

3.1 3.1 Teoria

Step 1

Przykłady znajdują się pod

adresem: https://gitlab.com/mmiotk/technologieinternetu_nst_examples/-/tree/lecture_31 (https://gitlab.com/mmiotk/technologieinternetu_nst_examples/-/tree/lecture_31)

Cascading Style Sheets (CSS) to standard W3C służący do definiowania prezentacji dokumentów napisanych w HTML , a właściwie w dowolnym języku XML . Prezentacja ponownie odnosi się do sposobu, w jaki dokument jest dostarczany użytkownikowi, niezależnie od tego, czy jest wyświetlany na ekranie komputera, wyświetlany na telefonie komórkowym, drukowany na papierze, czy czytany na głos przez czytnik ekranu. Dzięki arkuszom stylów obsługującym prezentację, HTML może zająć się definiowaniem struktury i znaczenia dokumentu zgodnie z zamierzeniami. CSS to osobny język z własną składnią. Nie musimy dalej przekonywać, że arkusze stylów są właściwą drogą, ale oto krótki przegląd korzyści płynących z używania arkuszy stylów.

- **Precyjne sterowanie typem i układem.** Możemy osiągnąć precyzję zbliżoną do druku za pomocą CSS . Istnieje nawet zestaw właściwości skierowanych konkretnie do drukowanej strony.
- **Mniej pracy.** Możemy zmienić wygląd całej witryny, edytując jeden arkusz stylów. Zapewnia to również spójność formatowania w całej witrynie.
- **Bardziej dostępne witryny.** Gdy wszystkie kwestie związane z prezentacją są obsługiwane przez CSS , możemy w znaczący sposób oznaczyć swoje treści, czyniąc je bardziej dostępnymi dla urządzeń niewizualnych lub mobilnych.

Naprawdę nie ma żadnych wad korzystania z arkuszy stylów. Istnieją pewne utrzymujące się problemy związane z niespójnością przeglądarki, ale można ich uniknąć lub obejść je, jeśli wiemy, gdzie ich szukać.

Nie mówimy tutaj o drobnych poprawkach wizualnych, takich jak zmiana koloru nagłówków lub dodanie wcięć tekstu. W pełni wykorzystany potencjał CSS jest solidnym i potężnym narzędziem do projektowania. Na stronie CSS Zen Garden (www.csszengarden.com (www.csszengarden.com)) można zaobserwować to co można uzyskać dzięki CSS . Tworzenie układów CSS wymaga dużo praktyki. Pomagają też zabójcze umiejętności projektowania graficznego. Trzeba być świadomym potencjału projektowania opartego na CSS , zwłaszcza że przykłady wydają się być proste i bezpośrednie.

Jak działają kaskadowe arkusze stylów? Wyjaśnijmy to w paru krokach.

1. Zaczniemy od dokumentu, który został oznaczony w HTML .
2. Napisz zasady stylu określające, jak chcemy zmienić wygląd niektórych elementów.
3. Dołącz reguły stylu do dokumentu. Kiedy przeglądarka wyświetla dokument, przestrzega reguł dotyczących renderowania elementów.

Wiemy, że ważne jest, aby wybierać elementy, które dokładnie opisują znaczenie treści. Znaczniki tworzą strukturę dokumentu, czasami nazywaną warstwą strukturalną, na którą można zastosować warstwę prezentacji. Na potrzeby prezentacji będziemy działać z następującym dokumentem HTML .

```

<!--cooking.html-->
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Cooking with Mateusz Miotk</title>
    <style>
    </style>
</head>

<body>
<h1>Cooking with Mateusz</h1>

<p>I had the pleasure of spending a crisp, Spring day in Portsmouth, NH cooking and chatting with Mateusz Miotk as he prepared a gourmet, sit-down dinner for 28 pals.</p>

<p>When I first invited Mateusz to be on the show, I was told that Mateusz Miotk was the guy I wanted to talk to. Then Mateusz emailed his response: "i'm way into it, but i don't want to talk about it, i wanna do it." After years of only having access to touring bands between their sound check and set, I've been doing a lot of <em>talking</em> about cooking with rockstars. To actually cook with a band was a dream come true.</p>

<h2>Six-hour Salad</h2>

<p></p>

<p>Mateusz prepared a salad of arugula, smoked tomatoes, tomato jam, and grilled avocado (it's as good as it sounds!). I jokingly called it "6-hour Salad" because that's how long he worked on it. The fresh tomatoes were slowly smoked over woodchips in the grill, and when they were softened, Mateusz separated out the seeds which he reduced into a smoky jam. The tomatoes were cut into strips to put on the salads. As the day meandered, the avocados finally went on the grill after dark. I was on flashlight duty while Mateusz checked for the perfect grill marks.</p>

<p>I wrote up a streamlined adaptation of his recipe that requires <em>much</em> less time and serves 6 people instead of <em>five</em>times that amount.</p>

<h2>The Main Course</h2>

<p>In addition to the smoky grilled salad, Mateusz served tarragon cornish hens with a cognac cream sauce loaded with chanterelles and grapes, and wild rice with grilled ramps (wild garlicky leeks). Dinner was served close to midnight, but it was a party so nobody cared.</p>

<p>We left that night (technically, early the next morning) with full bellies, new cooking tips, and nearly 5 hours of footage. I'm considering renaming the show "Cooking with Mateusz Miotk". </p>

</body>
</html>

```

Cooking with Mateusz

I had the pleasure of spending a crisp, Spring day in Portsmouth, NH cooking and chatting with Mateusz Miotk as he prepared a gourmet, sit-down dinner for 28 pals.

When I first invited Mateusz to be on the show, I was told that Mateusz Miotk was the guy I wanted to talk to. Then Mateusz emailed his response: "I'm way into it, but i don't want to talk about it, i wanna do it." After years of only having access to touring bands between their sound check and set, I've been doing a lot of *talking* about cooking with rockstars. To actually cook with a band was a dream come true.

Six-hour Salad



Mateusz prepared a salad of arugula, smoked tomatoes, tomato jam, and grilled avocado (it's as good as it sounds!). I jokingly called it "6-hour Salad" because that's how long he worked on it. The fresh tomatoes were slowly smoked over woodchips in the grill, and when they were softened, Mateusz separated out the seeds which he reduced into a smoky jam. The tomatoes were cut into strips to put on the salads. As the day meandered, the avocados finally went on the grill after dark. I was on flashlight duty while Mateusz checked for the perfect grill marks.

I wrote up a streamlined adaptation of his recipe that requires *much* less time and serves 6 people instead of *fivetimes* that amount.

The Main Course

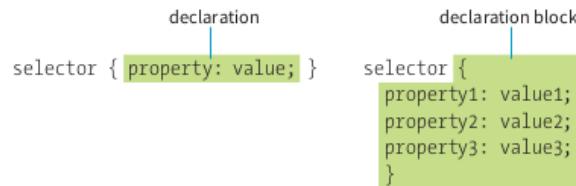
In addition to the smoky grilled salad, Mateusz served tarragon cornish hens with a cognac cream sauce loaded with chanterelles and grapes, and wild rice with grilled ramps (wild garlicky leeks). Dinner was served close to midnight, but it was a party so nobody cared.

We left that night (technically, early the next morning) with full bellies, new cooking tips, and nearly 5 hours of footage. I'm considering renaming the show "Cooking with Mateusz Miotk".

Arkuszów stylów składa się z jednej lub więcej **instrukcji stylów** (zwanych **regułami stylów**), które opisują sposób wyświetlania elementu lub grupy elementów. Pierwszym krokiem w nauce CSS jest zapoznanie się z częściami reguły. Jak zobaczymy, śledzenie ich jest dość intuicyjne. Każda reguła wybiera element i deklaruje, jak powinien wyglądać. Poniższy przykład zawiera dwie reguły. Pierwszy sprawia, że wszystkie elementy `h1` w dokumencie są zielone; druga określa, że akapity powinny być pisane dużą czcionką bezszeryfową. Czcionki bezszeryfowe nie mają małej płytki (szeryf) na końcach pociągnięć i wyglądają bardziej elegancko i nowocześnie.

```
h1 { color: green; }
p { font-size: large; font-family: sans-serif; }
```

W terminologii CSS dwie główne sekcje reguły to **selektor**, który identyfikuje element lub elementy, których ma dotyczyć, oraz **deklaracja** zawierająca instrukcje renderowania. Z kolei deklaracja składa się z właściwości (takiej jak `color`) i jej wartości (`green`), oddzielonych dwukropkiem i spacją. Jedna lub więcej deklaracji jest umieszczanych w nawiasach klamrowych.



W poprzednim przykładzie małego arkusza stylów elementy `h1` i `p` są używane jako **selektry**. Nazywa się to **selektorem elementu** i jest to najbardziej podstawowy typ selektora. Właściwości zdefiniowane dla każdej reguły będą miały zastosowanie odpowiednio do każdego elementu `h1` i `p` w dokumencie. Innym typem selektora jest **selektor ID**, który wybiera element na podstawie wartości atrybutu `id` elementu. Jest to oznaczone symbolem `#`. Na przykład selektor `#recipe` kieruje do elementu z `id="recipe"`.

Deklaracja składa się z pary **właściwość/wartość**. W jednej regule może być więcej niż jedna deklaracja; na przykład reguła dla elementu `p` pokazana wcześniej w przykładzie kodu ma zarówno właściwości `font-size`, jak i `font-family`. **Każda deklaracja musi kończyć się średnikiem**, aby oddzielić ją od następnej deklaracji. Jeśli pominiemy średnik, deklaracja i następująca po nim deklaracja zostaną zignorowane. Nawiązy klamrowe i zawarte w nich deklaracje są często nazywane **blokiem deklaracji**. Ponieważ CSS ignoruje białe znaki i zwraca wiersze w bloku deklaracji, autorzy zazwyczaj piszą każdą deklarację w bloku w osobnym wierszu, jak pokazano w poniższym przykładzie. Ułatwia to znalezienie właściwości zastosowanych do selektora i określenie, kiedy kończy się reguła stylu.

```
p {
  font-size: large;
  font-family: sans-serif;
}
```

Zauważmy, że tutaj tak naprawdę nic się nie zmieniło – wciąż jest jeden zestaw nawiasów klamrowych, średniki po każdej deklaracji i tak dalej. Jedyną różnicą jest wstawienie końca linii i niektórych spacji znakowych do wyrownania. Sercem arkuszy stylów jest **zbiór standardowych właściwości**, które można zastosować do wybranych elementów. Pełna specyfikacja CSS definiuje dziesiątki właściwości dla wszystkiego, od wcięć tekstu po sposób, w jaki nagłówki tabel powinny być odczytywane na głos. **Wartości są zależne od właściwości**. Niektóre właściwości wykonują pomiary długości, inne wartości kolorów, a inne mają predefiniowaną listę słów kluczowych. Kiedy używamy właściwości, ważne jest, aby wiedzieć, jakie wartości akceptuje.

Istnieją trzy sposoby umieszczania arkuszy stylów w dokumencie HTML. Należą do nich **osadzone arkusze stylów**, **zewnętrzne arkusze stylów** oraz **wbudowane style**. Osadzone arkusze stylów jest umieszczany w dokumencie poprzez element `style`, a jego zasady dotyczą tylko tego dokumentu. Element `style` należy umieścić w nagłówku dokumentu (`head`).

```
<!--cooking.html-->
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Cooking with Mateusz Miotk</title>
    <style>
        h1 {
            color: green;
        }
        p {
            font-size: large;
            font-family: sans-serif;
        }
        img {
            float: right;
            margin: 0 12px;
        }
    </style>
</head>

<body>
<h1>Cooking with Mateusz</h1>

<p>I had the pleasure of spending a crisp, Spring day in Portsmouth, NH cooking and chatting with Mateusz Miotk as he prepared a gourmet, sit-down dinner for 28 pals.</p>

<p>When I first invited Mateusz to be on the show, I was told that Mateusz Miotk was the guy I wanted to talk to. Then Mateusz emailed his response: "i'm way into it, but i don't want to talk about it, i wanna do it." After years of only having access to touring bands between their sound check and set, I've been doing a lot of <em>talking</em> about cooking with rockstars. To actually cook with a band was a dream come true.</p>

<h2>Six-hour Salad</h2>

<p></p>

<p>Mateusz prepared a salad of arugula, smoked tomatoes, tomato jam, and grilled avocado (it's as good as it sounds!). I jokingly called it "6-hour Salad" because that's how long he worked on it. The fresh tomatoes were slowly smoked over woodchips in the grill, and when they were softened, Mateusz separated out the seeds which he reduced into a smoky jam. The tomatoes were cut into strips to put on the salads. As the day meandered, the avocados finally went on the grill after dark. I was on flashlight duty while Mateusz checked for the perfect grill marks.</p>

<p>I wrote up a streamlined adaptation of his recipe that requires <em>much</em> less time and serves 6 people instead of <em>five</em>times that amount.</p>

<h2>The Main Course</h2>

<p>In addition to the smoky grilled salad, Mateusz served tarragon cornish hens with a cognac cream sauce loaded with chanterelles and grapes, and wild rice with grilled ramps (wild garlicky leeks). Dinner was served close to midnight, but it was a party so nobody cared.</p>

<p>We left that night (technically, early the next morning) with full bellies, new cooking tips, and nearly 5 hours of footage. I'm considering renaming the show "Cooking with Mateusz Miotk". </p>

</body>
</html>
```

Cooking with Mateusz

I had the pleasure of spending a crisp, Spring day in Portsmouth, NH cooking and chatting with Mateusz Miotk as he prepared a gourmet, sit-down dinner for 28 pals.

When I first invited Mateusz to be on the show, I was told that Mateusz Miotk was the guy I wanted to talk to. Then Mateusz emailed his response: "i'm way into it, but i don't want to talk about it, i wanna do it." After years of only having access to touring bands between their sound check and set, I've been doing a lot of talking about cooking with rockstars. To actually cook with a band was a dream come true.

Six-hour Salad

Mateusz prepared a salad of arugula, smoked tomatoes, tomato jam, and grilled avocado (it's as good as it sounds!). I jokingly called it "6-hour Salad" because that's how long he worked on it. The fresh tomatoes were slowly smoked over woodchips in the grill, and when they were softened, Mateusz separated out the seeds which he reduced into a smoky jam. The tomatoes were cut into strips to put on the salads. As the day meandered, the avocados finally went on the grill after dark. I was on flashlight duty while Mateusz checked for the perfect grill marks.

I wrote up a streamlined adaptation of his recipe that requires *much* less time and serves 6 people instead of *fivetimes* that amount.

The Main Course

In addition to the smoky grilled salad, Mateusz served tarragon cornish hens with a cognac cream sauce loaded with chanterelles and grapes, and wild rice with grilled ramps (wild garlicky leeks). Dinner was served close to midnight, but it was a party so nobody cared.

We left that night (technically, early the next morning) with full bellies, new cooking tips, and nearly 5 hours of footage. I'm considering renaming the show "Cooking with Mateusz Miotk".



Zewnętrzny arkusz stylów to osobny, tylko tekstowy dokument, który zawiera szereg reguł stylów. Musi mieć nazwę z sufiksem `.css`. Dokument `.css` jest następnie linkowany (poprzez element `Link`) lub importowany (poprzez regułę `@import` w arkuszu stylów) do jednego lub więcej dokumentów `HTML`. W ten sposób wszystkie pliki w witrynie mogą mieć ten sam arkusz stylów. Jest to najpotężniejsza i preferowana metoda dołączania arkuszy stylów do treści.

```
/*cooking.css*/
h1 {
    color: green;
}
p {
    font-size: large;
    font-family: sans-serif;
}
img {
    float: right;
    margin: 0 12px;
}
```

```

<!--cooking.html-->
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Cooking with Mateusz Miotk</title>
    <link href="cooking.css" rel="stylesheet" type="text/css">
</head>

<body>
<h1>Cooking with Mateusz</h1>

<p>I had the pleasure of spending a crisp, Spring day in Portsmouth, NH cooking and chatting with Mateusz Miotk as he prepared a gourmet, sit-down dinner for 28 pals.</p>

<p>When I first invited Mateusz to be on the show, I was told that Mateusz Miotk was the guy I wanted to talk to. Then Mateusz emailed his response: "i'm way into it, but i don't want to talk about it, i wanna do it." After years of only having access to touring bands between their sound check and set, I've been doing a lot of <em>talking</em> about cooking with rockstars. To actually cook with a band was a dream come true.</p>

<h2>Six-hour Salad</h2>

<p></p>

<p>Mateusz prepared a salad of arugula, smoked tomatoes, tomato jam, and grilled avocado (it's as good as it sounds!). I jokingly called it "6-hour Salad" because that's how long he worked on it. The fresh tomatoes were slowly smoked over woodchips in the grill, and when they were softened, Mateusz separated out the seeds which he reduced into a smoky jam. The tomatoes were cut into strips to put on the salads. As the day meandered, the avocados finally went on the grill after dark. I was on flashlight duty while Mateusz checked for the perfect grill marks.</p>

<p>I wrote up a streamlined adaptation of his recipe that requires <em>much</em> less time and serves 6 people instead of <em>five</em>times that amount.</p>

<h2>The Main Course</h2>

<p>In addition to the smoky grilled salad, Mateusz served tarragon cornish hens with a cognac cream sauce loaded with chanterelles and grapes, and wild rice with grilled ramps (wild garlicky leeks). Dinner was served close to midnight, but it was a party so nobody cared.</p>

<p>We left that night (technically, early the next morning) with full bellies, new cooking tips, and nearly 5 hours of footage. I'm considering renaming the show "Cooking with Mateusz Miotk". </p>

</body>
</html>

```

Cooking with Mateusz

I had the pleasure of spending a crisp, Spring day in Portsmouth, NH cooking and chatting with Mateusz Miotk as he prepared a gourmet, sit-down dinner for 28 pals.

When I first invited Mateusz to be on the show, I was told that Mateusz Miotk was the guy I wanted to talk to. Then Mateusz emailed his response: "i'm way into it, but i don't want to talk about it, i wanna do it." After years of only having access to touring bands between their sound check and set, I've been doing a lot of *talking* about cooking with rockstars. To actually cook with a band was a dream come true.

Six-hour Salad

Mateusz prepared a salad of arugula, smoked tomatoes, tomato jam, and grilled avocado (it's as good as it sounds!). I jokingly called it "6-hour Salad" because that's how long he worked on it. The fresh tomatoes were slowly smoked over woodchips in the grill, and when they were softened, Mateusz separated out the seeds which he reduced into a smoky jam. The tomatoes were cut into strips to put on the salads. As the day meandered, the avocados finally went on the grill after dark. I was on flashlight duty while Mateusz checked for the perfect grill marks.

I wrote up a streamlined adaptation of his recipe that requires *much* less time and serves 6 people instead of *fivetimes* that amount.

The Main Course

In addition to the smoky grilled salad, Mateusz served tarragon cornish hens with a cognac cream sauce loaded with chanterelles and grapes, and wild rice with grilled ramps (wild garlicky leeks). Dinner was served close to midnight, but it was a party so nobody cared.

We left that night (technically, early the next morning) with full bellies, new cooking tips, and nearly 5 hours of footage. I'm considering renaming the show "Cooking with Mateusz Miotk".



Możemy zastosować właściwości i wartości do pojedynczego elementu, używając atrybutu `style` w samym elemencie, jak pokazano poniżej:

```
<h1 style="color: red">Introduction</h1>
```

Aby dodać wiele właściwości, po prostu należy je rozdzielić znakami średnika, w ten sposób:

<h1>Introduction</h1>

Style wbudowane mają zastosowanie tylko do konkretnego elementu, w którym się pojawiają. Należy unikać stylów wbudowanych, chyba że jest to absolutnie konieczne do zastąpienia stylów z osadzonego lub zewnętrznego arkusza stylów. Style wbudowane są problematyczne, ponieważ przeplatają informacje z prezentacji ze znacznikami strukturalnymi. Utrudniają również wprowadzanie zmian, ponieważ każdy atrybut `style` musi zostać wyszukany w źródle.

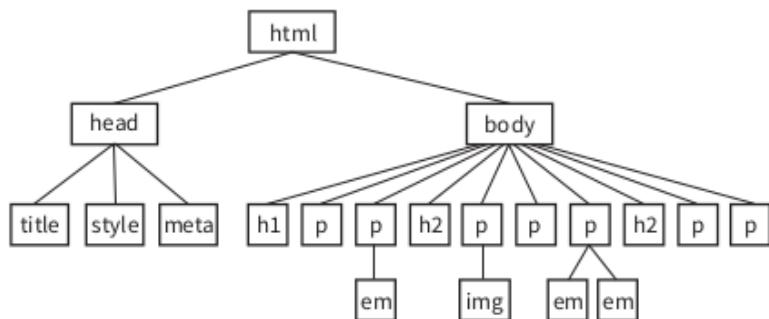
Step 2

Jest kilka wielkich pomysłów, które musimy przedyskutować, aby czuć się komfortowo z tym, jak zachowują się kaskadowe arkusze stylów. Cóż, tak jak rodzice przekazują cechy swoim dzieciom, stylizowane elementy HTML przekazują pewne właściwości stylu do elementów, które zawierają. Zauważmy, że kiedy stylowaliśmy elementy `p` dużą czcionką bezszeryfową, element `em` w drugim akapicie również stał się duży i bezszeryfowy, mimo że nie napisaliśmy specjalnie dla niego reguły. Dzieje się tak dlatego, że element `em` odziedziczył style z akapitu, w którym się znajduje. **Dziedziczenie** zapewnia mechanizm stylizowania elementów, które nie mają żadnych własnych wyraźnych reguł stylów.

Unstyled paragraph
I've been doing a lot of `talking` about cooking

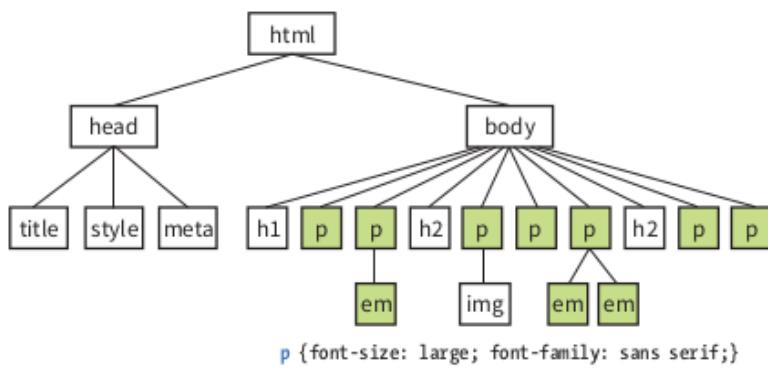
Paragraph with styles applied
I've been doing a lot of `talking` about cooking

W tym momencie ważne staje się zrozumienie struktury dokumentu. Jak już wcześniej zauważyliśmy, dokumenty HTML mają niejawną strukturę, czyli **hierarchię**. Na przykład przykładowy artykuł, z którym się bawiliśmy, ma główny element `HTML`, który zawiera nagłówek i treść, a treść zawiera elementy nagłówka i akapitu. Z kolei kilka akapitów zawiera elementy śródliniowe, takie jak obrazy (`img`) i podkreślony tekst (`em`). Możemy zobrazować strukturę jako **odwrócone drzewo**, rozgałęziające się od korzenia, jak pokazano poniżej.

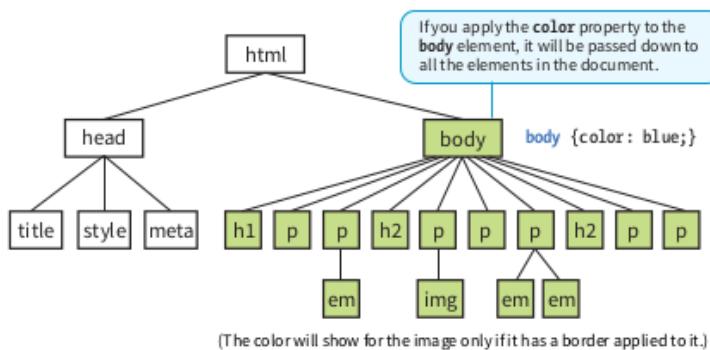


Drzewo dokumentu staje się drzewem genealogicznym, jeśli chodzi o odnoszenie się do relacji między elementami. Mówiąc, że wszystkie elementy zawarte w danym elemencie są jego **potomkami**. Na przykład, elementy `h1`, `h2`, `p`, `em` i `img` w dokumencie są potomkami elementu `body`. O elemencie, który jest bezpośrednio zawarty w innym elemencie (bez pośredniczących poziomów hierarchicznych) mówi się, że jest **dzieckiem tego elementu**. I odwrotnie, **element zawierający jest rodzicem**. Na przykład element `em` jest dzieckiem elementu `p`, a element `p` jest jego rodzicem. Wszystkie elementy znajdujące się wyżej niż dany element w hierarchii są jego przodkami. Dwa elementy z tym samym rodzicem to rodzeństwo. To wszystko może wydawać się akademickie, ale przyda się podczas pisania selektorów CSS.

Kiedy piszemy regułę stylu związaną z czcionką, używając elementu `p` jako selektora, reguła ta dotyczy wszystkich akapitów w dokumencie, a także zawartych w nich elementów tekstu śródliniowego. Widzieliśmy dowody na to, że element `em` dziedziczy właściwości stylu zastosowane do swojego rodzica (`p`), co pokazuje, co się dzieje w kontekście diagramu struktury dokumentu. Zwróciły uwagę, że element `img` jest wykluczony, ponieważ właściwości związane z czcionkami nie mają zastosowania do obrazów. Zauważmy, że „pewne” właściwości są dziedziczone.



Należy zauważyć, że niektóre właściwości arkusza stylów dziedziczą, a inne nie. Ogólnie rzecz biorąc, przekazywane są właściwości związane ze stylami tekstu – rozmiar czcionki, kolor, styl i tym podobne. Właściwości, takie jak obramowania, marginesy, tła itp., które wpływają na obramowany obszar wokół elementu, nie są przekazywane dalej. Na przykład, jeśli umieścimy obramowanie wokół akapitu, nie będziemy chcieli obramować każdego elementu wbudowanego (takiego jak `em`, `strong` lub `a`), który zawiera. Możemy wykorzystać dziedziczenie na swoją korzyść podczas pisania arkuszy stylów. Na przykład, jeśli chcemy, aby wszystkie elementy tekstowe były niebieskie, możemy napisać osobne reguły stylu dla każdego elementu w dokumencie i ustawić kolor na `blue`. Lepszym sposobem byłoby napisanie pojedynczej reguły stylu, która stosuje właściwość `color` do elementu `body` i niech wszystkie elementy zawarte w `body` odziedziczą ten styl. Każda właściwość zastosowana do określonego elementu zastępuje odziedziczone wartości tej właściwości.



CSS pozwala na zastosowanie kilku arkuszy stylów do tego samego dokumentu, co oznacza, że mogą wystąpić konflikty. Na przykład, co powinna zrobić przeglądarka, jeśli importowany arkusz stylów dokumentu mówi, że elementy `h1` powinny być czerwone, ale osadzony w nim arkusz stylów ma regułę, która sprawia, że `h1` jest fioletowy? Dwie reguły stylu z selektorami `h1` mają jednakową wagę, prawda? Ludzie, którzy napisali specyfikację arkusza stylów, przewidzieli ten problem i opracowali hierarchiczny system, który przypisuje różne wagi różnym źródłom informacji o stylu. **Kaskada** odnosi się do tego, co dzieje się, gdy kilka źródeł informacji o stylu rywalizuje o kontrolę elementów na stronie: informacje o stylu są przekazywane dalej („kaskady” w dół), dopóki nie zostanie zastąpiona przez regułę stylu o większej wadze. Waga jest uwzględniana na podstawie priorytetu źródła reguły stylu, specyfiki selektora i kolejności reguł. **Jeśli nie zastosujemy żadnych informacji o stylu do strony internetowej, zostanie ona zrenderowana zgodnie z wewnętrznym arkuszem stylów przeglądarki.** Nazywamy to domyślnym renderowaniem; W3C nazywa to **arkuszem stylów agenta użytkownika**. Poszczególni użytkownicy mogą również stosować własne style (arkusz stylów użytkownika, zwany także arkuszem stylów czytnika), które zastępują domyślne style w ich przeglądarce. Jeśli jednak autor strony internetowej dołączył arkusz stylów (arkusz stylów autora), zastępuje on zarówno style użytkownika, jak i agenta użytkownika. Informacje o stylu mogą pochodzić z różnych źródeł, wymienione tutaj od najwyższego do najniższego priorytetu. Innymi słowy, pozycje znajdujące się wyżej na liście zastępują pozycje poniżej.

- Dowolna reguła stylu oznaczona jako `!important` przez użytkownika
- Dowolna reguła stylu oznaczona jako `!important` przez autora
- Arkusze stylów napisane przez autora
- Arkusze stylów tworzone przez użytkownika
- Domyślne reguły stylów przeglądarki („arkusz stylów agenta użytkownika”)

Mogą powstać konflikty, w których element otrzymuje instrukcje dotyczące stylu z więcej niż jednej reguły. Na przykład może istnieć reguła odnosząca się do akapitów i inna reguła dla akapitu, który ma identyfikator `intro`. Jakiej reguły powinien używać akapit o identyfikatorze `intro`? W przypadku konfliktu dwóch zasad w arkuszu stylów, typ selektora jest używany do określenia zwycięzcy. Im

bardziej szczegółowy selektor, tym większą wagę przypisuje się przesłonięciu sprzecznych deklaracji. W naszym przykładzie selektor zawierający nazwę identyfikatora (`#intro`) jest bardziej szczegółowy niż selektor elementu ogólnego (np. `p`), więc ta reguła będzie miała zastosowanie do akapitu `intro`, zastępując reguły ustawione dla wszystkich akapitów.

Po posortowaniu wszystkich źródeł arkuszy stylów według priorytetów i po przetasowaniu wszystkich połączonych i zimportowanych arkuszy stylów, prawdopodobnie wystąpią konflikty w regułach o równych wagach. W takim przypadku ważna jest kolejność, w jakiej pojawiają się reguły. Kaskada działa zgodnie z zasadą „ostatni wygrywa”. Ostatnie słowo ma ta reguła, która pojawi się jako ostatnia. W arkuszu stylów, jeśli występują konflikty w regułach stylów o identycznej wadze, ta ostatnia na liście „wygrywa”. Weźmy na przykład te trzy zasady:

```
p { color: red; }
p { color: blue; }
p { color: green; }
```

W tym scenariuszu tekst akapitowy będzie zielony, ponieważ ostatnia reguła w arkuszu stylów – czyli ta najbliższa treści w dokumencie – zastępuje wcześniejsze. Proceduralnie, paragrafowi przypisywany jest kolor, następnie nowy, a na końcu trzeci (zielony), który jest używany. To samo dzieje się, gdy sprzeczne style występują w pojedynczym stosie deklaracji:

```
p { color: red;
    color: blue;
    color: green;
}
```

Wynikowy kolor będzie zielony, ponieważ ostatnia deklaracja zastępuje poprzednie dwie. Gdy wchodzimy do właściwości złożonych, łatwo jest przypadkowo zastąpić poprzednie deklaracje w regule, więc jest to ważne zachowanie, o którym należy pamiętać. To bardzo prosty przykład. Co się stanie, gdy w grę wchodzą zasady arkusza stylów z różnych źródeł? Rozważmy dokument `HTML`, który ma osadzony arkusz stylów (dodany z elementem `style`), który zaczyna się od reguły `@import` do importowania zewnętrznego pliku `.css`. Ten sam dokument `HTML` ma również kilka wbudowanych atrybutów stylu zastosowanych do poszczególnych elementów `h1`.

```
/*external.css/
h1 {
    color:red;
}
```

```
<!--external.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>External HTML</title>
    <style>
        @import url(external.css);
        h1 {color: purple;}
    </style>
</head>
<body>
    <h1 style="color: blue">Heading</h1>
    <h1>Heading 2</h1>
</body>
</html>
```

Heading

Heading 2

Kiedy przeglądarka analizuje plik, najpierw trafia do importowanego arkusza stylów, który ustawia element `h1` na czerwony. Następnie znajduje regułę o równej wadze w osadzonym arkuszu stylów, która zastępuje zimportowaną regułę, więc `h1` są ustawiane na kolor fioletowy. Kontynuując, natyka się na regułę stylu w `h1`, która ustawia jej kolor na niebieski. Ponieważ ta zasada jest ostatnia, jest zwycięzcą, a `h1` będzie niebieskie. Zauważmy, że inne `h1` w tym dokumencie bez reguły stylu śródliniowego byłyby fioletowe, ponieważ był to ostatni kolor `h1` zastosowany do całego dokumentu.

Jeśli chcemy, aby reguła nie była zastępowana przez kolejną sprzeczną regułę, należy dołączyć słowo kluczowe `!important` tuż za wartością właściwości i przed średnikiem dla tej reguły. Na przykład, aby zagwarantować, że tekst akapitu będzie niebieski, zastosujmy następującą regułę:

```
p {color: blue !important;}
```

Nawet jeśli przeglądarka napotka później w dokumencie styl wbudowany (który powinien zastąpić arkusz stylów całego dokumentu), taki jak ten:

```
<p style="color: red">
```

ten akapit nadal będzie niebieski, ponieważ reguła ze wskaźnikiem `!important` nie może zostać zastąpiona przez inne style w arkuszu stylów autora. Jedynym sposobem, w jaki `!important` reguła może zostać ominięta, jest konfliktowa reguła w arkuszu stylów czytnika (użytkownika), która również została oznaczona `!important`. Ma to na celu zapewnienie, że specjalne wymagania czytelnika, takie jak duża czcionka lub tekst o wysokim kontraście dla osób niedowidzących, nigdy nie zostaną pominięte. Na podstawie poprzednich przykładów, jeśli arkusz stylów czytelnika zawiera tę zasadę:

```
p {color: black;}
```

tekst nadal byłby niebieski, ponieważ wszystkie style autorskie (nawet te, które nie są oznaczone jako `!important`) mają pierwszeństwo przed stylami czytelnika. Jeśli jednak styl czytelnika będącego w konflikcie jest oznaczony jako `!important`, tak jak to

```
p {color: black !important;}
```

akapity będą czarne i nie można ich przesłonić żadnym stylem dostarczonym przez autora. Najlepsze praktyki podpowiadają, że słowa kluczowego `!important` powinno się go używać oszczędnie, jeśli w ogóle, i na pewno nigdy tylko po to, aby wydostać się z trudnej sytuacji dziedziczenia i kaskady.

Kamieniem węgielnym systemu formatowania wizualnego CSS jest **model pudełkowy**. Najprostszym sposobem myślenia o modelu pudełkowym jest to, że przeglądarki widzą każdy element na stronie (zarówno blokowy, jak i śródliniowy) jako zawarty w małym prostokątnym pudełku. Możemy zastosować do tych pól właściwości, takie jak obramowania, marginesy, dopełnienie i tła, a nawet zmienić ich położenie na stronie. Aby zobaczyć elementy z grubsza tak, jak widzi je przeglądarka, poniżej znajdują się zasady stylu, które dodają ramki wokół każdego elementu treści w naszym przykładowym artykule:

```
/*cooking.css*/
h1 {
    color: green;
    border: 1px solid blue;
}
p {
    font-size: large;
    font-family: sans-serif;
    border: 1px solid blue;
}
img {
    float: right;
    margin: 0 12px;
    border: 1px solid blue;
}
em {
    border: 1px solid blue;
}
```

Cooking with Mateusz

I had the pleasure of spending a crisp, Spring day in Portsmouth, NH cooking and chatting with Mateusz Miotk as he prepared a gourmet, sit-down dinner for 28 pals.

When I first invited Mateusz to be on the show, I was told that Mateusz Miotk was the guy I wanted to talk to. Then Mateusz emailed his response: "i'm way into it, but i don't want to talk about it, i wanna do it." After years of only having access to touring bands between their sound check and set, I've been doing a lot of talking about cooking with rockstars. To actually cook with a band was a dream come true.

Six-hour Salad

Mateusz prepared a salad of arugula, smoked tomatoes, tomato jam, and grilled avocado (it's as good as it sounds!). I jokingly called it "6-hour Salad" because that's how long he worked on it. The fresh tomatoes were slowly smoked over woodchips in the grill, and when they were softened, Mateusz separated out the seeds which he reduced into a smoky jam. The tomatoes were cut into strips to put on the salads. As the day meandered, the avocados finally went on the grill after dark. I was on flashlight duty while Mateusz checked for the perfect grill marks.

I wrote up a streamlined adaptation of his recipe that requires [much](#) less time and serves 6 people instead of [fivetimes](#) that amount.

The Main Course

In addition to the smoky grilled salad, Mateusz served tarragon cornish hens with a cognac cream sauce loaded with chanterelles and grapes, and wild rice with grilled ramps (wild garlicy leeks). Dinner was served close to midnight, but it was a party so nobody cared.

We left that night (technically, early the next morning) with full bellies, new cooking tips, and nearly 5 hours of footage. I'm considering renaming the show "Cooking with Mateusz Miotk".



Obramowania ujawniają kształt każdego pola elementu blokowego. Wokół elementów wbudowanych (`em` i `img`) znajdują się również ramki. Jeśli spojrzymy na nagłówki, zobaczymy, że pola elementów blokowych rozszerzają się, aby wypełnić dostępną szerokość okna przeglądarki, co jest naturą elementów blokowych w normalnym przepływie dokumentu. Pola śródliniowe obejmują tylko zawarte w nich znaki lub obraz. To dobra okazja, aby pokazać przydatny skrót do reguły stylu. Jeśli kiedykolwiek będziemy musieli zastosować tę samą właściwość stylu do kilku elementów, możemy zgrupować selektory w jedną regułę, oddzielając je przecinkami. Ta jedna zasada ma taki sam efekt jak wymienione wcześniej zasady. Grupowanie ich sprawia, że przyszłe edycje są bardziej wydajne i skutkują mniejszym rozmiarem pliku: Teraz mamy w swoim przyborniku dwa typy selektorów: prosty selektor elementów i zgrupowane selektory.

```
/*cooking.css*/
h1, h2, p, em, img {
    border: 1px solid blue;
}
h1 {
    color: green;
}
p {
    font-size: large;
    font-family: sans-serif;
}
img {
    float: right;
    margin: 0 12px;
}
```

Step 3

Jest to dobry czas na zapoznanie się z jednostkami miary używanymi w `CSS`. Będziemy ich używać do ustawiania rozmiaru czcionki, szerokości i wysokości elementów, marginesów, wcięć i tak dalej. Niektóre będą wyglądać znajomo (takie jak `cale` i `milimetry`), ale istnieją jednostki, które dają więcej wyjaśnień: jednostki bezwzględne, `rem`, `em` i `vw/vh`. Umiejętność efektywnego korzystania z jednostek `CSS` to kolejna z tych podstawowych umiejętności `CSS`.

Jednostki bezwzględne mają predefiniowane znaczenia lub odpowiedniki w świecie rzeczywistym. Są zawsze tej samej wielkości, niezależnie od kontekstu, w jakim się pojawiają. Najpopularniejszą jednostką bezwzględną do projektowania stron internetowych jest **piksel**, który `CSS3` definiuje jako **1/96 cala**. Piksele są jak w domu na ekranie opartym na pikselach i zapewniają precyzyjną kontrolę nad rozmiarem tekstu i elementów na stronie. Przez jakiś czas używany był tylko piksel. Potem zdano sobie sprawę, że są one zbyt sztywne dla stron, które muszą dostosować się do różnych rozmiarów ekranu i preferencji użytkownika. Pomiary wzgldne, takie jak `rem`, `em` i `%`, są bardziej odpowiednie dla płynnej natury urządzeń. Dopóki wyrzucamy `px` do krawężnika, wszystkie jednostki bezwzględne – takie jak `pt`, `pc`, `in`, `mm` i `cm` – są pomijane, ponieważ nie mają znaczenia na ekranach, chociaż mogą być przydatne w przypadku arkuszy stylów drukowania. To trochę zawęża wybory jednostek. To powiedziawszy, piksele nadal mają swoje miejsce w projektowaniu stron internetowych dla elementów, które naprawdę powinny pozostać tego samego rozmiaru niezależnie od kontekstu. Szerokości obramowania są odpowiednie w pikselach, podobnie jak obrazy, które mają własne wymiary w pikselach.

```
<!--units_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Units example</title>
</head>
<body>
    <h1 style="border: 10px solid blue">10px</h1>
    <h1 style="border: 10pt solid blue">10pt</h1>
    <h1 style="border: 1pc solid blue">1pc</h1>
    <h1 style="border: 1in solid blue">1in</h1>
    <h1 style="border: 10mm solid blue">10mm</h1>
    <h1 style="border: 1cm solid blue">1cm</h1>
</body>
</html>
```

10px

10pt

1pc

1in

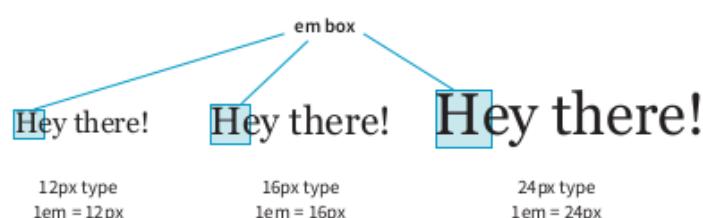
10mm

1cm

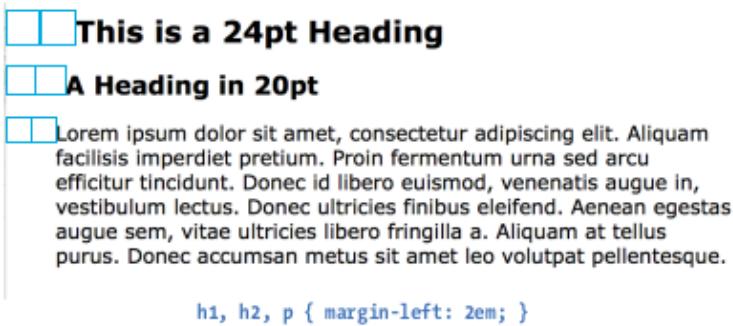
Jednostki względne są sposobem na większość pomiarów internetowych i istnieje kilka opcji: `em`, `rem` i `vw/vh`.

CSS3 wprowadził wzgólną miarę zwaną `rem` (od root `em`), która jest oparta na rozmiarze czcionki elementu root (`html`), cokolwiek to jest. W nowoczesnych przeglądarkach domyślny główny rozmiar czcionki to 16 pikseli; dlatego `rem` jest odpowiednikiem jednostki 16-pikselowej (chyba że ustawimy ją jawnie na inną wartość). Element o rozmiarze `10rem` miałby 160 pikseli. W większości przypadków możemy używać jednostek `rem`, takich jak miara bezwzględna w regułach stylu; jednak, ponieważ jest to względne, jeśli zmienia się podstawowy rozmiar czcionki, zmienia się również rozmiar `rem`. Jeśli użytkownik zmieni podstawowy rozmiar czcionki na 24 piksele, aby ułatwić czytanie z odległości, lub jeśli strona jest wyświetlana na urządzeniu z domyślnym rozmiarem czcionki 24 piksele, element `10rem` staje się 240 pikselami. Wydaje się to podejrzane, ale zapewniamy, że jest to funkcja, a nie błąd. Istnieje wiele przypadków, w których element układu graficznego powinien się rozwinąć, gdy zwiększy się rozmiar tekstu. Zachowuje proporcjonalność strony do rozmiaru czcionki, co może pomóc w utrzymaniu optymalnej długości linii.

`Em` jest wzgólną jednostką miary, która w tradycyjnej typografii opiera się na szerokości wielkiej litery M (stąd nazwa `em`). W specyfikacji CSS `em` jest obliczany jako odległość między liniami bazowymi, gdy czcionka jest ustawiona bez dodatkowego odstępu między wierszami (tzw. interlinia). W przypadku tekstu z czcionką o rozmiarze 16 pikseli, `em` mierzy 16 pikseli; dla tekstu 12-pikselowego `em` równa się 12 pikselom;



Po obliczeniu przez przeglądarkę wymiaru `em` dla elementu tekstu można go użyć do wielu innych pomiarów, takich jak wcięcia, marginesy, szerokość elementu na stronie i tak dalej. Oparcie pomiarów na rozmiarze tekstu pomaga zachować proporcje w przypadku zmiany rozmiaru tekstu. Sztuczka do pracy z emami polega na tym, aby pamiętać, że są one zawsze związane z bieżącym rozmiarem czcionki elementu. Jeśli ustawimy `2em` lewy margines na `h1`, `h2` i `p`, te elementy nie będą się dobrze układać, ponieważ jednostki `em` są oparte na ich odpowiednich rozmiarach elementu.



The screenshot shows a heading "This is a 24pt Heading" and a paragraph "A Heading in 20pt" with some placeholder text. The heading has a margin-left of 2em, which is visually represented by a wider left margin compared to the paragraph's margin-left of 1em.

```
h1, h2, p { margin-left: 2em; }
```

Jednostki `viewport width (vw)` i `viewport height (vh)` są zależne od rozmiaru okna przeglądarki. `vw` jest równe 1/100 szerokości. Podobnie, `vh` jest równe 1/100 wysokości. Jednostki te są przydatne do tworzenia obrazów i elementów tekstowych na pełnej szerokości lub wysokości widocznego obszaru.

```
header {  
width: 100vw;  
height: 100vh; }
```

Łatwo jest również określić jednostkę jako określony procent rozmiaru okna, na przykład 50%

```
img {  
width: 50vw;  
height: 50vh; }
```

Powiązane są jednostki `vmin` (równe wartości `vw` lub `vh`, w zależności od tego, która jest mniejsza) i `vmax` (równe wartości `vw` lub `vh`, w zależności od tego, która jest większa). Więcej informacji o jednostkach znajduje się w module **CSS Values and Units** (www.w3.org/TR/css3-values/ (<https://www.w3.org/TR/css3-values/>)). Oprócz jednostek długości zawiera wartości tekstowe (takie jak słowa kluczowe, ciągi tekstowe i adresy URL), liczby i wartości procentowe, kolory i inne.

```
<!--units_example2.html-->  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <title>Units example 2</title>  
</head>  
<body>  
    <h1 style="border: 1rem solid blue">1 rem</h1>  
    <h1 style="border: 1em solid blue">1 em</h1>  
    <h1 style="border: 1vh solid blue">1 vh</h1>  
    <h1 style="border: 1vw solid blue">1 vw</h1>  
    <h1 style="width: 50vw; height: 50vh; border: 10px solid blue">50vw and 50vh</h1>  
</body>  
</html>
```

1 rem

1 em

1 vh

1 vw

50vw and 50vh

Step 4

Każdej właściwości CSS towarzyszą informacje o tym, jak się ona zachowuje i jak jej używać. Wykaz obejmuje:

- **Wartości** - Są to akceptowane wartości dla właściwości. Wstępnie zdefiniowane wartości słów kluczowych są wyświetlane czcionką kodu (na przykład `small`, `italic` lub `small-caps`) i należy je wpisywać dokładnie tak, jak pokazano.
- **Wartość domyślna** - jest to wartość, która będzie domyślnie używana dla właściwości (jej wartość początkowa) – to znaczy, jeśli nie określono żadnej innej wartości. Pamiętajmy, że domyślne wartości arkusza stylów przeglądarki mogą różnić się od domyślnych zdefiniowanych w CSS.
- **Dotyczy** - Niektóre właściwości dotyczą tylko niektórych typów elementów.
- **Dziedziczność** - Wskazuje, czy właściwość jest przekazywana potomkom elementu.

Wszystkie właściwości CSS akceptują trzy słowa kluczowe dla całego CSS: `initial`, `inherit` i `unset`. Ponieważ są one wspólne dla wszystkich właściwości, nie są wymienione z wartościami dla poszczególnych wykazów właściwości.

- Słowo kluczowe `initial` jawnie ustawia właściwość na jej wartość domyślną (początkową).
- Słowo kluczowe `inherit` pozwala na jawnie wymuszenie dziedziczenia przez element właściwości stylu z jego rodzica. Może się to przydać do zastąpienia innych stylów zastosowanych do tego elementu i zagwarantowania, że element zawsze pasuje do swojego rodzica.
- `unset` usuwa zadeklarowane wartości występujące wcześniej w kaskadzie, ustawiając właściwość na dziedziczenie (`inherit`) lub początkową (`initial`), w zależności od tego, czy dziedziczy, czy nie

Kiedy projektujemy dokument tekstowy (do druku lub do Internetu), jedną z pierwszych rzeczy, które robimy, jest **określenie czcionki**. W CSS czcionki są określane za pomocą zestawu właściwości związanych z czcionką dla kroju czcionki, rozmiaru, grubości, stylu czcionki i znaków specjalnych. Istnieją również właściwości skrótów, które pozwalają określić wiele atrybutów czcionki w jednej regule. Wybór kroju lub rodziny czcionek, jak to się nazywa w CSS, jest dobrym miejscem do rozpoczęcia. Zaczniemy od właściwości `font-family` i jej wartości.

`font-family`

Wartości: co najmniej jedna czcionka lub nazwa rodziny czcionek rozdzielonych przecinkami

Domyślnie: zależy od przeglądarki

Dotyczy: wszystkich elementów

Dziedziczy: tak

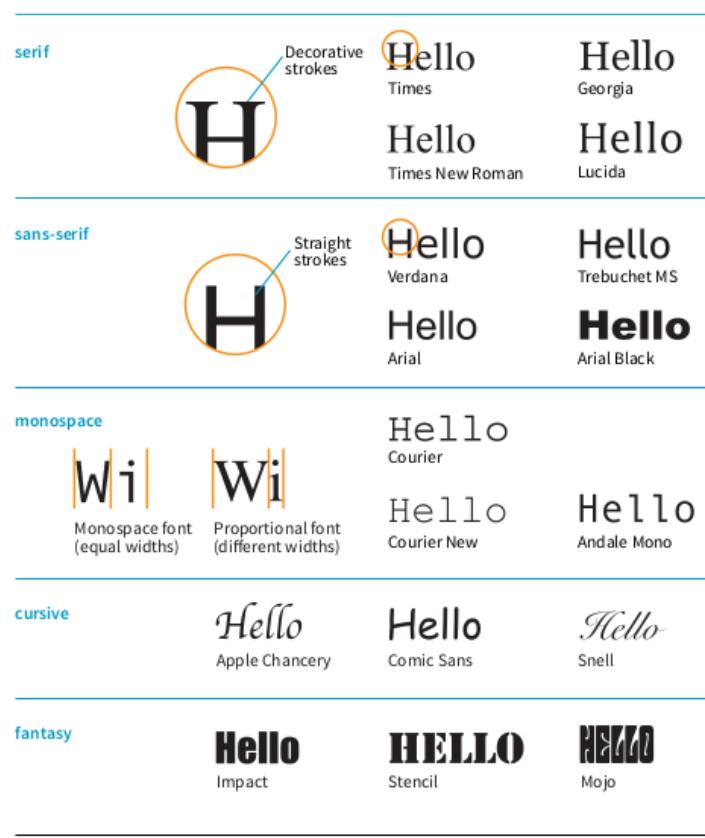
Właściwość `font-family` określa czcionkę lub listę czcionek (znaną jako stos czcionek) według nazwy, jak pokazano w poniższych przykładach:

```
body { font-family: Arial; }
var { font-family: Courier, monospace; }
p { font-family: "Duru Sans", Verdana, sans-serif; }
```

Oto kilka ważnych wymagań dotyczących składni:

- Wszystkie nazwy czcionek, z wyjątkiem ogólnych rodzin czcionek, muszą się rozpoczynać wielką literą. Na przykład `Arial` zamiast `arial`.
- Używaj przecinków, aby oddzielić wiele nazw czcionek, jak pokazano w drugim i trzecim przykładzie.
- Jeśli nazwa czcionki zawiera spację (np. `Duru Sans` w trzecim przykładzie), to musi znajdować się w cudzysłowie.

Przeglądarki ograniczają się do wyświetlania czcionek, do których mają dostęp. Tradycyjnie oznaczało to czcionki, które były już zainstalowane na dysku twardym użytkownika. Jednak w 2010 r. nastąpił boom w obsłudze wbudowanych czcionek internetowych w przeglądarkach przy użyciu reguły `CSS @font-face`, dzięki czemu projektanci mogli tworzyć własne czcionki. Wróćmy jednak do naszej reguły dotyczącej rodzin czcionek. Nawet jeśli określmy, że czcionka powinna być `Futura` w regule stylu, jeśli przeglądarka nie może jej znaleźć (na przykład, jeśli ta czcionka nie jest zainstalowana na komputerze użytkownika lub podana czcionka internetowa nie może się załadować), przeglądarka używa zamiast tego jego domyślną czcionkę. Na szczęście `CSS` pozwala nam dostarczyć listę czcionek zapasowych, gdyby nasz pierwszy wybór nie był dostępny. Jeśli pierwsza określona czcionka nie zostanie znaleziona, przeglądarka spróbuje użyć następnej i prześledzić listę, aż znajdzie taką, która działa. W trzeciej regule rodziny czcionek pokazanej w poprzednim przykładzie, jeśli przeglądarka nie znajdzie `Duru Sans`, użyje `Verdana`, a jeśli `Verdana` nie jest dostępna, zastąpi inną czcionką bezszeryfową. Ta ostatnia opcja, „inna czcionka bezszeryfowa”, budzi więcej dyskusji. `sans-serif` to tylko jedna z pięciu ogólnych rodzin czcionek, które można określić za pomocą właściwości `font-family`. Po określeniu ogólnej rodziny czcionek przeglądarka wybiera dostępną czcionkę z tej kategorii stylistycznej. Poniższy rysunek przedstawia przykłady z każdej rodziny.



- `serif` - Kroje pisma szerygowego mają ozdobne wypustki przypominające płytę (szeryfy) na końcach niektórych pociągnięć liter.
- `sans-serif` - Czcionki bezszeryfowe mają proste pociągnięcia liter, które nie kończą się szeryfami.
- `monospace` - W krojach pisma o stałej szerokości (zwanej również stałą szerokością) wszystkie znaki zajmują taką samą ilość miejsca w wierszu. Na przykład wielkie W nie będzie szersze niż małe i.
- `cursive` - Czcionki kursywne emulują wygląd pisma lub pisma odręcznego.
- `fantasy` - Czcionki fantasy są czysto dekoracyjne i nadają się do nagłówków i innych rodzajów prezentacji.

Najlepszą praktyką przy określaniu czcionek dla stron internetowych jest rozpoczęcie od pierwszego wyboru, zapewnienie kilku podobnych alternatyw, a następnie założenie ogólnej rodziny czcionek, która przynajmniej zapewni użytkownikom odpowiedni styl. Na przykład, jeśli potrzebujemy czcionki pionowej, bezszeryfowej, możemy zacząć od czcionki internetowej, jeśli ją podamy (`Oswald`),

wymienimy też kilka, które są bardziej popularne (Univers , Tahoma , Geneva) i zakończymy ogólną czcionką bezszeryfową. Nie ma ograniczeń co do liczby czcionek, które można dołączyć, ale wielu projektantów stara się, aby była mniejsza niż 10. Dobry stos czcionek powinien zawierać stylistycznie powiązane czcionki, o których wiadomo, że są instalowane na większości komputerów. Pozostawienie przy czcionkach dostarczanych z systemami operacyjnymi Windows, macOS i Linux, a także czcionkach instalowanych z popularnymi pakietami oprogramowania, takimi jak Microsoft Office i Adobe Creative Suite, daje solidną listę „bezpiecznych w Internecie” czcionek do wyboru z. Dobrym miejscem do szukania stylistycznie powiązanych czcionek bezpiecznych w sieci jest **CSS Font Stack** (<https://cssfontstack.com> (<https://cssfontstack.com>)).

Na potrzeby przykładów będziemy korzystać z pliku HTML:

```
<!--menu.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Black Goose Bistro Summer Menu</title>
    <style>
        </style>
</head>

<body>

<div id="info">
    <h1>Black Goose Bistro &bull; Summer Menu</h1>

    <p>Baker's Corner, Seekonk, Massachusetts<br>
        <span class="label">Hours: Monday through Thursday:</span> 11 to 9, <span class="label">Friday and Saturday;</span> 11 to midnight</p>
</div>

<div id="appetizers">
    <h2>Appetizers</h2>
    <p>This season, we explore the spicy flavors of the southwest in our appetizer collection.</p>

    <dl>
        <dt>Black bean purses</dt>
        <dd>Spicy black bean and a blend of mexican cheeses wrapped in sheets of phyllo and baked until golden. <span class="price">$3.95</span></dd>

        <dt class="newitem">Southwestern napoleons with lump crab &mdash; <strong>new item!</strong></dt>
        <dd>Layers of light lump crab meat, bean and corn salsa, and our handmade flour tortillas. <span class="price">$7.95</span></dd>
    </dl>
</div>

<div id="entrees">

    <h2>Main courses</h2>
    <p>Big, bold flavors are the name of the game this summer. Allow us to assist you with finding the perfect wine.</p>

    <dl>
        <dt class="newitem">Jerk rotisserie chicken with fried plantains &mdash; <strong>new item!</strong></dt>
        <dd>Tender chicken slow-roasted on the rotisserie, flavored with spicy and fragrant jerk sauce and served with fried plantains and fresh mango. <strong>Very spicy.</strong> <span class="price">$12.95</span></dd>

        <dt>Shrimp sate kebabs with peanut sauce</dt>
        <dd>Skewers of shrimp marinated in lemongrass, garlic, and fish sauce then grilled to perfection. Served with spicy peanut sauce and jasmine rice. <span class="price">$12.95</span></dd>

        <dt>Grilled skirt steak with mushroom fricassee</dt>
        <dd>Flavorful skirt steak marinated in Asian flavors grilled as you like it<sup>*</sup>. Served over a blend of sauteed wild mushrooms with a side of blue cheese mashed potatoes. <span class="price">$16.95</span></dd>
    </dl>
</div>

<p class="warning"><sup>*</sup> We are required to warn you that undercooked food is a health risk.</p>

</body>
</html>
```

Black Goose Bistro • Summer Menu

Baker's Corner, Seekonk, Massachusetts

Hours: Monday through Thursday: 11 to 9, Friday and Saturday; 11 to midnight

Appetizers

This season, we explore the spicy flavors of the southwest in our appetizer collection.

Black bean purses

Spicy black bean and a blend of mexican cheeses wrapped in sheets of phyllo and baked until golden. \$3.95

Southwestern napoleons with lump crab — **new item!**

Layers of light lump crab meat, bean and corn salsa, and our handmade flour tortillas. \$7.95

Main courses

Big, bold flavors are the name of the game this summer. Allow us to assist you with finding the perfect wine.

Jerk rotisserie chicken with fried plantains — **new item!**

Tender chicken slow-roasted on the rotisserie, flavored with spicy and fragrant jerk sauce and served with fried plantains and fresh mango. **Very spicy.** \$12.95

Shrimp sate kebabs with peanut sauce

Skewers of shrimp marinated in lemongrass, garlic, and fish sauce then grilled to perfection. Served with spicy peanut sauce and jasmine rice. \$12.95

Grilled skirt steak with mushroom fricassee

Flavorful skirt steak marinated in Asian flavors grilled as you like it*. Served over a blend of sauteed wild mushrooms with a side of blue cheese mashed potatoes. \$16.95

* We are required to warn you that undercooked food is a health risk.

Utwórzmy plik menu.css o następującej zawartości:

```
body {  
    font-family: Verdata, sans-serif;  
}  
  
h1 {  
    font-family: "Marko One", Georgia, serif;  
}
```

Widzimy, że chcemy użyć nietypowej czcionki o nazwie Marko One, pobierzmy ją do naszego pliku za pomocą znacznika link oraz strony <http://fonts.googleapis.com> (<http://fonts.googleapis.com>) (niedostępna bez podawania parametrów).

```

<!--menu.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Black Goose Bistro Summer Menu</title>
  <link href="http://fonts.googleapis.com/css?family=Marko+One" rel="stylesheet">
  <link href="menu.css" rel="stylesheet" type="text/css">
</head>

<body>

<div id="info">
  <h1>Black Goose Bistro &bull; Summer Menu</h1>

  <p>Baker's Corner, Seekonk, Massachusetts<br>
    <span class="label">Hours: Monday through Thursday:</span> 11 to 9, <span class="label">Friday and Saturday;</span> 11 to midnight</p>
</div>

<div id="appetizers">
  <h2>Appetizers</h2>
  <p>This season, we explore the spicy flavors of the southwest in our appetizer collection.</p>

  <dl>
    <dt>Black bean purses</dt>
    <dd>Spicy black bean and a blend of mexican cheeses wrapped in sheets of phyllo and baked until golden. <span class="price">$3.95</span></dd>

    <dt class="newitem">Southwestern napoleons with lump crab &mdash; <strong>new item!</strong></dt>
    <dd>Layers of light lump crab meat, bean and corn salsa, and our handmade flour tortillas. <span class="price">$7.95</span></dd>
  </dl>
</div>

<div id="entrees">

  <h2>Main courses</h2>
  <p>Big, bold flavors are the name of the game this summer. Allow us to assist you with finding the perfect wine.</p>

  <dl>
    <dt class="newitem">Jerk rotisserie chicken with fried plantains &mdash; <strong>new item!</strong></dt>
    <dd>Tender chicken slow-roasted on the rotisserie, flavored with spicy and fragrant jerk sauce and served with fried plantains and fresh mango. <strong>Very spicy.</strong> <span class="price">$12.95</span></dd>

    <dt>Shrimp sate kebabs with peanut sauce</dt>
    <dd>Skewers of shrimp marinated in lemongrass, garlic, and fish sauce then grilled to perfection. Served with spicy peanut sauce and jasmine rice. <span class="price">$12.95</span></dd>

    <dt>Grilled skirt steak with mushroom fricassee</dt>
    <dd>Flavorful skirt steak marinated in Asian flavors grilled as you like it<sup>*</sup>. Served over a blend of sauteed wild mushrooms with a side of blue cheese mashed potatoes. <span class="price">$16.95</span></dd>
  </dl>
</div>

<p class="warning"><sup>*</sup> We are required to warn you that undercooked food is a health risk.</p>
</body>
</html>

```

Gdy klikniemy na link <http://fonts.googleapis.com/css?family=Marko+One> (<http://fonts.googleapis.com/css?family=Marko+One>) otrzymujemy następującą zawartość:

```
/* latin */
@font-face {
    font-family: 'Marko One';
    font-style: normal;
    font-weight: 400;
    src: url(https://fonts.gstatic.com/s/markoone/v22/9Btq3DFG0cnVM5lw1haqLZ8e.woff2) format('woff2');
    unicode-range: U+0000-00FF, U+0131, U+0152-0153, U+02BB-02BC, U+02C6, U+02DA, U+02DC, U+2000-206F,
    U+2074, U+20AC, U+2122, U+2191, U+2193, U+2212, U+2215, U+FEFF, U+FFFFD;
}
```

W rezultacie nasza strona wygląda następująco:



Aby określić rozmiar tekstu należy skorzystać z właściwości `font-size`:

```
font-size

Wartości: jednostka referencyjna | procent | xx-small | x-small | small | medium | large |
           x-large | xx-large | smaller | larger
Domyślnie: medium
Dotyczy: wszystkich elementów
Dziedziczy: tak
```

Rozmiar tekstu można określić na kilka sposobów:

- Używając jednej z jednostek długości `CSS`, jak pokazano tutaj:

```
h1 { font-size: 1.5em; }
```

Określając liczbę jednostek, upewnijmy się, że skrót jednostki następuje bezpośrednio po liczbie, bez dodatkowego odstępu między znakami.

- Jako wartość procentowa, powiększona lub zmniejszona w stosunku do odziedziczonego rozmiaru czcionki elementu:

```
h1 { font-size: 150%; }
```

- Użycie jednego z bezwzględnych słów kluczowych (`xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`). W większości obecnych przeglądarek średnia odpowiada domyльнemu rozmiarowi czcionki.

```
h1 { font-size: x-large; }
```

- Użycie względnego słowa kluczowego (`larger` lub `smaller`) w celu przesunięcia tekstu większego lub mniejszego niż otaczający go tekst:

```
strong { font-size: larger; }
```

Pomimo tych wszystkich opcji preferowanymi wartościami rozmiaru czcionki we współczesnym projektowaniu stron internetowych są jednostki długości względnej `em` i `rem`, a także **wartości procentowe**. Możemy określić rozmiar czcionki w pikselach (`px`), ale generalnie nie zapewniają one elastyczności wymaganej przy projektowaniu stron internetowych. Wszystkie inne jednostki bezwzględne (`pt`, `pc`, `in` itd.) również są nieaktualne, chyba że tworzymy arkusz stylów specjalnie do druku.

Najlepszą praktyką przy ustawianiu rozmiaru czcionki elementów strony internetowej jest robienie tego w sposób respektujący preferencje użytkownika. Względne wartości rozmiaru `%`, `rem` i `em` pozwalają na użycie domyślnego rozmiaru czcionki jako podstawy proporcjonalnego rozmiaru innych elementów tekstu. Zwykle nie ma znaczenia, że nagłówki mają dokładnie 24 piksele; ważne jest, aby były 1,5 raza większa niż tekst główny, żeby się wyróżniały. Jeśli użytkownik zmieni swoje preferencje, aby zwiększyć domyślny rozmiar czcionki, nagłówki również będą większe. Aby zachować domyślny rozmiar przeglądarki, należy ustawić rozmiar czcionki elementu głównego na 100%.

```
html {  
    font-size: 100%;  
}
```

To stanowi podstawę do względnego rozmiaru.

Jednostka `rem`, która oznacza „root em”, jest zawsze zależna od rozmiaru elementu głównego (`html`). Jeśli rozmiar główny wynosi 16 pikseli, to `rem` wynosi 16 pikseli. To, co jest miłe w jednostkach `rem`, to to, że zawsze odnoszą się do tego samego elementu, mają ten sam rozmiar, niezależnie od tego, gdzie są używane w całym dokumencie. W ten sposób działają jak absolutna jednostka. Jeśli jednak rozmiar główny będzie inny niż 16 pikseli, elementy określone w wartościach `rem` zmienią rozmiar odpowiednio i proporcjonalnie. Oto ten sam nagłówek o rozmiarze z wartościami `rem`:

```
/*menu.css*/  
body {  
    font-family: Verdata, sans-serif;  
}  
  
h1 {  
    font-family: "Marko One", Georgia, serif;  
    font-size: 1.5rem;  
}
```

Black Goose Bistro • Summer Menu

Baker's Corner, Seekonk, Massachusetts

Hours: Monday through Thursday: 11 to 9, Friday and Saturday; 11 to midnight

Appetizers

This season, we explore the spicy flavors of the southwest in our appetizer collection.

Black bean purses

Spicy black bean and a blend of mexican cheeses wrapped in sheets of phyllo and baked until golden. \$3.95

Southwestern napoleons with lump crab — **new item!**

Layers of light lump crab meat, bean and corn salsa, and our handmade flour tortillas. \$7.95

Main courses

Big, bold flavors are the name of the game this summer. Allow us to assist you with finding the perfect wine.

Jerk rotisserie chicken with fried plantains — **new item!**

Tender chicken slow-roasted on the rotisserie, flavored with spicy and fragrant jerk sauce and served with fried plantains and fresh mango. **Very spicy.** \$12.95

Shrimp sate kebabs with peanut sauce

Skewers of shrimp marinated in lemongrass, garlic, and fish sauce then grilled to perfection. Served with spicy peanut sauce and jasmine rice. \$12.95

Grilled skirt steak with mushroom fricassee

Flavorful skirt steak marinated in Asian flavors grilled as you like it*. Served over a blend of sauteed wild mushrooms with a side of blue cheese mashed potatoes. \$16.95

* We are required to warn you that undercooked food is a health risk.

Jednostki `Em` są oparte na rozmiarze czcionki bieżącego elementu. Kiedy określmy rozmiar czcionki w emach, będzie on zależny od odziedziczonego rozmiaru tego elementu. Po obliczeniu `em` dla elementu można go również użyć do innych pomiarów, takich jak marginesy, dopełnienie, szerokości elementów i wszelkie inne ustawienia, które chcemy zawsze uwzględniać w stosunku do rozmiaru czcionki.

```
/*menu.css*/
body {
    font-family: Verdata, sans-serif;
}

h1 {
    font-family: "Marko One", Georgia, serif;
    font-size: 1.5em;
}
```

Black Goose Bistro • Summer Menu

Baker's Corner, Seekonk, Massachusetts

Hours: Monday through Thursday: 11 to 9, Friday and Saturday; 11 to midnight

Appetizers

This season, we explore the spicy flavors of the southwest in our appetizer collection.

Black bean purses

Spicy black bean and a blend of mexican cheeses wrapped in sheets of phyllo and baked until golden. \$3.95

Southwestern napoleons with lump crab — **new item!**

Layers of light lump crab meat, bean and corn salsa, and our handmade flour tortillas. \$7.95

Main courses

Big, bold flavors are the name of the game this summer. Allow us to assist you with finding the perfect wine.

Jerk rotisserie chicken with fried plantains — **new item!**

Tender chicken slow-roasted on the rotisserie, flavored with spicy and fragrant jerk sauce and served with fried plantains and fresh mango. **Very spicy.** \$12.95

Shrimp sate kebabs with peanut sauce

Skewers of shrimp marinated in lemongrass, garlic, and fish sauce then grilled to perfection. Served with spicy peanut sauce and jasmine rice. \$12.95

Grilled skirt steak with mushroom fricassee

Flavorful skirt steak marinated in Asian flavors grilled as you like it*. Served over a blend of sauteed wild mushrooms with a side of blue cheese mashed potatoes. \$16.95

* We are required to warn you that undercooked food is a health risk.

Jest kilka problemów związanych z pracą z `emami`. Jednym z nich jest to, że z powodu błędów zaokrąglania istnieje pewna niespójność w sposobie, w jaki przeglądarki i platformy renderują tekst ustawiony w `emach`. Innym trudnym aspektem używania `emów` jest to, że są one oparte na dziedziczym rozmiarze elementu, co oznacza, że ich rozmiar jest oparty na kontekście, w którym są stosowane. `H1` w poprzednim przykładzie był oparty na odziedziczym rozmiarze 16 pikseli. Ale gdyby to `h1` pojawiło się w elemencie artykułu, którego rozmiar czcionki był ustawiony na 14 pikseli, odziedziczyłby rozmiar 14 pikseli, a wynikowy rozmiar wynosiłby tylko 21 pikseli ($1.5 \times 14 = 21$).

Widzieliśmy wartość procentową (`100%`) używaną do zachowania domyślnego rozmiaru czcionki, ale możemy użyć wartości procentowych dla dowolnego elementu. Są całkiem proste. W tym przykładzie `h1` dziedziczy domyślny rozmiar `16px` z elementu `html`, a zastosowanie `150%` wartości mnoży tę odziedziczoną wartość, dając w wyniku `h1` o 24 pikselach:

```
/*menu.css*/
body {
    font-family: Verdata, sans-serif;
}

h1 {
    font-family: "Marko One", Georgia, serif;
    font-size: 150%;
}
```

Black Goose Bistro • Summer Menu

Baker's Corner, Seekonk, Massachusetts

Hours: Monday through Thursday: 11 to 9, Friday and Saturday; 11 to midnight

Appetizers

This season, we explore the spicy flavors of the southwest in our appetizer collection.

Black bean purses

Spicy black bean and a blend of mexican cheeses wrapped in sheets of phyllo and baked until golden. \$3.95
Southwestern napoleons with lump crab — **new item!**

Layers of light lump crab meat, bean and corn salsa, and our handmade flour tortillas. \$7.95

Main courses

Big, bold flavors are the name of the game this summer. Allow us to assist you with finding the perfect wine.

Jerk rotisserie chicken with fried plantains — **new item!**

Tender chicken slow-roasted on the rotisserie, flavored with spicy and fragrant jerk sauce and served with fried plantains and fresh mango. **Very spicy.** \$12.95

Shrimp sate kebabs with peanut sauce

Skewers of shrimp marinated in lemongrass, garlic, and fish sauce then grilled to perfection. Served with spicy peanut sauce and jasmine rice. \$12.95

Grilled skirt steak with mushroom fricassee

Flavorful skirt steak marinated in Asian flavors grilled as you like it*. Served over a blend of sauteed wild mushrooms with a side of blue cheese mashed potatoes. \$16.95

* We are required to warn you that undercooked food is a health risk.

Alternatywnym sposobem określenia rozmiaru czcionki jest użycie jednego z predefiniowanych bezwzględnych słów kluczowych: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large` i `xx-large`. Słowa kluczowe nie odpowiadają poszczególnym pomiarom, ale są skalowane w sposób spójny względem siebie. W obecnych przeglądarkach domyślny rozmiar to `medium`. Poniżej pokazano, jak każde z bezwzględnych słów kluczowych jest renderowane w przeglądarce, gdy domyślny tekst jest ustawiony na 16 pikseli. W przypadku użycia różnych czcionek można zaobserwować różnicę w użyciu tych rozmiarów.

```
<!--keywords_size.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Keywords size example</title>
</head>
<body>
    <h1>Example of default text size in Verdana/serif font</h1>
    <p style="font-family: Verdana,serif; font-size:xx-small">This is some text.</p>
    <p style="font-family: Verdana,serif; font-size:x-small">This is some text.</p>
    <p style="font-family: Verdana,serif; font-size:small">This is some text.</p>
    <p style="font-family: Verdana,serif; font-size:medium">This is some text.</p>
    <p style="font-family: Verdana,serif; font-size:large">This is some text.</p>
    <p style="font-family: Verdana,serif; font-size:x-large">This is some text.</p>
    <p style="font-family: Verdana,serif; font-size:xx-large">This is some text.</p>
</body>
</html>
```

Example of default text size in Verdana/serif font

This is some text.

This is some text.

Względne słowa kluczowe, `larger` i `smaller`, są używane do zmiany rozmiaru tekstu względem rozmiaru tekstu elementu nadzawanego. Dokładna wielkość zmiany rozmiaru jest określana przez każdą przeglądarkę i jest poza naszą kontrolą. Pomimo tego ograniczenia, jest to łatwy sposób na nieco większe lub mniejsze litery, jeśli dokładne proporcje nie są krytyczne.

Step 5

Po rodzinach czcionek i rozmiarze pozostałe właściwości czcionki są proste. Na przykład, jeśli chcemy, aby element tekstowy był pogrubiony, należy właściwości `font-weight`, aby dostosować pogrubienie tekstu.

`font-weight`

Wartości: `normal` | `bold` | `bolder` | `lighter` | `100` | `200` | `300` | `400` | `500` | `600` | `700` | `800` | `900`

Domyślnie: `normal`

Dotyczy: Wszystkich elementów

Dziedziczy: tak

Jak widać, właściwość `font-weight` ma wiele predefiniowanych wartości, w tym terminy opisowe (`normal`, `bold`, `bolder` i `lighter`) oraz dziewięć wartości liczbowych (od 100 do 900) do określania różnych grubości czcionki, jeśli są one dostępne. Ponieważ większość czcionek powszechnie używanych w Internecie ma tylko dwie grubości, `normal` (lub `roman`) i `bold`, jedyną wartością grubości czcionki, której użyjemy w większości przypadków, jest `bold`. Możemy także użyć `normal`, aby tekst, który w przeciwnym razie byłby pogrubiony (np. mocny tekst lub nagłówki), miał normalną wagę. Wykres numeryczny może się przydać podczas korzystania z czcionek internetowych o dużym zakresie wag, aczkolwiek przeglądarki różnie mogą interpretować wagę, więc efekt może nie być identyczny.

```
<!--font_weight_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Font weight example</title>
</head>
<body>
    <p style="font-weight: normal">This is a default text.</p>
    <p style="font-weight: bold">This is a default text.</p>
    <p style="font-weight: bolder">This is a default text.</p>
    <p style="font-weight: lighter">This is a default text.</p>
    <p style="font-weight: 100">This is a default text.</p>
    <p style="font-weight: 200">This is a default text.</p>
    <p style="font-weight: 300">This is a default text.</p>
    <p style="font-weight: 400">This is a default text.</p>
    <p style="font-weight: 500">This is a default text.</p>
    <p style="font-weight: 600">This is a default text.</p>
    <p style="font-weight: 700">This is a default text.</p>
    <p style="font-weight: 800">This is a default text.</p>
    <p style="font-weight: 900">This is a default text.</p>
</body>
</html>
```

This is a default text.

This is a default text.

This is a default text.

This is a default text.

This is a default text.

Właściwość `font-style` wpływa na ułożenie tekstu – to znaczy, czy litery są pionowe (normalne), czy pochyłe (kursywa i ukośne).

font-style

Wartości: `normal` | `italic` | `oblique`

Domyślnie: `normal`

Dotyczy: Wszystkich elementów

Dziedziczy: tak

Innym powszechnym zastosowaniem jest sprawienie, by tekst pisany kursywą w domyślnych stylach przeglądarki (np. tekst wyróżniony) był wyświetlany w normalny sposób. Istnieje wartość `oblique`, która określa pochyloną wersję czcionki; jednak przeglądarki generalnie wyświetlają `oblique` dokładnie tak samo jak `italic`.

```
<!--font_style_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Font style example</title>
</head>
<body>
    <p style="font-style: normal">This is a default text.</p>
    <p style="font-style: oblique">This is a default text.</p>
    <p style="font-style: italic">This is a default text.</p>
</body>
</html>
```

This is a default text.

This is a default text.

This is a default text.

Określanie wielu właściwości czcionki dla każdego elementu tekstowego może być powtarzalne i długie, więc twórcy CSS dostarczyli skróconą właściwość `font`, która łączy wszystkie właściwości związane z czcionkami w jedną regułę.

font

Wartości: `font-style font-weight font-variant font-stretch font-size/line-height font-family | caption | icon | menu | message-box | small-caption | status-bar`

Domyślnie: zależy od właściwości

Dotyczy: Wszystkich elementów

Dziedziczy: tak

Wartość właściwości `font` jest listą wartości wszystkich właściwości fontu, które właśnie przyjrzaliśmy, oddzielonych spacjami znakowymi. W tej właściwości kolejność wartości jest ważna.

```
{ font: style weight stretch variant size/line-height font-family; }
```

Jako minimum, właściwość `font` musi zawierać wartość rozmiaru czcionki i wartość rodziny fontów, w tej kolejności. Pominiecie jednego lub umieszczenie ich w złej kolejności powoduje, że cała reguła jest nieważna. Oto przykład minimalnej wartości właściwości czcionki:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Font example</title>
</head>
<body>
    <p style="font: 1em sans-serif;">This is a default text</p>
</body>
</html>
```

This is a default text

Po spełnieniu wymagań dotyczących rozmiaru i rodziny pozostałe wartości są opcjonalne i mogą pojawiać się w dowolnej kolejności przed rozmiarem czcionki. Gdy styl, waga, rozciągnięcie lub wariant zostaną pominięte, ich wartość zostanie ustawiona na normalną. Ułatwia to przypadkowe nadpisanie poprzedniego ustawienia właściwością skróconą, więc należy zachować ostrożność podczas jej używania. Jest tam jedna wartość, `line-height`, której jeszcze nie widzieliśmy. Jak się wydaje, dostosowuje wysokość wiersza tekstu i służy do dodawania odstępu między wierszami tekstu. Pojawia się tuż po rozmiarze czcionki, oddzielony ukośnikiem.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Font example</title>
</head>
<body>
    <p style="font: 1em sans-serif;">This is a default text</p>
    <h3 style="font: oblique bold small-caps 1.5em/1.8em sans-serif;">This is a header</h3>
</body>
</html>
```

This is a default text

THIS IS A HEADER

Właściwość `font` zawiera również kilka wartości słów kluczowych (`caption`, `icon`, `menu`, `message-box`, `small-caption` oraz `status-bar`), które reprezentują czcionki systemowe, czcionki używane przez systemy operacyjne do takich rzeczy, jak etykiety ikon i elementy menu. Mogą być przydatne, gdy projektujemy aplikację internetową, aby pasowała do środowiska, w którym pracuje użytkownik. Są to wartości skrócone, ponieważ zawierają one czcionkę, rozmiar, styl i wagę czcionki używanej do każdego celu za pomocą tylko jednego słowa kluczowego.

Teraz mamy dobry podstawowy zestaw narzędzi do formatowania czcionek za pomocą `CSS`. Jeśli chcemy dowiedzieć się więcej, to należy przeczytać o wszystkich właściwościach w module `CSS Fonts Level 3`, który daje znacznie większą kontrolę nad wyborem i pozycją znaków. Więcej informacji pod adresem <https://www.w3.org/TR/css-fonts-3/> (<https://www.w3.org/TR/css-fonts-3/>)

Zbiór właściwości `font-variant-` w `CSS3` ma na celu zapewnienie projektantom i programistom dostępu do znaków specjalnych (glifów) w czcionkach, które mogą uczynić typografię na stronie bardziej wyrafinowaną. Jak wspomniano wcześniej, `CSS3 Font Module` znacznie rozszerzył definicję `font-variant`. Teraz może służyć jako skrócona właściwość dla wielu właściwości z przedrostkiem `font-variant`. Te właściwości są nadal uważane za eksperymentalne, chociaż obsługa przeglądarek zaczyna się zwiększać. Mimo to ciekawie jest zobaczyć, jak ewoluje kontrola czcionek w projektowaniu stron internetowych. Trzeba pamiętać, że tylko niektóre z poniższych właściwości działa na wszystkich przeglądarkach.

Ligatura (`font-variant-ligatures`) to glif, który łączy dwa lub więcej znaków w jeden symbol. Jednym z typowych przykładów jest kombinacja małych liter `f` oraz `i`, gdzie kropka nad `i` staje się częścią `f` (`fi`). Ligatury mogą wygładzić wygląd znanych niezręcznych par liter, a wiele czcionek zawiera glify ligatur. Właściwość `font-variant-ligatures` umożliwia kontrolowanie użycia ligatur na stronach internetowych. Przykład: <https://developer.mozilla.org/en-US/docs/Web/CSS/font-variant-ligatures> (<https://developer.mozilla.org/en-US/docs/Web/CSS/font-variant-ligatures>)

`font-variant-caps` umożliwia wybór glifów kapitalików (kapitału) z zestawu znaków czcionki zamiast symulowania ich w przeglądarce. Wartość `all-small-caps` używa wielkich i małych literach. `unicase` używa małych kapitalików tylko do wielkich liter, a małe litery w słowie pozostają takie same. `titling-caps` jest używany w tytułach z wielkimi literami, ale ma być mniej mocny. Inne opcje to `petite-caps` i `all-petite-caps`. Przykład: <https://developer.mozilla.org/en-US/docs/Web/CSS/font-variant-caps> (<https://developer.mozilla.org/en-US/docs/Web/CSS/font-variant-caps>)

`font-variant-position` wybiera glify indeksu górnego (górnego) lub indeksu dolnego (podrzędnego) z zestawu znaków czcionki, jeśli są one dostępne. W przeciwnym razie przeglądarka tworzy tekst w indeksie górnym lub dolnym dla elementów `sup` i `sub`, zmniejszając znak i przesuwając go powyżej lub poniżej linii bazowej.

`font-variant-numeric` umożliwia wybór różnych stylów znaków liczbowych, jeśli są one dostępne. Na przykład możemy wybrać cyfry proporcjonalne lub ułożyć się w kolumnach, tak jak w arkuszu kalkulacyjnym (`proportional-numbers` / `tabular-numbers`), wybrać cyfry w starym stylu (`old-style-nums`), w których niektóre znaki zanurzają się poniżej linii bazowej i określić, czy ułamki powinny być ułożone po przekątnej, czy ułożone w stos (`diagonal-fractions` / `stacked-fractions`). Pozwala także na sprawienie, by liczby porządkowe wyglądały z indeksem górnym zamiast drugiej (`ordinal`) i daje możliwość używania zer z ukośnikami, co jest preferowane w niektórych kontekstach (`slashed-zero`). Przykład: <https://developer.mozilla.org/en-US/docs/Web/CSS/font-variant-numeric> (<https://developer.mozilla.org/en-US/docs/Web/CSS/font-variant-numeric>)

Czcionki czasami oferują więcej niż jeden glif dla określonej postaci – na przykład kilka pluszowych wzorów na literę S lub staromodne s, które wygląda bardziej jak f. `font-variant-alternates` umożliwia określenie znaków kaligraficznych i innych znaków alternatywnych. Wiele z jego wartości jest specyficznych dla czcionki i należy je najpierw zdefiniować za pomocą reguły `@font-features-values`.

Powysze właściwości można skrócić do postaci `font-variant`, podobnie jak w przypadku `font`.

Rozmiar tekstu, który wygląda na stronie, często ma więcej wspólnego z wysokością małej litery x (jej wysokość x) niż określony rozmiar tekstu. Na przykład czcionka 10-punktowa o stosunkowo dużej wysokości x jest prawdopodobnie łatwiejsza do odczytania niż czcionka 10-punktowa z delikatnymi małymi literami. Właściwość `font-size-adjust` umożliwia przeglądarce dostosowywanie rozmiaru czcionki zastępczej, dopóki jej wysokość x nie będzie odpowiadać wysokości x czcionki pierwszego wyboru. Może to zapewnić lepszą czytelność nawet w przypadku konieczności użycia czcionki zastępczej.

Kerning to spacja między glifami znaków. Czcionki zazwyczaj zawierają metadane o tym, które pary liter muszą być ze sobą zwarte, aby odstęp w słowie wyglądały spójnie. Właściwość `font-kerning` umożliwia zastosowanie informacji o kerningu czcionki (`normal`), wyłączenie (`none`) lub pozostawienie do uznania przeglądarki (`auto`). Przykład: <https://developer.mozilla.org/en-US/docs/Web/CSS/font-kerning> (<https://developer.mozilla.org/en-US/docs/Web/CSS/font-kerning>)

Właściwość `font-feature-settings` daje autorom możliwość kontrolowania zaawansowanych funkcji typograficznych w czcionkach OpenType, które nie są powszechnie używane, takich jak znaki kaligraficzne, kapitaliki, ligatury, automatyczne ułamki i inne. Te funkcje powinny wyglądać znajomo, ponieważ wiele z nich można kontrolować za pomocą różnych właściwości wariantów czcionki. W rzeczywistości specyfikacja zaleca używanie wariantu czcionki, gdy tylko jest to możliwe, i rezerwowanie ustawień funkcji czcionki dla skrajnych przypadków. Przykład: <https://developer.mozilla.org/en-US/docs/Web/CSS/font-feature-settings> (<https://developer.mozilla.org/en-US/docs/Web/CSS/font-feature-settings>)

Step 6

Kolor tekstu można zmienić za pomocą właściwości `color`.

`color`

Wartości: koloru (w postaci numerycznej lub słownej)

Domyślnie: zależy od ustawień przeglądarki lub użytkownika

Dotyczy: Wszystkich elementów

Dziedziczy: tak

Korzystanie z właściwości `color` jest bardzo proste. Wartością właściwości `color` może być predefiniowana nazwa koloru lub wartość liczbową opisującą określony kolor `RGB`. Oto kilka przykładów, z których wszystkie sprawiają, że elementy `h1` w dokumencie są szare.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Color example</title>
</head>
<body>
    <h1 style="color: gray">This is a gray color</h1>
    <h1 style="color: #666666">This is a gray color</h1>
    <h1 style="color: #666">This is a gray color</h1>
    <h1 style="color: rgb(102,102,102)">This is a gray color</h1>
</body>
</html>
```

This is a gray color
This is a gray color
This is a gray color
This is a gray color

Kolor jest dziedziczony, więc możemy zmienić kolor całego tekstu w dokumencie, stosując właściwość `color` do elementu `body`, jak pokazano tutaj.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Color example</title>
<style>
    body {
        color: red;
    }
</style>
</head>
<body>
    <h1 style="color: gray">This is a gray color</h1>
    <h1 style="color: #666666">This is a gray color</h1>
    <h1 style="color: #666">This is a gray color</h1>
    <h1 style="color: rgb(102,102,102)">This is a gray color</h1>
    <h1>This is a red color</h1>
</body>
</html>
```

This is a gray color
This is a red color

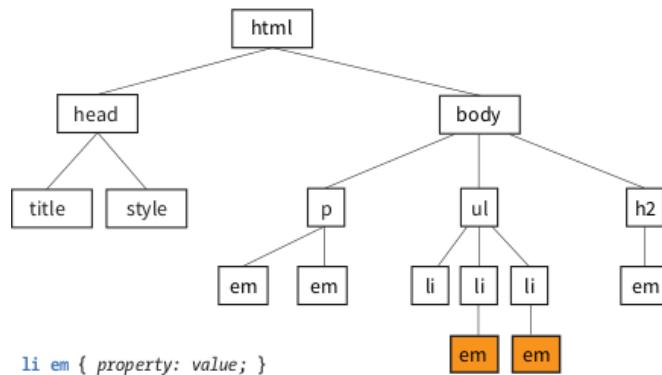
Właściwość `color` nie jest właściwościąścią związaną z tekstem. W rzeczywistości, zgodnie ze specyfikacją CSS, służy do zmiany koloru pierwszego planu (w przeciwieństwie do tła) elementu. Pierwszy plan elementu składa się zarówno z zawartego w nim tekstu, jak i jego obramowania. Tak więc, gdy zastosujemy `color` do elementu (w tym elementów obrazu), kolor zostanie również użyty do obramowania, chyba że istnieje określona właściwość koloru obramowania, która go zastępuje.

Przedstawimy teraz kilka innych typów selektorów, które dadzą nam większą elastyczność w kierowaniu elementów w dokumencie do stylizacji. Do tej pory używaliśmy nazw elementów jako selektorów. Zobaczyliśmy, jak grupować selektory na liście oddzielonej przecinkami, dzięki czemu możemy zastosować właściwości do kilku elementów jednocześnie. Oto przykłady selektorów, które już znamy:

```
Selektor elementów p { color: navy; }
Zgrupowane selektory p, ul, td, th { color: navy; }
```

Wadą wyboru elementów w ten sposób jest oczywiście to, że właściwość (w tym przypadku tekst w kolorze granatowym) jest stosowana do każdego akapitu i innych wymienionych elementów w dokumencie. Czasami chcemy zastosować regułę do określonego akapitu lub akapitów. W tej sekcji przyjrzymy się trzem typom selektorów, które pozwalają nam to zrobić: **selektorem potomka**, **selektorem identyfikatorów** i **selektorem klas**.

Selektor potomka wybiera elementy, które są zawarte w (i dlatego są potomkami) innego elementu. Jest to przykład selektora kontekstowego, ponieważ wybiera element na podstawie jego kontekstu lub relacji z innym elementem. Selektory potomków są wskazane na liście oddzielonej spacją. Ten przykład dotyczy wyróżnionych elementów tekstowych (`em`), ale tylko wtedy, gdy pojawiają się w elementach listy (`li`). Wyróżniony tekst w akapitach i inne elementy pozostaną nienaruszone.



Oto kolejny przykład, który pokazuje, jak selektory kontekstowe można pogrupować na liście oddzielonej przecinkami, tak jak widzieliśmy wcześniej. Ta reguła dotyczy elementów, ale tylko wtedy, gdy występują w nagłówkach `h1`, `h2` i `h3`.

```
<!--selector_examples.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Selector examples</title>
    <style>
        h1 em, h2 em, h3 em {
            color: red;
        }
    </style>
</head>
<body>
    <h1>Heading <em>one</em></h1>
    <h2>Heading <em>two</em></h2>
    <h3>Heading <em>three</em></h3>
</body>
</html>
```

Heading one

Heading two

Heading three

Możliwe jest również zagnieżdżanie selektorów potomka na kilka warstw. Ten przykład dotyczy elementów `em`, które pojawiają się w kotwicach (`a`) w uporządkowanych listach (`ol`).

```
<!--selector_examples.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Selector examples</title>
  <style>
    h1 em, h2 em, h3 em {
      color: red;
    }
    ol a em {
      font-variant: small-caps;
    }
  </style>
</head>
<body>
  <h1>Heading <em>one</em></h1>
  <h2>Heading <em>two</em></h2>
  <h3>Heading <em>three</em></h3>
  <ol>
    <li><a href="https://google.pl"><em>Google.pl</em></a></li>
    <li><a href="https://google.pl">Google.pl2</a></li>
  </ol>
</body>
</html>
```

Heading one

Heading two

Heading three

1. [GOOGLE.PL](#)
2. [Google.pl2](#)

Dowiedzieliśmy się o atrybucie `id`, który nadaje elementowi unikalną nazwę identyfikującą (jego referencję `id`). Atrybut `id` może być używany z dowolnym elementem i jest powszechnie używany do nadawania znaczenia ogólnym elementom `div` i `span`.

Selektory identyfikatorów umożliwiają kierowanie elementów według ich wartości identyfikatorów. Symbolem identyfikującym selektory identyfikatora jest hash (`#`), znana również jako symbol krzyżyka lub funta. Oto przykład elementu listy z identyfikatorem.

```
<li id="sleestak">Sleestak T-shirt</li>
```

Teraz możemy napisać regułę stylu tylko dla tego elementu listy, używając selektora identyfikatora, w ten sposób (zwróciśmy uwagę na `#` poprzedzający odwołanie do identyfikatora).

```
<!--id_selector_examples.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Id selector examples</title>
    <style>
        li#sleestak {
            color: olive;
        }
    </style>
</head>
<body>
    <ul>
        <li id="sleestak">Sleestak T-shirt</li>
    </ul>
</body>
</html>
```

- **Sleestak T-shirt**

Ponieważ wartości `id` muszą być unikalne w dokumencie, dopuszczalne jest pominięcie nazwy elementu. Poniższa reguła jest równoważna ostatniej.

```
<!--id_selector_examples.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Id selector examples</title>
    <style>
        #sleestak {
            color: olive;
        }
    </style>
</head>
<body>
    <ul>
        <li id="sleestak">Sleestak T-shirt</li>
    </ul>
</body>
</html>
```

- **Sleestak T-shirt**

Możemy także użyć selektora identyfikatora jako części selektora kontekstowego. W tym przykładzie styl jest stosowany tylko do elementów, które pojawiają się w elemencie zidentyfikowanym jako `resource`. W ten sposób możemy traktować linki w elemencie o nazwie `resource` inaczej niż wszystkie pozostałe linki na stronie bez żadnych dodatkowych znaczników.

```
<!--id_selector_examples.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Id selector examples</title>
    <style>
        #sleestak {
            color: olive;
        }
        #resources a {
            text-decoration: none;
        }
    </style>
</head>
<body>
    <ul>
        <li id="sleestak">Sleestak T-shirt</li>
        <li id="resources"><a href="https://google.pl">Google.pl</a></li>
        <li><a href="https://google.pl">Google.pl</a></li>
    </ul>
</body>
</html>
```

-
- Sleestak T-shirt
 - Google.pl
 - Google.pl

Ostatnim typem selektora jest **selektor klasy**. Identyfikator klasy, używany do klasyfikowania elementów do grupy pojęciowej. W przeciwieństwie do atrybutu `id`, wiele elementów może mieć wspólną nazwę klasy. Nie tylko to, ale element może należeć do więcej niż jednej klasy. Nazwy klas są oznaczone kropką (`.`) na początku selektora. Na przykład, aby zaznaczyć wszystkie akapity z `class="special"`, użyjmy tego selektora (kropka wskazuje, że następujące słowo jest selektorem klasy).

```
<!--class_selector_examples.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Class selector examples</title>
    <style>
        p.special {
            color: orange;
        }
    </style>
</head>
<body>
    <p>This is a default text.</p>
    <p class="special">This is a orange text.</p>
</body>
</html>
```

This is a default text.

This is a orange text.

Aby zastosować właściwość do wszystkich elementów tej samej klasy, można pominąć nazwę elementu w selektorze (pamiętajmy o pozostawieniu kropki; to znak wskazuje klasę).

```
<!--class_selector_examples.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Class selector examples</title>
    <style>
        .special {
            color: orange;
        }
    </style>
</head>
<body>
    <p>This is a default text.</p>
    <p class="special">This is a orange text.</p>
    <div class="special">
        In this case all text will be orange.
    </div>
</body>
</html>
```

This is a default text.

This is a orange text.

In this case all text will be orange.

Selektory potomków są jednym z czterech typów selektorów kontekstowych (nazywanych kombinatorami w specyfikacji selektorów Poziom 3 i Poziom 4). Pozostałe trzy to **selektory podzielne (child selectors)**, **selektory następnego rodzeństwa (next-sibling selectors)** i **selektory kolejnego rodzeństwa (subsequent-sibling-selectors)**.

Selektor podzielny jest podobny do selektora potomka, ale jest skierowany tylko do bezpośrednich dzieci danego elementu. Pomiędzy nimi może nie być żadnych innych poziomów hierarchicznych. Są one oznaczone symbolem większości (`>`). Poniższa reguła dotyczy tekstu wyróżionego, ale tylko wtedy, gdy jest on bezpośrednio zawarty w elemencie `p`. Nie będzie to miało wpływu na element `em` wewnętrz łącza (`a`) w akapicie.

```
<!--other_contextual_selectors.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Other Contextual Selectors</title>
    <style>
        p > em {font-weight: bold;}
    </style>
</head>
<body>
    <p>This is a <em>default text</em></p>
    <em>This is another default text</em>
</body>
</html>
```

This is a **default text**

This is another **default text**

Selektor następnego rodzeństwa wskazuje element, który występuje bezpośrednio po innym elemencie o tym samym rodzicu. Jest to oznaczone znakiem plus (`+`). Ta zasada daje specjalne traktowanie akapitom następującym po `h1`. Pozostałe akapy pozostają nienaruszone.

```
<!--other_contextual_selectors.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Other Contextual Selectors</title>
    <style>
        p > em {font-weight: bold;}
        h1 + p {font-style: italic;}
    </style>
</head>
<body>
    <p>This is a <em>default text</em></p>
    <em>This is another default text</em>
    <hr>
    <h1>This is a heading</h1>
    <p>This is a default text</p>
    <hr>
    <h2>Another heading</h2>
    <p>This is a default text</p>
</body>
</html>
```

This is a **default text**

This is another default text

This is a heading

This is a default text

Another heading

This is a default text

Selektor kolejnego rodzeństwa wybiera element, który współdzieli element nadzędny z określonym elementem i występuje po nim w kolejności źródłowej. Nie muszą podążać za sobą bezpośrednio. Ten typ selektora jest nowy w CSS3. Poniższa reguła wybiera dowolne `h2`, które mają wspólny element nadzędny (takie jak `section` lub `article`) z `h1` i pojawiają się po nim w dokumencie.

```
<!--other_contextual_selectors.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Other Contextual Selectors</title>
    <style>
        p > em {font-weight: bold;}
        h1 + p {font-style: italic;}
        section ~ h1 {color: red;}
    </style>
</head>
<body>
    <p>This is a <em>default text</em></p>
    <em>This is another default text</em>
    <hr>
    <h1>This is a heading</h1>
    <p>This is a default text</p>
    <hr>
    <h2>Another heading</h2>
    <p>This is a default text</p>
    <hr>
    <section>
        <h1>This is a heading in section</h1>
        <h2>This is another heading in section</h2>
    </section>
    <hr>
    <h1>This is a heading in not section</h1>
    <h2>This is another heading in not section</h2>
</body>
</html>
```

This is a **default text**

This is another default text

This is a heading

This is a default text

Another heading

This is a default text

This is a heading in section

This is another heading in section

This is a heading in not section

This is another heading in not section

Step 7

Specyficzność, odnosi się do faktu, że bardziej szczegółowe selektory mają większą wagę, jeśli chodzi o rozwiązywanie konfliktów reguł stylów. Teraz, gdy znamy jeszcze kilka selektorów, nadszedł dobry moment, aby ponownie przyjrzeć się tej bardzo ważnej koncepcji. Ta lista typów selektorów, od najbardziej do najmniej szczegółowej, powinna nam dobrze służyć w większości scenariuszy:

- Style wbudowane z atrybutem `style` są bardziej szczegółowe niż (i będą nadpisane...)
- Selektory ID, które są bardziej szczegółowe niż (i zastąpią...)
- Selektory klas, które są bardziej szczegółowe niż (i zastąpią...)
- Indywidualne selektory elementów

Aby obliczyć specyficzność, zacznijmy od narysowania trzech ramek:

```
[ ] [ ] [ ]
```

Teraz policzymy liczbę identyfikatorów w selektorze i umieścimy tę liczbę w pierwszym polu. Następnie policzymy liczbę klas i pseudoklas w selektorze i umieścimy tę liczbę w drugim polu. Po trzecie, policzymy nazwy elementów i umieścimy tę liczbę w trzecim polu. Pierwsze pole, które nie jest remisem, określa, który selektor wygrywa. Oto prosty przykład dwóch sprzecznych reguł dla elementu `h1`:

```
h1 { color: red; } [0] [0] [1]  
h1.special { color: lime; } [0] [1] [1]
```

Drugi ma selektor klasy, a pierwszy nie; dlatego druga jest bardziej szczegółowa i ma większą wagę.

Co powiemy na coś bardziej skomplikowanego.

```
article#main aside.sidebar:hover > h1:first-of-type [1] [3] [3]  
.x.x.x.x.x.x.x a:link [0] [8] [1]
```

Drugi selektor celuje w `Link` w elemencie z ciągiem nazw klas (reprezentowanych przez „x”). Ale pierwszy selektor ma identyfikator (`#main`) i dlatego jest bardziej szczegółowy. Być może będziemy musieli wykonać to pełne obliczenie specyficzności, ale w większości przypadków będziemy mieli wyczucie, który selektor jest bardziej szczegółowy, postępując zgodnie z wcześniejszymi wymienionymi ogólnymi wskazówkami. Możemy strategicznie używać szczegółowości, aby arkusze stylów były proste, a znaczniki minimalne. Na przykład możliwe jest ustawnie stylu dla elementu (w tym przykładzie `p`), a następnie nadpisanie go w razie potrzeby za pomocą bardziej szczegółowych selektorów.

```
p { line-height: 1.2em; } [0] [0] [1]  
blockquote p { line-height: 1em; } [0] [0] [2]  
p.intro { line-height: 2em; } [0] [1] [1]
```

W tych przykładach elementy `p`, które pojawiają się w cudzysłowie, mają mniejszą wysokość linii niż zwykłe akapity. Jednak wszystkie akapity z klasą „wstępu” będą miały wysokość wiersza `2 em`, nawet jeśli występuje w cudzysłowie, ponieważ selektory klas są bardziej szczegółowe.

Więcej informacji na temat specyficzności w CSS można znaleźć pod adresem <https://www.w3.org/TR/selectors-4/#specificity> (<https://www.w3.org/TR/selectors-4/#specificity>) oraz <https://www.smashingmagazine.com/2007/07/css-specificity-things-you-should-know/> (<https://www.smashingmagazine.com/2007/07/css-specificity-things-you-should-know/>).

Następna partia właściwości tekstu dotyczy traktowania całych wierszy tekstu, a nie kształtów znaków. Pozwalają autorom stron internetowych na formatowanie tekstu internetowego z wcięciami, dodatkowymi odstępami między wierszami (interlinią) i różnymi poziomami wyrównaniami, podobnymi do drukowania.

line-height

Wartości: number | length measurement | percentage | normal
Domyślnie: normal
Dotyczy: Wszystkich elementów
Dziedziczy: tak

Właściwość `line-height` określa minimalną odległość od linii bazowej do linii bazowej w tekście. Mówiąc, że właściwość `line-height` określa „minimalną” odległość, ponieważ jeśli umieścimy wysoki obraz lub duże znaki w linii, wysokość tej linii rozszerzy się, aby ją dostosować. Linia bazowa to wyimaginowana linia, na której siedzą dna postaci. Ustawianie wysokości linii w CSS jest podobne do dodawania interlinii w tradycyjnym ustawianiu tekstu; jednak zamiast dodawania spacji między wierszami, dodatkowa przestrzeń jest dzielona nad i pod tekstem. W rezultacie `line-height` określa wysokość linii, w której linia tekstu jest wyśrodkowana w pionie.

The line-height property defines the minimum distance from baseline to baseline in text.

Baseline A baseline is the imaginary line upon which the bottoms of characters sit. Line height in CSS is similar to leading in traditional typesetting.

line-height: 2em;

Te przykłady pokazują trzy różne sposoby, aby wysokość linii była dwukrotnie większa od wysokości rozmiaru czcionki:

```
<!--text_line_examples.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Text line examples</title>
</head>
<body>
    <p style="line-height:2;">This is a default text</p>
    <p style="line-height:2em;">This is a default text</p>
    <p style="line-height:200%;">This is a default text</p>
</body>
</html>
```

This is a default text

This is a default text

This is a default text

Gdy liczba jest określona sama, jak pokazano w pierwszym przykładzie, działa ona jako współczynnik skalowania, który jest mnożony przez bieżący rozmiar czcionki w celu obliczenia wartości wysokości wiersza. Wysokości linii można również określić w jednej z jednostek długości CSS. Wartości em i wartości procentowe są oparte na bieżącym rozmiarze czcionki elementu. W trzech przykładach, jeśli rozmiar czcionki wynosi 16 pikseli, obliczona wysokość linii wyniesie 32 piksele. Różnica między używaniem współczynnika skalowania (wartości liczbowej) a wartością względną (em lub %) polega na tym, jak dziedziczą. Jeśli ustawimy wysokość linii ze współczynnikiem skalowania dla całego dokumentu na elemencie treści, jego potomkowie dziedziczą mnożnik. Jeśli współczynnik skalowania jest ustawiony na 2 dla body , nagłówek o 24 pikselach będzie miał wysokość 48 pikseli. Jeśli ustawimy wysokość linii w elemencie body za pomocą znaków em lub wartości procentowych, jego potomkowie dziedziczą obliczony rozmiar na podstawie rozmiaru czcionki treści. Na przykład, jeśli wysokość linii jest ustawiona na 1em dla elementu body (obliczona na 16 pikseli), nagłówek 24-pikselowy dziedziczy obliczoną wysokość linii 16-pikselową, a nie wartość 1em .

Właściwość text-indent wcina pierwszy wiersz tekstu o określoną wartość.

text-indent

Wartości: length measurement | percentage

Domyślnie: 0

Dotyczy: bloków

Dziedziczy: tak

Możemy określić pomiar długości lub wartość procentową dla wcięcia tekstu.

```
<!--text_line_examples.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Text line examples</title>
</head>
<body>
    <p style="line-height:2;">This is a default text</p>
    <p style="line-height:2em;">This is a default text</p>
    <p style="line-height:200%;">This is a default text</p>
    <hr>
    <p style="text-indent: 2em">This is a default text</p>
    <p style="text-indent: 25%">This is a default text</p>
    <p style="text-indent: -35px">This is a default text</p>
</body>
</html>
```

This is a default text

This is a default text

This is a default text

This is a default text

This is a default text

is a default text

Wartości procentowe są obliczane na podstawie szerokości elementu nadzielnego i są przekazywane do elementów potomnych jako wartości procentowe (wartości nieobliczone). Więc jeśli `div` ma wcięcie tekstu równe `10%`, tak samo będzie z wszystkimi jego potomkami. W trzecim przykładzie zauważmy, że podano wartość ujemną i to jest w porządku. Spowoduje to, że pierwszy wiersz tekstu będzie zwisał na lewo od lewej krawędzi tekstu (nazywane również wcięciem wysuniętym).

Tekst na stronach internetowych można wyrównywać tak samo, jak w edytorze tekstu lub programie DTP za pomocą właściwości `text-align`.

text-align

Wartości: `left | right | center | justify | start | end`

Domyślnie: `start`

Dotyczy: bloków

Dziedziczy: tak

```

<!--text_line_examples.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Text line examples</title>
</head>
<body>
    <p style="line-height:2;">This is a default text</p>
    <p style="line-height:2em;">This is a default text</p>
    <p style="line-height:200%;">This is a default text</p>
    <hr>
    <p style="text-indent: 2em">This is a default text</p>
    <p style="text-indent: 25%">This is a default text</p>
    <p style="text-indent: -35px">This is a default text</p>
<hr>
    <p style="text-align: left">This is a default text</p>
    <p style="text-align: right">This is a default text</p>
    <p style="text-align: center">This is a default text</p>
    <p style="text-align: justify">This is a default text</p>
    <p style="text-align: start">This is a default text</p>
    <p style="text-align: end">This is a default text</p>
</body>
</html>

```

This is a default text

This is a default text

This is a default text

This is a default text

This is a default text

is a default text

This is a default text

Jeśli chcemy umieścić linię pod, nad lub w tekście, lub jeśli chcemy wyłączyć podkreślenie pod linkami, to należy skorzystać z właściwości `text-decoration`.

`text-decoration`

Wartości: `none` | `underline` | `overline` | `line-through` | `blink`

Domyślnie: `none`

Dotyczy: wszystkich elementów

Dziedziczy: nie, ale ponieważ linie są rysowane przez elementy podzielone, mogą wyglądać, jakby były „ozdobione”

```

<!--text_decoration_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Text decoration example</title>
</head>
<body>
    <p style="text-decoration: underline">This is a default text</p>
    <p style="text-decoration: overline">This is a default text</p>
    <p style="text-decoration: line-through">This is a default text</p>
    <p style="text-decoration: blink">This is a default text</p>
</body>
</html>

```

This is a default text

This is a default text

~~This is a default text~~

This is a default text

Najpopularniejszym zastosowaniem właściwości `text-decoration` jest wyłączenie podkreśleń, które pojawiają się automatycznie pod połączonym tekstem.

```
<!--text_decoration_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Text decoration example</title>
</head>
<body>
    <p style="text-decoration: underline">This is a default text</p>
    <p style="text-decoration: overline">This is a default text</p>
    <p style="text-decoration: line-through">This is a default text</p>
    <p style="text-decoration: blink">This is a default text</p>
    <a href="https://google.pl">Google.pl</a>
    <a style="text-decoration: none" href="https://google.pl">Google.pl</a>
</body>
</html>
```

This is a default text

This is a default text

~~This is a default text~~

This is a default text

[Google.pl](https://google.pl) [Google.pl](https://google.pl)

W związku z dekoracją tekstu należy powiedzieć kilka ostrzegawczych słów. Po pierwsze, jeśli usuniemy podkreślenia pod linkami, upewnijmy się, że istnieją inne wskazówki do skompensowania, takie jak kolor i waga. Z drugiej strony, ponieważ podkreślenia są tak silną wskazówką wizualną do „kliknięcia tutaj”, podkreślanie tekstu, który nie jest linkiem, może być mylące i frustrujące. Zastanów się, czy kursywa może być akceptowlą alternatywą. Wreszcie nie ma powodu, aby tekst mrugał. Twórcy przeglądarek zgadzają się i dlatego zrezygnowali z obsługi migającego tekstu.

Programy DTP wprowadziły funkcję, która pozwalała mi na bieżąco zmieniać wielkość liter w tekście. Dzięki temu łatwo było zobaczyć, jak moje nagłówki mogą wyglądać wielkimi literami, bez konieczności ich ponownego wpisywania. CSS zawiera tę funkcję również we właściwości `text-transform`.

`text-transform`

Wartości: `none` | `capitalize` | `lowercase` | `uppercase` | `full-width`

Domyślnie: `none`

Dotyczy: wszystkich elementów

Dziedziczy: tak

Po zastosowaniu właściwości `text-transform` do elementu tekstowego zmienia on wielkość liter podczas renderowania bez zmiany sposobu wpisywania w źródle.

```

<!--text_decoration_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Text decoration example</title>
</head>
<body>
    <p style="text-decoration: underline">This is a default text</p>
    <p style="text-decoration: overline">This is a default text</p>
    <p style="text-decoration: line-through">This is a default text</p>
    <p style="text-decoration: blink">This is a default text</p>
    <a href="https://google.pl">Google.pl</a>
    <a style="text-decoration: none" href="https://google.pl">Google.pl</a>
    <hr>
    <p style="text-transform: capitalize">This is a default text</p>
    <p style="text-transform: lowercase">This is a default text</p>
    <p style="text-transform: uppercase">This is a default text</p>
    <p style="text-transform: full-width">This is a default text</p>
</body>
</html>

```

This is a default text

—————
This is a default text

This is a default text

This is a default text

Google.pl Google.pl
—————

This Is A Default Text

Następne dwie właściwości tekstu służą do wstawiania spacji między literami (`letter-spacing`) lub słowami (`word-spacing`), gdy tekst jest wyświetlany.

letter-spacing

Wartości: length measurement | normal
 Domyślnie: normal
 Dotyczy: wszystkich elementów
 Dziedziczy: tak

word-spacing

Wartości: length measurement | normal
 Domyślnie: normal
 Dotyczy: wszystkich elementów
 Dziedziczy: tak

Warto zauważyć, że kiedy określamy wymiary `em`, obliczony rozmiar jest przekazywany do elementów potomnych, nawet jeśli mają mniejszy rozmiar czcionki niż nadzędny.

```

<!--text_decoration_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Text decoration example</title>
</head>
<body>
    <p style="text-decoration: underline">This is a default text</p>
    <p style="text-decoration: overline">This is a default text</p>
    <p style="text-decoration: line-through">This is a default text</p>
    <p style="text-decoration: blink">This is a default text</p>
    <a href="https://google.pl">Google.pl</a>
    <a style="text-decoration: none" href="https://google.pl">Google.pl</a>
    <hr>
    <p style="text-transform: capitalize">This is a default text</p>
    <p style="text-transform: lowercase">This is a default text</p>
    <p style="text-transform: uppercase">This is a default text</p>
    <p style="text-transform: full-width">This is a default text</p>
    <hr>
    <p style="letter-spacing: 8px">This is a default text</p>
    <p style="word-spacing: 8px">This is a default text</p>
</body>
</html>

```

This is a default text

This is a default text

~~This is a default text~~

This is a default text

[Google.pl](#) Google.pl

This Is A Default Text

T h i s i s a d e f a u l t t e x t

This is a default text

Właściwość `text-shadow` dodaje „cień” pod tekstem, który sprawia, że wydaje się unosić lub wyskakiwać nad stroną. Ponieważ modne stało się projektowanie w jednolitych kolorach, cienie wyszły z mody, ale nadal mogą być użytecznym narzędziem wizualnym, zwłaszcza gdy tekst znajduje się na tle wzorzystego lub fotograficznego tła. Cienie tekstu są rysowane za tekstem, ale przed tłem i krawędzią, jeśli istnieje.

text-shadow

Wartości: ‘horizontal offset’ ‘vertical offset’ ‘blur radius’ ‘color’ | none
 Domyślnie: none
 Dotyczy: wszystkich elementów
 Dziedziczy: tak

Wartość właściwości `text-shadow` to dwa lub trzy pomiary (przesunięcie w poziomie, przesunięcie w pionie i opcjonalny promień rozmycia) oraz kolor.

```

<!--text_decoration_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Text decoration example</title>
    <style>
        h1 {
            color: darkgreen;
            text-shadow: .2em .2em silver;
        }
    </style>
</head>
<body>
    <p style="text-decoration: underline">This is a default text</p>
    <p style="text-decoration: overline">This is a default text</p>
    <p style="text-decoration: line-through">This is a default text</p>
    <p style="text-decoration: blink">This is a default text</p>
    <a href="https://google.pl">Google.pl</a>
    <a style="text-decoration: none" href="https://google.pl">Google.pl</a>
    <hr>
    <p style="text-transform: capitalize">This is a default text</p>
    <p style="text-transform: lowercase">This is a default text</p>
    <p style="text-transform: uppercase">This is a default text</p>
    <p style="text-transform: full-width">This is a default text</p>
    <hr>
    <p style="letter-spacing: 8px">This is a default text</p>
    <p style="word-spacing: 8px">This is a default text</p>
    <hr>
    <h1>This is a shadow heading</h1>
</body>
</html>

```

This is a default text

This is a default text

~~This is a default text~~

This is a default text

[Google.pl](#) Google.pl

This Is A Default Text

T h i s i s a d e f a u l t t e x t

This is a default text

This is a shadow heading

Pierwsza wartość to przesunięcie w poziomie, które ustawia cień po prawej stronie tekstu (wartość ujemna przeciąga cień po lewej stronie tekstu). Drugi pomiar to pionowe przesunięcie, które przesuwa cień w dół o określoną wartość (wartość ujemna przesuwa cień w górę). Deklaracja kończy się specyfikacją koloru (srebrny). Jeśli kolor zostanie pominięty, zostanie użyty kolor tekstu. To powinno dać nam wyobrażenie o tym, jak działają pierwsze dwa pomiary, ale ten ostry cień nie wygląda zbyt... cóż... zaciemiony. Potrzebny jest pomiar promienia rozmycia. Zero (0) oznacza brak rozmycia, a przy wyższych wartościach rozmycie staje się bardziej miękkie.

```

<!--text_decoration_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Text decoration example</title>
    <style>
        h1 {
            color: darkgreen;
            text-shadow: .2em .2em .1em silver;
        }
    </style>
</head>
<body>
    <p style="text-decoration: underline">This is a default text</p>
    <p style="text-decoration: overline">This is a default text</p>
    <p style="text-decoration: line-through">This is a default text</p>
    <p style="text-decoration: blink">This is a default text</p>
    <a href="https://google.pl">Google.pl</a>
    <a style="text-decoration: none" href="https://google.pl">Google.pl</a>
    <hr>
    <p style="text-transform: capitalize">This is a default text</p>
    <p style="text-transform: lowercase">This is a default text</p>
    <p style="text-transform: uppercase">This is a default text</p>
    <p style="text-transform: full-width">This is a default text</p>
    <hr>
    <p style="letter-spacing: 8px">This is a default text</p>
    <p style="word-spacing: 8px">This is a default text</p>
    <hr>
    <h1>This is a shadow heading</h1>
</body>
</html>

```

This is a default text

This is a default text

~~This is a default text~~

This is a default text

[Google.pl](#) Google.pl

This Is A Default Text

T h i s i s a d e f a u l t t e x t

This is a default text

This is a shadow heading

Do tego samego elementu można zastosować kilka cieni tekstu. Jeśli zmienimy położenie i stopień rozmycia, możemy nadać tekstowi wygląd wielu źródeł światła. Rzucanie cieni może utrudniać czytanie tekstu, ale dodanie cienia do wszystkiego może również spowolnić działanie strony (przewijanie, interakcje myszy itp.), co jest szczególnie problematyczne w przypadku przeglądarki mobilnych bez dużej mocy obliczeniowej.

Step 8

Pokażemy teraz kilka poprawek, które możemy wprowadzić do list punktowanych i numerowanych. Jak wiadomo, przeglądarki automatycznie wstawiają punktory przed nieuporządkowanymi pozycjami list, a numery przed pozycjami na uporządkowanych listach (znaczniki list). W większości przypadków renderowanie tych znaczników jest określane przez przeglądarkę. Jednak CSS udostępnia kilka właściwości, które pozwalają autorom wybrać typ i położenie znacznika lub całkowicie je wyłączyć.

Właściwość `list-style-type` stosuje się do elementu `ul`, `ol` lub `li` wybierając typ znacznika, który pojawi się przed każdym elementem listy.

`list-style-type`

Wartości: `none` | `disc` | `circle` | `square` | `decimal` | `decimal-leading-zero` | `lower-alpha` | `upper-alpha` | `lower-latin` | `upper-latin` | `lower-roman` | `upper-roman` | `lower-greek`

Domyślnie: `disc`

Dotyczy: `ul`, `ol` oraz `li`

Dziedziczy: tak

Najczęściej programiści używają właściwości `list-style-type` z jej wartością ustawioną na `none`, aby całkowicie usunąć punktory lub liczby. Jest to przydatne, gdy używamy znaczników listy jako podstawy poziomego menu nawigacyjnego lub wpisów w formularzu internetowym. Możemy zachować semantykę, ale pozbądź się nieznośnych znaczników. Wartości `disc`, `circle` i `square` generują kształty pocisków, tak jak robiły to przeglądarki od początku samej sieci. Niestety nie ma możliwości zmiany wyglądu (rozmiaru, koloru itp.) generowanych pocisków, więc utknimy tutaj z domyślnym renderowaniem przeglądarki. Poniższa tabela przedstawia wartości dla poszczególnych wartości właściwości `list-style-type`.

Wartość	Numeracja
<code>decimal</code>	1,2,3,4,5,...
<code>decimal-leading-zero</code>	01,02,03,04,05,...
<code>lower-alpha</code>	a,b,c,d,e,...
<code>upper-alpha</code>	A,B,C,D,E,...
<code>lower-latin</code>	a,b,c,d,e,...
<code>upper-latin</code>	A,B,C,D,E,...
<code>lower-roman</code>	i,ii,iii,iv,v,...
<code>upper-roman</code>	I,II,III,IV,V,...
<code>lower-greek</code>	α, β, γ, δ, ε...

```
<!--list_properties_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>List properties example</title>
</head>
<body>
    <ul style="list-style-type: square">
        <li>One</li>
        <li>Two</li>
    </ul>
    <ol style="list-style-type: lower-greek">
        <li>One</li>
        <li>Two</li>
    </ol>
</body>
</html>
```

- One
 - Two
- α. One
β. Two

Domyślnie znacznik wisi poza obszarem zawartości elementu listy, wyświetlając jako wiszące wcięcie. Właściwość `list-style-position` umożliwia wciągnięcie punktora do obszaru zawartości, tak aby trafił do zawartości listy.

list-style-position

Wartości: inside | outside | hanging

Domyślnie: outside

Dotyczy: ul, ol oraz li

Dziedziczy: tak

```
<!--list_properties_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>List properties example</title>
    <style>
        li {
            background-color:#F99;
        }
    </style>
</head>
<body>
    <ul style="list-style-type: square">
        <li>One</li>
        <li>Two</li>
    </ul>
    <ol style="list-style-type: lower-greek">
        <li>One</li>
        <li>Two</li>
    </ol>
    <hr>
    <ul style="list-style-position: outside">
        <li>One</li>
        <li>Two</li>
    </ul>
    <ul style="list-style-position: inside">
        <li>One</li>
        <li>Two</li>
    </ul>
</body>
</html>
```

- One
- Two

- α. One
β. Two

-
- One
 - Two

- One
- Two

Możemy również użyć własnego obrazu jako punktora, używając właściwości `list-style-image`.

list-style-image

Wartości: url(location) | none

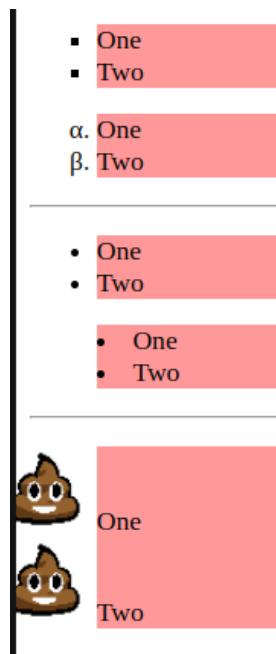
Domyślnie: none

Dotyczy: ul, ol oraz li

Dziedziczy: tak

Wartość właściwości `list-style-image` to adres URL obrazu, którego chcemy użyć jako znacznika. Typ listy jest ustawiony na `disc` jako kopię zapasową na wypadek, gdyby obraz się nie wyświetlał lub właściwość nie jest obsługiwana przez przeglądarkę lub innego klienta użytkownika.

```
<!--list_properties_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>List properties example</title>
    <style>
        li {
            background-color:#F99;
        }
    </style>
</head>
<body>
    <ul style="list-style-type: square">
        <li>One</li>
        <li>Two</li>
    </ul>
    <ol style="list-style-type: lower-greek">
        <li>One</li>
        <li>Two</li>
    </ol>
    <hr>
    <ul style="list-style-position: outside">
        <li>One</li>
        <li>Two</li>
    </ul>
    <ul style="list-style-position: inside">
        <li>One</li>
        <li>Two</li>
    </ul>
    <hr>
    <ul style="list-style-type: disc; list-style-image: url(poo.png); list-style-position: outside">
        <li>One</li>
        <li>Two</li>
    </ul>
</body>
</html>
```



Istnieje właściwość `list-style`, która łączy wartości typu, pozycji i obrazu w dowolnej kolejności. Na przykład:

```
ul {  
    list-style: url(poo.png) disc outside;  
}
```

Przykłady znajdują się pod

adresem: https://gitlab.com/mmiotk/technologieinternetu_nst_examples/-/tree/lecture_31 (https://gitlab.com/mmiotk/technologieinternetu_nst_examples/-/tree/lecture_31)