HTML I CSS

v.1.6

Coders Lab
SZKOŁA PROGRAMOWANIA

Plan

- Co to jest HTML?
- Znaczniki HTML – struktura
- Struktura dokumentu HTML
- Formatowanie tekstu w HMTL
- Hiperłącza
- Obrazki w dokumentach HTML
- Tabele
- Elementy liniowe a elementy blokowe
- Identyfikatory i klasy
- CSS – kaskadowe arkusze stylów
- Kolory i jednostki w CSS
- Stylowanie tekstu
- Kolor i tło w CSS
- Stylowanie list
- Stylowanie tabel
- Projektowanie formularzy
- Relacje między elementami HTML
- Polecane materiały online

Co to jest HTML?

Jak wygląda HTML?

Na początek

Najpierw zobacz, jak wygląda kod HTML. Omówimy, czym jest, z czego się składa i do czego służy.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Strona startowa</title>
6  </head>
7  <body>
8     Hello World
9  </body>
10 </html>
```

HyperText Markup Language (HTML)

- Hipertekstowy język znaczników.
- Język komputerowy (służący do komunikacji z komputerem) używany do opisu stron internetowych.
- Nie jest językiem programowania – w jego składni nie przewidziano wyrażeń obliczeniowych, warunkowych czy iteracyjnych.

- Znaczniki – to jego elementy składowe, opisują strukturę dokumentu HTML.
- HTML opisuje tylko strukturę, a nie wygląd strony.

Trochę historii

- HTML4 – oficjalnie zatwierdzony w **1999 r.**
- Organizacje odpowiedzialne za rozwój HTML:
  - W3C *(*World Wide Web Consortium*)*,
  - WHATWG *(*Web Hypertext Application Technology Working Group*)*.
- HTML5 – pierwszy szkic (working draft) opublikowany przez W3C w **2008 r.**
- HTML5 – zarekomendowany **28.10.2014 r.** przez W3C.

Znaczniki HTML – struktura

Znaczniki (tagi)

Komentarze w HTML

Czym są komentarze?

- W języku HTML istnieją **komentarze**.
- Są to części dokumentu omijane podczas wyświetlania strony HTML.

Przykład

```
<!--  Main navigation -->
<nav>
<ul>
        <li>Start</li>
        <li>Kontakt</li>
</ul>
</nav>
```

Znaczniki – atrybuty

```
<!-- Atrybut attribute przypisany do tagu html -->
<html attribute="value">
    ...
</html>

<!-- Atrybut lang przypisany do tagu html -->
<html lang="pl-PL">
    ...
</html>
```

Atrybuty są to pary nazwa – wartość rozdzielone znakiem **"="** i przypisane do znacznika np. **html**, **p**, **section**.
Wartość przypisana do atrybutu może być zawarta w pojedynczych lub podwójnych cudzysłowach.

Atrybuty zawsze zapisujemy w znaczniku otwierającym. W jednym znaczniku możemy zapisać wiele atrybutów.

Znaczniki (tagi) – atrybuty

Struktura dokumentu HTML

Deklaracja dokumentu

DOCTYPE

Pierwszym tagiem, który powinien znaleźć się na początku każdej strony HTML, jest **DOCTYPE**. Informuje on przeglądarkę, że jest to dokument HTML, który można wyświetlić.

Zapis w HTML5

```
<!DOCTYPE html>
```

Deklaracja języka HTML

HTML

Cała treść strony powinna się znaleźć pomiędzy tymi znacznikami **<html> </html>**.

Przykład

```
<html>
    <!-- treść strony  -->
</html>
```

Deklaracja języka strony

Język strony

- Język strony (nie mylić z HTML) deklarujemy za pomocą atrybutu **lang**.
- Pierwsze dwie litery (małe) są deklaracją używanego języka, kolejne dwie litery (wielkie) są deklaracją kraju, w którym tego języka się używa.

Przykład

```
<html lang="pl-PL">
  <!-- treść strony -->
</html>
```

Sekcja nagłówkowa

Nagłówek strony

- Nagłówek strony znajduje się w znaczniku **<head>**.
- Opisuje on ustawienia dokumentu HTML.
- Jego zawartość jest niewidoczna na stronie.

Przykład

```
<head>
  <!-- Zawartość sekcji head -->
</head>
```

Kodowanie strony

Znacznik meta, atrybut charset

- Tag **meta** z atrybutem **charset** służy do ustawienia odpowiedniego kodowania strony.
- Potrzebny jest do prawidłowego wyświetlania
- np. polskich znaków.

Przykład

```
<meta charset="UTF-8">
```

Opis strony

Meta description

- Tag meta z atrybutem **name** ustawionym na wartość **description** i atrybutem **content** służy do ustawienia opisu strony.
- Opis strony jest niewidoczny na stronie, wyświetlany w wynikach wyszukiwarek oraz istotny z punktu widzenia SEO.
- Opis powinien być zrozumiały, zarówno dla człowieka, jak i dla wyszukiwarki (silnika indeksującego strony).
- Zalecana długość to nie więcej niż 160 znaków.

Przykład

```
<meta name="description" content="opis">
```

Viewport

Meta viewport

- Tag meta z atrybutem **name** ustawionym na **viewport** i atrybutem **content** służy do ustawiania szerokości strony
-
- Użycie **width=device-width** sprawia, że szerokość strony internetowej odpowiada szerokości ekranu w pikselach niezależnych od urządzenia, **initial-scale=1** nakazuje tym samym pikselom zachowanie proporcji 1:1 do pikseli CSS niezależnie od orientacji urządzenia
-
- Przykładowo: iPhone 6 ma fizyczną rozdzielczość 1334px x 750px. Przy użyciu tagu meta z atrybutem **name** ustawionym na **viewport** strona na powyższym iPhonie będzie wyświetlać się w rozdzielczości 667px x 375px i będzie skalować się niezależnie od orientacji

Przykład

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

Tytuł strony

Tytuł

- Tag **title** służy do ustawienia tytułu strony.
- Nie jest wyświetlany na samej stronie (ale pojawia się np. na karcie w przeglądarce).
- Jest istotny z punktu widzenia SEO, ma wpływ na pozycję strony w wyszukiwarkach.
- Zalecana długość to nie więcej niż 65 znaków.

Przykład

```
<title>To jest tytuł</title>
```

Sekcja body

Body

Ciało strony – jak sama nazwa wskazuje – znajduje się w **body**.

Wszystko to, co widzimy na stronie, jest umieszczone między **<body>** a **</body>**.

Przykład

```
<body>
  <!-- treść strony -->
</body>
```

Inne elementy

div, span,

div – to blokowy element HTML, którego celem jest obejmowanie większych partii kodu, np. sekcji. Jego przeznaczenie nie jest określone tak jak np. elementu p (paragraf).

span – to element inline, jego celem jest obejmowanie mniejszych części kodu np. tekstu, wyrazów, obrazków.

Przykład

```
<body>

  <div class="header">

    <h1>
      To jest nagłówek
      <span> pierwszego </span> rzędu

    </h1>

  </div>

</body>
```

Struktura dokumentu HTML – podsumowanie


```
<!DOCTYPE html>
<html lang="pl-PL">
  <head>
    <meta charset="UTF-8">
    <meta name="description" content="opis">
    <meta name="keywords" content="key1, key2">
    <title>To jest tytuł</title>
  </head>
  <body>
        <!-- treść strony -->
  </body>
</html>
```

Formatowanie tekstu w HTML

Formatowanie tekstu w HTML

Tekst formatujemy za pomocą określonego zbioru znaczników. Służą one do porządkowania strony i określania, czym jest dany tekst (np. nagłówek, akapit, lista).

Przeglądarki mają predefiniowane sposoby wyświetlania zawartości takich znaczników (np. nagłówek będzie miał duży rozmiar czcionki).

Za pomocą znaczników można osiągnąć **podstawowe formatowanie**.
Bardziej zaawansowane dostępne jest przy użyciu **kaskadowych arkuszy stylów (CSS)**.

Nagłówki

Od h1 do h6

Nagłówki są używane do tytułowania większych partii treści na stronie, możemy je porównać do tytułów i śródtytułów artykułów prasowych.

Znacznik **<h1>** to najważniejszy w hierarchii nagłówek (wyświetlany największą czcionką), **<h6>** − najmniej znaczący.

W praktyce najczęściej używane są nagłówki **h1**, **h2**, **h3**.

Przykład

```
<!-- tutaj DOCTYPE i head -->
<body>
    <h1>Witaj na mojej stronie</h1>
    <h2>O mnie</h2>
    <h3>Moje projekty</h3>
</body>
```

Od tego momentu będziemy w przykładach kodu, pokazywać tylko cześć pomiędzy znacznikami body.

Nagłówki

Na stronie

Witaj na mojej stronie

O mnie

Moje projekty

Moje projekty

Moje projekty

W kodzie

```
<h1>Witaj na mojej stronie</h1>
<h2>O mnie</h2>
<h3>Moje projekty</h3>
<h4>Moje projekty</h4>
<h5>Moje projekty</h5>
```

Paragraf

Znacznik p

Znacznik **p** (paragraph) jest używany do grupowania tekstu w **akapity**.

Będzie bardzo przydatny, gdy zaczniemy stylować w CSS!

W kodzie

```
<h1>Pan Tadeusz</h1>
<p> Litwo! Ojczyzno moja! Ty jesteś jak zdrowie.
Ile cię trzeba cenić, ten tylko się dowie…
</p>
```

Na stronie

Pan Tadeusz

Litwo! Ojczyzno moja! Ty jesteś jak zdrowie. Ile cię trzeba cenić, ten tylko się dowie...

Znaki specjalne

Od ampersandu do średnika

Znaki specjalne mają – zgodnie z nazwą – specjalny zapis w języku HTML.
Zawsze zaczynają się znakiem ampersand – **&**,
a kończą średnikiem – **;**.

Używamy ich, jeśli chcemy wyświetlić znaki zarezerwowane w HTML lub gdy nie jest możliwe wprowadzenie danego znaku z klawiatury.

Znaki specjalne

Na stronie

 - niełamliwa spacja (non-breaking space)
™ - znak handlowy (trade mark sign)
< - znak mniejszy niż (less-than sign)
> - znak większy niż (greater-than sign)
© - znak copyright (copyright sign)
® - zastrzeżony znak handlowy (registered sign)
α - mała grecka litera alfa

W kodzie

```
<p>
  - niełamliwa spacja (non-breaking space)<br>
&trade; - znak handlowy (trade mark sign)<br>
&lt; - znak mniejszy niż (less-than sign)<br>
&gt; - znak większy niż (greater-than sign)<br>
&copy; - znak copyright (copyright sign)<br>
&reg; - zastrzeżony znak handlowy (registered sign)<br>
&alpha; - mała grecka litera alfa<br>
</p>
```

Niełamliwa spacja

- Niełamliwa spacja (non-breaking space) powinna być używana, aby nie zostawiać pojedynczych znaków na końcu linii.
- Oznacza się są następująco: ** **
- Zapobiega łamaniu wiersza w miejscu, w którym została użyta.
-
- Nie powinna być używana do wprowadzania dodatkowego odstępu (ciąg znaków ** **).

W kodzie

```
<p>
 Litwo! Ojczyzno Moja. Ty&n
 bsp; jesteś jak zdrowie. 
 </p>
```

Na stronie

Litwo! Ojczyzno Moja. Ty jesteś jak zdrowie.

Nowa linia

br

- Przełamanie linii (line break) to symbol oznaczający przełamanie linii w tekście HTML. Przeglądarki ignorują białe znaki (enter, tabulator, ciąg spacji) w kodzie HTML.
- Zamiast tego należy zastosować element **
**.
- **
** nie ma znacznika zamykającego, jest to tzw. znacznik samozamykający.

W kodzie

```
<p>
Sprawdzamy, jak
    przeglądarka    interpretuje  białe znaki.

    Do przełamania linii należy użyć <br>
elementu    &lt;br&gt;
</p>
```

Na stronie

Sprawdzamy jak przeglądarka interpretuje białe znaki. Do przełamania linii należy użyć elementu

Linia horyzontalna

hr

Ten znacznik służy do narysowania poziomej linii (horizontal line).

Na stronie

Radziwiłł z których później odkładać goście proszeni.

Ty, co Francuz wymyśli, to mówiąc, że słuchał rozmowy.

W kodzie

```
<p>
    Radziwiłł z których później odkładać goście proszeni.
</p>
<hr>
<p>
    Ty, co Francuz wymyśli, to mówiąc, że słuchał rozmowy.
</p>
```

Wyróżnienie

strong

- **Pogrubia** czcionkę i wzmacnia znaczenie słowa.

em

- *Pochyla* czcionkę i wzmacnia znaczenie słowa.
- Oba służą do wyróżnienia tekstu i wzmocnienia znaczenia, w przypadku jeśli czytnik ekranu czyta dany tekst.
-
- Są to nowe elementy wprowadzone w HTML5.

W kodzie

```
<p>
    W tym zdaniu pewne <strong> słowo </strong> ma bardzo dużą wagę. Większa część kolejnego zdania <em>warta jest podkreślenia</em>.
</p>
```

Na stronie

W tym zdaniu pewne **słowo** ma bardzo dużą wagę. Większa część kolejnego zdania *warta jest podkreślenia*.

Wyróżnienie

Znacznik b

- **pogrubia** czcionkę.

Znacznik i

- *pochyla* czcionkę.
- Znaczniki te służą do wyróżnienia tekstu podobnie jak **** i ****. Nie wzmacniają jednak znaczenia tekstu, czytnik ekranu nie zaakcentuje takiego tekstu.

W kodzie

```
<p>
  W tym zdaniu pewne <b>słowo</b> ma
  bardzo dużą wagę. Większa część kolejnego
  zdania <i>warta jest podkreślenia</i>.
</p>
```

Na stronie

W tym zdaniu pewne **słowo** ma bardzo dużą wagę. Większa część kolejnego zdania *warta jest podkreślenia*.

Pomniejszenie

Duży, mniejszy, najmniejszy

- Znacznik **small** mniejsza czcionkę o jeden rozmiar.
- Jest to rzadko używany znacznik, ponieważ
- pełną możliwość doboru wielkości czcionki daje technologia CSS.

W kodzie

```
<p>
    Duży, <small> mniejszy, <small>
    najmniejszy </small> </small>.
</p>
```

Na stronie

Duży, mniejszy, najmniejszy.

Zaznaczenie

Podświetlenie tła

- Znacznik **mark** zaznacza tekst
- przez podświetlenie jego tła.

W kodzie

```
<p>
Myślę, że powinniście <mark>zwrócić na to
  uwagę</mark>.
</p>
```

Na stronie

Myślę, że powinniście zwrócić na to uwagę.

Indeks górny i dolny

Indeks górny

- Znacznik **sup** umieszcza tekst w indeksie górnym.

Indeks dolny

- Znacznik **sub** umieszcza tekst w indeksie dolnym.

Jeśli masz problem z zapamiętaniem, wyobraź sobie, że „p” ma brzuszek na górze, a „b” na dole ☺

W kodzie

```
<p>
    Każdy chemik wie, że wzór sumaryczny kwasu siarkowego to H<sub>2</sub>SO<sub> 4 </sub>. Każdy matematyk wie, że 2<sup>0</sup> = 1.
</p>
```

Na stronie

Każdy chemik wie, że wzór sumaryczny kwasu siarkowego to $H_2 SO_4$. Każdy matematyk wie, że $2^0 = 1$.

Przekreślanie

Tekst przekreślony

- Znacznik **s** oznacza przekreślenie (strikethrough).
-
- Używamy go, gdy tekst jest nieaktualny, ale nie powinien zostać usunięty.

W kodzie

```
<p>
  <s>Oferta aktualna tylko do poniedziałku! </s>
</p>
```

Na stronie

~~Oferta aktualna tylko do poniedziałku!~~

Usuwanie i wstawianie

Para znaczników

Znacznika **del** używamy w parze ze znacznikiem **ins**, gdy chcemy zwrócić uwagę, że zastąpiliśmy fragment tekstu innym tekstem.

W kodzie

```
<p>
 Zajmuję się fotografią natury,
 <del> ślubną</del> <ins> głębinową</ins>.
      <br>
 Cena za sesję to <s> 98,94 zł </s>
 <ins>86,00  zł</ins>.
</p>
```

Na stronie

Zajmuję się fotografią natury, ~~ślubną~~ głębinową.
Cena za sesję to ~~98,94 zł~~ 86,00 zł.

Listy

W HTML istnieją trzy rodzaje list:

- uporządkowana (**<ol>**, ordered list),
- nieuporządkowana (**<ul>**, unordered list)
- definicji (**<dl>**, definition list).

- Listy składają się z kolejnych punktów (numerowanych bądź oznaczanych innymi symbolami).
- Każda lista składa się ze znacznika określającego jej typ (**<ol>**, **<ul>** lub **<dl>**) oraz znaczników pełniących funkcję elementów listy.
- Listy możemy w sobie zagnieżdżać. Dzięki temu tworzymy listę wewnątrz listy.

Lista uporządkowana

- Lista uporządkowana (ordered list) ma elementy posiadające pewną kolejność oznaczaną za pomocą cyfr bądź liter alfabetu.
- Domyślnie elementy listy są oznaczane cyframi arabskimi (od 1).
- Cyfry rzymskie bądź litery można uzyskać dzięki stylom CSS.

```
<ol>
    <li>Element 1</li>
    <li>Element 2</li>
    <li>Element 3</li>
</ol>
```

Lista nieuporządkowana

- Lista nieuporządkowana (unordered list) – jej wszystkie elementy są oznaczane tym samym symbolem.
- Domyślnie elementy listy są oznaczane symbolem kropki (disc).
- Nie ma określonej kolejności elementów.
- Symbol można zmienić za pomocą CSS.

```
<ul>
    <li>Element 1</li>
    <li>Element 2</li>
    <li>Element 3</li>
</ul>
```

Lista definicji

- Lista definicji (definition list) składa się z par pojęcie – definicja.
- Do każdego pojęcia przypisujemy
- określoną definicję.
- Pojęcia zapisujemy w znacznikach **dt**
- (definition term).
-
- Definicje zapisujemy w znacznikach **dd** (definition description).
- Jedna definicja może opisywać wiele pojęć.
- Jedno pojęcie może mieć wiele definicji.

```
<dl>
    <dt>Pojęcie 1</dt>
    <dd>Definicja pojęcia 1</dd>
    <dt>Pojęcie 2</dt>
    <dt>Pojęcie 3</dt>
    <dd>Definicja pojęcia 2 i 3</dd>
    <dt>Pojęcie 3</dt>
    <dd>Pierwsza definicja pojęcia 3</dd>
    <dd>Druga definicja pojęcia 3</dd>
</dl>
```

Zagnieżdżanie list

- Możemy zagnieździć jedną listę w innej liście.
- Zagnieżdżać można wszystkie rodzaje list.
- Liczba zagnieżdżeń jest nieograniczona.
- **Zwróć uwagę jak jest zagnieżdżona wewnętrzna lista. Znajduje się ona w elemencie li.**

```
<ol>
    <li>
        Element 1
        <ul>
            <li>Podpunkt 1.1</li>
            <li>Podpunkt 1.2</li>
        </ul>
    </li>
    <li>Element 2</li>
    <li>Element 3</li>
</ol>
```

Hiperłącza

Hiperłącza

Linki

- Nazwa pochodzi od angielskiego słowa hyperlink, inaczej odsyłacze, linki.
- Łączą ze sobą poszczególne strony w internecie, umożliwiają przechodzenie od strony do strony.
- Linki dzielimy na wewnętrzne i zewnętrzne.
- Linki wewnętrzne dzielimy na:
  - prowadzące do sekcji na tej samej stronie,
  - prowadzące do innej podstrony w ramach tej samej strony.

Hiperłącza

Element a

- Tekst, który ma być wyświetlony w podanym linku, umieszczamy między następującymi znacznikami:
  - **<a>**,
  - ****.
  -
- Jest to tzw. tekst anchora.

```
<a>tekst linka</a>
<!-- np: -->
<a>Kliknij żeby przejść do artykułu</a>
```

Element a – atrybut href

Atrybut href

Wartość atrybutu **href** (hypertext reference) informuje przeglądarkę, na jaką stronę ma przenieść nas link.

```
<a href="http://www.coderslab.pl">
      Szkoła programowania
</a>
```

Element a – atrybut rel

Atrybut rel

- Atrybut **rel** (relation) jest stosowany na potrzeby pozycjonowania.
- Określa, jak powinien zachować się Googlebot indeksujący daną stronę.
- Może przyjmować wartości:
  - **dofollow** (podążaj, domyślna),
  - **nofollow** (nie podążaj).

```
<a href="http://www.coderslab.pl" rel="nofollow">
      Szkoła programowania
</a>
```

Element a – nofollow

Podążaj, nie podążaj

- Im więcej zewnętrznych linków prowadzących do strony, tym wyższą ma ona rangę.
- Bot indeksując (przeszukując) daną stronę, analizuje jej zawartość.
- Jeśli trafi na link (element **a**), przechodzi na stronę, do której prowadzi link – podnosi to rangę tej strony.
- Jeżeli nie chcemy podnosić rangi innej, zewnętrznej strony, ale jednocześnie chcemy umieścić do niej link, stosujemy atrybut **nofollow** – bot nie podąży za linkiem, ranga strony nie będzie zmieniona.

Ranga zewnętrznej strony

- Strona, która ma zewnętrzne linki bez atrybutu **nofollow**, przekazuje część swojej reputacji stronom, do których odsyła.
- Używamy linków z atrybutem **nofollow,** aby nie podnosić rangi stron reklamowych (reklamy występują także w formie linków).
- Linków z atrybutem **nofollow** używamy też, aby zapobiec spamowi na forach internetowych.
- W komentarzach każdy umieszczałby mniej lub bardziej sensowne treści zawierające linki zewnętrzne do swoich stron, aby podnieść rangę swoich stron.

Element a – atrybut target

Atrybut target

- Ten atrybut określa,
- w jakim miejscu ma być otwarty link.
- Dostępne wartości to:
  - **_blank** – nowa karta,
  - **_self** – ta sama karta.

```
<a href="http://www.coderslab.pl" target="_self">
    Szkoła programowania
</a>
```

Ramka pływająca

Strona w stronie

- Element **iframe** (inline frame, ramka pływająca) umożliwia zawieranie dokumentu HTML wewnątrz innego dokumentu HTML.
- W atrybucie **src** elementu **iframe** podaje się adres URL strony mającej wyświetlić się w tej pływającej ramce.

```
<iframe src="http://www.coderslab.pl">
    Szkoła programowania
</iframe>
```

Element a – sekcje strony

Linki do sekcji

- Sekcje pomagają w nawigacji po stronie, szczególnie gdy zawiera dużo informacji,
- a zobaczenie jej treści wymaga przewijania
- w pionie.

- Przez sekcję rozumiemy dowolny element html, dla którego zdefiniowaliśmy atrybut **id**.

```
<p id="sekcja1">
    Lorem ipsum dolor sit amet
</p>
```

Element a – sekcje strony

Przeniesienie do miejsca na stronie

- Do każdego elementu możemy dodać atrybut **id** (identyfikator) o dowolnej wartości, który jednoznacznie określa ten element w obrębie danej strony.
- Aby utworzyć link do elementu, który ma określony atrybut **id**, należy w atrybucie **href** elementu **a** podać **id** elementu, do którego linkujemy, i poprzedzić go hashtagiem (#).

```
<a href="#sekcja1">
    Przejdź do sekcji 1
</a>

<!-- tu dużo tekstu -->

<p id="sekcja1">
    Lorem ipsum dolor sit amet
</p>
```

Linki do podstron

Wiele plików html

- Każda podstrona to oddzielny plik **.html**, wszystkie razem tworzą jeden serwis www.
- Aby utworzyć link do podstrony umieszczamy nazwę pliku wraz z rozszerzeniem w atrybucie **href** linka.

```
<a href="podstrona1.html">
    Przejdź do podstrony 1
</a>
<a href="podstrona2.html">
    Przejdź do podstrony 2
</a>
<a href="#">
    Link do bieżącej strony
</a>
```

Obrazki w dokumentach HTML

Obrazki – umieszczanie grafiki

Linki, odnośniki, odsyłacze

- Element **image** służy do umieszczania grafiki, jest to element samozamykający.
- Element ma atrybut **src** (source) określający położenie pliku graficznego, który chcemy umieścić na stronie.
- Najczęściej wykorzystywane są pliki w następujących formatach:
  - JPG,
  - PNG
  - GIF.

```
<img src="coderslab_logo.png">

<img src="images/HTML5_sticker.png">
```

Obrazki – szerokość i wysokość

Linki, odnośniki, odsyłacze

- Podanie szerokości (**width**) i wysokości (**height**) jest opcjonalne, ale pozwala przeglądarce zarezerwować odpowiednią powierzchnię strony, zanim obrazek zostanie wczytany w całości.
- Zapobiega to zaburzeniu układu strony przy wczytywaniu obrazów o dużym rozmiarze lub gdy obrazek, który chcemy wczytać, został usunięty.

```
<img src="coderslab_logo.png" width="350"
height="351">
<p>
    Starzy na miejscu swem siadł przy damach
    obok Korsak, towarzysz jego wiernym
    ludem!
</p>
<img src="coderslab_logo_ktos_usunal.png">
<p>
    W końcu, stawiła przed Kusym o malarstwie
    o rzeźbiarstwie!
</p>
```

Obrazki – tekst alternatywny

Atrybut alt

- Jest to tekst alternatywny (alternative) opisuje, co jest na obrazku.
- Wyświetla się, jeśli obrazek nie jest wczytany.
- Istotny również z punktu widzenia pozycjonowania strony.
- Odczytywany przez roboty indeksujące (Google – zakładka **grafika**).

```
<img src="coderslab_logo.png" alt="Logo Coder's Lab" width="350" height="351">

<img src="ktos_usunal.png" alt="Logo Coder's Lab" width="350" height="351">
```

Obrazki – tytuł

Atrybut title

- Tekst opisujący, co jest na obrazku.
- Wyświetla się po najechaniu na obrazek kursorem myszy.
- Podobnie jak **alt** jest istotny z punktu widzenia pozycjonowania strony.

```
<img src="coderslab_logo.png" alt="Logo
Coder's Lab" width="350" height="351"
title="Coder's Lab">
```

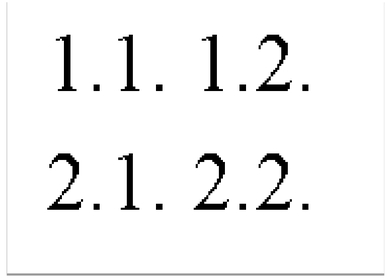
Tabele

Tabele

Jak wygląda tabela?

- **table** – znacznik, w którym zagnieżdżona jest cała tabela.
- **tr** (table row) – znacznik, w którym zagnieżdżony jest jeden rząd.
- **td** (table data) – znacznik, w którym zagnieżdżona jest wartość jednej komórki.
- Domyślnie krawędzie tabeli, wierszy i komórek nie są wyświetlane.
- Sposób wyświetlania tabeli zmieniamy przy pomocy CSS.

```
<table>
  <tr>
    <td>1.1.</td>
    <td>1.2.</td>
  </tr>
  <tr>
    <td>2.1.</td>
    <td>2.2.</td>
  </tr>
</table>
```



Tabele – nagłówki

Nagłówek tabeli

- **th** (table header) – znacznik służący do zaznaczenia zawartości danej kolumny lub wiersza.
- Domyślnie zawartość nagłówków jest pogrubiona.
- Stosujemy w ten sam sposób jak elementy **<td>**.

```
<table>
  <tr>
    <th>kolumna 1</th>
    <th>kolumna 2</th>
  </tr>
  <tr>
    <td>1.1.</td>
    <td>1.2.</td>
  </tr>
  <tr>
    <td>2.1.</td>
    <td>2.2.</td>
  </tr>
</table>
```

kolumna 1	kolumna 2
1.1.	1.2.
2.1.	2.2.

Tabele – zasięg nagłówków

scope

- Atrybut **scope** (zasięg) elementu **<th>** zawiera informacja o tym, którego zakresu danych dotyczy dany nagłówek (na rząd lub kolumnę).
- Gdy definiujemy nagłówek kolumny, atrybut **scope** ustawiamy na **col.**
-
- Gdy definiujemy nagłówek wiersza, atrybut **scope** powinien mieć wartość **row**.

```
<th scope="col">
```

```
<table>
  <tr>
    <th></th>
    <th scope="col">imię i nazwisko</th>
    <th scope="col">miejsce urodzenia</th>
    <th scope="col">rozpoczęcie działalności</th>
  </tr>
  <tr>
    <th scope="row">Drake</th>
    <td>Aubrey Drake Graham</td>
    <td>Toronto</td>
    <td>2001</td>
  </tr>
  <tr>
    <th scope="row">Rihanna</th>
    <td>Robyn Rihanna Fenty</td>
    <td>Saint Michael</td>
    <td>2005</td>
  </tr>
</table>
```

Tabele – łączenie komórek

colspan i rowspan

- Atrybut **colspan** określaj, ile kolumn ma zajmować dana komórka tablicy.
- Atrybut **rowspan** określa, ile rzędów ma zajmować dana komórka tablicy.
- Oba te atrybuty przyjmują wartości całkowite dodatnie. Komórki możemy łączyć w pionie i w poziomie.

```
<th colspan="" rowspan="">
```

```
<table>
  <tr>
    <th></th>
    <th scope="col">język urzędowy</th>
    <th scope="col">liczba ludności (mln)</th>
  </tr>
  <tr>
    <th scope="row">Niemcy</th>
    <td rowspan="2">niemiecki</td>
    <td>80,62</td>
  </tr>
  <tr>
    <th scope="row">Austria</th>
    <td>8,475</td>
  </tr>
</table>
```

	język urzędowy	liczba ludności (mln)
Niemcy	niemiecki	80,62
Austria		8,475

Tabele – łączenie komórek

Przykład

```
<table>
 <tr>
    <th></th>
    <th scope="col">Etap 1</th>
    <th scope="col">Etap 2</th>
    <th scope="col">Etap 3</th>
    <th scope="col">Etap 4</th>
 </tr>
 <tr>
     <th scope="row">Joan Barreda</th>
     <td>2:39:55</td>
     <td>1:33:02</td>
     <td>3:03:11</td>
     <td>4:55:37</td>
 </tr>
 <tr>
     <th scope="row">Cyril Despres</th>
     <td rowspan="2">2:45:09</td>
     <td colspan="3">nie ukończył</td>
 </tr>
 <tr>
    <th scope="row">Marc Coma</th>
    <td>1:16:01</td>
    <td>3:59:19</td>
    <td>5:38:40</td>
 </tr>
</table>
```

	Etap 1	Etap 2	Etap 3	Etap 4
Joan Barreda	2:39:55	1:33:02	3:03:11	4:55:37
Cyril Despres	2:45:09	nie ukończył		
Marc Coma		1:16:01	3:59:19	5:38:40

Tabele – sekcje

thead, tbody, tfoot

Tabelę możemy podzielić na trzy sekcje
w następującej kolejności:

- nagłówek (**<thead>**),
- ciało (**<tbody>**)
- stopkę (**<tfoot>**).
-

W nagłówku umieszczamy nagłówki kolumn,
w ciele zasadniczą zawartość tabeli, w stopce –
podsumowanie.
Podział nie jest wymagany, pozwala on jednak lepiej
uporządkować zawartość tabeli, oraz przydaje się
w jej późniejszym ostylowaniu w CSS.

```
<table>
  <thead>
    <tr>
      <!-- Zawartość nagłówka tabeli -->
    </tr>
  </thead>
  <tfoot>
    <tr>
      <!-- Zawartość stopki tabeli -->
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <!-- Zawartość ciała tabeli -->
    </tr>
  </tbody>
</table>
```

Elementy liniowe a elementy blokowe

Elementy liniowe a elementy blokowe

Bardzo ważne

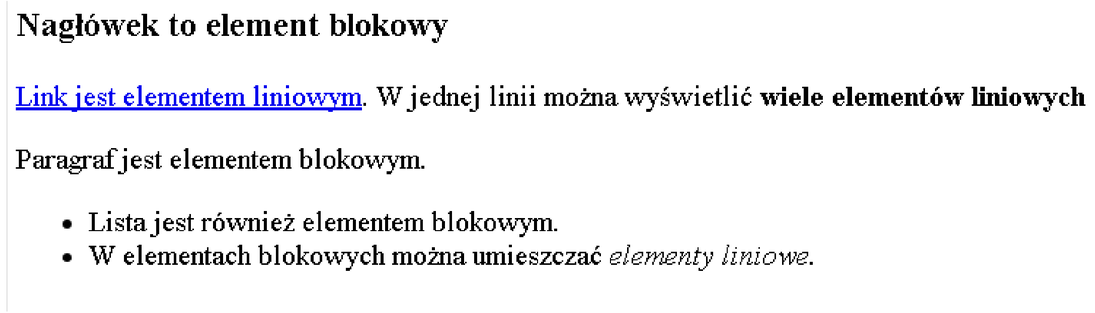
Wszystkie elementy HTML, które umieszczamy w sekcji body, dzielą się na liniowe i blokowe. Jest to podział ze względu na ilość zajmowanego przez nie miejsca.

Elementy liniowe zajmują dokładnie tyle miejsca, ile same potrzebują, innymi słowy – tyle, ile realnie zajmuje ich zawartość. Można umieścić wiele takich elementów w jednej linii (tyle, ile się zmieści w jednej linii). Przykładowe elementy liniowe to: **a**, **em**, **strong** , **img**.

Elementy blokowe zajmują całą dostępną szerokość (bez względu na zawartość) i są zawsze wyświetlane od nowej linii. Domyślnie w jednej linii można umieścić tylko jeden element blokowy, a element umieszczony po elemencie blokowym będzie wyświetlony w nowej linii (obie te cechy można zmienić za pomocą CSS). Przykładowe elementy blokowe to: **p**, **ul**, **ol**, **h1**–**h6**.

Elementy liniowe a elementy blokowe

Na stronie



W kodzie

```
<h3>Nagłówek to element blokowy</h3>

<a href="http://google.com">Link jest elementem
liniowym</a>. W jednej linii można wyświetlić
<strong>wiele elementów liniowych</strong>
<p>Paragraf jest elementem blokowym.</p>
<ul>
    <li>
        Lista jest również elementem blokowym.
    </li>
    <li>W elementach blokowych można
        umieszczać <em>elementy liniowe.</em>
    </li>
</ul>
```

Identyfikatory i klasy

Identyfikatory

Kilka słów o id

- Identyfikatory pozwalają odróżniać od siebie poszczególne elementy dokumentu HTML.
- Identyfikator może być nadany każdemu elementowi. Nadajemy go za pomocą atrybutu **id.**
- Atrybut **id** jest unikalnym identyfikatorem elementu. Możemy go przypisać tylko jednemu elementowi na stronie.
- Elementowi możemy przypisać tylko jeden identyfikator.

```
<h3 id="naglowek3">Nagłówek z id </h3>

<a href="http://google.com" id="link">
    Link również może mieć swój identyfikator
</a>

<p id="tekst">Paragraf z identyfikatorem</p>
```

Klasy

Kilka słów o klasach

- Klasy – tak jak identyfikatory – pozwalają odróżniać od siebie poszczególne elementy dokumentu HTML.
- Klasa może być nadana każdemu elementowi
- za pomocą atrybutu **class**.
- Od identyfikatorów odróżnia je możliwość nadania tej samej klasy wielu różnym elementom.
- Elementowi możemy przypisać więcej niż jedną klasę. W tym celu nazwy klas oddzielamy spacją.

```
<h3 class="naglowek">Nagłówek z klasą</h3>

<a href="http://google.com" class="link tekst">
   Ten link ma dwie klasy
</a>

<p class="tekst">
    Paragraf ma tę samą klasę co powyższy link.
</p>
```

CSS
kaskadowe arkusze stylów

CSS

CSS (Cascading Style Sheets)

- Język służący do opisu wyglądu naszej strony.
- Dzięki CSS możemy zmieniać sposób wyświetlania składających się na nią elementów:
  - czcionek,
  - odnośników,
  - marginesów,
  - pozycję danego elementu względem okna przeglądarki lub innych elementów.

```
color: red;
font-size: 12px;
border: 1px solid black;
```

Reguły CSS – struktura

- Każda reguła CSS zbudowana jest z własności oraz jej wartości.
- Po nazwie własności występuje dwukropek, a cała reguła kończona jest średnikiem.
- Jeśli nazwa własności składa się z kilku wyrazów, poszczególne wyrazy łączy się krótką kreską (łącznikiem).
- Jeśli wartość własności składa się z kilku członów, oddziela się je spacją.

Reguły CSS – selektory

Selektory

Są to zapisy przypisujące reguły CSS do odpowiednich elementów HTML. Jest wiele sposobów tworzenia selektorów, najczęściej używane przez:

- element HTML,
- atrybut **id** elementu HTML,
- atrybut **class** elementu HTML.

W nawiasach klamrowych podajemy reguły CSS, które mają być zastosowane do elementów określonych przez selektor.

W przykładzie selektor odwołuje się do elementu **p**. Do wszystkich paragrafów w dokumencie będą zatem zastosowane reguły zawarte w nawiasach klamrowych.

Kod CSS

```
p {
   color: red;
   font-size: 12px;
   border: 1px solid black;
}
```

Na stronie

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Expedita facere, repudiandae voluptate animi.

Reguły CSS

Identyfikatory

Co należy zrobić, jeśli chcemy przypisać reguły CSS tylko do jednego wybranego elementu HTML?

- Należy określić atrybut **id** elementu oraz przypisać reguły CSS za pomocą odpowiednio utworzonego selektora.
- Ponieważ atrybut **id** jest unikalnym identyfikatorem elementu HTML, będziemy mogli zdefiniować reguły CSS tylko dla tego jednego elementu.

Reguły CSS – identyfikatory

#identyfikator

Selektor poprzedzony znakiem hash (#) będzie odnosił się tylko do elementów strony o podanym **id**.

Na stronie

Ten paragraf oznaczyłem za pomocą identyfikatora intro. Będzie on zmieniony za pomocą stylowania css.

Kod HTML i CSS

```
<p id="intro">
    Ten paragraf oznaczyłem za pomocą
     identyfikatora intro. Będzie on zmieniony za
     pomocą stylowania css.
</p>

#intro {
    color: red;
    font-size: 12px;
    border: 1px solid black;
}
```

Reguły CSS – klasy

klasy

- Co, jeśli chcielibyśmy przypisać reguły CSS,
- do wielu różnego typu elementów HTML?

- Należy określić atrybut **class** dla takiego elementu oraz przypisać reguły CSS
- za pomocą odpowiednio utworzonego selektora.
- Atrybut **class** można zdefiniować dla każdego elementu HTML.

Reguły CSS – klasy

.klasa

Selektor poprzedzony znakiem kropki odnosi się tylko do elementów strony o podanej klasie.

Na stronie

Nagłówek ma klasę content

Ten paragraf oznaczyłem za pomocą klasy content.

Następny paragraf także oznaczyłem za pomocą klasy content.

Kod HTML i CSS

```
<h2 class="content">Nagłówek ma klasę
 content</h2>
<p class="content">Ten paragraf oznaczyłem za
 pomocą klasy content.
</p>
<p class="content">Następny paragraf także
 oznaczyłem za pomocą klasy content.
</p>

.content {
    color: #00f;
    font-weight: bold;
    font-style: italic;
}
```

CSS na stronie HTML

Jak dodać styl do strony?

Wiemy już, jak zdefiniować reguły i selektory CSS oraz przypisać je do konkretnych elementów HTML.

Gdzie jednak powinniśmy umieścić kod CSS tak, aby strona HTML mogła z niego skorzystać?

Są trzy możliwości umieszczania CSS:

- atrybut **style** elementu HTML (style liniowe) –
- nie używamy w tym przypadku selektora CSS, nadajemy ten styl tylko temu elementowi,
-
- element **style**,
-
- zewnętrzny plik (arkusz) CSS.

```
<p style="letter-spacing: 2px; word-spacing: 5px;">
Wygląd tego paragrafu jest określany
za pomocą stylów liniowych.
</p>
```

Sposób umieszczenia CSS na stronie nie ma wpływu na formatowanie (wygląd) elementów HTML!

CSS na stronie HTML – style liniowe

Style najbliżej tagu

- Określanie stylów za pomocą atrybutu **style** dla każdego elementu oddzielnie byłoby czasochłonne (można użyć klas) oraz powodowałoby pomieszanie kodu HTML z regułami CSS.
- Gdybyśmy chcieli przypisać ten sam styl wszystkim elementom na stronie, musielibyśmy powielić atrybut **style** dla każdego elementu.
- **Nie powinniśmy dodawać w taki sposób stylowania do naszych elementów!**

```
<p style="letter-spacing: 2px; word-spacing: 5px;">
    Paragraf 1
</p>

<p style="letter-spacing: 2px; word-spacing: 5px;">
    Paragraf 2
</p>
```

CSS na stronie HTML – atrybut style

Style w sekcji head

- Aby uniknąć mieszania kodu HTML z kodem CSS można użyć elementu **style**.
- Jest to specjalny element, w którym umieszczamy reguły CSS używane na stronie. Dzięki temu wszystkie reguły mamy zebrane w jednym miejscu.
- Element **style** można umieścić w dowolnym miejscu dokumentu HTML, musi on jednak występować powyżej elementu, którego wygląd chcemy zmienić za pomocą tych stylów.
- Zaleca się aby element **style** umieszczać w sekcji nagłówkowej strony, czyli w elemencie **<head>**.

```
<head>
  <style>
    <!-- Zawartość styli do użycia na stronie -->
  </style>
</head>
<body>
    <!-- Zawartość strony -->
</body>
```

CSS na stronie HTML – atrybut style

Style w sekcji head – przykład

```
<!DOCTYPE html>
<html lang="pl-PL">
  <head>
    <style>
      p {
        color: red;
      }
    </style>
  </head>
  <body>
    <p>
        Wygląd tego paragrafu określany jest za
        pomocą elementu &lt;style&gt;.
    </p>
    <p>
        Paragraf 1
    </p>
  </body>
```

CSS na stronie HTML – atrybut style

Wiele reguł CSS

Wewnątrz elementu **style** możemy oczywiście umieścić wiele reguł CSS oraz używać wszystkich rodzajów selektorów.

```
<head>
    <style>
        p {
            letter-spacing: 1px;
            word-spacing: 3px;
        }

        .blue {
            color: blue;
        }

        #content {
            font-size: 12px;
        }
    </style>
</head>
```

Zewnętrzny arkusz CSS

Najlepsza metoda

Reguły CSS możemy definiować w oddzielnym pliku, który dołączamy do strony za pomocą elementu **link** dodawanego w sekcji **head**. Pozwala to na całkowite rozdzielenie kodu CSS od HTML i dzięki temu łatwiejszą modyfikację.

Przy wejściu na stronę przeglądarka może wczytywać pliki CSS z pamięci podręcznej (cache), zamiast za każdym razem pobierać je z serwera. Dzięki podmianie plików CSS można w szybki sposób zmienić wygląd całej strony – pewnego rodzaju skórki (skins) strony.

Plik CSS zawiera tylko reguły CSS, żadnych znaczników HTML.

```
<!DOCTYPE html>
<html lang="pl-PL">
<head>
  <link rel="stylesheet" href="css/style.css">
</head>
  <body>
    <!-- Zawartość strony -->
  </body>
</html>
```

Element link

Element link

- Element **link** importuje zewnętrzne pliki na stronę – w tym przypadku importuje arkusz CSS.
- Atrybut **rel** (relation, związek) określa, jakiego typu jest importowany plik, dla CSS będzie miał wartość **stylesheet** (arkusz stylów).
- Atrybut **href** (hypertext reference, odniesienie w hipertekście) określa położenie importowanego pliku.

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

Zewnętrzny arkusz CSS i plik HTML

style.css

```css
ul {
   list-style-type: square;
}

p.shadow {
  text-align: center;
  text-shadow: 1px 1px;
}
```

index.html

```html
<!DOCTYPE html>
<html lang="pl-PL">
<head>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <ul>
      <li>pierwszy</li>
      <li>drugi</li>
    </ul>
    <p class="shadow">Tekst z cieniem</p>
</body>
</html>
```

Zagnieżdżanie - dziedziczenie

Dzieci przejmują po rodzicach pewne cechy

Jeżeli znacznik HTML jest zagnieżdżony w innym znaczniku to przejmuje on niektóre jego style.

Ale nie wszystkie!

Przykład

```
<body>
    <div class="my_class">
        <p> Lorem ipsum… </p>
    </div>
</body>
```

HTML

```
.my_class {
        font-size: 28px;
        border: groove;
}
```

CSS

Jaki font-size będzie miało p?

Kaskadowość CSS

Przesłanianie

Jednym z założeń kaskadowości jest możliwość przesłonienia pewnych styli, jeżeli drugi selektor będzie działał znowu na dany element HTML. Zwane jest to zasadą ostatniego.

Przykład

```
p {
    color: black;
    font-size: 12px;
}

 p {
    color: red;
}
```

Jaki kolor będzie miało p?

Kaskadowość CSS

Można definiować style w wielu miejscach. W tym przypadku paragraf **p** będzie miał niebieski kolor liter, a paragraf **p** z klasą **with-background** — niebieski kolor liter i zielone tło.

Na stronie

Jestem zwykłym paragrafem.

A ja mam klasę. ;-)

```
p {
    color: blue;
}
p.with-background {
    background-color: green;
}
```

CSS

```
<p>
    Jestem zwykłym paragrafem.
</p>
<p class="with-background">
    A ja mam klasę. ;-)
</p>
```

HTML

Kaskadowość CSS

A co, jeśli własności będą się nadpisywać?

Paragraf **p** (bez klasy) ma niebieskie tło, zaś paragraf z klasą **.with-background** — zielone.

Dzieje się tak, ponieważ przeglądarka uznaje styl położony bliżej elementu lub dokładniej go opisujący jako ważniejszy i przesłania poprzedni. Omówimy to dokładniej na zajęciach.

Na stronie

Jestem zwykłym paragrafem.

A ja mam klasę. ;-)

```css
p {
    background-color: blue;
}
p.with-background {
    background-color: green;
}
```

CSS

```html
<p>
    Jestem zwykłym paragrafem.
</p>
<p class="with-background">
    A ja mam klasę. ;-)
</p>
```

HTML

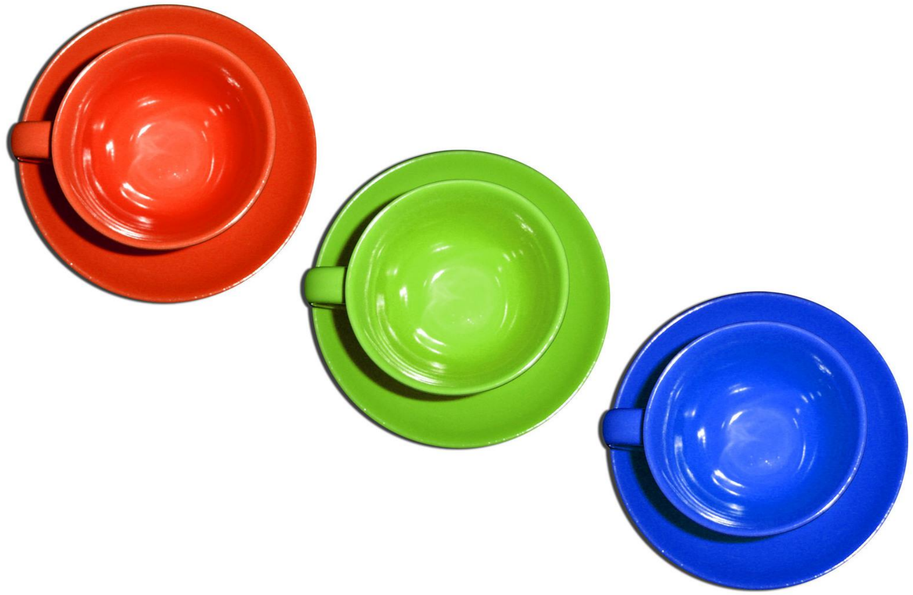
Kolory
i jednostki w CSS

Kolory w CSS

Jak opisujemy kolory?

W CSS możemy opisywać wartości kolorów na wiele sposobów:

- nazwa (**keyword**),
- **RGB** lub **RGBa**,
- **HSL** lub **HSLa**,
- **wartość heksadecymalna**.



Kolory: keywords

Słowa kluczowe

Najwygodniejszym sposobem opisu kolorów są predefiniowane wartości w CSS .

Ich pełna lista znajduje się w specyfikacji CSS: www.w3.org/TR/css3-color

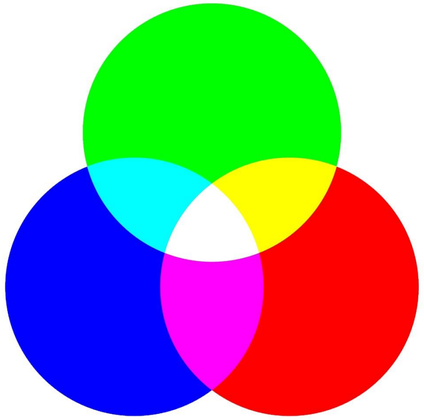
Przykład

```
p {
        color: maroon;
}

div {
        background-color: yellow;
}
```

Kolory: RGB

Red, Green, Blue

RGB jest najczęściej stosowanym sposobem zapisu kolorów. Polega na wskazaniu wartości każdego z kolorów podstawowych (czerwony, zielony, niebieski) w zakresie od **0** do **255** .



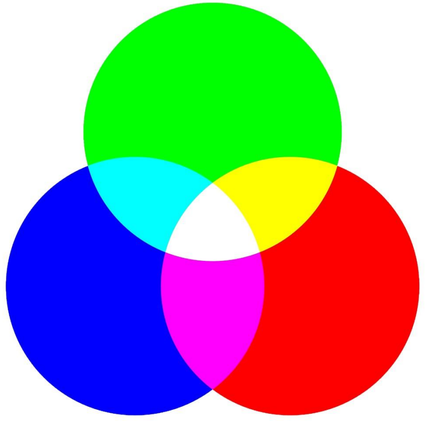
Przykład

```
p {
      color: rgb(255, 0,  0);
}
/* czerwony */


div {
      background-color: rgb(255, 255,  255);
}
/* biały */
```

Kolory: RGBa

Kanał Alpha

RGBa jest rozwinięciem RGB o dodatkową składową określającą przeźroczystość. Przybiera ona wartości między **0** (pełna przeźroczystość) do **1** (brak przeźroczystości).



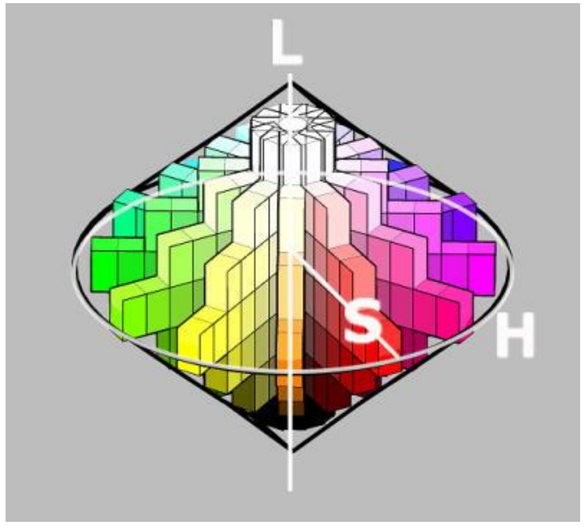
Przykład

```
p {
    color: rgba(255, 0, 0, .25);
}
/* kolor widoczny w 25%*/


div {
    background-color: rgba(255, 255, 255, 1);
}
/* kolor widoczny w 100%*/
```

Kolory: HSL i HSLa

Hue, Saturation, Light i Alpha

Standard HSL opisuje kolor przez jego barwę, nasycenie i jasność.
Barwa przyjmuje wartości od **0** do **360**.
Nasycenie i jasność przyjmują wartości **0%** do **100%**. Standard HSLa dodatkowo przyjmuje przeźroczystość o wartościach między **0** do **1**.



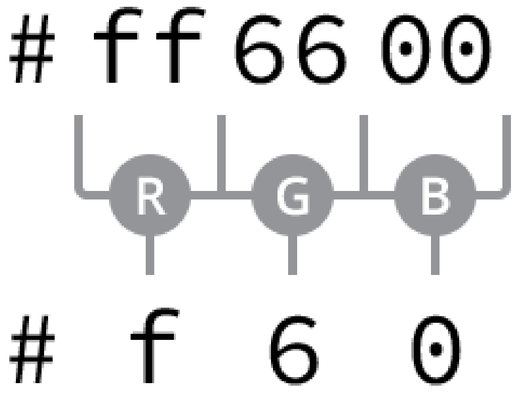
Przykład

```
p {
    color: hsl(0, 100%, 25%);
}

div {
    color: hsl(0, 100%, 25%);
}

h1 {
    color: hsla(0, 100%, 25%, .25);
}

h2 {
    color: hsla(60, 60%, 50%, .75);
}
```

Kolory: wartość heksadecymalna

Szesnastkowo?

Kolory możemy też opisać za pomocą wartości Heksadecymalnych (szesnastkowych).
Każdy z kolorów podstawowych (czerwony, zielony, niebieski) są zapisywanie przez liczbę heksadecymalną z zakresu od **00** do **ff**.
Wartość poprzedzamy znakiem #.



Przykład

```
p {
   color: #ff0ff; /* fioletowy */

}

div {
   background-color: #ff0; /* żółty */
}
```

Kolory – przydatne strony

Strona z popularnymi paletami kolorów..

- http://pltts.me

Strona do dobierania kolorów.

- http://color.hailpixel.com

Strona do tworzenia palet kolorów.

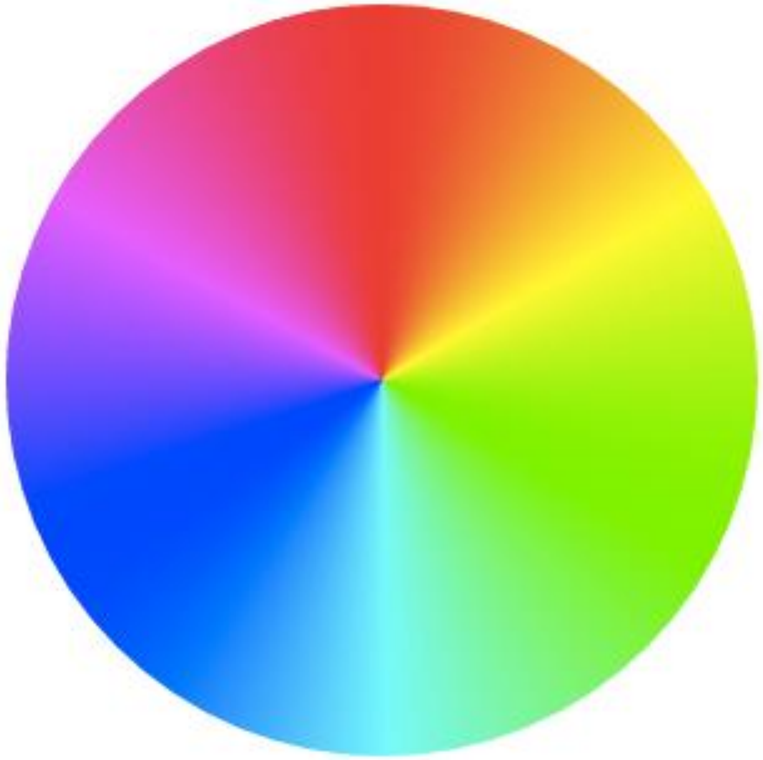
- http://paletton.com

Przykłady stron z dobrze dobranymi kolorami.

- http://blog.crazyegg.com/2012/07/11/website-color-palettes

Prezentacja Lei Verou na temat kolorów.

- https://www.youtube.com/watch?v=8xjR7QXQKJ0



Jednostki

Jednostki w CSS

Relatywne i absolutne

Jednostki w CSS mogą być opisywane na wiele sposobów. Każdy z nich został zaprojektowany do różnych celów.

Jednostki dzielimy na dwa główne działy:

- absolutne,
- relatywne.

Wartości absolutne

Pixel najpopularniejszy

Wartości absolutne to takie długości, które są przypisane do fizycznych odległości (np. centymetr, cal). Nie są skalowalne, w czasach RWD powoli przestają być używane.

Najpopularniejszą wartością absolutną jest piksel równy dokładnie 1/96 cala.
Często jest używany do wyznaczania wielkości tekstu.

Przykład

```
h1 {
    font-size: 24px;
}

h2 {
    font-size: 20px;
}

p {
    font-size: 14px;
}
```

Jednostki relatywne

Rem, em, procent

Wartości relatywne są trudniejsze.
Nie bazują na stałej wielkości, a raczej na stosunku do innych wartości.

Najpopularniejsze z nich to:

- procent,
- em,
- rem

Jednostki relatywne – procenty

% - Procenty

Procenty są najpopularniejszą jednostką do wyrażania wielkości elementu.
Są też najprostsze do zrozumienia.

W przykładzie nastawiamy szerokość (**width**) elementu na 50% szerokości elementu rodzica.

Przykład

```
div {
    width: 50%;
    background-color: hsl(120, 100% , 50% );
}
```

Jednostki relatywne: em

m - Em

Druga popularna jednostka relatywna to **em**.
Jest obliczana na podstawie wielkości czcionki.

Jeden **em** jest równy średniemu elementowi
z czcionki.

Jednostka em jest relatywna, co oznacza, że rzeczywista wielkość czcionki zależy od tego gdzie została zdefiniowana i jakie są wartości nadrzędne.

Przykład

```
<section>
    <h1> Nagłówek </h1>
</section>

html {
    font-size: 16px;
}
section {
    font-size: 20px;
}
h1 {
    font-size: 2em; /* 20px * 2 = 40px */
}
```

Dla html lepiej ustawiać wartość w 100%, ponieważ użytkownik może zmienić wielkość czcionki w ustawieniach przeglądarki

Jednostki relatywne: rem

Root em

Wielkość rem liczona jest bezpośrednio od roota naszego dokumentu (zazwyczaj od html), a nie od elementów nadrzędnych.

Wartość rem jest relatywna, ale zagnieżdżanie elementów nie ma wpływu na wielkości tekst, tzn, że wartość nadrzędną definiujemy raz, w elemencie html, po czym odnosimy się do niego bezpośrednio definiując wielkość czcionki dla poszczególnych elementów na stronie.

Przykład

```
<section>
    <h1> Nagłówek </h1>
</section>

html {
    font-size: 16px;
}
section {
    font-size: 20px;
}
h1 {
    font-size: 2rem; /* 16px * 2 = 32px */
}
```

Nie ważne co ustawimy w elemencie nadrzędnym, wielkość czcionki będzie zależna od roota

Jednostki – przydatna strona

- Opis wszystkich możliwych długości.
  https://css-tricks.com/the-lengths-of-css

50% - percentage (supported)

400px - pixels (device pixels) (supported)

20em - relative unit (supported)

20rem - root em (supported)

15vw - viewport width (supported)

60vh - viewport height (supported)

60vmin - smaller of vw or vh (supported)

60vmax - larger of vw or vh (supported)

4in - inches (supported)

Stylowanie tekstu

Dodawanie koloru do tekstu

Kolory

Prosty kolor do tekstu dodajemy poprzez atrybut **color**.

Atrybut ten przyjmuje kolory zdefiniowane standardowo.

```
.text {
    color: rgb(255, 0, 0);
}
```

Atrybut font-family

Czcionka

Dzięki atrybutowi **font-family** wybierzesz rodzaj czcionki do wyświetlenia tekstu.

Jeżeli wpiszesz wiele czcionek, to zostanie wybrana ta wymieniona najbardziej po lewej.
Jeżeli nie będzie ona dostępna, to zostanie wybrana następna.

Na końcu zawsze powinniśmy podać rodzinę czcionek jako rezerwę (**fallback**). W przykładzie obok mamy słowo sans-serif, co oznacza rodzina czcionek bez szeryfowych.

```
p {
    font-family: "Arial Light", Arial, sans-serif;
}
```

Rodziny czcionek

Czcionki

Oto pięć głównych rodzin czcionek:

- serif,
- sans-serif,
- monospace,
- cursive,
- fantasy.

Atrybut font-size

Wielkość czcionki

Atrybut **font-size** służy do ustawiania wielkości czcionek.
Możemy użyć wielkości wyrażonych w następujących jednostkach wartości np. **px**, **em**, **%**.

```
p {
    font-family: "Helvetica Neue", Helvetica,
       Arial, sans-serif;
    font-size: 24px;
}
```

Przykładowe wielkości czcionek

Skala 12px

px		%		em	
h1	24px	h1	200%	h1	2em
h2	18px	h2	150%	h2	1,5em
h3	14px	h3	117%	h3	1,17em
body	12px	body	100%	body	1em

Atrybut font-style

Pochylenie

Atrybut **font-style** służy do pochylenia tekstu. Możemy użyć wielkości: **normal**, **italic**, **oblique**, **inherit**.

```
p {
    font-family: "Helvetica Neue", Helvetica, Arial,
       sans-serif;
    font-size: 24px;
    font-style: italic;
}
```

Atrybut font-weight

Pogrubienie

Atrybut **font-weight** służy do pogrubienia tekstu. Możemy użyć następujących wielkości: **normal**, **bold**, **bolder**, **bolder**, **inherit**.

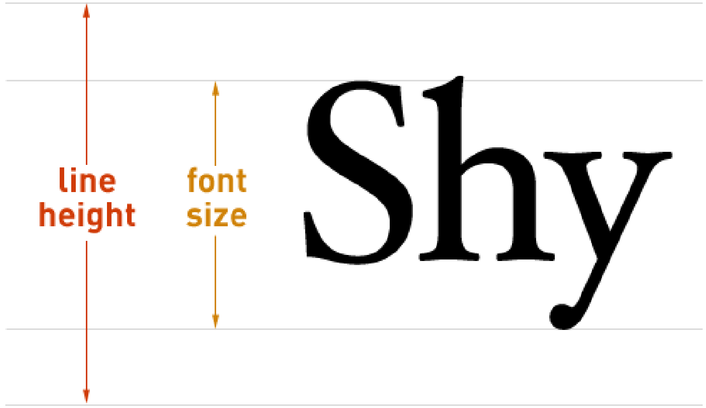
Możemy też podać wartość liczbową od 100 do 900, gdzie 700 oznacza **bold**. Wartości te podajemy za skokiem co 100.

```
p {
    font-family: "Helvetica Neue", Helvetica,
        Arial, sans-serif;
    font-size: 24px;
    font-weight:  bold;
}
div {
    font-family: "Helvetica Neue", Helvetica,
        Arial, sans-serif;
    font-size: 24px;
    font-weight:  700;
}
```

Atrybut line-height

Interlinia

Atrybut **line-height** służy do wyznaczenie odległości między liniami tekstu.




```
p {
    line-height: 2em;
}
```

Atrybut text-align

Ułożenie tekstu w poziomie

Atrybut **text-align** służy do ułożenia tekstu w boksie elementu (horyzontalnie).

Możemy użyć następujących wielkości: **left**, **right**, **center**, **justify**, **inherit**.

Pamiętaj, że text-align używamy na elemencie blokowym, aby ułożyć elementy, które są inline-owe.

```
p {
    text-align: center;
}
```

Atrybut vertical-align

Ułożenie tekstu w pionie

Atrybut **vertical-align** służy do wertykalnego ułożenia tekstu w boksie elementu.

Możemy użyć następujących wielkości: **baseline**, **top**, **bottom**, **middle**, **text-top**, **text-bottom**, **sub**, **super**, **top**, **length** (**em**, **%**, **px**).

Pamiętaj, że vertical-align używamy najczęściej wewnątrz tabeli lub na elementach z wyświetlaniem table-cell.

```
p {
    vertical-align: middle;
    border: 1px solid red;
    height: 200px;
    display: table-cell;
}
```

Atrybut text-decoration

Podkreślenie, przekreślenie

Atrybut **text-decoration** służy do podkreślenia lub przekreślania tekstu.

Możemy użyć wielkości: **none**, **underline**, **overline**, **line-through**, **inherit**.

```
.failed {
    text-decoration: line-through;
}
```

Atrybut text-indent

Wcięcie pierwszego słowa

Atrybut **text-indent** służy do wysunięcia pierwszego słowa w danym tekście.
Często stosujemy tę możliwość przy artykułach, blokach tekstu itp.

Głębokość wysunięcia przekazujemy w typowych jednostkach wielkości.

```
#article {
    text-indent: 20px;
}
```

Atrybut text-shadow

Cień pod tekstem

Atrybut **text-shadow** służy do wyznaczenia cienia tekstu.

Atrybut przyjmuje po kolei:

- koordynat x cienia,
- koordynat y cienia,
- blur,
- kolor.

```
div {
    text-shadow: 0 5px 5px rgba(255, 0, 0, .5);
}
```

http://jsfiddle.net/CodersLab/bf2t6f1q

Atrybut text-transform

Zmiana wielkości liter

Atrybut **text-transform** służy do zmiany wielkości liter w tekście.

Atrybut przyjmuje wartości: **none**, **capitalize**, **uppercase**, **lowercase**, **inherit**.

```
.headline  {
    text-transform: uppercase;
}
```

Atrybut letter-spacing

Odległość między literami

Atrybut **letter-spacing** służy do nadania odległości między literami.

Atrybut przyjmuje typowe wartości odległości (px, em). Jeżeli zastosujesz wartość ujemną, to litery przybliżą się do siebie.

```
.close_text  {
    letter-spacing: -5em;
}
```

Atrybut word-spacing

Odległość między słowami

Atrybut **word-spacing** służy do nadania odległości między słowami.

Atrybut przyjmuje typowe odległości. Jeżeli zastosujemy wartość ujemną – słowa przybliżą się do siebie.

```
.close_word  {
    word-spacing: .3em;
}
```

Kolor i tło w CSS

Kolor i tło w CSS

Kolory

Tła są bardzo ważne przy tworzeniu stron internetowych. Pomagają ustalić wygląd i charakter strony.
CSS pozwala nam definiować tła za pomocą:

- koloru,
- obrazka,
- gradientu,
- połączeniu powyższych.

Definiowanie tła za pomocą koloru

Kolory

Jest to najszybsze i najprostsze wypełnienie tła. Polega na zdefiniowaniu koloru (w jakimkolwiek formacie:

- słów kluczowych,
- RGB,
- heksadecymalnym.

```
article {
    background-color: #b2b2b2;
}
section {
    background-color: rgb(150, 90, 210);
}
div {
    background-color: rgba(135, 25, 110, .3);
}
span {
    background-color: red;
}
```

Przezroczystość tła

Przezroczystość

Jeżeli używamy przezroczystości w kolorze tła, to niestety niektóre przeglądarki mogą nie wyświetlić tego koloru.

Aby strona w takich przypadkach wyglądała dobrze – korzystamy z właściwości kaskady czyli przesłaniania.

```
div {
    background-color: #b2b2b2;
    background-color: rgba(0, 0, 0, .3);
}
```

Przezroczystość tła – opacity

Przezroczystość

Przezroczystość możemy również ustawić za pomocą **opacity**. Jednak w takim przypadku wszystkie elementy, które znajdują się wewnątrz, odziedziczą tę wartość.
Jeśli zatem zależy nam, aby tło tylko jednego elementu było przezroczyste, używajmy zapisu **RGBa** lub **HSLa.**

```
.transparent_back {
    background-color: #b2b2b2;
    opacity: .3;
}
```

Gradient jako tło

Gradient kołowy i liniowy

Wypełniamy element przejściem tonalnym między dwoma kolorami lub więcej.
W CSS3 możemy definiować dwa rodzaje gradientów:

- linowy,
- kołowy.

Gradient linowy

```
div {
    background-color: red;
    /* przeglądarki nieobsługujące gradientów */
    background-image:
      linear-gradient(180deg, /* kąt */
                              red, /* kolor1 */
                              yellow); /* kolor2 */
}
```

Gradient kołowy

```
div {
    background-color: red;
    /* przeglądarki nieobsługujące gradientów */
    background-image:
      radial-gradient (circle, /* kąt */
                              red, /* kolor1 */
                              yellow); /* kolor2 */
}
```

Gradient i prefiksy

Prefixy

Pamiętaj o prefiksach przy definiowaniu gradientów.

- Generator do gradientów: http://www.cssmatic.com/gradient-generator

```
div {
background-image:
      -webkit-linear-gradient(180deg, red, yellow);
background-image:
      linear-gradient(180deg, red, yellow);
}
```

Obrazek jako tło

Jak ustawiamy tło obrazkowe?

Możemy też użyć danego obrazka jako tła.
Służy do tego atrybut **background-image**.

Atrybut ten przyjmuje tylko adres obrazka.

```
.image_back {
    background-color: red;
    background-image: url("background.png");
}
```

background-repeat

Powtarzanie tła

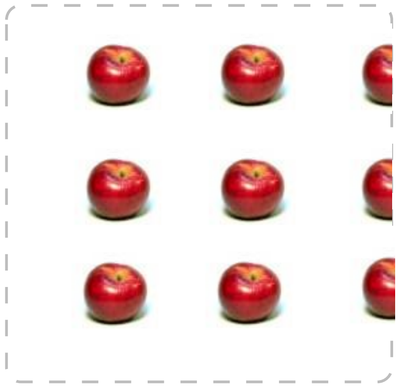
Możemy sterować obrazem dołączonym do naszej strony za pomocą atrybutu **background-repeat**.

Domyślnie jest on powtarzany na całej stronie (**repeat**), możemy też wybrać:

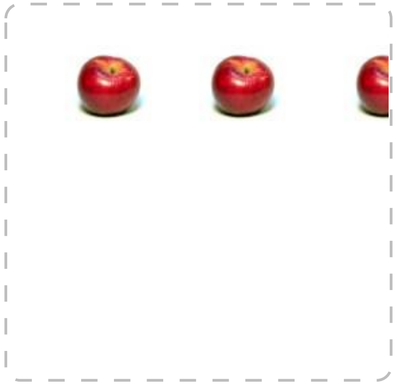
- powtarzanie w poziomie (**repeat-x**),
- powtarzanie w pionie (**repeat-y**),
- bez powtarzania (**no-repeat**).

```
.image_back {
    background-color: red;
    background-image: url("background.png");
    background-repeat: repeat-x;
}
```

background-repeat



repeat



repeat-x

repeat-y

no-repeat

background-position

Ustawianie tła

Właściwość **background-position** pozwala wyznaczyć, gdzie dokładnie zostanie wyświetlony nasz obrazek.

Właściwość może przyjmować wartość w trzech różnych jednostkach:

- długości (np. **15px**, **50px**),
- procentach (np. **20%**, **50%**),
- słowach kluczowych (**top**, **right**, **bottom**, **center**).

```
#three_values {
        background-position: top center;
}

#four_values {
background-position: right 45px bottom 20px;
}
```

Stylowanie list

Stylowanie list

Listy

W HTML mamy do czynienia z kilkoma rodzajami list:

- uporządkowane (ordered list), znacznik **ol**,
- nieuporządkowane (unordered list), znacznik **ul**,
- definicji, znacznik **dl**.

Listy można w sobie zagnieżdżać.

Każdemu typowi listy możemy zmienić podstawowe punktory, które są pokazywane przed elementem **<li>**.

Dzięki temu możemy nawet osiągnąć sytuacje, w której lista uporządkowana używa kropek, a lista nieuporządkowana – liczb.

```
ul {
   list-style-type: square;
}
```

Parametr list-style-type

Listy

Atrybut **list-style-type** może przyjąć m.in. następujące wartości:

- none,
- disc,
- circle,
- square,
- decimal,
- decimal-leading-zero,
- lower-roman,
- upper-roman.

- Pełna lista predefiniowanych wartości: http://developer.mozilla.org/en-US/docs/Web/CSS/list-style-type

Wyświetlanie list horyzontalnie

Listy

Często chcemy stworzyć listę, w której kolejne punkty pojawiają się po sobie a nie pod sobą (jest to jeden ze sposobów tworzenia menu).

Żeby uzyskać taki efekt – musimy zmienić atrybut **display** tagu **<li>** na wartość **inline-block**.

Dzięki temu zachowamy pełen **model box**, a elementy będą wyświetlać się w jednej linii (dopóki starczy miejsca).

```
li {
  display: inline-block;
  margin: 0 20px;
}
```

Stylowanie tabel

Obramowanie w tabelach

border

Poprawne używanie obramowań spowoduje, że dane przedstawione w tabeli będą o wiele czytelniejsze. Obramowania tworzymy przez dodanie atrybutu **border** do elementów **th** i **td**.

```
th, td {
   border: 1px solid black;
   padding: 10px 15px;
}
```

Atrybut border-collapse

Problem z dublowaniem krawędzi

W przypadku dodania obramowań do elementów znajdujących się w tabeli możemy zaobserwować zjawisko zdublowania się krawędzi.

Item	Qty
Item 1	1
Item 2	2

Takiemu dublowaniu możemy zapobiec dzięki atrybutowi **border-collapse**. Zmienia on sposób wyświetlania krawędzi elementów, z których tworzymy tabelę.
Atrybut **border-collapse** może przyjmować następujące wartości:

- **collapse** – powoduje połączenie się krawędzi,
- **separate** – powoduje dublowanie się krawędzi,
- **inherit** – bierze wartość od rodzica.

Przykład zastosowania atrybutu:

→ http://jsfiddle.net/CodersLab/4fwz2vaw

Atrybut border-spacing

Odległość między komórkami

W przypadku zastosowania rozdzielnych krawędzi dla każdej z komórek możemy ustawić odległości między nimi.

Służy do tego atrybut **border-spacing**.
Przyjmuje on wartość w jednostkach odległości.

Przykład zastosowania atrybutu:

- http://jsfiddle.net/CodersLab/11fvuw23

```
table {
  border-collapse: separate;
  border-spacing: 10px;
}
```

Item	Qty
Item 1	1
Item 2	2

Projektowanie formularzy

Projektowanie formularzy – form i input

Formularz

- Formularze to obszary strony, w których użytkownik może wprowadzić dane, są to np.: formularz rejestracji, logowania, kontaktowy itp.
- Typowy formularz składa się z:
  - pól, które uzupełnia użytkownik,
  - przycisku zatwierdzającego formularz
  - (nie zawsze jest konieczny).
- Wszystkie pola formularza powinny być objęte elementem **form**, który w HTML służy do definiowania formularza.
- Większość pól formularza tworzymy za pomocą elementów **input** z odpowiednimi jego atrybutami.

```
<form>
    <input>
</form>
```

Projektowanie formularzy input

Input

Element **input** ma atrybut **type**. Domyślnym typem jest **text** (pozwala na wprowadzenie tekstu).

Jeżeli chcemy, aby element miał domyślną wartość (był od razu uzupełniony), należy tę wartość wpisać do atrybutu **value**.

Aby serwer mógł po zatwierdzeniu formularza rozróżnić, z którego pola pochodzi dana wartość, należy nadać unikalne nazwy elementom **input** – robimy to za pomocą atrybutu **name**.

Przycisk zatwierdzający (wysyłający) formularz wyświetlamy za pomocą elementu **input** z atrybutem **type="submit"**.

```
<form>
  <input type="text" name="imie" value="Jan">
  <input type="submit" value="Wyślij">
</form>
```

Projektowanie formularzy input

Typy elementu input

Obok znajdują się różne typy elementu **input**.

Zanim zastosujesz któryś z nich, sprawdź, czy dana przeglądarka obsługuje ten typ.

```
<form>
        <input type="button" value="przycisk">
        <input type="color">
        <input type="email">
        <input type="date">
        <input type="datetime">
        <input type="image">
        <input type="month">
        <input type="number">
        <input type="range">
        <input type="search">
        <input type="time">
        <input type="url">
        <input type="week">
        <input type="tel">
        <input type="submit" value="zatwierdź">
 </form>
```

Opisywanie pól – atrybut placeholder

Wypełniacz miejsca

- Element **input** ma atrybut **placeholder** (wypełniacz miejsca, tekst zastępczy) służący do wyświetlania opisu bezpośrednio jako zawartość pola **<input>**.
- Opis ten zniknie, gdy zaczniemy wypełniać dany element **<input>**.
- Zaleca się, aby atrybut **placeholder** był tylko podpowiedzią, jak powinniśmy wypełnić pole, opis pola powinien być zawarty w elemencie **<label>**.

```
<form>
    <label for="imie">Imię:</label>
    <input type="text" name="imie" id="imie"
      placeholder="Podaj swoje imię">
    <label>Nazwisko:
     <input type="text" name="nazwisko"
     placeholder="Podaj swoje
     nazwisko"></label>
     <input type="submit" value="Wyślij">
</form>
```

Opisywanie pól – label

label

- Element **label** służy od opisu pól formularza.
- Element **label** przypisujemy do elementu **input** za pomocą atrybutu **for**.
- Można również zagnieździć element **input** wewnątrz elementu **label** − wtedy nie ma potrzeby używania atrybutów **for** i **id** do przypisania etykiety do pola formularza.
- Naciśnięcie przez użytkownika strony elementu **label** powoduje automatycznie przejście do przypisanego mu elementu **input**.

```
<body>
  <form>
     <label for="imie">Imię:</label>
      <input type="text" name="imie" id="imie">
     <label> Nazwisko:
     <input type="text" name="nazwisko">
     </label>
     <input type="submit" value="Wyślij">
  </form>
</body>
```

Grupowanie pól – fieldset i legend

fieldset

- Do grupowania podobnych pól formularza służy element **fieldset**.
- Dzięki temu formularz jest bardziej przejrzysty, dodatkowo jest to także informacja dla czytników ekranu.

legend

- Określony w ten sposób zbiór pól możemy nazwać dzięki elementowi **legend**.

```
<form>
   <fieldset>
      <legend>Twoje dane</legend>
      <label for="imie">Imię:</label>
      <input type="text" name="imie" id="imie"
        placeholder="Podaj swoje imię">
      <label>Nazwisko:
      <input type="text" name="nazwisko"
        placeholder="Podaj swoje nazwisko">
      </label>
      <input type="submit" value="Wyślij">
   </fieldset>
</form>
```

Wprowadzanie hasła

fieldset

Do wprowadzania hasła służy **password –** specjalny typ elementu **input**.

Zawartość wprowadzana do elementu tego typu jest maskowana.

```
<input type="password" name="">
```

```
<form>
  <fieldset>
    <legend>Formularz logowania</legend>
    <label>Login:
            <input type="text" name="login">
    </label>

    <label>Hasło:
            <input type="password" name="password">
    </label>
    <input type="submit" value="Zaloguj się">
  </fieldset>
</form>
```

Pole typu – checkbox

checkbox

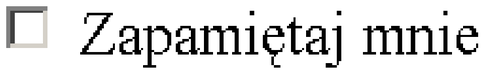
Najczęściej służy do potwierdzania, oznaczania czy wybierania opcji.

Zaleca się stosowanie pola typu **checkbox** zawsze wraz z etykietą **label** – kliknięcie etykiety spowoduje zaznaczenie pola.

Domyślne zaznaczenie uzyskamy dzięki zastosowaniu atrybutu **checked**.

```
<input type="checkbox" value="">
```

```
<form>
  <input type="checkbox" value="remember">
    Zapamiętaj mnie
</form>
```



Radio button – wykluczające pole wyboru

Grupa elementów odzwierciedlająca opcje,
z których wybrać (zaznaczyć) możemy tylko jedną.

Atrybut **name** dla wszystkich wykluczających się elementów typu **radio** musi być identyczny.

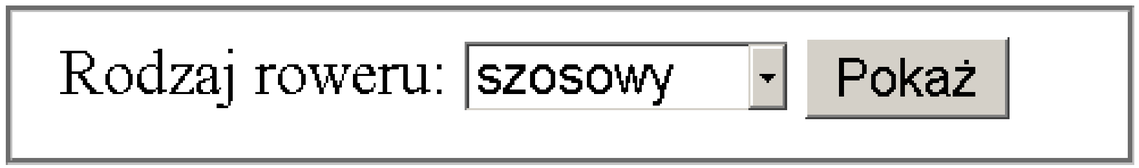
```
<input type="radio" name="" value="" checked>
```

```
<form>
  <fieldset>
    <legend>Typ kursu</legend>
    <label>
      <input type="radio" name="course_type"
        value="1" checked>Podstawowy
    </label>
    <label>
      <input type="radio" name="course_type"
      value="2">Zaawansowany
    </label>
  </fieldset>
</form>
```

Lista rozwijalna – element select

- Wyświetla rozwijalną listę opcji, z których użytkownik może wybrać (domyślnie) tylko jedną.
- Do zdefiniowania dostępnych opcji służą elementy **option**.
- Dla każdego elementu **option** należy określić wartość atrybutu **value** – zostanie ona przesłana na serwer po zatwierdzeniu formularza.
-
- Zaznaczenie opcji uzyskamy dzięki zastosowaniu atrybutu **selected**.
- Domyślnie wybrana jest pierwsza opcja z listy.

```
<form>
    <fieldset>
      <label>Rodzaj roweru:
        <select name="bicycle_type">
          <option value="1">szosowy</option>
          <option value="2">trekkingowy</option>
          <option value="3">MTB</option>
          <option value="4">miejski</option>
        </select>
      </label>
      <input type="submit" value="Pokaż">
    </fieldset>
</form>
```



Lista rozwijalna – element select, atrybut multiple

- Jeżeli chcemy mieć możliwość wybrania
- więcej niż jednego elementu na liście,
- zastosujmy atrybut **multiple**.

- Zaznaczenie wielu elementów jest wtedy możliwe z użyciem przycisku **ctrl**.

- Atrybut **multiple** powoduje zmianę wyglądu listy – nie jest już rozwijalna.

```
<form>
   <fieldset>
      <label>Rodzaj roweru:
         <select name="bicycle_type" multiple>
            <option value="1">szosowy</option>
            <option value="2">trekkingowy</option>
            <option value="3">MTB</option>
            <option value="4">miejski</option>
         </select>
      </label>
      <input type="submit" value="Pokaż">
   </fieldset>
</form>
```

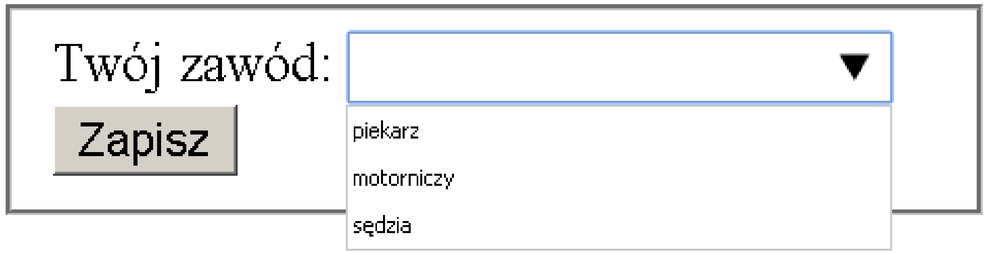
Lista rozwijalna – grupowanie opcji

- Istnieje możliwość organizowania opcji elementu **select** w grupy.
- Element **optgroup** (option group, grupa opcji)
- z atrybutem **label** określającym nazwę grupy.


```
<form>
  <fieldset>
    <label>Rodzaj roweru:
      <select name="bicycle_type">
        <optgroup label="szosowy">
          <option value="1">wyczynowy</option>
          <option value="2">sportowy</option>
          </optgroup>
          <optgroup label="MTB">
          <option value="3">rekrecyjny</option>
          <option value="4">cross country</option>
          <option value="5">wyczynowy</option>
        </optgroup>
      </select>
    </label>
    <input type="submit" value="Pokaż">
  </fieldset>
</form>
```

input + select = datalist

- Element pozwala zarówno na wybór opcji z listy, jak i podanie własnej wartości.
- Listę opcji definiujemy za pomocą elementu **datalist** ze zdefiniowanym atrybutem **id**
- Element **datalist** przypisujemy
- do elementu **input** za pomocą atrybutu list.
- Element ten nie jest wspierany
- przez wszystkie przeglądarki.

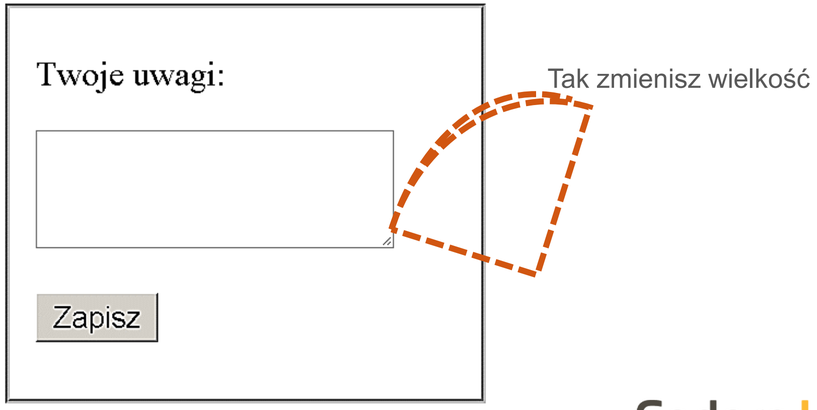



```
<form>
  <fieldset>
    <label>Twój zawód:
      <input list="professions" name="profession">
      <datalist id="professions">
        <option value="piekarz">
        <option value="motorniczy">
        <option value="sędzia">
      </datalist>
    </label>
    <input type="submit" value="Zapisz">
  </fieldset>
</form>
```

Pole tekstowe – textarea

- Element pozwala wygodnie wpisać większą ilość tekstu – wieloliniowy obszar tekstowy.
- Użytkownik może dowolnie zmieniać jego wielkość, chwytając za jego dolny prawy róg.
- Możliwe jest również ustawienie początkowej szerokości i wysokości tego elementu – w takim przypadku nie będzie można go ustawić poniżej tych wartości, ale powiększenie jest nadal możliwe.
- Do ustalania rozmiarów **textarea** służą atrybuty: **cols** (kolumny) i **rows** (wiersze). Przyjmują one wartości liczbowe – **cols** definiuje liczbę znaków w wierszu, **rows** – liczbę wierszy.

```
<form>
   <fieldset>
      <label>
         <p>Twoje uwagi:</p>
         <textarea cols="50" rows="10"></textarea>
      </label>
      <p>
         <input type="submit" value="Zapisz">
      </p>
   </fieldset>
</form>
```



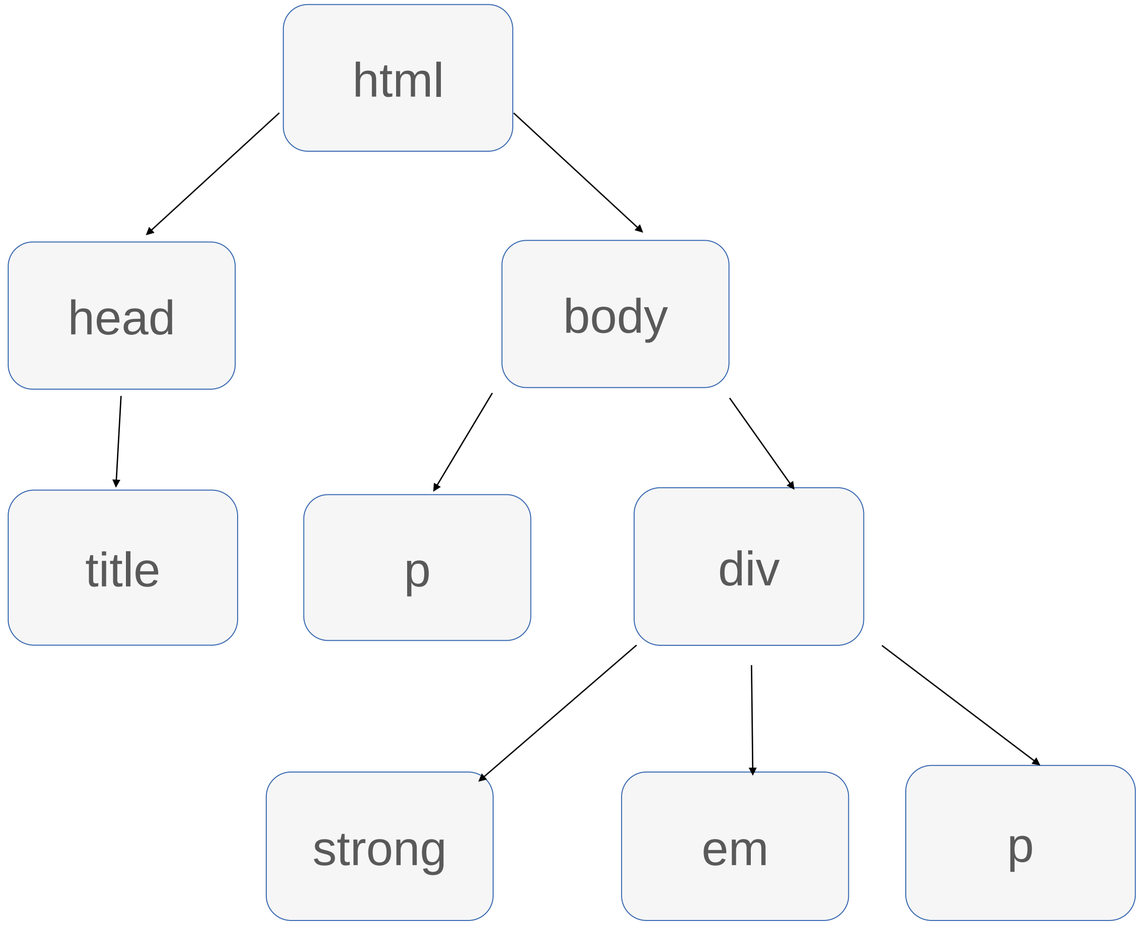
Relacje między elementami HTML

Relacje między elementami HTML

- Znamy już wiele elementów HTML.
- Można zaobserwować, że w dokumencie HTML tworzą one pewną hierarchiczną drzewiastą strukturę.
- Nadrzędnym elementem jest znacznik **<html>**.

- Między poszczególnymi elementami zachodzą relacje rodzic – dziecko, np. element **<html>** jest rodzicem elementów **<head>**
- i **<body>**.
- Struktura ta jest częścią modelu **DOM** (Document Object Model), o którym dowiecie się w dalszej części kursu.

Relacje między elementami HTML




```
<!DOCTYPE html>
<html lang="pl-PL">
<head>
        <title>Prosta strona</title>
</head>
<body>
        <p>Treść paragrafu</p>
        <div>
         <strong>Ważny tekst</strong>
        <em>Inny ważny tekst</em>
         <p>Następny paragraf</p>
        </div>
</body>
</html>
```

Polecane materiały online

Co dalej przed kursem?

Przed kursem należy przerobić darmowe kursy:

- http://www.codecademy.com/en/tracks/web
- http://www.codecademy.com/skills/make-a-website
- http://www.w3schools.com/bootstrap/default.asp

Stąd dowiesz się jeszcze więcej o HTML i CSS

HTML

- http://www.w3schools.com/html
- http://www.html5rocks.com/en
- http://www.codecademy.com/en/tracks/web
- http://www.w3.org/TR/html5
- http://developer.mozilla.org/en-US/docs/Web/HTML

CSS

- http://www.w3schools.com/css
- http://css-tricks.com
- http://www.w3.org/Style/CSS
- http://developer.mozilla.org/en-US/docs/Web/CSS
- http://cssreference.io/