

HTML



CSS



RWD



Maciej Wójcik

HTML CSS

- Co to jest html, css?
- Czemu używamy ich wspólnie?
- Jak tworzyć stronę w html'u?

HTML – struktura dokumentu

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6 </head>
7 <body>
8
9 <!-- content -->
10
11 </body>
12 </html>
```

Składnia , Podstawowe <tagi>

- Składnia:
 - **<tag>** zawartość tagu **</tag>**
 - **<tag atrybuty tagu >** zawartość tagu **</tag>**
- Co się zmieniło od czasu HTML'a 5?
- Dlaczego istotne jest używanie odpowiednich tagów?
- Podstawowe tagi:
 - header, footer
 - div, span
 - p
 - nav, ul, li
 - a
 - img
 - form, button, input

Arkusze stylów - CSS

- Zmiany przy okazji wprowadzenia html5 i css3/css2
 - Rezygnacja z budowania obszernego pliku .html
 - Wprowadzenie względnie przejrzystego dokumentu z strukturą strony oraz osobnego pliku z jej wyglądem
 - Wyraźne oddzielenie treści strony i jej szkieletu od sposobu wyświetlania
 - Łatwość zmiany wartości atrybutu występującego w każdym elemencie
 - Modularność – budowa strony w oparciu o arkusze css odpowiadających za poszczególne funkcjonalności
- 3 sposoby deklaracji

Arkusze stylów - CSS

```
1 <body>
2   <div width:"100px"><center><b><h1>Title</h1></b></center></div>
3   <div><b><h2>SubTitle</h2></b></div>
4   <p>
5     <b>F</b>irst Letter
6     <article>Some text...</article>
7     <center><h3><b>a</b>nother subTitle with first letter bold</center>
      </h3>
8   </p>
9 </body>
```

```
1 <body>
2   <h1>Title</h1>
3   <h2>SubTitle</h2>
4   <p>
5     First Letter
6     <article>Some text...</article>
7     <h3>another subTitle with first letter bold</h3>
8   </p>
9 </body>
```

```
1 h1{
2   width: 100px;
3 }
4 p::first-letter, h3::first-letter{
5   font-weight: bold;
6 }
7
```

Identyfikacja elementów html

- Jak odwoływać się w css do konkretnego elementu w html'u?
- Do wielu naraz:
 - Używając tagu np. div, p, img
 - Używając klasy danego/danych elementów np. my-class
- Konkretnie do unikalnego elementu:
 - Używając unikalnego id np. my-id

Identyfikacja elementów html

```
1 <body>
2
3     <div>content</div>
4
5     <div class="my-class">content</div>
6
7     <div id="my-id">content</div>
8
9 </body>
```

```
1 div{
2 /*content*/
3 }
4 .my-class{
5 /*content*/
6 }
7 #my-id{
8 /*content*/
9 }
```


Tworzenie szkieletu strony

- 4 główne sposoby na tworzenie szkieletu stron
 - **Tabele**
 - intuicyjne, niestety dość prymitywne i nie responsywne
 - **Użycie właściwości Float**
 - dosyć uniwersalne, ciężko jednak zbudować na tym całą stronę
 - **FlexBox**
 - duża responsywność, wygoda, wspierane tylko przez najnowsze wersje przeglądarek
 - **CSS Frameworks**
 - gotowe rozwiązanie, często bardzo skraca czas tworzenia strony, zawiera błędy, niedoskonałości

Tworzenie szkieletu strony

- Stwórzmy szkielet strony w oparciu o właściwość float

```
1 <body>
2   <header>
3     HEADER CONTENT
4   </header>
5
6   <main>
7     <nav>navigation</nav>
8     <div>content</div>
9   </main>
10
11  <footer>
12    FOOTER
13  </footer>
14 </body>
```

```
1 header, main, footer{
2   width: 100%;
3   height: 100px;
4   border: 1px solid black;
5 }
6
7 div, nav{
8   float: left;
9   width: 400px;
10 }
```

HTML



Tekst

- Tekst na stronie jest każdym znakiem w dokumencie .html, który jest wewnątrz tagu, a nie jako jego właściwość.
- Dobra praktyka – do formatowania tekstu nie używajmy html'a tylko css'ów!
- System nagłówków:
 - Gotowe zaimplementowane w stylach przeglądarki nagłówki
 - h1, h2, h3, h4,

```
<p>  
    <h1>Title</h1>  
    <h3>SubTitle</h3>  
    Text...  
</p>
```

Obrázky

- Składnia: ``

- Można dołączać pliki lokalne jak i z url

```
  

```

- alt czyli tekst alternatywny

```

```

Linki

- Składnia:

```
<a href="www.google.pl">tekst odnośnika</a>
```

- Zastosowanie:

- Zewnętrzne odnośniki
- Wewnętrzne odnośniki – nawigacja po stronie, menu

Listy

- `` , ``
 - kontener listy
- ``
 - element listy
- ul – lista nieuporządkowana (punkty)
- ol – lista uporządkowana (1,2,3...)
- Często wykorzystywane do budowy prostego menu

```
15▼ <ul>
16     <li>item</li>
17     <li>item</li>
18     <li>item</li>
19     <li>item</li>
20 </ul>
21
22▼ <ol>
23     ..... <li>item</li>
24     <li>item</li>
25     <li>item</li>
26     <li>item</li>
27 </ol>|
```

Interakcja z użytkownikiem

- Formularze, pola do wpisywania znaków, przyciski ...
- Kontener do zbierania danych od użytkownika <form>
- Przyciski <button>
- Najprostsze pole <input>

```
15 <form>
16     Input1:
17     <input type="text" name="input1">
18
19     Input2:
20     <input type="radio" name="input2">
21
22     <button>Enter</button>
23 </form>
```


Dodanie nowych elementów do naszej strony

- Cel:
 - Wykorzystanie poznanych elementów w celu zwiększenia funkcjonalności i atrakcyjności strony
 - Możliwość wprowadzania danych przez użytkownika
 - Nawigacja, menu

CSS



Składnia

- Selector{
 property: value;
}

```
1 |div{  
2 |    width: 50px;  
3 |}
```

- Dodanie pliku css do html'a w <head>
 - <link rel="stylesheet" href="style.css" />
- Kilka zasad:
 - Nowsze style nadpisują starsze
 - Im dalej zadeklarowany plik css w <head> tym jest ważniejszy i nadpisuje resztę
 - Bardziej dokładne opisy zastępują mniej dokładne

Selektory

- Proste selektory:
 - Jeśli selektor ma być tagiem to piszemy po prostu jego nazwę
 - Jeśli selektor to klasa to dodajemy przed nazwą kropką np. „.class”
 - Jeśli selektor to id to dodajemy # np. „#id”
- Bardziej skomplikowane konstrukcje (wybrane)

```
6  div:hover{
7      /*<div>, na którym jest aktualnie kursor*/
8  }
9
10 div > p{
11     /*<p> gdzie rodzicem jest<div>*/
12 }
13
14 *{
15     /*wszystkie elementy*/
16 }
17
18 div, p{
19     /*wszystkie <div> i wszystkie <p>*/
20 }
```

Podstawowe własności

- Margin / Padding
 - Width & height
 - Position
 - Float
 - Color
 - Background
 - Text-size
-
- Własności jest o wiele więcej, im bardziej skomplikowane tym większe ryzyko nie kompatybilności z wszystkimi przeglądarkami!
 - caniuse.com

Przykładowe jednostki

- **margin / padding**

- px, %, vw, vh

- **width & height**

- px, %, vw, vh

- **color**

- rgb, text, #hex

- **text-size**

- px, %, em

Float

- Istotne zarówno do budowy layoutu strony jak i do formatowania większego tekstu z ilustracjami itd.
- Tracimy kontrolę nad dokładną (absolutną) pozycją, ale zyskujemy responsywność i uzależnienie od innych obiektów

Position

- Static (zwykłe)
- Absolute
- Relative
- Fixed
- Sticky

Dodanie stylu do naszej strony

- Ustawienie poprawnych rozmiarów naszych elementów
- Lepsze ustawienie położenia elementów
- Dodanie kolorów, podświetleń

RWD



Responsive Web Design

- Jak stworzyć stronę, która będzie się poprawnie wyświetlać na różnych ekranach?
- Telewizory, lodówki, zegarki, telefony, tablety, okulary
- Trzeba tworzyć uniwersalne rozwiązania pasujące do wszystkich typów ekranu i mieć na względzie
 - Różne proporcje ekranu
 - Różną rozdzielczość
 - Różną wielkość wyświetlacza

Jak to osiągnąć?

- Nie koniecznie trzeba używać JavaScript'a
- W wielu przypadkach wystarczy dobra znajomość CSS
- Sposoby na implementacje strony zgodnie z rwd
 - Zamiana jednostek statycznych na dynamiczne. (% , em, vw, vh)
 - Relatywne określanie pozycji, zamiast absolutnie.
 - Media Queries
 - CSS Frameworks

Media Queries

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

```
6▼ @media screen and (min-width: minimalna szerokość w px) {  
7    /*style które nadpiszą aktualne,  
8    gdy szerokość okna będzie mniejsza niż...  
9    */  
10 }
```

```
1 <div> Responsywny element</div>
```

```
1 div{  
2     width: 1000px;  
3 }  
4  
5  
6 @media screen and (min-width: 480px) {  
7     div{  
8         width: 320px;  
9     }  
10 }
```

A może coś gotowego?

- Frameworki CSS



Bootstrap

- Zawiera przygotowany zestaw plików html, css ,js
- Jak dodać do projektu?
 - getbootstrap.com → getting started

```
1 <head>
2
3   <!-- Latest compiled and minified CSS -->
4   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/
bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg320mUcww7on3RYdg4Va+PmSTsz
/K68vbdEjh4u" crossorigin="anonymous">
5
6   <!-- Latest compiled and minified JavaScript -->
7   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNlcpPD7Txa"
crossorigin="anonymous"></script>
8
9 </head>
```

Bootstrap – jak korzystać?

- Tworzymy dokument html tak jak normalnie, z tą różnicą, że poza naszymi klasami w tagach dodajemy klasy zaimplementowane w bootstrapie.

```
1 <button type="button" class="btn btn-default">Default</button>
2
3 <button type="button" class="btn btn-primary">Primary</button>
4
5 <button type="button" class="btn btn-success">Success</button>
6
7 <button type="button" class="btn btn-info">Info</button>
8
9 <button type="button" class="btn btn-warning">Warning</button>
10
11 <button type="button" class="btn btn-danger">Danger</button>
12
13 <button type="button" class="btn btn-link">Link</button>
```

Default

Primary

Success

Info

Warning

Danger

Link

Bootstrap – wielkie możliwości niskim kosztem

- **Grid System**
- Formularze
- Obrazki, buttony, tabele, wells, ikony, nawigacje...

Bootstrap – Formularze

```
1 ▾ <form>
2   <div class="form-group">
3     <label for="exampleInputEmail1">Email address</label>
4     <input type="email" class="form-control" id="exampleInputEmail1" placeholder="Email">
5   </div>
6 </form>
```

Email address

Bootstrap – Grid System

- 12 zdefiniowanych kolumn w <row>
- Prostota i szybkość budowania layoutu

```
1 <div class="container">
2   <div class="row">
3     <div class="col-md-3">item</div>
4     <div class="col-md-3">item</div>
5     <div class="col-md-3">item</div>
6     <div class="col-md-3">item</div>
7   </div>
8   <div class="row">
9     <div class="col-md-4">item</div>
10    <div class="col-md-4">item</div>
11    <div class="col-md-4">item</div>
12  </div>
13  <div class="row">
14    <div class="col-md-6">item</div>
15    <div class="col-md-6">item</div>
16  </div>
17  <div class="row">
18    <div class="col-xs-12 col-md-6">mobile item</div>
19    <div class="col-xs-12 col-md-6">mobile item</div>
20  </div>
21 </div>
```

Dlaczego warto korzystać z Bootstrapa?

- W kilka minut od podstaw stworzymy stronę o podobnych elementach jak poprzednio, ale w krótszym czasie i będzie to strona w pełni responsywna.
- Wykorzystamy Grid System w celu poprawy responsywności zawartości całej strony

Edycja Bootstrap'a

- Kolejność stylów,
- !important
- Nadpisanie plików css
- Ściągnięcie scss'ów, edycja i kompilacja

Dziękuję za uwagę

- Prezentacja wraz z przykładami jest umieszczona na

github.com/MaciejWWojcik/lecture-html-css-rwd

Dodatki



SCSS

- Sass to preprocesor CSS
- SCSS = SASS + CSS
- Szybsze, bardziej intuicyjne programowanie CSSów
- Zagnieżdżanie cssów,
- Mixiny
- Importy plików

```
1  .foo p {
2      font-size: 12px;
3  }
4  .foo small {
5      font-size: 10px;
6  }
7  .foo small:hover {
8      color: blue;
9  }
10
11 /*SCSS*/
12 |
13 .foo {
14     p {
15         font-size: 12px;
16     }
17     small {
18         font-size: 10px;
19     }
20     &:hover {
21         color: blue;
22     }
23 }
24 }
```


CSS – calc()

- Zalety:
 - Proste i intuicyjne podejście
 - 1 linijka CSS, zamiast kilku JS
- Użycie:
 - Załóżmy, że chcemy mieć na całej szerokości strony dwie kolumny. Jedna o szerokości stałej 500px, a druga dopełniająca się na całą pozostałą szerokość.

```
1 |.left{
2 |    width: 500px;
3 |}
4 |
5 |.right{
6 |    width: calc( 100% - 500px)
7 |}
```

CSS - animacje

```
1  div {  
2      width: 100px;  
3      height: 100px;  
4      background-color: red;  
5      animation-name: example;  
6      animation-duration: 4s;  
7  }  
8  
9  @keyframes example {  
10     from {background-color: red;}  
11     to {background-color: yellow;}  
12 }
```