

CSS

JFDZ8 - 24.03.2018

| Prowadzący: Tomasz | Nastały

1. Poznajemy CSS



Poznajemy CSS

CSS (Cascading Style Sheets) jest to język służący do formatowania wyglądu znaczników HTML

W pliku z rozszerzeniem .css, wypisujemy reguły określające wygląd danego elementu na stronie.

Poznajemy CSS – dodawanie deklaracji

* W <head>, PLIK ZEWNĘTRZNY - **DOBRA PRAKTYKA**

<link rel="stylesheet" href="style.css"/> (HTML 5)

<link rel="stylesheet" type="text/css" href="style.css"/> (przed HTML5)

* W <head>, DEKLARACJA OSADZONA

<style type="text/css"> element {atrybut: wartość;} </style>

* Deklaracja na poziomie elementu strony (**ZŁA PRAKTYKA!**)

<div style="artybut: wartość; atrybut: wartosć; atrybut: wartość;"></div>

Poznajemy CSS – składnia arkuszy

```
ELEMENT1 {  
  atrybut: wartość;  
  atrybut: wartość;  
  atrybut: wartość;  
}
```

```
ELEMENT1, ELEMENT2 { (styl zostanie przypisany do obu elementów)  
  atrybut: wartość;  
}
```

Poznajemy CSS – model kaskadowy

Wszystkie deklaracje uzupełniają się w jeden "wirtualny" styl, który obowiązuje dla strony. Kolejność aplikowania:

- 1) Domyślny arkusz przeglądarki
- 2) Domyślny arkusz użytkownika przeglądarki
- 3) Zewnętrzne arkusze dołączone w sekcji <head> strony
- 4) Deklaracje w atrybucie style="" elementu html

PRZYKŁAD: <https://jsfiddle.net/13a4tngd/4/>

Poznajemy CSS – model kaskadowy

Kaskada i dziedziczenie określają porządek przyjmowania stylów przez elementy.

Porządek ten może zostać zmieniony przez dodanie dyrektywy **!important**

do wartości danego atrybutu:

```
p { artybut: wartość !important; }
```

Jednak nie jest to dobra praktyka. Należy się wystrzegać używania flagi **!important**

Poznajemy CSS – nadpisanie domyślnego arkusza przeglądarki

Dobłą praktyką jest dodanie arkusza reset.css, który nadpisuje domyślne style zaaplikowane przez przeglądarkę.

Przykład domyślnego stylu: dolny padding paragrafu (9px), lewy padding listy (40px)

Style dla pliku reset.css możemy pobrać z:

<https://meyerweb.com/eric/tools/css/reset/>

Poznajemy CSS – co możemy opisać

jednostki miary elementu: px, %, em, pt, cm, in ...

kolor: #ff0000, red, rgb(255,0,0), rgba(255,0,0,1) ...

rozmiar elementu: width, height, min-width, max-width, min-height, max-height

parametry blokowe: padding (padding-top, padding-left...), margin, border

tekst: font-family, font-size, font-weight, line-height, color, text-align ...

parametry tła: background-color, background-image, background-repeat ...

pozycjonowanie i typ_widoczności: position, display, overflow

Poznajemy CSS – selektory

▪ typ elementu	h1 {atrybut: wartość;}	<h1>Tytuł</h1>
▪ klasa elementu	.mojStyl {atrybut: wartość;}	<div class="mojStyl"></div>
▪ id elementu	#mojDiv {atrybut: wartość;}	<div id="mojDiv"></div>
▪ pseudoklasy akcji	a.mojStyl:hover {...}	
▪ pseudoklasy	#paragraf44:first-letter {...}	<p id="paragraf44"></p>
▪ pseudoklasy typu	input[type="text"] {...}	<input type="text" value="test" />

Poznajemy CSS – precyzyjne selektory













```
#naglowek nav.navigacja ul > li:nth-child(2n):hover span { }
```

Nie jest to dobra praktyka – należy się wystrzegać takich selektorów, są nieczytelne i ciężko je nadpisać.

Poznajemy CSS – selektor łączony i selektor ze spacją

- Element o klasie active zagnieżdżony w elemencie o klasie .submenu
`.active .submenu` { (spacja – zagnieżdżenie!)
 atrybut: wartość;
 atrybut: wartość;
 atrybut: wartość;
}
- Element posiadający klasę active i submenu (brak spacji między nazwami)
`.active.submenu` { (brak spacji!, element musi mieć obie klasy!)
 atrybut: wartość;
}

Poznajemy CSS – hierarchia selektorów

 <p>a</p> <p>1 x element selector</p> <p>Sith: 0, 0, 1</p>	 <p>p a</p> <p>2 x element selectors</p> <p>Sith: 0, 0, 2</p>	 <p>.whatever</p> <p>1 x class selector</p> <p>Sith: 0, 1, 0</p>	 <p>a.whatever</p> <p>1 x element selector 1 x class selector</p> <p>Sith: 0, 1, 1</p>
 <p>p a.whatever</p> <p>2 x element selectors 1 x class selector</p> <p>Sith: 0, 1, 2</p>	 <p>.whatever .whatever</p> <p>2 x class selectors</p> <p>Sith: 0, 2, 0</p>	 <p>p.whatever a.whatever</p> <p>2 x element selectors 2 x class selectors</p> <p>Sith: 0, 2, 2</p>	 <p>#whatever</p> <p>1 x id selector</p> <p>Sith: 1, 0, 0</p>
 <p>a#whatever</p> <p>1 x element selector 1 x id selector</p> <p>Sith: 1, 0, 1</p>	 <p>.whatever a#whatever</p> <p>1 x element selectors 1 x class selector 1 x id selector</p> <p>Sith: 1, 1, 1</p>	 <p>.whatever .whatever #whatever</p> <p>2 x class selectors 1 x id selector</p> <p>Sith: 1, 2, 0</p>	 <p>#whatever #whatever</p> <p>2 x id selectors</p> <p>Sith: 2, 0, 0</p>

Poznajemy CSS – ostatnia deklaracja wygrywa

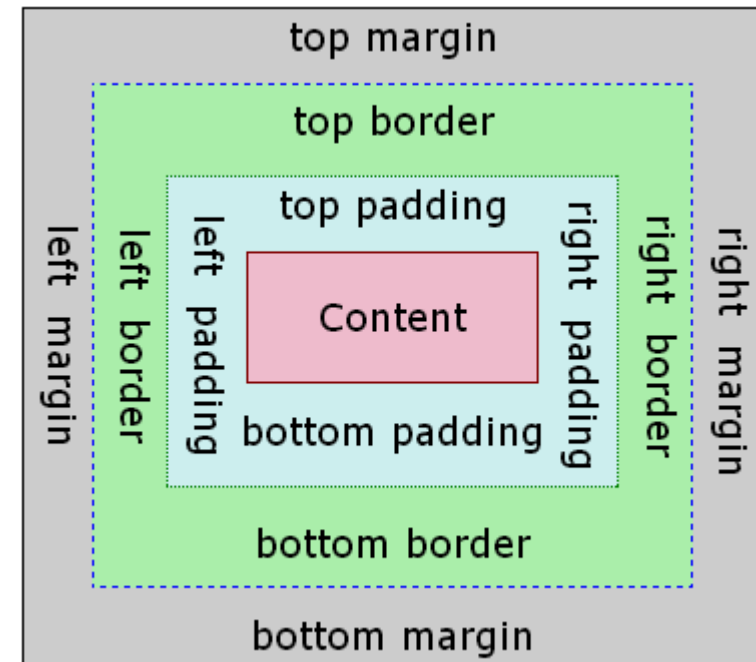
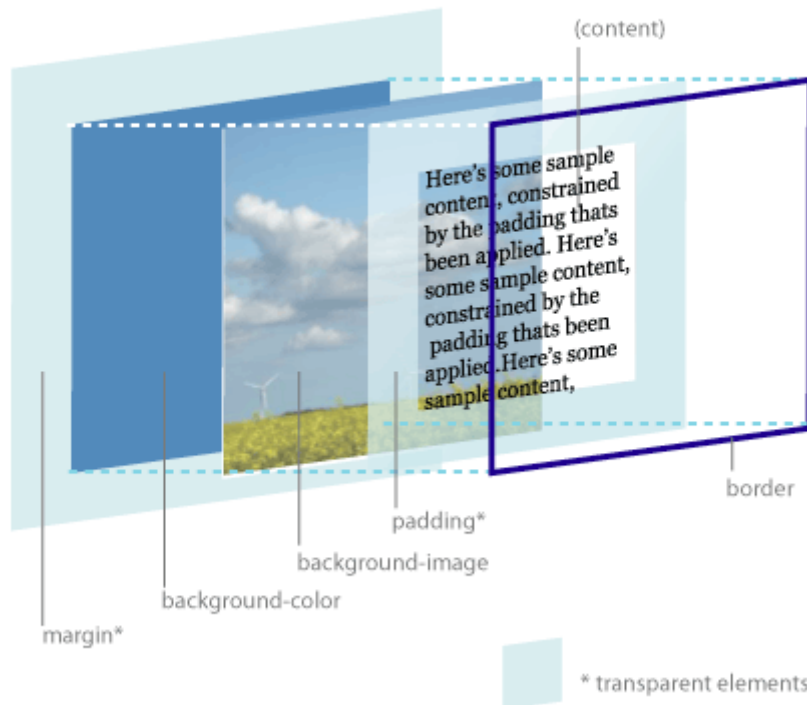
```
div {  
  margin: 5px;  
  margin: 10px;  
  margin: 2px;  
}
```

div będzie posiadać margines o wartości 2px

2. Model pudełkowy

Model pudełkowy CSS

THE CSS BOX MODEL HIERARCHY

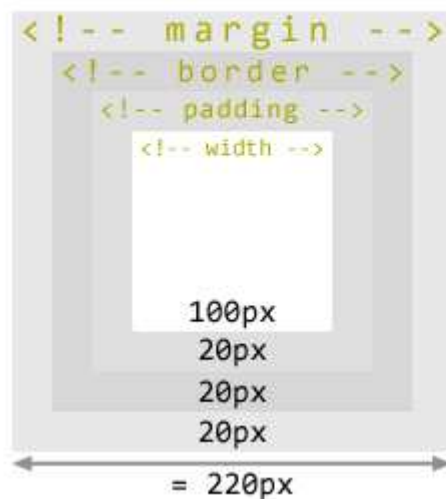


Model pudełkowy CSS

Zasada Box-sizing

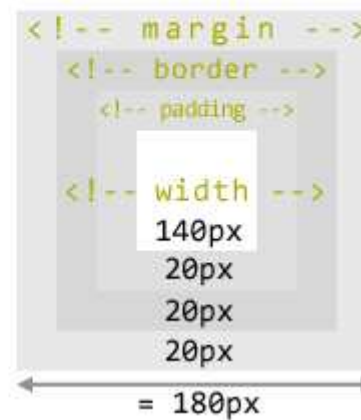
CSS Box Model

`box-sizing: content-box;`



`box-sizing: border-box;`

As opposed to the content-box model, the border-box model includes the border and padding inside of the width.



Model pudełkowy CSS

Właściwości dla elementu blokowego

margin: 10px; (w każdym kierunku)
margin: 10px 20px; (górny/dolny prawy/lewy)
margin: 10px 20px 5px 3px; (zgodnie z zegarem)
margin-left: 5px;
margin-top: 10px;

Paddingi, zasady analogicznie jak dla marginesów:

padding: 10px;
padding : 10px 20px;
padding : 10px 20px 5px 3px;
padding-right: 50px;
padding-bottom: 10px;

border: 1px solid black;
border: 5px dashed red;

width: 100px;
height: 500px;
width: 30%;
height: 70%;

3.Elementy liniowe, liniowo-blokowe, blokowe

Poznajemy CSS

Display: inline – element zajmuje tyle miejsca, ile potrzebuje

Display: inline-block – element zajmuje miejsca tyle ile potrzebuje, ale zachowuje jeszcze pewne cechy elementu blokowego

Display: block – element zajmuje zawsze 100% dostępnej szerokości rodzica.

Poznajemy CSS

`display: block`

display:
block

display:
block

display:
block

`display: inline`

display: inline display:
inline display: inline

`display: inline-block`

display: display: display:
inline- inline- inline-
block block block

Poznajemy CSS – display

Elementy z display: inline nie posiadają górnego i dolnego marginesu.

Jeśli chcemy do nich dodać margines górny lub dolny, musimy przestawić ich wartość display na inline-block lub block.

Elementowi inline nie możemy również nadać stałej szerokości i wysokości (możliwe wyłącznie przy zmianie trybu display)

```
span {  
  width: 300px;  
  height: 100px;  
}
```

NIE ZADZIAŁA

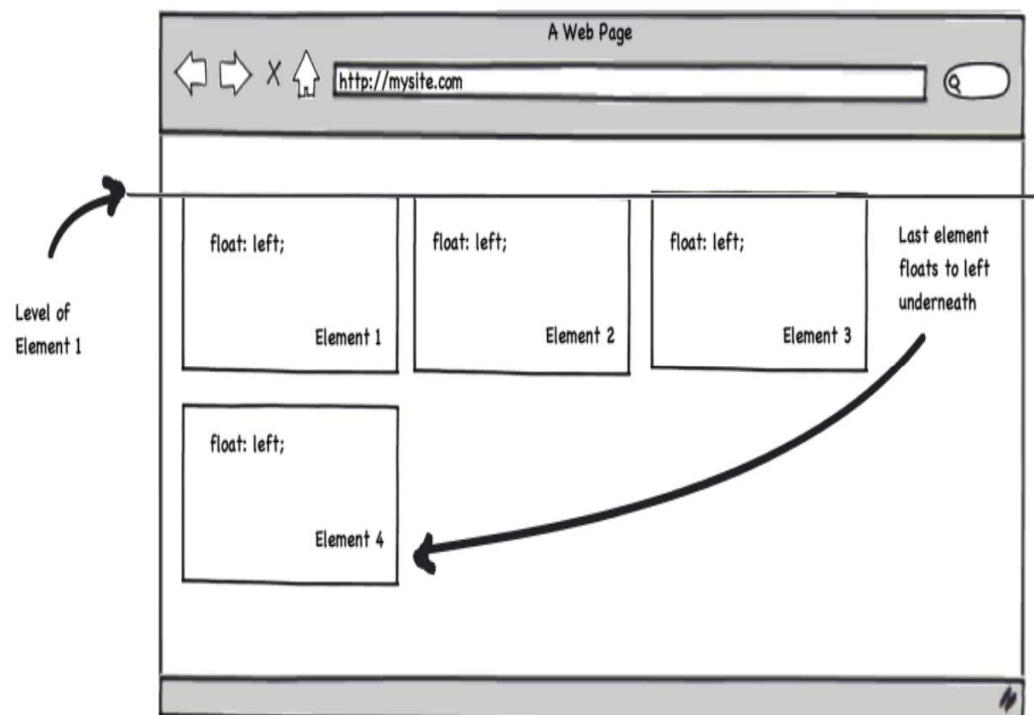
```
span {  
  width: 300px;  
  height: 100px;  
  display: block; // lub display: inline-block;  
}
```

ZADZIAŁA

4. Floats

Floats

W celu ustawienia elementów obok siebie, możemy wykorzystać Float'y.



Floats

```
#rodzic {  
  background-color:black;  
  border:3px dashed red;  
  font-size:1.4em;  
}  
  
#dziecko1 {  
  width:40%;  
  background-color:lightblue;  
}  
  
#dziecko2 {  
  width:30%;  
  background-color:lightgreen;  
}
```

div - szerokość 40%

div - szerokość
30%

Floats

```
#dziecko1 {
  float:left;
  width:40%;
  background-color:lightblue;
}
```

```
#dziecko2 {
  float:left;
  width:30%;
  background-color:lightgreen;
}
```

PROBLEM – rodzic nie rozciągnął się na wysokość swoich dzieci.

ROZWIĄZANIE -

Dodać overflow: auto; do stylu rodzica

div - szerokość 40%

div - szerokość 30%

Floats

div - szerokość 40% div - szerokość 30% to jest kolejny element - div -
który został dodany do naszego układu elementów HTML

Floats

```
#dziecko3 {  
  clear:both;  
  background-color:gold;  
}
```

div - szerokość 40%

div - szerokość
30%

to jest kolejny element - div - który został dodany do
naszego układu elementów HTML

Floats

```
.floatfix:after {  
  content: '';  
  display: block;  
  clear: both;  
}  
  
</style>  
</head>  
  
<body>  
  
  <div id="rodzic" class="floatfix">  
    <div id="dziecko1">div - 1</div>  
    <div id="dziecko2">div - 2</div>  
  </div>
```

div - szerokość 40%

div - szerokość
30%

to jest kolejny element - div - który został dodany do
naszego układu elementów HTML

Floats vs Inline-block –problem white space'ów

```
<ul>  
  <li>one</li>  
  <li>two</li>  
  <li>three</li>  
  <li>four</li>  
  <li>five</li>  
</ul>
```

display: inline-block

one	two	three	four	five
-----	-----	-------	------	------

float: left

one	two	three	four	five
-----	-----	-------	------	------

Floats vs Inline-block –problem white space'ów

```
<ul>  
  <li>one</li>  
  <li>two</li>  
  <li>three</li>  
  <li>four</li>  
  <li>five</li>  
</ul>
```

display: inline-block

one	two	three	four	five
-----	-----	-------	------	------

float: left

one	two	three	four	five
-----	-----	-------	------	------

Inline-block white space – możliwe rozwiązania

Inline-block / white-space bug

original...

one two three

fixed by funky code formatting...

one two three

fixed by adding html comments...

one two three

fixed by CSS margin-right: -4px; (breaks in IE6&7)...

one two three

fixed by omitting the

one two three

fixed with font-size: 0 via: <http://twitter.com/#!/garand/status/183253526313566208>

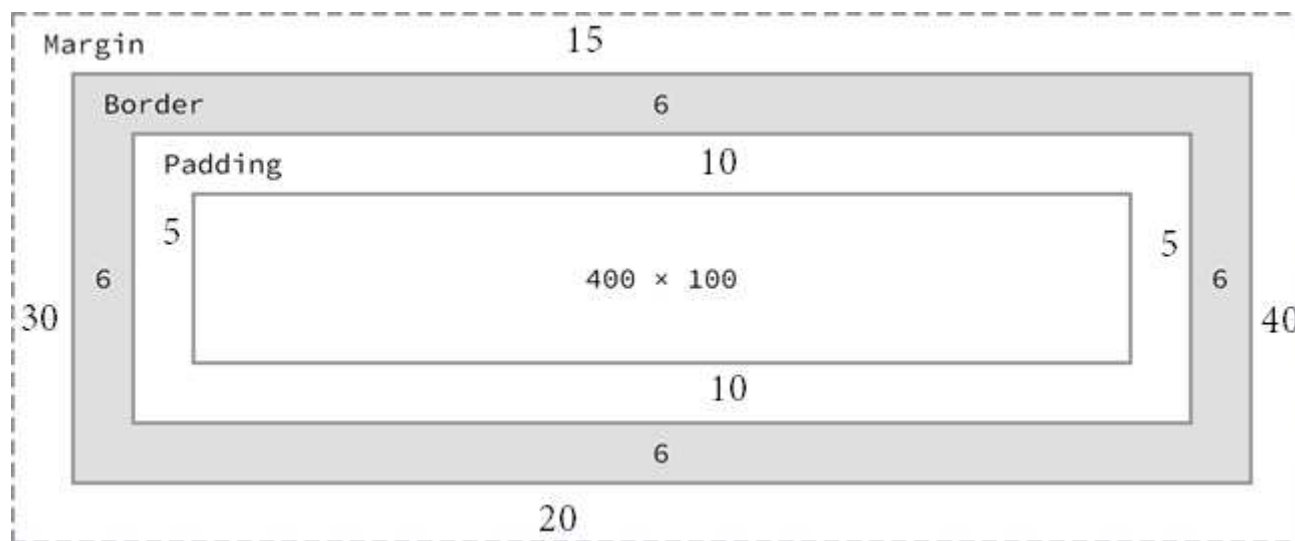
one two three

flexbox

one two three

ZADANIE 1 – model pudełkowy

Stwórz poniższy blokowy element HTML i nadaj mu następujące cechy.
Sprawdź różnicę w rozmiarze elementu z zastosowaniem zasady **border-box**.

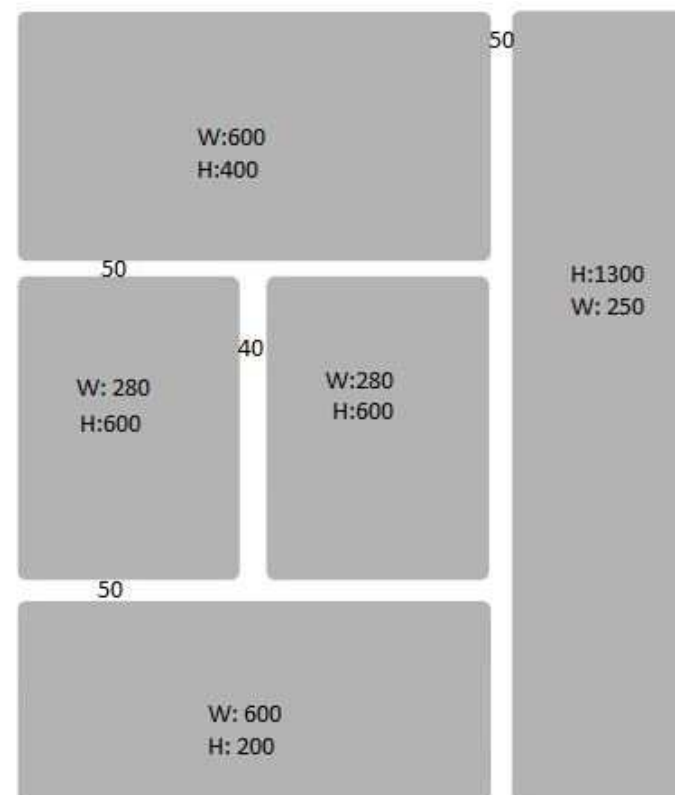


ZADANIE 2 – stwórz strukturę z użyciem float (wspólnie)

Utwórz plik **floats.html** i stwórz dla niego arkusz stylów o nazwie **styles** i podłącz go w sekcji HEAD.

Następnie wykorzystaj atrybuty: **width, height, float, clear, margin** do stworzenia następującego układu elementów. Wykorzystaj klasy w celu uchwycenia elementów w arkuszu.

Możesz użyć atrybutu **background-color** w celu rozróżnienia elementów



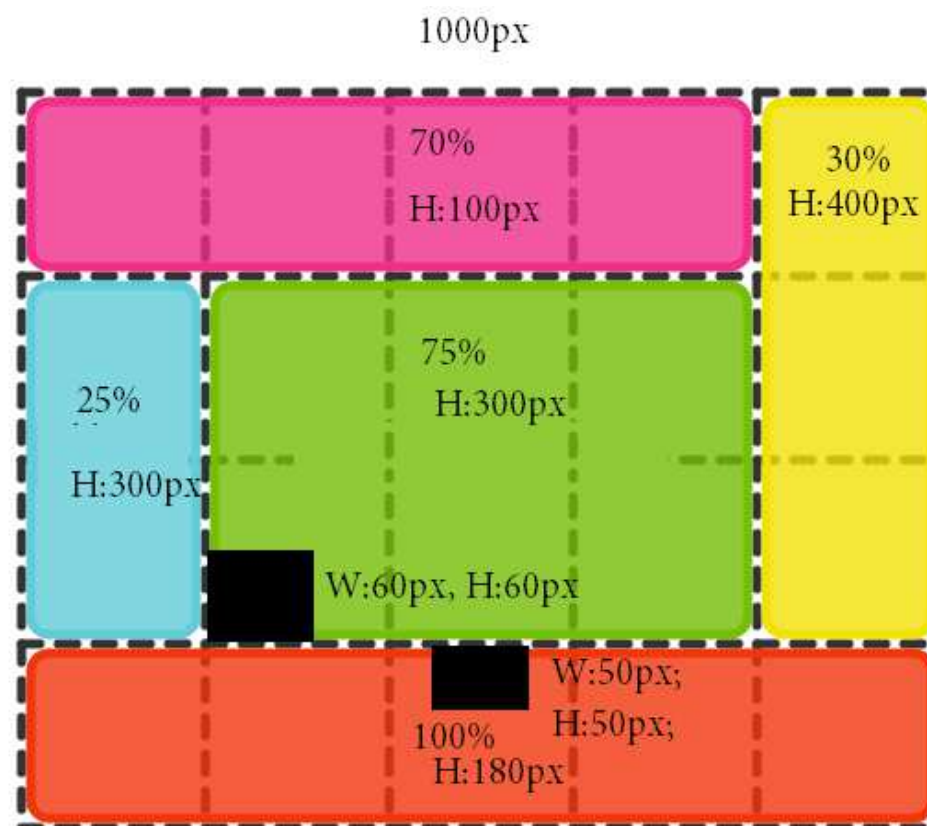
ZADANIE 3.A – display inline-block + float

Otwórz plik index.html z ostatnich zajęć. Otwórz plik **styles.css** w katalogu **assets/styles**. Upewnij się, że arkusz jest podłączony w sekcji head pliku index.html.

1. Wykorzystaj poznany atrybut **display: inline-block** w celu ustawienia elementów listy w nawigacji <nav> w orientacji poziomej.
* Wykorzystaj właściwości list-style, background, color, font-size do odwzorowania stylu.
2. Wykorzystaj płynięcie elementów w celu wyrównania całej listy do prawej strony headera, a loga do lewej.
3. Wykorzystaj płynięcie elementów w celu ustawienia sekcji features jako 4 kafelki po szerokościach 50%, oraz icon-container powinien otrzymać szerokość 40px, a opis feature zająć pozostałą przestrzeń.

* ZADANIE EKSTRA

Odtwórz strukturę:



5. REGUŁY DZIEDZICZONE

Reguły dziedziczone

Pewne reguły są dziedziczone przez elementy potomne, np.:

- color
- text-align
- font-size
- font-family
- i inne

6. JEDNOSTKI WYMIAROWE

Jednostki wymiarowe

vh – wielkość viewport (okna przeglądarki) – przydatne przy tworzeniu obrazków rozciągniętych na całe okno (<https://codepen.io/Tipue/pen/tJIHk>)

px – wielkość w pikselach ekranu (najbardziej popularna jednostka)

% - wielkość procentowa. Przydatna przy tworzeniu gridów i layoutów stron

em – np. do elastycznych wielkości fontów, przykład <https://jsfiddle.net/m0m1tqtc/3/>

Opis wszystkich jednostek:

https://www.tutorialspoint.com/css/css_measurement_units.htm

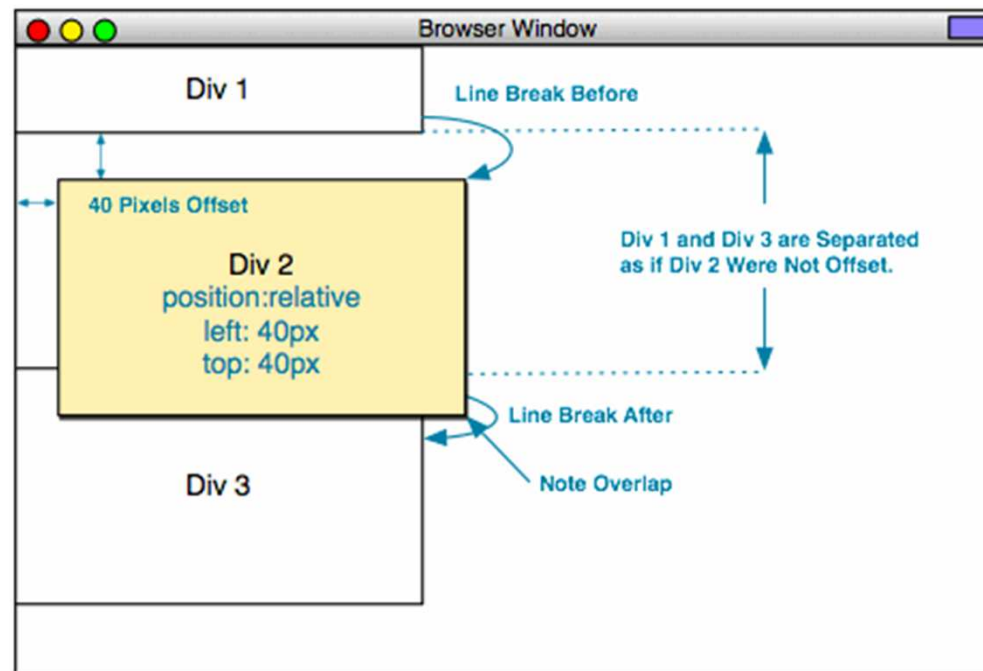
7. POZYCJA RELATYWNA, ABSOLUTNA, USTALONA

Typy pozycji

Element na stronie może przyjąć 4 różne pozycje:

- **Static** (domyślna, position: static)
- **Relative** (position: relative)
- **Absolute** (position: absolute)
- **Fixed** (position: fixed)

Pozycja relatywna



Pozycja relatywna

- Pozycja relatywna przesuwa element o odległości ustawione w TOP / BOTTOM / LEFT / RIGHT
- Obiekt pozostaje w tym samym miejscu w biegu dokumentu

<https://jsfiddle.net/g34mgsy8/4/>

- Zmienia się tylko miejsce gdzie jest rysowany
- Ta właściwość nie może być ustawiona na element tabeli (np. td) ale na ich zawartości już tak

Pozycja absolutna

```
div {  
  position: absolute;  
  top: 0;  
  left: 0;  
}
```

```
div {position: relative;}
```

```
div {  
  position: absolute;  
  bottom: 50%;  
  right: 0;  
}
```

Pozycja absolutna

- Umieszcza obiekt w pozycji wskazanej poprzez top, right, bottom, left
- Element zostaje wyjęty z biegu dokumentu, przestaje zajmować miejsce w swojej oryginalnej pozycji
<https://jsfiddle.net/g34mgsy8/5/>
- Pozycja zostaje ustalona względem elementu zawierającego, którym domyślnie jest <body>
- Jeśli rodzic elementu ma pozycję **inną niż static**, to położenie elementu zostaje wyliczone **względem tego rodzica** a nie już względem <body>

Pozycja absolutna

- Position absolute najczęściej jest używane wraz z rodzicem, który ma position: relative
- Ustawienie pozycji absolutnej, zdejmuje z elementu blokowego jego pewne cechy (przestaje zajmować 100% dostępnej szerokości rodzica)
PRZYKŁAD: <https://jsfiddle.net/taxhkpvm/11/>
- Pozycjonowanie absolutne względem <body> z reguły nie ma sensu, ponieważ element się przemieszcza wraz ze zmianą szerokości okna

Pozycja absolutna + relatywna

PRZYKŁAD RODZICA Z POZYCJĄ RELATYWNĄ I DZIECKA Z POZYCJĄ
ABSOLUTNĄ:

<https://jsfiddle.net/taxhkpvm/16/>

Pozycja absolutna – rozciągnięcie elementu

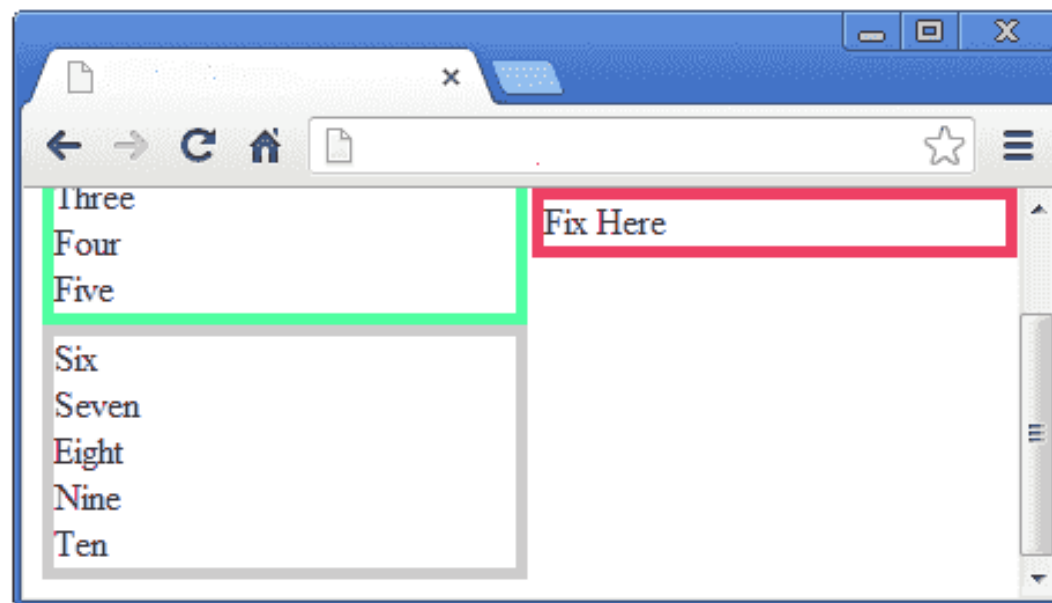
- Element możemy rozciągnąć jeśli nadamy mu jednocześnie Właściwość left + right, ale nie nadamy właściwości width.

Analogicznie dla top + bottom i bez height.

Przykład rozciągnięcia:

<https://jsfiddle.net/taxhkpvm/9/>

Pozycja stała (FIXED)



Pozycja stała – FIXED rozciągnięcie elementu

Pozycja FIXED idealnie nadaje się np dla nawigacji, która ma przesuwać się wraz z przewijaniem strony. Otrzymujemy efekt przyczepienia się elementu na stałe na ekranie.

- Umieszcza obiekt w pozycji wskazanej poprzez top, right, bottom, left
- Pozycja fixed jest **zawsze** wyliczona względem okna przeglądarki
- Position: fixed ustawione na elemencie blokowym, zdejmuje z niego pewne cechy elementu blokowego (nie rozciąga się już na 100% dostępnej szerokości). Aby rozciągnąć element na całkowitą szerokość, dodajemy left: 0 i right: 0 lub width: 100%

Pozycja stała – FIXED rozciągnięcie elementu

Przykład użycia:

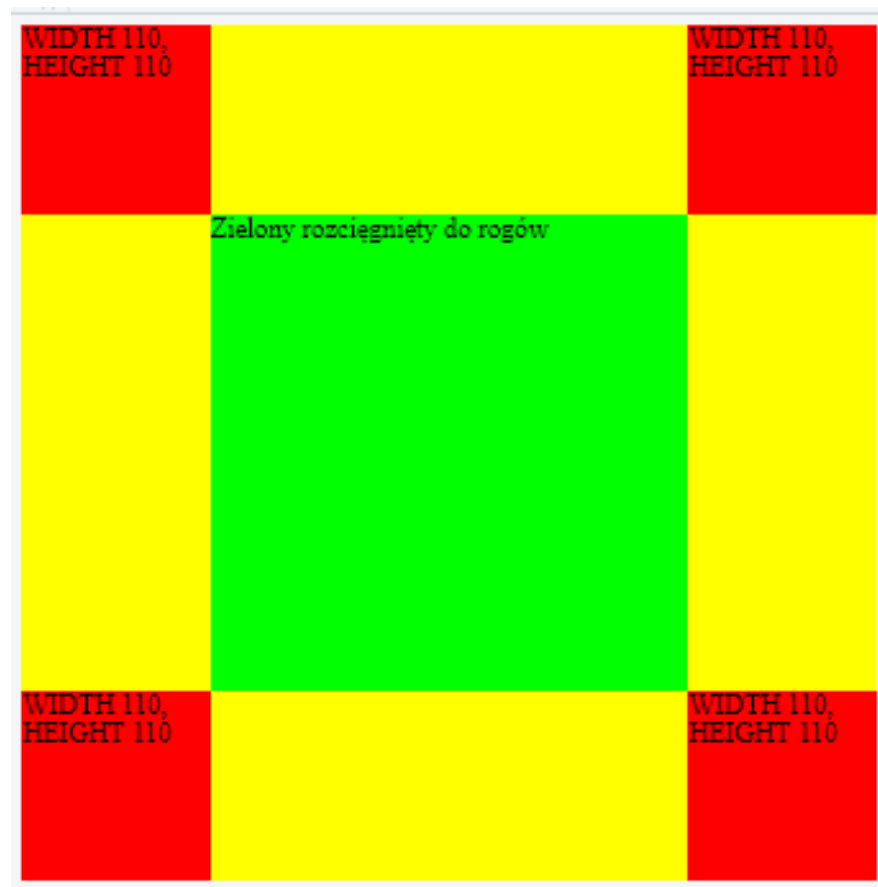
<https://jsfiddle.net/ebrs0zqk/>

ZADANIE 4.

Stwórz plik zadania-inne.html i odpowiadający mu arkusz zadania-inne.css

Przy użyciu poznanych zasad dotyczących pozycjonowania relatywnego i absolutnego, odtwórz widoczną strukturę.

Wymiar całkowity:
Width: 500px
Height: 500px



ZADANIE 5.

W pliku index.html stworzonym w ubiegłym tygodniu, element z nawigacją.

Używając odpowiedniej wartości position, przypnij nawigację na stałe do okna przeglądarki oraz rozciągnij ją do lewej i prawej krawędzi okna.

8. Z-INDEX A STOS ELEMENTÓW

Z-INDEX

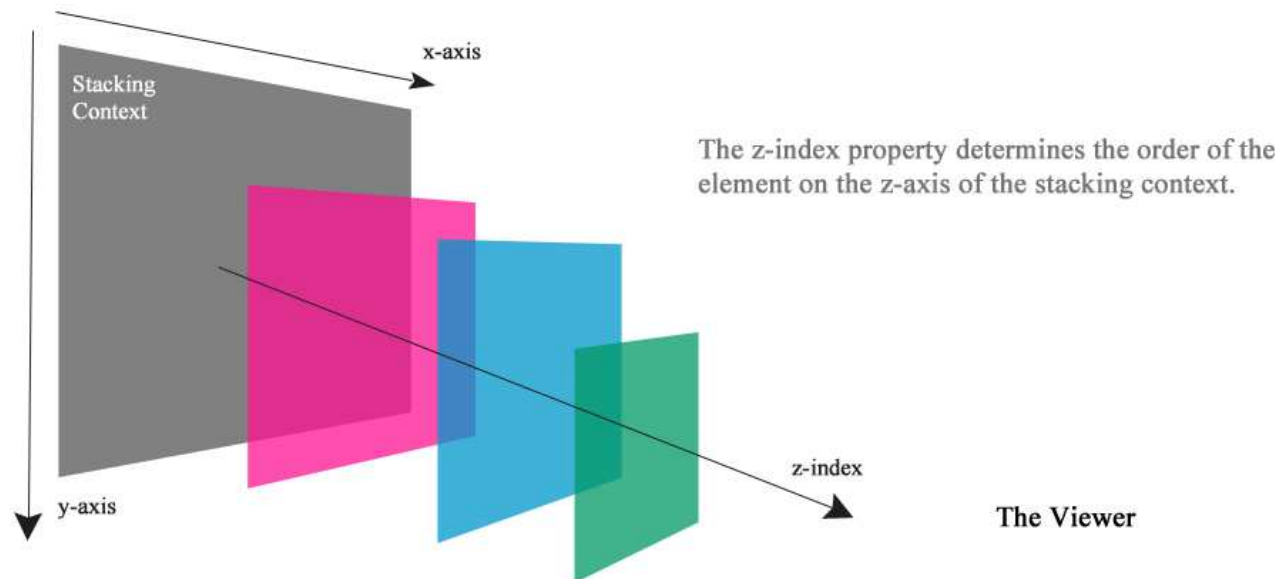
Jeśli nadamy elementom pozycje fixed / absolute / relative, to mogą zdarzyć się sytuacje, że jeden element przysłoni inny.

Aby nad tym zapanować, stosujemy właściwość Z-INDEX.

PRZYKŁAD: <http://jsfiddle.net/jpGdP/269>

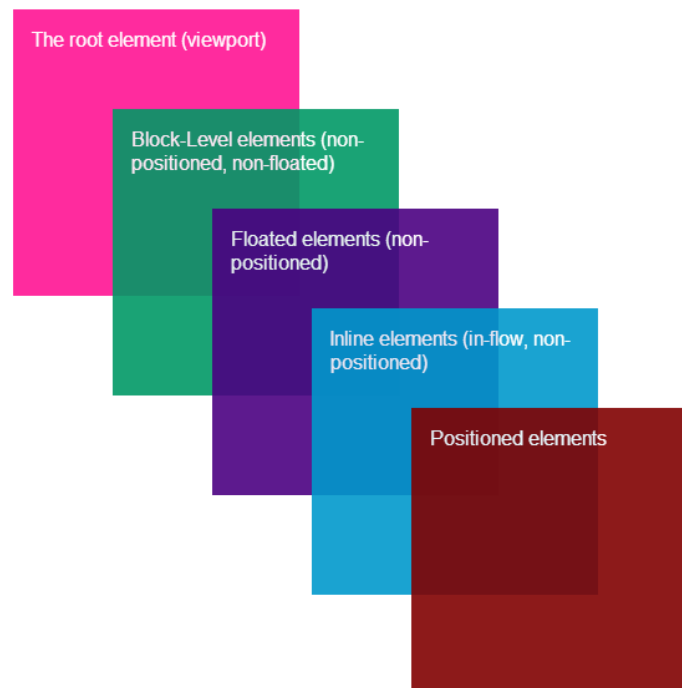
Z-INDEX

Im wyższy Z-index, tym wyższy priorytet elementu w stosie



Z-INDEX

Domyślne wyświetlanie elementów w stosie przez przeglądarkę:



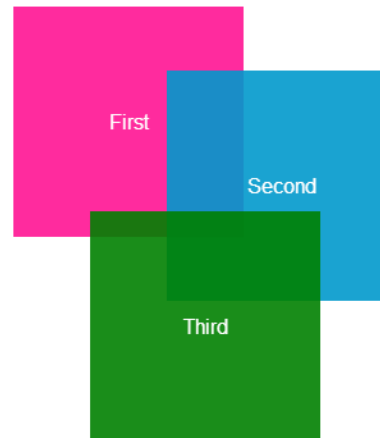
Z-INDEX



Z-INDEX

```
#first, #second, #third {  
    /* general styling here */  
    /* ... */  
  
    position: absolute;  
}  
  
#first {  
    /* position offsets.. */  
}  
  
#second {  
    /* position offsets... */  
}  
  
#third {  
    /* position offsets... */  
}
```

```
<div id="first">First</div>  
<div id="second">Second</div>  
<div id="third">Third</div>
```



Elements are positioned absolutely and overlap.

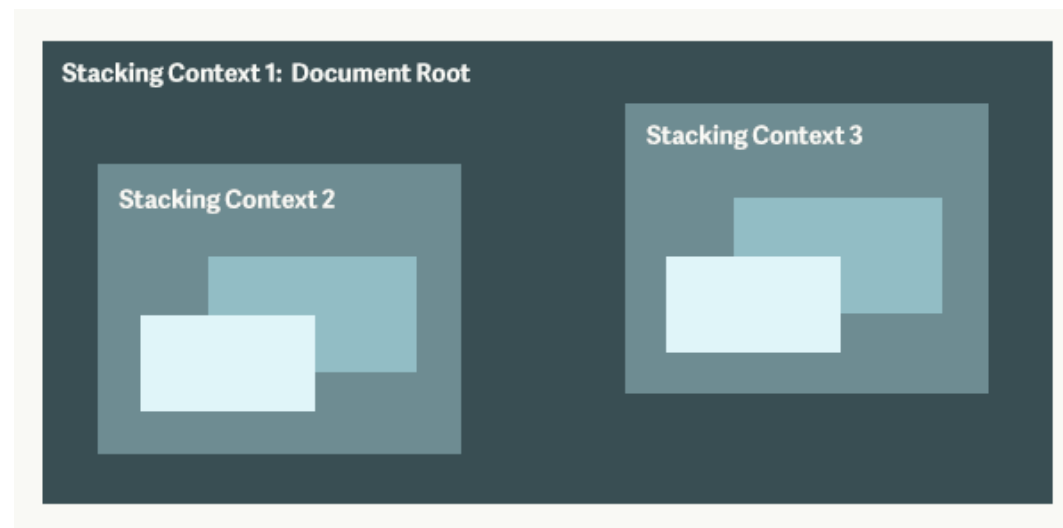
They are stacked according to their order in the source document.



Z-INDEX – STACK CONTEXT

Czasem zdarza się tak, że mimo ustawienia z-index na 999999, to i tak element z mniejszym z indexem go przykrywa. Znaczenie ma kontekst stosu.

<https://jsfiddle.net/yk2xmaq7/1/>

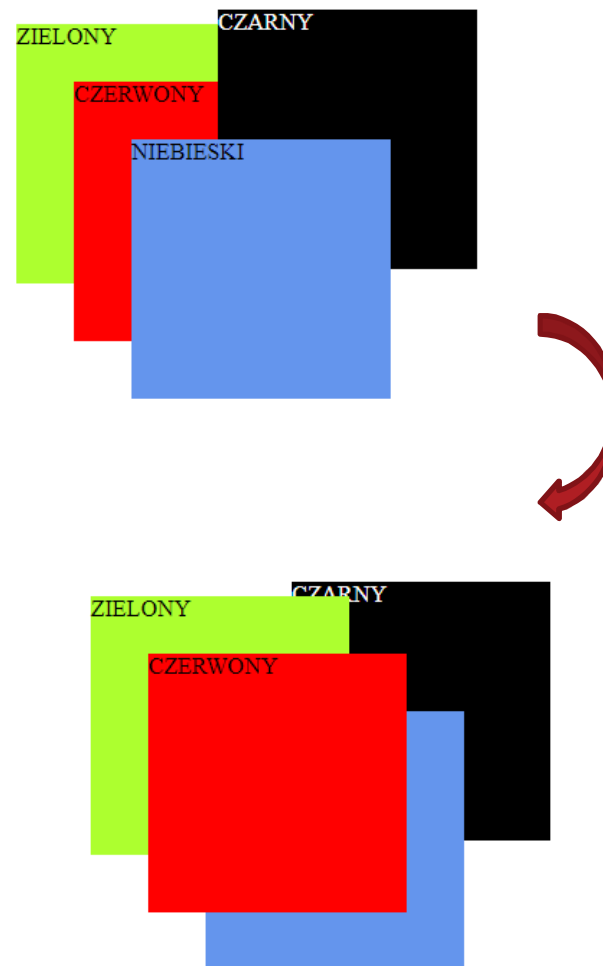


ZADANIE 6.

Otwórz plik zadania-inne.
Korzystając z pozycji absolute,
odtwórz podobny układ
elementów. Wymiar 200x200px

Ustawiając odpowiednie indexy,
ustaw elementy w następującej
kolejności:

1. Czerwony (górze stosu)
2. Niebieski
3. Zielony
4. Czarny



9. WYŚRODKOWYWANIE ELEMENTÓW W POZIOMIE I PIONIE

WYŚRODKOWANIE ELEMENTU LINIOWEGO W POZIOMIE

Element liniowy centrujemy w **poziomie** poprzez nadanie rodzicowi atrybutu **text-align: center;**

PRZYKŁAD: <https://jsfiddle.net/taxhkpvm>

```
<div>
  <span>Jestem liniowy</span>
</div>
```

```
div {
  text-align: center;
}
```

WYŚRODKOWANIE ELEMENTU BLOKOWEGO W POZIOMIE

Element blokowy centrujemy w **poziomie** poprzez nadanie elementowi wartości margin-left: auto i margin-right: auto;

PRZYKŁAD: <https://jsfiddle.net/taxhkpvm/1/>

```
<div>
  Jestem blokowy
</div>

div {
  margin-left: auto;
  margin-right: auto;
  /* lub margin: 0 auto; */
}
```

WYŚRODKOWANIE ELEMENTU LINIOWEGO W PIONIE

1) **Element liniowy** możemy wyrównać w pionie poprzez nadanie równych paddingów góra + dół i dodanie `display: block` lub `display: inline-block`;

PRZYKŁAD: <https://jsfiddle.net/taxhkpvm/4/>

2) Element liniowy możemy wyrównać w pionie poprzez nadanie równej wartości dla **line-height** elementu liniowego oraz **height** jego rodzica

PRZYKŁAD: <https://jsfiddle.net/taxhkpvm/2/>

WYŚRODKOWANIE ELEMENTU BLOKOWEGO W PIONIE

Element blokowy możemy wyrównać w **pionie** poprzez nadanie rodzicowi elementu **pozycji relatywnej** a elementowi pozycji **absolutnej wraz z top: 50%** i **transform: translateY(-50%)**

PRZYKŁAD: <https://jsfiddle.net/taxhkpvm/6/>

WYŚRODKOWANIE ELEMENTU BLOKOWEGO W PIONIE I POZIOMIE

Element blokowy możemy wyrównać w **pionie** i **poziomie** jednocześnie poprzez nadanie rodzicowi elementu **pozycji relatywnej** a elementowi pozycji **absolutnej wraz z top: 50%, left: 50% i transform: translate(-50%, -50%);**

PRZYKŁAD: <https://jsfiddle.net/taxhkpvm/8/>

ZADANIE 7.

Otwórz plik index.html

1. Kontener z nagłówkiem H1
wyśrodkuj w poziomie i pionie
względem rodzica. Rodzicowi
nadaj wysokość 100vh i
background np.: #292929;
2. Wyśrodkuj nagłówki h2
poszczególnych sekcji.

10.10 ELEMENTU

Stylowanie tła

Background-color: red;	<- ustawia jednolity kolor tła
Background-image: url(images/image.jpg);	<- ustawia obrazek jako tło
Background-repeat: repeat no-repeat repeat-x repeat-y	<- powtarzanie tła
Background-position: 0 0 ;	<- pozycja tła
Background-attachment: scroll fixed	<- przypięcie tła na stałe

Stylowanie tła

```
/* Example 1: default. */  
background-position: 0 0; /* i.e. Top-left corner of element. */  
  
/* Example 2: move the image to the right. */  
background-position: 75px 0;  
  
/* Example 3: move the image to the left. */  
background-position: -75px 0;  
  
/* Example 4: move the image down. */  
background-position: 0 100px;
```



Stylowanie tła – prosty parallax efekt

<http://callmenick.com/post/simple-parallax-scrolling-effect>

http://callmenick.com/_development/simple-parallax-effect/

```
background-repeat: no-repeat;  
background-attachment: fixed;  
background-size: cover;
```

ZADANIE 8.

W pliku index.html, stworzonym w zeszłym tygodniu, wyszukaj sekcję o klasie „hero”

Następnie elementowi o klasie **hero**, ustaw tło, które będzie zawsze rozciągnięte na całe okno (100vh, mamy;). Obrazek na tło znajdziesz w katalogu **assets/images**

Jeśli uda Ci się wyświetlić tło, dodaj efekt pseudo paralaxy za pomocą właściwości **background-attachment: fixed;**

11.STYLOWANIE TEKSTU

Stylowanie tekstu

<u>color</u>	Kolor tekstu
<u>direction</u>	Kierunek tekstu
<u>letter-spacing</u>	Odstęp między literkami
<u>line-height</u>	Wysokość linii
<u>text-align</u>	Pozycja tekstu (lewo / centrum / prawo)
<u>text-decoration</u>	Dekoracja tekstu
<u>text-indent</u>	Wcięcie dla pierwszej linii tekstu
<u>text-shadow</u>	Cień tekstu
<u>text-transform</u>	Wielkość czcionki (uppercase / lowercase)

https://developer.mozilla.org/en-US/docs/Learn/CSS/Styling_text/Fundamentals

Stylowanie czcionki

<u>font</u>	Ustawienie wielu właściwości od razu
<u>font-family</u>	Rodzaj czcionki (np. Comic Sans)
<u>font-size</u>	Wielkość czcionki
<u>font-style</u>	Styl czcionki
<u>font-weight</u>	Grubość czcionki (light / regular / bold)

Dodanie fontów

ZBIÓR FONTÓW:

<https://fonts.google.com/>

W <head> dodajemy:

```
<link href="https://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">
```

W arkuszu styli:

```
body {  
  font-family: 'Open Sans', sans-serif;  
}
```

Dodanie fontów

ZBIÓR FONTÓW:

<https://fonts.google.com/>

Można również pobrać czcionkę i wgrać ją do folderu assets/fonts a następnie w arkuszu stylu użyć deklaracji **@font-face**:

```
@font-face {  
  font-family: 'myFontName';  
  src: url ('myFontName.eot');  
}
```

Więcej informacji:

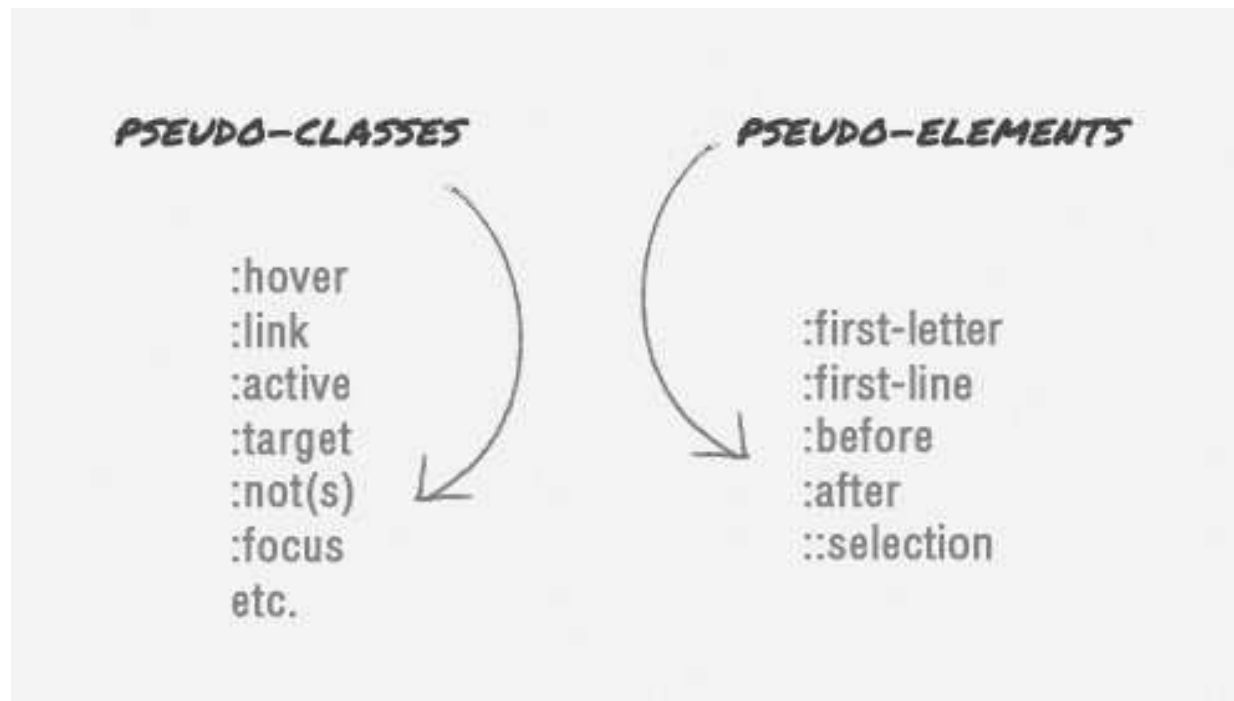
<https://css-tricks.com/snippets/css/using-font-face/>

ZADANIE 9.

Korzystając z google fonts, podłącz czcionkę Open Sans

12.PSEUDO KLASY I ELEMENTY

Pseudoklasy i pseudoelementy



Pseudoklasy i pseudoelementy

Pseudoklasy są selektorami z odpowiednią flagą, uprzedzoną dwukropkiem. Pseudoelementy (np. after) uprzedzamy podwójnym dwukropkiem (dobra praktyka)

Pseudoklasy są bardzo przydatne w konkretnych sytuacjach, np. Zmiany stylu przycisku, po najechaniu na niego myszką

Przykład:

```
button:hover {  
    background: red;  
}
```

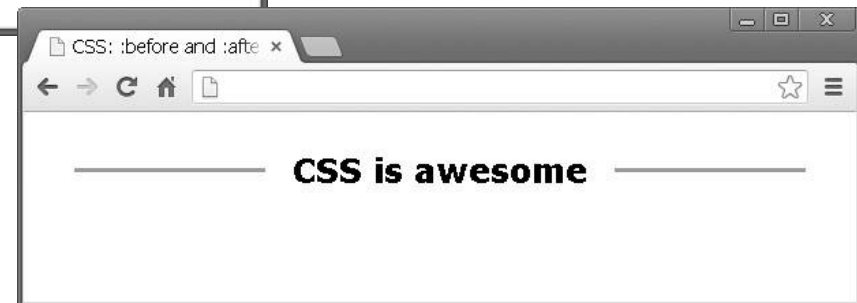
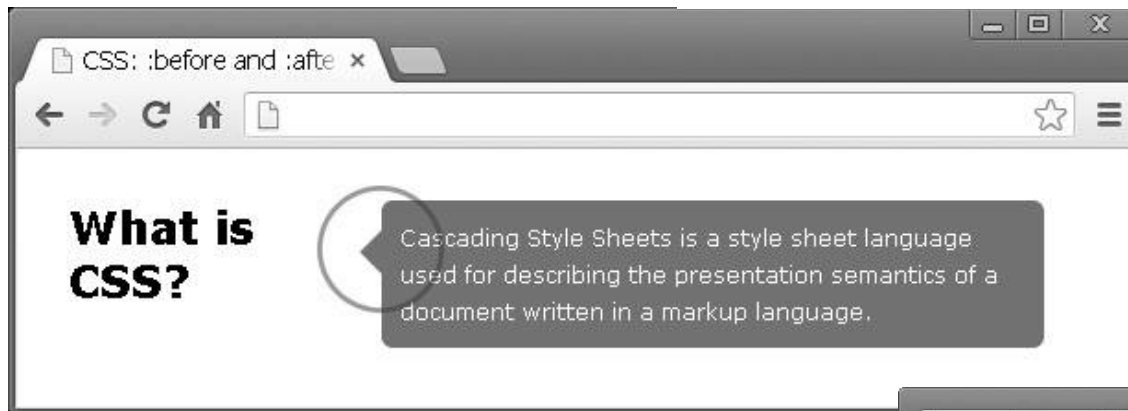
Pseudoklasy i pseudoelementy - spis

Spis wszystkich pseudoklas:

<https://developer.mozilla.org/en/docs/Web/CSS/Pseudo-classes>

PSEUDO-ELEMENTY

BEFORE & AFTER



PSEUDO-ELEMENTY AFTER & BEFORE

Pseudo-elementy BEFORE & AFTER pozwalają nam w łatwy sposób dekorować elementy

Przykład użycia ::after do stworzenia krótkiego podkreślnika nagłówka:

<https://jsfiddle.net/nf0td14q/>

INNE PRZYKŁADY:

<http://red-team-design.com/cool-headings-with-pseudo-elements>

PSEUDOKLASA HOVER

Pseudoklasa hover pozwala nam zmienić styl elementu po najechaniu myszką, lub wyświetlić ukryty element.

Dobłą praktyką jest dodanie do elementu właściwości transition, w celu stworzenia płynnej animacji.

Efekt hover & transition:

<https://jsfiddle.net/0envvf89/>

Pokazanie ukrytego elementu:

<http://jsfiddle.net/UB3UQ/236/>

ZADANIE 10.

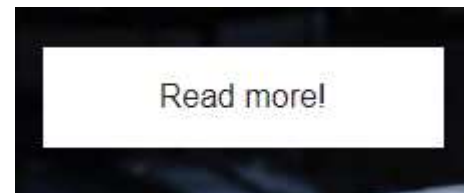
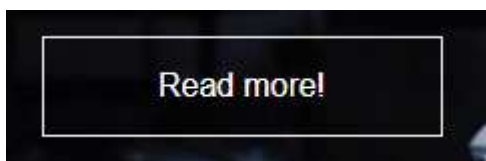
Do każdego nagłówka H2 w pliku index.html, dodaj pseudoelement, spróbuj możliwie najlepiej odwzorować poniższy element:

Latest blog posts

ZADANIE 11.

Spróbuj możliwie najlepiej odwzorować poniższy styl dla buttona „purchase now” (przeźroczyste tło) i dodać do niego pseudoklasę hover:

PO NAJECHANIU MYSZKĄ:



WARTOŚCI WYMIAROWE PRZYCISKU:

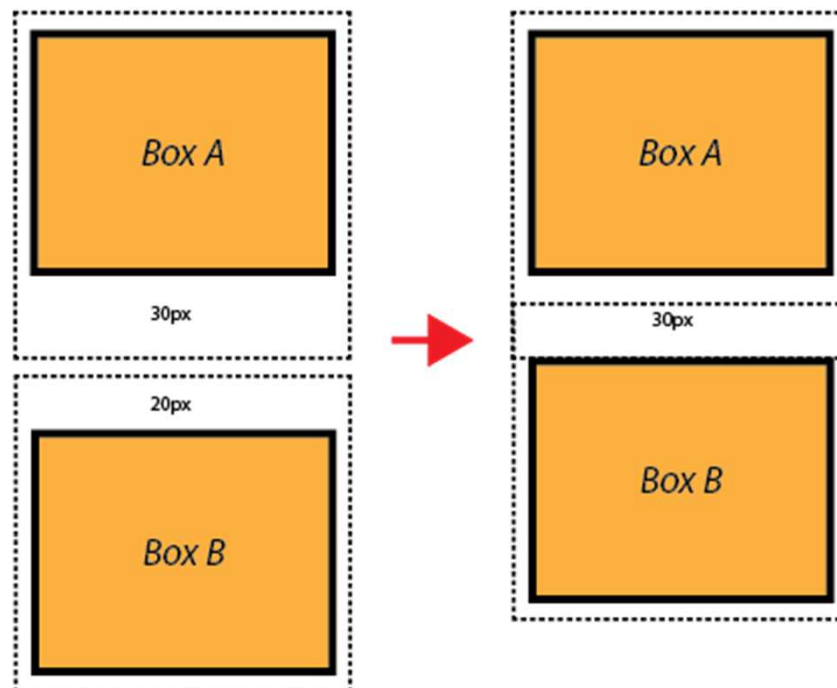
Padding: 15px 30px;

Width: 200px;

13.PROBLEM COLLAPSE MARGIN

Collapse margin problem

Marginesy BOTTOM i TOP nie sumują się:

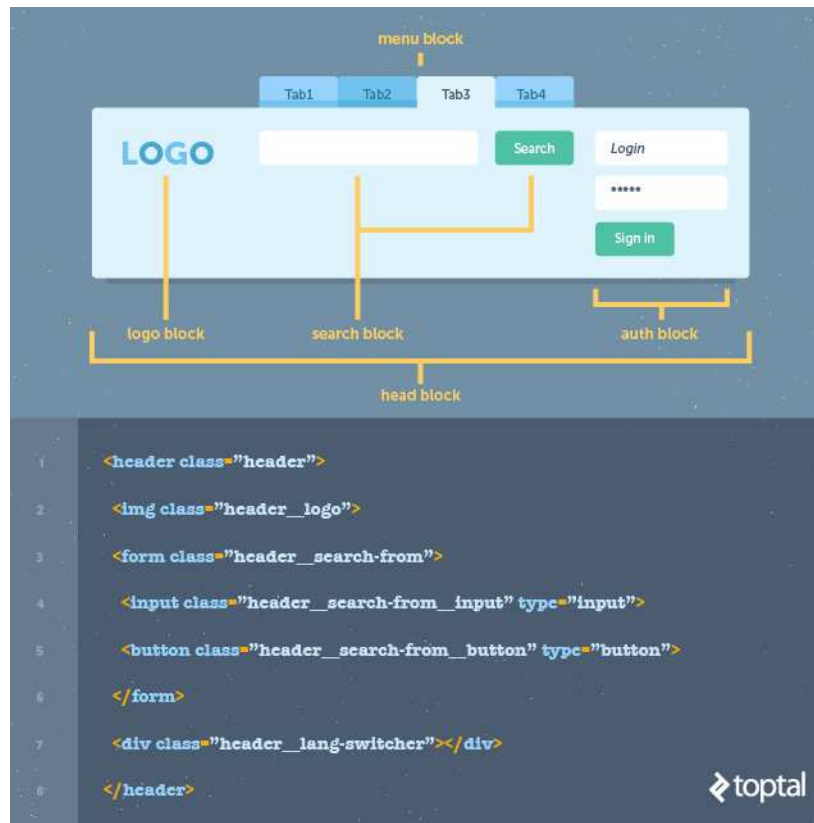


14.METODYKI CSS

Metodyki CSS

- BEM (najbardziej popularna)
<https://nafrontendzie.pl/metodyki-css-2-bem/>
- OOCSS
<https://nafrontendzie.pl/metodyki-css-1-oocss/>
- SMACSS
<https://nafrontendzie.pl/metodyki-css-3-smacss/>
- ACSS
<https://nafrontendzie.pl/metodyki-css-5-atomic-css-acss/>

BEM



TIPS - BEM

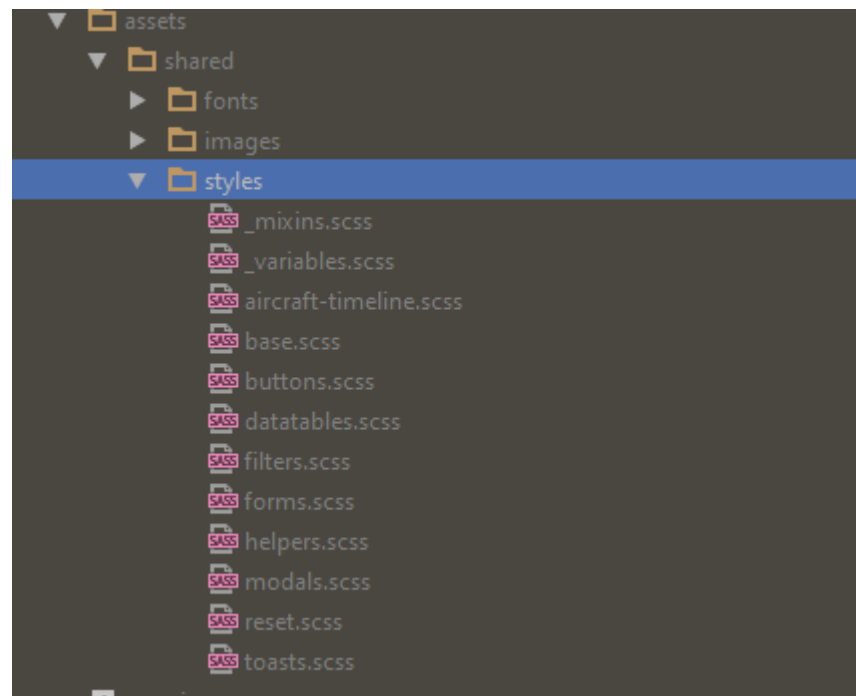
- BLOCK
- ELEMENT
- MODIFY
- Example: list_item-star

15.SKALOWANIE STYLI W APLIKACJI

Stylowanie aplikacji/strony – jak robić to dobrze

Cechy dobrego stylowania:

- Reużywalność
- Przewidywalność
- Utrzymywalność



16.PREFIXES

Prefixes

- Prefixy wspierają support nowych zasad CSS dla poszczególnych przeglądarek
- Większości prefixów nie trzeba już używać
- Warto sprawdzić:
<http://shouldiprefix.com/>
- Istnieją narzędzia, które same dodają za nas prefixy (autoprefixer).
- Pewne frameworki JS do tworzenia aplikacji dodają już za nas prefixy w stylach (np. Angular 4)

Vendor-specific Properties

	Webkit	-webkit-
	Mozilla	-moz-
	Opera	-o-
	IE	-ms-
	Konqueror	-khtml-

```

96  div {
97      -moz-border-radius: 10px;
98      -ms-border-radius: 10px;
99      -o-border-radius: 10px;
100     -webkit-border-radius: 10px;
101     -khtml-border-radius: 10px;
102     border-radius: 10px;
103 }
```

17.KOLORY

Kolory

RGBA Color

Like RGB color definitions, but allows a fourth field, defining the alpha value of the color being applied.

Like opacity, the alpha value is between 0.0 (fully transparent) and 1.0 (fully opaque).

```
div { color: rgb(0,255,0); }
```



```
div { color: rgba(0,255,0,0.5); }
```



RGBA color

Browser Support:



Sunday, July 19, 2009

Kolory

RGB Colors Notation

rgb(255, 0, 0)

RED

GREEN

BLUE

HEX Colors Notation

#FF0000

RED

GREEN

BLUE

Kolory

Preferuje się używanie hexów zamiast nazw kolorów – dzięki temu mamy pewność, że dany kolor będzie wyglądał identycznie w każdej przeglądarce. Hex również łatwiej się czyta, niż RGB.

```
div {  
  color: red;  
}
```

PREFEROWANE:

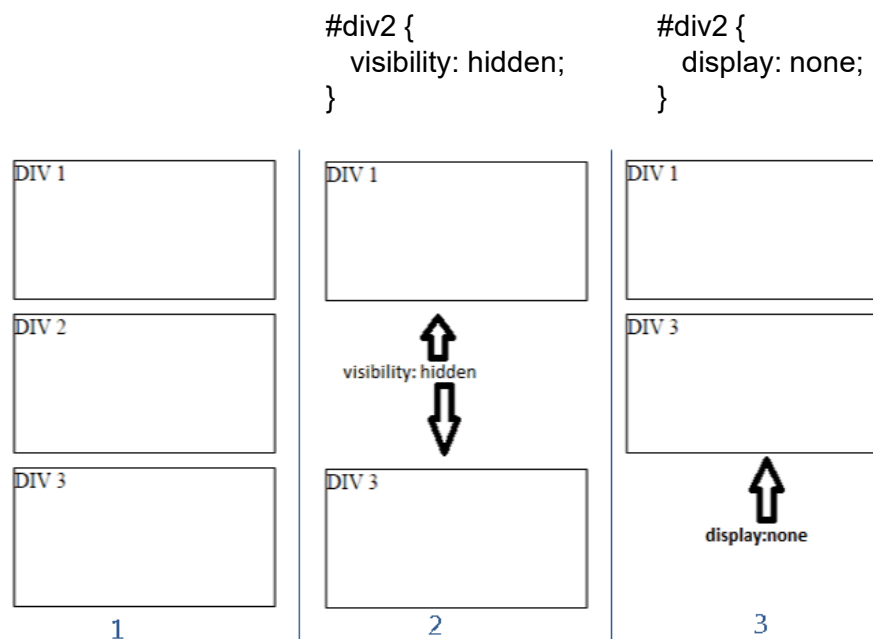
```
div {  
  color: #ff0000;  
}
```

18.UKRYWANIE ELEMENTÓW

Ukrywanie elementów na stronie

display: none; - element zostanie ukryty i **zwolni** swoje miejsce dla pozostałych elementów

visibility: hidden; - element zostanie ukryty **ale nie zwolni** miejsca dla pozostałych elementów



Ukrywanie elementów na stronie – po co?

Ukrywanie elementów ma duże zastosowanie w przygotowaniu wersji strony na urządzenia mobilne. Często poniżej danej rozdzielczości, chcemy ukryć zadany element.

Często również ukrywamy element, a dopiero na daną akcję użytkownika, go pokazujemy, Np. na kliknięcie przycisku.

19.WARTO POZNAĆ

Warto znać:

Stylowanie placeholderów w inputach:

<https://css-tricks.com/almanac/selectors/p/placeholder/>

Box-shadow:

<http://www.cssmatic.com/box-shadow>

Gradients:

<https://css-tricks.com/css3-gradients/>

Combinators:

https://www.w3schools.com/css/css_combinators.asp

Opacity:

https://www.w3schools.com/css/css_image_transparency.asp

20.HARDCORE CSS ;)

CSS dla hardkorów ;)

<https://codepen.io/elad2412/pen/hBaqo>

<http://www.romancortes.com/blog/pure-css-coke-can/>

<https://codepen.io/DawidKrajewski/pen/GgErVO>

<https://codepen.io/waddington/pen/ucCIF>

<https://github.com/tomasznastaly/box-3d-css>

<https://htmlpreview.github.io/?https://github.com/tomasznastaly/box-3d-css/blob/master/box.html>

<http://cssdeck.com/labs/wall-clock-in-pure-css3>

<http://www.hongkiat.com/blog/css-gradient-border/>



THE END

Pytania?

nastalytomasz@gmail.com

tomasz.nastaly @slack