CSS

Cz.1

CSS - Cascading Style Sheets (z ang. Kaskadowe Arkusze Stylów) jest to specjalny język opracowany tylko w jednym celu: stworzenie możliwości bardziej elastycznego zarządzania sposobem formatowania (wyglądem) elementów znajdujących się w dokumentach elektronicznych. CSS nie może zatem istnieć samodzielnie, gdyż jest ściśle powiązane z językiem opisu struktury dokumentów takim jak (X)HTML. CSS daje możliwość globalnego zarządzania formą prezentacji całej witryny internetowej. Pomysł ten nie jest wcale nowy.

Wada

Część poleceń stylów nie jest interpretowana przez niektóre przeglądarki internetowe lub jest obsługiwana odmiennie. Dlatego zawsze należy sprawdzać efekty w praktyce - jeśli to możliwe, to najlepiej w kilku najbardziej popularnych przeglądarkach: Microsoft Internet Explorer, Netscape/Mozilla/Firefox…

Idea

Najważniejszym powodem wprowadzenia stylów było rozdzielenie struktury i prezentacji dokumentów.

Język HTML wywodzi się od SGML, a ten miał opisywać ogólną strukturę strony: nagłówek oraz ciało dokumentu, w którym mogły znajdować się akapity z tekstem, wykazy, tabele i inne elementy. Odpowiada on tylko za wstawienie tych elementów, ale nie określa ich wyglądu.

Jak łatwo się domyślić, szybko przestało to wystarczać - dlatego wprowadzono HTML. Zawarcie poleceń formatujących w samym HTML spowodowało jednak, że modyfikacja wyglądu elementów strony stała się bardzo żmudna (atrybuty i znaczniki które za to odpowiadają, są porozrzucane w różnych miejscach kodu, mieszając się ze strukturą dokumentu).

Dzięki wprowadzeniu stylów CSS, wszystkie polecenia dotyczące formatowania można umieścić w jednym miejscu (tzw. arkuszu) i powiązać je z konkretnymi elementami, wstawionymi za pomocą czystego (X)HTML. Taka koncepcja sprawia, że modyfikacja wyglądu stron może przebiegać dużo sprawniej.

Dlaczego warto używać CSS?

1. Style stały się już praktycznie podstawowym narzędziem formatującym.
2. Przestarzałe atrybuty i znaczniki, znajdujące się bezpośrednio w składni HTML, które dotyczą formatowania (zobacz: Elementy zdeprecjonowane), będą stopniowo wycofywane przez producentów przeglądarek internetowych, na rzecz rekomendowanych analogicznych deklaracji CSS.
3. Dokumenty pisane z wykorzystaniem arkuszy stylów są zwykle bardziej przejrzyste i krótsze.
4. Style pozwalają w łatwy sposób zarządzać całą serią dokumentów, poprzez stosowanie zewnętrznych arkuszy stylów. Dzięki temu w łatwy i wygodny sposób, można dokonać modyfikacji rodzaju formatowania jednocześnie we wszystkich dokumentach, zmieniając dane tylko w jednym pliku!
5. Dzięki możliwości stosowania klas selektorów, znacznie oszczędzamy sobie pisania. W jednym miejscu określamy wszystkie atrybuty formatowania (których może być bardzo dużo), odnoszące się do wielu elementów, które mają wyglądać tak samo. Bezpośrednio przy elemencie wystarczy podać tylko nazwę klasy i nie musimy już wypisywać "litanii" poleceń.
6. Możliwość stosowania różnorodnych jednostek oraz sposobów definiowania kolorów.
7. Style dają autorowi możliwości, które do tej pory były praktycznie niemożliwe do osiągnięcia:
   a. Różne wartości pogrubienia czcionki (9 rodzajów)
   b. Dodatkowe możliwości formatowania tekstu

Stosowanie CSS

Kaskadowe Arkusze Stylów CSS nie mogą funkcjonować samodzielnie, ponieważ definiują jedynie sposób formatowania (wygląd) elementów dokumentu, ale same ich nie tworzą. Elementy muszą zostać wstawione do dokumentu w postaci struktury znaczników np. za pomocą języka XHTML lub HTML. Dlatego aby poznać ogrom dodatkowych możliwości, jakie dają style CSS, konieczna jest wcześniejsza znajomość zasad języka (X)HTML.

Sposoby wstawiania stylów do gotowych dokumentów są różne. Nie znaczy to, że jedne są lepsze od drugich. Każdy sposób jest przydatny w innych sytuacjach.

Większość witryn stosuje jednocześnie wszystkie z przedstawionych metod osadzania CSS - w zależności od konkretnej potrzeby.

Styl lokalny

<selektor style="cecha: wartość; cecha2: wartość2...">...</selektor>

Selektorem może być praktycznie dowolny znacznik, np. p (akapit), h1 (tytuł), td (komórka tabeli) i inne. To właśnie elementom znajdującym się pomiędzy tymi znacznikami, nadajemy atrybuty formatowania.

Jako "cecha" czy inaczej własność bądź właściwość (ang. "property") należy wpisać o jakie konkretnie atrybuty formatowania nam chodzi (np. kolor tekstu - color). Opis wszystkich cech znajdziesz w następnych rozdziałach. Tutaj zajmiemy się tylko sposobami wstawiania stylów na stronach.

Jako wyraz "wartość" (w deklaracji stylu) wpisujemy dokładną wartość atrybutu (np. dla koloru tekstu będzie to: red, blue czy też #31F4A5 itd.). Wartość zależy ściśle od cechy, po której stoi w deklaracji. Nie można przecież jako kolor tekstu podać np. left (bez sensu). Opis wszystkich wartości, jakie mogą występować przy konkretnych cechach, znajdziesz w dalszych rozdziałach kursu.

Styl lokalny pozwala na nadanie formatowania konkretnemu pojedynczemu elementowi strony. Dlatego właśnie styl tego rodzaju nazywa się także styl inline (ang. "w linii"), ponieważ jest wstawiany w tej samej linii, w której znajduje się element formatowany. O tym który to będzie element, decyduje słowo kluczowe "selektor" (widoczny powyżej, w deklaracji stylu).

Zwróć uwagę, że jednemu selektorowi możemy nadać kilka atrybutów (cech). Są one wtedy rozdzielone średnikami.

```
<p style="color: red">To jest jakiś tekst</p>
```

Rozciąganie stylu

...

Jako "cecha" należy wpisać o jakie konkretnie atrybuty formatowania nam chodzi.

Natomiast jako wyraz "wartość" wpisujemy dokładną wartość atrybutu.

Znacznik ... pozwala na objęcie pewnego większego fragmentu dokumentu.

Pojedynczym znacznikiem ... możemy objąć kilka różnych elementów, które są wyświetlane w linii (sam element SPAN tak właśnie jest wyświetlany), np. wytłuszczenie tekstu oraz kursywę. Element ten tak dobrze nadaje się do osadzania stylów, ponieważ sam w sobie nie ma określonego żadnego formatowania, które mogłoby kolidować z efektem, jaki chcemy uzyskać.

Zwróć uwagę, że jednemu selektorowi możemy nadać kilka atrybutów (cech). Są one wtedy rozdzielone średnikami.

Przykład:

```xhtml
<span style="color: red">
        <b>To jest pogrubienie (element wyświetlany w linii) w ramach SPAN</b>,
        a to jest zwykły tekst - również wewnątrz span.
</span>
```

Otrzymamy:

To jest pogrubienie (element wyświetlany w linii) w ramach SPAN, a to jest zwykły tekst - również wewnątrz SPAN.

Wydzielone bloki

** ... **

Jako "cecha" należy wpisać o jakie konkretnie atrybuty formatowania nam chodzi.

Natomiast jako wyraz "wartość" wpisujemy dokładną wartość atrybutu.

Fragment dokumentu wydzielony za pomocą bloku ... możemy swobodnie formatować.

Element ten tak dobrze nadaje się do osadzania stylów, ponieważ sam w sobie nie ma określonego żadnego formatowania, które mogłoby kolidować z efektem, jaki chcemy uzyskać.

Metoda ta jest bardzo podobna do SPAN, lecz obejmuje zwykle większe fragmenty dokumentu (może zawierać w sobie różne znaczniki jak również inne bloki).

Dodatkowo **DIV** domyślnie jest wyświetlany:

XXXXXXXXXXXXXXXXXXXXXXXX

w bloku

XXXXXXXXXXXXXXXXXXXXXXXX

Natomiast **SPAN**

XXXXXXXXXXXXXXXXXXXXXXXXXXXXw liniiXXXXXXXXXXXXXXXXXXXXXXXXXX

DIV vs SPAN

Generalnie element blokowy (DIV) może zawierać wewnątrz siebie zwykły tekst, jak również inne elementy blokowe. Został on pomyślany do tworzenia obszerniejszych struktur.

Natomiast elementy wyświetlanie w linii (SPAN) nie mogą zawierać elementów blokowych, ale mogą inne elementy wyświetlane w linii oraz zwykły tekst.
Wewnątrz DIV można oczywiście wpisać również SPAN

Przykład

```xhtml
<div style="background-color: yellow">
        <span style="color: red">
                <b>To jest pogrubienie w ramach DIV oraz SPAN i dlatego jest czerwone na żółtym tle</b>,
                a to jest zwykły tekst - również wewnątrz DIV i SPAN.
        </span>
        To jest zwykły tekst, ale tylko wewnątrz DIV - dlatego nie jest czerwony, ale ma żółte tło.
        <p>A to jest akapit (element blokowy) w ramach DIV.</p>
</div>
```

Otrzymamy:

To jest pogrubienie w ramach DIV oraz SPAN i dlatego jest czerwone na żółtym tle, a to jest zwykły tekst - również wewnątrz DIV i SPAN. To jest zwykły tekst, ale tylko wewnątrz DIV - dlatego nie jest czerwony, ale ma żółte tło.

A to jest akapit (element blokowy) w ramach DIV.

Wewnętrzny arkusz stylów

```
<head>
    (...)
<style type="text/css">
        selektor { cecha: wartość; cecha2: wartość2... }
        selektor2 { cecha: wartość; cecha2: wartość2... }
        (...)
</style>
    (...)
</head>
```

W miejsce kropek (...) można wpisać dalsze polecenia.

Selektorem może być praktycznie dowolny znacznik, np. p (akapit), h1 (tytuł), td (komórka tabeli) i inne. To właśnie elementom, które znajdują się w kodzie źródłowym, pomiędzy tymi znacznikami, nadajemy atrybuty formatowania opisane w arkuszu.

Jako "cecha" (w deklaracji stylu - powyżej) należy wpisać o jakie konkretnie atrybuty formatowania nam chodzi (opisane w kolejnych rozdziałach).

Natomiast jako wyraz "wartość" wpisujemy dokładną wartość atrybutu.

Zwróć uwagę, że jednemu selektorowi możemy nadać kilka atrybutów (cech). Są one wtedy rozdzielone średnikami.

Wewnętrzny arkusz stylów wstawia się zawsze w części nagłówkowej dokumentu (pomiędzy znacznikami <head> a </head>).

Można go zastosować, gdy elementy które pragniemy poddać formatowaniu, występują wielokrotnie na stronie i wszystkim chcemy nadać takie same atrybuty (inne niż domyślne).

Na przykład chcemy, aby wszystkie wykazy miały automatycznie kolor niebieski. Wystarczy wpisać odpowiednią deklarację stylów w arkuszu (w treści nagłówkowej) i nie trzeba już nic dopisywać przy samym elemencie.

UWAGA: Znacznik STYLE może znajdować się tylko i wyłącznie w nagłówku dokumentu.

Przykład:

Jeśli w treści nagłówkowej strony zostałby umieszczony następujący wewnętrzny arkusz stylów:

```xhtml
<style type="text/css">
/* <![CDATA[ */
h6 { color: red }
/* ]]> */
</style>
```

to po wpisaniu w dowolnym miejscu strony po prostu:

```xhtml
<h6>To jest tytuł rzędu 6</h6>
```

otrzymalibyśmy tytuł rzędu szóstego koloru czerwonego (color: red) i to niezależnie od tego, ile będzie na stronie takich tytułów. Wygodne i szybkie rozwiązanie!

Zewnętrzny arkusz stylów

```
<head>
    (...)
    <link rel="Stylesheet" type="text/css" href="style.css" />
    (...)
</head>
```

gdzie "style.css" jest zewnętrznym arkuszem stylów. Natomiast znaki (...) oznaczają inne polecenia, które zwykle pojawiają się w nagłówku dokumentu, np. deklaracja strony kodowej.

Możliwość wstawiania **zewnętrznego arkusza** jest chyba jedną z największych zalet stosowania stylów.

Pozwala to zdefiniować takie samo formatowanie określonych elementów na wielu stronach jednocześnie.

Dzięki temu, za pomocą tego jednego arkusza, wszystkie nasze strony w obrębie całego serwisu mogą mieć pewne wspólne cechy. Dodatkowo jeśli w ostatniej chwili zdecydujemy się zmienić np. rodzaj czcionki na wszystkich stronach, możemy to zrobić, modyfikując jedynie zewnętrzny arkusz stylów, bez konieczności zmiany każdej strony osobno. Pozwala to zaoszczędzić mnóstwo czasu…!

UWAGA

W pojedynczym dokumencie (X)HTML można dołączyć dowolną liczbę zewnętrznych arkuszy stylów - każdy jako osobny element <link rel="Stylesheet" />.

W przypadku konfliktów, ważniejsze będą deklaracje z arkusza dołączonego później.

Zwykle tworzy się pojedynczy zewnętrzny arkusz i załącza go w całym serwisie, czyli na wszystkich podstronach. Czasami jednak dodatkowo poza nim projektuje się osobne arkusze, ustalające wygląd np. odrębnych kategorii tematycznych serwisu.

Edycja zewnętrznego CSS

Zewnętrzny arkusz stylów jest po prostu zwykłym plikiem tekstowym.

Aby go utworzyć, wystarczy zwykły edytor tekstu, w którym piszemy takie same deklaracje stylów selektor { cecha: wartość }, jak w przypadku wewnętrznego arkusza stylów.

Oczywiście można się posłużyć specjalnym edytorem CSS. Należy jedynie pamiętać, że plik będący zewnętrznym arkuszem stylów musi mieć rozszerzenie *.css!

UWAGA

Wstawienie białych znaków (spacje, tabulacje, znaki nowej linii) w arkuszu CSS nie ma wpływu na jego działanie. Dlatego możesz ułożyć wpisywane reguły CSS w taki sposób, aby były bardziej czytelne.

UWAGA2

Znacznik LINK może znajdować się tylko i wyłącznie w nagłówku dokumentu.

Przykład

```css
/* SELEKTORY: */
body {
        font-family: Verdana, Arial, Helvetica, sans-serif;
        font-size: 10pt;
        color: #003868;
        background-color: #80B8E8;
        margin: 6mm;
}

p {
        text-align: justify;
}

pre, code {
        font-size: 8pt;
}
```

Uwaga: zwróć uwagę na pierwszą linię, w której umieszczono komentarz!

Kaskadowość stylów

Kaskadowość stylów określa pierwszeństwo w oddziaływaniu na te same elementy strony stylów z różnych źródeł.

Na przykład: na naszej stronie używamy zewnętrznego arkusza stylów, deklaracji stylów w nagłówku strony, a także stylów typu inline, przy czym dotyczą one formatowania dokładnie tego samego elementu (np. kroju czcionki).

Co się stanie w takim przypadku? Czy nie powstaną konflikty? A może komputer się "zawiesi"? Nic z tych rzeczy! A to właśnie ze względu na kaskadowość. To od niej wzięły swoją nazwę style: CSS - (ang. Cascading Style Sheets) Kaskadowe Arkusze Stylów.

Priorytet

Jeśli w dokumencie znajduje się kilka źródeł stylów, pierwszeństwo mają te, które znajdują się "bliżej" formatowanego elementu. Oddziaływanie stylów z arkuszy zewnętrznych może być modyfikowane przez style zdefiniowane w nagłówku dokumentu, to z kolei może być zmieniane przez style zdefiniowane bezpośrednio w ciele dokumentu (inline). Zatem priorytet ważności stylów (pierwszeństwo) wyglądałby tak:

1. Styl lokalny (inline)
2. Rozciąganie stylu (SPAN)
3. Wydzielone bloki (DIV)
4. Wewnętrzny arkusz stylów
5. Zewnętrzny arkusz stylów

Style o wyższym priorytecie ważności (na początku listy) mają pierwszeństwo w modyfikowaniu elementów dokumentu.

UWAGA!

UWAGA! Polecenie dołączenia zewnętrznego arkusza powinno znajdować się w dokumencie wcześniej niż wewnętrzny arkusz. Odwrotna kolejność złamie zasady kaskadowości!

Reguły stylów

Atrybuty formatowania w języku CSS definiuje się za pomocą tzw. reguł stylów.

Każda reguła odnosi się do konkretnego elementu (znacznika) i składa się z dwóch części: selektora i deklaracji.

Selektor określa do jakich elementów ma zostać przypisane formatowanie, a deklaracja podaje to formatowanie i jest umieszczona w nawiasie klamrowym {...}.

Każda deklaracja składa się przynajmniej z jednego zespołu cecha lub inaczej własność albo właściwość (ang. property) - wartość (ang. value), przy czym można podać dowolną liczbę, rozdzielając kolejne znakiem średnika (;). Średnik na końcu deklaracji nie jest konieczny.

Reguły stylów

Każda grupa elementów (znaczników) ma określony zespół cech CSS, które można jej przypisać, a każda cecha ma ściśle wyszczególnioną listę wartości, które może przyjąć.

Na przykład: cecha text-align (wyrównanie tekstu) może być przypisana tylko i wyłącznie do elementów blokowych, ponieważ podanie jej dla elementów wyświetlanych w linii nie miałoby sensu.

Z drugiej strony cecha ta może przyjmować tylko wartości takie jak: left, right, center, justify. Przypisanie do niej np. wartości koloru nie miałoby sensu.

Przykład dwóch reguł stylów:

```css
/* Pierwsza reguła: */
p { color: red }
/* Druga reguła: */
p b { color: red; background-color: yellow }
```

W pierwszej selektorem jest znacznik p, a deklaracja ma postać { color: red }. Cechą jest color, a wartością red.

W drugiej selektorem jest zestawienie znaczników p b. Deklaracja zawiera dwa zespoły cecha-wartość.

Dziedziczenie stylów

Z drzewem dokumentu związana jest własność dziedziczności stylów.

Polega ona na tym, że elementy leżące niżej w hierarchii (potomkowie), jeśli nie zaznaczymy inaczej, dziedziczą (przejmują) atrybuty formatowania, które zostały nadane ich przodkom. Niestety w niektórych przeglądarkach internetowych zdarza się błędna interpretacja dziedziczenia stylów. Dlatego zawsze sprawdzaj w praktyce zastosowanie tej własności.

PRZYKŁAD:

To jest akapit koloru zielonego, wewnątrz którego znajduje się: *pochylenie* oraz podkreślenie, którym nie zostały nadane żadne style, a więc dziedziczą je po przodku, czyli po akapicie (są również zielone).
A to jest **pogrubienie**, które znajduje się także wewnątrz tego samego akapitu, ale został mu nadany atrybut koloru czcionki (biały) oraz koloru tła (niebieski) i dlatego nie odziedziczył stylu po przodku.

Selektor typu

selektor { cecha: wartość }

gdzie wyrazem selektor może być praktycznie dowolny znacznik, np. p (akapit), h1 (tytuł), td (komórka tabeli) i inne...

To właśnie elementom znajdującym się pomiędzy tymi znacznikami, nadajemy atrybuty formatowania.

Wyrazy "cecha" (ang. property) oraz "wartość" (ang. value) określają atrybuty elementu i zostaną opisane w dalszych rozdziałach. Tutaj będą przedstawione jedynie selektory.

Selektor typu jest podstawowym rodzajem selektora. Pozwala on wybrać pojedynczy zwykły element dokumentu (X)HTML podanego typu, czyli o określonej nazwie, a następnie nadać mu atrybuty.

Przykład:

```css
h6 { color: red }
```

Jak widać jest to selektor typu (podstawowy). Dzięki niemu wystarczyłoby teraz napisać:

```xhtml
<h6>To jest tytuł rzędu szóstego</h6>
```

aby otrzymać tytuł, napisany czerwoną czcionką (red).

Selektor uniwersalny

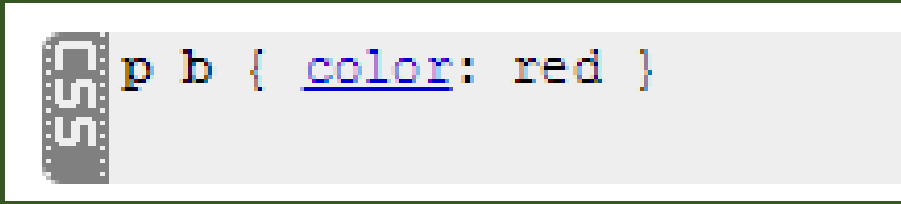
*** { cecha: wartość }**

Selektor taki pozwala ustalić określone atrybuty dla wszystkich elementów strony, a więc dla różnych selektorów typu. Możemy za pomocą tej komendy nadać to samo formatowanie dla wszystkich elementów na całej stronie, niezależnie od ich typu (p, h1, li itd.).

Selektor potomka

przodek1 przodek2 ... potomek { cecha: wartość }

gdzie wyrazy "przodek1, przodek2,..." oraz "potomek" (wielokropek należy pominąć!) są selektorami typu, przy czym przodek leży wyżej w hierarchii drzewa dokumentu.



Dzięki temu, jeśli wewnątrz znacznika p (akapit) znajdzie się znacznik b (czyli pogrubienie tekstu), to zostanie on automatycznie napisany czcionką koloru czerwonego.

To jest akapit, a to jest ***pogrubienie (czerwone)***, *umieszczone wewnątrz tego akapitu oraz wewnątrz znacznika tekstu pochylonego*. A to jest **pogrubienie (czerwone)** bezpośrednio w akapicie, które nie zawiera się w znaczniku pochylenia.

Selektor dziecka

rodzic > dziecko { cecha: wartość }

Selektor tego typu pozwala nadać atrybuty elementom, które leżą o jeden rząd niżej w hierarchii drzewa dokumentu (zawierają się w innym zewnętrznym znaczniku).

W odróżnieniu do poprzedniego przypadku, tutaj znacznik będący dzieckiem, musi znajdować się bezpośrednio wewnątrz znacznika rodzica.

```css
p > b { color: red }
```

Dzięki temu, jeśli bezpośrednio wewnątrz znacznika p (akapit) znajdzie się znacznik b (czyli pogrubienie tekstu), to zostanie on automatycznie napisany czcionką koloru czerwonego.

To jest akapit, a to jest ***pogrubienie (nie powinno być czerwone)*** *umieszczone wewnątrz p, ale i wewnątrz znacznika pochylenia*. Natomiast to jest **pogrubienie (powinno być czerwone)** umieszczone bezpośrednio wewnątrz znacznika akapitu.

Grupowanie selektorów

selektor1, selektor2, selektor3... { cecha: wartość }

Taka deklaracja stylu pozwala nadać te same wartości atrybutów kilku różnym selektorom jednocześnie (bez względu na ich położenie w hierarchii drzewa dokumentu).

Oznacza to, że zamiast wpisywać kilka razy te same deklaracje, wystarczy wymienić dowolną liczbę selektorów po przecinku i jednorazowo przypisać im wszystkim ten sam styl.

```css
b, i { color: red }
```

po wpisaniu:

```xhtml
<b>pogrubienie</b>
<i>pochylenie</i>
```

Otrzymamy: **pogrubienie** *pochylenie*

Prosty selektor atrybutu

selektor[atrybut] { cecha: wartość }

Selektorem może być dowolny znacznik, np. p - akapit, h1 - tytuł czy td - komórka tabeli i inne...

Atrybut oznacza konkretny parametr nadany znacznikowi z poziomu języka (X)HTML (np. atrybut title="...").

Przykład:

```css
p[title] { color: red }
```

Efekt:

To jest akapit, któremu został nadany atrybut TITLE (aby to sprawdzić, wskaż go myszką) i dlatego powinien być koloru czerwonego.

Klasy selektorów

Selektorem może być dowolny znacznik, np. p - akapit, h1 - tytuł czy td - komórka tabeli i inne…

Klasa to wartość atrybutu class="..." nadanego selektorowi z poziomu języka (X)HTML.

UWAGA: Jako klasa należy podać dowolny pojedynczy wyraz, który nie może zawierać znaków: spacji, kropki, przecinka, dwukropka, pytajnika, nawiasów, znaku równości, plusa itp. Może natomiast zawierać litery (A-Z, a-z), cyfry (0-9), myślniki ("-") i podkreślniki ("_"). Lepiej nie używać polskich liter. Nie może się on również rozpoczynać cyfrą ani myślnikiem.

Przykład:

```css
p.przyklad_klasa { color: red }
```

Użycie w HTML

```xhtml
<p class="przyklad_klasa">To jest akapit.</p>
```

Efekt:

To jest akapit.

Selektor identyfikatora

selektor#identyfikator { cecha: wartość }

Selektorem może być dowolny znacznik, np. p - akapit, h1 - tytuł czy td - komórka tabeli i inne...

Identyfikator to wartość atrybutu id="..." nadanego selektorowi z poziomu języka (X)HTML.

UWAGA: nazwa identyfikatora tak jak w poprzednim wypadku obarczona jest restrykcjami.

przykład:

```css
p#przyklad_identyfikator { color: red }
```

Użycie:

```xhtml
<p id="przyklad_identyfikator">To jest akapit.</p>
```

Efekt: To jest akapit.

Użycie **selektora uniwersalnego** pozwala przypisać styl do dowolnego znacznika z określonym **identyfikatorem**:

```css
*#przyklad_uniwersalny { color: red }
```

W tym przypadku gwiazdkę (*) w regule stylu można pominąć:

```css
#przyklad_uniwersalny { color: red }
```

Selektory pseudoklas

Pseudoklasy klasyfikują elementy inaczej niż po ich nazwie, atrybutach czy zawartości, tzn. w zasadzie nie są ustalane na podstawie drzewa dokumentu. Mogą być dynamiczne w tym sensie, że element "nabywa" lub "traci" pseudoklasę podczas interakcji z użytkownikiem. Przykładem jest podświetlenie elementu po wskazaniu go myszką przez użytkownika.

Wszystkie pseudoklasy można podzielić w następujący sposób:

Pseudoklasy dynamiczne

- Pseudoklasy linków: :link, :visited
- Pseudoklasy akcji użytkownika: :active, :hover, :focus

Pseudoklasa etykiety: :target

Pseudoklasa języka: :lang()

Pseudoklasy interfejsu użytkownika:

- :enabled, :disabled
- :checked

Pseudoklasy strukturalne:

- :root
- :nth-child(), :nth-last-child(), :nth-of-type(), :nth-last-of-type()
- :first-child
- :last-child
- :only-child
- :first-of-type, :last-of-type, :only-of-type
- :empty

Pseudoklasa negacji: :not()

Odsyłacz podstawowy

a:link { cecha: wartość }

gdzie litera "a" na początku deklaracji, jest selektorem odsyłacza.

Polecenie pozwala nadać określone atrybuty formatowania dla wszystkich podstawowych odsyłaczy na stronie czyli takich, które nie zostały jeszcze odwiedzone przez użytkownika.

Odsyłacz odwiedzony

a:visited { cecha: wartość }

gdzie litera "a" na początku deklaracji, jest selektorem odsyłacza.

Polecenie pozwala nadać określone atrybuty formatowania dla wszystkich odsyłaczy, które zostały już odwiedzone przez użytkownika.

Aktywacja

selektor:active { cecha: wartość }

Selektorem wg CSS 2 teoretycznie może być dowolny znacznik.

Polecenie pozwala nadać określone atrybuty formatowania dla elementów, które zostały aktywowane przez użytkownika. Może to mieć miejsce np. kiedy użytkownik wciśnie i przytrzyma przycisk myszki na odsyłaczu - aktywacja będzie miała miejsce do momentu zwolnienia przycisku myszki.

Wskazanie myszką

selektor:hover { cecha: wartość }

Selektorem wg CSS 2 teoretycznie może być dowolny znacznik.

Polecenie pozwala nadać określone atrybuty formatowania dla elementów, nad którymi znajduje się wskaźnik myszki, kiedy użytkownik jeszcze ich nie kliknął, tzn. nie zostały jeszcze aktywowane.

Przykład:

```
a:link { cecha: wartość }
a:visited { cecha: wartość }
a:hover { cecha: wartość }
a:active { cecha: wartość }
```

Własności CSS

(wybrane)

Własności Czcionki

Wielkość

selektor { font-size: rozmiar }

Selektorem może być dowolny znacznik, w którym można wpisywać tekst, np. p - akapit, h1 - tytuł czy td…

Wielkość

selektor { font-size: rozmiar }

jako "rozmiar" należy podać konkretną wielkość czcionki. Możliwe są tutaj cztery sposoby:

Imienne wartości absolutne:

xx-small - najmniejsza

x-small - mniejsza

small - mała

medium - średnia

large - duża

x-large - większa

xx-large – największa

Wartości względne:

smaller - mniejsza od bieżącej

larger - większa od bieżącej

Jednostki długości

Jednostka "em" jest w tym przypadku miarą względną i odnosi się do rozmiaru czcionki zdefiniowanej dla elementu rodzica, czyli: 1em = 100%.

Procent wielkości bieżącej

Przykłady – wartości absolutne:

PRZYKŁAD:

rozmiar **xx-small**

rozmiar **x-small**

rozmiar **small**

rozmiar **medium**

rozmiar **large**

rozmiar **x-large**

rozmiar **xx-large**

Przykłady – jednostki długości:

PRZYKŁAD:

rozmiar **0.5cm**

rozmiar **4mm**

rozmiar **0.25in**

rozmiar **10pt**

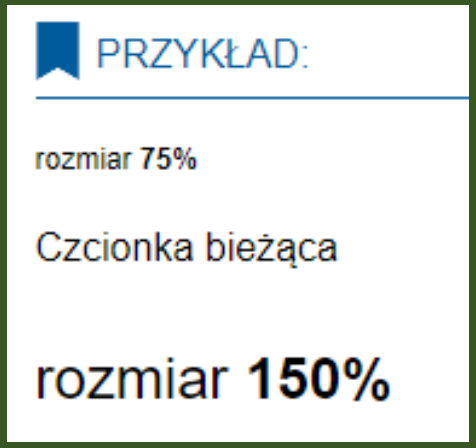
rozmiar **1pc**

rozmiar **15px**

rozmiar **1em**

rozmiar **2ex**

Przykłady – procent:



Rodzaj

selektor { font-family: rodzaj, rodzaj1, rodzaj2,... }

Selektorem może być dowolny znacznik, w którym można wpisywać tekst.

Natomiast jako "rodzaj, rodzaj1, rodzaj2,..." należy podać rodzaje czcionek. Podanie kilku rodzajów spowoduje, że jeśli użytkownik nie będzie posiadał pierwszego, to zostanie wybrany następny w kolejności.

```css
body { font-family: Arial, Helvetica, Verdana, sans-serif }
```

Styl

selektor { font-style: styl }

Selektorem może być dowolny znacznik, w którym można wpisywać tekst.

Natomiast jako "styl" należy wpisać:

normal - czcionka normalna (podstawowa)

italic - czcionka pochylona (jeżeli niedostępna, automatycznie wybierany jest styl oblique)

oblique - również czcionka pochylona (podobna jak poprzednio)

Waga

selektor { font-weight: waga }

Selektorem może być dowolny znacznik, w którym można wpisywać tekst.

Natomiast jako "waga" należy wpisać:

Wartości absolutne:

- **normal - czcionka normalna (podstawowa)**
- **bold - czcionka pogrubiona**
- **100, 200, 300, 400 (odpowiednik "normal"), 500, 600, 700 (odpowiednik "bold")**

Wartości względne:

- **lighter**
- **bolder**

Wariant

selektor { font-variant: wariant }

Selektorem może być dowolny znacznik, w którym można wpisywać tekst.

Natomiast jako "wariant" należy wpisać:

normal - czcionka normalna (podstawowa)

small-caps – kapitaliki

Atrybuty mieszane

selektor { font: wartości atrybutów }

Polecenie to pozwala w wygodny sposób zdefiniować wszystkie atrybuty dotyczące czcionek. Już nie musimy wypisywać kolejno wszystkich cech, a jedynie ich konkretne wartości.

Można tak:

```css
p { font: 12pt Arial; font-weight: bold }
```

Ale lepiej tak:

```css
p { font: bold 12pt Arial }
```

Własności Tekstu

Kolor

selektor { color: kolor }

Selektorem może być dowolny znacznik, w którym można wpisywać tekst.

Natomiast jako "kolor" należy podać definicję koloru.

Polecenie pozwala ustalić dowolny kolor tekstu.

Dekoracja

selektor { text-decoration: dekoracja }

Jako "dekoracja" należy podać:

none - bez zmian

underline - podkreślenie

line-through - przekreślenie

overline - nadkreślenie

blink - migotanie tekstu (tylko Netscape/Mozilla/Firefox i Opera 7)

Polecenie umożliwia na wybór określonej dekoracji tekstu (np. podkreślenie).
Migotanie może nie być interpretowane przez wszystkie przeglądarki!

Wyrównanie

selektor { text-align: wyrównanie }

Jako "wyrównanie" należy podać:

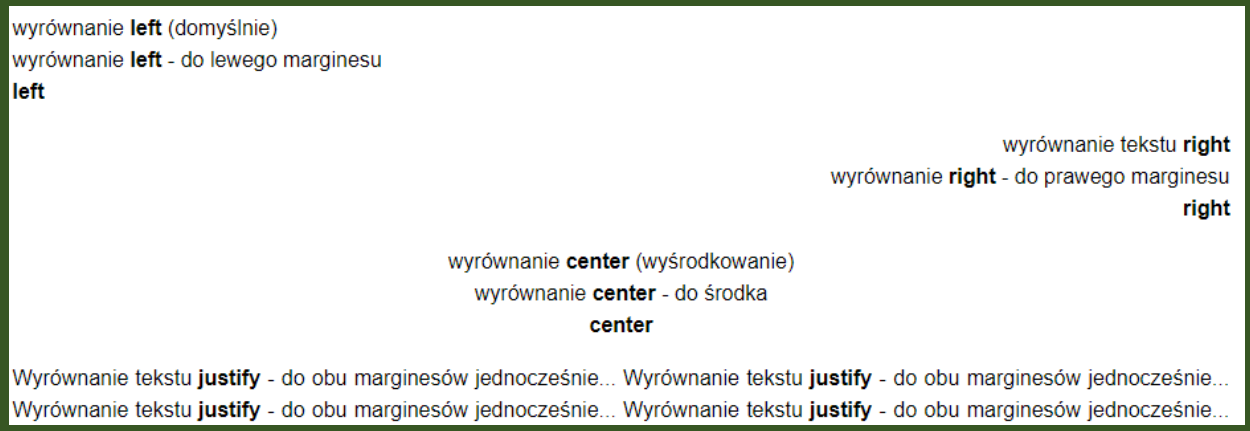
left - wyrównanie tekstu do lewego marginesu (domyślnie)

right - wyrównanie do prawego marginesu

center - do środka (wyśrodkowanie)

justify - do obu marginesów jednocześnie (justowanie)

Polecenie pozwala wybrać jeden z możliwych sposobów wyrównania tekstu, czyli jego ułożenia na ekranie.



Własności Tła

Kolor

selektor { background-color: kolor }

Selektorem może być praktycznie dowolny znacznik.

Natomiast jako "kolor" należy podać definicję koloru. Wpisanie "transparent" ustali tło przezroczyste.

Tło obrazkowe

selektor { background-image: url(ścieżka dostępu do obrazka) }

Selektorem może być praktycznie dowolny znacznik.

Natomiast jako "ścieżka dostępu do obrazka" należy podać miejsce, gdzie znajduje się obrazek, który chcemy wstawić jako tło. Wpisanie "none" usunie obrazek.

UWAGA: Ścieżkę dostępu należy konstruować względem lokalizacji arkusza CSS, a nie względem adresu dokumentu HTML!

Powtarzanie

selektor { background-repeat: powtarzanie }

Selektorem może być praktycznie dowolny znacznik.

Natomiast jako "powtarzanie" należy wpisać:

repeat - powtarzanie tła w obu kierunkach (domyślnie)

repeat-x - powtarzanie tła tylko w kierunku poziomym

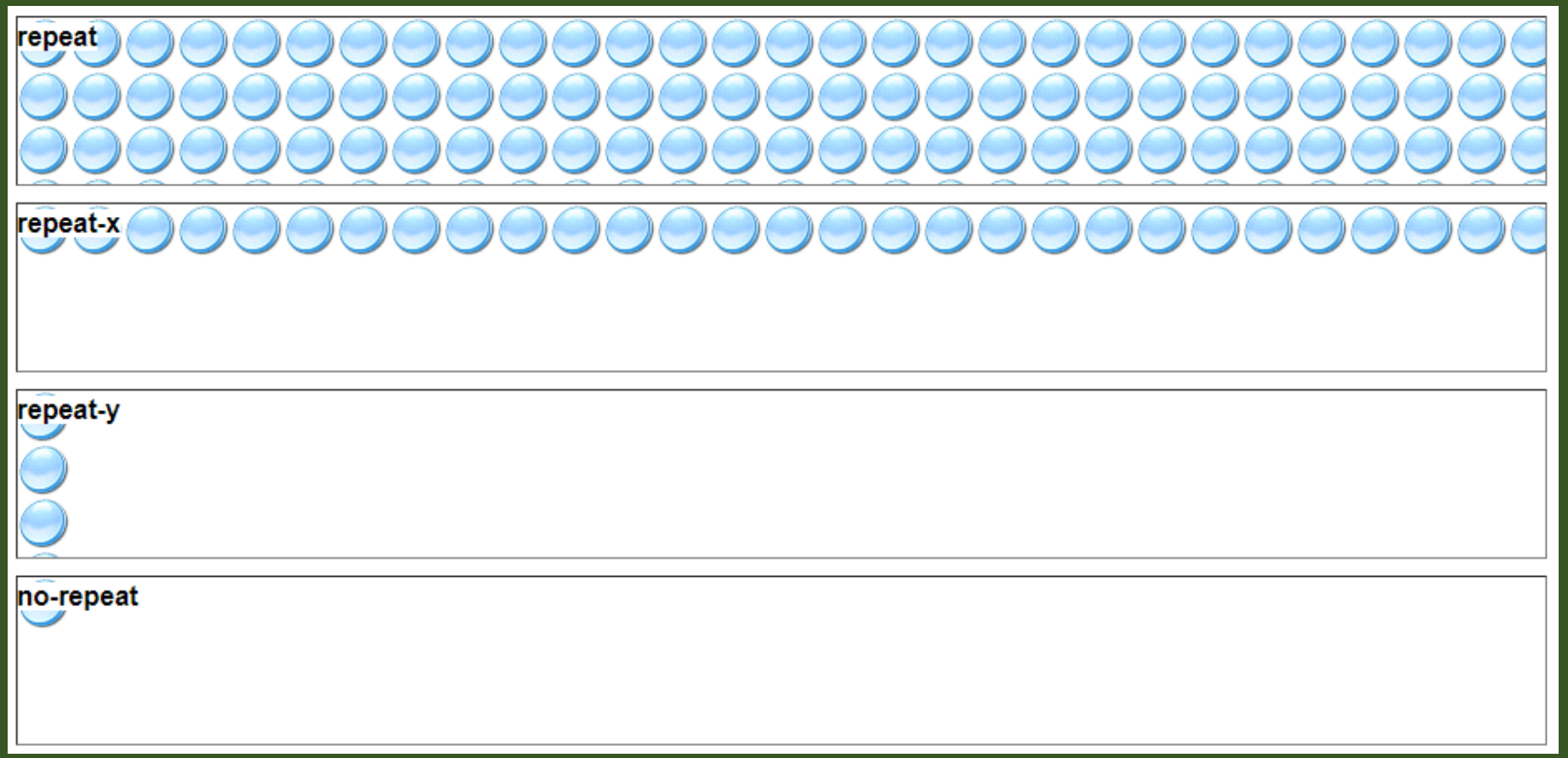
repeat-y - powtarzanie tła tylko w kierunku pionowym

no-repeat - brak powtarzania tła (zostanie wyświetlone jako pojedynczy obrazek)

space - przestrzeń pomiędzy obrazkami w tle zostanie dobrana w taki sposób, aby żadna grafika nie była przycięta (CSS 3 - MSIE 9, Opera)

round - wymiary obrazka w tle zostaną dopasowane w taki sposób, aby żadna grafika nie była przycięta (CSS 3 - MSIE 9, Opera)

Przykład

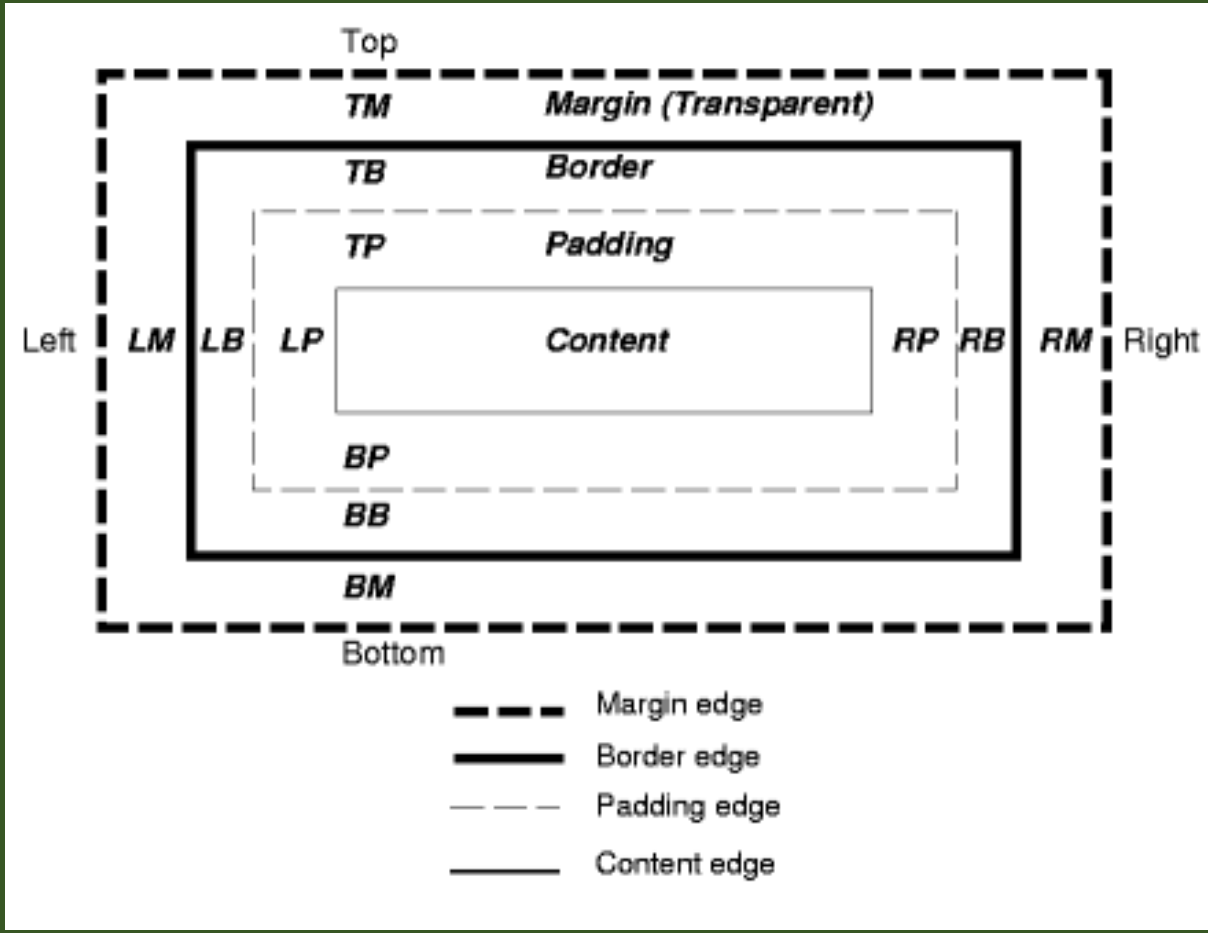


Własności Marginesów

Model pudełkowy

Każdy element generuje w dokumencie prostokątny obszar zwany pudełkiem (ang. Box model). Pudełko składa się z:

- **Zawartości - "Content" (np. test, obrazek itd.)**
- **Otaczających marginesów wewnętrznych - "Padding"**
- **Obramowania - "Border"**
- **Marginesów - "Margin"**



Górny

selektor { margin-top: rozmiar }

Selektorem może być praktycznie dowolny znacznik.

Natomiast jako "rozmiar" należy podać konkretną wartość w jednostkach długości. Wpisanie auto ustali wartość automatyczną.

Polecenie wprowadza dodatkowy odstęp między danym elementem a elementem poprzedzającym.

Dolny

selektor { margin-bottom: rozmiar }

Lewy

selektor { margin-left: rozmiar }

Prawy

selektor { margin-right: rozmiar }

Analogicznie jak poprzedni margines – działanie odnosi się odpowiedni do dolnej, lewej lub prawej krawędzi.

Atrybuty mieszane

selektor { margin: wartości atrybutów }

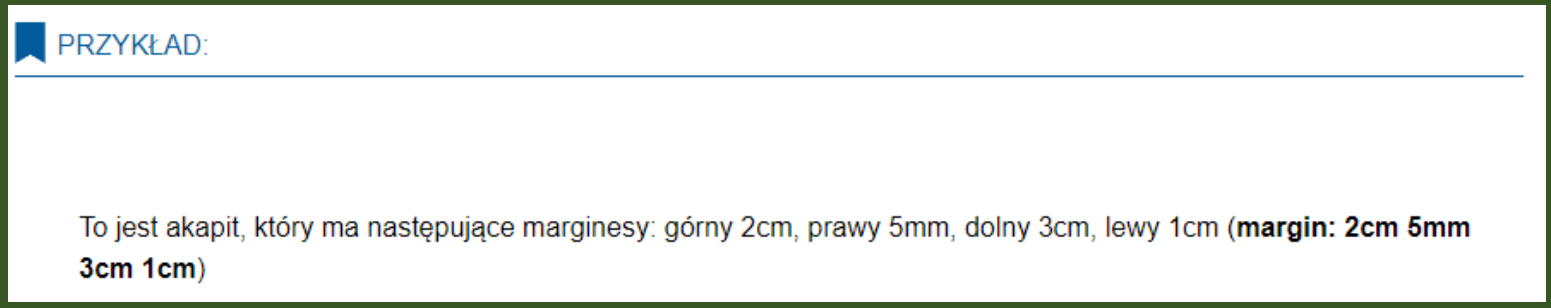
Jako "wartości atrybutów" należy podać:

Jedną wartość - wtedy wszystkie marginesy będą jednakowe.

Dwie wartości - z których pierwsza oznacza górny i dolny margines, natomiast druga - lewy i prawy.

Trzy wartości - z których pierwsza oznacza górny margines, druga - jednocześnie lewy i prawy, a ostatnia - dolny.

Cztery wartości - które oznaczają kolejno marginesy: górny, prawy, dolny, lewy.



Załamywanie marginesów

W przypadku sąsiadowania ze sobą lub zagnieżdżania wewnątrz siebie elementów posiadających marginesy, może zajść proces załamywania marginesów zewnętrznych (ang. collapsing margins), polegający na połączeniu kilku sąsiadujących odstępów w jeden o rozmiarze pojedynczego marginesu, a nie sumy składowych.

Przykład:

```xhtml
<div style="margin-bottom: 20px">1</div>
<div style="margin-top: 50px">2</div>
```

na ekranie zobaczymy dwa bloki, pomiędzy którymi margines będzie wynosił 50px, a nie 70px (nie suma).

Górny wewnętrzny

selektor { padding-top: rozmiar }

jako "rozmiar" należy podać konkretną wartość w jednostkach długości.

Polecenie to pozwala zdefiniować dodatkowy wewnętrzny odstęp pomiędzy elementami, np. tekstem i obramowaniem tabeli.

PRZYKŁAD:

Tekst w tej komórce tabeli jest oddalony od górnego brzegu obramowania o 1cm (**padding-top: 1cm**)

Analogicznie można wpływać na wewnętrzne marginesy dolne, prawe i lewe:

selektor { padding-bottom: rozmiar } **selektor { padding-right: rozmiar }**

selektor { padding-left: rozmiar }

Więcej w Internecie i literaturze.

Uwaga: CSS ma to do siebie, że wymaga sporo wprawy, zmysłu artystycznego i doświadczenia – im więcej będziesz ćwiczyć i testować różne rozwiązania tym lepiej.

KONIEC cz.1