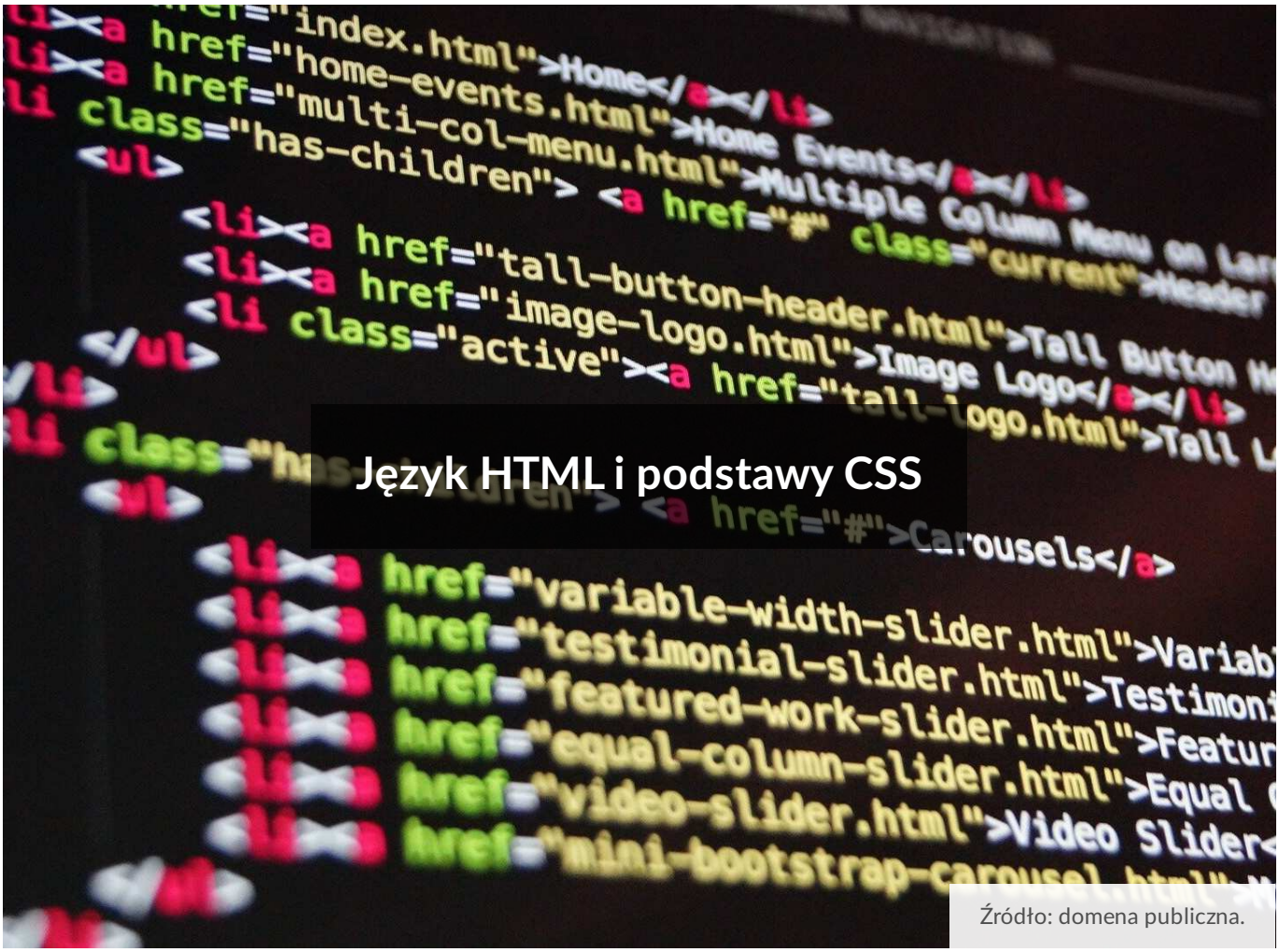
Język HTML i podstawy CSS

- Wprowadzenie
- Przeczytaj
- Infografika
- Sprawdź się
- Dla nauczyciela



Źródło: domena publiczna.

Przy tworzeniu stron internetowych niezbędna jest znajomość języków HTML i CSS. W tym e-materiale zastosujemy obowiązujące w nich podstawowe zapisy.

HTML (ang. *HyperText Markup Language*) to język opisowy, w którym za pomocą znaczników (inaczej mówiąc: tagów) określamy **zawartość witryny**. Mogą to być również hiperłącza, obrazki, akapity tekstu, nagłówki oraz wiele innych elementów składowych strony, w tym kontrolki formularzy.

CSS (ang. *Cascading Style Sheets*) to kaskadowe arkusze stylów, służące do opisania **wyglądu** oraz **położenia** elementów witryny, zdefiniowanych uprzednio w HTML.

Więcej informacji o tworzeniu stron internetowych znajdziesz w e-materiałach:

- Tworzenie stron internetowych,
- Szkielet strony internetowej.

> **Twoje cele**
>
> - Poznasz podstawowe konstrukcje językowe wykorzystywane w HTML.
> - Umieścisz w witrynie WWW hiperłącza, grafiki oraz teksty.
> - Wyszczególnisz elementarne sposoby wpływania na styl wybranych elementów witryny w języku CSS.

Przeczytaj

Znaczniki (tagi) w HTML

Przypomnijmy ogólną konwencję języka opartego na znacznikach (pojedynczych lub podwójnych). Znaczniki zapisujemy w nawiasach ostrych, wewnątrz których mogą również znajdować się tak zwane atrybuty z zapisanymi w cudzysłowie wartościami.

Znacznik pojedynczy:

```
<znacznik atrybut="wartosc">
```

Znacznik podwójny (tj. złożony z dwóch tagów – otwierającego i zamykającego):

```
<znacznik atrybut="wartosc"> </znacznik>
```

Zwróćmy uwagę, że jeżeli znacznik jest podwójny, to jego ewentualne atrybuty umieszczamy w tagu otwierającym. W klasycznej konwencji wartość atrybutu powinna zostać zapisana w cudzysłowie, nawet jeśli jest typu liczbowego.

Hiperłącza (linki)

Znacznik definiujący hipełącze na stronie internetowej wygląda następująco:

```
<a href="https://zpe.gov.pl">ZPE</a>
```

Nazwa tagu `<a>` pochodzi najprawdopodobniej od angielskiego słowa *anchor*, które oznacza kotwicę. Zwróćmy uwagę, że w kodzie źródłowym hiperłącze jest złożone z dwóch znaczników: ma tag otwierający oraz zamykający.

Wynika to z faktu, że należy zdefiniować obszar, którego kliknięcie aktywuje link. Czasem podlinkowany może być wieloliniowy tekst albo np. tekst wraz z obrazem, wówczas przeglądarka musi wiedzieć, dokąd obowiązuje link.

Atrybut `href` to skrót od ang. *hypertext reference*. Określa adres dokumentu, do którego hiperłącze ma zaprowadzić. *Reference* w języku angielskim oznacza odniesienie i rzeczywiście czasami tak określamy linki – mówimy, że są to odnośniki do innych dokumentów.

> **Ciekawostka**
>
> Atrybut `href` nie jest niezbędny. Standard HTML dopuszcza istnienie znaczników `<a>`, bez wpisania adresu dla linku – jest to wówczas informacja, że w tym miejscu może się jeszcze pojawić link z adresem (tzw. *placeholder*).

Hiperłączem nazywamy jedynie taki element `<a>`, który posiada określoną wartość atrybutu `href` – sam element `<a>` odnośnikiem nie jest.

Hiperłącze może także mieć atrybut `target` (z ang. cel), który określa, gdzie docelowo w hierarchii kart przeglądarki ma trafić podlinkowany dokument:

```
1 <a href="https://zpe.gov.pl" target="_blank">ZPE</a>
```

Możliwe wartości atrybutu `target` :

- `target="_self"` – otwórz stronę w tej samej karcie/ramce, w której znajduje się link; ponieważ jest to zachowanie domyślne, można ten atrybut pominąć;
- `target="_blank"` – otwórz adres w nowej, nieużywanej karcie przeglądarki; nie należy nadużywać tego mechanizmu; aby zapewnić użytkownikowi komfort, otwieraj nowe karty tylko tam, gdzie rzeczywiście jest to potrzebne, np. przy wskazywaniu linków prowadzących poza stronę;
- `target="_parent"` oraz `target="_top"` – otwórz adres hiperłącza w odpowiedniej ramce – jest to związane z tzw. framesetem (ang. zestaw ramek). Wartość `_parent` otworzy witrynę w ramce o jeden poziom wyższej we framesetowej hierarchii, zaś `_top` w nadrzędnej ramce.

Warto wiedzieć, że budowanie witryny na ramkach jest przestarzałym rozwiązaniem, z którego rezygnuje się ze względu na SEO oraz wygodę użytkownika.

Obrazy w witrynie

Obrazek można wstawić z użyciem `HTML` dzięki znacznikowi `<img>` (od ang. *image* – obrazek).

```
1 <img src="logo.jpg" alt="Moja firma">
```

Zwróćmy uwagę, że jest to **tag pojedynczy**. Mamy tu bowiem do czynienia z obrazem, który jest obiektem. W takim wypadku nie trzeba dookreślać, gdzie się zaczyna, a gdzie kończy, ponieważ o tym decyduje rozmiar źródłowej grafiki lub ewentualnie określone przez nas właściwości CSS (szerokość, wysokość).

Atrybut `src` to z ang. *source* – ścieżka dostępu do źródłowej grafiki. Atrybut `src` **jest wymagany** dla obrazu (ang. *required*), ponieważ bez źródłowego pliku graficznego ciężko mówić o istnieniu obrazu w dokumencie.

Atrybut `alt`, czyli ang. *alternative*, to alternatywny, tekstowy opis, który oddaje, czym jest dana grafika, co na niej się znajduje. Jest to również **atrybut wymagany**. Dlaczego atrybut `alt` jest według standardu konieczny?

Pamiętajmy, że na stronę internetową trafią także użytkownicy, którzy wyłączyli w swojej przeglądarce ładowanie obrazów, bądź są osobami słabowidzącymi, korzystającymi z czytników witryn. Wówczas taki alternatywny opis, zapewniający słowną reprezentację zawartości obrazu, pomoże im komfortowo przeglądać witrynę.

Wspomnijmy jeszcze o znanej rozterce dotyczącej sposobu domykania tagów w standardzie HTML5. Porównajmy je ze starszymi standardami XHTML i HTML 4.01 – jak powinien wyglądać zapis?

```
1 <img src="logo.jpg" alt="Moja firma">
```

A może znacznik powinno się zakończyć w taki sposób?

```
1 <img src="logo.jpg" alt="Moja firma" />
```

Zgodnie ze specyfikacją HTML5 w tagach pojedynczych **obowiązuje brak kończącego znaku** `/`, czyli poprawna jest wersja pierwsza. Jednak warto wiedzieć, że zapis znany z XHTML (domknięcie tagu, wersja druga) **nie spowoduje błędu w przeglądarce** – będzie ona wiedzieć, iż jest to domknięcie znacznika „w starym stylu".

Wiele osób nadal używa wersji znanej z XHTML – jest to przykład pokazujący, jak płynne, powolne i **kompatybilne wstecz** jest wprowadzanie nowych standardów sieciowych.

Paragrafy czy podwójne złamanie linii?

Akapit (paragraf, ustęp) tekstu to znacznik `<p>` (od ang. *paragraph*). Znacznik ten pozwala podzielić dłuższy tekst na krótsze akapity:

```
1 <p>Lorem ipsum dolor sit amet, consectetur elit,
2 vivamus blandit varius dui, in congue urna fringi.</p>
3
4 <p>Suspendisse a magna felis. Donec fermentum
```

```
5 lacus quis quam tincidunt lobortis.</p>
```

Osoby po raz pierwszy używające języka HTML bardzo często (siłą nawyków z edytorów tekstowych) zamiast stosować akapity tekstu, wstawiają co kilka zdań podwójne znaczniki `<br><br>`, które również tworzą wizualne wrażenie istnienia akapitu:

```
1 Lorem ipsum dolor sit amet, consectetur elit,
2 vivamus blandit varius dui, in congue urna fringi.
3 <br><br>
4 Suspendisse a magna felis. Donec fermentum
5 lacus quis quam tincidunt lobortis.
```

Tag `<br>` od ang. *break*, czyli złamanie linii – jest w HTML odpowiednikiem użycia klawisza `Enter` na klawiaturze (kod sterujący końca linii). Natomiast dwa następujące po sobie złamania linii, jedno po drugim, siłą rzeczy tworzą przerwę w tekście, która wygląda podobnie do akapitu. Lepiej jednak użyć znacznika `<p>`, bo zyskujemy dzięki temu wpływ np. na rozmiar odstępów pomiędzy akapitami poprzez użycie stylów CSS.

Znaczników `<br><br>` nie da się dopasować do indywidualnych potrzeb, gdyż stanowią po prostu złamanie linii. Paragraf rządzi się innymi prawami – za jego pomocą określamy przeglądarce fragment tekstu, w którym ma zastosować wskazany krój czcionki, jej rozmiar, kolor, odstępy poziome i pionowe oraz szereg innych właściwości.

W kontekście SEO paragrafy również spisują się efektywniej niż `<br><br>`, gdyż lepiej przekazują robotowi modularną strukturę zawartości tekstowej witryny. Sprawniej także współpracują z mechanizmem responsywności serwisu.

Natomiast znacznika `<br>` można używać, aby złamać linię, gdy to rzeczywiście jest potrzebne – np. w tabeli albo gdy publikujemy poezję lub poemat:

```
1 <p>
2   Litwo, Ojczyzno moja! ty jesteś jak zdrowie;<br>
3   Ile cię trzeba cenić, ten tylko się dowie,<br>
4   Kto cię stracił. Dziś piękność twą w całej ozdobie<br>
5   Widzę i opisuję, bo tęsknię po tobie.
6 </p>
```

Nagłówki w HTML

Sekcje dokumentu HTML, podobnie jak w tradycyjnych, papierowych wydaniach gazet, rozpoczynają się od nagłówków (z ang. *headings*). W standardzie HTML ustanowiono sześć rozmiarów nagłówków – odpowiadają im znaczniki od `<h1>` do `<h6>`:

```
<h1>Nagłówek stopnia pierwszego</h1>
<h2>Nagłówek stopnia drugiego</h2>
<h3>Nagłówek stopnia trzeciego</h3>
<h4>Nagłówek stopnia czwartego</h4>
<h5>Nagłówek stopnia piątego</h5>
<h6>Nagłówek stopnia szóstego</h6>
```

Jak widać, nagłówki są znacznikami podwójnymi – wynika to z faktu, że zawierają one tekst. Domyślnie największy rozmiar czcionki ma nagłówek `<h1>`, a każdy kolejny przyjmuje proporcjonalnie mniejszą wielkość, aż do nagłówka `<h6>`:

Nagłówek stopnia pierwszego

Nagłówek stopnia drugiego

Nagłówek stopnia trzeciego

Nagłówek stopnia czwartego

Nagłówek stopnia piątego

Nagłówek stopnia szóstego

Rozmiary czcionki oddają także **hierarchię ważności** nagłówków. W kontekście SEO szczególnie ważny jest tekst zamknięty w nagłówku `<h1>` – powinien zawierać frazy ważne dla rezultatów wyszukiwania, zgodne z faktyczną zawartością podstrony.

Nagłówki definiują poszczególne sekcje dokumentu – mechanizm znany z gazet, ulotek, edytorów tekstu przenieśliśmy także do internetu.

Po poznaniu elementarnych zapisów języka HTML, czas na opis fundamentów użycia technologii CSS.

Selektory, atrybuty i wartości w CSS

Jak wiemy, arkusze CSS służą do opisania **wyglądu elementów witryny**, uprzednio zdefiniowanych w HTML. Kodem CSS można wpływać na **pozycję elementów**. W związku z taką definicją pojawia się pytanie, w jaki sposób przypisać style do konkretnego elementu HTML.

Aby móc nadać wybranemu obiektowi styl, należy go uchwycić, czyli wybrać spośród wszystkich innych stylów w dokumencie. Jako **uchwyty** stosujemy w CSS tzw. **selektory**. Ogólny schemat zapisywania kodu CSS prezentuje się następująco:

```
selektor
{
   właściwość: wartość;
}
```

Selektor (jak sama nazwa wskazuje: selekcja to inaczej wybór) określa jednoznacznie, do których uchwyconych elementów kodu HTML zostaną zastosowane właściwości podane w tzw. **ciele selektora**, czyli te zawarte pomiędzy nawiasami klamrowymi.

Każda **właściwość** (atrybut, cecha) określa, jaki aspekt wyglądu obiektu zmieniamy, zaś wartość tego atrybutu zapisana jest po dwukropku i zakończona średnikiem, np.:

```
body
{
   background-color: gray;
   font-family: Arial;
}
```

W tym przypadku selektorem uchwyciliśmy sekcję `<body>`, czyli całe ciało witryny, nadając mu szary kolor tła oraz krój pisma Arial. Wstawione znaki białe (spacje, przejścia do nowej linii) są opcjonalne – można je usunąć z kodu CSS, przygotowując tzw. wersję minimalną arkusza. Wówczas plik zajmie mniej miejsca i nieco szybciej się wczyta do przeglądarki internauty. Z kolei dla programisty będzie on mniej czytelny.

W praktyce najczęściej pracujemy w klasycznym arkuszu posiadającym znaki białe, zaś jego tzw. *wersję min* (minimum) przygotujemy dopiero po zakończeniu prac projektowych. Jest to tzw. **wersja produkcyjna kodu**, która ostatecznie trafi na serwer. Zapis pozbawiony znaków białych wyglądałby tak:

```
body{background-color:gray;font-family:Arial;}
```

Operator myślnika umieszczony w kodzie oddaje **hierarchię właściwości**: najpierw wystąpiła **właściwość główna** `background`, następnie operator myślnika `-`, a potem **subatrybut** właściwości głównej, czyli `color`. Taka hierarchia oddaje **logiczne powiązania** pomiędzy wybranymi atrybutami obiektów.

Uchwycenie elementów: ID czy class?

Zapoznajmy się z następującym zapisem:

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

Taki selektor uchwyci wszystkie akapity `<p>` w dokumencie i ustawi w nich czerwony kolor tekstu. Jednak nie zawsze jest to sytuacja pożądana – czasami chcemy ostylować tylko pojedynczy akapit. W celu dystynktywnego uchwycenia konkretnych obiektów w dokumencie zastosujemy w arkuszach stylów **klasę** (ang. *class*) albo **identyfikator** (ang. ID).

Identyfikatory mogą zostać użyte do uchwycenia jednego elementu w sposób unikalny – nie mogą istnieć w kodzie HTML dwa elementy o takiej samej wartości atrybutu `id`. Zapis prezentuje się następująco:

Arkusz stylów CSS:

```
#czerwony_akapit
{
   color:red;
}
```

Kod `HTML`:

```
 <p id="czerwony_akapit"></p>
```

Jak widać, w przypadku zastosowania identyfikatora selektor w CSS dodatkowo poprzedzamy operatorem #. Jest to informacja dla przeglądarki, że mamy w tym selektorze do czynienia z identyfikatorem. Jeśli zaś chodzi o nazwę identyfikatora (tutaj jest to nazwa:

`container`), to w standardzie HTML 4 identyfikator musi zaczynać się literą (a-z, A-Z), a dopiero po niej może nastąpić dowolna liczba liter, cyfr, myślników, podkreśleń.

Natomiast w HTML5 identyfikator musi zawierać przynajmniej jeden znak i nie może zawierać żadnej spacji. Unikalność identyfikatora w HTML jest wymagana i obsługiwana na poziomie tzw. hierarchii DOM. Na przykład:

```
<p id="pierwszy"></p> <p id="pierwszy"></p> //źle!
```

To duży błąd! Każdy identyfikator w kodzie musi przecież posiadać **własną, unikalną nazwę**:

```
<p id="pierwszy"></p> <p id="drugi"></p> //ok!
```

W przeciwieństwie do identyfikatorów klasy mogą zostać użyte do uchwycenia **dowolnej liczby elementów.** Zapis prezentuje się następująco:

Arkusz stylów `CSS`:

```
.zielony_akapit {
  color: green;
}
```

Kod `HTML`:

```
<p class="zielony_akapit">Zielony tekst</p>
```

Jak widzimy, w wypadku zastosowania klas selektor w CSS dodatkowo poprzedzamy znakiem kropki. Częstym błędem jest założenie, że skoro za pomocą identyfikatora możemy uchwycić tylko jeden element, to w takim razie używając klas, można uchwycić tylko i wyłącznie kilka obiektów. To nieprawda – elementów mających atrybut `class` może być w HTML dowolna liczba, w tym także tylko jeden obiekt.

Kiedy więc użyć identyfikatora, a kiedy klasy? Jeżeli chcemy uchwycić dwa lub więcej elementów w CSS, używamy klasy, ponieważ nadanie tego samego `id` różnym elementom łamie regułę unikalności identyfikatora. Co zrobić w przypadku, gdy chcemy uchwycić tylko jeden element? Możemy użyć zarówno identyfikatora, jak i klasy. Który sposób jest lepszy?

To złożony problem. Ustanowienie identyfikatora w HTML wymusza na przeglądarce obsługę mechanizmu unikalności tego elementu – obiekt zostaje wówczas wyróżniony w specjalny sposób w tzw. hierarchii DOM. Jeżeli jedynym celem jest ostylowanie elementu, to ustanowienie unikalnego `id` wydaje się niepotrzebnym, nadmiarowym zabiegiem.

Identyfikator stosujemy w tych elementach, które zamierzamy później uchwycić w skryptach JavaScript albo które będą służyć jako tzw. punkty nawigacyjne witryny, inaczej nazywane kotwicami nawigacyjnymi. Natomiast w innych przypadkach użycie identyfikatora – chociaż nieoptymalne pod kątem wykorzystania zasobów – nie jest błędem.

Rozmiar elementu: `width`, `height`

Za określenie rozmiaru danego elementu, np. obrazka na płótnie przeglądarki, odpowiadają właściwości `width` (ang. szerokość) oraz `height` (ang. wysokość).

Arkusz stylów CSS:

```
.logo {
  width: 250px;
  height: 100px;
}
```

Kod HTML:

```
<img class="logo" src="logo.jpg" alt="Moja firma">
```

> **Ciekawostka**
>
> Wiele osób popełnia literówki w zapisie obu tych właściwości ze względu na różne końcówki tych słów: `th` oraz `ht`. Zwróćmy uwagę, iż **poprawne** są formy: `width` i `height`, zaś **najczęstsze pomyłki** wyglądają tak: `widht` lub `heigth`.

Stylizowanie czcionek w CSS

Poznajmy teraz najpopularniejsze właściwości CSS, za pomocą których wpłyniemy na najważniejsze aspekty wyglądu tekstów umieszczonych na stronie internetowej.

Krój czcionki: `font-family`

Ustawienie kroju czcionki polega na podaniu jako wartości jej rodziny (ang. *family*), czyli w praktyce nazwy. W wielu projektach krój czcionki warto ustawić dla sekcji `body` – wówczas jednokrotne wpisanie `font-family` sprawi, że ta rodzina czcionek będzie obowiązywała na całej stronie, dopóki w jakimś selektorze tego nie zmienimy. Dzięki wspomnianemu **mechanizmowi kaskadowości** nie ma potrzeby wielokrotnej redeklaracji zapisów dotyczących czcionki – cała siła arkuszy CSS prezentuje się następująco:

```
1 body { font-family: Arial; }
```

Rozmiar czcionki: `font-size`

Jest to właściwość pozwalająca określić rozmiar tekstu. Często stosowane i zalecane jednostki wielkości czcionki to: `px`, `%`, `em`, `ex`. Niezalecane jednostki tej wartości to z kolei: `pt`, `cm`, `mm`, `in`, `pc`.

Można także użyć **stałych tekstowych**: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`. Rozszyfrowanie znaczenia tych zapisów okazuje się proste – *small* – ang. mały, *medium* – ang. średni, *large* – ang. duży, x – ang. *extra*, xx – ang. *double extra* – ang. dodatkowo, dodatkowo oraz podwójnie.

Można też rozmiar czcionki określić relatywnie do elementu nadrzędnego: *smaller* – **mniejszy rozmiar tekstu** niż w elemencie nadrzędnym, *larger* – większy rozmiar w porównaniu do elementu nadrzędnego.

Przykład użycia właściwości:

```
1 p { font-size: 24px; }
```

Waga czcionki: `font-weight`

Ustawiając wagę czcionki, regulujemy tzw. tłustość tekstu. Do wykorzystania mamy stałe tekstowe: `normal` – standardowa grubość tekstu, `bold` – czcionka pogrubiona. Oprócz tego możemy też użyć wartości liczbowej od `100` do `900`, oczywiście pod warunkiem, iż wybrana czcionka obsługuje wskazane grubości.

Wartości zmieniamy **zawsze o pełne** `100`, przy czym `400` odpowiada wartości `normal`, a `700` to czcionka `bold`. Wygląd opcji liczbowych dotyczących grubości czcionki odnajdziemy poniżej – jest to zrzut ekranu z popularnej usługi o nazwie Google Fonts dla czcionki o nazwie Roboto:

Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Istnieje także możliwość określenia grubości czcionki **relatywnie do elementu nadrzędnego**: *lighter* – ang. lżejsza czcionka, czyli mniejsza grubość tekstu, *bolder* – tekst grubszy niż w elemencie nadrzędnym.

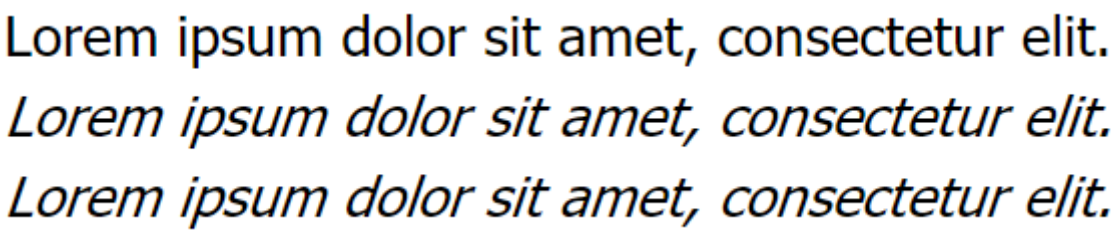
Przykład użycia:

```
p { font-weight: 700; }
```

Styl zapisu czcionki: `font-style`

Ta właściwość czcionki może przyjąć następujące wartości: `normal`, `italic` albo `oblique`.

Czcionka `italic` jest (najprościej mówiąc) delikatnie pochylona w prawo, zaś `oblique` nadaje jej większe pochylenie. Porównajmy, jak wygląda ten sam tekst, który w kolejnych zapisach różny się zastosowaniem innego `font-style` – kolejno: `normal`, `italic`, `oblique`:



Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Przykład użycia właściwości:

```
p { font-style: italic; }
```

Wyrównanie tekstu: `text-align`

Atrybut ten pozwala określić, w jaki sposób wewnętrzna zawartość, np. akapitu (w tym także tekst), zostanie ułożona w tym pojemniku. Popularne wartości to przede wszystkim: `left`, `center`, `right`, `justify`. Ta ostatnia wartość to tzw. **justowanie czcionki** – tekst będzie wówczas tak poukładany, aby zajmował całą dostępną przestrzeń pojemnika.

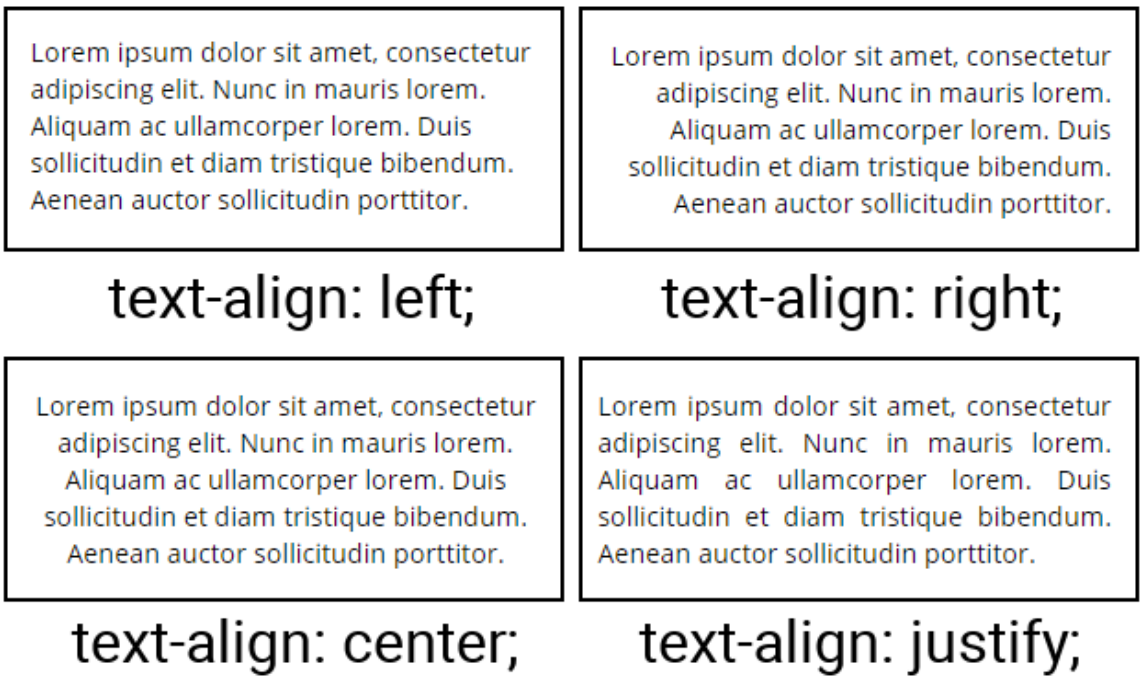
Inne możliwe wartości to:

- `justify-all` – jest to takie samo ustawienie jak `justify`, z tym wyjątkiem, że ostatnia linia tekstu także ulegnie takiemu wyrównaniu;
- `start` – to samo co `left`, chyba że tekst ma być czytany od prawej do lewej (w niektórych językach rzeczywiście tak jest, decyduje o tym wartość właściwości `direction: rtl;` albo `direction: ltr;`);
- `end` – wyrównanie w prawo, również dopasowujące się do wartości `direction` i w razie potrzeby zmieniające się na lewe;
- `match-parent` – wartość dopasowana do elementu nadrzędnego i również reagująca na wartość `direction`.

Przykład użycia:

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

Ilustracja przedstawia cztery podstawowe sposoby ułożenia zawartości:



Zmiana wielkości liter: `text-transform`

Język CSS umożliwia **zmianę rodzaju liter** (wielkie, małe) występujących w oryginalnym tekście, co szczególnie przydaje się w kontekście pozycjonowania – litery nie muszą być zapisane jako duże w kodzie HTML, aby móc zostać zaprezentowane jako takie w przeglądarce.

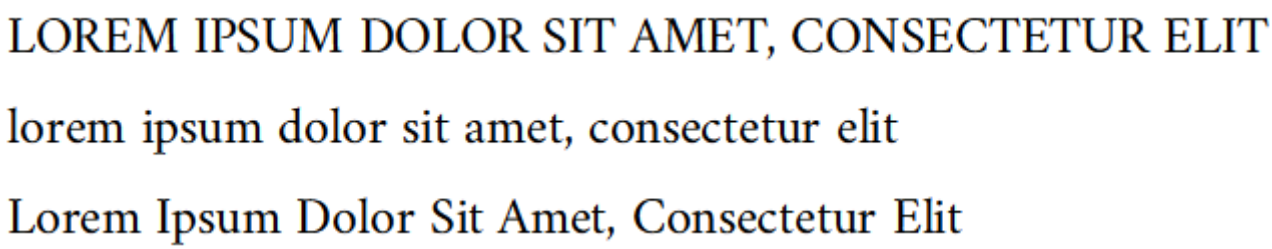
Możliwe wartości to przede wszystkim:

- `uppercase` – zamiana wszystkich litery na wielkie,
- `lowercase` – zamień wszystkie litery na małe,
- `capitalize` – wielkie litery tylko na początku wyrazów,
- `none` – brak zamiany wielkości, przyda się do przywrócenia zmian poczynionych w elementach nadrzędnych.

Przykład użycia:

```
p { text-transform: uppercase; }
```

Oto ten sam akapit tekstu, jednak zastosowano w nim inny rodzaj transformacji tekstu – kolejno: `uppercase`, `lowercase`, `capitalize`:



Dekoracja tekstu: `text-decoration`

Właściwość przydatna do ustalenia **sposobu dekoracji tekstu** z użyciem linii podkreślającej, przekreślającej lub znajdującej się nad tekstem. W praktyce jest to właściwość stosowana dla określania wyglądu linków – często bywają one podkreślone. Możliwe sposoby dekoracji określone są przez następujące subatrybuty:

- `text-decoration-color` – kolor linii wyróżniającej,
- `text-decoration-style` – rodzaj linii, możliwe wartości: `solid`, `double`, `dotted`, `dashed`, `wavy`,
- `text-decoration-line` – położenie linii, możliwe wartości to: `underline`, `overline`, `line-through`, `blink`, `underline overline`, `none`.

Przykład użycia:

```
p
{
  text-decoration-line: underline;
  text-decoration-style: solid;
  text-decoration-color: red;
}
```

Bardzo często w kodzie CSS zastosujemy jednak **zapis skrócony**, czyli jedynie z użyciem głównej właściwości `text-decoration`, za to z odpowiednim doborem liczby kolejno następujących po sobie wartości subatrybutów, np.:

```
p { text-decoration: underline solid red; }
```

Odstęp pomiędzy znakami: `letter-spacing`

Odstęp taki możemy określić z użyciem od jednej do trzech wartości – kolejno będzie to: **minimalny**, **maksymalny** i **optymalny** odstęp między znakami:

```
p { letter-spacing: 0.5em 1em 0.7em; }
```

Wysokość linii tekstu: `line-height`

Za pomocą tej właściwości zdefiniujemy wysokość pojedynczej linii tekstu. Możemy użyć stałej tekstowej `normal` lub liczby określającej mnożnik aktualnej wysokości, podanej np. w procentach lub pikselach:

```
p { line-height: 140%; }
```

Na tym zakończymy przegląd podstawowych właściwości modyfikujących styl tekstu.

Obramowanie: `border`

Do ustawienia obramowania używamy właściwości `border` (ang. *border* – obwódka na krawędzi, obszar graniczny elementu). Możemy zastosować trzy podstawowe subatrybuty:

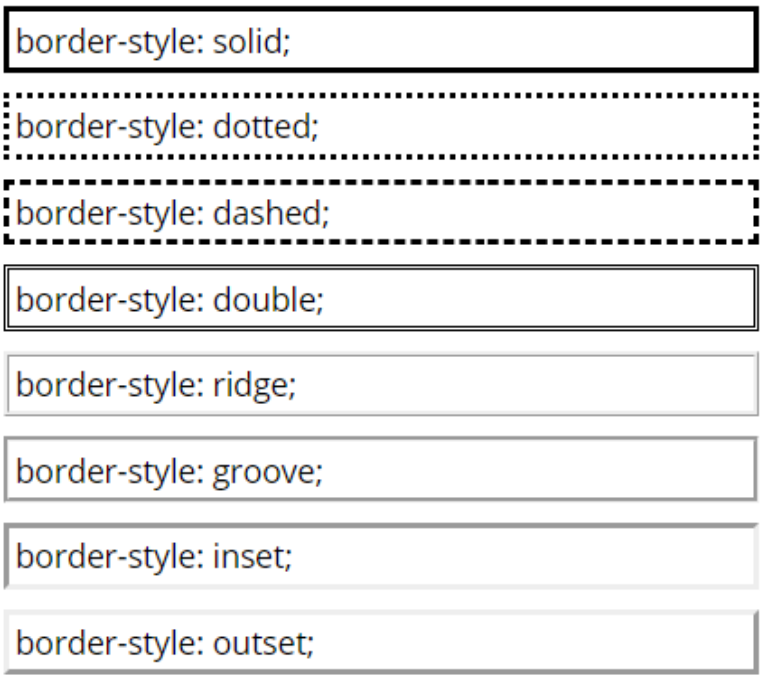
- `border-width` – szerokość obramowania w pikselach,
- `border-style` – rodzaj linii stanowiącej obwódkę (patrz: ilustracja),
- `border-color` – czyli kolor obwódki.

Przykładowe style linii:

```
border-style: solid;
border-style: dotted;
border-style: dashed;
border-style: double;
border-style: ridge;
```

```
6 border-style: groove;
7 border-style: inset;
8 border-style: outset;
```

Wygląd linii obramowania w przeglądarce:



W praktyce często nie zapisujemy następujących po sobie trzech kolejnych właściwości, ograniczając się do zastosowania zapisu skróconego – używamy jedynie właściwości głównej `border`, a po dwukropku wpisujemy trzy wartości rozdzielone spacją i zakończone średnikiem. Najpierw podajemy szerokość w pikselach, potem rodzaj linii i na końcu jej kolor.

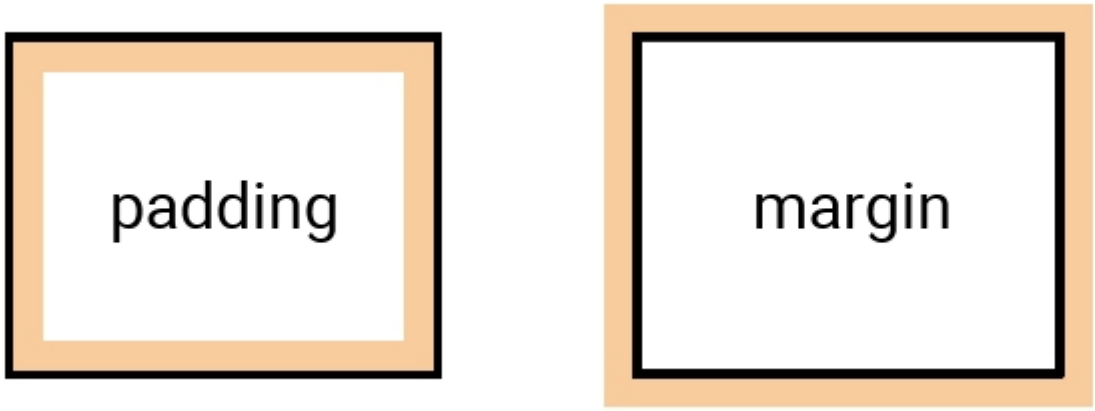
```
1 p { border: 2px solid green; }
```

Można także użyć subatrybutów: `left`, `right`, `top`, `bottom`:

```
1 border-left: 1px solid black;
2 border-right: 1px dashed blue;
3 border-top: 1px groove green;
4 border-bottom: 1px dotted yellow;
```

Odstępy: margin i padding

Właściwość o nazwie `padding` to odstęp nadawany wewnątrz elementu, zaś `margin` definiuje odstęp nadawany na zewnątrz niego:

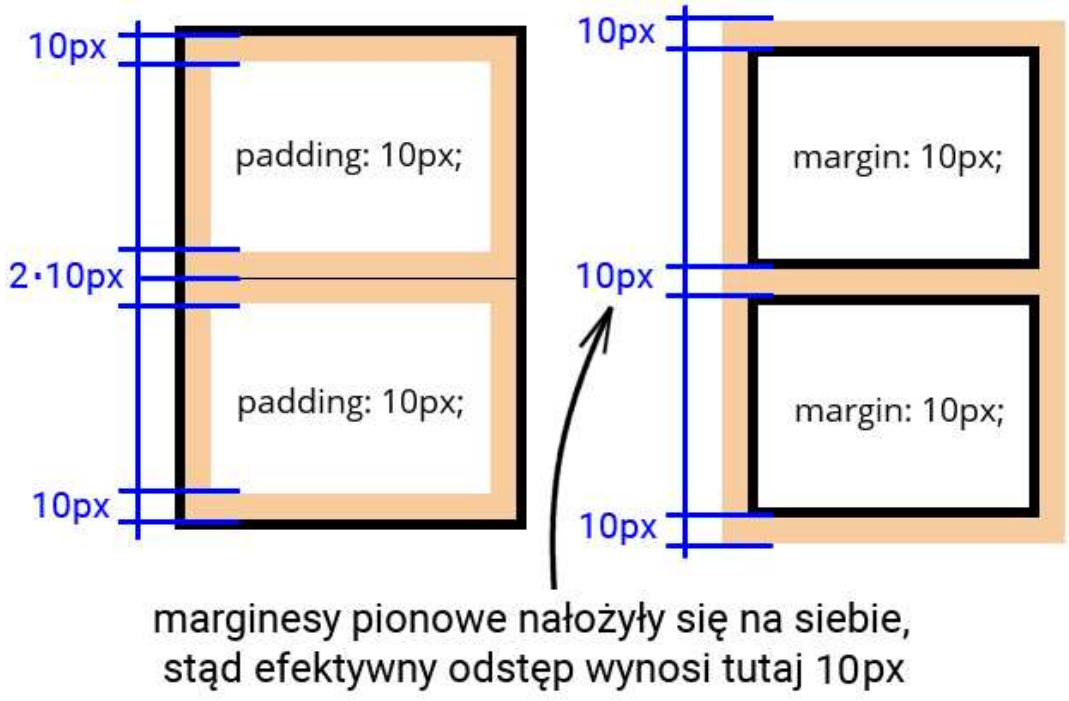


Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

W praktyce bardzo często używamy wymiennie marginesów i paddingów, gdyż nie ma pomiędzy nimi wielu znaczących różnic. Najlepszym przykładem różnicy są pionowe marginesy, które w odróżnieniu od paddingu nakładają się na siebie, zmniejszając odstęp pomiędzy zawartościami obu pojemników.

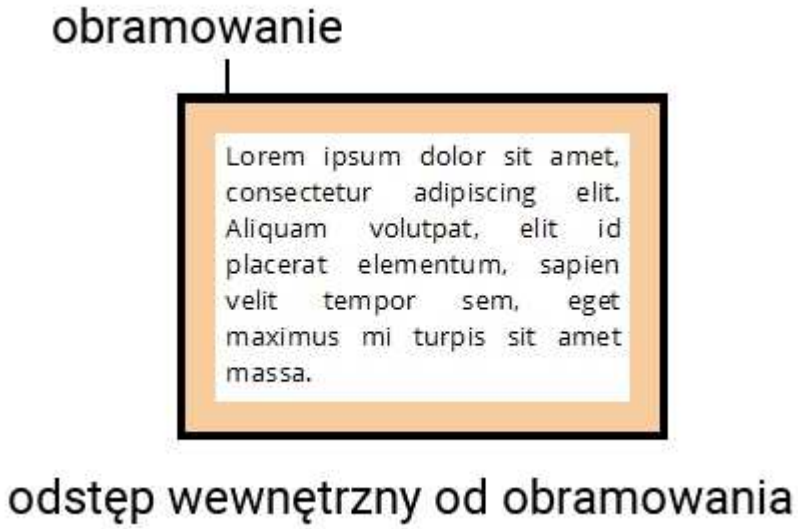
Weźmy jako przykład dwa akapity, ułożone jeden pod drugim. Jeżeli oba te znaczniki HTML będą miały ustawione po 10 pikseli wewnętrznego marginesu (`padding`) z każdej strony, to oznacza, że w pionie wystąpi łącznie 20 pikseli odstępu pomiędzy tekstami wewnątrz akapitów.

Użycie marginesów o tej samej wartości 10 pikseli spowoduje natomiast nałożenie pionowych odstępów na siebie, a zatem w praktyce odstęp pomiędzy zawartościami obu divów będzie wynosić dokładnie 10, a nie 20 pikseli. Przypomnijmy – stanie się tak jedynie w przypadku marginesów pionowych, nie poziomych:



Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Ponadto, jeżeli ustawimy obramowanie, np. akapitu, i chcemy odsunąć zawartość od jego obramowania, odstęp musi być wewnętrzny, czyli trzeba użyć właściwości `padding`:



Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Możliwe zapisy odstępów w CSS

Zapisy marginesów są analogiczne dla właściwości `padding` – to dobra wiadomość, gdyż zapamiętanie przyjętej konwencji dla odstępów wewnętrznych automatycznie sprawi, iż znamy także zapisy odstępów zewnętrznych.

- Taki sam odstęp określony ze wszystkich czterech stron:

```
margin: 10%;
padding: 5px;
```

- Zapis podwójny: najpierw odstęp górny i dolny, potem lewy i prawy:

```
 margin: 20px 10px;
 padding: 5px 15px;
```

- Zapis potrójny: najpierw odstęp górny, potem lewy i prawy (taki sam), a na końcu podajemy wartość odstępu dolnego:

```
margin: 20px 5% 5px;
padding: 5px 5px 10%;
```

- Zapis poczwórny: odstęp górny, prawy, dolny, lewy (zgodnie z ruchem wskazówek zegara):

```
margin: 10px 15px 20px 5px;
padding: 5px 10px 15px 5px;
```

- Odstęp z jednej, wybranej strony – subatrybuty `left`, `right`, `top`, `bottom`:

```
margin-left: 5px; margin-right: 15px;
margin-top: 10px; margin-bottom: 20px;

padding-left: 25px; padding-right: 5px;
padding-top: 5px; padding-bottom: 10%;
```

Słownik

> **hierarchia DOM**
>
> (od ang. *Document Object Model*) określony przez organizację W3C standardowy model struktury całego dokumentu HTML, wykorzystywany przez współczesne przeglądarki internetowe; model ten jest obiektowy i niezależny od platformy sprzętowej czy używanego w projekcie języka programowania
> **kotwica nawigacyjna**
>
> punkt o unikalnym identyfikatorze id zdefiniowany w obrębie dokumentu HTML, do którego można odnosić się z innego miejsca w witrynie; użytkownik serwisu używając kotwicy, szybko przenosi się do wybranych informacji – przeglądarka przewija widok strony do miejsca wystąpienia identyfikatora
> **responsywność**
>
> wspierany przez języki CSS i JavaScript mechanizm dopasowywania sposobu wyświetlania strony internetowej do aktualnie dostępnego rozmiaru okna przeglądarki; wpływa na poprawę komfortu użytkowania witryny na urządzeniach mobilnych
> **SEO**
>
> (od ang. *Search Engine Optimization*) ogół działań, których dokonują programiści webowi w celu poprawy widoczności witryny w wynikach wyszukiwania – w szczególności w najbardziej popularnej wyszukiwarce Google
> **W3C (World Wide Web Consortium)**
>
> organizacja, która zajmuje się ustanawianiem standardów pisania i przesyłu stron WWW

Infografika

Polecenie 1

Zapoznaj się z infografiką. Co jeszcze można na niej umieścić?

Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Polecenie 2

Opracuj notatkę podsumowującą najważniejsze informacje przedstawione na infografice.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Przyporządkuj poszczególnym wartościom atrybutu `target` hiperłącza odpowiednie opisy ich działania.

`_top`	otwórz adres hiperłącza w tej samej karcie, w której znajduje się link
`_blank`	otwórz adres hiperłącza w ramce nadrzędnej
`_self`	otwórz adres hiperłącza w nowej karcie

Ćwiczenie 2

Wskaż, który z nagłówków domyślnie zostanie zapisany najmniejszą czcionką.

- ◯ `<h6>`
- ◯ `<h1>`
- ◯ `<h8>`
- ◯ `<h4>`

Ćwiczenie 3

Dany jest zapis HTML:

```
Google
```

Przyporządkuj części składowe instrukcji do odpowiednich grup.

znacznik

atrybut

wartość

`href` `_blank` `<a>` `</a>` https://google.com `target`

Ćwiczenie 4

Wskaż zdania fałszywe.

- [ ] Znaczników `<br><br>` nie powinno się używać zamiast paragrafów `<p></p>`.
- [ ] Znacznik `<br>` może być używany tylko w podwójnej formie: `<br><br>`.
- [ ] Z użyciem znaczników `<br><br>` łatwiej ostylować tekst akapitu, aniżeli z użyciem `<p></p>`.
- [ ] Znaczników `<br>` nie powinno się w ogóle używać wewnątrz paragrafów `<p></p>`.

Ćwiczenie 5

Akapit tekstu znajdujący się w witrynie powinien mieć `15px` marginesu z lewej strony, `20px` marginesu górnego, `30px` marginesu po prawej stronie oraz tylko `5px` marginesu u dołu elementu. Wstaw w poniższy kod CSS odpowiednie wartości skróconego zapisu, tak aby spełnić te założenia.

```
p
{
margin: [ ]px [ ]px [ ]px [ ]px;
}
```

5 | 15 | 30 | 20

Ćwiczenie 6

Wskaż, której właściwości należy użyć, aby w języku CSS ustawić krój czcionki akapitu na wartość Verdana.

- ◯ `font-style: Verdana;`
- ◯ `font-family: Verdana;`
- ◯ `font-name: Verdana;`
- ◯ `font-weight: Verdana;`

Ćwiczenie 7

Wstaw do opisu działania klas i identyfikatorów w CSS odpowiednie brakujące elementy:

Klasy mogą zostać użyte do uchwycenia _____.

Identyfikatory mogą zostać użyte do uchwycenia _____.

W przypadku zastosowania klasy, selektor w CSS dodatkowo poprzedzamy operatorem _____.

W przypadku zastosowania identyfikatora, selektor w CSS dodatkowo poprzedzamy operatorem _____.

tylko jednego elementu | # | dowolnej liczby elementów | : | @

tylko dwóch elementów | . | *

Ćwiczenie 8

Wskaż, jakiego należy użyć zapisu, aby w języku CSS ustanowić obramowanie elementu niebieską linią przerywaną.

- ◯ `border: 1px ridge blue;`
- ◯ `border: 1px inset blue;`
- ◯ `border: 1px groove blue;`
- ◯ `border: 1px dashed blue;`

Dla nauczyciela

Autor: Mirosław Zelent
Przedmiot: Informatyka

Temat: Język HTML i podstawy CSS

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres podstawowy

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> Zakres podstawowy. Uczeń:
>
> 3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami:
>
> f) tworzy stronę internetową zgodnie ze standardami, wzbogaconą tabelami, listami, elementami dynamicznymi, posługuje się arkuszem stylów, korzysta z oprogramowania i serwisów przeznaczonych do tworzenia stron; potrafi opublikować własną stronę w internecie;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Poznasz podstawowe konstrukcje językowe wykorzystywane w HTML.
- Umieścisz w witrynie WWW hiperłącza, grafiki oraz teksty.

- Wyszczególnisz elementarne sposoby wpływania na styl wybranych elementów witryny w języku CSS.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Język HTML i podstawy CSS". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel prosi o intuicyjne wyjaśnienie, czym jest HTML, a czym – CSS.
2. Ustalenie celu lekcji i kryteriów sukcesu.

Faza realizacyjna:

1. Nauczyciel dzieli klasę na grupy. Każda z nich opracowuje jeden fragment e-materiału. Do swojego wystąpienia przygotowuje się, korzystając również z zasobów internetowych.
2. Uczniowie w grupach przygotowują mapy myśli dotyczące danych zagadnień.
3. Grupy prezentują swoją pracę. Pozostała część klasy zadaje pytania. W razie konieczności nauczyciel zadaje pytania pomocnicze, które pozwolą członkom grupy

uzupełnić wypowiedź.
4. Nauczyciel inicjuje dyskusję na temat tego, jak zmieniały się możliwości języka HTML od czasu jego powstania.
5. Uczniowie wykonują wskazane przez nauczyciela ćwiczenia interaktywne.

Faza podsumowująca:

1. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie przygotowują swoje propozycje rozszerzenia infografiki.

Wskazówki metodyczne:

- E-materiał może być wykorzystany do wprowadzenia uczniów w projekt polegający na stworzeniu prostej strony WWW.