Zaawansowany HTML i CSS
Dzień 1

v3.1

Plan

- Wstęp
- Identyfikatory i klasy
- Relacje między elementami HTML
- HTML5 - nowe elementy
- CSS - kaskadowe arkusze stylów
- Kolory i jednostki

Wstęp

Quiz na początek

Quiz na początek

- Jak zapisujemy komentarze w HTML?
- Co to są atrybuty?
- Jak wygląda podstawowa struktura strony?
- Jakie znacie tagi do formatowania tekstu?
- Jakie mamy listy w HTML?
- Jak tworzymy linki w HTML?
- Jakich znaczników używamy do tworzenia tabel w HTML?
- Na czym polega kaskadowość w CSS?

Przydatne linki

- 300 darmowych rzeczy dla programisty - strony. http://bit.ly/28SLo1E
- Generator efektów CSS3. http://www.cssmatic.com
- Generator czcionek http://www.fontsquirrel.com
- Aplikacja do testowania kodu. http://jsfiddle.net

Przydatne linki

- Najczęstsze błędy i problemy HTML i CSS.
  http://wtfhtmlcss.com
- Interaktywna prezentacja box-sizing.
  http://codepen.io/carolineartz/full/ogVXZj
- Serwis, w którym sprawdzisz, co z HTML, CSS, JS obsługuje dana przeglądarka.
  http://caniuse.com
- Inny serwis dotyczący rozwiązań HTML5.
  http://html5please.com
- Świetna prezentacja podstawowych funkcjonalność HTML i CSS.
  http://jgthms.com/web-design-in-4-minutes/

Różnice między przeglądarkami

Z powodu różnic w przeglądarkach, do niektórych atrybutów musimy dopisywać prefixy stworzone specjalnie pod dany silnik.

Robi się to głównie w celu wsparcia starszych wersji przeglądarek.

Prefiksy dla najpopularniejszych przeglądarek to:

- **-moz – Mozilla Firefox,**
- **-ms – Internet Explorer,**
- **-webkit – Google Chrome, Apple Safari, Opera.**

Identyfikatory i klasy

Identyfikatory

Czym są identyfikatory?

Identyfikatory pozwalają odróżniać od siebie poszczególne elementy dokumentu HTML. Identyfikator może być nadany każdemu elementowi za pomocą atrybutu **`id`**.

Atrybutu **`id`** jest unikalny. Możemy go przypisać tylko jednemu elementowi na stronie.

Przykład

```
<h3 id="naglowek3">
    Nagłówek z identyfikatorem
</h3>
<a id="link" href="http://google.com">
    Link również może mieć swój identyfikator
</a>
<p id="tekst">
    Paragraf z identyfikatorem
</p>
```

Klasy

Czym są klasy?

Klasy – podobnie jak identyfikatory – również pozwalają odróżniać od siebie poszczególne elementy dokumentu HTML. Klasa może być nadana każdemu elementowi za pomocą atrybutu **`class`**.

Klasy od identyfikatorów odróżnia możliwość nadania ten samej klasy wielu różnym elementom.

Jednemu elementowi możemy przypisać więcej niż jedną klasę. W tym celu oddzielamy nazwy klas spacją.

Przykład

```
<h3 class="naglowek">
    Nagłówek z przypisaną 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>
```

Relacje między elementami HTML

Model DOM

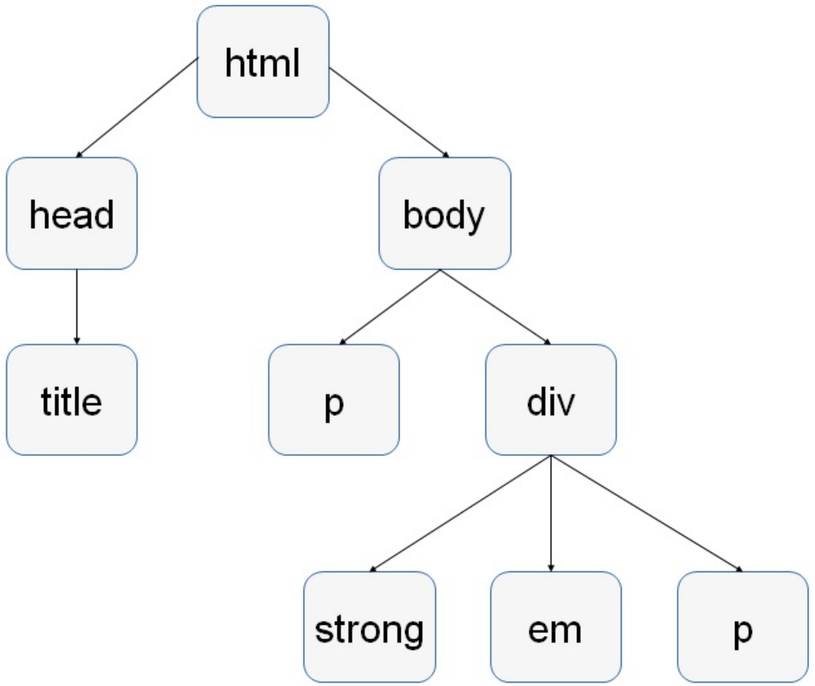
DOM

Znamy już wiele elementów HTML. Można zaobserwować, że w dokumencie HTML tworzą one pewną hierarchiczną, drzewiastą strukturę.

Nadrzędnym elementem jest **`<html>`**. Między poszczególnymi elementami zachodzą relacje rodzic – dziecko.

Element **`<html>`** jest rodzicem elementów **`<head>`** i **`<body>`**. Struktura ta jest częścią modelu **DOM (Document Object Model)**.

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>
```

HTML5
nowe elementy

HTML5

HTML5 jest rekomendowany przez W3C od 28 października 2014.

W HTML5 pisanie semantycznego kodu, czyli stosowania tagów zgodnie z ich przeznaczeniem ma ogromny wpływ na:

- Czytelność kodu,
- SEO - optymalizacja dla wyszukiwarek internetowych.

Accessibility - zapewnienie dostępu do prezentowanych treści jak najszerszemu gronu użytkowników, uwzględnienia:

- niepełnosprawność zmysłową,
- niepełnosprawność ruchową,
- upośledzenie umysłowe,
- wiek i różnice wynikające z innej percepcji.

HTML5 - najpopularniejsze tagi

`<section>`

Tematyczna grupa treści np. sekcja sportowa w portalu. Element ten nie zastępuje elementu div.

`<article>`

Zawiera samodzielną część dokumentu, przykładem może być wpis na blogu lub komentarze użytkowników. Mogą to też być np. poszczególne teksty w sekcji sportowej portalu.

`<header>`

Zawiera informacje będące wprowadzeniem na przykład do sekcji czy innego elementu (może zawierać **`h1, h2`** itp.). Może również zawierać elementy nawigacyjne.

`<footer>`

Zawiera stopkę dla elementu **`section, aside`** lub **`article`** (może zawierać takie informacje jak autor, prawa autorskie, adres).

HTML5 – najpopularniejsze tagi

`<main>`

Zawiera główne treści strony, które są bezpośrednio związane z tematem dokumentu lub funkcjonalności aplikacji. Uwaga! Może być tylko jeden element **`<main>`** na stronie.

`<aside>`

Zawiera treści nieznacznie powiązane z sekcją, do której należą na przykład cytaty, banery reklamowe. Gdybyśmy usunęli taki element ze strony, to sekcja, w której jest ten element, nie straci sensu.

HTML5 – najpopularniejsze tagi

`<nav>`

Zawiera elementy nawigujące. Powinno się go używać dla głównego menu strony.

`<figure>`

Powinien zawierać elementy **`<img>`**, które mają podpis.

`<figcaption>`

Podpis ilustracji lub zdjęcia. Musi znajdować się wewnątrz tagu **`<figure>`**

- Więcej elementów wraz z opisem można znaleźć na https://www.w3.org/wiki/HTML/Elements

Przykładowa struktura dokumentu

HTML4

```
<body>
    <div id="header"> </div>
    <div id="nav"> </div>
    <div class="section">
        <div class="article"> </div>
     </div>
    <div id="footer"> </div>
<body>
```

HTML5

```
<body>
    <header> </header>
    <nav> </nav>
    <section>
        <article> </article>
    </section>
    <footer> </footer>
<body>
```

CSS - kaskadowe arkusze stylów

Czym jest CSS?

Cascading Style Sheets

CSS (Cascading Style Sheets) – język służący do opisu wyglądu naszej strony.

Dzięki CSS możemy zmieniać sposób wyświetlania:

- czcionek,
- odnośników,
- elementów,
- obrazków...
- ... w zasadzie wszystkiego :)

Przykład

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

Reguły CSS - struktura

Jak zbudowane są reguły?

Każda reguła CSS jest zbudowana z **własności** oraz jej **wartości**. Po nazwie własności występuje dwukropek, a cała reguła kończona jest średnikiem.

Jeśli nazwa własności składa się z kilku wyrazów, to są one połączone krótką kreską (łącznikiem). Jeśli wartość własności składa się z kilku członów, to oddziela się je spacją.

Przykład

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

Reguły CSS - Selektory

Zapis selektorów

Selektory to zapisy przypisujące reguły CSS do odpowiednich elementów HTML. Jest wiele sposobów tworzenia selektorów.

Najczęściej używane to:

- 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.

Poniżej selektorem jest element **`span`.** Przeglądarka do wyświetlenia wszystkich spanów zastosuje reguły zawarte w nawiasach klamrowych.

```
span {
  color: #fff;
  font-size: 18px;
  border: 1px solid blue;
}
```

Kaskadowość CSS

Jak dokładnie działa CSS?

CSS działa na zasadzie **kaskadowości**. Wynika ona z parsowania arkuszy CSS (od góry do dołu).

Dzięki kaskadowości jesteśmy w stanie (w bardzo wyrafinowany sposób) nadpisywać wartości selektorów znajdujące się niżej w hierarchii selektorami znajdującymi się powyżej.

Jak działa 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?

Zagnieżdżanie - dziedziczenie

Dzieci przejmują po rodzicach pewne cechy

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

Przykład

HTML

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

CSS

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

Jaki font-size będzie miało p?

CSS specificity

Wstęp

Zagnieżdżanie daje nam m.in. następujące możliwości:

- wybierania tylko tych znaczników HTML, które znajdują się w innych,
- nadpisywania wartości nadanych przez style ogólne przez te bardziej szczegółowe – nazywane jest to zasadą specyficzności.

CSS specificity

Lista ulubionych trunków

Żeby dobrze zobrazować temat zacznijmy od przykładu.

- Mamy listę trzech trunków,
- Chcesz zaznaczyć swój ulubiony. W związku z tym nadajesz elementowi **`li`** odpowiednią klasę favorite,
- Dodajesz inny styl dla ulubionego trunku.

Przykład

HTML

```
<ul id="drinks">
    <li>Herbata</li>
    <li>Kawa</li>
    <li class="favorite">Piwo</li>
</ul>
```

CSS

```
.favorite {
  color: red;
}
```

CSS specificity

Nie działa!

Po odświeżeniu strony okazuje się, że coś jest nie tak, - Twój ulubiony napój ma jakiś inny styl, nie wiadomo skąd :(

Przeglądasz swój plik css, który napisałeś wcześniej, w poszukiwaniu rozwiązania i niespodziewanie znajdujesz następujący kod:

Przykład

```
ul#drinks li {
  color: blue;
}
.favorite {
  color: red;
}
```

CSS specificity

Nie działa!

Po odświeżeniu strony okazuje się, że coś jest nie tak, - Twój ulubiony napój ma jakiś inny styl, nie wiadomo skąd :(

Przeglądasz swój plik css, który napisałeś wcześniej, w poszukiwaniu rozwiązania i niespodziewanie znajdujesz następujący kod:

Przykład

```
ul#drinks li {
  color: blue;
}
.favorite {
  color: red;
}
```

Tu tkwi problem. Chcesz dwa razy ostylować ten sam element, na dwa różne sposoby.

CSS specificity

Obliczanie specyficzności

Dwa razy próbujesz ostylować ten sam element. Trzeba tutaj zastanowić się nad obliczeniem **„specyficzności selektora”.**

Zobacz na kolejnych slajdach, jak można obliczyć, który selektor będzie ważniejszy.

Przykład

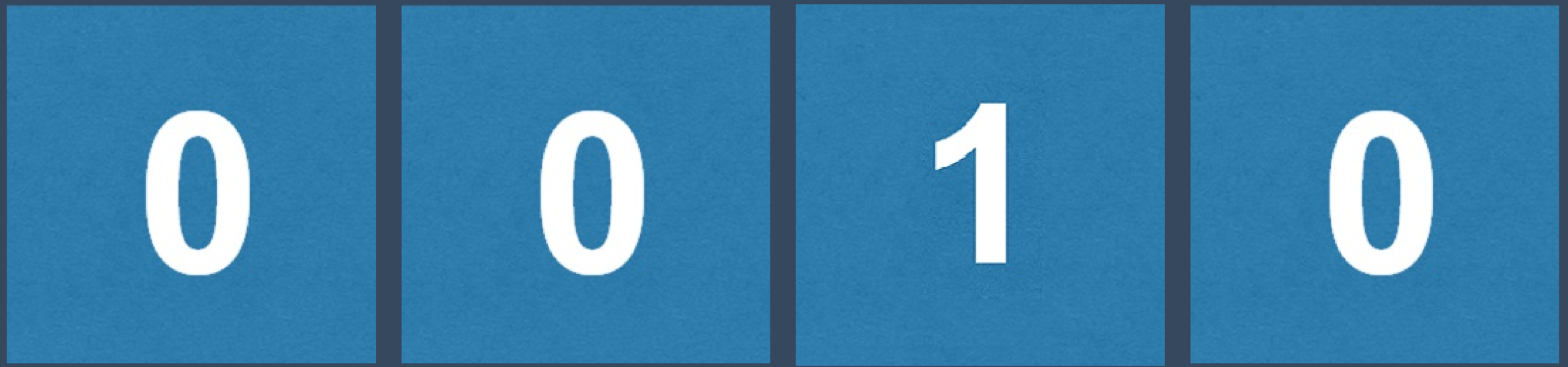
```
ul#drinks li {
  color: blue;
}
.favorite {
  color: red;
}
```

CSS specificity

inline	id	Klasy, atrybuty i pseudoklasy	Elementy i pseudoelementy
0	1	0	2

ul#drinks li

CSS specificity



.favorite

CSS specificity

Przydatne linki

Te odnośniki pomogą zrozumieć, czym jest waga selektorów.

- SpecifityWars
- Specifity Calculator

Słowo kluczowe – !important

Bomba atomowa na selektorze

W CSS istnieje słowo kluczowe **`!important`**. Tego słowa używamy, gdy chcemy, żeby jakaś wartość z selektora zawsze była najważniejsza.

Przykład

```
p {
  color: white !important;
  background-color: blue;
}
p {
  color: blue;
  background-color: black !important;
}
```

Słowo kluczowe – !important

Bomba atomowa na selektorze

W CSS istnieje słowo kluczowe **`!important`**. Tego słowa używamy, gdy chcemy, żeby jakaś wartość z selektora zawsze była najważniejsza.

Przykład

```
p {
  color: white !important;
  background-color: blue;
}
p {
  color: blue;
  background-color: black !important;
}
```

Używaj **`!important`** z rozwagą!

CSS na różnych przeglądarkach

Wsparcie przeglądarek

Nie wszystkie przeglądarki poprawnie interpretują kod. Jest to spowodowane tym, że producenci przeglądarek korzystają z różnych silników.

Dlatego każdą stronę powinniśmy testować na największej możliwej liczbie przeglądarek.

Pomocne strony do testowania

- caniuse.com
- BrowserShots.org
- www.browsera.com
- www.browserling.com

Czas na zadania

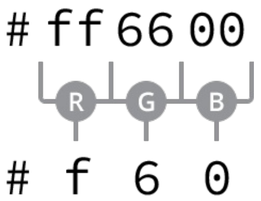
Kolory i jednostki w CSS

Kolory w CSS - przypomnienie

Jak opisujemy kolory? Przypomnienie

W CSS możemy opisywać wartości kolorów na wiele sposobów:

- nazwa (**keyword) np.: blue, red;**
- **RGB** lub **RGBa** np.: **`background-color: rgb( 255, 255, 255)`**;
- **HSL** lub **HSLa** np.: **`color:hsl( 0, 100%, 25%)`**;
- **wartość heksadecymalna** np.: **`color: #ff0ff`**;



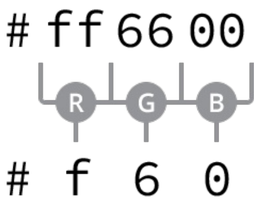

```
p {
  color: rgba(255, 0, 0, .25);
  background-color: #ffff00;
}
```

Kolory w CSS - przypomnienie

Jak opisujemy kolory? Przypomnienie

W CSS możemy opisywać wartości kolorów na wiele sposobów:

- nazwa (**keyword) np.: blue, red;**
- **RGB** lub **RGBa** np.: **`background-color: rgb( 255, 255, 255)`**;
- **HSL** lub **HSLa** np.: **`color:hsl( 0, 100%, 25%)`**;
- **wartość heksadecymalna** np.: **`color: #ff0ff`**;




```
p {
  color: rgba(255, 0, 0, .25);
  background-color: #ffff00;
}
```

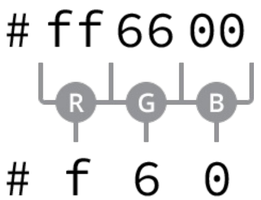
Kolor widoczny w 25%.

Kolory w CSS - przypomnienie

Jak opisujemy kolory? Przypomnienie

W CSS możemy opisywać wartości kolorów na wiele sposobów:

- nazwa (**keyword) np.: blue, red;**
- **RGB** lub **RGBa** np.: **`background-color: rgb( 255, 255, 255)`**;
- **HSL** lub **HSLa** np.: **`color:hsl( 0, 100%, 25%)`**;
- **wartość heksadecymalna** np.: **`color: #ff0ff`**;



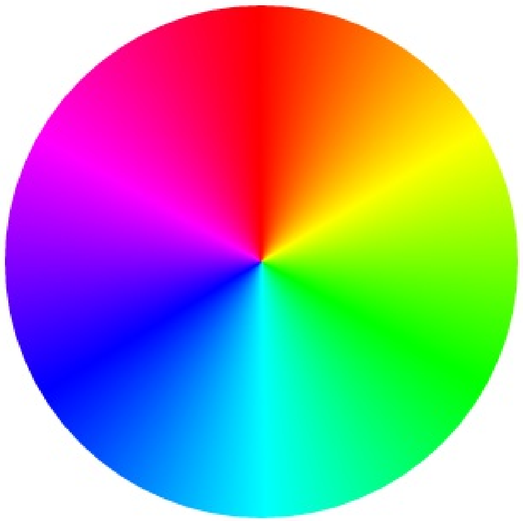

```
p {
  color: rgba(255, 0, 0, .25);
  background-color: #ffff00;
}
```

Kolor widoczny w 25%.

Żółty.

Kolory – przydatne strony

- Strona z popularnymi paletami kolorów: http://www.colourlovers.com/palettes
- 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 w CSS - przypomnienie

Relatywne i absolutne jednostki - przypomnienie

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.

```
h1 {
  font-size: 24px;
  width: 50%;
}
```

Jednostki relatywne: em

em

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.

Jednostki relatywne: em

Przykład

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

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

Jednostki relatywne: em

Przykład

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

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

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

Jednostki relatywne: em

Przykład

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

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

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

20px * 2 = 40px

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 tekstu - wartość nadrzędną definiujemy raz, w elemencie html, po czym odnosimy się do niego bezpośrednio definiując wielkość czcionki dla poszczególnych elementów na stronie.

Jednostki relatywne: rem

Przykład

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

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

Jednostki relatywne: rem

Przykład

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

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

Nieważne co ustawimy w elemencie nadrzędnym, wielkość czcionki będzie zależna od **`roota`**.

Jednostki relatywne: rem

Przykład

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

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

Nieważne co ustawimy w elemencie nadrzędnym, wielkość czcionki będzie zależna od **`roota`**.

16px * 2 = 32px

Czas na zadania

Podstawowe selektory CSS

Selektory

Selektory w CSS - wrażliwość na wielkość liter

W CSS jest wiele różnych typów selektorów. Takie połączenie selektorów oddziałujących tylko na elementy, na których nam zależy, to największa trudność w pisaniu kodu CSS.

Selektory w CSS są wrażliwe na wielkość znaków (case sensitive).

Selektor uniwersalny

Selektor uniwersalny oddziałuje na wszystkie elementy znajdujące się na stronie bądź wszystkie elementy znajdujące się wenątrz innego selektora.

```html
<h1>Nagłówek</h1>
<p>Paragraf</p>
<div>Div</div>
```

```css
* {
  color: red;
  font-size: 14px;
}
```

Selektor typu

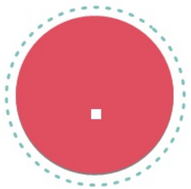
Selektor typu wybiera wszystkie elementy danego typu znajdujące się na stronie.

```html
<h1>Nagłówek</h1>
<p>Paragraf</p>
<div>Div</div>
```

```css
p {
  color: red;
  font-size: 14px;
}
```

Selektor klasy

```
<h2 class="content">Nagłówek ma klasę content</h2>
<p class="content">Paragraf ma klasę content</p>
<p class="content">Ten paragraf tez ma klasę content</p>
```



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

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

Nagłówek ma klasę content.

Paragraf ma klasę content.

Ten paragraf tez ma klasę content.

Selektor identyfikatora

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

```
<p id="intro">
    Ten paragraf ma identyfikator intro
</p>
```

```
#intro {
  color: red;
  font-weight: bold;
  font-style: 1px solid black;
}
```

Ten paragraf ma identyfikator `intro`.

Łączenie selektorów

Jeżeli chcemy zastosować jeden kod CSS do wielu selektorów – możemy wypisać je wszystkie, używając przecinka do ich rozdzielenia.

```
<h1>Nagłówek 1</h1>
<p>Paragraf</p>
<h4>Nagłówek 4</h4>
```

```
p, h1, h4  {
  color: #282828;
}
```

Selektor dziecka

selektor1 > selektor2

Jeżeli dwa dowolne selektory rozdzielimy znakiem **>**, wtedy szukamy wszystkich **selektorów2**, które są bezpośrednimi dziećmi **selektora1**.

```
<p class="red">Paragraf</p>
<div>
    <h1 class="red">Nagłówek</h1>
    <p class="red">Paragraf</p>
    <h2>Nagłówek</h2>
</div>
```

```
div > .red {
  color: navy;
}
```

Selektor dziecka

selektor1 > selektor2

Jeżeli dwa dowolne selektory rozdzielimy znakiem **>**, wtedy szukamy wszystkich **selektorów2**, które są bezpośrednimi dziećmi **selektora1**.

```
<p class="red">Paragraf</p>
<div>
    <h1 class="red">Nagłówek</h1>
    <p class="red">Paragraf</p>
    <h2>Nagłówek</h2>
</div>
```

```
div > .red {
  color: navy;
}
```

Selektor dziecka

selektor1 > selektor2

Jeżeli dwa dowolne selektory rozdzielimy znakiem **>**, wtedy szukamy wszystkich **selektorów2**, które są bezpośrednimi dziećmi **selektora1**.

```html
<p class="red">Paragraf</p>
<div>
    <h1 class="red">Nagłówek</h1>
    <p class="red">Paragraf</p>
    <h2>Nagłówek</h2>
</div>
```

```css
div > .red {
  color: navy;
}
```

Selektor potomka

selektor1 selektor2

Jeżeli dwa dowolne selektory rozdzielimy tylko spacją – szukamy wszystkich **selektów2**, które są potomkami **selektor1**.

```
<article>
    <div>
        <p> Tekst 1 </p>
    </div>
    <p> Tekst 2 </p>
</article>
```

```
article p {
  color: orange;
  font-size: 14px;
}
```

Selektor potomka

selektor1 selektor2

Jeżeli dwa dowolne selektory rozdzielimy tylko spacją – szukamy wszystkich **selektów2**, które są potomkami **selektor1**.

```
<article>
    <div>
        <p> Tekst 1 </p>
    </div>
    <p> Tekst 2 </p>
</article>
```

```
article p {
  color: orange;
  font-size: 14px;
}
```

Selektor potomka

selektor1 selektor2

Jeżeli dwa dowolne selektory rozdzielimy tylko spacją – szukamy wszystkich **selektów2**, które są potomkami **selektor1**.

```
<article>
    <div>
        <p> Tekst 1 </p>
    </div>
    <p> Tekst 2 </p>
</article>
```

```
article p {
  color: orange;
  font-size: 14px;
}
```

Selektor bezpośredniego rodzeństwa

selektor1 + selektor2

Jeżeli dwa dowolne selektory rozdzielimy znakiem **+**, wtedy szukamy wszystkich **selektorów2**, które występują bezpośrednio po **selektorze1**.

```
<h1>Nagłówek</h1>
<p>Paragraf</p>
<p>Paragraf</p>
<div>Nagłówek</div>
<p>Paragraf</p>
<h1>Nagłówek</h1>
<p>Paragraf</p>
```

```
h1 + p {
  color: red;
}
```

Selektor bezpośredniego rodzeństwa

selektor1 + selektor2

Jeżeli dwa dowolne selektory rozdzielimy znakiem **+**, wtedy szukamy wszystkich **selektorów2**, które występują bezpośrednio po **selektorze1**.

```
<h1>Nagłówek</h1>
<p>Paragraf</p>
<p>Paragraf</p>
<div>Nagłówek</div>
<p>Paragraf</p>
<h1>Nagłówek</h1>
<p>Paragraf</p>
```

```
h1 + p {
  color: red;
}
```

Selektor bezpośredniego rodzeństwa

selektor1 + selektor2

Jeżeli dwa dowolne selektory rozdzielimy znakiem **+**, wtedy szukamy wszystkich **selektorów2**, które występują bezpośrednio po **selektorze1**.

```
<h1>Nagłówek</h1>
<p>Paragraf</p>
<p>Paragraf</p>
<div>Nagłówek</div>
<p>Paragraf</p>
<h1>Nagłówek</h1>
<p>Paragraf</p>
```

```
h1 + p {
  color: red;
}
```

Selektor rodzeństwa

selektor1 ~ selektor2

Jeżeli dwa dowolne selektory rozdzielimy znakiem ~, wtedy szukamy wszystkich **selektorów2**, które mają tego samego rodzica co **selektor1** oraz występują poniżej elementu wskazanego **selektorem1**.

```
<div>
    <p>Paragraf</p>
    <h1>Nagłówek</h1>
    <p>Paragraf</p>
    <p>Paragraf</p>
    <div>Nagłówek</div>
    <p>Paragraf</p>
    <h1>Nagłówek</h1>
    <p>Paragraf</p>
</div>
```

```
h1 ~ p {
 color: red;
}
```

Selektor rodzeństwa

selektor1 ~ selektor2

Jeżeli dwa dowolne selektory rozdzielimy znakiem ~, wtedy szukamy wszystkich **selektorów2**, które mają tego samego rodzica co **selektor1** oraz występują poniżej elementu wskazanego **selektorem1**.

```
<div>
    <p>Paragraf</p>
    <h1>Nagłówek</h1>
    <p>Paragraf</p>
    <p>Paragraf</p>
    <div>Nagłówek</div>
    <p>Paragraf</p>
    <h1>Nagłówek</h1>
    <p>Paragraf</p>
</div>
```

```
h1 ~ p {
 color: red;
}
```

Selektor rodzeństwa

selektor1 ~ selektor2

Jeżeli dwa dowolne selektory rozdzielimy znakiem ~, wtedy szukamy wszystkich **selektorów2**, które mają tego samego rodzica co **selektor1** oraz występują poniżej elementu wskazanego **selektorem1**.

```
<div>
    <p>Paragraf</p>
    <h1>Nagłówek</h1>
    <p>Paragraf</p>
    <p>Paragraf</p>
    <div>Nagłówek</div>
    <p>Paragraf</p>
    <h1>Nagłówek</h1>
    <p>Paragraf</p>
</div>
```

```
h1 ~ p {
 color: red;
}
```

Selektor rodzeństwa

selektor1 ~ selektor2

Jeżeli dwa dowolne selektory rozdzielimy znakiem ~, wtedy szukamy wszystkich **selektorów2**, które mają tego samego rodzica co **selektor1** oraz występują poniżej elementu wskazanego **selektorem1**.

```
<div>
    <p>Paragraf</p>
    <h1>Nagłówek</h1>
    <p>Paragraf</p>
    <p>Paragraf</p>
    <div>Nagłówek</div>
    <p>Paragraf</p>
    <h1>Nagłówek</h1>
    <p>Paragraf</p>
</div>
```

```
h1 ~ p {
 color: red;
}
```

Łączenie selektorów

Łączenie selektorów

Łączenie spacją

Selektory klas i identyfikatorów możemy ze sobą łączyć.

```
<section id="news">
    <article class="sport">
        <div class="football">
            <aside class="quote">
            </aside>
        </div>
    </article>
</section>
```

Łączenie selektorów

Łączenie spacją

Selektory klas i identyfikatorów możemy ze sobą łączyć.

```
#news .sport
```

```
<section id="news">
    <article class="sport">
        <div class="football">
            <aside class="quote">
            </aside>
        </div>
    </article>
</section>
```

Łączenie selektorów

Łączenie spacją

Selektory klas i identyfikatorów możemy ze sobą łączyć.

```
#news .sport

#news .sport .football
```

```
<section id="news">
    <article class="sport">
        <div class="football">
            <aside class="quote">
            </aside>
        </div>
    </article>
</section>
```

Łączenie selektorów

Łączenie spacją

Selektory klas i identyfikatorów możemy ze sobą łączyć.

```
#news .sport

#news .sport .football

#news .sport .footbal .quote
```

```
<section id="news">
    <article class="sport">
        <div class="football">
            <aside class="quote">
            </aside>
        </div>
    </article>
</section>
```

Kaskadowość przy łączeniu selektorów

Co oznacza kaskadowość?

Słowo **kaskadowość** w nazwie stylów oznacza, że style mogą się wzajemnie przesłaniać lub nakładać się na siebie.

```
<div class="red box"></div>
<div class="blue box"></div>
<div class="green box"></div>
```

```
.box {
  width: 100px;
  float: left;
  height: 100px;
}
.red {
  background: red;
}
.blue {
  background: blue;
}
.green {
  background: green;
}
```

Selektor atrybutów

[atrybut="wartość"]

Atrybut jest równy

Znajdź elementy, których atrybut **`id`** jest równy **panel**.

```
<div id="panel"></div>
<div id="top-panel"></div>
<div id="panel-bottom"></div>
```

```
div[id="panel"] {
  color: red;
}
```

[atrybut="wartość"]

Atrybut jest równy

Znajdź elementy, których atrybut **id** jest równy **panel**.

```
<div id="panel"></div>
<div id="top-panel"></div>
<div id="panel-bottom"></div>
```

```
div[id="panel"] {
  color: red;
}
```

[atrybut*="wartość"]

Atrybut zawiera

Znajdź elementy, których atrybut **id** zawiera ciąg znaków **panel**.

```
<div id="panel"></div>
<div id="top-panel"></div>
<div id="panel-bottom"></div>
<div id="panelLeft"></div>
```

```
div[id*="panel"] {
  text-transform: uppercase;
}
```

[atrybut*="wartość"]

Atrybut zawiera

Znajdź elementy, których atrybut **id** zawiera ciąg znaków **panel**.

```html
<div id="panel"></div>
<div id="top-panel"></div>
<div id="panel-bottom"></div>
<div id="panelLeft"></div>
```

```css
div[id*="panel"] {
  text-transform: uppercase;
}
```

[atrybut^="wartość"]

Atrybut zaczyna się

Znajdź elementy, których atrybut **id** zaczyna się od ciągów znaków **panel**.

```
<div id="panel"></div>
<div id="top-panel"></div>
<div id="panel-bottom"></div>
<div id="panelLeft"></div>
```

```
div[id^="panel"] {
  text-transform: uppercase;
}
```

[atrybut^="wartość"]

Atrybut zaczyna się

Znajdź elementy, których atrybut **id** zaczyna się od ciągów znaków **panel**.

```
<div id="panel"></div>
<div id="top-panel"></div>
<div id="panel-bottom"></div>
<div id="panelLeft"></div>
```

```
div[id^="panel"] {
  text-transform: uppercase;
}
```

[atrybut$="wartość"]

Atrybut kończy się

Znajdź elementy, których atrybut **id** kończy się ciągiem znaków **panel**.

```
<div id="panel"></div>
<div id="top-panel"></div>
<div id="panel-bottom"></div>
<div id="panelLeft"></div>
```

```
div[id$="panel"] {
  text-transform: capitalize;
}
```

[atrybut$="wartość"]

Atrybut kończy się

Znajdź elementy, których atrybut **id** kończy się ciągiem znaków **panel**.

```
<div id="panel"></div>
<div id="top-panel"></div>
<div id="panel-bottom"></div>
<div id="panelLeft"></div>
```

```
div[id$="panel"] {
  text-transform: capitalize;
}
```

[atrybut~="wartość]

Rozdzielone spacją

Znajdź elementy, których atrybut **id** zawiera ciąg znaków **panel**, rozdzielony spacją od innych.

Samo słowo panel też zostanie znalezione.

```
<div id="panel"></div>
<div id="top-panel"></div>
<div id="panel-bottom"></div>
<div id="panelLeft"></div>
```

```
div[id~="panel"] {
    border: 1px dotted black;
}
```

[atrybut~="wartość]

Rozdzielone spacją

Znajdź elementy, których atrybut **id** zawiera ciąg znaków **panel**, rozdzielony spacją od innych.

Samo słowo panel też zostanie znalezione.

```
<div id="panel"></div>
<div id="top-panel"></div>
<div id="panel-bottom"></div>
<div id="panelLeft"></div>
```

```
div[id~="panel"] {
    border: 1px dotted black;
}
```

[atrybut|="wartość]

Rozdzielone kreską

Znajdź elementy, których atrybut **id** zaczyna się od ciągu znaków **panel** i jest rozdzielony kreską (łącznikiem).

Samo słowo panel też zostanie znalezione.

```
<div id="panel"></div>
<div id="top-panel"></div>
<div id="panel-bottom"></div>
<div id="panelLeft"></div>
```

```
div[id|="panel"] {
    border: 1px dotted black;
}
```

[atrybut|="wartość]

Rozdzielone kreską

Znajdź elementy, których atrybut **id** zaczyna się od ciągu znaków **panel** i jest rozdzielony kreską (łącznikiem).

Samo słowo panel też zostanie znalezione.

```
<div id="panel"></div>
<div id="top-panel"></div>
<div id="panel-bottom"></div>
<div id="panelLeft"></div>
```

```
div[id|="panel"] {
    border: 1px dotted black;
}
```

Czas na zadania

Czcionki

Czcionka

Czcionka - przypomnienie

Dzięki właściwości **`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 bezszeryfowych.

Czcionka

```css
p {
  font-family: "Arial Light", Arial, sans-serif;
  font-size: 24px;
  font-style: italic;
  font-weight:  700;
  line-height: 2em;
  text-align: center;
  text-decoration: line-through;
  text-indent: 20px;
  text-shadow: 0 5px 5px rgba(255, 0, 0, .5);
  text-transform: uppercase;
  letter-spacing: -5em;
  word-spacing: .3em;
}
```

Czcionki bezpieczne

Jaki rodzaj czcionki wybrać?

Możemy założyć, że na każdym urządzeniu (komputerze, telefonie czy tablecie) na pewno będą zainstalowane następujące czcionki:

- Arial
- Courier New, Courier,
- Garamond,
- Georgia,
- Lucida Sans, Lucida Grande, Lucida,
- Palatino Linotype,
- Tahoma,
- Times New Roman, Times,
- Trebuchet,
- Verdana.

Czcionki hostowane

Jaki rodzaj czcionki wybrać?

Mamy możliwość trzymania czcionek na własnym serwerze i udostępniania ich do automatycznego ściągnięcia przez użytkownika.

Robimy to, używając reguły CSS
`@font-face`

- Darmowe czcionki do użytku komercyjnego, generator czcionek. http://www.fontsquirrel.com

```
@font-face {
  font-family: "Myfont";
  src: local("Myfont"), url("my_font.woff"), format("woff");
}
p {
 font-family: "Myfont", cursive;
}
```

Czcionki hostowane zewnętrznie

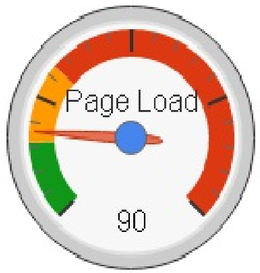
Jaki rodzaj czcionki wybrać?

Możemy korzystać z czcionek utrzymywanych przez zewnętrzne serwery.

Jeśli te serwery zostaną zamknięte lub przeniesione – to nasza czcionka przestanie działać.

Łatwy do użycia i obszerny zbiór czcionek, hostowany przez Google to: http://www.google.com/fonts.

Google Fonts



font-family vs font-face

Opis	Czcionki bezpieczne	Lokalizacja czcionki jest wskazana na naszym serwerze	Czcionka jest utrzymywana na zewnętrznym serwerze
Problemy	Mała liczba czcionek	Spowalnia czas ładowania się strony	Spowalnia czas ładowania się strony
Licencja	Nie dotyczy	Musimy mieć licencję na dystrybucje czcionki	Serwis musi mieć licencję
Liczba czcionek	Mała	Duża (ale część jest płatna)	Duża (ale część jest płatna)

Czas na zadania

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 stare przeglądarki mogą nie wyświetlić tego koloru.

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

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`**.

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

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

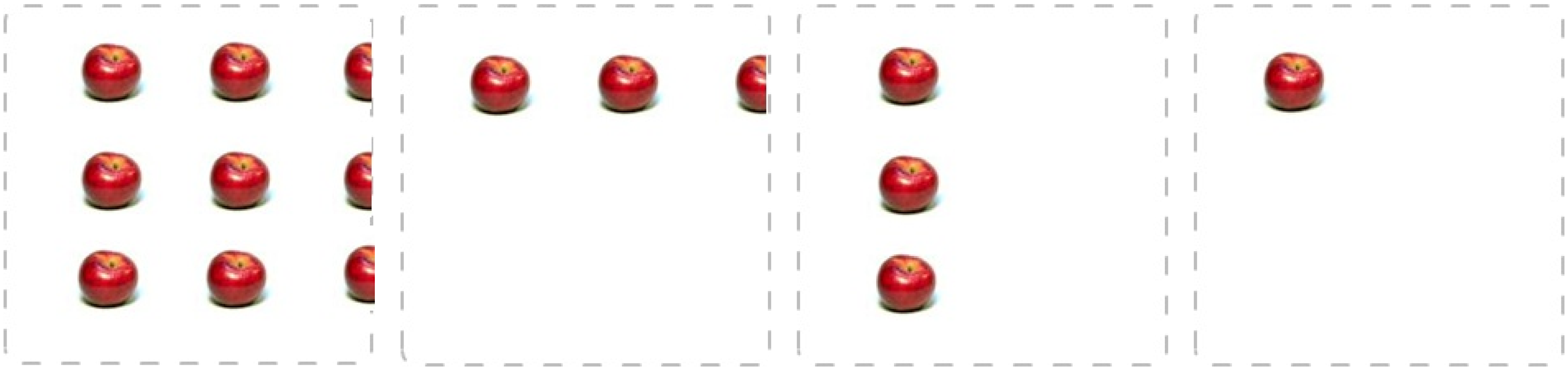
Możemy sterować obrazem dołączonym do naszej strony za pomocą atrybutu **`background-repeat`**.

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-size

Określenie wielkości tła

Właściwość **`background-size`** pozwala ustawić rozmiar obrazka w tle.

Właściwość może przyjmować kilka różnych wartości:

- **jednostki**, **procenty** (np. **`15px, 50%`**): obrazek jest skalowany do rozmiarów elementu, nie zachowuje proporcji,
- **`cover`**: skaluje obraz tak, by był duży jak to tylko możliwe, tak, że obszar tła jest całkowicie pokryty przez obraz. Zachowane są proporcje, więc w przypadku innych proporcji obrazu i elementu, obraz jest przycięty,
- **`contain`**: skaluje obraz do maksymalnego rozmiaru, tak że cały obraz mieści się w elemencie zachowując proporcje.

```
.grunwald {
  width: 300px;
  height: 300px;
  background-image: url("grunwald.jpg");
}
```

background-size

Określenie wielkości tła

Zdefiniowaliśmy element klasy **`.grunwald`** o wielkości **`300px*300px`**. Oryginalny rozmiar obrazu **`grunwald.jpg`**, to **`1200px*522px`**. Obraz jest większy niż element, zatem będzie przycięty przez przeglądarkę:

```
.grunwald {
  background-image:url("grunwald.jpg");
  width: 300px;
  height: 300px;
}
```

background-size

`background-size: 100% 100%;`

`background-size: 50% 100%;`

`background-size: cover;`

`background-size: contain;`

background-position

Ustawienie tła

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

Właściwość może przyjmować wartość w trzech różnych jednostkach:

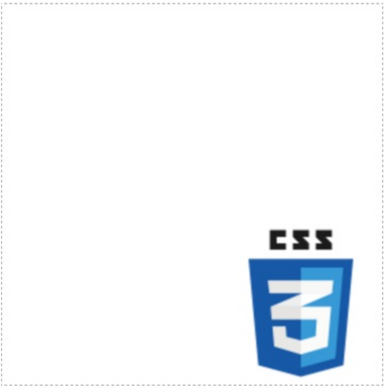
- długości (np. **15px, 50px**),
- procentach (np. **20%, 50%**),
- słowach kluczowych (**left, right, top, center, bottom**).

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

background-position

```
background-repeat:
no-repeat;
background-position:
0 150px;
```

```
background-repeat:
no-repeat;
background-position:
10% 70%;
```



```
background-repeat:
no-repeat;
background-position:
right bottom;
```

background-position

```
background-repeat:
no-repeat;
background-position:
center;
```

```
background-repeat:
repeat-x;
background-position:
center;
```

background

Background

Wszystkie własności tła możemy zawrzeć w jednej linii używając właściwości **background**.

W wersji skróconej własności są w następującej kolejności:

- **background-color**
- **background-image**
- **background-repeat**
- **background-position**
- **background-size**

Własności tła jest więcej jednak te pięć wymienionych to najważniejsze własności.

```
#fancy_image {
  background: black url("img/obrazek.jpg") no-repeat center/cover;
}
```

Czas na zadania