Kaskadowe arkusze stylów CSS

Warstwowa struktura dokumentu

- Warstwa treści
  - zawsze obecna w dokumencie
  - oprócz tekstu może zawierać też inne obiekty
- Warstwa prezentacji
  - definiuje „wizualizację” dokumentu
  - dokument może mieć *różne* wizualizacje
- Warstwa zachowania
  - umożliwia interakcję z użytkownikiem
  - możliwe różne poziomy, od prostej walidacji danych, aż po rozbudowane elementy interfejsu użytkownika

Warstwowa struktura dokumentu

- Podstawowe narzędzia
  - warstwa treści
    - język (X)HTML(5)
  - warstwa prezentacji
    - język **CSS**
  - warstwa zachowania
    - język JavaScript
- Poszczególne warstwy warto od siebie **odseparować**

Kaskadowe arkusze stylów

- Style **CSS** pozwalają modyfikować
  - *pierwszy plan oraz tło elementów*
    - kolory, elementy graficzne i ich rozmieszczenie, …
  - *właściwości tekstu*
    - krój, styl i inne parametry czcionki, …
  - *właściwości „pudełek” elementów*
    - wymiary, wyścielanie, marginesy, ramki, …
  - *wzajemne położenie elementów w dokumencie*
    - el. pływające, pozycjonowanie
  - *układ oraz niektóre aspekty „interfejsu” strony*
  - …

Kaskadowe arkusze stylów

- Nieco historii
  - CSS 1.0 – pierwsza specyfikacja (1996); większość właściwości dobrze obsługiwana przez współczesne przeglądarki
  - CSS 2.0 – specyfikacja powstała w 1998 r.; żadna współczesna przeglądarka nie implementuje jej w pełni
  - **CSS 2.1** – uproszczona wersja specyfikacji CSS 2.0; aktualny standard; wciąż drobne problemy w obsłudze przez przeglądarki; często określana jako **CSS2**
  - CSS 3.0 – wersja „rozwojowa”; jeszcze nie w pełni implementowana przez współczesne przeglądarki

`http://webdesign.about.com/od/css3/a/differences-css2-css3.htm`

Anatomia stylu CSS

- **Styl** – ciąg reguł definiujących wygląd dokumentu
- Podstawowa postać reguły

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

- **Elementy reguły**
  - `selektor` określa do których elementów dokumentu „stosuje się” reguła
  - `właściwość` oznacza cechę składu dokumentu
  - `wartość` jest to wielkość związana z daną cechą
- Dalej `<deklarcja>` oznacza „`właściwość: wartość`”

Anatomia stylu CSS

- **Deklaracje**
  - wielkość liter użytych w nazwach właściwości CSS nie jest istotna; to samo dotyczy wartości, które mają postać identyfikatorów CSS, jak np. `green` w deklaracji
    - `Color : GrEen` ≡ `color : green`
  - wszelkie odstępy pomiędzy nazwą właściwości, znakiem dwukropka oraz wartością są pomijane
    - `color : green` ≡ `color:green`
- **Komentarze**
  - otwarcie: `/*`
  - zamknięcie: `*/`

Anatomia stylu CSS

- **@-reguły**
  - `@charset` definiuje kodowanie znaków użyte w pliku ze stylem; musi wystąpić na jego początku
  - `@import` pozwala na dołączenie do stylu innego stylu; wszystkie reguły `@import` muszą wystąpić na początku stylu – przed jakimikolwiek „zwykłymi” regułami
  - `@media` pozwala na stworzenie różnych stylów dla różnych typów mediów np. stylu do wyświetlania dokumentu na ekranie i oddzielnego – do jego drukowania
  - CSS2 definiuje jeszcze regułę `@page`, ale nie jest ona obsługiwana przez żadną przeglądarkę poza Operą

Anatomia stylu CSS

- **@-reguły** – przykład

```
@charset "ISO-8859-2" /* domyślnie UTF-8 */
@import url("/css/main.css");
@import "local.css";
@import url(/css/screen.css) screen, projection;
/* styl do wyświetlania */
@media screen, projection {
  html { background: #fffef0; color: #300; }
}
/* styl do druku */
@media print {
  html { background: #fff; color: #000; }
}
```


Anatomia stylu CSS

- **@-reguły** – uwagi
  - `@charset` na ogół nie jest potrzebny – przeglądarka ma kilka innych metod, aby ustalić kodowanie pliku ze stylem CSS
  - `@import` – przeglądarki IE 5.5-7 nie implementują specyfikacji typu medium dla tej reguły

Umieszczanie stylów w dokumencie

- **Styl zewnętrzny** – element nagłówka dokumentu

```
<link rel="stylesheet" href="<url>">
```

`<url>` – adres URL pliku zawierającego styl

dla wersji < HTML5 konieczny atrybut: `type="text/css"`

- **Styl wewnętrzny** – element nagłówka dokumentu

```
<style> <reguły> </style>
```

`<reguły>` – treść stylu (ciąg reguł)

dla wersji < HTML5 konieczny atrybut: `type="text/css"`

- **Styl zintegrowany** – wartość atrybutu

```
style="<deklaracje>"
```

`<deklaracje>` – ciąg deklaracji CSS oddzielonych średnikami

Umieszczanie stylów w dokumencie

- **Uwagi**
  - do stylów zewnętrznych lepiej stosować `<link>` niż `@import`
  - `@import` musi poprzedzać ewentualne reguły

```
<style>
@import url(css/styl.css);
@import url(css/form.css);
p { color:red; }
</style>
```

✔

```
<style>
p { color:red; }
@import url(css/styl.css);
@import url(css/form.css);
</style>
```

✘

Selektory CSS

Identyfikują elementy, którym chcemy nadać styl

Selektory CSS

- Podstawowe rodzaje selektorów
  - uniwersalny, znacznika, identyfikatora, atrybutu, pseudo-klasy i pseudo-elementu
- **Selektor prosty** *to selektor uniwersalny, bądź selektor znacznika, po którym bezpośrednio następuje zero lub więcej selektorów: identyfikatora, atrybutu oraz pseudo-klasy (w dowolnej kolejności)*, np.

```
h1#tytul[lang|="pl"]:hover
```

- Selektor prosty pasuje do danego elementu, jeśli pasują do niego wszystkie jego składowe

Selektory CSS

- Selektor **uniwersalny**
  - `* { <deklarcja>; }`
  - oznacza, że reguła odnosi się do każdego elementu
  - stosowany głównie w selektorach „kontekstowych”
- Przykład
  - `* { margin : 0; }`

Selektory CSS

- Selektor **znacznika**
  - `znacznik { <deklarcja>; }`
  - oznacza, że reguła stosuje się do wszystkich elementów `znacznik`
- Przykłady
  - `h1 { color : green; }`
  - `p { text-align : center; }`
  - `body { margin-left : 30px; }`

Selektory CSS

- Selektor **identyfikatora**
  - `#nazwa { <deklarcja>; }`
  - reguła stosuje się do elementu, któremu w dokumencie przypisaliśmy (za pomocą atrybutu id) unikatowy identyfikator `nazwa`
- Przykłady
  - `#adres { font-family : sans-serif; }`
  - `p#adres { border-style : solid; }`

Technologie internetowe – język CSS

Selektory CSS

- Selektor **atrybutu**
  - `[atrybut] { <deklarcja>; }`
  - `[atrybut<op>wartość] { <deklarcja>; }`
  - operator `<op>` może mieć 3 formy:
    - `=` atrybut ma podaną wartość
    - `|=` atrybut ma podaną `wartość` lub zawiera listę ciągów znaków rozdzielonych myślnikami rozpoczynającą się od łańcucha `wartość`
    - `~=` wartość atrybutu zawiera listę ciągów znaków rozdzielonych odstępami, a wśród tych ciągów występuje łańcuch `wartość`
    - **CSS3:** `^=`, `*=`, `$=`

Technologie internetowe – język CSS

Selektory CSS

- Selektor **atrybutu** c.d.
  - reguła opatrzona tym selektorem stosuje się do elementu, który w dokumencie ma zdefiniowany `atrybut`, i dodatkowo, w przypadku obecności operatora `<op>` `wartość` atrybutu spełnia odpowiedni warunek
- Przykłady
  - `p[title] { font-style : italic; }`
  - `img[src="1.jpg"] { border-style : none; }`
  - `p[lang="en"] { color : #EEEEEE; }`

Technologie internetowe – język CSS

Selektory CSS

- Selektor **klasy**
  - `.klasa { <deklarcja>; }`
  - stosuje się do elementów, którym w dokumencie przypisano (z pomocą atrybutu class) klasę `klasa`
  - `.klasa` to skrót dla selektora `[class|="klasa"]`
- Przykłady
  - `.wazne { color : red; }`
  - `h1.wazne.bardzo { font-weight : bold; }`

Selektory CSS

- Selektor **pseudo-klasy**
  - `:nazwa { <deklarcja>; }`
  - reguła stosuje się do elementów w pewnych specyficznych dla danego selektora pseudo-klasy sytuacjach
- pseudo-klasa `:link`
  - odnosi się do nieodwiedzonych jeszcze odsyłaczy
  - `:link { color : blue; }`

Technologie internetowe – język CSS

Selektory CSS

- pseudo-klasa `:visited`
  - odnosi się do odsyłaczy, które zostały już przynajmniej raz odwiedzone
  - `:visited { color : gray; }`
- pseudo-klasa `:hover`
  - odnosi się do elementów, nad którymi właśnie znajduje się kursor myszy
  - `p:hover { background-color : yellow; }`

Technologie internetowe – język CSS

Selektory CSS

- pseudo-klasa `:active`
  - odnosi się do elementów, które zostały „właśnie uaktywnione”, np. odsyłacz w momencie naciśnięcia na nim klawisza myszy
  - `:active { font-size : larger; }`
- pseudo-klasa `:focus`
  - odnosi się do elementów, które są gotowe do czytania znaków z klawiatury; może to dotyczyć zarówno elementów formularzy, jak i odsyłaczy (jeśli użytkownik nawiguje za pomocą klawiatury)
  - `:focus { border : 2px solid; }`

Technologie internetowe – język CSS

Selektory CSS

- pseudo-klasa `:lang`
  - odnosi się do elementów, które są używają danego języka (również wtedy, gdy jest on „odziedziczony” – to podstawowa różnica w stosunku do `[lang|=...]`)
  - `:lang(en) { color : gray; }`
- pseudo-klasa `:first-child`
  - odnosi się do elementu, jeśli jest on „pierwszym potomkiem” swojego elementu nadrzędnego (w sensie hierarchii drzewa dokumentu)
  - `ul:first-child { font-style : italic; }`

Selektory – obsługa w przeglądarkach*

http://www.quirksmode.org/css/selectors/

* główny problem – „co nie tak w **IE**”…

Reguły złożone – grupowanie selektorów

- Zbiór reguł z identyczną deklaracją

```
sel1 { <deklaracja>; }
sel2 { <deklaracja>; }
       ...
seln { <deklaracja>; }
```

można zastąpić pojedynczą regułą „złożoną”:

```
sel1,sel2,...,seln { <deklaracja>; }
```


Reguły złożone – grupowanie deklaracji

- Zbiór reguł z identycznymi selektorami

```
sel { <deklaracja1>; }
sel { <deklaracja2>; }
      ...
sel { <deklaracjan>; }
```

można zastąpić pojedynczą regułą „złożoną”:

```
sel { <deklaracja1>; ... <deklaracjan>; }
```


Selektory strukturalne

- Selektor **bezpośredniego potomka** (**dziecka**)
  - `sel1 > sel2 { <deklarcje> }`
  - stosuje się do elementów, które pasują do selektora `sel2`, oraz są bezpośrednimi potomkami (w sensie hierarchii drzewa dokumentu) elementów pasujących do selektora `sel1`
- Uwagi o obsłudze w przeglądarkach
  - brak obsługi w IE < 7
  - IE 7 błędnie obsługuje pewne specyficzne sytuacje (np. nie można wstawić komentarza pomiędzy „>” i następujący po nim selektor)

Selektory strukturalne

- Selektor **potomka**
  - `sel1 sel2 { <deklarcje> }`
  - stosuje się do elementów, które pasują do selektora `sel2`, oraz są potomkami (w sensie hierarchii drzewa dokumentu) elementów pasujących do selektora `sel1`
- Uwagi o obsłudze w przeglądarkach
  - w IE < 7 `sel1:hover sel2` nie działa
  - IE 6, 7 błędnie obsługuje pewne specyficzne sytuacje związane z użyciem komentarzy

Selektor dziecka/potomka – różnica

```
ul>li { border : 1px solid; }
```

- ul - potomek 1
  1. ol - potomek 1
  2. ol - potomek 2
- ul - potomek 2

```
<ul>
  <li> ul - potomek 1
      <ol>
        <li>ol - potomek 1</li>
        <li>ol - potomek 2</li>
      </ol>
  </li>
  <li>ul - potomek 2</li>
</ul>
```

```
ul li { border : 1px solid; }
```

- ul - potomek 1
  1. ol - potomek 1
  2. ol - potomek 2
- ul - potomek 2

Selektor dziecka/potomka – różnica c.d.

```
ul>li { color  : red;
        border : 1px solid; }
```

- ul - potomek 1
  1. ol - potomek 1
  2. ol - potomek 2
- ul - potomek 2

```
<ul>
  <li> ul - potomek 1
      <ol>
        <li>ol - potomek 1</li>
        <li>ol - potomek 2</li>
      </ol>
  </li>
  <li>ul - potomek 2</li>
</ul>
```

```
ul li { color  : red;
        border : 1px solid; }
```

- ul - potomek 1
  1. ol - potomek 1
  2. ol - potomek 2
- ul - potomek 2

Dziedziczność właściwości CSS

- Pytanie
  - dlaczego w przypadku właściwości `color` nie widać różnicy pomiędzy selektorami dziecka i potomka?
- Odpowiedź
  - właściwość `color` jest dziedziczna, podczas gdy właściwość `border` dziedziczna nie jest
  - dziedziczność właściwości oznacza, że jeśli jakiś element ją posiada, to posiadają ja również elementy znajdujące się „pod nim” w drzewie dokumentu
  - dziedziczność jest cechą specyficzną każdej z właściwości CSS – na ogół ma ona wartość podyktowaną zdrowym rozsądkiem

Selektory strukturalne

- Selektor **przyległego rodzeństwa**
  - `sel1 + sel2 { <deklarcje> }`
  - stosuje się do elementów, które pasują do selektora `sel2`, oraz są bezpośrednim rodzeństwem elementu pasującego do selektora `sel1`; w szczególności oznacza to, że oba elementy występują „obok siebie”, na tym samym poziomie w hierarchii dokumentu
- Uwagi o obsłudze w przeglądarkach
  - brak obsługi w IE < 7
  - IE 7 błędnie obsługuje pewne specyficzne sytuacje związane z użyciem komentarzy

Selektory strukturalne

- Selektor **przyległego rodzeństwa** c.d.
  - `h1+p { font-style : italic; }`

```
<body>
<h1>Rozdział 1</h1>
<div>
  Lorem ipsum dolor sit amet tempus leo in
  mattis lectus blandit risus at tortor.
</div>
<p>
  Lorem ipsum dolor sit amet tempus leo in
  mattis lectus blandit risus at tortor.
</p>
<h1>Rozdział 2</h1>
<p>
  Lorem ipsum dolor sit amet tempus leo in
  mattis lectus blandit risus at tortor.
</p>
<p>
  Lorem ipsum dolor sit amet tempus leo in
  mattis lectus blandit risus at tortor.
</p>
</body>
```

Rozdział 1

Lorem ipsum dolor sit amet tempus leo in mattis lectus blandit risus at tortor.

Lorem ipsum dolor sit amet tempus leo in mattis lectus blandit risus at tortor.

Rozdział 2

Lorem ipsum dolor sit amet tempus leo in mattis lectus blandit risus at tortor.

Lorem ipsum dolor sit amet tempus leo in mattis lectus blandit risus at tortor.

  - **CSS3:** `h1~ul { font-style : italic; }`

„Konflikty” pomiędzy regułami

- Rozważmy następujący styl
  - `.wazne { color : red; }`
    `h1 { color : blue; }`
- Pytanie: jak wyświetlony zostanie element?
  - `<h1 class="wazne">Tytuł rozdziału</h1>`
- Odpowiedź
  - **Tytuł rozdziału**
- Dlaczego?
  - bo reguła pierwsza jest „bardziej specyficzna” niż druga

Kaskada stylów

- Style CSS są „kaskadowe” ponieważ na ostateczny wygląd elementu w dokumencie mają wpływ style, które mogą „napływać” z różnych źródeł
- **Kaskada** bierze pod uwagę cztery własności reguł
  - „ważność” oraz źródło pochodzenia
  - „specyficzność”
  - kolejność wystąpienia w dokumencie
- Jeśli styl nie zawiera żadnych reguł odnoszących się do danego elementu na jego wygląd mają wpływ właściwości dziedziczone z elementów nadrzędnych

Kaskada stylów

- **Etap 1**
  - wyszukiwane są wszystkie deklaracje odnoszące się do danej właściwości danego elementu; mogą pochodzić z trzech źródeł:
    - stylu przeglądarki (**SP**)
    - stylu autora dokumentu (**SA**)
    - stylu użytkownika (**SU**)
  - jeśli znaleziono więcej niż jedną deklarację, odnoszącą się do danej właściwości elementu, kaskada przechodzi do Etapu 2 – w przeciwnym wypadku deklaracja jest stosowana do elementu

Kaskada stylów

- **Etap 2**
  - deklaracje porządkowane są względem „ważności” oraz źródła pochodzenia
  - deklaracja może być albo „ważna” (kiedy reguła ją zawierająca używa słowa kluczowego `!important`), albo „zwykła”
  - porządek ustalany jest następująco (priorytet rosnący)
    - deklaracje (**SP**)
    - zwykłe deklaracje (**SU**) < (**SA**)
    - ważne deklaracje (**SA**) < (**SU**)
  - jeśli znaleziono więcej niż jedną deklarację o takim samym priorytecie, kaskada przechodzi do Etapu 3

Kaskada stylów

- **Etap 3**
  - deklaracje porządkowane są według „specyficzności”
  - specyficzność reprezentowana jest jako ciąg czterech liczb: `a`,`b`,`c`,`d`; jeśli deklaracja pochodzi ze stylu zintegrowanego to jej specyficzność wynosi `1,0,0,0`
  - niech `sel` oznacza selektor reguły z której pochodzi deklaracja, wówczas `a=0` oraz
    - `b` = „liczba selektorów identyfikatora w `sel`”
    - `c` = „liczba selektorów atrybutu (w tym klasy) oraz pseudo-klasy w `sel`”
    - `d` = „liczba elementów oraz pseudo-elementów w `sel`”

Kaskada stylów

- **Etap 3** – przykład „Tytułu rozdziału”
  - styl

```
.wazne { color : red; }
h1 { color : blue; }
```

  - specyficzność deklaracji
    - `spec(color:red)  = 0,0,1,0`
    - `spec(color:blue) = 0,0,0,1`
  - stosując porządek leksykograficzny otrzymujemy, że `spec(color:red) > spec(color:blue)`, skąd

```
<h1 class="wazne">Tytuł rozdziału</h1>
```

daje: `Tytuł rozdziału`

Kaskada stylów

- **Etap 3**
  - jeśli znaleziono więcej niż jedną deklarację o takiej samej specyficzności, kaskada przechodzi do Etapu 4
- **Etap 4**
  - na tym etapie brany jest pod uwagę tekstowy porządek występowania reguł w dokumencie; deklaracja występująca w obrębie „ostatniej” reguły jest stosowana do elementu.
  - style zewnętrzne dołączane do dokumentu traktowane są tak, jak gdyby znajdowały się w treści dokumentu (w miejscu dołączenia)

Technologie internetowe – język CSS

Elementy i ich rodzaje

- **Elementy** – podstawa struktury dokumentów
  - **p**, **div**, **head**, **body**, …
- Elementy podmieniane i niepodmieniane
  - *podmieniane*: ich zawartość stanowi coś, co nie ma bezpośredniej reprezentacji w dokumencie
    - **img**
    - **object**
    - elementy formularzy (**input**, **textarea**, **select**, …)
    - …
  - *niepodmieniane*: zdecydowana większość pozostałych elementów HTML-a

Technologie internetowe – język CSS

Elementy blokowe i wewnętrzne

- Elementy **blokowe** w HTML-u
  - na ogół (choć nie zawsze) zawierają inne elementy (przykładem elementu blokowego, który nie zawiera niczego jest **<hr>**)
  - niektóre elementy blokowe mogą zawierać jedynie inne elementy blokowe (np. element **<body>**...**</body>**)
  - pewne elementy blokowe mogą zawierać jedynie dane tekstowe (tekst) lub *elementy wewnętrzne* (np. element ** **...** ** w HTML 4.01 Strict)
  - jeszcze inne mogą zawierać zarówno elementy blokowe jak i elementy wewnętrzne lub dane tekstowe (jak np. ** **...** ** czy **
...
**)

Technologie internetowe – język CSS

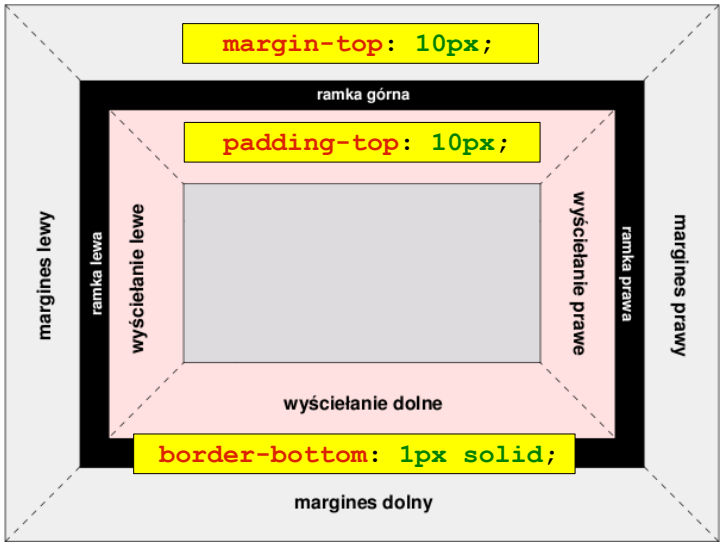
Elementy blokowe i wewnętrzne

- Elementy **wewnętrzne** w HTML-u
  - na ogół zawierają dane tekstowe (tekst) oraz inne elementy wewnętrzne (przykładem elementu wewnętrznego bez zawartości jest **
**)
  - często oznakowują „semantycznie” fragmenty tekstu (np. ****...**** czy **<q>**...**</q>**)
- Elementy blokowe/wewnętrzne a DTD
  - podział zdefiniowany w DTD dla konkretnej wersji (X)HTML-a
  - DTD określa też zasady dotyczące dopuszczalnej zawartości elementów

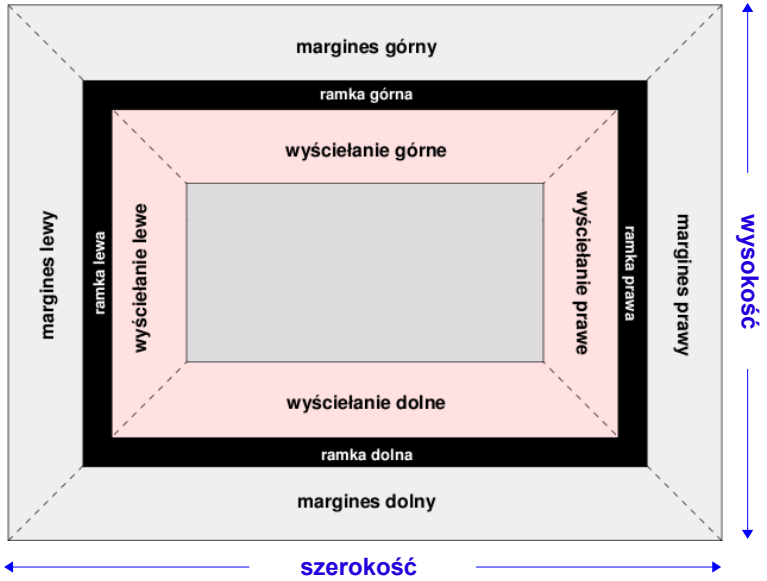
Elementy blokowe i wewnętrzne w CSS

- Elementy **blokowe**/**wewnętrzne** w CSS
  - zdefiniowane niezależnie od analogicznego pojęcia w języku (X)HTML
  - tak naprawdę CSS wykorzystuje jeszcze inne rodzaje elementów, ale wszystkie one mogą być uważane za „podtypy" elementów blokowych lub wewnętrznych
  - zarówno elementy blokowe jak i wewnętrzne CSS mają postać prostokątnych „pudełek"
  - właściwości CSS pozwalają na elastyczne wpływanie na różne cechy pudełek elementów
  - właściwość `display` pozwala ustalać typ elementu; nie przenosi się to na poziom języka (X)HTML (!!!)

„Pudełko" elementu (blokowego) CSS

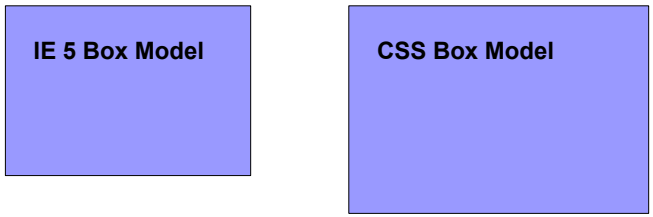


„Pudełko" elementu (blokowego) IE 5



„Pudełko" elementu (blokowego) IE 5

- Konsekwencja



oba pudełka mają identyczny styl CSS (sic!)

- Czy jest sens mówić o elementach blokowych **IE 5** ?
  - niestety IE 6, 7, 8 wciąż używają modelu elementów blokowych **IE 5** jeśli pracują w trybie odmienności

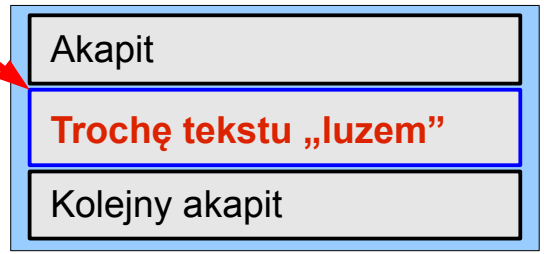
Formatowanie blokowe

- Zadanie i właściwości:
  - układa w pionie pudełka elementów blokowych
  - układanie rozpoczyna się od góry strony (okna)
  - jednorodność zawartości
    - element blokowy tworzy tzw. „blok główny” pudełka, którego „potomkami”, mogą być **albo** pudełka elementów blokowych, **albo** pudełka elementów wewnętrznych
  - elementy anonimowe
    - jeśli element „zawiera mieszaninę” elementów, to występujące w niej elementy wewnętrzne otaczane są *automatycznie* anonimowymi elementami blokowymi

Anonimowe elementy blokowe

```
<div>
<p>Akapit</p>
Trochę teksu „luzem”
<p>Kolejny akapit</p>
</div>
```

„Anonimowy” element blokowy



Formatowanie blokowe

- Elementy anonimowe – dziedziczenie własności
  - dziedziczenie od otaczającego elementu blokowego
  - właściwości niedziedziczne przyjmują wartość domyślną
- Odstęp pionowy pomiędzy sąsiadującymi pudełkami
  - określany na podstawie wartości ich marginesów pionowych (górnego i dolnego)
  - zależy od obecności wyścielania i ramek

Formatowanie blokowe

- Zapadanie się marginesów
  - jeśli pudełka nie mają (pionowego) wyścielania oraz ramek, to marginesy pionowe pomiędzy sąsiadującymi pudełkami **nie są brane pod uwagę**
  - jeśli sąsiadujące w pionie pudełka mają marginesy, to **jedynie większy** z nich **brany** jest **pod uwagę**;
  - jeśli któryś z marginesów ma wartość ujemną, to ich wartości **sumują się** (!)

„Zapadanie się" marginesów

HTML: sąsiadujące elementy blokowe

```
<h1>Tytuł rozdziału</h1>
<p>
  Maginesy pionowe
  „zapadają się"
</p>
<p>
  jak widać...
</p>
```

CSS:

```
h1 { margin-bottom: 40px; }
p  { margin-top: 20px;
     margin-bottom: 20px; }
```

Przeglądarka:

„Zapadanie się" marginesów

HTML: zagnieżdżone elementy blokowe

```
<div class="box a">
  <div class="box b">
    <div class="box c">
      <div class="box d">
        <div class="box e">
        Maginesy pionowe
        „zapadają się", ale
        marginesy poziome nie.
        </div>
      </div>
    </div>
  </div>
</div>
```

CSS:

```
.box { margin: 10px;       }
.a   { background: #777;   }
.b   { background: #999;   }
.c   { background: #bbb;   }
.d   { background: #ddd;   }
.e   { background: yellow;}
```

W przypadku **zagnieżdżonych** elementów zjawisko zachodzi, jeśli nie mają one zdefiniowanego wyścielania ani ramek

Przeglądarka:

Maginesy pionowe „zapadają się", ale marginesy poziome nie.

„Zapadanie się" marginesów

HTML: zagnieżdżone elementy blokowe

```
<div class="box a">
  <div class="box b">
    <div class="box c">
      <div class="box d">
        <div class="box e">
        Maginesy pionowe
        „zapadają się", ale
        marginesy poziome nie.
        </div>
      </div>
    </div>
  </div>
</div>
```

CSS:

```
.box { margin: 10px;
       padding-top: 1px;  }
.a   { background: #777;  }
.b   { background: #999;  }
.c   { background: #bbb;  }
.d   { background: #ddd;  }
.e   { background: yellow;}
```

Przeglądarka:

Maginesy pionowe „zapadają się", ale marginesy poziome nie.

„Zapadanie się" marginesów

HTML: zagnieżdżone elementy blokowe

```
<div class="box a">
  <div class="box b">
    <div class="box c">
      <div class="box d">
        <div class="box e">
        Maginesy pionowe
        „zapadają się", ale
        marginesy poziome nie.
        </div>
      </div>
    </div>
  </div>
</div>
```

CSS:

```
.box { margin: 10px;
       padding: 1px;       }
.a   { background: #777;   }
.b   { background: #999;   }
.c   { background: #bbb;   }
.d   { background: #ddd;   }
.e   { background: yellow;}
```

Przeglądarka:

Maginesy pionowe „zapadają się", ale marginesy poziome nie.

„Zapadanie się" marginesów

HTML: zagnieżdżone elementy blokowe

```
<div class="box a">
  <div class="box b">
    <div class="box c">
      <div class="box d">
        <div class="box e">
        Maginesy pionowe
        „zapadają się", ale
        marginesy poziome nie.
        </div>
      </div>
    </div>
  </div>
</div>
```

CSS:

```
.box { margin: 10px;       }
.a   { background: #777;   }
.b   { background: #999;
       padding: 1px;  }
.c   { background: #bbb;   }
.d   { background: #ddd;   }
.e   { background: yellow;}
```

Przeglądarka:

Maginesy pionowe „zapadają się", ale marginesy poziome nie.

Formatowanie wewnętrzne

- Zajmuje się układaniem w pudełek elementów wewnętrznych (w poziomie), rozpoczynając od lewej, bądź prawej strony – w zależności od ustawień językowych
- Pudełkom elementów wewnętrznych
  - można ustalać wymiary i atrybuty ramek, oraz poziome marginesy i wyścielania – ich odpowiedniki pionowe są ignorowane
  - nie można modyfikować szerokości ani wysokości – nie dotyczy to elementów podmienianych (ang. *replaced elements*): **`img`**, **`object`**, **`button`**, **`textarea`**, **`input`**

Formatowanie wewnętrzne

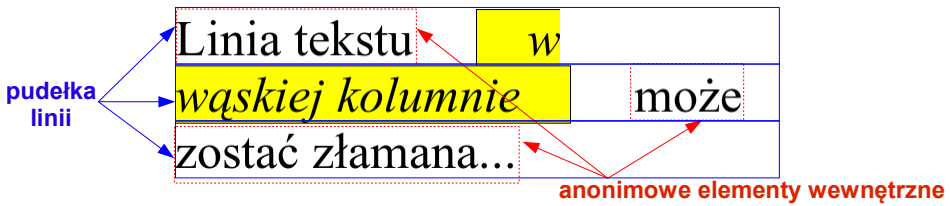
- Pudełka elementów wewnętrznych zawarte w jednej linii tworzą „pudełko linii"

HTML: elementy wewnętrzne w akapicie

```
<p>
  Linia tekstu
  <em>w wąskiej kolumnie</em>
  może zostać złamana...
</p>
```

CSS:

```
em {
  margin: 0 1em;
  padding: 0 1em;
  border: 1px solid black;
  background-color: yellow;
}
```



Formatowanie wewnętrzne

- Pudełka linii składane są w pionie bez dodawania żadnych dodatkowych odstępów; na ich wysokość można wpłynąć za pomocą **`line-height`**
- Jeśli suma długości pudełek elementów wewnętrznych w ramach pudełka linii jest mniejsza niż jego długość, o sposobie rozmieszczenia pudełek decydują wartości parametrów **`direction`** oraz **`text-align`**
- Normalnie lewa i prawa krawędź pudełek linii stykają się z krawędziami bloku „otaczającego". Wyjątkiem jest sytuacja, gdy w układzie występują elementy „pływające" – wówczas pudełka linii stykające się z nimi są odpowiednio skracane

Pozycjonowanie elementów CSS

- Pudełko elementu CSS2 ma 3 współrzędne
  - punkt „zaczepienia” (x,y)
  - indeks głębokości na „stosie pudełek”
- Pozycjonowanie określa gdzie umieszczone zostanie pudełko elementu – ustala zatem jego współrzędne; ma również wpływ na ustalanie wymiarów pudełka
- Metody pozycjonowania – właściwość **`position`**
  - statyczne (**`static`**)
  - względne (**`relative`**)
  - bezwzględne (**`absolute`**)
  - ustalone (**`fixed`**)

Pozycjonowanie elementów CSS

- Pozycjonowanie **statyczne**
  - **`position : static`**
  - domyślny sposób układania pudełek elementów; zgodny ze strukturą dokumentu
  - układanie kolejnych pudełek rozpoczyna się od lewego górnego rogu pudełka nadrzędnego – w przypadku pudełka elementu reprezentującego korzeń drzewa dokumentu, jest to lewy górny róg okna przeglądarki
  - elementom pozycjonowanym statycznie nie można zmienić głębokości na stosie pudełek (**`z-index`**)

Pozycjonowanie elementów CSS

- Pozycjonowanie **względne**
  - **`position : relative`**
  - pudełko elementu jest najpierw układane tak, jak w przypadku pozycjonowania statycznego, a następnie przesuwane w pionie (**`top`**, **`bottom`**) i/lub poziomie (**`left`**, **`right`**)
  - z punktu widzenia pozostałych elementów zmiana jest niedostrzegalna – po elemencie pozycjonowanym względnie pozostaje „dziura” równa jego wymiarom

Pozycjonowanie względne

CSS: pozycjonwanie statyczne

```
em {
  background-color: yellow;
}
```

Linia tekstu *w wąskiej kolumnie* może zostać złamana...

CSS: pozycjonwanie względne

```
em {
  background-color: yellow;
  position: relative;
  left: 1em;
  bottom: 1ex;
}
```

Linia tekstu *w wąskiej kolumnie* może zostać złamana...

Pozycjonowanie elementów CSS

- Pozycjonowanie **bezwzględne**
  - `position : absolute`
  - pudełko elementu jest najpierw całkowicie eliminowane ze składu, a następnie układane zgodnie z wartościami właściwości `top`, `bottom`, `left`, `right`, przy czym ich wartości odnoszą się do pudełka nadrzędnego w sensie hierarchii elementów w dokumencie
  - jeśli w ramach danego pudełka nadrzędnego określimy kilka elementów pozycjonowanych bezwzględnie, to ich wzajemne położenie na „stosie elementów" możemy regulować za pomocą właściwości `z-index`

Pozycjonowanie bezwzględne

CSS: pozycjonwanie statyczne

```
em {
  background-color: yellow;
}
```

Linia tekstu *w wąskiej kolumnie* może zostać złamana...

CSS: pozycjonwanie bezwzględne

```
em {
  background-color: yellow;
  position: absolute;
  width: 9em;
  height: 5em;
  left: 1.2em;
  top: 4ex;
}
```

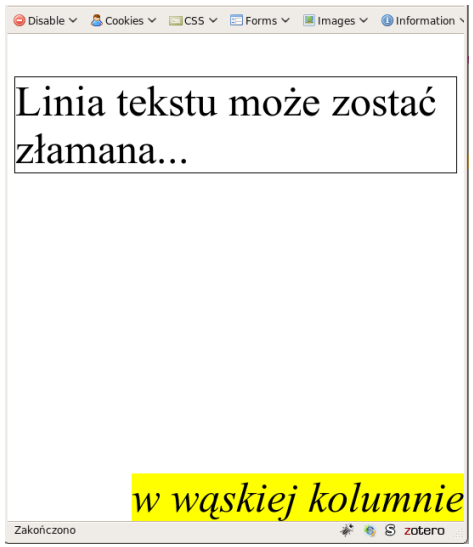
Linia tekstu może zostać zła
w wąskiej kolumnie

elementowi wewnętrznemu pozycjonowanemu bezwzględnie możemy zmieniać wymiary (!)

Pozycjonowanie elementów CSS

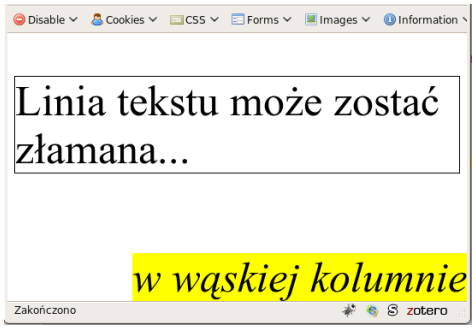
- Pozycjonowanie **ustalone**
  - `position : fixed`
  - podobnie, jak przy pozycjonowaniu bezwzględnym, pudełko elementu jest najpierw całkowicie eliminowane ze składu, a następnie układane zgodnie z wartościami właściwości `top`, `bottom`, `left`, `right`, przy czym ich wartości odnoszą się do okna przeglądarki jako „pudełka nadrzędnego"
  - pozycjonowanie „względem okna przeglądarki" oznacza w szczególności, że pozycja elementu nie zmienia się podczas przesuwania zawartości strony za pomocą suwaków przeglądarki

Pozycjonowanie ustalone



CSS: pozycjonwanie ustalone

```
em {
  background-color: yellow;
  position: fixed;
  bottom: 0;
  right: 0;
}
```



Pozycjonowanie elementów CSS

- Obsługa w przeglądarkach
  - IE < 7 nie obsługuje pozycjonowania ustalonego
  - IE < 7 mają problemy z marginesami elementów pozycjonowanych bezwzględnie

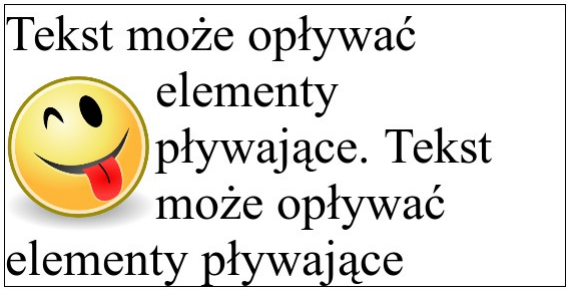
Elementy pływające

HTML: obrazek „opływany” przez tekst

```
<p>
 Tekst
 <img src="buzka.png"
  alt="buźka" /> może
 opływać elementy pływające.
 Tekst może opływać elementy
 pływające
</p>
```

CSS:

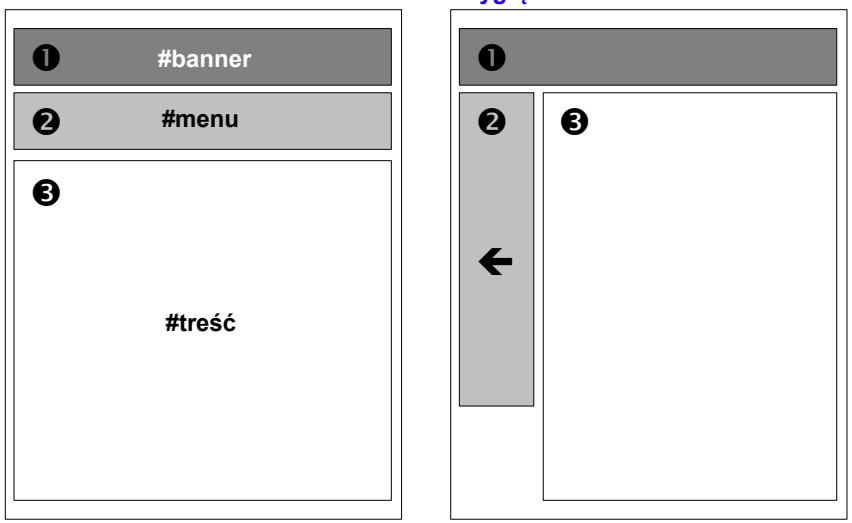
```
p   { border: 1px solid; }
img { float:  left;      }
```



CSS – definiowanie układu strony

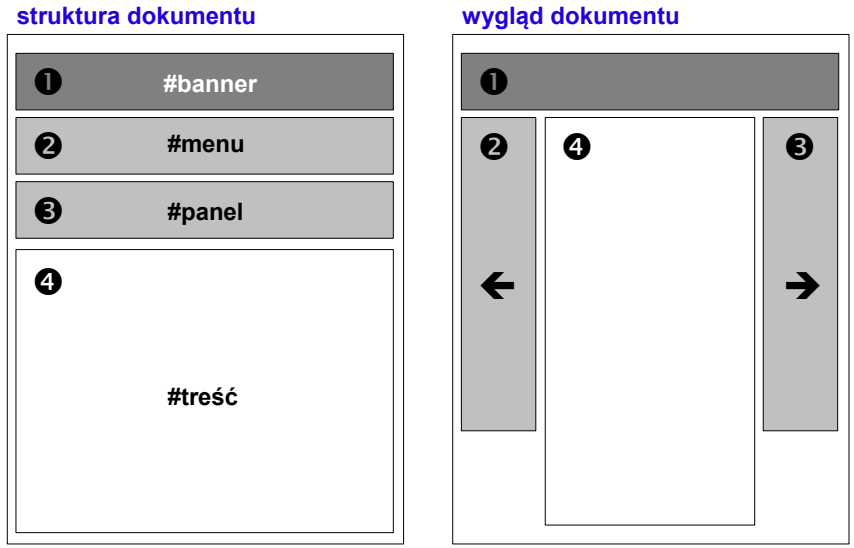
- Zanim obsługa CSS w przeglądarkach dojrzała, do definiowania układu strony powszechnie używano tabel HTML
- Obecnie zamiast tabel stosujemy mechanizmy oferowane przez język CSS
  - elementy div
  - elementy pływające
  - pozycjonowanie
  - ...

Prosty układ dwukolumnowy

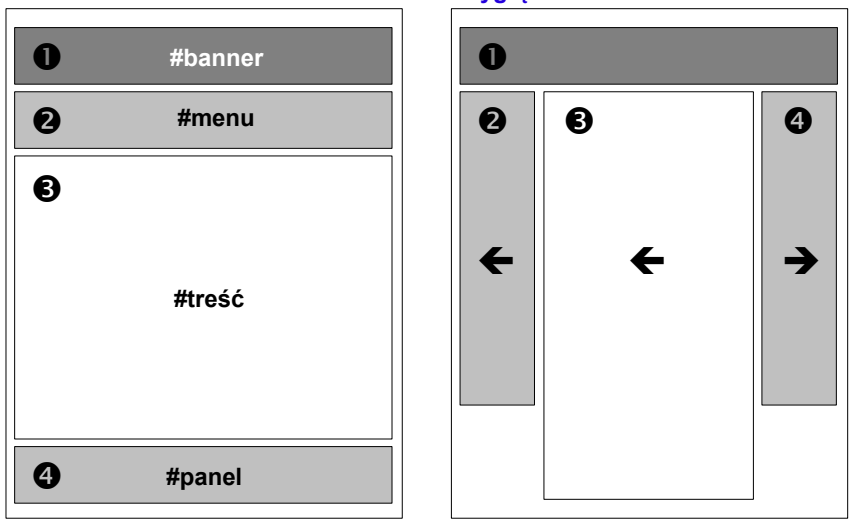


Prosty układ dwukolumnowy z tłem

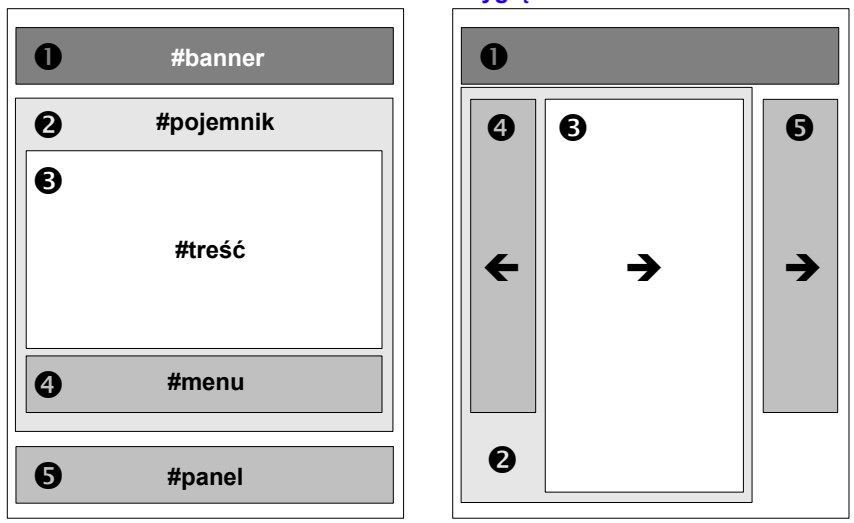
Prosty układ trójkolumnowy



Prosty układ trójkolumnowy inaczej

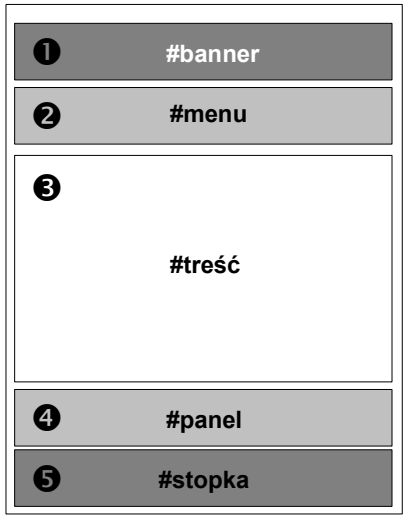


Prosty układ trójkolumnowy jeszcze inaczej

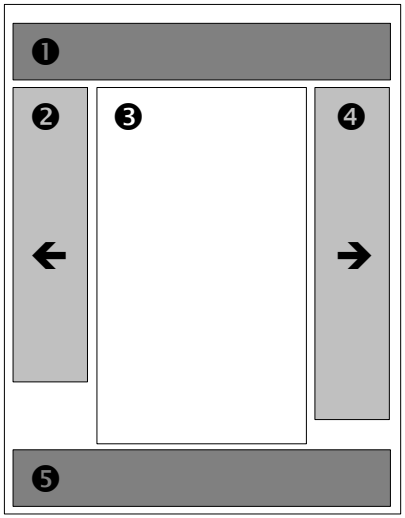


Prosty układ trójkolumnowy ze stopką

struktura dokumentu

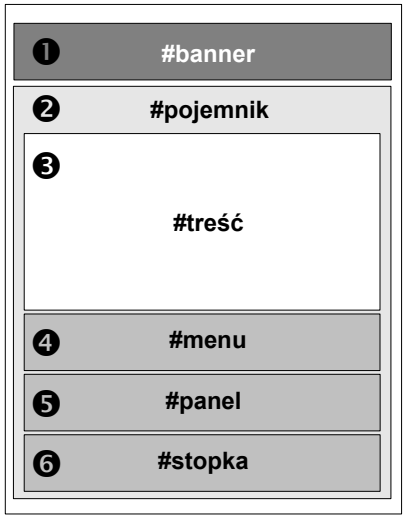


wygląd dokumentu

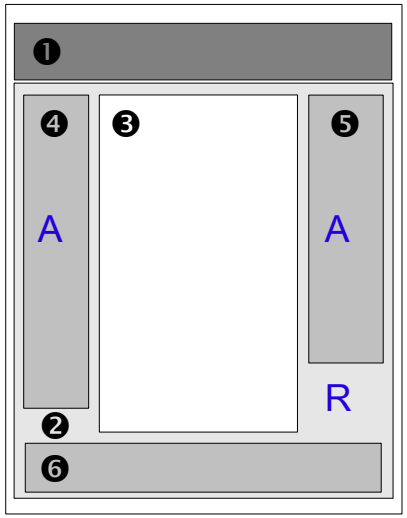


Wykorzystanie pozycjonowania

struktura dokumentu

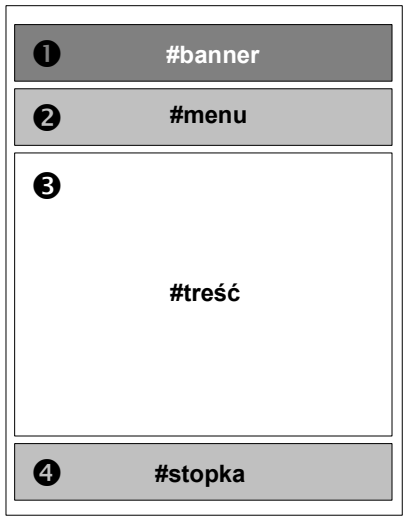


wygląd dokumentu

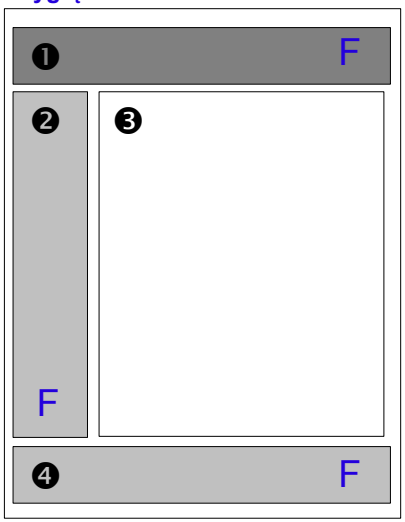


Wykorzystanie pozycjonowania c.d.

struktura dokumentu



wygląd dokumentu



Testy zgodności ze standardami

- **ACID1** (`http://en.wikipedia.org/wiki/Acid1`)
  - CSS 1.0
- **ACID2** (`http://en.wikipedia.org/wiki/Acid2`)
  - HTML, CSS 2.1, PNG, URL zawierające dane
- **ACID3** (`http://en.wikipedia.org/wiki/Acid3`)
  - DOM, elementy CSS3, Unicode, ECMA Script (JavaScript), SVG, XHTML, HTTP 1.1, ...

`http://www.sciactive.com/main/acid-test-results`

CSS – przydatne narzędzia

normalize.css

- Inteligentne „zerowanie” stylu przeglądarki
  - pozostawia przydatne wartości domyślne
  - normalizuje wygląd większości elementów
  - koryguje błędy przeglądarek (w miarę możliwości)
  - szczegółowe komentarze w kodzie
- Wsparcie dla:
  - Chrome, Firefox 3+, Safari 4+, Opera 10+, IE 6+

http://necolas.github.com/normalize.css/

SCSS (Sass) i Less

- Narzędzia do strukturalizacji stylów CSS
  - **SCSS** – wymaga środowiska języka Ruby
  - **Less** – skrypt w języku JavaScript
- Porównywalne możliwości
  - zmienne
  - zagnieżdżanie selektorów
  - „miksowanie” stylów
  - operacje na kolorach, liczbach, …
  - struktury kontrolne (**if**, …)
  - …

SCSS (Sass) i Less

- Zmienne

```
Sass              | Less
------------------+-----------------
$color: red;      | @color: red;
div {             | div {
  color: $color; |   color: @color;
}                 | }
```

W porównaniu wykorzystano: **https://gist.github.com/674726**

SCSS (Sass) i Less

- Zmienne – zasięg

```
Sass                          | Less
------------------------------+------------------------------
$color: black;                | @color: black;
.scoped {                     | .scoped {
  $bg: blue;                  |   @bg: blue;
  $color: white;              |   @color: white;
  color: $color;              |   color: @color;
  background-color: $bg;      |   background-color: @bg;
}                             | }
.unscoped {                   | .unscoped {
  color: $color;              |   color: @color;
  // To byłby błąd:           |   // To byłby błąd:
  // background: $bg;         |   // background: @bg;
}                             | }
```

Technologie internetowe – język CSS

SCSS (Sass) i Less

- Zmienne – zasięg

```
Sass output                   | Less output
------------------------------+------------------------------
.scoped {                     | .scoped {
  color: white;               |   color: white;
  background-color: blue;     |   background-color: blue;
}                             | }
.unscoped { color: white; }   | .unscoped { color: black; }
```

Technologie internetowe – język CSS

SCSS (Sass) i Less

- Zagnieżdżanie selektorów

```
Sass                | Less
--------------------+-----------------
p {                 | p {
  a {               |   a {
    color: red;     |     color: red;
    &:hover {       |     &:hover {
      color: blue;  |       color: blue;
    }               |     }
  }                 |   }
}                   | }
```

Technologie internetowe – język CSS

SCSS (Sass) i Less

- „Miksowanie” stylów

```
Sass                          | Less
------------------------------+------------------------------
@mixin bordered {             | .bordered {
  border: solid 1px black;    |   border: solid 1px black;
}                             | }
                              |
#menu a {                     | #menu a {
  @include bordered;          |   .bordered;
}                             | }
```

SCSS (Sass) i Less

- „Miksowanie” stylów z parametrami

```
Sass                                | Less
------------------------------------+------------------------------
@mixin bordered($width: 2px) {      | .bordered(@width: 2px) {
  border: $width solid black;       |   border: @width solid black;
}                                   | }
                                    |
#menu a {                           | #menu a {
  @include bordered(4px);           |   .bordered(4px);
}                                   | }
```

Technologie internetowe – język CSS

SCSS (Sass) i Less

- Operacje na kolorach – **Less**

```
lighten(@color, 10%);     // color 10% *lighter* than @color
darken(@color, 10%);      // color 10% *darker* than @color

saturate(@color, 10%);    // color 10% *more* saturated
desaturate(@color, 10%); // color 10% *less* saturated

fadein(@color, 10%);      // color 10% *less* transparent
fadeout(@color, 10%);     // color 10% *more* transparent
fade(@color, 50%);        // @color with 50% transparency

spin(@color, 10);         // color with 10 degree larger hue
spin(@color, -10);        // color with 10 degree smaller hue

mix(@color1, @color2);    // mix of @color1 and @color2
```

Technologie internetowe – język CSS

SCSS (Sass) i Less

- Operacje na wielkościach – problemy

```
Sass                          | Less
------------------------------+------------------------------
1cm * 2em => 2 cm * em        | 1cm * 2em => 2cm
2in * 3in => 6 in * in        | 2in * 3in => 6in
(1cm / 1em) * 4em => 4cm      | (1cm / 1em) * 4em => 4cm
2in + 3cm + 2pc => 3.514in    | 2in + 3cm + 2pc => 3.514in
3in / 2in => 1.5              | 3in / 2in => 1.5in
```

Technologie internetowe – język CSS

Małe „demo”

```
git clone https://github.com/wppj/tin-css-01.git
```