

Podstawy pracy z Angulariem

Dokumentacja Angulara: <https://angular.io/docs>

Zadanie 1.

1. Zainstaluj na swoim komputerze *node.js* oraz *NPM*.

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

<https://www.npmjs.com/get-npm>

2. Upewnij się, że instalacja przebiegła pomyślnie.

`node -v`

`npm -v`

3. Następnie poprzez *NPM* zainstaluj *Angular CLI*.

<https://github.com/angular/angular-cli>

`npm install -g @angular/cli`

lub:

`npm install -g angular-cli`

4. Utwórz projekt Angulara o nazwie *Test*.

`ng new Test`

`cd Test`

5. Ściągnij zależności, które są w pliku *package.json* poprzez *NPM*.

`npm install`

6. Jeśli w konsoli jest informacja o brakujących zależnościach dodaj je w pliku

package.json i ponownie ściągnij zależności przez *NPM*.

7. Przeanalizuj budowę projektu.

8. Uruchom projekt.

`ng serve`

9. Przejdź pod adres <http://localhost:4200>.

Zadanie 2.

1. Stwórz nowy komponent o nazwie *Testing*.

```
ng g c Testing
```

2. Przeanalizuj budowę powstałego komponentu.
3. Z pliku **testing.component.ts** skopiuj wartość selektora z dekoratora **@Component**.
4. Na końcu pliku **app.component.html** dodaj element:

```
<app-testing></app-testing>
```

5. Uruchom projekt i zobacz zmiany w przeglądarce.
6. W kodzie **testing.component.ts** dodaj zmienną o nazwie **test** typu **String** „Ala ma kota”.

7. W kodzie **testing.component.html** dodaj następującą linię:

```
<p>{{ test + ' a kot ma Alę.' }}</p>
```

8. Zobacz zmiany w przeglądarce.
9. W kodzie **testing.component.ts** dodaj zmienną o nazwie **isVisible** typu **Boolean** z wartością **true**.

10. W kodzie **testing.component.ts** dodaj zmienną o nazwie **arr** typu **Array<Number>** bez wartości. W metodzie **constructor()** ustaw jej wartość na **[]**.

11. W kodzie **testing.component.html** dodaj następujący kod:

```
<p *ngIf="isVisible">Visible text.</p>
<p *ngIf="!isVisible">Invisible text.</p>
<p *ngIf="!arr.length">The length of array arr equals
zero.</p>
<p *ngIf="arr.length == 0">The length of array arr equals
zero.</p>
```

12. Zobacz zmiany w przeglądarce.
13. Tablicę **arr** wypełnij danymi (np. każdy element to kolejna liczba naturalna).
14. W kodzie **testing.component.html** dodaj następujący kod:

```
<div>
  <p *ngFor="let element of arr">{{ element }}</p>
</div>
```

15. Zobacz zmiany w przeglądarce.

16. W kodzie **testing.component.ts** dodaj metodę `clickMe()`, która będzie wypisywała w konsoli kliknięty element:

```
clickMe(event: Event): void {  
    console.log(event.target);  
}
```

17. Kod z punktu 14. Zmodyfikuj dodając do każdego elementu `<p>` po dyrektywie `*ngFor`:

```
(click)="clickMe($event) "
```

18. Przetestuj działanie w przeglądarce.

19. W kodzie **testing.component.ts** dodaj metodę `show()`:

```
show(element: any): void {  
    console.log(element);  
}
```

20. W kodzie **testing.component.html** dodaj następujący kod:

```
<h1>*ngFor directive testing with click event:</h1>  
<p *ngFor="let el of arr" (click)="show(el)"><b>{{ el  
  }}</b></p>
```

21. Przetestuj działanie w przeglądarce.

Krótki kurs TypeScript: <https://typeofweb.com/kurs/typescript/>

Zadanie 3.

Przydatne informacje, które pomogą w rozwiązaniu zadania znajdują się na stronie

<https://angular.io/guide/forms>.

Szczególnie warto zwrócić uwagę na dwukierunkowe bindowanie danych (konstrukcja

`[(ngModel)]`). Więcej o `ngModel` na stronie: <https://angular.io/api/forms/NgModel>.

1. Stwórz projekt o nazwie **ToDoList**.
2. W głównym komponencie powinien znaleźć się nagłówek **ToDoList**.
3. Stwórz komponent **Tasks**.
4. W komponencie **Tasks** powinna znaleźć się lista zadań, gdzie przy każdym zadaniu będzie przycisk do usuwania **Delete**.
5. W tym komponencie powinien także znaleźć się przycisk **Add task** oraz pole tekstowe z placeholderem **Add task...**

6. Zadania będą przechowywane w tablicy **tasks** w komponencie **Tasks**.
7. Tablicę należy wypełnić na sztywno danymi, gdzie jedno zadanie to zwykły string.
8. Kiedy użytkownik kliknie na przycisk **Delete** to zadanie jest usuwane z tablicy a widok jest prerenderowany w przeglądarce.
9. Po wciśnięciu przycisku **Add task** wartość z pola tekstowego jest zapisywana na końcu tablicy a widok jest prerenderowany w przeglądarce. Wartość pola tekstowego jest czyszczona.
10. Jeśli użytkownik spróbuje dodać puste zadanie powinien otrzymać alert **You cannot add an empty task**.
11. Jeśli użytkownik spróbuje dodać zadanie, które już jest na liście także powinien otrzymać alert **You cannot add a task which already exists**.
12. Jeśli nie ma żadnych zadań w tablicy powinien zostać wyświetlony napis **No tasks found**.

Rozwiązanie tego zadania należy umieścić na swoim repozytorium do zajęć na GitHubie w katalogu „ANGULAR-CW01” do następnych zajęć.

Proszę przed wysłaniem się upewnić, że w pliku .gitignore znajduje się katalog node_modules!