Arkusze stylów CSS

Kaskadowe Arkusze Stylów CSS (ang. *Cascading Style Sheets*) to język służący do formatowania dokumentów WWW, utworzonych w językach HTML oraz XHTML.

- CSS stworzono w celu odseparowania formy prezentacji dokumentu (CSS) od jego struktury i zawartości (HTML)
- Arkusz CSS składa się z listy dyrektyw (reguł) określających sposób prezentacji określonej cechy określonego elementu HTML/XHTML (np. kolor czcionki nagłówka poziomu 3, szerokość lewego wewnętrznego marginesu elementu listy, …)
- Kaskada stylów polega na możliwości definiowania stylu w kilku miejscach i określeniu priorytetów poszczególnych arkuszy
  (na wypadek gdyby reguły były sprzeczne)
- Począwszy od CSS 2.1 można definiować odrębne style dla różnych mediów (ekran, drukarka, tv, …)

CSS wywodzą się z języka DSSSL (ang. *Document Style Semantics and Specification Language*).

- DSSSL jest językiem formatowania dokumentów SGML
  (SGML jest zorientowany na przetwarzanie programowe)
- DSSSL pozwala przetwarzać dokumenty SGML na dokumenty bardziej czytelne dla człowieka, w tym szeroko znane formaty RTF, HTML, LaTeX (LaTeX można przetworzyć np. na PDF);
- Inne znane zastosowanie DSSSL to język DocBook – język znaczników zdefiniowany pierwotnie w SGML (obecnie XML), służący do tworzenia dokumentacji technicznej sprzętu komputerowego i oprogramowania, w sposób niezależny od platformy prezentacji;
  Dzięki DSSSL można przetwarzać dokumenty DocBook na różne formaty, np. HTML, HTML HELP, PDF, bez zmian w źródłach

Wersje języka CSS na tle rozwoju HTML

- 1980 – prototyp hipertekstowego systemu informacyjnego
- 1989 – propozycja systemu dla Internetu, projekt WWW
- 1991 – pierwsza publicznie dostępna specyfikacja HTML
- 1993 – propozycja (draft) HTML opublikowana przez IETF
- 1994 – pierwsza propozycja CHSS
- 1995 – HTML 2.0, pierwsza oficjalna specyfikacja IETF
- 1996 – CSS level 1: pierwsza oficjalna specyfikacja CSS
  – HTML 3.0; powstaje W3C
- 1997 – HTML 4.0
- 1998 – CSS level 2:
  – HTML 4.01
- 2004 – CSS level 2.1: Zalecenie (2005: draft; 2007: zalecenie)
- 2008 – wersja robocza (working draft) HTML 5.0
- 2011 – CSS level 3; Podział na moduły (>50), rozwijane niezależnie;
  Obecnie niektóre moduły mają level 3, inne już 4

Zalety CSS:

- Zwiększenie dostępności witryny
  Oddzielenie formy od struktury i zróżnicowanie mediów,
  w tym dla urządzeń przenośnych, osób niewidzących, …
- Ułatwienie modyfikacji formy i treści WWW,
  Zmiana formy sprowadza się do zmiany CSS (nie HTML).
  Zmiana treści nie wymaga formatowania HTML (jest CSS).
- Mniej skomplikowane dokumenty HTML
  Dokumenty HTML zawierające tylko opis struktury są mniej skomplikowane – łatwiej zachować poprawność i czytelność dokumentów; Dokumenty o prostszej i bardziej przejrzystej strukturze są łatwiejsze w utrzymaniu i modyfikacji.
- Większe możliwości
  CSS oferuje większe możliwości formatowania (a zwłaszcza pozycjonowania) niż HTML

Wady CSS:

- Brak ścisłego związku z HTML/XHTML
  Twórcy CSS zakładali, że CSS będzie służył także do innych celów – nie ma ścisłego związku między CSS a HTML
  (specyfikacja CSS nie mówi nic o HTML i *vice versa*)
- Niejednoznaczność
  Oddzielenie CSS od HTML powoduje niejednoznaczności – różne przeglądarki różnie interpretują te same arkusze
- Problemy z implementacją
  Żadna przeglądarka nie implementuje w pełni żadnej wersji CSS (najlepszy wynik – IE 4, ok. ~99% zgodności z CSS 1.0,
  najgorszy wynik – NN 4, część CSS 1.0 ale z grubymi błędami)
  Obecnie ważniejsze przeglądarki (oprócz IE!) obsługują niemal w całości CSS 2.1; obsługa CSS 3.0 jest szczątkowa

Kaskada stylów CSS

- Styl każdego elementu jest efektem wypadkowym działania kaskady stylów: styl wbudowany przeglądarki może być modyfikowany przez dołączony arkusz CSS, element "style" nagłówka dokumentu HTML i atrybut "style" elementu.

Metody dołączania stylów:

- Element *link* w nagłówku HTML:

```
<link rel="stylesheet" type="text/css"
 href="style.css">
```

- Element *style* w nagłówku HTML:

```
<style>
<!--

  -->
</style>
```

- Atrybut *style* elementu HTML

```
<??? style=" … ">
```

Struktura CSS:

- Arkusz CSS składa się z listy deklaracji (statement).
  Wyróżnia się dwa rodzaje deklaracji:
  - @reguła (at-rule), np.:
  - zbiory reguł (rule set), np.:

- @reguła może mieć jedną z dwu postaci:

```
@reguła zawartość;

@reguła {
  zawartość
}
```

  Zawartość zależy od samej reguły.
- Najczęściej są stosowane:
  - @media
  - @import

- Zbiór reguł ma postać:

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

  Można łączyć reguły dla tego samego selektora w bloki:

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

  Można też łączyć selektory w różny sposób:

```
S1, S2, S3 { cecha: wartość }
S1.clA + S2 > S3.clB { cecha: wartość }
```

Słowa kluczowe:

- Słowa kluczowe (należy pisać bez cudzysłowów!)
  predefiniowane wartości, zależnie od cechy:

```
font-style: normal | italic
border-style: solid | dotted |dashed
```

Jednostki długości:

- Jednostki względne
  - em - wysokość czcionki,
  - ex - wysokość litery x,
  - px - piksele (rozmiar px zależy od urządzenia)

```
h1 { margin: 2em; font-size: 1.2em; }
img.thumb { width: 75px; height: 100px; }
```

- Jednostki bezwzględne:
  - cm - centymetry,
  - mm - milimetry
  - in - cale,
  - pt - punkty (1/72 in)
  - pc - pica (12 pt = 1/6 in)
- Wartości procentowe:
  - % - procent

```
div.banner { width: 75% }
td.lp { width: 10% }
```

Zasoby Internetu

- URI (postaci "protokół://domena.com/folder/plik.typ")
  - url("łańcuch") – ścieżka może być bezwzględna lub względna

```
p { background: url("http://www.d.pl/r.png") }
p { background: url("www.d.pl/r.png") }
p { background: url("./r.png") }
```

Kolory:

- Predefiniowane nazwy
  (są to identyfikatory, nie należy stosować cudzysłowów)

```
p { color: teal }
```

- Wartość RGB:
  - #RRGGBB
  - #RGB
  - rgb (r, g, b)
  - rgb (r%, g%, b%)

```
p { color: #00FFFF; }
p { color: #0FF; }
p { color: rgb(0, 255, 255); }
p { color: rgb(0%, 100%, 100%); }
```

Dziedziczenie właściwości

- Element zawarty w innym elemencie dziedziczy właściwości tego elementu:

```
p { color: blue }
<p><em>Lorem</em> ipsum dolor sit amet
```

- Właściwości danego elementu przesłaniają właściwości odziedziczone:

```
p { color: blue }
em { color: teal }
<p><em>Lorem</em> ipsum dolor sit amet
```

Selektory CSS:

- * dowolny element
- E – element E, umieszczony bezpośrednio w body lub wewnątrz innego elementu

```
p { bordercolor: teal; }
```

- E, F – elementy E oraz F, niezależnie od położenia względem siebie oraz innych elementów

```
p, li { bordercolor: teal; }
```

- E F – element F zawarty w elemencie E, bezpośrednio lub pośrednio

```
div p { bordercolor: teal; }
```

- E > F – element F zawarty bezpośrednio w elemencie E

```
div > p { bordercolor: teal; }
```

- E + F – element F umieszczony bezpośrednio za elementem E

```
h1 + p { bordercolor: teal; }
```

- E[atr] – element E z atrybutem atr (dowolnej wartości)

```
img[alt]
```

- E[atr="w"] – element E z atrybutem atr o wartości w

```
td[colspan=2]
```

- E.cl – element E z atrybutem class="cl"
  nazwa klasy może wystąpić wielokrotnie w dokumencie HTML

```
p.comment
<p class="comment">Lorem ipsum dolor sit amet
```

- E#i – element E z atrybutem id="i"; identyfikator elementu musi być unikalny

```
div#banner
<div id="banner">
```

Pseudoklasy elementów aktywnych:

- E:active element (łącze) właśnie aktywowany
- E:hover element (łącze) wskazany, ale nie aktywowany
- E:focus element posiadający fokus

Pseudoklasy odnośników:

Są wykorzystywane do przekształcenia listy odnośników w menu

- a:link łacze jeszcze nie odwiedzone
- a:visited łącze odwiedzone
- a:hover łącze wskazane, ale nie aktywowane (nie kliknięte)
- a:active łącze właśnie aktywowany (w czasie kliknięcia)

Pseudoklasy strukturalne:

- E:first-child element e, który jest pierwszy w innym elemencie
  (podobnie :first-of-type, :last-child, :last-of-type, :nth-child, …)

```
div > p:first-child { … }
```

Pseudoelementy:

- E::first-letter pierwsza litera
- E::first-line pierwsza linia
- E::before łańcuch znaków lub licznik przed elementem
- E::after łańcuch znaków lub licznik za elementem

```
p.new::before { content: "Nowość! " }
li:before { content: counter(…) }
```

Łączenie selektorów:

- Można stosować bardziej złożone konstrukcje selektorów, łącząc różne warianty relacji wzajemnego zawierania. W praktyce większość autorów wykorzystuje znacznie prostsze podejście, w dwóch krokach:
  1. formatowanie domyślne elementów (E)
  2. formatowanie specyficzne dla klas elementów (E.class)
     oraz ewentualnie pojedynczych elementów (E#id)

```
div             { … }
div.menu        { … }
div#banner      { … }
```

W uzasadnionych wypadkach można zastosować łączenie selektorów,
bez przesadnych komplikacji i z użyciem klas:

```
div.menu li         { … }
ol.contens em       { … }
```

Media

Arkusze CSS mogą definiować różne sposoby formatowania, zależnie od medium prezentacji. Wyróżnia się media:

- all – wszystkie media (wartość domyślna)
- screen – ekran komputera
- print – drukarka
- projection – projektor
- handheld – urządzenia przenośne (mały ekran i przepustowość)
- tv – ekran o małej rozdzielczości, bez przewijania
- speech – syntezator mowy
- braille – urządzenia z pismem Brailla

Metody przypisania różnych stylów dla mediów:

- Element link z atrybutem media – w nagłówku HTML:

```
<link rel="stylesheet" href="main.css">
<link rel="stylesheet" href="print.css" media="print">
```

- Dyrektywa @import i/lub @media w pliku CSS:

```
@import url(./print.css) print;
@media screen {
  /* styl dla ekranu komputera */
}
```

RWD

RWD (Responsive Web Design) – technika projektowania stron WWW w taki sposób, aby ich układ dostosowywał się do rozmiaru okna przeglądarki

- Początkowo strony WWW były projektowane wyłącznie z myślą o ekranach komputerów; Pierwsze urządzenia mobilne skalowały stronę tak, aby zmieściła się na ekranie w całości
- Z czasem zaczęły powstawać odrębne strony dla urządzeń mobilnych (domena.com/ → domena.com/m/);
  Wada: trzeba zaprojektować dwa odrębne serwisy WWW
- Obecnie większość stron stosuje podejście RWD,
  najczęściej wykorzystuje się do tego framework CSS Bootstrap

Elementy RWD

- Vieport
  Dyrektywa wyłączająca skalowanie okna (HTML5)
- CSS Media Queries (CSS3)
  Rozszerzenie reguł CSS @media o warunki,
  np. minimalna/maksymalna szerokość okna
- Techniki zmiany układu strony
  – elementy pływające
  – systemy gridowe
  – elementy flex
  – gotowe frameworki CSS

Vieport

- Dyrektywa wyłączająca skalowanie okna (HTML5):

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

CSS Media Queries

- Rozszerzenie reguł CSS @media o warunki,
  np. minimalna/maksymalna szerokość okna;
  Dotyczy tylko mediów: all, screen, print
- Składnia:

```
@media not|only mediatype
  and (mediafeature and|or|not mediafeature)
```

- Przykłady

```
@media (min-width: 951px) {…}
@media only screen and (max-width: 950px) {…}
@media (orientation: landscape) {…}
```

- Można również stosować oddzielne arkusze CSS dla różnych mediów i różnych warunków:

```
<link rel="stylesheet" href="style.css">
<link rel="stylesheet"
   media="screen and (min-width: 951px)"
   href="style-wide.css">
<link rel="stylesheet"
   media="print"
   href="style-print.css">
```

- Zalecane jest podejście "Mobile first" – najpierw reguły jak dla smartfona, potem @media dla innych urządzeń

```
aside {
  display: block;
}

@media (min-width: 951px) {
aside {
  float: left;
  width: 200px;
}
```

Techniki zmiany układu strony

- Elementy pływające
- Systemy gridowe
- Flexbox
- Gotowe frameworki CSS, w tym Bootstrap

  Bootstrap jest systemem gridowym; Definiuje 12 kolumn i 5 szerokości okna: xsm (<576 px), sm (≥576 px) md (≥768 px) lg (≥992 px), xlg (≥1200 px)
  Wersja 3 wykorzystuje elementy pływające, wersja 4 – elementy pływające i flexbox
  https://getbootstrap.com/docs/4.0/getting-started/introduction/

- CSS box-sizing
  Domyślnie ma wartość content-box – rozmiar pudełka BEZ odstępu (padding) i ramki (border); Dla RWD lepsza jest wartość border-box – rozmiar obejmuje też odstęp i ramkę:

  ```
  * {
    box-sizing: border-box;
  }
  ```

- CSS display
  Domyślny sposób wyświetlania elementów (jako liniowe lub blokowe) można zmienić przez właściwość display; Do wyboru jest ponad 20 wartości, (lista wszystkich jest np. tu: https://www.w3schools.com/cssref/pr_class_display.asp), w tym m.in. inline, block, inline-block, flex oraz none:

  ```
  div.flexbox {
    display: flex;
  }
  ```

- Elementy pływające uzyskuje się przez właściwość float:

  ```
  nav, main {
    float: left;
  }
  ```

Przydatne linki:

- Poradnik HTML/CSS W3Schools,
  CSS: https://www.w3schools.com/css/default.asp
  RWD: https://www.w3schools.com/css/css_rwd_intro.asp
  Index reguł: https://www.w3schools.com/cssref/default.asp
  Layout: https://www.w3schools.com/html/html_layout.asp
- Poradnik nt. elementów flex:
  https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- Walidatory HTML/CSS
  – HTML: https://validator.w3.org/
  – CSS: https://jigsaw.w3.org/css-validator/validator

Czcionki

- Czcionki:
  - font-family: rodzaj, rodzaj-ogólny
  - font-size: wielkość [em, pt, px, %] | small | medium| large
  - line-height: wielkość [em, pt, px, %]
  - font-style: normal | italic
  - font-weight: normal | bold | liczba
    100, 200, 300, 400=normal, 500, 600, 700=bold, 800, 900
  - font-variant: normal | smallcaps

```
p { font-family: Arial, sans-serif; }
h1 { font-size: 150% }
```

- Łączone cechy czcionki:
  - font: [font-style font-variant font-weight] font-size [/line-height] font-family

```
body { font: bold italic 12pt/15pt serif }
```

Tekst

- Poszczególne cechy tekstu:
  - color: RGB
  - text-decoration: none | underline | line-through |overline
  - text-transform: none | capitalize | uppercase | lowercase
  - text-align: left | right | center | justify
  - text-indent: wielkość
  - line-height: wielkość
  - word-spacing: normal | wielkość
  - letter-spacing: normal | wielkość

```
p { color: teal; }
p.warning {text-transform: capitalize }
a { text-decoration: none; }
h1 { letter-spacing: 0.25em }
```

Tło

- Tło:
  - background-color: transparent | kolor | RGB
  - background-image: URL
  - background-position: center | left | right | top | bottom
  - background-repeat: repeat | repeat-x | repeat-y| no-repeat
  - background-attachment: scroll | fixed

```
h1 { background-color: #ddf; }
div { background-image: url("./tlo.png") }
```

- Łączone cechy tła:
  - background: [background-color |
    background-image | background-repeat |
    background-attachment | background-position]

```
div { background: no-repeat url("./tlo.png") }
```

Marginesy

- Marginesy zewnętrzne:
  - margin-left, margin-right,
    margin-top, margin-bottom: wielkość |auto

```
div { margin-left: auto; margin-right: auto; }
```

- Marginesy wewnętrzne:
  - padding-left, padding-right,
    padding-top, padding-bottom: wielkość

```
p { padding-top: 1cm; }
```

- Łączone cechy marginesów:
  - margin: w | w w | w w w w
  - padding: w | w w | w w w w
    (wszystkie | góra-dół lewo-prawo | góra prawo dół lewo)

```
p { margin: 1em 2em; }
```

Obramowanie

- Obramowanie:
  - border-top-style: none | dashed | dotted | solid | double | …
  - border-top-width: thin | medium | thick | wielkość
  - border-top-color: kolor | RGB

  Analogicznie dla border-left-, border-bottom- i border-right

```
h2 { border-top-style: solid;
     border-top-color: #faa; }
```

- Łączone cechy obramowania:
  - border-style, border-width, border-color: wszystkie krawędzie
  - border-top: [style | width | color]
  - border: [style | width | color]

```
h1 { border-color: teal; }
h2 { border-top: 1px solid blue; }
div.warning { border: 2px double red; }
```

Rozmiary

- Szerokość:
  - width: auto | wielkość
  - min-width: wielkość
  - max-width: wielkość

```
p.narrow { width: 50% }
img.thumb { width: 100px; }
```

- Wysokość
  - height: auto | wielkość
  - min-height: wielkość
  - max-heihgt: wielkość

```
td { height: 3em; }
```

- Ma zastosowanie do elementów blokowych (table, td, p, h1 … h6, div, img, …)

Pozycjonowanie

- Elementy pływajace
  - float: left | right | none

```
img.photo { float: left; }
```

- Pozycjonowanie
  - position: static | relative | absolute | fixed
    static – normalnie,
    relative – względem normalnej pozycji,
    absolute – względem elementu zawierającego,
    fixed – względem okna (jak absolute, ale nie jest przewijany)
  - left, right, top, bottom: wielkość

```
div#menu {
  position: absolute;
  top: 100px;
  left: -50px;
}
```

Wyświetlanie

- Sposób wyświetlania
  - display: block | inline | none

```
div.descr { display: none; }
```

- Widzialność
  - visibility: visible | hidden

```
div#warn { visibility: hidden; }
```

- Można wykorzystać w skryptach (ukrywanie elementów lub ich wyświetlanie na życzenie) lub w różnych mediach (pomijanie na wydruku np. menu, bannera itp.);

Listy

- Poszczególne cechy list:
  - list-style-type : disc | circle | square | decimal | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | lower-alpha | upper-alpha | none
  - list-style-image: url
  - list-style-position : inside | outside

```
ul { list-style-type: disc; }
```

- Łączone cechy listy:
  - list-style: [list-style-type | list-style-image | list-style-position]

```
ul.todo { list-style: outside url(“dot.png”) }
```

Tabele

- Poszczególne cechy tabeli:
  - caption-side: top | bottom | left | right
  - table-layout: auto | fixed
  - border-collapse: collapse | separated
  - border-spacing: wielkość

```
table { caption-side: bottom; }
```

- Sposób obramowania można zdefiniować używając
  border (dla tabeli jako całości lub td i th);
  Odstępy można określić używając padding (dla td)
  Wielkość tabeli (jako całości) oraz szerokość kolumn i wysokość rzędów można określić przez width i height (dla td, colgroup, col, tr oraz td albo th)

Wskazówki i przykłady

- CSS powinno się projektować hierarchicznie, zaczynając od elementów ogólniejszych, kończąc na szczegółowych :

```
body { font-family: Arial, sans-serif; }
p { color: navy; }
p.mid { text-align: center; }
```

- CSS warto zacząć od elementów *body* oraz *body **
  - margines *body* wyznaczy odstęp całej zawartości od krawędzi okna przeglądarki

```
body {
  width: 80%;
  margin-left: auto;
  margin-right: auto;
}
```

  - formatowanie tekstu (czcionka, kolor, wyrównanie, …) będzie odziedziczone jako domyślne do wszystkich elementów tekstowych (p, h1..h6, td, li, …)

```
body * {
  font-family: Arial, Tahoma, sans-serif;
  color: black;
}
```

- Formatowanie rożnych elementów można dowolnie grupować, a każdy selektor można wymieniać wielokrotnie:

```
h1, h2 {
  text-align: center;
}
h1, h2, h3 {
  color: #yyyyyy;
}
h3 {
  text-align: left;
  color: #xxxxxx;
}
```

  Jedynym ograniczeniem jest czytelność
  (zobacz kolor h3 wyżej; nie jest to błąd – brany jest kolor podany jako ostatni)

- Elementy mające kilka wariantów formatowania
  (np. <p> - akapit zwykły, podpis rysunku, cytat, …)
  - Należy wybrać wariant najczęściej występujący i zapisać format bez użycia klasy:

```
p {
  font-family: Arial, Tahoma, sans-serif;
  font-style: normal;
  text-align: left;
}
```

- Następnie należy zdefiniować klasy, podając tylko elementy różniące ten styl od „bezklasowego”

```
p.rys {
  font-style: italic;
  text-align: center;
}
```

- Aby ograniczyć liczbę klas i ułatwić zmiany struktury strony, można stosować relację zawierania selektorów, np.

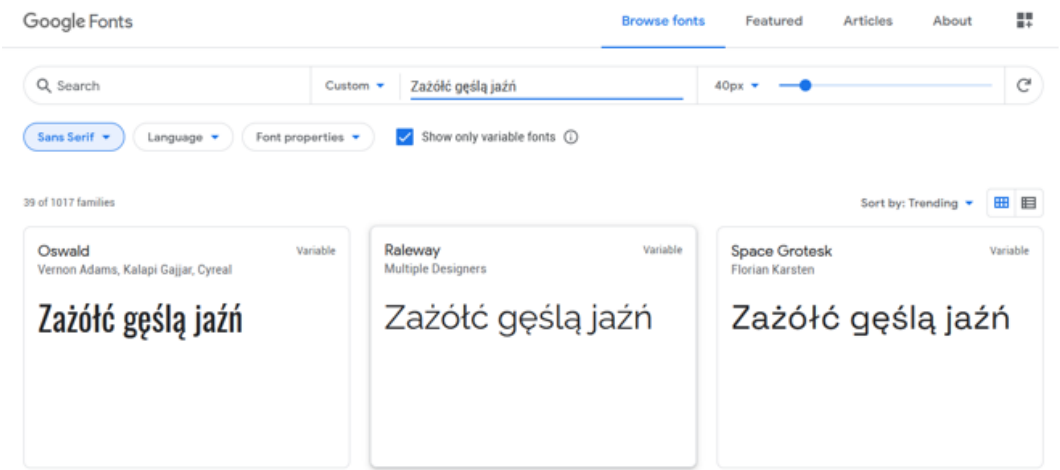
```
a { color: red; }
td.menu a { color: green; }
td.foot a { color: blue; }
```

```
<td class="menu"><a href="...">"a" w menu</a>

<td>
  <a href="...">"a" podstawowe</a>

<td class="foot"><a href="...">"a" w stopce</a>
```

- Łącze w menu i w stopce strony będzie sformatowane inaczej, a klasę wystarczy przypisać komórce tabeli (zamiast każdemu łączu)

Czcionki

- Czcionki najlepiej pobrać z Google Fonts: https://fonts.google.com/
  Warto zapamiętać zdanie "Zażółć gęślą jaźń"



Kolory

- Powinny tworzyć harmonijną paletę zamiast kakofonii; można użyć gotowej palety lub utworzyć własną. Kolory z palety należy używać w różnych intensywnościach (można użyć programu do obróbki grafiki, np. Gimp)
- Obecnie dominuje pogląd, że strona powinna przyciągać użyteczną zawartością, a nie efekciarstwem.
- Dostępne są narzędzia precyzyjne dostosowanie palety, z różnymi wariantami kolorów głównych, predefiniowanymi ustawieniami kontrastu, intensywności, jasności itp., np.:
  http://kuler.adobe.com/#themes/
  https://paletton.com/#uid=33F0k0kopqJfUBLkmtYrelBvefX

Łącza

- Jeżeli zmienia się w CSS kolor tekstu, to łącze zawarte w tym tekście nie dziedziczy koloru – jeżeli nie ma być niebiesko-fioletowe, to trzeba skorzystać z pseudoklas łączy
- Jeżeli strona w zasadniczej treści zawiera liczne łącza, zwłaszcza jeżeli występują pętle podstron, to kolory łączy nieodwiedzonych i odwiedzonych powinny być różne; Typowy przykład – Wikipedia.
- Jeżeli jedyne łącza stanowi menu z kilkoma pozycjami, to ww. kolory mogą być jednakowe;
- Użytkownicy już przywykli, że łącze może nie mieć podkreślenia, ale zaleca się je nadal stosować. Jeżeli nie, to łącze powinno mieć kolor inny, niż otaczające tekst i zmieniać go "pod myszką"
- Łącza tworzące menu można zmienić w elementy blokowe;
  w połączeniu ze zmienionym kolorem tła utworzą coś na kształt przycisków (jeden pod drugim, na całą dostępną szerokość – zatem muszą być w czymś, np. w komórce tabeli)

```
a {
  display: block;
  padding: 5px 5px;
}
```

- Margines wewnętrzny (padding) domyślnie jest minimalny; Jeżeli "przycisk" ma inny kolor tła, to bez zmiany marginesu źle wygląda; Zazwyczaj wykorzystuje się pseudoklasy łączy (a:link, a:visited, a:hover) aby wygląd przycisku zmieniał się po wskazaniu go myszką:

- Łącza zmienione w elementy blokowe, jeżeli mają być wyświetlane obok siebie, trzeba też zmienić w pływające:

```
a {
  display: block;
  float: left;
  padding: 5px 5px;
}
```

albo inline-block:

```
a {
  display: inline-block;
  padding: 5px 5px;
}
```

Marginesy

- Zalecana jest zasada *wszystko albo (prawie) nic* – albo ustala się wielkość wszystkich marginesów (wewn. i zewn.), albo zostawia domyślne i tylko sporadycznie zmienia gdzie trzeba (np. łącza w menu, jak opisano na poprzednich slajdach)
- Elementy blokowe sąsiadujące w pionie (np. <h1> <h2> <p> <p> <h3> <p> <p>…) powinny mieć taki sam odstęp od lewej krawędzi (padding-left + margin-left)
  - tekst ze schodkami źle wygląda
- Trzeba pamiętać, że marginesy elementów umieszczanych po sobie (np. <h1> <h2><p>) nie sumują się, ale umieszczane jedne w drugich (np. <table> <tr> <td> <p>) – owszem;
- Marginesy elementów użytych tylko celem uzyskania układu strony (np. tabela z czterema komórkami: banner, menu, treść, stopka) powinny mieć marginesy zewnętrzne i wewnętrzne równe 0

```
table.layout td {
  margin: 0;
     padding: 0;
}
```

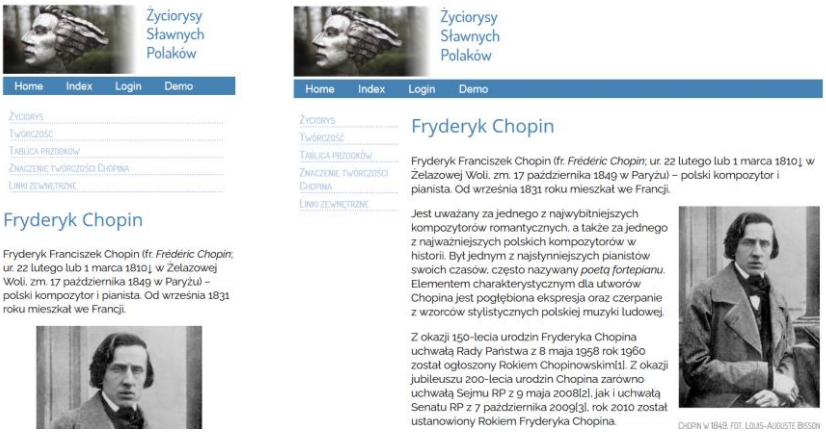
- Do sprawdzenia jak sumują się marginesy, warto włączyć (tymczasowo) cienką ramkę wokół wszystkiego, ewentualnie z innym kolorem dla wybranego rodzaju elementów

```
body * {
  border 1px dotted gray;
}

p {
  border 1px dotted red;
}
```

Zadania

Proszę opracować arkusz CSS do dokumentu HTML z poprzedniego ćwiczenia.
Do uzyskania różnego wyglądu tych określonych elementów HTML, np. odnośników, w różnych miejscach dokumentu, należy wykorzystać relację zawierania selektorów CSS (np. `nav a {…}` oraz `aside a {…}`) oraz klasy CSS (np. `p {…}` oraz `p.picture {…}`)

1. Tekst powinien być wyświetlany z użyciem dwóch lub trzech różnych czcionek, pobieranych z serwisu Google Fonts (np. nagłówki jedna czcionka, akapity druga, menu trzecia), z różnym formatowaniem (np. treść, podpisy rysunków i cytaty mogą mieć tę samą czcionkę, ale w różnych wariantach – kursywa, pogrubienie, kolor, kapitaliki itp.)
2. Do wyświetlania tekstu i elementów menu należy wybrać nie więcej niż 3-4 kolory; Kolory powinny pasować do siebie (pochodzić z jednej palety). Do wyświetlania tekstu głównego powinien być użyty jeden kolor, ewentualnie dwa warianty koloru, różniące się jasnością lub intensywnością (np. jeden – akapity, drugi – cytaty, podpisy rysunków). Kolejne kolory powinny użyte w menu (tło i tekst), dodatkowe akcenty kolorystyczne mogą też być użyte w nagłówkach, odnośnikach w tekście itp.
3. Nagłówki różnych poziomów oraz różne kategorie tekstu (np. zwykłe akapity, podpisy rysunków, cytaty, …) powinny różnić się od siebie – zauważalnie, choć nie przesadnie
4. Odnośniki w menu (element `nav`) powinny mieć formę przycisków wyświetlanych w poziomie, z wyraźną zmianą wyglądu po wskazaniu odnośnika myszką; Odnośniki odwiedzone i nieodwiedzone powinny mieć jednakowy wygląd
5. Odnośniki w spisie treści (element `aside`) powinny być wyświetlane w pionie (czyli jeden pod drugim), w formie przycisków albo tekstu, ze zmianą wyglądu po wskazaniu myszką; Odnośniki odwiedzone i nieodwiedzone powinny mieć jednakowy wygląd
6. Strona powinna być responsywna: układ strony powinien być pionowy dla mniejszych szerokości okna przeglądarki (np. ≤ 900px) oraz mieszany dla szerokości większych; Obrazki powinny mieć szerokość określoną w procentach dla mniejszych szerokości okna (np. ≤ 600px) oraz stałą szerokość dla szerokości okna większych. Punt przełączenia dla obrazków może być ten sam, co dla układu strony, albo inny.



7. Tabela powinna mieć wygląd (tj. formatowanie tekstu, odstępy między komórkami, kolor i sposób obramowania itp.) pasujący do reszty strony; Tabela powinna być responsywna;
8. Strona powinna być drukowana w układzie pionowym, ale bez nagłówka, menu oraz spisu treści (tj. bez `header`, `nav` i `aside`), co powinno być widoczne w podglądzie wydruku