Notatki do nauki

--------------------------------------------------------------------------------------------------------------------------------------

- - - CSS - - -

--------------------------------------------------------------------------------------------------------------------------------------

**1. Składnia i dodawanie kodu CSS:**

* Składnia CSS: p { color : red; }

legenda: p selektor

color właściwość

red wartość

color : red; deklaracja

{ color : red; } blok deklaracji

p { color : red; } reguła CSS

Arkusz stylów - wiele reguł zwykle w oddzielnym pliku CSS

Sposoby dodawania kodu CSS:

I sposób: poprzez link do zewnętrznego pliku css np.

<link rel="stylesheet" href="nazwa\_stylu.css">

II sposób: poprzez znacznik <style> w sekcji <head> np.

<style> p { border: 2px dotted white; } </style>

III sposób: poprzez atrybut style np.

<p style="color: white;"> przykładowy tekst </p>

**2. Selektory:**

* Selektor typu:

Selektor utworzony poprzez wskazanie konkretnego taga i wprowadzenie bloku deklaracji, czyli przykładowo:

body

{

background: black;

color: white;

}

Jeżeli w <body> zostaną użyte kolejne znaczniki takie jak <p> lub <div> to są to dzieci <body> i odziedziczą po nim ten styl CSS. Jeżeli chcemy w takim przypadku nadać indywidualny styl dla paragrafu <p> inny niż ma <body> to musimy stworzyć kolejną regułę CSS dla paragrafu. Można stosować kilka selektorów po przecinku jeżeli chcemy nadać im ten sam styl, czyli przykładowo:

p, div

{

font-size: 20px;

}

* Selektor id:

Selektor utworzony poprzez odwołanie się ze znakiem # do atrybutu id zawartego w znaczniku, czyli przykładowo:

Kod CSS:

#styl-paragrafu

{

color: red;

}

Kod html:

<p id="styl-paragrafu"> przykładowy tekst </p>

* Selektor klasy:

Selektor utworzony poprzez odwołanie się do wartości atrybutu class w danym znaczniku poprzez poprzedzenie go kropką. Przykład:

Kod CSS:

.chapter

{

color: red;

}

.first-chapter

{

color: blue;

}

Kod html:

<!-- element z dwiema klasami first-chapter oraz chapter -->

<p class="first-chapter chapter">przykładowy tekst</p>

<!-- element z jedną klasą chapter -->

<p class="chapter">przykładowy tekst</p>

Liczy się kolejność odwołań do klas. Ostateczny rezultat, gdy dla dwóch klas zmieniamy tę samą właściwość, zależy od tego która klasa w kodzie css jest ostatnia i tej właśnie klasy kod będzie zastosowany dla elementu posiadającego dwie klasy.

* Łączenie selektorów klasy i id:

Łączenia takiego dokonujemy wpisując selektory jeden po drugim poprzedzając je znakiem # oraz znakiem kropki np.

Kod CSS:

#nazwa\_identyfikatora.nazwa\_klasy { color: red; }

Kod html:

<div class=" nazwa\_klasy " id=" nazwa\_identyfikatora ">przykładowy tekst</div>

* Kombinatory:

Selektor potomka:

<style> article h1 {color: red;} </style>

<article>

<header>

<h1> przykładowy tekst </h1>

</header>

</article>

Potomkowie znacznika <article> to w powyższym kodzie <header> i <h1>

Selektor dziecka:

<style> header > h1 {color: red;} </style>

<article>

<header>

<h1> przykładowy tekst </h1>

</header>

</article>

Dzieckiem znacznika <header> jest <h1>. Znacznik <h1> nie jest dzieckiem tagu <article>. Selektor potomka tworzymy z użyciem znaku >.

Selektor następnego sąsiada:

<style> h1 + p {color: red;} </style>

<article>

<header>

<h1> przykładowy tekst </h1>

<p> inny przykładowy tekst nr 1 </p>

<p> inny przykładowy tekst nr 2 </p>

</header>

</article>

Następnym sąsiadem znacznika <h1> jest paragraf <p> o zawartości „inny przykładowy tekst nr 1” i tylko ten paragraf dostanie styl z selektora. Dla utworzenia takiego selektora stosujemy znak +.

Selektor następnych sąsiadów:

<style> h1 ~ p {color: red;} </style>

<article>

<header>

<h1> przykładowy tekst </h1>

<p> inny przykładowy tekst nr 1 </p>

<p> inny przykładowy tekst nr 2 </p>

<p> inny przykładowy tekst nr 3 </p>

<p> inny przykładowy tekst nr 4 </p>

</header>

</article>

Następnymi sąsiadami znacznika <h1> są wszystkie paragrafy <p> i one wszystkie dostaną styl z selektora. Dla utworzenia takiego selektora stosujemy znak ~.

* Selektor uniwersalny:

Selektor nadający styl wszystkim elementom strony:

Kod CSS:

\* { border: 2px solid red; }

Kod html:

<body>

<ul>

<li>Przykładowy tekst</li>

<li>Przykładowy tekst</li>

<li>Przykładowy tekst</li>

</ul>

</body>

Wszystkie elementy strony mają czerwoną ramkę czyli: <body>, <ul>, <li> i nawet <html>.

* Selektor atrybutu:

Pozwala wybrać te elementy html, które posiadają konkretny atrybut lub posiadają atrybut razem z wartością. Przykładowo:

Kod CSS:

[data-red] {background-color: red;}

[data-color="orange"] {background-color: orange;}

Kod html:

<article>

<ul>

<li data-red> przykładowy tekst </li>

<li data-color="orange"> przykładowy tekst </li>

<li> przykładowy tekst </li>

</ul>

</article>

Jeżeli w kodzie CSS zastosujemy zapis [atrybut \*= "wartość"] to dany styl zostanie zastosowany do wszystkich elementów html, które posiadają wartość atrybutu, która wynosi „wartość”, nawet jeżeli jest częścią dłuższej nazwy wartości i wielkość liter ma znaczenie.

Jeżeli w kodzie CSS zastosujemy zapis [atrybut ~= "wartość"] to dany styl zostanie zastosowany do wszystkich elementów html, które posiadają wartość atrybutu, która wynosi „wartość”, nawet jeżeli jest fragmentem dłuższej nazwy wartości, ale przy jednoczesnym oddzieleniu wyrazów spacjami. Wielkość liter ma znaczenie.

Jeżeli w kodzie CSS zastosujemy zapis [atrybut |= "wartość"] to dany styl zostanie zastosowany do wszystkich elementów html, które posiadają wartość atrybutu, która wynosi „wartość”, nawet jeżeli jest fragmentem dłuższej nazwy wartości, ale przy jednoczesnym oddzieleniu wyrazów myślnikami lub spacją. Wielkość liter ma znaczenie.

Jeżeli w kodzie CSS zastosujemy zapis [atrybut = "wartość" i] to dany styl zostanie zastosowany do wszystkich elementów html, które posiadają wartość atrybutu, która wynosi „wartość” przy czym ignorowana jest wielkość znaków.

* Łączenie wielu selektorów:

Przykład:

Kod CSS:

div p.test[class \*= "PL"] {background: green;}

Kod html:

<div>

<p class="PL"> przykładowy tekst nr 1 </p>

<p class="PL-pl"> przykładowy tekst nr 2 </p>

<p class="PL test"> przykładowy tekst nr 3 </p>

<p class="EN test"> przykładowy tekst nr 4 </p>

</div>

W powyższym kodzie styl jest nadany przykładowemu tekstowi nr 3 ponieważ zastosowane zostały selektory dla elementów znajdujących się w paragrafie, który występuje wewnątrz znacznika <div> i który jest klasy „test” oraz w atrybucie class posiada fragment tekstu „PL”.

**3. Pseudoklasy:**

* Pseudoklasy są specyficznym rodzajem selektorów ponieważ pozwalają wybierać elementy w zależności od stanu, w którym się znajdują.
* Pseudoklasy checked, disabled, focus:

Kod CSS:

input:checked {outline: 2px solid red;}

input:disabled {outline: 2px dotted grey;}

input:focus {outline: 1px solid orange;}

Kod:html

<form action="test.php">

<input type="checkbox" name="city" id="wawa">

<label for="wawa">Warszawa</label>

<input type="checkbox" disabled name="city" id="krk">

<label for="krk">Kraków</label>

<input type="checkbox" name="city" id="gd">

<label for="gd">Gdańsk</label>

</form>

Pseudoklasy wybierają element ze względu na stan w jakim się znajduje:

:checked – zaznaczony element

:disabled – nieaktywny

:focus – element posiadający focus

Powyższy kod nadaje styl czerwonego obramowania zaznaczonemu checkbox’owi, szarego kropkowanego obramowania nieaktywnemu checkbox’owi oraz pomarańczowego obramowania checkbox’owi z focus’em. Gdy wyżej wymienione check’boxy nie są zaznaczone lub nie ma na nich focus’u, to style nie są na nich widoczne.

* Pseudoklasy first, last, nth child:

Kod CSS:

ul li:first-child {text-decoration: underline;}

li:last-child {text-decoration: underline;}

li:nth-child(3n) {color: red;}

li:nth-child(2) {color: green;}

li:nth-child(odd) {background-color: grey;}

li:nth-child(even) {background-color: yellow;}

Kod html:

<ul>

<li>przykładowy tekst nr 1</li>

<li>przykładowy tekst nr 2</li>

<li>przykładowy tekst nr 3</li>

<li>przykładowy tekst nr 4</li>

<li>przykładowy tekst nr 5</li>

<li>przykładowy tekst nr 6</li>

<li>przykładowy tekst nr 7</li>

<li>przykładowy tekst nr 8</li>

</ul>

Powyższy kod poprzez pseudoklasy nadaje następujące style dla elementów listy:

:first-child pierwszy element listy ma podkreślenie

:last-child ostatni element listy ma podkreślenie

:nth-child(3n) co trzeci element listy ma czerwony tekst

:nth-child(2) drugi element listy ma zielony tekst

:nth-child(odd) parzyste elementy listy mają szare tło

:nth-child(even) nieparzyste elementy listy mają żółte tło

Można łączyć selektory ul oraz li tak jak poniżej lecz nie jest to konieczne, bo wystarczyłoby użycie jedynie selektora li w tym przypadku:

ul li:first-child {text-decoration: underline;}

* Pseudoklasa not:

Kod CSS:

input:not(.pierwsza-klasa):not(.druga-klasa) {background-color: yellow;}

input:not([type = "email"]) {background-color: grey;}

Kod html:

<input type="text" class="pierwsza-klasa" value="1">

<input type="text" class="druga-klasa" value="2">

<input type="email" value="testowymail@nazwa.com">

Pseudoklasa :not wybiera elementy niespełniające warunków selektora. W powyższym kodzie zostanie nadany styl żółtego tła dla pola <input>, w którym nie występuje klasa „pierwsza-klasa” oraz nie występuje klasa „druga-klasa”, czyli w ostatnim elemencie zawierającym adres e-mail. Natomiast styl szarego tła zostanie nadany pierwszym dwóm elementom <input> ponieważ nie zawierają one atrybutu type z wartością „email”.

* Pseudoklasy hover, active, visited:

Kod CSS:

a:hover {color: red;}

a:active {color: orange;}

a:visited {color: grey;}

Kod html:

<ul>

<li><a href="https://google.com"> gogle </a></li>

<li><a href="https://youtube.com"> YouTube </a></li>

<li><a href="https://facebook.com"> facebook </a></li>

</ul>

:hover - nadanie stylu elementowi gdy znajduje się nad nim kursor

:active - nadanie stylu elementowi gdy go klikamy

:visited - nadanie stylu elementowi gdy już raz został kliknięty

**4. Pseudoelementy:**

* Pseudoelementy first-letter, first-line, selection:

Pseudoelementy służą do nadania stylu specyficznym częściom elementów np.

::selection - nadanie stylu tekstowi zaznaczonemu przez użytkownika

::first-letter - nadanie stylu pierwszej literze

::first-line - nadanie stylu pierwszej linii

Kod CSS:

p::first-letter {font-style: italic;}

div::first-line {color: red;}

article::selection {background: yellow;}

Kod html:

<article> przykładowy tekst </article>

<div> przykładowy tekst </div>

<p> przykładowy tekst </p>

* Pseudoelementy before, after:

Kod CSS:

li::after {content: "XYZ"; border: 2px solid red;}

p::before {content: "123";}

ul {list-style-type: none;}

li:before {content: "symbol unicode";}

Kod html:

<ul>

<li> przykładowy tekst </li>

<li> przykładowy tekst </li>

</ul>

Wyłączenie stylu punktowania listy oraz nadanie punktora w postaci symbolu unicode w kodzie CSS:

ul {list-style-type: none;}

li:before {content: "symbol unicode";}

**5. Kaskadowość:**

* Kaskada to zestaw reguł decydujących o tym jak rozwiązać konflikty, które występują w naszym kodzie CSS. Gdy zachodzi konflikt między deklaracjami to żeby je rozróżnić kaskada bierze pod uwagę trzy elementy według następującej kolejności:

- Pochodzenie arkusza stylów np. domyślne style przeglądarki, zdefiniowane przez twórcę strony itd.

- Specyficzność selektorów, określenie ich kolejności względem siebie, który uznać za pierwszy.

- Kolejność źródłowa w arkuszu stylów.

Te zasady pozwalają zdecydować, która reguła zwycięży, gdy istnieje konflikt.

* Gdy kod CSS określa na kilka różnych sposobów jaki styl ma zostać zastosowany dla danego elementu, to powstaje konflikt. Poniżej widzimy sytuację gdzie rozwiązanie konfliktu następuje poprzez zastosowanie stylu o najwyższym priorytecie. Kolor czcionki jest niebieski ponieważ największy priorytet z pośród selektora typu, selektora id i selektora klasy ma selektor identyfikatora:

Kod CSS:

h2 {color: red;}

#heading {color: blue;}

.article-title {color: yellow;}

Kod html:

<h2 id="heading" class="article-title">Nagłówek</h2>

* Style domyślne przeglądarki, zwane inaczej stylami agenta użytkownika są to style, które istnieją na stronie nawet wtedy gdy nie mamy określonych jeszcze żadnych styli css. Są to style o niskim priorytecie, domyślnie zapisane w kodzie źródłowym przeglądarki. Można sprawdzić te style wchodząc na odpowiednią stronę www danej przeglądarki.
* Priorytety źródeł stylów w kaskadzie:

IV. Agent użytkownika - style domyślne przeglądarki (najniższy priorytet).

III. Style autora - style twórcy strony www.

II. Style autora ważne - style twórcy strony www z deklaracjami oznaczonymi adnotacją !important, które nadpisują normalne style, niezależnie gdzie są na stronie.

1. Ważne style użytkownika przeglądającego stronę np. gdy stosuje dodatki do przeglądarki w postaci rozszerzeń, pluginów (najwyższy priorytet).

Kod CSS:

h1 {color: green !important;}

h1 {color: red;}

W powyższym kodzie kolor zielony czcionki w nagłówku h1 ma wyższy priorytet.

* Jeżeli nie da się rozwiązać konfliktu reguł CSS za pomocą źródła stylów, to przeglądarka sprawdza specyficzność deklaracji. Wtedy następuje określenie specyficzności przez przeglądarkę w dwóch częściach:

1. Dla stylów liniowych w atrybucie „style” elementów html, wpływają tylko na ten element i przesłaniają style z arkuszów styli z osobnych plików czy znacznika style.
2. Specyficzność selektora tzn. reguły określające, które selektory są ważniejsze, czyli mające większą specyficzność. Na przykład dwie klasy w selektorze zwyciężą jedną klasę. Jedna klasa zwycięży nad typami znaczników. Identyfikator zwycięży nad klasami itd.

Kod CSS:

#top-header {color: silver;} /\* priorytet 2 \*/

.header.heading {color: blue;} /\* priorytet 3 \*/

.header {color: orange;} /\* priorytet 4 \*/

h1 { color: green;} /\* priorytet 6 najniższy \*/

h1 {color: red;} /\* priorytet 5 \*/

Kod html:

<h1 id="top-header" class="header heading" style="color: aqua;"><!--priorytet 1 najwyższy-->

Przykładowa treść nagłówka

</h1>

* Specyficzność selektorów:

Im bardziej specyficzny jest selektor tym bardziej wygrywa względem innych selektorów. Natomiast jeśli oba selektory mają tą samą specyficzność to wygrywa ostatni zapisany w kodzie.

Kolejność priorytetów selektorów:

Selektor znaczników: div { }

Klasy, pseudoklasy, selektory atrybutów: .klasa { } [type=”text”] { } :hover { }

Identyfikatory: #top-header { }

Style inline: <p id=”paragraph1” class=”info” style=”color: yellow”> </p>

* Klasyfikacja medalowa w specyficzności:

Specyficzność jest obliczana za pomocą konwencji klasyfikacji medalowej w postaci 4 slotów. Każdy slot ma wartość początkową zero: 0 0 0 0

Dla uproszczenia stosuje się klasyfikację medalową porównując specyficzność:

0 0 0 0

0 złoto srebro brąz

Dwa brązowe medale są warte więcej niż jeden brązowy. Jedno srebro jest warte więcej niż wiele brązowych medali. Złoto jest warte więcej niż pozostałe medale. Większą wartość od złota ma styl inline w elemencie html, ale zwykle jest pomijany i niezalecany.

* Obliczanie specyficzności dla brązowych medali:

Skrajnie prawy slot ma najwyższą wartość, określa się go pierwszym slotem oraz medalem brązowym. Jego wartość powiększa się z każdym kolejnym znacznikiem w selektorze jak widzimy poniżej:

Kod CSS:

p {color: silver;} /\* 0 0 0 1 \*/

p span {color: blue;} /\* 0 0 0 2 \*/

article > section p span {color: red;} /\* 0 0 0 4 \*/

Kod html:

<article>

<section>

<p>Przykładowy <span>testowy</span> tekst</p>

</section>

</article>

Im wyższa wartość slotu tym specyficzność jest większa. W powyższym kodzie odnoszącym się do tego samego elementu w <span> wygra ostatnia reguła ponieważ 4 brązowe medale mają większą wartoś niż 2 czy 1 brązowy medal. Jeżeli występuje sytuacja, że mamy taką samą ilość np. brązowych medali to wyższy priorytet będzie miał zapis, który jest ostatni w kodzie.

* Obliczanie specyficzności dla srebrnych medali:

Drugi slot od prawej strony jest ważniejszy od pierwszego, jest to medal srebrny. Jego wartość rośnie o jeden za każdym razem gdy pojawia się selektor:

- klasy .small-text

- pseudoklasy :hover

- atrybutu [type=”text”]

Kod CSS:

.small-text:first-child {color: blue;} /\* 0 0 2 0 \*/

.box .small-text[data-color="red"] {color: red;} /\* 0 0 3 0 \*/

.small-text {color: green;} /\* 0 0 1 0 \*/

Kod html:

<p class="box content">

Przykładowy <span data-color="red" class="small-text">tekst</span> testowy </p>

Trzy srebrne medale powodują wygraną koloru czerwonego czcionki.

* Obliczanie specyficzności dla złotych medali:

Trzeci slot od prawej strony jest ważniejszy od drugiego i pierwszego, jest to medal złoty. Jego wartość rośnie o jeden za każdym razem gdy pojawia się selektor identyfikatora.

Kod CSS:

p #word {color: red;}

#paragraph #word {color: blue;}

span.small-text[data-color] {color: green;}

Kod html:

<p id="paragraph">

Przykładowy <span id="word" data-color="red" class="small-text">tekst</span> testowy

</p>

Dwa złote medale powodują wygraną koloru niebieskiego czcionki. Jeżeli mamy tyle samo medali złotych i srebrnych to przeważa większa ilość medali brązowych itp.

* Jeżeli skonfliktowane style mają takie same źródło np. arkusz stylów autora strony www, oraz taką samą specyficzność to wybierana jest ostatnia reguła z arkusza stylów. Wartość właściwości, która wygrała kaskadę nazywamy wartością kaskadową.

**6. Dziedziczenie styli:**

* Dziedziczenie po rodzicu:

Dziedziczenie wartości właściwości CSS polega na tym, że niektóre style mogą zostać odziedziczone jeżeli dzieci elementów nie mają wartości kaskadowej i wtedy wartość dziedziczą po rodzicu. Dotyczy to części właściwości CSS, przeważnie związanych z czcionkami, właściwościami list, ułożeniem tekstu na elemencie, właściwościami obramowania tabel i innymi pomniejszymi stylami.

* Przesłonięcie dziedziczenia:

Gdy wprowadzamy kaskadę to przesłania ona zależności dziedziczenia styli. Staje się nadrzędna nad mechanizmem dziedziczenia. Nie następuje w takim przypadku dziedziczenie domyślnego stylu. Jeżeli mamy element html będący dzieckiem innego elementu i został nadany mu styl w kodzie CSS odnoszący się konkretnie tylko do niego, to element ten nie dziedziczy stylu po rodzicu, tylko pomija dziedziczenie i stosuje styl z kodu CSS.

* Właściwości dziedziczone i niedziedziczone:

Aby sprawdzić czy dana właściwość podlega dziedziczeniu czy nie, można wejść np. na stronę Mozilla Developer Network. Jest to portal skupiający wiedzę dotyczącą html, CSS oraz JS. Najważniejsze style, które są dziedziczone to:

border-collapse font-stretch text-align-last

border-spacing font text-decoration-color

caption-side letter-spacing text-indent

color line-height text-justify

cursor list-style-image text-shadow

direction list-style-position text-transform

empty-cells list-style-type visibility

font-family list-style white-space

font-size orphans widows

font-style quotes word-break

font-variant tab-size word-spacing

font-weight text-align word-wrap

font-size-adjust

* Wymuszenie dziedziczenia z inherit oraz wartość initial:

Polecenie initial powoduje użycie domyślnej dla przeglądarki wartości koloru. Polecenie inherit powoduje wymuszenie dziedziczenia ponad priorytet kaskady. Przykład:

Kod CSS:

a {color: blue;}

header {color: green;}

header a {color: inherit;} /\* Tu kolor czcionki będzie zielony \*/

header p {color: initial;} /\* Tu kolor czcionki będzie domyślny \*/

Kod html:

<header>

<h2>Witamy na stronie</h2>

<ul>

<li><a href="#">Strona główna</a></li> <!-- Kolor czcionki będzie zielony -->

<li><a href="#">Blog</a></li> <!-- Kolor czcionki będzie zielony -->

</ul>

<p>Testowy tekst</p> <!-- Kolor czcionki będzie domyślny -->

</header>

**7. Kolory:**

* Nadawanie kolorów może być przeprowadzone dla elementów takich jak tło, czcionka czy obramowanie. Przykładowo:

Kod CSS:

p {

border-style: dotted;

border-color: red;

border-width: 3px;

color: green; }

article {

background-color: orange;

border: 2px solid red; }

Kod html:

<p> Testowy tekst </p>

<article> Kolejny testowy tekst </article>

* Można posługiwać się precyzyjnym określaniem koloru z funkcją rgb stosując zapis liczbowy z zakresu od 0 do 255. Dodatkowo funkcja rgba daje możliwość określenia przezroczystości stosując zapis liczbowy z zakresu od 0 do 1. Przykładowo:

<p style="color: rgb(0, 0, 0);">czarny rgb</p>

<p style="color: rgb(255, 255, 255); background-color: black;">biały rgb</p>

<p style="color: rgb(255, 255, 0);">żółty rgb</p>

<p style="color: rgb(0, 255, 0);">zielony rgb</p>

<p style="color: rgb(0, 0, 255);">niebieski rgb</p>

<p style="color: rgb(0, 255, 255);">aqua rgb</p>

<p style="color: rgb(255, 0, 255);">różowy magenta rgb</p>

<p style="color: rgb(128, 128, 128);">szary rgb</p>

<p style="color: rgb(255, 0, 0);">czerowny rgb</p>

<!-- Funkcja rgba umożliwiająca ustalanie przezroczystości: -->

<p style="color: rgba(255, 0, 0, 1);">czerowny rgba</p>

<p style="color: rgba(255, 0, 0, 0.75);">czerowny 75% rgba</p>

<p style="color: rgba(255, 0, 0, 0.50);">czerowny 50% rgba</p>

<p style="color: rgba(255, 0, 0, 0.25);">czerowny 25% rgba</p>

<p style="color: rgba(255, 0, 0, 0.1);">czerowny 10% rgba</p>

* Kolory mogą być określane w postaci szesnastkowej tzn. w szesnastkowym systemie liczbowym. Przykładowo:

<p style="color: #000000;">czarny</p>

<p style="color: #FF0000;">czerowny</p>

<p style="color: #FFFFFF; background-color: black;">biały</p>

<p style="color: #F00;">czerowny skrócony</p>

Kolory jako wartość HSL, gdzie H oznacza kolor, S to nasycenie w procentach, L to jasność w procentach. Przykładowo:

<p style="color: hsl(360, 100%, 50%);">czerwony</p>

<p style="color: hsl(360, 100%, 0%);">czarny</p>

**8. Jednostki:**

* Piksele i procenty:

Kod CSS:

#content {

font-size: 18px; /\* Wielkość czcionki \*/

border: 3px solid red; } /\* Szerokość, rodzaj i kolor obramowania \*/

div.test {

background: green; /\* Kolor tła \*/

width: 200px; /\* Szerokość tła \*/

height: 40px; /\* Wysokość tła \*/

padding: 5px;

margin: 12px;

color: black; } /\* Kolor czcionki \*/

article {

background: red; /\* Kolor tła \*/

width: 70%; /\* 70% szerokości elementu nadrzędnego czyli div \*/

height: 55%; } /\* 55% wysokości elementu nadrzędnego czyli div \*/

Kod html:

<p id="content"> testowy tekst </p>

<div class="test">

testowy tekst

<article> testowy tekst </article>

</div>

Właściwości CSS padding służą do generowania przestrzeni wokół zawartości elementu, wewnątrz dowolnych zdefiniowanych granic.

Właściwości CSS margin służą do tworzenia przestrzeni wokół elementów, poza zdefiniowanymi granicami. Istnieją właściwości umożliwiające ustawienie marginesu dla każdej strony elementu: górny, prawy, dolny i lewy.

* Jednostki rzeczywiste:

Kod CSS:

.cm {font-size: 1cm;}

.mm {font-size: 7mm;}

.in {font-size: 0.6in;} /\* 96px = 2,54cm \*/ /\* cal \*/

.pt {font-size: 10pt;} /\* 1pt = 1/72in \*/ /\* point \*/

.pc {font-size: 5pc;} /\* 1pc = 12pt \*/ /\* pika \*/

Kod html:

<article>

<p class="cm"> 1cm testowy tekst </p>

<p class="mm"> 7mm testowy tekst </p>

<p class="in"> 0.6in testowy tekst </p>

<p class="pt"> 10pt testowy tekst </p>

<p class="pc"> 5pc testowy tekst </p>

</article>

Zalecane do stosowania dla stron www są jednostki px, %, em.

* Jednostki względne em:

Jednostki względne jek em określają wielkość względem innego elementu, w przeglądarce to będzie domyślna wielkość czcionki np. 16px, która jest dziedziczona po <body>.

Kod CSS:

article p {font-size: 1.2em;} /\* 16px x 1.2 = 19.2px \*/

p span {font-size: 1.25em;} /\* 19.2px x 1.25 = 24px ponieważ span dziedziczy po p \*/

Kod html:

<body>

<article>

<h1> Naglówek </h1>

<p><span> Autor: Łukasz </span></p>

<p>Testowy tekst</p>

<p>Testowy tekst</p>

</article>

</body>

* Problem jednostki em:

Ponieważ poprzez zagnieżdżenie znacznik <ul> występuje wielokrotnie na coraz to kolejnych poziomach, to w wyniku mechanizmu dziedziczenia jednostka względna em powoduje każdorazowe zmniejszanie się czcionki, na każdym poziomie.

Kod CSS:

body {font-size: 16px;}

ul {font-size: 0.8em;}

ul ul {font-size: 1em;} /\* Mechanizm znoszący zmniejszanie się czcionki dla potomka \*/

Kod html:

<ul>

<li> Poziom 1

<ul>

<li> Poziom 2

<ul>

<li> Poziom 3 </li>

</ul>

</li>

</ul>

</li>

</ul>

* Jednostki względne rem:

Jednostka względna rem odwołuje się do elementu głównego strony tzw root'a którym jest selektor html. Z niego brana jest wielkość dla wszystkich elementów korzystających z rem. Problemy związane z dziedziczeniem nie występują.

Kod CSS:

html {font-size: 32px;} /\* Zamiast html można napisać :root \*/

ul {font-size: 0.75krem;} /\* 0.75 x 32px = 24px \*/

Kod html:

<ul>

<li>Poziom 1

<ul>

<li>Poziom 2

<ul>

<li>Poziom 3</li>

</ul>

</li>

</ul>

</li>

</ul>

* Jednostki względne do obszaru roboczego:

Jednostki względne do obszaru roboczego przeglądarki czyli okna renderowania treści bez paska adresu, menu itd.

vh - 1/100 dostępnej wysokości

vw - 1/100 dostępnej szerokości

vmin - 1/100 mniejszej wartości wysokości lub szerokości

vmax - 1/100 większej wartości wysokości lub szerokości

Kod CSS:

div {

background: red;

width: 50vw;

font-size: 5vh; }

article {



background: green;

width: 30vmin; }



Kod html:

<div> Div element </div>

<article> Article element </article>

**9. Czcionki:**

* Czcionki szeryfowe:

Są to czcionki posiadające w poszczególnych literach zakończenia w postaci kreseczek. Przykładem czcionki szeryfowej jest Times New Roman.

* Czcionki bez szeryfowe:

Są to czcionki nie posiadające w poszczególnych literach zakończenia w postaci kreseczek, tak jak ma to miejsce w czcionkach szeryfowych. Przykładem czcionki bez szeryfowej jest Arial.

* Czcionki o stałej szerokości:

Czcionki te charakteryzują się tym, że każda litera, cyfra i symbol mają tą samą szerokość. Przykładem takiej czcionki jest Courier New.

* Właściwość font-family:

<style>

/\* Wypisując w font-family kilka czcionek dajemy możliwość załadowania się

kolejnej czcionki z listy, gdy na urządzeniu nie występuje dana czcionka. \*/

article { font-family: Arial, Helvetica, sans-serif; } /\* Czcionki bezszeryfowe \*/

/\* Dłuższe nazwy czcionek możemy zaspisać w cudzysłowie

lub apostrofach \*/

div { font-family: 'Times New Roman', Times, serif; } /\* Czcionki szeryfowe \*/

/\* Czcionka o stałej szerokości znaków \*/

p { font-family: 'Courier New', Courier, monospace; }

</style>

<body>

<article>Czcionka bezszeryfowa</article>

<div>Czcionka szeryfowa</div>

<p>Czcionka o stałej szerokości znaków</p>

</body>

* Właściwość font-weight czyli grubość czcionki:

<style>

body {

font-size: larger;

font-family: Arial, Helvetica, sans-serif;

}

/\*

Wartości szerokości są z przedziału od 100 do 900 lub

określone są jako: normal, bold, bolder, lighter.

\*/

</style>

<body>

<p style = "font-weight: normal;"> Czcionka normal czyli taka jak 400 </p>

<p style = "font-weight: bold;"> Czcionka bold czyli taka jak 700 </p>

<p style = "font-weight: bolder;"> Czcionka bolder </p>

<p style = "font-weight: lighter;"> Czcionka lighter </p>

<p style = "font-weight: 200;"> Czcionka 200 </p>

<p style = "font-weight: 600;"> Czcionka 600 </p>

<p style = "font-weight: 900;"> Czcionka 900 </p>

</body>

* Zewnętrzne czcionki na stronie www:

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

<style>

/\* Wchodzimy na stronę zawierającą czcionki np. https://fonts.google.com/

Wybieramy czcionkę i kopiujemy komendę linkującą do sekcji <head>.

Następnie odowłujemy się do nowych czcionek poprzez font-family.

Aby znać nazwy naszych nowych czcionek z powyższej strony kopiujemy sobie

do kodu komentarz. \*/

/\* font-family: 'Dosis', sans-serif; \*/

body { font-family: 'Dosis', sans-serif; }

</style>

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Dosis:wght@200&display=swap" rel="stylesheet">

</head>

<body>

<p> Testujemy czcionkę zewnętrzną </p>

</body>

**10. Pozostałe:**

* Emmet to umowny sposób zapisu tekstu, który zostanie przekształcony przez IDE lub jakiś plugin na kod html. Stosowanie go potrafi znacznie przyspieszyć pisanie kodu.
* Selektory możemy wypisywać po przecinku przed blokiem deklaracji. Styl w ten sposób określony będzie działał na zawartość wewnątrz wszystkich tagów przywołanych jako selektory.