

Zaczynamy!

Mikołaj Dyczkowski
Frontend 4 Beginners

Gdzie szukać informacji

- MDN
- CSS Tricks
- Stackoverflow
- W trakcie kursu - github issues i pytania na zajęciach :)

Struktura strony/aplikacji webowej

1. Treść
2. Wygląd
3. Zachowanie

1. HTML
2. CSS
3. JS

HTML

Struktura treści na stronie

```
<!DOCTYPE html>
<html lang="pl">
  <head>...</head>
  <body>
    <header>
      
      <nav>...</nav>
    </header>
    ...
  </body>
</html>
```

Semantyczny HTML

```
<header class="header">
  <nav class="nav">...</nav>
</header>
<main class="main">
  ...
</main>
<aside class="aside">
  ...
</aside>
<footer class="footer">
  ...
```

<https://developer.mozilla.org/en-US/docs/Glossary/Semantics>

Kolejny przykład

```
<h1 class="main-heading">I am important (and semantic)!</  
<h2 class="secondary-heading">  
    I am a little less important (but as much semantic)!  
</h2>
```

Hierarchia nagłówków

Nagłówki strukturyzują zawartość strony.

```
<article>
  <h1>Fantastic beasts</h1>
  <section>
    <h2>Where to find them</h2>
    <section>
      <h3>Africa</h3>
      <p>There are many beasts in Africa...</p>
    </section>
    <section>
      <h2>Europe</h2>
      <section>
```

Further reading

- <http://accessiblehtmlheadings.com/>
- https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Using_HTML_sections_and_outlines
- Specyfikacja HTML

Co nam daje semantyka?

- łatwiej nam znaleźć odpowiedni blok kodu, niż przeszukiwać ścianę divów i spanów
- technologia asystująca lepiej przetworzy strukturę naszej strony (accessibility)
- SEO - wyszukiwarki lepiej zindeksują zawartość, wiedząc co jest czym

Grafika

```

```



Linki

```
<a href="https://www.boredpanda.com/cute-baby-highland-ca  
  target="_blank"  
  rel="noopener noreferrer"  
>Even more cute cows</a>  
<a href="mailto:frontend@daftacademy.pl">email us!</a>  
<a href="tel:123456789">call us!</a>
```

Inputy

```
<input> // default type - text  
<input type="number">  
<input type="password">  
<input type="checkbox" name="terms_accepted">  
...
```

☐

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input>

Label

```
<label for="some-input">Email</label>  
<input id="some-input">  
</label>  
Email  
<input name="email">  
</label>
```

Email

a@example.com

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/label>

CSS

Warstwa prezentacji

```
.button {  
    background-color: orange;  
    border-radius: 10px;  
}
```

Załączanie stylów do strony

inline'owo:

```
<button style="background-color: orange; border-radius: 10px; padding: 10px 20px;">
    Click!
</button>
```


Załączanie stylu do strony

tag <style>

```
<style>
  .button {
    background-color: orange;
    border-radius: 10px;
  }
</style>
```

Załączanie styli do strony

oddzielny plik css

```
// my-styles.css
```

```
.button {  
    background-color: orange;  
    border-radius: 10px;  
}
```

```
// plik html
```

```
<head>
```

```
...
```

```
<link href="my-styles.css" rel="stylesheet">
```

```
...
```

Selektory

Określają czego dotyczy dany blok styli.

```
nav { // selektor taga
    ...
}

.button { // selektor klasy
    ...
}

#email { // selektor id
    ...
}
```

Kolejność ma znaczenie
pliki CSS są czytane z góry na dół

```
nav {  
    color: black  
}  
  
nav {  
    color: orange  
}
```

Selektory

Łączenie selektorów

```
a.link { // <a> o klasie link
    ...
}

.button span { // <span> będący potomkiem elementu o klasie button
    ...
}

...
```

https://developer.mozilla.org/en-US/docs/Glossary/CSS_Selector

Specificity

Dokładność selektorów

```
.link {  
    color: black  
}  
  
a.link {  
    color: orange  
}  
  
#about-link {  
    color: purple  
}
```

https://developer.mozilla.org/en-US/docs/Glossary/CSS_Selector

<https://specifishity.com/>

To wygląda strasznie, serio trzeba to liczyć?

Nie. Najlepszym rozwiązaniem (i dobrą praktyką) jest stylowanie tylko po klasach.

BEM

Metodologia CSS, konwencja nazewnicza pozwalająca uniknąć problemu z nadpisywaniem się styli z różnych części strony, a także tzw. *specificity war.*

BEM

Block - Element - Modifier

```
<header class="page-header">
  
  <h1 class="page-header__heading">Heading</h1>
  <nav class="page-navigation">
    <a href="/other-page" class="page-navigation__link">
  </nav>
  <button class="button--primary">Zaloguj</button>
</header>
```

<https://css-tricks.com/bem-101/>

<https://en.bem.info/>

position

w jaki sposób element będzie pozycjonowany na stronie

static - element pozycjonuje się naturalnie we flow dokumentu -
default

position: relative

element pozycjonuje się naturalnie we flow dokumentu, ale dodatkowo możemy ustawić przesunięcie względem tej pozycji

position: absolute

usuwa element z flow, nie zajmuje on żadnego miejsca, możemy przesuwać względem najbliższego pozycjonowanego przodka

position: fixed

jak absolute, ale pozycjonuje względem okna przeglądarki (o ile jego przodek nie ma ustawionego transform, perspective lub filter)

`position: sticky`

pozycjonuje się we flow dokumentu, ale dodatkowo dostaje offset w stosunku do najbliższego, scrollującego się przodka

Elementy blokowe vs inline'owe

block: zajmują całe możliwe miejsce, następne elementy będą "w nowej linii" (np. `<div>`)

inline: zajmują tylko tyle miejsca ile potrzebują, nie mogą zawierać w sobie elementów blokowych, nie akceptują właściwości "width", "height", a marginesy i paddingi nie działają sensownie (przykładem jest ``)

CSS display

Można zmienić sposób wyświetlania elementu np. z inline na block:

```
span {  
    display: block;  
}
```

Nie zmienia to jego klasyfikacji, tj. dalej nie można w nim zagnieżdżać elementów blokowych.

CSS display

inline-block

```
div {  
    display: inline-block;  
}
```

Coś pomiędzy - element zajmuje tyle miejsca ile potrzebuje, ale respektuje ustawione właściwości "width" i "height", a marginesy i paddingi działają normalnie.

Developer Tools

- Prawy przycisk myszy - zbadaj (element)
- Skrót klawiszowy:
 - Mac: Cmd + alt + i
 - Windows/Linux: Ctrl + shift + i
 - F12

Elements

Podgląd struktury HTML i styli poszczególnych elementów

Zanim przejdziemy dalej, poczućmy tę moc.

<https://www.daftacademy.pl>

Konsola

Javascript na wyciągnięcie ręki

Konsola

- wykonywanie dowolnego kodu javascriptowego

```
alert("Hi there!");
```

```
while (true) {  
    console.log("I shouldn't have done this...")  
}
```

- wyświetlanie warningów i errorów przez przeglądarkę
- inne zastosowania:
 - Facebook
 - gog.com

Sources

- pliki strony
- debugger
- snippets

Network

- wszystkie zapytania wykonane przez stronę (obrazki, pliki itp.)

możemy filtrować po typie

na dole podsumowanie rozmiaru wykonanych requestów

- filmstrip - jak wyglądało ładowanie naszej strony klatka po klatce

Pytania?