Samsung R&D Institute Poland

# Angular Tech Guide

Poszerzaj i utrwalaj znajomość zagadnień technicznych, korzystając z praktycznych wskazówek naszych ekspertów.

Samsung Tech Guides

# Drogi kandydacie!
# Droga kandydatko!

Aby opanować podstawy Angulara najlepiej zacząć od zapoznania się z podstawowymi elementami, z których tworzymy aplikację single page.

## Komponenty (Components)

Główny element, z którego korzystamy to komponent. Składa się on z pliku szablonu, który zawiera HTML, pliku styli i dwóch plików typescriptowych: z testami oraz głównym zawierającym klasę komponentu.

**Sugerujemy zapoznać się ze strukturą komponentu:**
https://angular.io/guide/component-overview
https://angular.dev/guide/components

## Szablony (Templates)

W szablonie komponentu możemy stosować dyrektywy, potoki, oraz inne komponenty, które zawierają jakąś logikę. Przykładem może być lista użytkowników, gdzie wyświetleniem każdego użytkownika zajmuje się inny komponent.

**Sugerujemy zapoznać się z całym działem Templates:**
https://angular.io/guide/template-overview
https://angular.dev/guide/templates

## Dyrektywy

Dyrektywy służą do dodawania funkcjonalności do elementów HTML-a. W Angularze mamy szereg wbudowanych dyrektyw, możemy wyróżnić grupy:

- dyrektywy strukturalne *ngIf, *ngFor, *ngSwitch
- dyrektywy atrybutów, takie jak ngClass, ngStyle, ngModel

**Sugerujemy zapoznać się z dokumentacją:**
https://angular.io/guide/built-in-directives
https://angular.dev/guide/directives

## Potoki (Pipes)

Potoki pozwalają modyfikować sposób prezentacji wartości zmiennych umieszczonych w szablonie komponentu. Wbudowane to np. json, date, async.

**Sugerujemy zapoznać się z dokumentacją:**
https://angular.io/guide/pipes-overview
https://angular.dev/guide/pipes

## Moduły (Modules)

Komponenty grupujemy w moduły. To jak je grupujemy, zależy od architektury aplikacji oraz logiki biznesowej. Moduły możemy reużywać, ale dodajemy je tylko w innych modułach.

**W dokumentacji:**
https://angular.io/guide/architecture-modules
https://angular.dev/guide/ngmodules

## Komponenty standalone

Od Angular 14 możliwe jest tworzenie aplikacji bez użycia NgModule.

**Więcej na:**
https://angular.io/guide/standalone-components

## Serwisy (Services)

Do klasy komponentu możemy dodać serwisy, czyli osobne klasy zawierające logikę biznesową, pobierające dane z API itp. Serwisy są również elementami Angulara. Do klas komponentów są one dodawane przez dependecy injection, (to ważny termin: https://angular.io/guide/dependency-injection-overview) w konstruktorze klasy.

## Przekazywanie danych między komponentami

Aby możliwe był przepływ informacji między komponentami, musimy zapoznać się ze sposobami przekazywania danych z i do komponentów. Są to zmienne z dekoratorami @Inputt() i @Output(). Możemy również przekazywać dane przez klasy serwisów.

**W dokumentacji:**
https://angular.io/guide/component-interaction
oraz https://angular.io/guide/inputs-outputs

## Cykl życia komponentu

Każdy komponent ma swój cykl życia. Zaczyna się od konstruktora, a później wywołują się kolejne metody w ściśle określonej kolejności. Proponujemy zajrzeć do dokumentacji: https://angular.io/guide/lifecycle-hooks

## Projekcja treści

Często istnieje potrzeba utworzenia komponentu, który działałby jako kontener dla różnych rodzajów treści. Np. można mieć komponent karty, który akceptuje treść dostarczoną przez inny komponent. W tym celu używa się projekcji treści.

**Więcej na:**
https://angular.io/guide/content-projection
https://angular.dev/guide/components/content-projection

## Dostęp do elementów szablonu

Aby uzyskać dostęp do elementów szablonu, w klasie komponentu używa się dekoratorów @ViewChild, @ViewChildren, @ContentChild i @ContentChildren.

**Więcej na:**
https://angular.io/api/core/ViewChild
https://angular.io/api/core/ViewChildren
https://angular.io/api/core/ContentChild
https://angular.io/api/core/ContentChildren
https://angular.dev/guide/components/queries

## HttpClient

Aby pobrać dane z REST-owego API, używamy dostarczanej przez Angulara klasy HttpClient. Znajduje się ona w module HttpClientModule, który również musimy dodać do modułu, gdzie będziemy chcieli odwoływać się do jakiegoś zasobu API.

**Tak jak w REST-cie mamy metody get, post, put, delete, w dokumentacji:**
https://angular.io/guide/http-setup-server-communication
https://angular.io/guide/http-server-communication
https://angular.dev/guide/http

## Formularze

Do wprowadzania danych od użytkownika mamy formularze. W Angularze mamy dwa podejścia template driven oraz reaktywne, do większych formularzy stosujemy raczej reaktywne podejście.

**W dokumentacji:**
https://angular.io/guide/forms
https://angular.io/guide/reactive-forms
https://angular.dev/guide/forms

## Routing

Aby było możliwe przechodzenie z podstrony na podstronę, potrzebujemy routingu. Jest to specjalny moduł dostarczany przez Angulara, który zawiera również dyrektywy obsługujące dodawanie odnośników do HTML.

**Sugerujemy zapoznać się z działem:**
https://angular.io/guide/routing-overview

Do zarządzania dostępem do podstron służą guardy.
Są to klasy (w nowszych wersjach Angulara - funkcje), które zawierają logikę mówiącą, czy dana podstrona może być załadowana lub wyświetlona danemu użytkownikowi.

**Np tutaj w dokumentacji:**
https://angular.io/api/router/CanActivateFn

## Programowanie reaktywne, RxJS

- Czym jest programowanie reaktywne oraz biblioteka RxJS?
- Różnice między leniwym a zachłannym przetwarzaniem.

https://angular.io/guide/rx-library

## Klasy Observable i Promise

- Czym jest Observable, Observer?
- Znajomość różnic między Observable, a Promise.
- Przykłady Agularowych mechanizmów, które zwracają strumienie.

https://angular.io/guide/comparing-observables
https://angular.io/guide/observables

## Klasy Observer i Subscription

- Czym jest Subscription i Observer, jaka jest jego struktura?
- Sposoby na odsubskrybowanie się od Observable.
- Kiedy odsubskrybować, a kiedy nie jest to konieczne?
- Zagnieżdżone subskrypcje - czemu i jak uniknąć?

https://blog.bitsrc.io/6-ways-to-unsubscribe-from-observables-in-angular-ab912819a78f

## Cold vs Hot Observables

- Różnice między Hot i Cold Observables.
- Czym jest strumień typu Unicast I Multicast?

**Przykłady takich strumieni:**
https://luukgruijs.medium.com/understanding-hot-vs-cold-observables-62d04cf92e03
https://www.willtaylor.blog/rxjs-observables-hot-cold-explained/

## Klasa Subject

- Czym jest Subject i czym sie różni od Observable?
- Znajomość różnych typów Subjectów - BehaviorSubject, ReplaySubject, AsyncSubject.

https://www.learnrxjs.io/learn-rxjs/subjects
https://dev.to/this-is-learning/rxjs-subjects-4m12

## Operatory

Znajomość podstawowych operatorów:

- filter, first, take, takeUntil, debounce, distinctUntilChanged
- merge, forkJoin, startWith, combineLatest, concat
- map, switchMap, mergeMap, concatMap
- catch/catchError
- tap, finalize

https://www.learnrxjs.io/learn-rxjs/operators

# Podstawowe znaczniki HTML

Podstawowe znaczniki HTML

- <p>
- <h1>, <h2>, <h3>...
- <a>
- 
- <ol>
- <ul>
- <img>
- 
- <svg>
- <input>
- <button>
- <form>

**Więcej na:**
https://www.codebrainer.com/blog/top-10-html-tags

# Elementy blokowe i Inline

Element blokowy zawsze zajmuje całą dostępną szerokość (rozciąga się w lewo i prawo tak daleko, jak to możliwe).
Przykładowe tagi:  <p> <form> <main>

Elementy Inline nigdy nie rozpoczynają się od nowego wiersza i zajmują tylko szerokość zgodną z rozmiarem znaczników ograniczonych w elemencie HTML.
Przykładowe tagi:  <a> <img> <i> <label>

**Więcej na:**
https://www.w3schools.com/html/html_blocks.asp
https://www.scaler.com/topics/html/inline-and-block-elements/

# Elementy semantyczne HTML

W HTML istnieje kilka elementów semantycznych, których można użyć do zdefiniowania różnych części strony internetowej:

- <header>
- <footer>
- <nav>
- <main>
- <article>
- <section>
- <aside>

**Więcej na:**
https://www.w3schools.com/html/html5_semantic_elements.asp

# Import czcionek

Twoje „własne" czcionki są zdefiniowane w regule CSS @font-face. Najpierw zdefiniuj nazwę czcionki, a następnie wskaż plik czcionki. Aby użyć czcionki dla elementu HTML, odwołaj się do nazwy czcionki we właściwości rodziny czcionek.

```
@font-face {
    font-family: myFirstFont;
    src: url(sansation_light.woff);
}
div {
    font-family: myFirstFont;
}
```

# Import skryptów

Element <script> zawiera instrukcje skryptowe,

```
<script>
    document.getElementById("demo").
    innerHTML = "Hello JavaScript!";
</script>
```

**Więcej na:**
https://www.w3schools.com/tags/tag_script.asp

lub wskazuje na zewnętrzny plik skryptu poprzez atrybut src.

```
<script src="myscripts.js"></script>
```

**Więcej na:**
https://www.w3schools.com/tags/att_script_src.asp

# Import styli

Istnieją trzy sposoby wstawiania arkusza stylów:

- Style zewnętrzne są definiowane w elemencie <link>, wewnątrz sekcji <head> strony HTML

```
<head>
    <link rel="stylesheet" href="mystyle.css">
</head>
```

- Style wewnętrzne są definiowane w elemencie <style>, w sekcji <head> strony HTML

```
<head>
    <style>
    body {
      background-color: linen;
    }

    h1 {
      color: maroon;
      margin-left: 40px;
    }
    </style>
    </head>
```

- Style inline są zdefiniowane w atrybucie „style" odpowiedniego elementu

```
<h1 style="color:blue;text-align:center;">
    This is a heading</h1>
<p style="color:red;">
    This is a paragraph.</p>
```

**Więcej na:**

https://www.w3schools.com/css/css_howto.asp

## Znaczniki, które mogą być zawarte w kontenerze <head>

Element <head> jest kontenerem na metadane. Metadane to dane o dokumencie HTML. Metadane nie są wyświetlane.

Wewnątrz elementu <head> mogą znajdować się następujące elementy:

- <title>
- <style>
- <base>
- <link>
- <meta>
- <script>
- <noscript>

**Więcej na:**
https://www.w3schools.com/tags/tag_head.asp

## Dlaczego powinno się używać semantycznego kodu HTML

- **Dostępność** - pisanie semantycznego kodu HTML ułatwia zrozumienie kodu, dzięki czemu kod źródłowy jest bardziej czytelny dla innych programistów. Czytniki ekranu i przeglądarki mogą lepiej interpretować semantyczny kod HTML, co czyni go bardziej dostępnym np. dla osób z wadą wzroku.

- **SEO (Search Engine Optimization)** - wpływa to na SEO Twojej strony, dając jej lepszy ranking w wyszukiwarkach, odpowiednio ważących najważniejsze treści.

**Więcej na:**
https://www.semrush.com/blog/semantic-html5-guide/
https://www.thisdot.co/blog/semantic-html-why-it-matters-and-top-tips-on-how-to-apply-it/

# CSS & SCSS

**CSS (Cascading Style Sheet)** to język arkuszy stylów używany do stylowania strony internetowej napisanej w HTML. CSS posiada kaskadową hierarchię, w której style zdefiniowane na wyższym poziomie wpływają również na elementy niższego poziomu.
**Rozszerzenie: .css**

**SCSS (Syntactically Awesome Style Sheet)** to preprocesorowy język skryptowy będący nadzbiorem CSS. Składnia SCSS jest bardzo podobna do CSS, ale pozwala na stosowanie statycznych zmiennych, zagnieżdżanie, mixiny i inne.
**Rozszerzenie: .scss**

**Więcej na:**
https://www.geeksforgeeks.org/what-is-the-difference-between-css-and-scss/

# CSS/SCSS zmienne

Zmienne zawierają określone wartości, które można ponownie wykorzystać w kodzie w wielu miejscach. Zapewniają semantyczne identyfikatory i pozwalają na łatwiejsze wprowadzanie zmian.

| | CSS | SCSS |
|---|---|---|
| **Deklaracja** | --main: #aa013b; | $main: #aa013b; |
| **Użycie** | color: var(--main); | color: $main; |
| **Kompilacja** | Nie Wymaga wsparcia przeglądarki | Tak Kompilowany do CSS |
| **Kaskadowe** | Tak | Nie |
| **Dostęp z JS** | Tak | Nie |

**Więcej na:**
https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties
https://sass-lang.com/documentation/variables

# Selektory CSS

Selektory to wzorce używane do zaznaczania elementu, któremu chcesz nadać styl.

| | |
|---|---|
| Uniwersalny | ***** |
| Typ | **nazwaElementu** |
| Klasa | **.nazwaKlasy** |
| ID | **#nazwaId** |
| Atrybut | **[attr]** |
| Selektory grupujące | **A, B** |
| Kombinacje | **A B  A > B  A ~ B  A + B** |

**Więcej na:**

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

https://www.w3schools.com/cssref/css_selectors.php

# ID vs klasa

| **ID** | **Class** |
|---|---|
| Wybiera **grupę** elementów | Wybiera **pojedynczy** element |
| Musi być **unikalne** | **Kilka** elementów może mieć tą samą klasę |
| Bardziej specyficzny | Mniej specyficzny |

## Pseudo-elementy

- ::after
- ::before

**Więcej na:**
https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements

## Pseudo-klasy

Akcja użytkownika

- :hover
- :active
- :focus
- :focus-visible

Input

- :enabled, :disabled
- :valid, :invalid

Linki

- :visited

Struktura dokumentu

- :root
- :empty
- :first-child, :last-child, :nth-child

**Więcej na:**
https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes
https://bitsofco.de/when-is-focus-visible-visible/

## Specyficzność i waga selektorów

Algorytm specyficzności oblicza wagę selektora CSS, aby określić, która reguła z konkurencyjnych deklaracji CSS zostanie zastosowana do elementu. Kategorie wagowe selektora w kolejności malejącej specyficzności:

| | |
|---|---|
| ID | **1-0-0** |
| Klasa | **0-1-0** |
| Atrybut | **0-1-0** |
| Pseudo-klasa | **0-1-0** |
| Typ | **0-0-1** |
| Pseudo-element | **0-0-1** |
| Uniwersalny | **0-0-0** |

**Więcej na:**
https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity

## Jednostki

- **Bezwzględne:** px, pt
- **Względne:** %, em, rem

**Więcej na:**
https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units

## Kolory

| | |
|---|---|
| Nazwane kolory | **red blue** |
| RGB | **#RGB #RRGGBB rgb(R G B)** |
| RGBA | **#RGBA #RRGGBBAA rgb(R G B[ / A])** |
| HSL | **hsl(H S L [ / A])** |

**Więcej na:**
https://developer.mozilla.org/en-US/docs/Web/CSS/color_value

## Pozycja

- **Static** - element jest umieszczony zgodnie z normalnym przepływem dokumentu
- **Relative** - ustawienie właściwości top, right, bottom i left spowoduje odsunięcie jej od normalnej pozycji
- **Fixed** - umieszczony względem widocznego obszaru, pozostaje w tym samym miejscu, nawet jeśli strona jest przewijana
- **Absolute** - umieszczony względem najbliższego przodka, który ma wartość position inną od static (patrz https://developer.mozilla.org/en-US/docs/Web/CSS/Containing_block)

**Więcej na:**
https://developer.mozilla.org/en-US/docs/Web/CSS/position

## z-index

Określa przykrywanie się elementów. Element o wyższym indeksie znajduje się zawsze przed elementem o niższym indeksie jeśli mają wspólny containing block.

**Uwaga:** Z-index działa tylko na elementach pozycjonowanych i elastycznych.

**Więcej na:**
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_positioned_layout/Understanding_z-index

# Wyśrodkowanie elementu

### Transform i translate

```css
.container {
    position: relative;
}

.child {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}
```

### Flexbox

```css
.container {
    display: flex;
    justify-content: center;
    align-items: center;
}
```

# Media queries

Media queries umożliwiają stosowanie stylów CSS w zależności od typu urządzenia lub innych cech, takich jak rozdzielczość ekranu lub szerokość okna przeglądarki.

- min-width
- max-width
- podejście Mobile-first

**Więcej na:**
https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries

## calc

Funkcja CSS calc() umożliwia wykonywanie obliczeń podczas określania wartości właściwości CSS. Dostępne operatory:

- +
- -
- *
- /

**Więcej na:**
https://developer.mozilla.org/en-US/docs/Web/CSS/calc

## REST

REST jest stylem architektonicznym służącym do projektowania interfejsów API za pomocą protokołu HTTP. Najważniejsze to znać typy operacji CRUD i wiedzieć, do czego służą, chodzi tu o GET, PUT, POST, PATCH I DELETE.

**Więcej na:**
https://www.geeksforgeeks.org/rest-api-introduction/

## Developers Tools

Developer Tools, to zestaw narzędzi wbudowanych w praktycznie każdą przeglądarkę. Dzięki nim możemy obejrzeć kod źródłowy strony, style powiązane z każdym elementem na stronie, listę zapytań http, wraz z informacjami o każdym zapytaniu itd.

**Więcej na:**
https://developer.chrome.com/docs/devtools/overview/

## Angular-DevTools

Angular DevTools, to rozszerzenie przeglądarki, które musimy sobie zainstalować. Dzięki temu rozszerzeniu możemy podejrzeć, jakie komponenty I dyrektywy są używane na obecnie tworzonej stronie oraz jakie wartości mają zmienne. Mamy również opcję profilowania.

**Zachęcamy do odwiedzenia strony:**
https://angular.io/guide/devtools
https://angular.dev/tools/devtools

## Czym jest Git™

Git to system kontroli wersji używany do śledzenia zmian w plikach komputerowych. Jest powszechnie używany do zarządzania kodem źródłowym w tworzeniu oprogramowania, umożliwiając wielu programistom współpracę nad nieliniowym rozwojem.

**Podstawy komendy do pracy z Git-em**

- git log
- git clone
- git checkout
- git status
- git commit
- git pull
- git push
- git branch
- git merge
- git rebase

https://blogprogramisty.net/komendy-git-a-ktore-nalezy-znac/

## Pull request

Pull request pozwalają informować innych o zmianach, które wypchnięto do gałęzi w repozytorium w systemie kontroli wersji. Po otwarciu go można przedyskutować i przejrzeć potencjalne zmiany ze współpracownikami oraz dodać kolejne zatwierdzenia, zanim zmiany zostaną scalone z gałęzią podstawową.

## Code review

Code review to ocena kodu mająca na celu identyfikację błędów, poprawę jakości kodu i pomoc programistom w poznaniu kodu źródłowego.

Samsung R&D Institute Poland

Samsung Tech Guides