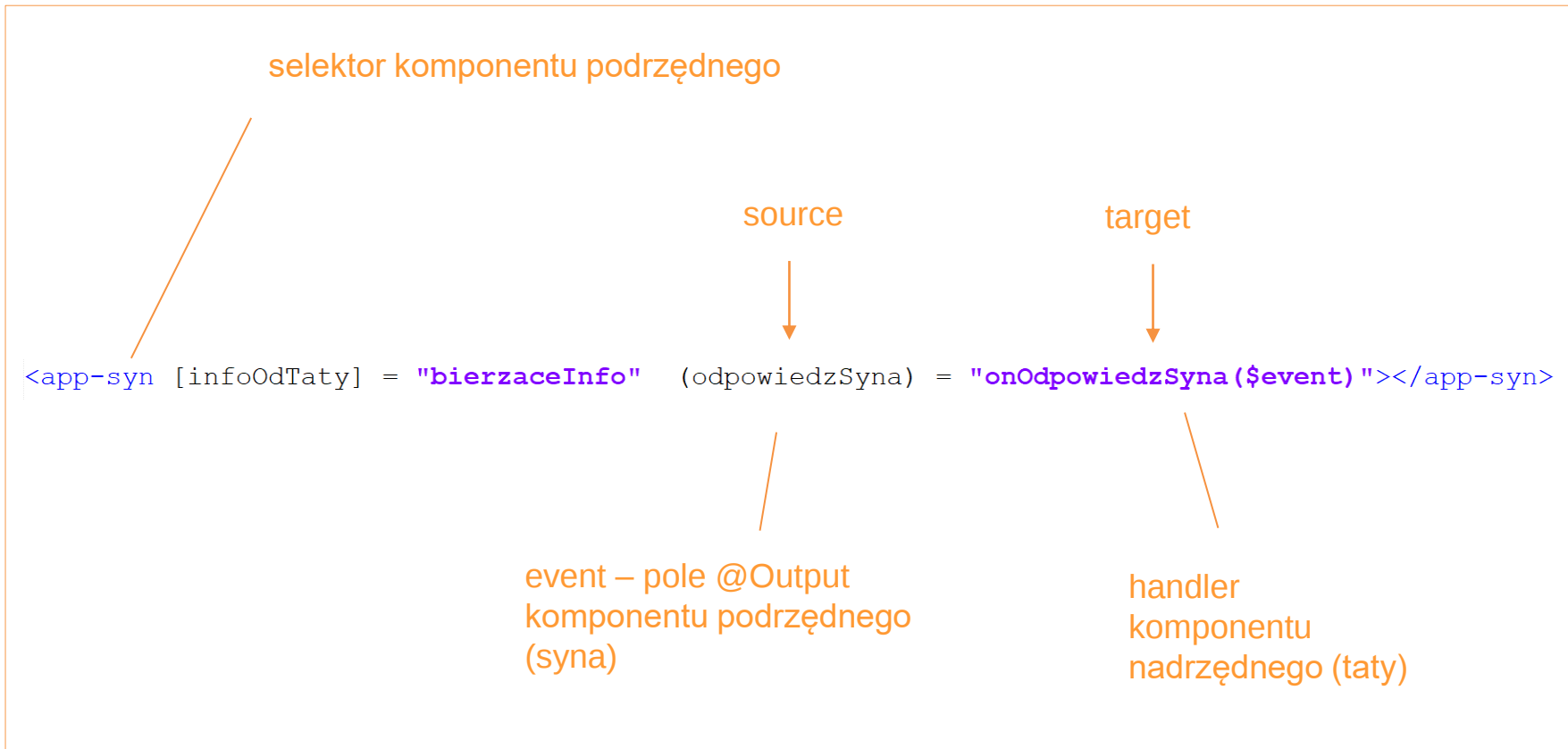


Komponenty – przesyłanie danych

z komponentu podrzędnego (syn) do nadrzędnego (tata)

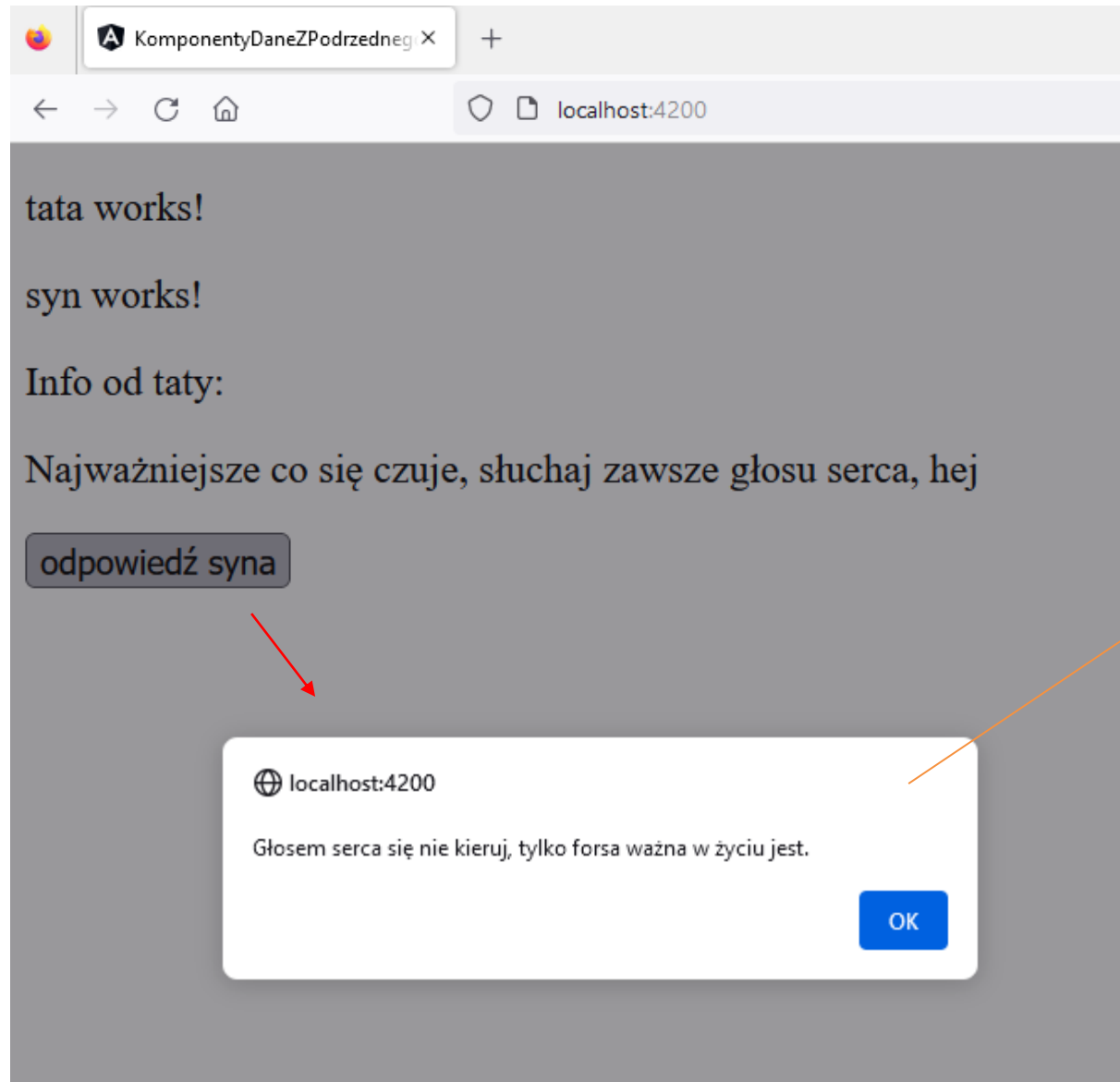
powiązanie pól komponentu podrzędnego (syn) z nadrzędnym (tata) (i na odwrót)

tata.component.html



Komponenty – przesyłanie danych

z komponentu podrzędnego (syn) do nadrzędnego (tata)



informacja przesłana
z komponentu
podrzędnego (syn)
do komponentu
nadrzędnego (tata)

Komponenty – przesyłanie danych z komponentu podrzędnego (syn) do nadrzędnego (tata)

1.konfiguracja komponentu podrzędnego (syn)

```
TS syn.component.ts U X
komponentyDaneZPodrzednegoDoNadrzednego > src > app > syn > TS syn.component.ts > ...
1  import { Component, OnInit ,Input, Output, EventEmitter } from '@angular/core';
2
3  @Component({
4    selector: 'app-syn',
5    templateUrl: './syn.component.html',
6    styleUrls: ['./syn.component.css']
7  })
8  export class SynComponent implements OnInit {
9
10     @Input() infoOdTaty: String = "" ;
11
12     @Output() odpowiedzSyna = new EventEmitter<string>();
13
14     constructor() { }
15
16     ngOnInit(): void {
17     }
18
19     odpowiedz(){
20         this.odpowiedzSyna.emit('Głosem serca się nie kieruj, tylko forsa ważna w życiu jest.');
```

import komponentów
Output, EventEmitter
(ten drugi służy do
przekazywania danych)

będziemy eksportować
dane typu string

u syna tworzymy pole odpowiedzSyna
(zdarzenie, obiekt typu EventEmitter) z
dekoratorem @Output

aby przekazać (wyeksportować) dane musimy użyć metody emit(),
metoda odpowiedz() jest trigerowana przez naciśnięcie przycisku

Komponenty – przesyłanie danych

z komponentu podrzędnego (syn) do nadrzędnego (tata)

1.konfiguracja komponentu podrzędnego (syn)

```
<> syn.component.html U X
komponentyDaneZPodrzednegoDoNadrzednego > src > app > syn > <> syn.component.html > ...
1  <p>syn works!</p>
2  <p>Info od taty:</p>
3  <p>{{infoOdTaty}}</p>
4  <button type = "button" (click) = "odpowiedz()">odpowieź syna</button>
```

aby wyeksportować dane do komponentu nadrzędnego (tata)
potrzebne jest zdarzenie, w tym przypadku kliknięcie przycisku,
zdarzenie click jest zbindowane z metodą odpowiedz()

Komponenty – przesyłanie danych z komponentu podrzędnego (syn) do nadrzędnego (tata)

2.konfiguracja komponentu nadrzędnego (tata)

```
<> tata.component.html U X
komponentyDaneZPodrzednegoDoNadrzednego > src > app > tata > <> tata.component.html > ...
1 <p>tata works!</p>
2 <app-syn [infoOdTaty] = "bierzaceInfo" (odpowiedzSyna) = "onOdpowiedzSyna($event)"></app-syn>
```

```
TS tata.component.ts U X
komponentyDaneZPodrzednegoDoNadrzednego > src > app > tata > TS tata.component.ts > ...
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-tata',
5   templateUrl: './tata.component.html',
6   styleUrls: ['./tata.component.css']
7 })
8 export class TataComponent implements OnInit {
9
10   bierzaceInfo: string = "Najważniejsze co się czuje, słuchaj zawsze głosu serca, hej";
11
12   onOdpowiedzSyna(informacja: string){
13     alert(informacja)
14   }
15
16   constructor() { }
17
18   ngOnInit(): void {
19   }
20
21 }
```

po kliknięciu na przycisk
emitowane jest zdarzenie
odpowiedzSyna
obsługiwane przez metodę
taty (handlera)
onOdpowiedzSyna(\$event),
\$event zawiera
przekazywane dane do taty

definicja metody (handlera)
onOdpowiedzSyna(\$event),

Komponenty sklep

między komponentami
przesyłane są dane – wybrany
towar (np. jabłka)



Sklep

towary

jabłka ▼

wybór towaru

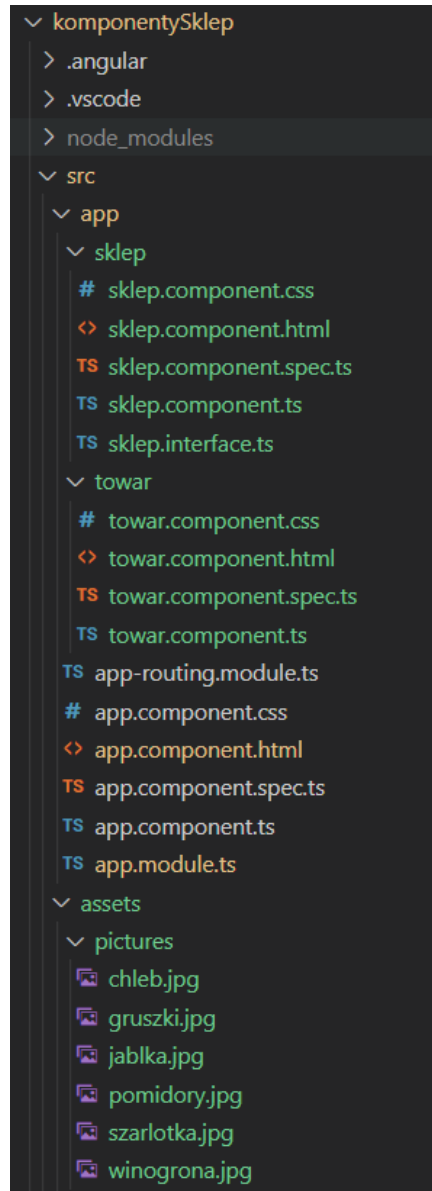
komponent nadrzędny sklep

komponent podrzędny towar

Wybrany towar:

zdjęcie	nazwa	cena
	jabłka	3 zł

Komponenty sklep



komponent nadrzędny sklep

komponent podrzędny towar

zdjęcia towarów w katalogu assets

Sklep

komponent nadrzędny sklep

TS sklep.component.ts U X

komponentySklep > src > app > sklep > TS sklep.component.ts > ...

```
1  import { Component, OnInit } from '@angular/core';
2  // import interfejsu Towar
3  import { Towar } from './sklep.interface';
4
5  @Component({
6    selector: 'app-sklep',
7    templateUrl: './sklep.component.html',
8    styleUrls: ['./sklep.component.css']
9  })
10 export class SklepComponent implements OnInit {
11   // pole towary będące tablicą typu Towar
12   towary: Towar[]=[
13     { nazwa:"chleb", cena: 6, zdjecie: "assets/pictures/chleb.jpg"},
14     { nazwa:"pomidory", cena: 10, zdjecie: "assets/pictures/pomidory.jpg"},
15     { nazwa:"jabłka", cena: 3, zdjecie: "assets/pictures/jablka.jpg"},
16     { nazwa:"gruszki", cena: 4, zdjecie: "assets/pictures/gruszki.jpg"},
17     { nazwa:"winogrona", cena: 15, zdjecie: "assets/pictures/winogrona.jpg"},
18     { nazwa:"szarlotka", cena: 5, zdjecie: "assets/pictures/szarlotka.jpg"}
19   ];
20
21   // pole towar
22   towar: Towar = this.towary[0];
23
24   constructor() { }
25
26   ngOnInit(): void {
27   }
```

TS sklep.interface.ts U X

komponentySklep > src > app > sklep > T

```
1  // interfejs Towar
2  export interface Towar{
3    nazwa: string;
4    cena:number;
5    zdjecie:string;
6  }
```

Sklep

komponent nadrzędny sklep

```
<> sklep.component.html U X
komponentySklep > src > app > sklep > <> sklep.component.html > ...
1  <h1>Sklep</h1>
2  <p>towary</p>
3  <!-- select -->
4  <!-- wiązanie dwukierunkowe pola towar - dzięki temu -->
5  <!-- jeśli klient wybierze nową opcję będzie to odwzorowane w polu towar-->
6  <select [(ngModel)] = "towar">
7    <!-- wypisujemy opcje korzystając z dyrektywy *ngFor, iterujemy po tablicy towary -->
8    <!-- ngValue pobiera iterowany obiekt i przekazuje do pola towar -->
9    <option *ngFor = "let item of towary" [ngValue] = "item">{{item.nazwa}}</option>
10 </select>
11
12 <!-- w komponencie sklep zagnieżdżamy komponent podrzędny towar-->
13 <!-- pole towar komponentu sklep jest związane z polem wybranyTowar komponentu towar-->
14 <!-- dzięki temu gdy klient wybierze towar z listy zostanie on przekazany do pola wybranyTowar-->
15 <app-towar [wybranyTowar] = "towar"></app-towar>
```

Sklep

komponent podrzędny towar

TS towar.component.ts U X

komponentySklep > src > app > towar > TS towar.component.ts > ...

```
1  // importujemy moduł Input
2  import { Component, OnInit, Input } from '@angular/core';
3  import { Towar } from '../sklep/sklep.interface';
4
5  @Component({
6    selector: 'app-towar',
7    templateUrl: './towar.component.html',
8    styleUrls: ['./towar.component.css']
9  })
10 export class TowarComponent implements OnInit {
11
12    // pole wybranyTowar z dekoratorem @Input()
13    @Input() wybranyTowar!: Towar;
14
15    constructor() {}
16
17    ngOnInit(): void {
18
19    }
20 }
```

Sklep

komponent podrzędny towar

```
<> towar.component.html U X
komponentySklep > src > app > towar > <> towar.component.html > ...
1  <br><br><p>Wybrany towar:</p>
2  <!-- tabela wyświetlająca wybrany towar-->
3  <table>
4      <tr>
5          <th>zdjęcie</th>
6          <th>nazwa</th>
7          <th>cena</th>
8      </tr>
9      <tr>
10         <!-- w tabeli wyświetlamy poszczególne pola obiektu wybranyTowar typu Towar-->
11         <td><img src={{wybranyTowar.zdjecie}} width="200px"></td>
12         <td>{{wybranyTowar.nazwa}}</td>
13         <td>{{wybranyTowar.cena}} zł</td>
14     </tr>
15 </table>
```

Sklep

komponent podrzędny towar

```
# towar.component.css U X
komponentySklep > src > app > towar > # towar.compo

1  table {
2      font-family: arial, sans-serif;
3      border-collapse: collapse;
4      width: 50%;
5  }
6
7  td, th {
8      border: 1px solid #dddddd;
9      text-align: left;
10     padding: 8px;
11 }
12
13 tr:nth-child(even) {
14     background-color: #dddddd;
15 }
```

Sklep

komponent główny

```
<> app.component.html M X
komponentySklep > src > app > <> app.component.html > ...
1  <!-- w głównym komponencie zagnieźdźamy komponent sklep-->
2  <app-sklep></app-sklep>
```

```

  assets
  pictures
    chleb.jpg
    gruszki.jpg
    jablka.jpg
    pomidory.jpg
    szarlotka.jpg
    winogrona.jpg
```

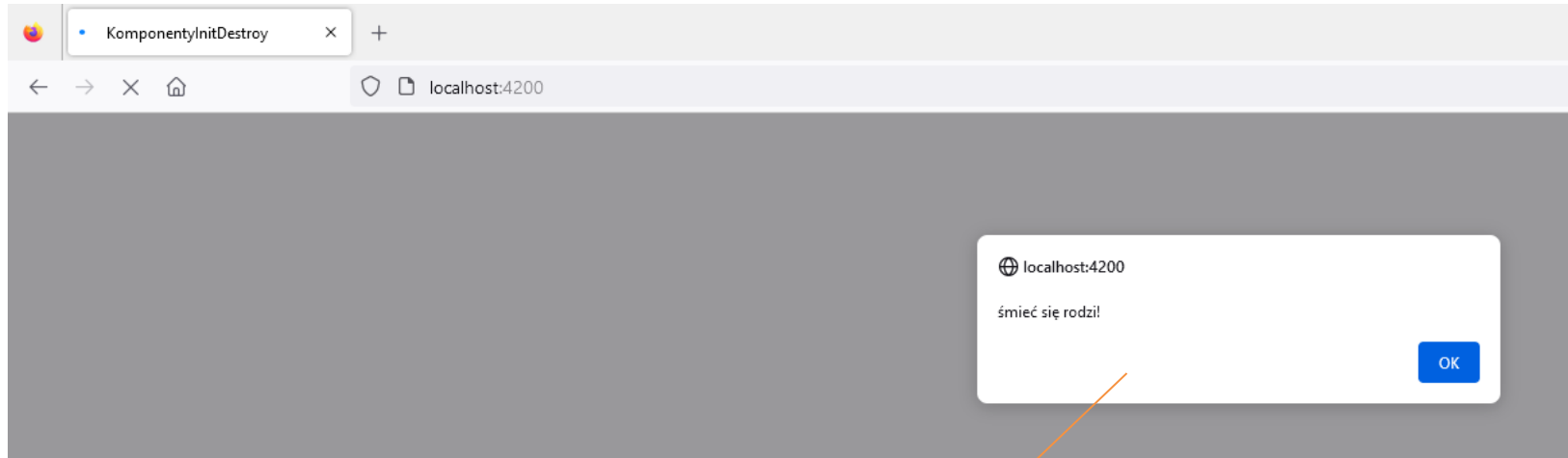
Sklep

komponent główny

```
TS app.module.ts M X
komponentySklep > src > app > TS app.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  // importujemy moduł do obsługi select
4  import { FormsModule } from '@angular/forms';
5
6  import { AppRoutingModule } from './app-routing.module';
7  import { AppComponent } from './app.component';
8  import { SklepComponent } from './sklep/sklep.component';
9  import { TowarComponent } from './towar/towar.component';
10
11  @NgModule({
12    declarations: [
13      AppComponent,
14      SklepComponent,
15      TowarComponent
16    ],
17    imports: [
18      BrowserModule,
19      AppRoutingModule,
20      // dodajemy zaimportowany moduł do importów
21      FormsModule
22    ],
23    providers: [],
24    bootstrap: [AppComponent]
25  })
26  export class AppModule { }
```

Niszczarka śmieci

ngOnInit(), ngOnDestroy()



wynik działania metody ngOnInit()
komponentu smiec

Niszczarka śmieci ngOnInit(), ngOnDestroy()

The image shows a web browser window with the title 'Komponenty/InitDestroy' and the address 'localhost:4200'. The page content includes the heading 'Niszczarka śmieci' (Trash Can) and a checkbox labeled 'odepnij komponent smiec od drzewa dokumentu' (detach trash component from document tree). Below this is a green-bordered box containing the text 'śmieć works!'. Annotations with lines point to these elements: 'komponent główny' (main component) points to the heading, and 'komponent podrzędny smiec' (sub-component trash) points to the 'śmieć works!' box. At the bottom, the Angular DevTools console is visible, showing messages from the 'webpack-dev-server' and a confirmation that 'Angular is running in development mode'.

Niszczarka śmieci

☐ odepnij komponent smiec od drzewa dokumentu

śmieć works!

komponent główny

komponent podrzędny smiec

Inspektor Konsola Debugger Sieć Edytor stylów Wydajność Pamięć Dane Dostępność Wątki Adblock Plus

Filtruj zawartość

- [webpack-dev-server] App updated. Reloading...
- [webpack-dev-server] Disconnected!
- [webpack-dev-server] Trying to reconnect...
- [webpack-dev-server] Server started: Hot Module Replacement disabled, Live Reloading enabled, Progress disabled, Overlay enabled.

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

>>

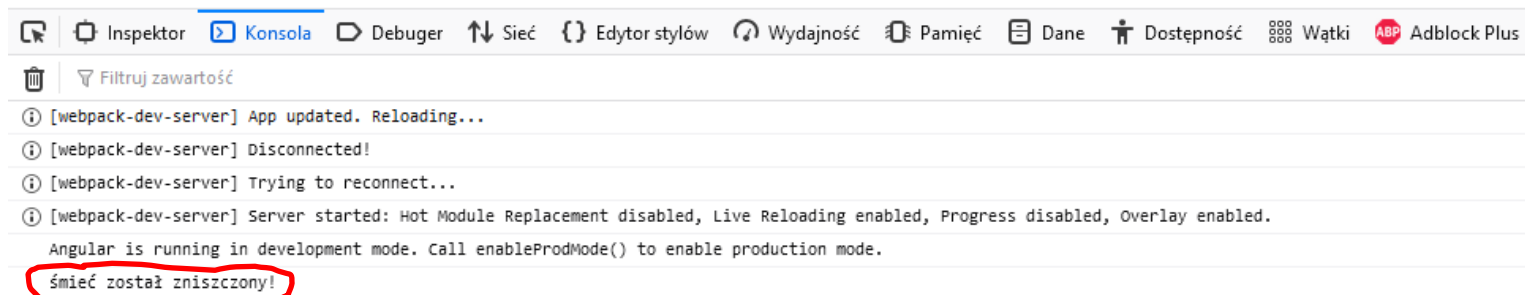
Niszczarka śmieci ngOnInit(), ngOnDestroy()



Niszczarka śmieci

☒ odepnij komponent smiec od drzewa dokumentu

po zaznaczeniu checkboxa
komponent smiec zostaje
odłączony od drzewa
dokumentu



wynik działania metody
ngOnDestroy() komponentu smiec

Niszczarka śmieci komponent smiec

<> smiec.component.html U X

komponentyInitDestroy > src > app > sr

1 <p>śmieć works!</p>

TS smiec.component.ts U X

komponentyInitDestroy > src > app > smiec > TS smiec.component.ts > S

```
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-smiec',
5   templateUrl: './smiec.component.html',
6   styleUrls: ['./smiec.component.css']
7 })
8 export class SmiecComponent implements OnInit {
9
10   constructor() { }
11
12   ngOnInit(): void {
13     alert("śmieć się rodzi!")
14   }
15
16   ngOnDestroy(): void {
17     console.log("śmieć został zniszczony!")
18   }
19 }
```

definicje metod
ngOnInit() oraz
ngOnDestroy()

Niszczarka śmieci komponent główny

```
<> app.component.html M X
komponentyInitDestroy > src > app > <> app.component.html > ...
1  <h1>Niszczarka śmieci</h1>
2  <label><input type="checkbox" [(ngModel)] = zniszcz>odepnij komponent smiec od drzewa dokumentu</label>
3  <app-smiec *ngIf = !zniszcz></app-smiec>
```

/

usuwamy komponent smiec za pomocą
dyrektywy *ngIf, która ma wartość true lub false
w zależności od ustawienia checkboxa

```
TS app.component.ts M X
komponentyInitDestroy > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    zniszcz: boolean = false;
10 }
```

/

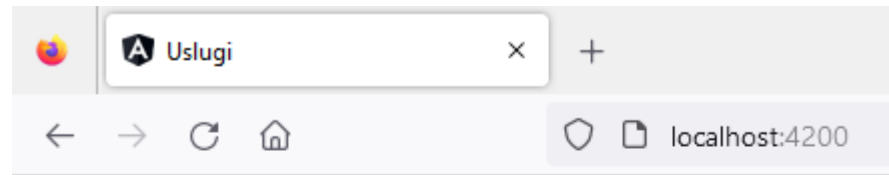
pole zniszcz przyjmuje
wartości true lub false

import FormsModule

```
TS app.module.ts M X
komponentyInitDestroy > src > app > TS app.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { FormsModule } from '@angular/forms';
4
5  import { AppComponent } from './app.component';
6  import { SmiecComponent } from './smiec/smiec.component';
7
8  @NgModule({
9    declarations: [
10     AppComponent,
11     SmiecComponent
12   ],
13   imports: [
14     BrowserModule,
15     FormsModule
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
```

Usługi (Services)

Usługa jest zazwyczaj klasą o wąskim, dobrze zdefiniowanym celu (robi coś konkretnego)

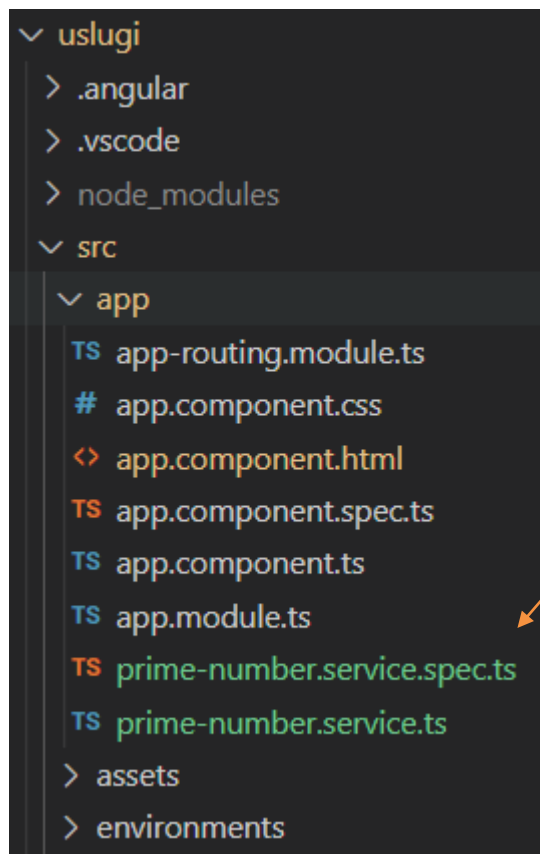


89

po naciśnięciu przycisku usługa (service) generuje liczbę pierwszą

Prime Number Please

Usługi



generate service nazwa usługi

generowanie usługi PrimeNumberService

```
ng g s primeNumber
```

```
CREATE src/app/prime-number.service.spec.ts (383 bytes)  
CREATE src/app/prime-number.service.ts (140 bytes)
```

Usługi

PrimeNumberService

konfiguracja usługi w pliku app.component.ts – dzięki temu będzie można odwoływać się do niej w komponencie głównym

TS app.component.ts M X

uslugi > src > app > TS app.component.ts > ...

```
1  import { Component, Inject } from '@angular/core';
2  import { PrimeNumberService } from '../prime-number.service';
3
4  @Component({
5    selector: 'app-root',
6    templateUrl: './app.component.html',
7    styleUrls: ['./app.component.css'],
8    providers: [PrimeNumberService]
9  })
10 export class AppComponent {
11   // liczba pierwsza
12   primeNumber!: number;
13   // w konstruktorze przypisujemy liczbę pierwszą zwracaną przez
14   // PrimeNumberService do lokalnego pola primeNumber
15   constructor(private primeNumberService: PrimeNumberService){
16     this.primeNumber = primeNumberService.primeNumberPlease();
17   }
18 }
```

dzięki umieszczeniu
serwisu w tabeli
providers będzie
można odwoływać się
do niego w innych
komponentach

Usługi

PrimeNumberService

```
TS prime-number.service.ts U X
uslugi > src > app > TS prime-number.service.ts > ...
1  import { Injectable } from '@angular/core';
2
3  @Injectable({
4    providedIn: 'root'
5  })
6  export class PrimeNumberService {
7
8    constructor() { }
9
10   // generator liczb pierwszych :)
11   primeNumberPlease(): number{
12     // tablica liczb pierwszych
13     let primeNumbers: number[] = [2,3,5,7,11,13,17,19,23,29,
14                                   31,37,41,43,47,53,59,61,67,
15                                   71, 73,79,83,89,97,101];
16     // losujemy indeks
17     let index = Math.floor(Math.random() * 25);
18     // zwracamy liczbę pierwszą z tablicy
19     return primeNumbers[index];
20   }
21 }
```

metoda usługi
PrimeNumberService
o nazwie
primeNumberPlease
generuje liczby
pierwsze

Usługi PrimeNumberService

konfiguracja widoku

<> app.component.html M X

uslugi > src > app > <> app.component.html > ...

```
1
2 <!-- wyświetlamy liczbę pierwszą wygenerowaną przez PrimeNumberService-->
3 <h1>{{primeNumber}}</h1>
4 <!-- przeładujemy stronę -->
5 <button onclick="location.reload();">Prime Number Please</button>
```

Formularz

Ankieta

Jan	imię
Kowalski	nazwisko
jan@kowalski.pl	e-mail

pleć

mężczyzna ☐ kobieta ☐

zainteresowania

sport ☒ film ☒ elektronika ☒ informatyka ☒

szkoła

Uniwersytet Łódzki ▼

krótko o sobie

jestem Jan Kowalski

submit reset

Ankieta obsługiwana
przez Angular

localhost:4200

Twoja ankieta:

imię: Jan
nazwisko: Kowalski
email: jan@kowalski.pl
pleć: kobieta
zainteresownia: sport film elektronika informatyka
szkoła: uniwersytet
o_sobie: jestem Jan Kowalski

OK

po naciśnięciu
przycisku wyświetlają
się zaznaczone opcje
Ankiety w oknie alert

Formularz Ankieta

```
<> app.component.html M X
formularz > src > app > <> app.component.html > ...
1  <h1>Ankieta</h1>
2  <form name="ankieta" #formularz = "ngForm">
3    <input type="text" name="imie" [(ngModel)] = "imie">imie<br>
4    <input type="text" name="nazwisko" [(ngModel)] = "nazwisko">nazwisko<br>
5    <input type="text" name="email" [(ngModel)] = "email">e-mail<br>
6
7    <h3>płeć</h3>
8    mężczyzna<input type="radio" name="plec" value="mężczyzna" [(ngModel)] = "plec">
9    kobieta<input type="radio" name="plec" value="kobieta">
10
11   <h3>zainteresowania</h3>
12
13   sport<input type="checkbox" name="sport" value="1" [(ngModel)] = "sport">
14   film<input type="checkbox" name="film" value="1" [(ngModel)] = "film">
15   elektronika<input type="checkbox" name="elektronika" value="1" [(ngModel)] = "elektronika">
16   informatyka<input type="checkbox" name="informatyka" value="1" [(ngModel)] = "informatyka">
17
18   <h3>szkoła</h3>
19   <select name="szkola" [(ngModel)] = "szkola">
20     <option value="podstawowa">Podstawowa</option>
21     <option value="liceum">Liceum Ogólnokształcące</option>
22     <option value="technikum">Technikum</option>
23     <option value="politechnika">Politechnika Łódzka</option>
24     <option value="uniwersytet">Uniwersytet Łódzki</option>
25     <option value="inna">inna</option>
26   </select>
27
28   <h3>krótco o sobie</h3>
29   <textarea name="o_sobie" rows=5 cols=40 [(ngModel)] = "o_sobie"></textarea>
30
31   <hr>
32   <input type="submit" value="submit" (click)="onSubmit()">
33   <input type="reset" value="reset">
34 </form>
```

każdy element
formularza jest
związany
dwukierunkowo z
polem o nazwie
odzwierciedlającej
funkcję elementu

po kliknięciu na
przycisk wywoływana
jest metoda
onSubmit()

Formularz Ankieta

```
TS app.component.ts M X
formularz > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    imie: string = "";
10   nazwisko: string = "";
11   email: string = "";
12   plec: string = "";
13   sport: string = "";
14   film: string = "";
15   elektronika: string = "";
16   informatyka: string = "";
17   szkola: string = "";
18   o_sobie: string = "";
19 }
```

pola klasy
AppComponent
odpowiadają
elementom ankiety

metoda onSubmit()
klasy AppComponent

```
onSubmit(){
  // zbieramy dane z ankiety
  if(this.plec!=""){
    if(this.plec=="męczyzna"){
      this.plec="mężczyzna";
    }else{
      this.plec="kobieta";
    }
  }
  let zainteresowania = "";
  if (this.sport == "1"){
    zainteresowania += " sport";
  }
  if (this.film == "1"){
    zainteresowania += " film";
  }
  if (this.elektronika == "1"){
    zainteresowania += " elektronika";
  }
  if (this.informatyka == "1"){
    zainteresowania += " informatyka";
  }
  alert("Twoja ankieta:\n" +
    "\nimie: " + this.imie +
    "\nnazwisko: " + this.nazwisko +
    "\nemail: " + this.email +
    "\npleć: " + this.plec +
    "\nzainteresowania: " + zainteresowania +
    "\nszkola: " + this.szkola +
    "\no_sobie: " + this.o_sobie
  );
}
```

Formularz

Ankieta

```
TS app.module.ts M X
zagadka > src > app > TS app.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { FormsModule } from '@angular/forms';
4  import { AppComponent } from './app.component';
5
6  @NgModule({
7    declarations: [
8      AppComponent
9    ],
10   imports: [
11     BrowserModule,
12     FormsModule
13   ],
14   providers: [],
15   bootstrap: [AppComponent]
16 })
17 export class AppModule { }
```

import modułu odpowiedzialnego za obsługę formularza

Formularz

Ankieta z walidacją

FormularzWalidacja x +

localhost:4200

Ankieta

imię
 nazwisko
 e-mail

pleć

mężczyzna ☐ kobieta ☐

zainteresowania

sport ☐ film ☐ elektronika ☐ informatyka ☐

szkoła

▼

krótko o sobie

submit reset

FormularzWalidacja x +

localhost:4200

Ankieta

imię
Imię jest wymagane
 nazwisko
Nazwisko jest wymagane
 e-mail
email jest wymagany
wpisz poprawny email

pleć

mężczyzna ☐ kobieta ☐

zainteresowania

sport ☐ film ☐ elektronika ☐ informatyka ☐

szkoła

▼

krótko o sobie

submit reset

polia wymagane są w
czerwonej ramce

Formularz Ankieta z walidacją

zmiany związane z
walidacją pola imie
(pole jest wymagane)

zmienna szablonowa
name przypisana do
ngModel

```
<> app.component.html M X
formularzWalidacja > src > app > <> app.component.html > ...
1   <h1>Ankieta</h1>
2   <form id = "ankietaForm" name="ankieta" #formularz = "ngForm">
3   |   <input type="text" name="imie" [(ngModel)] = "imie" required #name="ngModel">imie<br>
4   |   <div *ngIf="name.invalid && name.touched">Imię jest wymagane</div>
```

dyrektywa *ngIf z wykorzystaniem
zmiennej szablonowej name

dodatkowa informacja ukazuje się,
gdy pole imię nie jest wypełnione
i zostało kliknięte albo wybrane
klawiszem TAB oraz opuszczone

Formularz

Ankieta z walidacją

```
<> app.component.html M X
formularzWalidacja > src > app > <> app.component.html > ...
1  <h1>Ankieta</h1>
2  <form id = "ankietaForm" name="ankieta" #formularz = "ngForm">
3    <input type="text" name="imie" [(ngModel)] = "imie" required #name="ngModel">imie<br>
4    <div *ngIf="name.invalid && name.touched">Imię jest wymagane</div>
5    <input type="text" name="nazwisko" [(ngModel)] = "nazwisko" required #surname="ngModel">nazwisko<br>
6    <div *ngIf="surname.invalid && surname.touched">Nazwisko jest wymagane</div>
7    <input type="text" name="email" [(ngModel)] = "email" required pattern=".+@.+" #myEmail="ngModel">e-mail<br>
8    <div *ngIf="myEmail.invalid && myEmail.touched">email jest wymagany</div>
9    <div *ngIf="myEmail.touched && myEmail.errors?.['pattern']">wpisz poprawny email</div>
10   <h3>płeć</h3>
11   mężczyzna<input type="radio" name="plec" value="męczyzna" [(ngModel)] = "plec">
12   kobieta<input type="radio" name="plec" value="kobieta">
13
14   <h3>zainteresowania</h3>
15
16   sport<input type="checkbox" name="sport" value="1" [(ngModel)] = "sport">
17   film<input type="checkbox" name="film" value="1" [(ngModel)] = "film">
18   elektronika<input type="checkbox" name="elektronika" value="1" [(ngModel)] = "elektronika">
19   informatyka<input type="checkbox" name="informatyka" value="1" [(ngModel)] = "informatyka">
20
21   <h3>szkoła</h3>
22   <select name="szkola" [(ngModel)] = "szkola" required>
23     <option value="podstawowa">Podstawowa</option>
24     <option value="liceum">Liceum Ogólnokształcące</option>
25     <option value="technikum">Technikum</option>
26     <option value="politechnika">Politechnika Łódzka</option>
27     <option value="uniwersytet">Uniwersytet Łódzki</option>
28     <option value="inna">inna</option>
29   </select>
30
31   <h3>krótko o sobie</h3>
32   <textarea name="o_sobie" rows=5 cols=40 [(ngModel)] = "o_sobie"></textarea>
33
34   <hr>
35   <input type="submit" value="submit" [disabled]="!formularz.form.valid" (click)="onSubmit()">
36   <input type="reset" value="reset">
37 </form>
```

szablon ze zmianami
walidacyjnymi

Formularz Ankieta z walidacją

```
TS app.component.ts X
formularzWalidacja > src > app > TS app.component.ts ...
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   imie: string = "";
10  nazwisko: string = "";
11  email: string = "";
12  plec: string = "";
13  sport: string = "";
14  film: string = "";
15  elektronika: string = "";
16  informatyka: string = "";
17  szkola: string = "";
18  o_sobie: string = "";
19
20  onSubmit(){
21    // zbieramy dane z ankiety
22    if(this.plec!=""){
23      if(this.plec=="meczczyszna"){
24        this.plec="męczyzna";
25      }else{
26        this.plec="kobieta";
27      }
28    }
29    let zainteresowania = "";
30    if (this.sport == "1"){
31      zainteresowania += " sport";
32    }
33    if (this.film == "1"){
34      zainteresowania += " film";
35    }
36    if (this.elektronika == "1"){
37      zainteresowania += " elektronika";
38    }
39    if (this.informatyka == "1"){
40      zainteresowania += " informatyka";
41    }
42    alert("Twoja ankieta:\n" +
43      "\nimie: " + this.imie +
44      "\nnazwisko: " + this.nazwisko +
45      "\nemail: " + this.email +
46      "\nplec: " + this.plec +
47      "\nzainteresownia: " + zainteresowania +
48      "\nszkola: " + this.szkola +
49      "\no_sobie: " + this.o_sobie
50    );
51  }
52 }
53
```

```
TS app.module.ts M X
formularzWalidacja > src > app > TS app.module.ts ...
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { FormsModule } from '@angular/forms';
4
5 import { AppComponent } from './app.component';
6
7 @NgModule({
8   declarations: [
9     AppComponent
10  ],
11  imports: [
12    BrowserModule,
13    FormsModule
14  ],
15  providers: [],
16  bootstrap: [AppComponent]
17 })
18 export class AppModule { }
```

```
# app.component.css M X
formularzWalidacja > src > app > # app.compo
1 .ng-invalid{
2   border: 1px solid red;
3   width:30%;
4 }
5 #ankietaForm{
6   border:0px;
7 }
```

pliki:
app.component.ts
oraz
app.module.ts
są takie same jak w
formularzu bez
walidacji

dzięki stylom
wymagane pola mają
czerwoną obramówkę

Formularz

Ankieta z walidacją

stan weryfikacji
danych opisują klasy
w tabeli



klasa	Opis
valid	dobrze wypełniony
invalid	źle wypełniony
submitted	zatwierdzony
touched	pole zostało wybrane kliknięciem lub tabem
untouched	pole nie wybrane
dirty	użytkownik coś już wpisał (mogło to zostać wykasowane)
pristine	użytkownik nic nie wpisał

deployment

aby projekt strony wykonany w Angularze przygotować do umieszczenia na zewnętrznym serwerze (np. Apache w XAMPP) należy w folderze projektu wykonać polecenie: **ng build --base-href ./**

w wyniku wykonania polecenia w folderze projektu wygeneruje się folder `dist/nazwa_projektu`

folder `nazwa_projektu` umieszczamy w folderze `htdocs` serwera Apache (XAMPP)

```
ng build --base-href ./
```

Nazwa

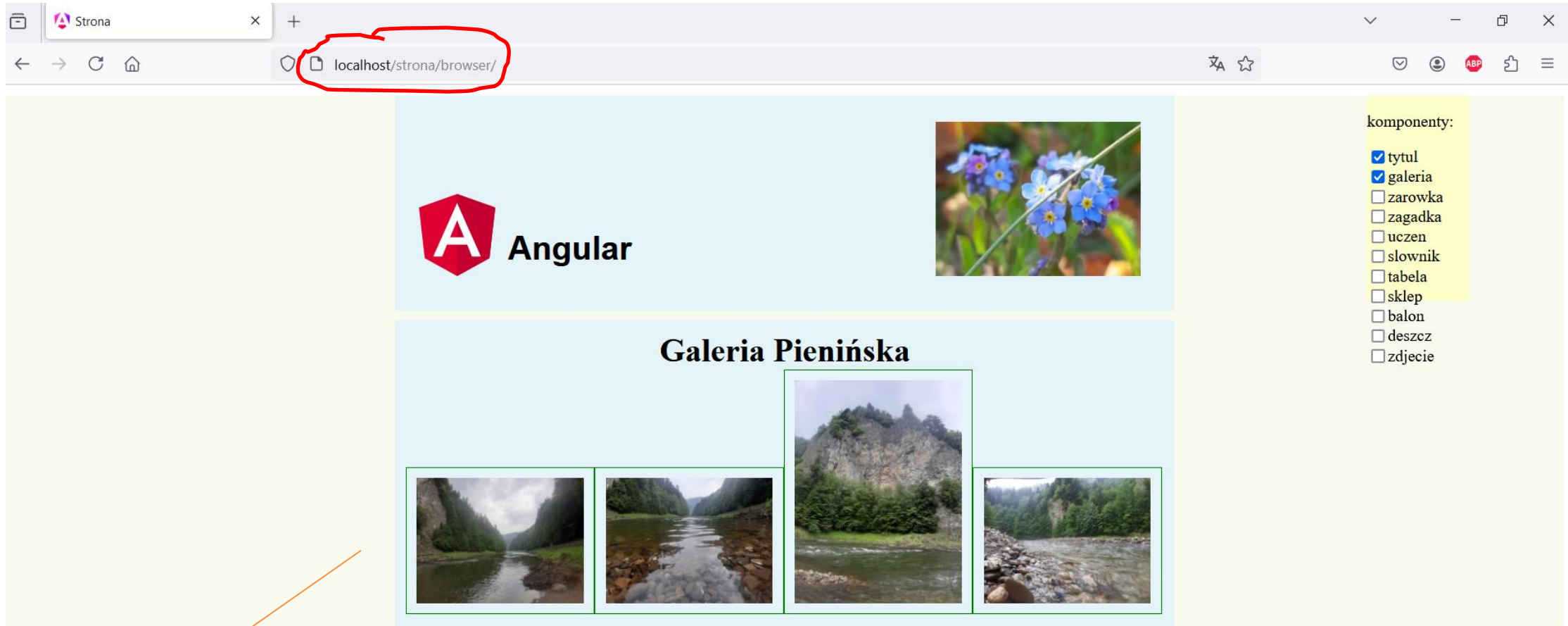
.angular
.git
.vscode
dist
node_modules
src
.editorconfig
.gitignore
angular.json

nazwa_projektu w przykładzie to strona

strona

Ten komputer > OS (C:) > xampp > htdocs

deployment



projekt „strona” hostowany za
pomocą programu XAMPP