

Wprowadzenie do Aplikacji Internetowych (Webowych)

wykład 3 – CSS 3

dr inż. Grzegorz Rogus



CSS Kaskadowe Arkusze Styli

Czym jest CSS

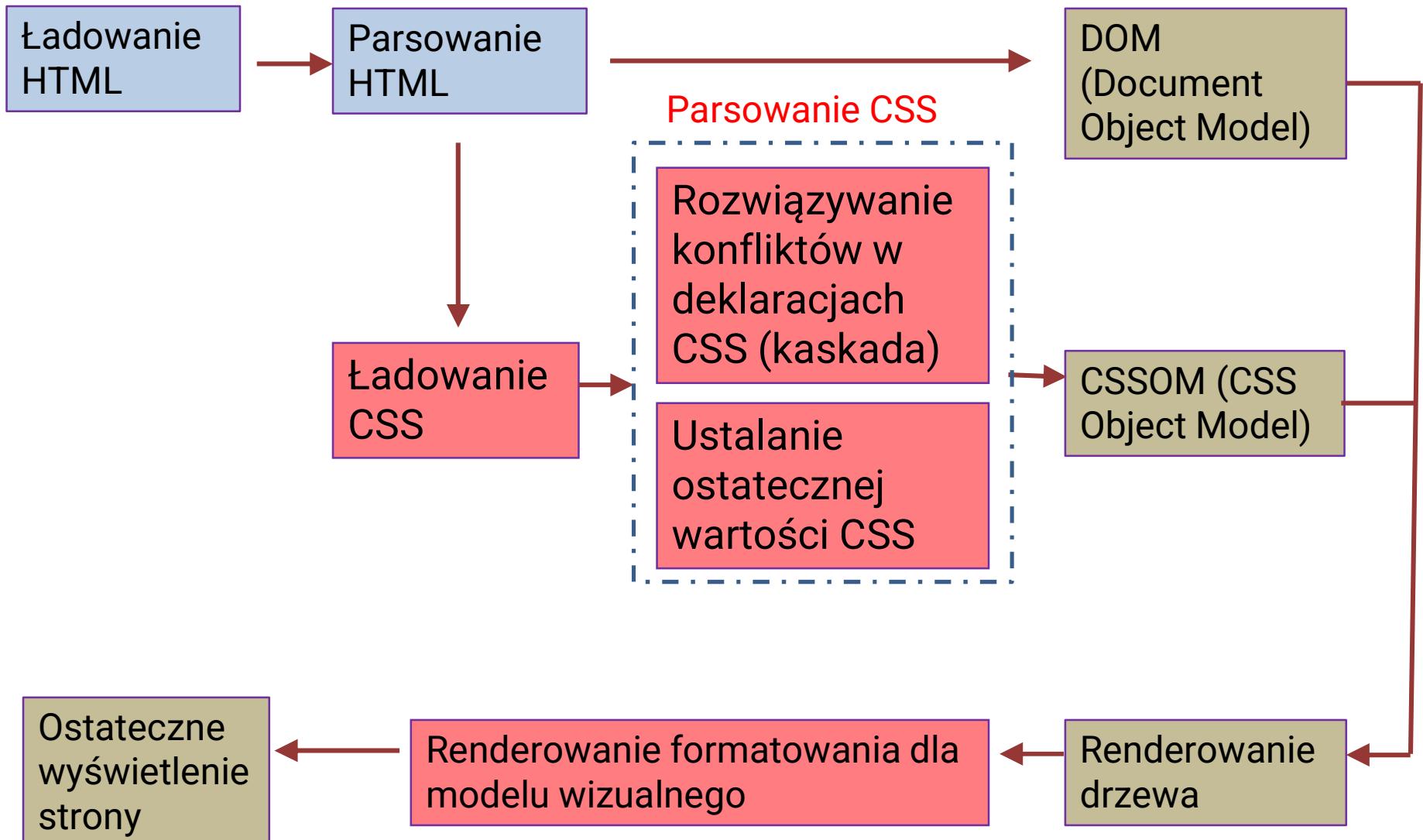
- Odpowiada za układ i wygląd strony
- CSS – Kaskadowe arkusze stylów

Kaskadowe – zawierające hierarchie ważności i definiujące zasady stosowania reguły css do danego elementu

Arkusze – grupowanie reguł w obrębie dokumentów

Styl definiuje wygląd elementów dokumentu HTML, kontrolując ich wizualne cechy w odseparowaniu od kodu HTML

Proces ładowania strony



Jak łączyć?

Zewnętrzny arkusz stylów

Przykład wykorzystania stylów osadzonych, dokument html:

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

  <body>
    .
    .
  </body>
</html>
```

Plik definicji stylów style.css:

```
h1 { color: red }
p { color: navy }
```

By biegły korzystać CSS trzeba...

1. Poznać zasady tworzenie selektorów -> sposobu wskazywania elementów w strukturze HTML

`div.first p` -> **SELEKTOR – GDZIE ZASTOSOWAC**

2. Poznać właściwości i wartości, które możemy użyć
`{ font-size: 20px; }` -> **DEKLARACJA WŁAŚCIWOSCI – CO ZROBIC**
czyli:

`div.first p { font-size: 20px; }` - **REGUŁA (STYL) CSS**

Szybkość i efektywność przyjdą z praktyka

KASKADOWE ARKUSZE STYLÓW

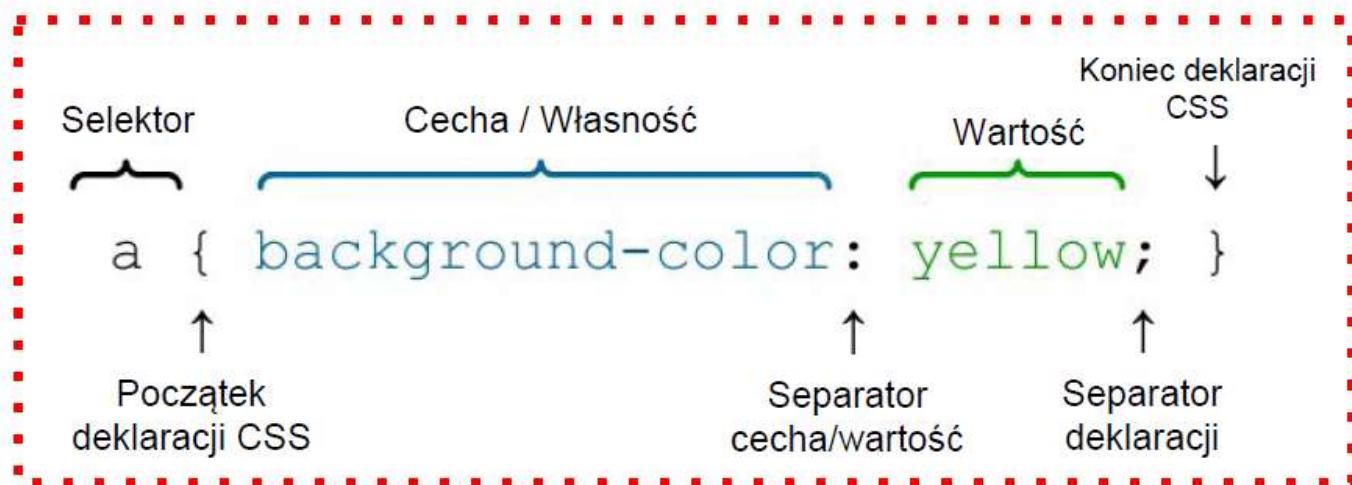
Można w ten sposób opisać wszystkie cechy odpowiedzialne za prezentację elementów dokumentów internetowych, takie jak:

- rodzina czcionek,
- kolor tekstu,
- marginesy,
- odstęp międzywierszowy,
- kolor tła,
- pozycja danego elementu względem innych,
- animacje i przejścia.



CSS3

SKŁADNIA ARKUSZY CSS



Przykładowa deklaracja powoduje ustawienie koloru tła dla wszystkich hiperłączy na kolor żółty.



```
<a href="http://www.onet.pl">onet</a>
```

SKŁADNIA ARKUSZY CSS

selektor { właściwość: wartość }

p {color:red;}

ul.nav-tab {display:block; margin:0px;}

Możliwe jest grupowanie selektorów i deklaracji:

**selektor1, selektor2 {
 właściwość1: wartość1;
 właściwość2: wartość2;
}**

a, a:hover {color:#FA4567; text-decoration:underline }

SKŁADNIA ARKUSZY CSS

Oprócz korzystania bezpośrednio z nazw znaczników HTML, w regułach CSS jako selektorów można używać atrybutów identyfikujących znaczniki HTML.

- Poprzez atrybut **class**:

```
<h1 class="text-primary">Nagłówek</h1>
<p class="text-primary">Podświetlony akapit</p>
.text-primary { color: navy; }
```

- Poprzez atrybut **id**:

```
<header id="site-top"></header>
#site-top { display:block; }
```

WARTOŚCI W ARKUSZACH CSS

Właściwości selektorów mogą przyjmować różne wartości:

1. Tekstowe, np.: block, uppercase, solid;
2. Jednostki miar, np.: 2px, 2em, 1rem, 3pt;
 - px – wielkość w pikselach;
 - pt – wielkość w punktach;
 - % - w procentach;
 - em – wielkość względem wysokości czcionki elementu, w którym znajduje się obiekt;
 - rem – wielkość względem wysokości czcionki zadeklarowanej w HTML.

WŁAŚCIWOŚCI SELEKTORÓW

Właściwości selektorów mogą być zapisywane na kilka sposobów:

- border: 2px dashed #ff0000;
- border-width: 2px;
border-style:solid;
border-color:#000000;
- border-top: 2px dashed #ff0000;
- border-top-color: #d4d4d4;

CSS (Kaskadowe Arkusze Stylu)

Kaskadowe arkusze stylu dzielimy na trzy grupy ze względu na miejsce wystąpienia:

1. **Wbudowane (inline-style)** - są zapisane w miejscu ich działania, tzn. w znaczniku, któremu mają nadawać specyficzne cechy.
2. **Osadzone (embedded-style)** - są zapisane za pomocą zacznika `<style>` w sekcji head dokumentu hipertekstowego.
3. **Dołączone (linked-style)** - są zapisane w osobnych plikach o rozszerzeniu .css.

Która z metod zapisywania CSS ma największy priorytet ?

KASKADOWOŚĆ

obowiązujący jest styl położony najbliżej elementu

Kaskadowość – metoda rozwiązywania problem redundancji reguł

- Proces łączenia różnych arkuszy stylów i rozwiązywania konflikty między różnymi regułami CSS i deklaracjami, kiedy więcej niż jedna reguła ma zastosowanie do określonego elementu.

ISTOTNOŚĆ

priorytet
według źródeł
stylów

>

SPECYFICZNOŚĆ

>

KOLEJNOŚĆ

Kaskadowość w obrębie dokumentu

CSS (Kaskadowe Akusze Stylu)

priorytet według źródeł stylów

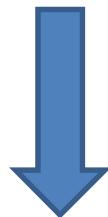


Uwaga na **!important** - łamie zasady kaskadowości

`a {background-color: white !important;}`

Reguły Kaskady

ISTOTNOŚĆ



Ten sam dokument

SPECYFICZNOŚĆ



Ten sam typ selektora

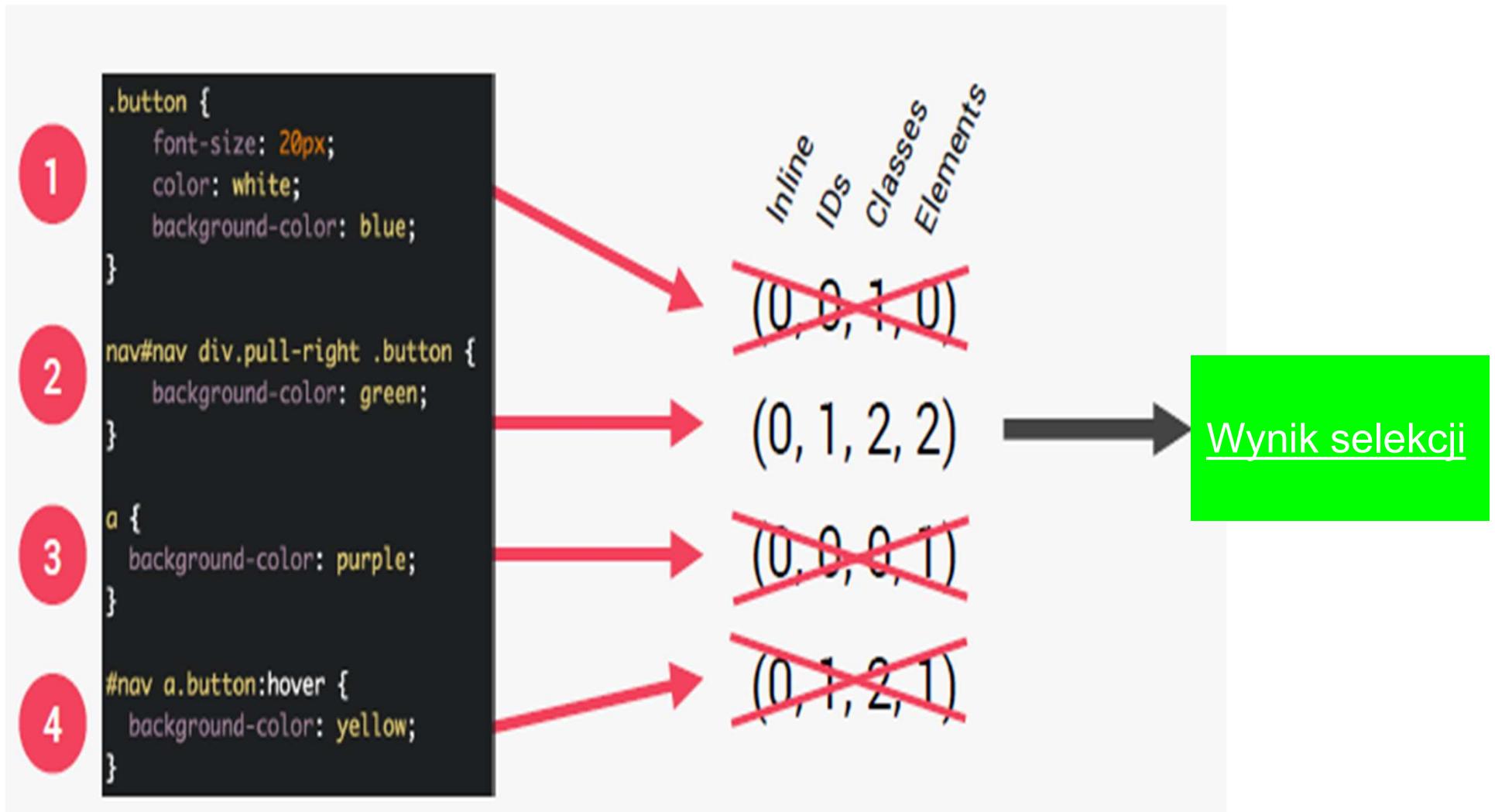
KOLEJNOŚĆ

1. deklaracja **!important**
2. Lokalne style inline
3. Wewnętrzny arkusz stylów
4. Importowane style do wewnętrznego arkusza CSS
5. Deklaracja domyślna przeglądarki

A	id selectors	#foo
B	class selectors	bar
	attribute selectors	[title], [colspan="2"]
	pseudo-classes	:hover, :nth-child(2)
C	type selectors	div, li
	pseudo-elements	::before, ::first-letter

Ostatnia deklaracja w kodzie będzie przesłaniała wszystkie inne deklaracje i to ona zostanie zastosowany.

Kaskadowość w obrębie dokumentu



Selektor zawierający 1 identyfikator jest bardziej szczegółowy niż selektor zawierający 1000 klas;
Selektor zawierający 1 klasę jest bardziej szczegółowy niż selektor zawierający 1000 elementów;
Selektor uniwersalny * nie ma wartości specyficzności (0, 0, 0, 0);

SELEKTORY CSS – IDENTYFIKACJA

***** - spełniony przez każdy element

E - spełniony przez każdy element E

```
body { font-family:Verdana; font-size:11px; }
```

E.value - spełniony przez te elementy E, które posiadają atrybut CLASS o wartości value

```
p.title1 { font-size:18px; }
```

E#value - spełniony przez te elementy E, które posiadają atrybut ID o wartości value

```
img#news { border:2px double #FFFFFF; }
```

SELEKTORY kontekstowe CSS – ZAGNIEŻDŻANIE (selektory potomków)

E F - spełniony przez każdy element F zagnieżdżony wewnątrz elementu E

```
p b { color:#FFFFFF; }
```

E > F - spełniony przez każdy element F zagnieżdżony bezpośrednio w E

```
ul > li { font-family:Verdana; font-size:11px; }
```

E + F - spełniony przez każdy element F, który następuje bezpośrednio za elementem E

```
i + b { color:#000000; }
```

Wykorzystanie selektorów potomków (selektory kontekstowe)

```
div ul { color: blue }

<div>
  <p>Moje ulubione sporty to:</p>
  <ul>
    <li>narciarstwo,</li>
    <li>kolarstwo,</li>
    <li>pływanie.</li>
  </ul>
</div>

<p>Moje ulubione sporty to:</p>
<ul>
  <li>narciarstwo,</li>
  <li>kolarstwo,</li>
  <li>pływanie.</li>
</ul>
```

Moje ulubione sporty to:

- narciarstwo,
- kolarstwo,
- pływanie.

Moje ulubione sporty to:

- narciarstwo,
- kolarstwo,
- pływanie.

```
div.figure p
{
  text-align: center;
  font-size: smaller;
  text-indent: 0;
}

<div class="figure">
  <p></p>
  <p>Kask Breeze firmy Mango</p>
</div>
```



Kask Breeze firmy Mango

Wykorzystanie selektorów potomków (selektory kontekstowe)

Wykorzystanie selektorów „dzieci”

```
body > p { color: blue }

<body>
  <p>To jest dziecko body</p>
  <div>
    <p>To jest potomek body</p>
  </div>
</body>
```

To jest dziecko body

To jest potomek body

Wykorzystanie selektorów „braci”

```
p + p
{
  text-indent: 0.7em;
  margin-top : 0
}

<p>Pierwszy paragraf w tym tekście nie musi
posiadać wcięcia akapitowego</p>

<p>Drugi paragraf w tym tekście juz powinien
takie wcięcie posiadać</p>

</p>

<p>Za wyjątkiem tych paragrafów które następują po
rysunkach</p>
```

Pierwszy paragraf w tym tekście nie musi posiadać wcięcia akapitowego

Drugi paragraf w tym tekście juz powinien takie wcięcie posiadać



Za wyjątkiem tych paragrafów które następują po rysunkach

Tytuł

SELEKTORY CSS – PSEUDOKLASY

A:link - spełniony przez każdy link, który nie został jeszcze odwiedzony

```
a:link { text-decoration:none; }
```

A:visited - spełniony przez każdy link, który został odwiedzony

```
a:visited { text-decoration:underline; }
```

E:hover - spełniony przez każdy element E, nad którym właśnie znajduje się wskaźnik

```
tr:hover { text-decoration:underline; }
```

E:focus - spełniony przez każdy element E, na którym właśnie znajduje się focus dokumentu

```
input:focus { border-radius:3px; }
```

DEMO -> linki.html

SELEKTORY CSS – KOLEJNOŚĆ

E:first-child – spełniony przez pierwszy element E

```
ul li:first-child {color:#f46c00;}
```

E:last-child – spełniony przez ostatni element E

```
ul li:last-child {color:#000000;}
```

E:nth-child(n) – spełniony przez n-ty element E

```
ul li:nth-child(2) {text-decoration:underline;}
```

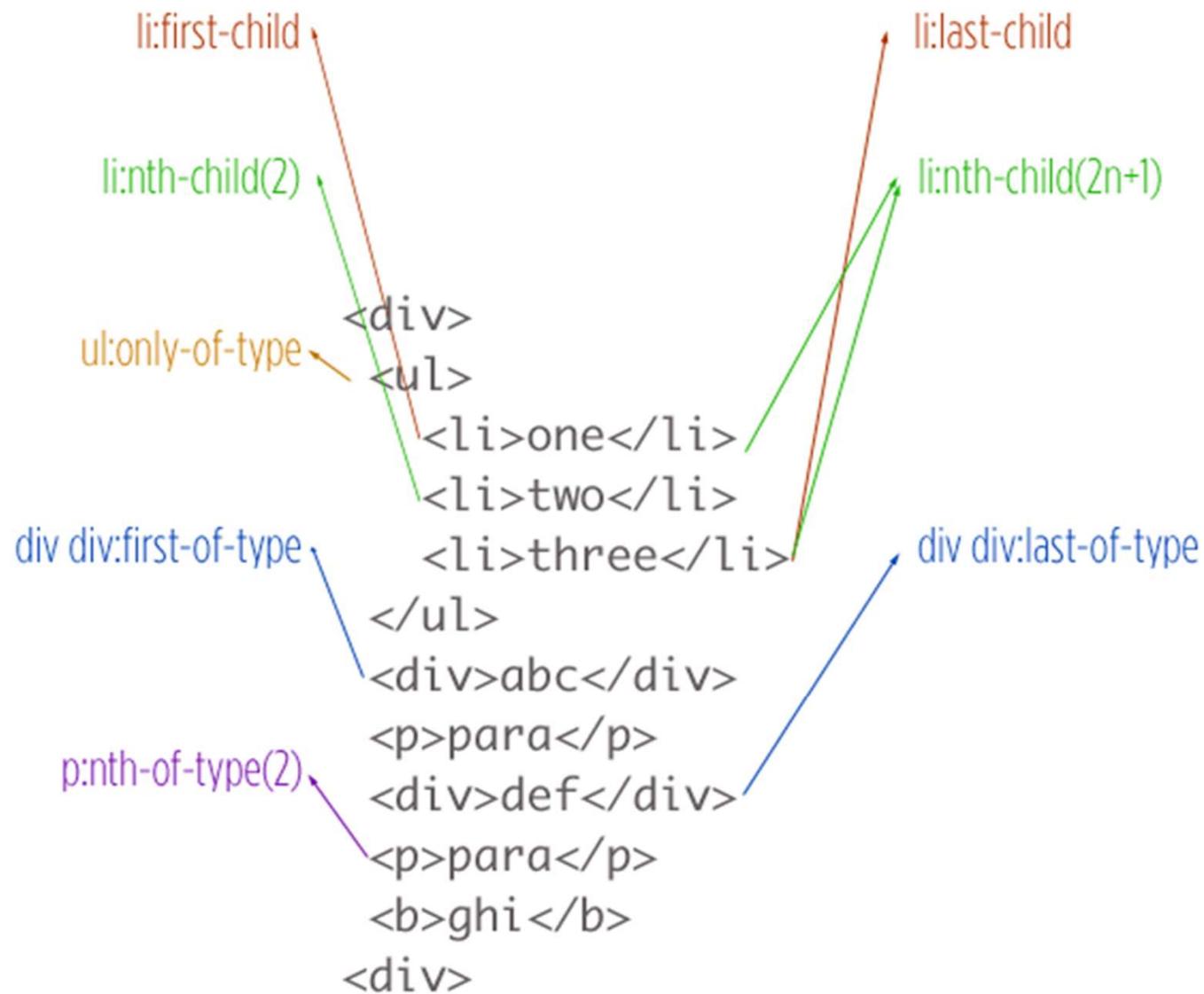
E:nth-child(even) – spełniony przez wszystkie nieparzyste elementy E

```
ul li:nth-child(even) {text-decoration:underline;}
```

E:nth-child(odd) – spełniony przez wszystkie parzyste elementy E

```
ul li:nth-child(odd) {text-decoration:none;}
```

SELEKTORY CSS - PSEUDOELEMENTY



SELEKTORY CSS - PSEUDOELEMENTY

E::after - wstawia coś po zawartości elementu E

```
p::after {content: url(smiley.gif) }
```

::before - wstawia coś przed zawartością elementu E

```
p::before {content: url(smiley.gif) }
```

E::first-letter - spełniony przez pierwszą literę elementu E

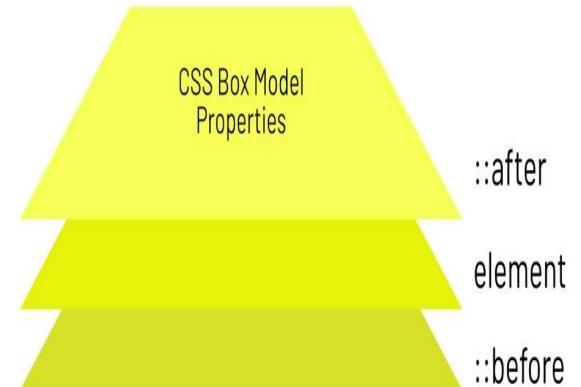
```
p::first-letter {font-size:2em; }
```

E::first-line - spełniony przez pierwszą linię elementu E

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

E::selection - spełniony przez część obiektu E wybranego przez użytkownika

```
p::selection {background-color:#d4d4d4; }
```



Zmienne w CSS – bez użycia preprocesora

Zmienne w CSS są definiowane za pomocą specjalnej notacji:

```
:root {  
    --primary-color: yellow;  
}
```

Dostęp do zmiennych jest możliwy za pomocą funkcji var() :

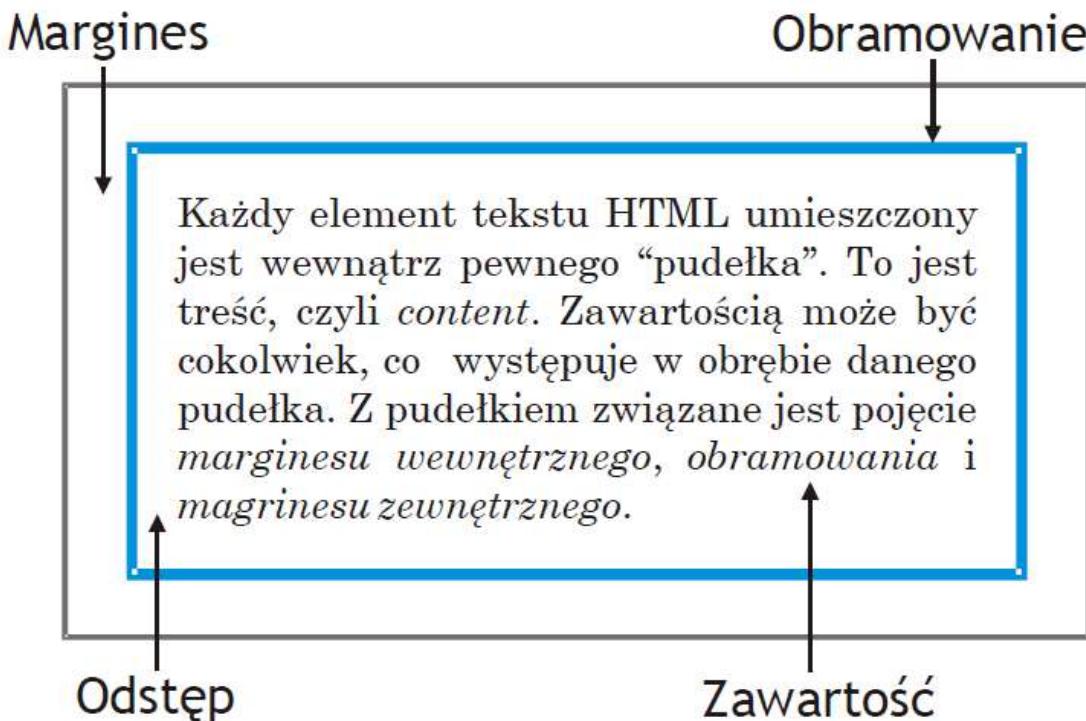
```
p {  
    color: var(--primary-color)  
}
```

Wartością zmiennych może być dowolna wartość CSS deklarowana w dowolnym selektorze:

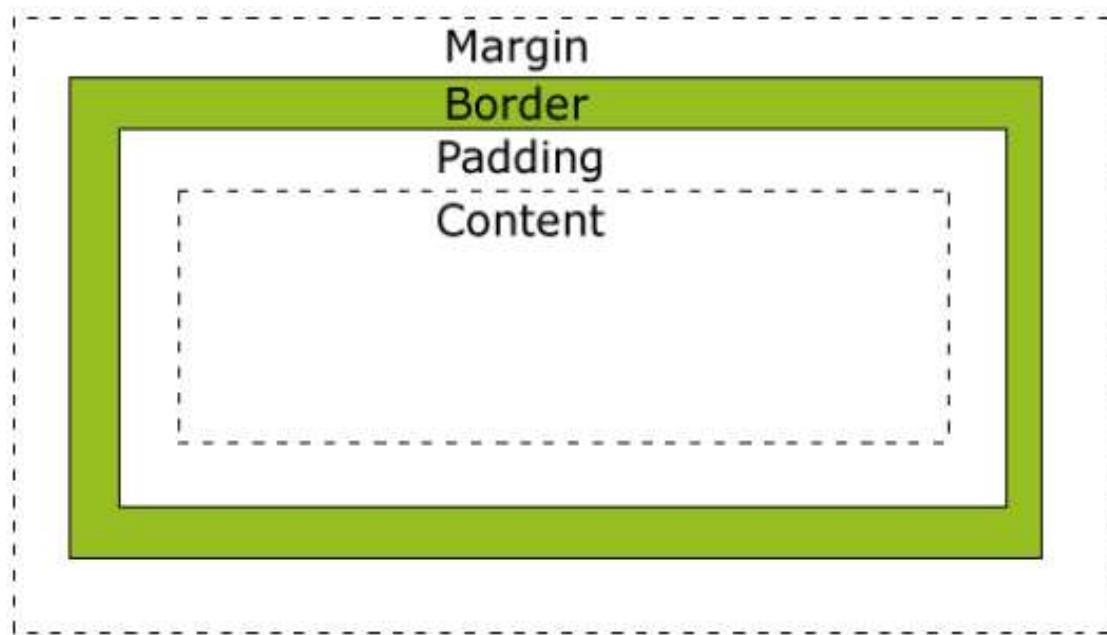
```
:header {  
    --default-padding: 30px 30px 20px 20px;  
    --default-color: red;  
    --default-background: #fff;  
}
```

BOX MODEL

Koncepcja modelu pojemnika zakłada, że każdy element dokumentu HTML może być traktowany jako prostokątny obszar, którego zawartość otoczona jest *marginesem wewnętrzny (odstępem), obramowaniem i marginesem zewnętrznym*.



BOX MODEL – całkowita szerokość



http://www.w3schools.com/css/css_boxmodel.asp

Oznacza to, że rzeczywista wielkość danego obiektu jest sumą wielkości zawartości, paddingu, ramki oraz marginesu.

Szerokość = szerokość elementu + lewy odstęp + prawy odstęp + lewa ramka + prawa ramka + lewy margines + prawy margines

BOX MODEL

Przykład wykorzystania modelu pojemnika

```
h1
{
    background-color: skyblue;
}

...
<h1>Model pojemnika — box model</h1>
```

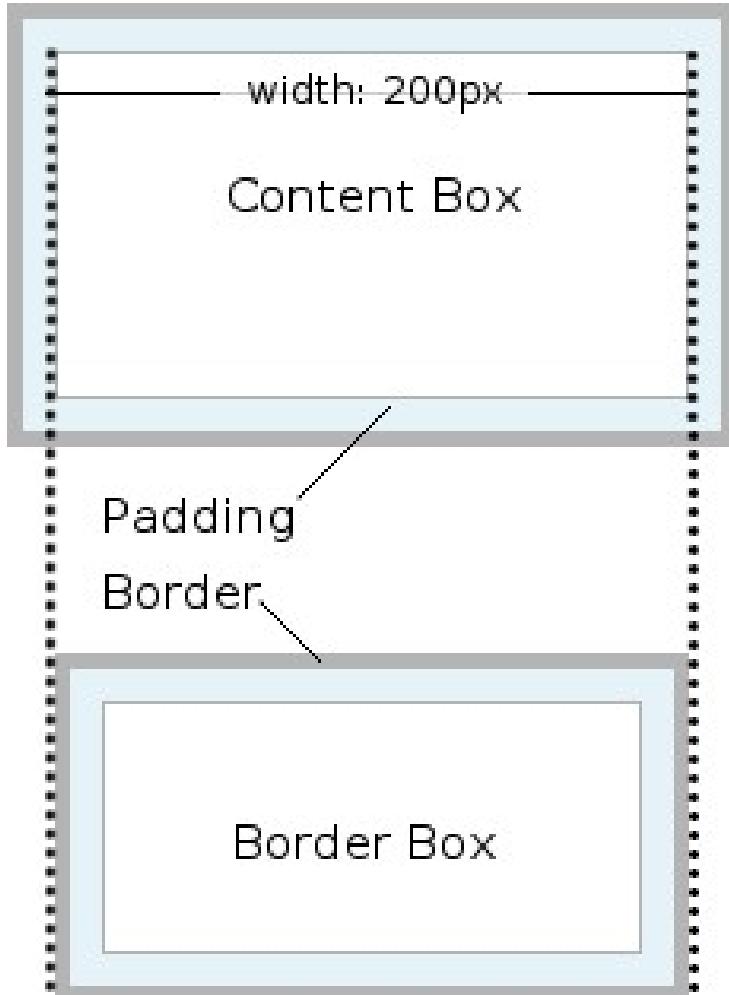
Model pojemnika — box model

```
h1
{
    background-color: skyblue;
    padding-left: 20px;
    padding-top: 15px;
    padding-bottom: 15px;
    padding-right: 20px;
}

...
<h1>Model pojemnika — box model</h1>
```

Model pojemnika — box model

box-sizing – inny model szerokości



Czym jest podana szerokość: ????

Całkowita (pułka) = 234px;
 $200+2\times 5+2\times 10+2\times 2$

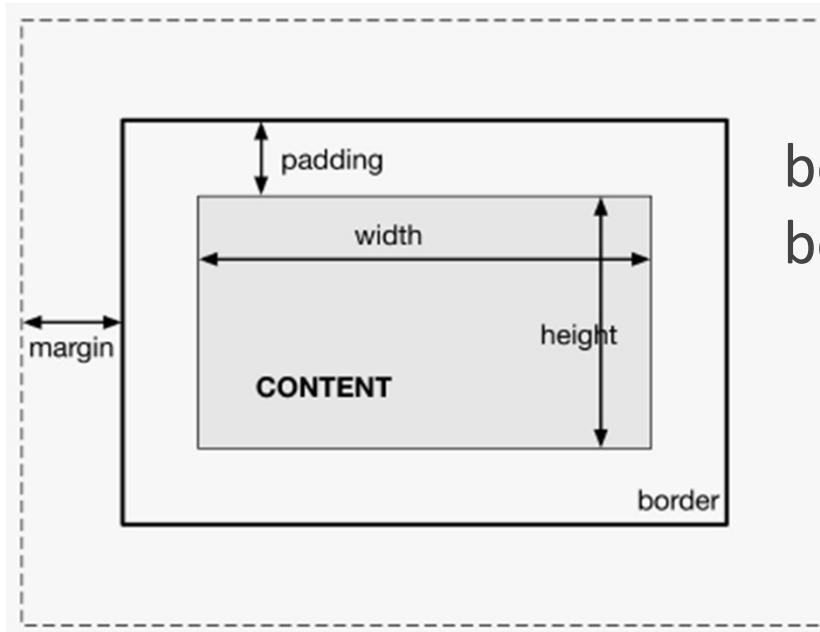
```
p {  
    margin: 10px;  
    padding: 5px;  
    width: 200px;  
    border-width: 2px;  
}
```

CSS3 wprowadza box-sizing

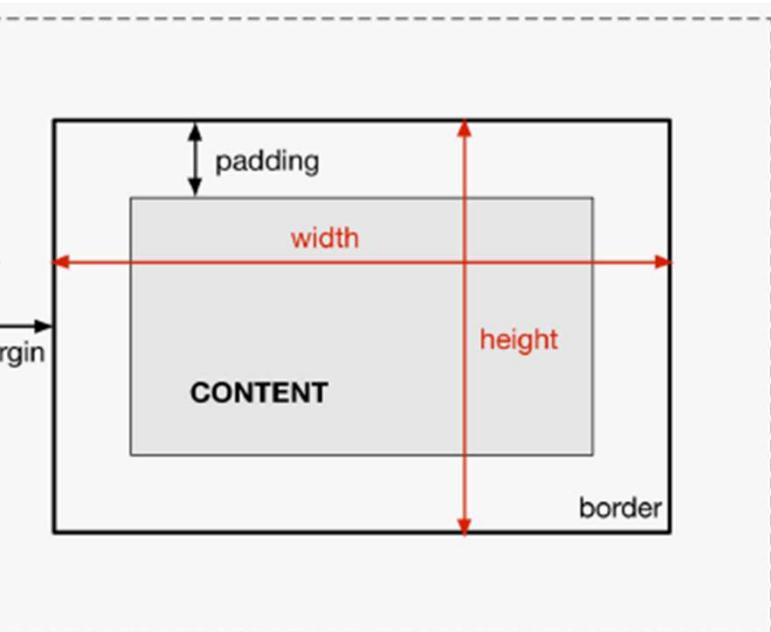
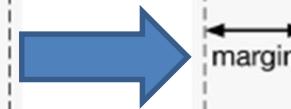
- content-box (szerokość = zawartość)
- border-box (szerokość = zawartość + obramowanie + odstęp)

Całkowita (pułka) = 220px;
 $200+2\times 10$

box-sizing – inny model szerokości cd.



box-sizing:
border-box



width = right border + right padding +
content width + left padding + left
border

height = top border + top padding +
content height + bottom padding +
bottom border

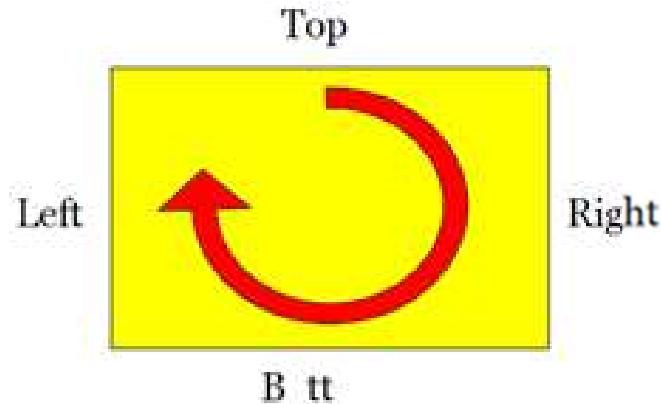
width = right border + right padding +
content width + left padding + left
border

height = top border + top padding +
content height + bottom padding +
bottom border

ZASADA WSKAZÓWEK ZEGARA

Jak zapamiętać przypisanie wartości do określonych boków pojemnika?

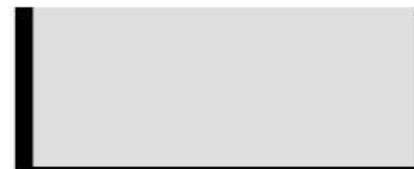
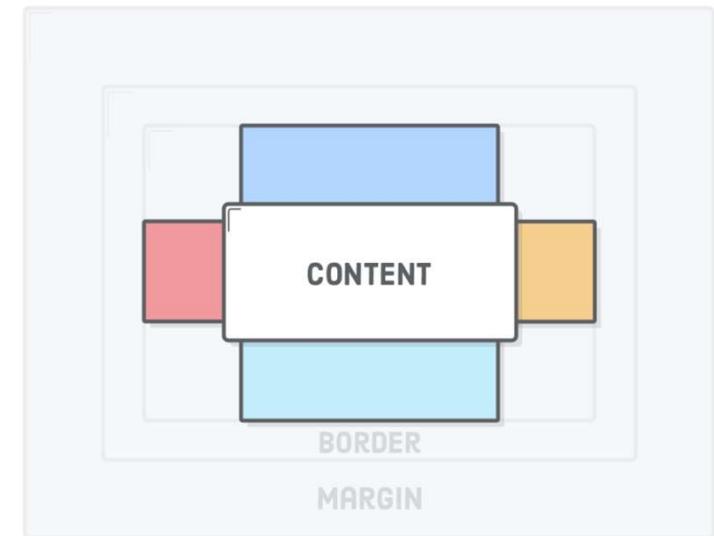
- ▶ *Reguła stopera lub zegara wskazującego 12-stkę,*
- ▶ *Reguła trouble – TopRightBottomLeft.*



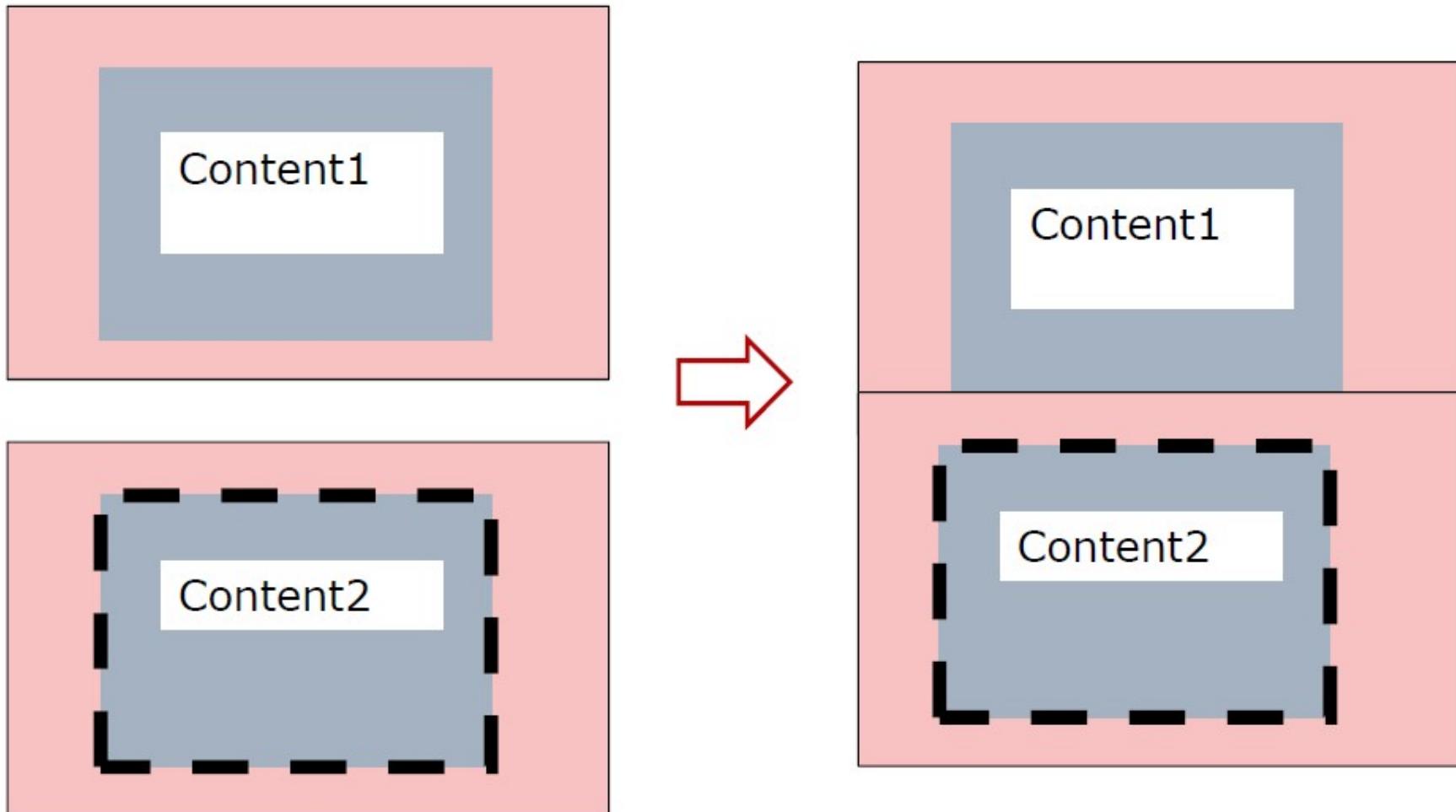
border-width: 6px;

border-width: 0px 3px 2px 10px;

border-width: 3px 0px;

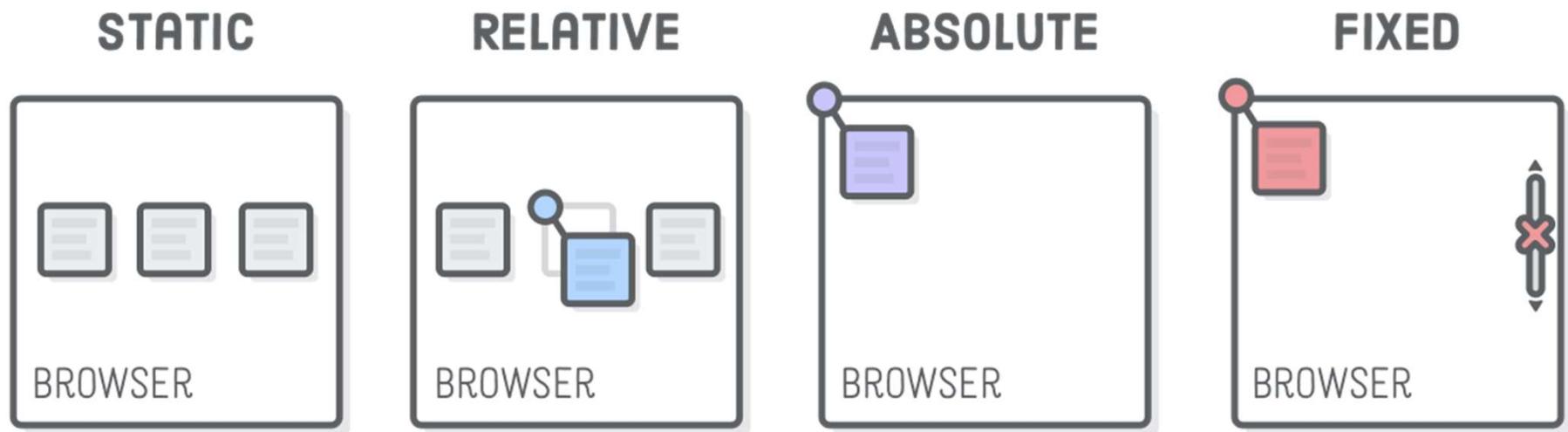


Zapadanie marginesów



- ▶ Jeżeli dwa, lub więcej, pojemniki sąsiadują obok siebie pionowo to niezależnie od ustalonych dla nich marginesów pionowych, odstęp pomiędzy nimi będzie równy *większej z ustalonych wartości*.

Pozycjonowanie elementów na stronie



Pozycjonowanie elementów na stronie

Pozycjonowanie służy w CSS do ustalania, względem czego układają się elementy. Występują przeważnie z właściwościami *top*, *bottom*, *left* i *right*.

Istnieją 4 rodzaje pozycjonowania:

- **position: static** – elementy układają się w kolejności ich umieszczenia, nie nakładają się na siebie;
- **position: fixed** - elementy układają się względem okna przeglądarki, np.:

```
div { position:fixed; top:20px; right:30px; }
```

ustawi element 20px od góry i 30px od prawej strony okna.

Pozycjonowanie elementów na stronie

- **position: relative** – element układa się względem elementu poprzedniego, np.:

```
div { position:relative; left:30px; }
```

ustawi element w odległości 30px od elementu poprzedniego.

- **position: absolute** – element układa się względem elementu, w którym jest zamknięty i który nie jest opisany jako position:static.

```
div { position:absolute; left:30px; }
```

ustawi element w odległości 30px od krawędzi elementu nadziędnego.

POSITION I PRZEPŁYW DOKUMENTU

Obiekty **static** i **relative** uczestniczą w tworzeniu przepływu dokumentu.

Obiekty **fixed** i **absolute** są wyrwane z przepływu dokumentu i przez to nie wpływają na wysokość elementów, w których się zawierają.

Aby to obejść bardzo często korzysta się z par obiektów o pozycjonowaniu relative i absolute.

Obiekty „relative” tworzą kontenery dla obiektów „absolute”, dzięki czemu można precyzyjnie rozmieścić elementy na stronie, a jednocześnie zachować prawidłowy przepływ.

PRZEPŁYW DOKUMENTU

Elementy HTML w dokumencie układają się na kilka sposobów:

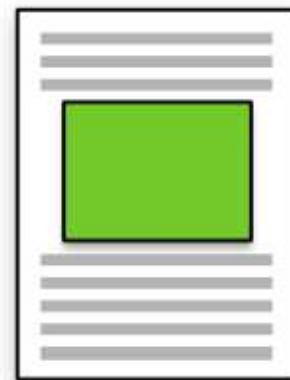
Normalny przepływ treści (ang. normal flow)

Elementy układają się kolejno jeden pod drugim.

Obecność elementu w dokumencie odsuwa inne elementy, tak że żadne na siebie się nie nakładają.

To jest domyślne zachowanie w CSS i określone jest pozycją statyczną:

position: static;



PRZEPŁYW DOKUMENTU

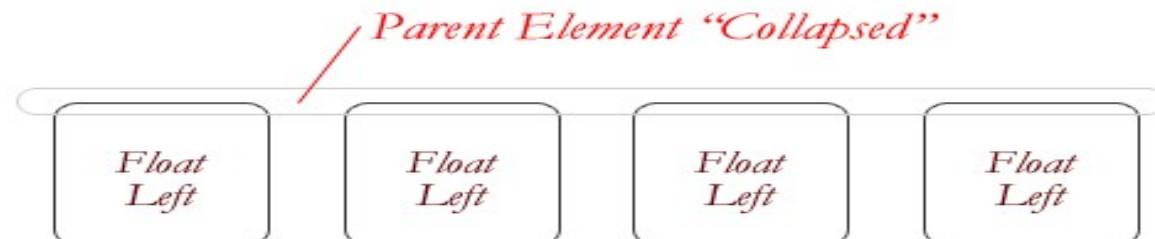
Wyjęcie z przepływu treści

Obiekt przestaje istnieć w swoim oryginalnym miejscu w dokumencie i pozostałe elementy są rozstawiane, jakby tego obiektu nie było.

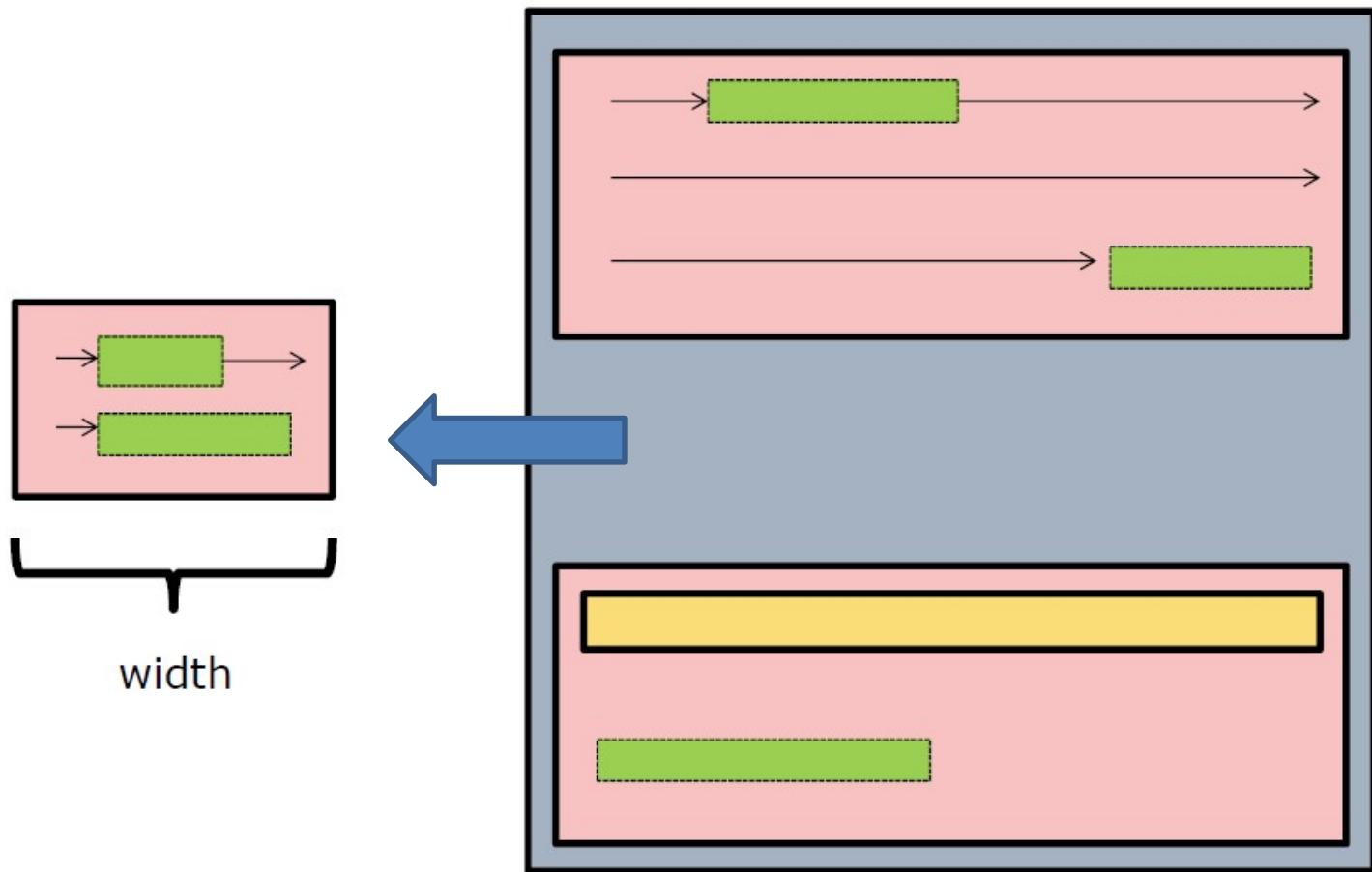
Obiekty wyjęte z biegu dokumentu najczęściej umieszczane są w innym miejscu strony przez **pozycjonowanie absolutne** lub **float**.



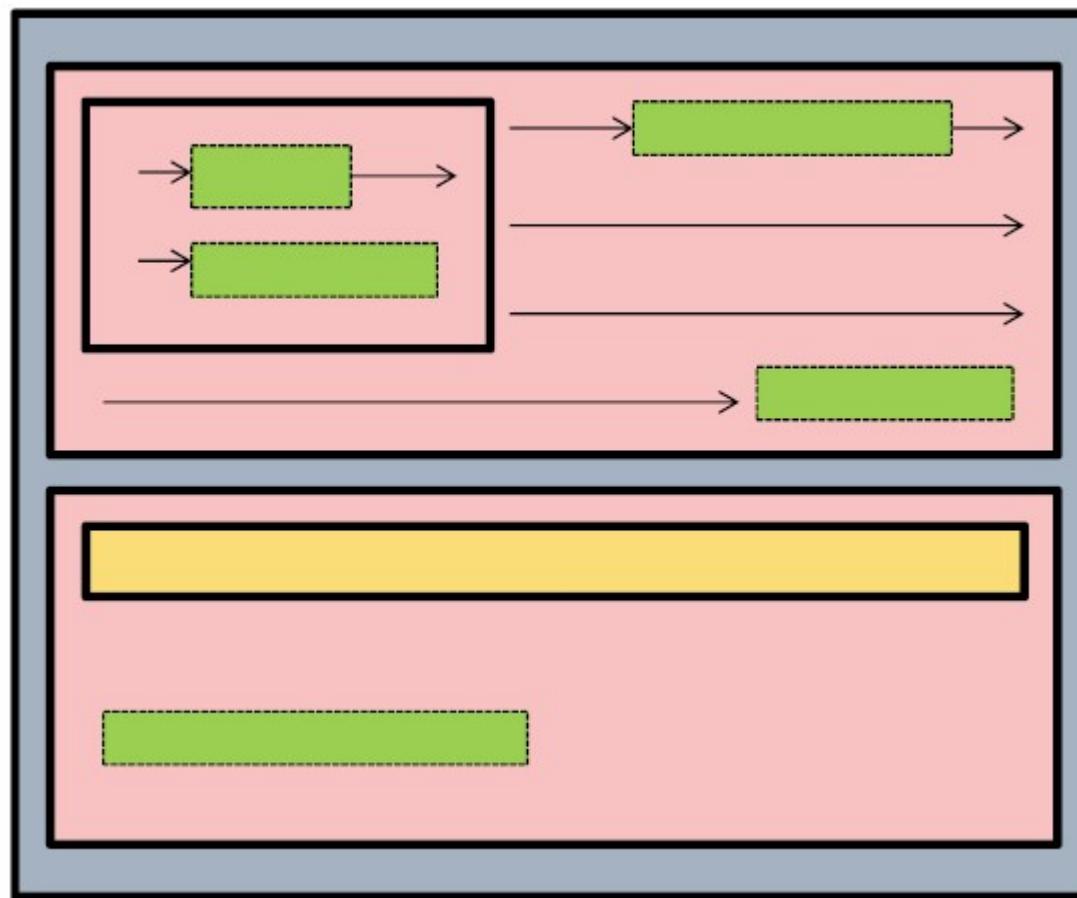
Jeżeli wszystkie dzieci danego elementu są wyjęte z biegu dokumentu, to element będzie miał zerową wysokość, ponieważ nie będzie rezerwował miejsca na żaden element w nim.



Floating – wyjście z normalnego przepływu

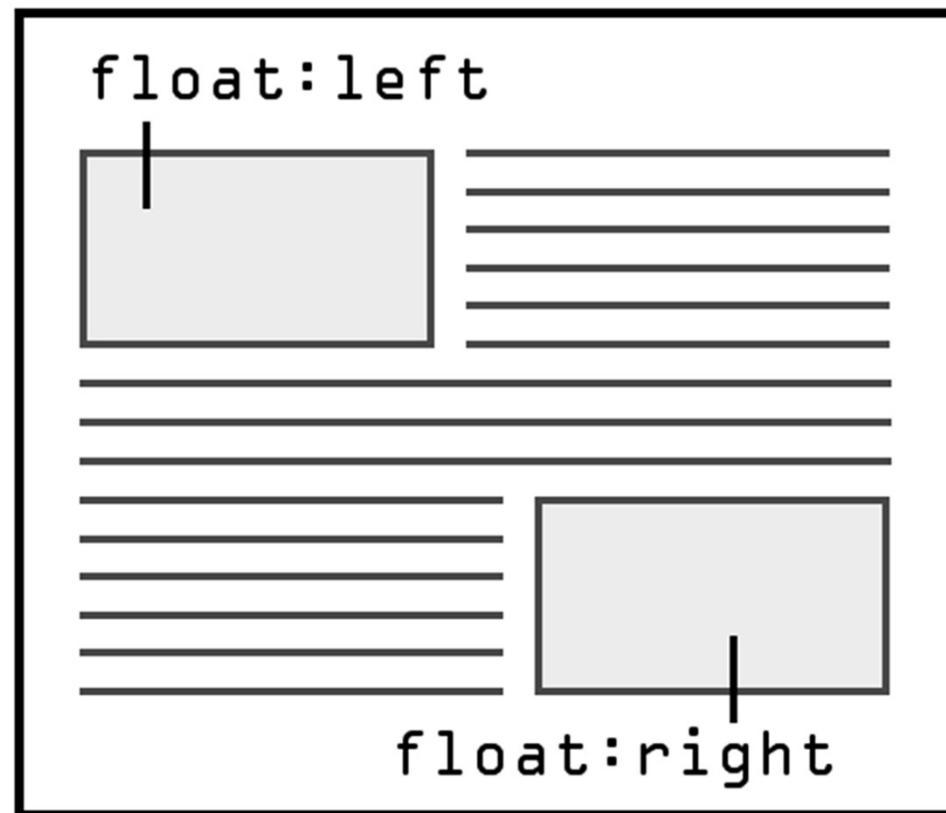


Przepływ dokumentu



FLOAT

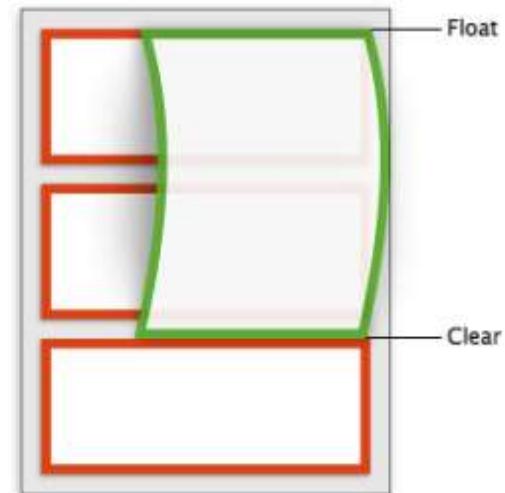
- **left** - dla umieszczenia elementu z lewej strony,
- **right** - dla umieszczenia elementu z prawej strony,
- **none** - oznacza, że element nie będzie pływający. To jest domyślna wartość dla wszystkich elementów i zazwyczaj nie trzeba jej nadawać.



ZAPOBIEGANIE OPŁYWANIU (CLEAR)

Przerywanie przez clear

Ponieważ obiekty z float są wyjęte z normalnego biegu dokumentu, mogą rozciągać się nad kilkoma innymi — np. ilustracja może rozciągać się z boku kilku akapitów.



Aby uniknąć tego efektu należy jakiśmuś elementowi za elementem z float nadać właściwość clear — powoduje ona ponowne „złączenie” float z biegiem dokumentu.

Overflow

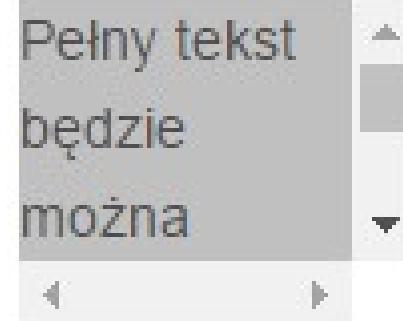
Za pomocą właściwości overflow możemy ustalić jak ma zachować się element HTML w momencie gdy jego zawartość nie będzie mieściła się w jego rozmiarach.

- visible - domyślne
- hidden - zakrywanie nie mieszczącej się zawartości w zadeklarowanych ramach
- scroll - przewijanie
- auto - w razie konieczności pojawią się przewijanie

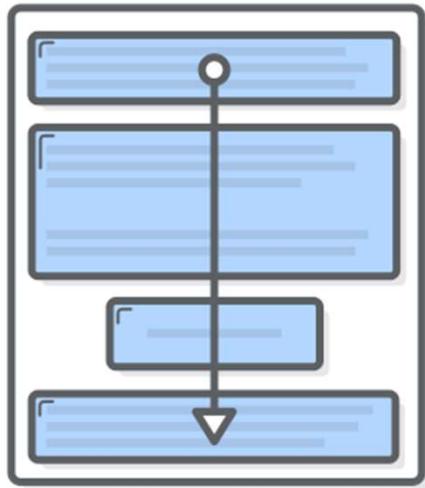
Nadwiar tekstu
w tym bloku
zostanie
przycięty

Cały tekst jest
widoczny, bez
przycinania i
konieczności
pojawienia się
przewijania.

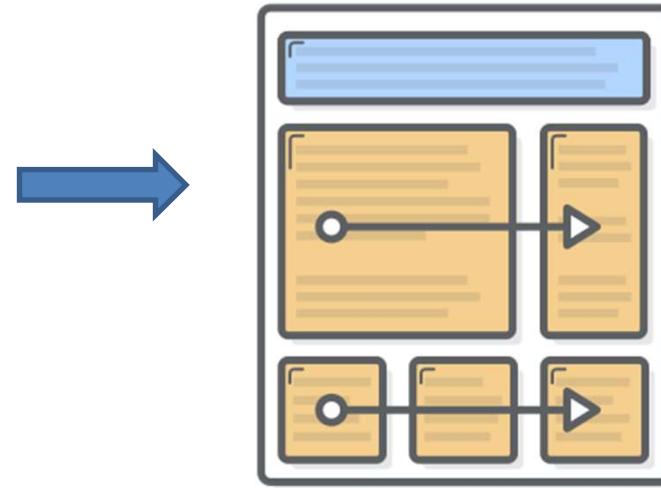
Pełny tekst
będzie
można



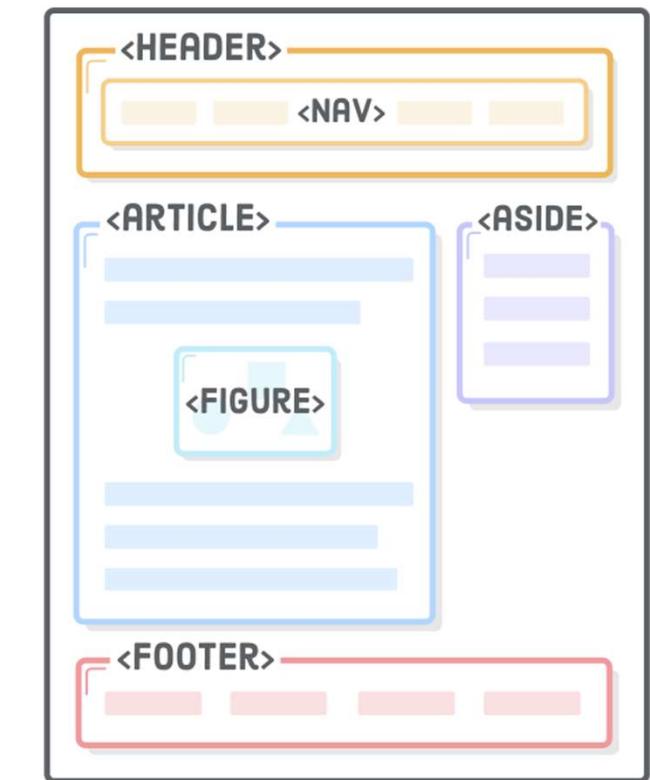
Tworzenie Layout (Układ strony)



VERTICAL FLOW

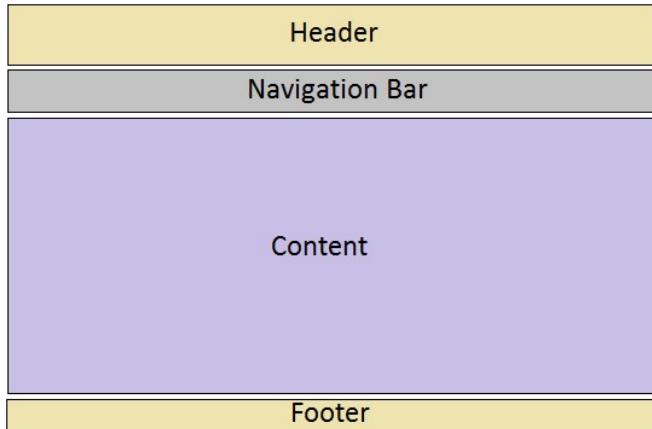


HORIZONTAL FLOW

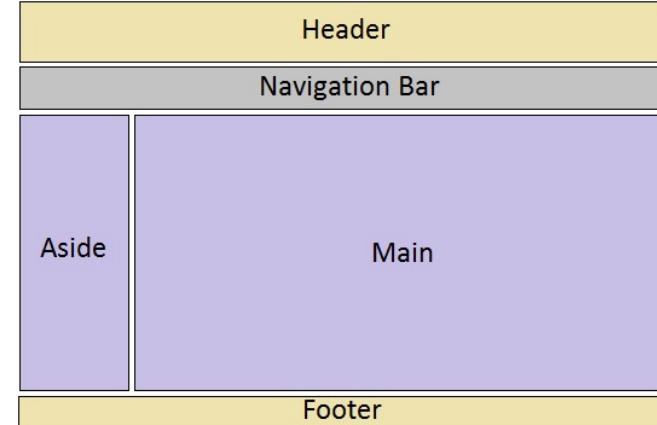


Najpopularniejsze szablony

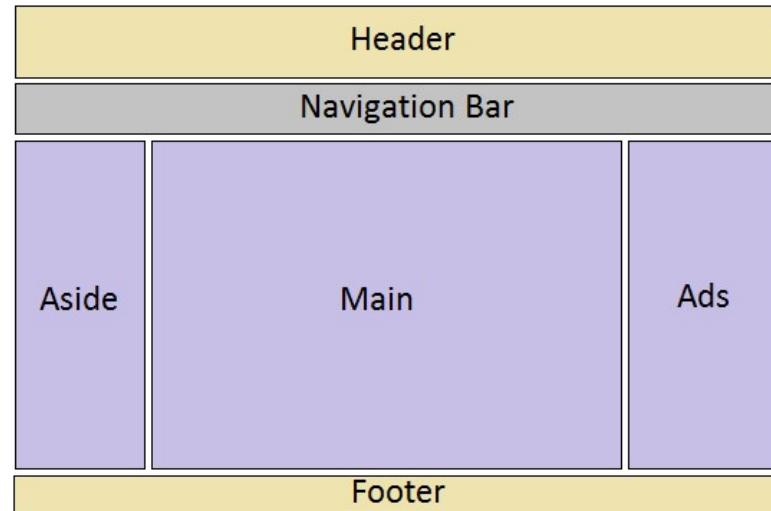
➤ One-column layout



➤ Two-column layout



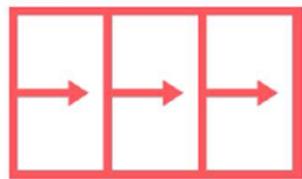
➤ Three-column layout



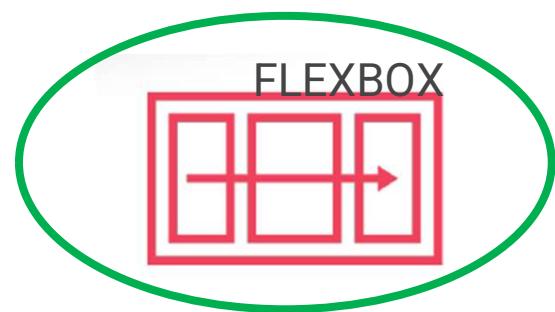
Tworzenie layout

- Floats
- Inline-block
- ~~display: table~~
- Absolute & Relative positioning

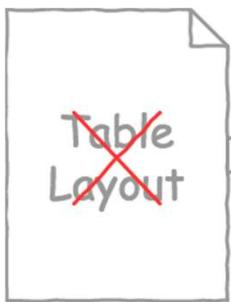
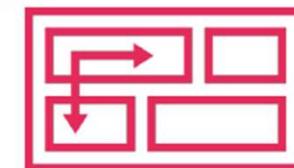
FLOAT LAYOUTS



FLEXBOX



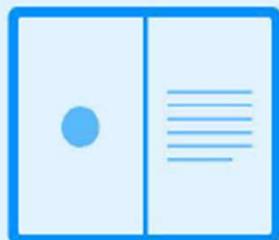
CSS GRID



OR



Flexbox - zastosowanie



Split-screen



Sidebar



Sticky footer



Centering



Fluid grid



Collection grid



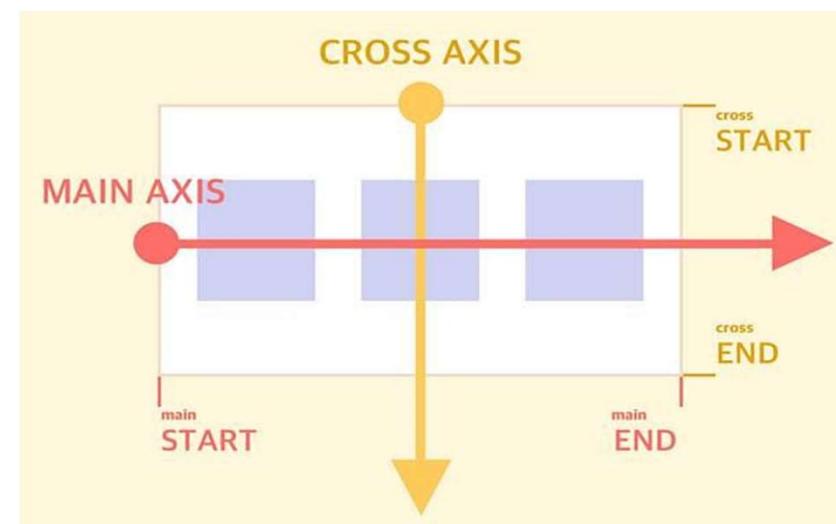
Equal-height modules

FLEXBOX – najważniejsze fakty

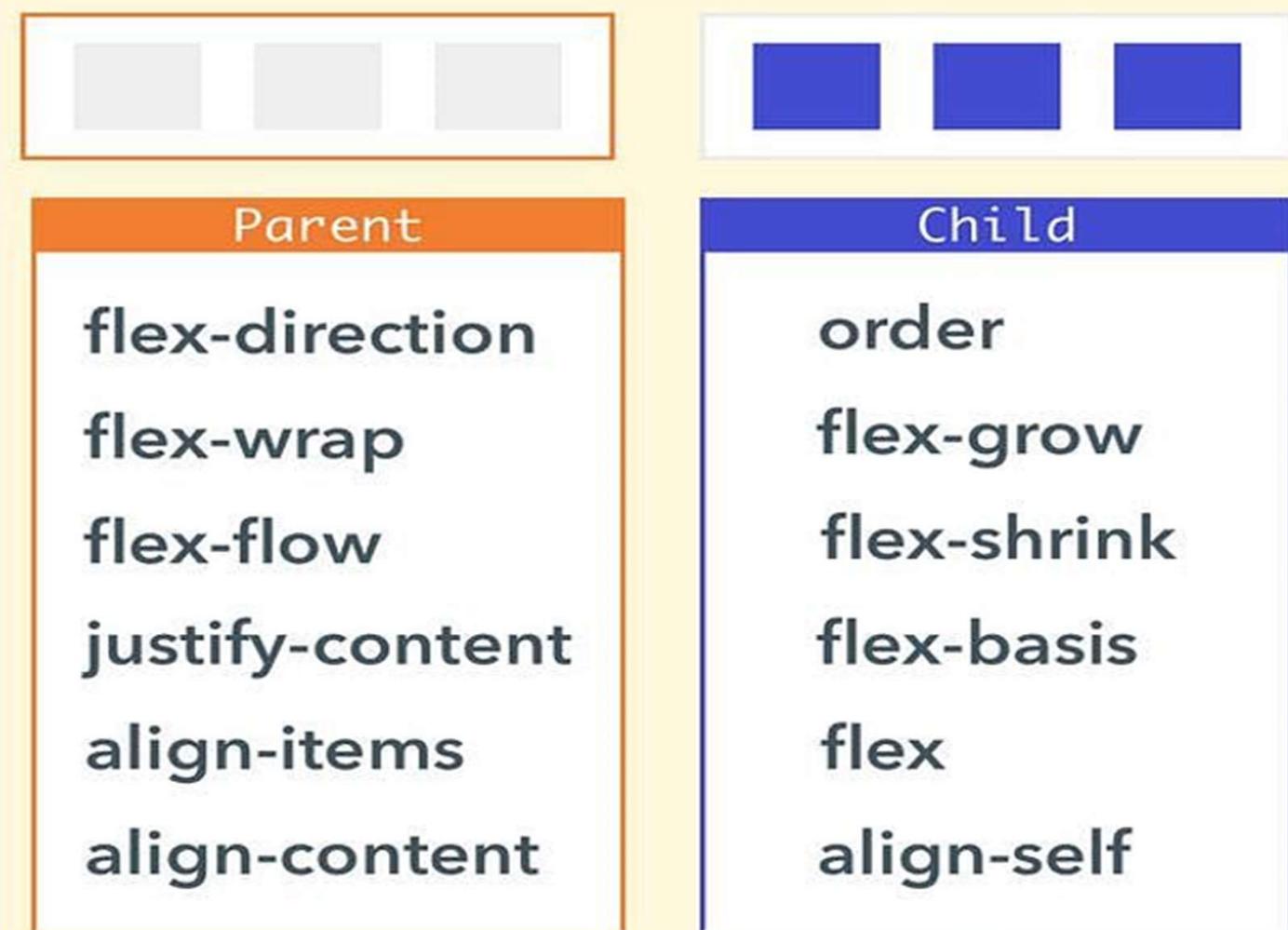
`id=flex-container`

Umieść w kontenerze: `display: flex;`

`class=`
`flex-`
`item`



Flexbox



FLEXBOX – co kontrolujemy?



ALIGNMENT



DIRECTION

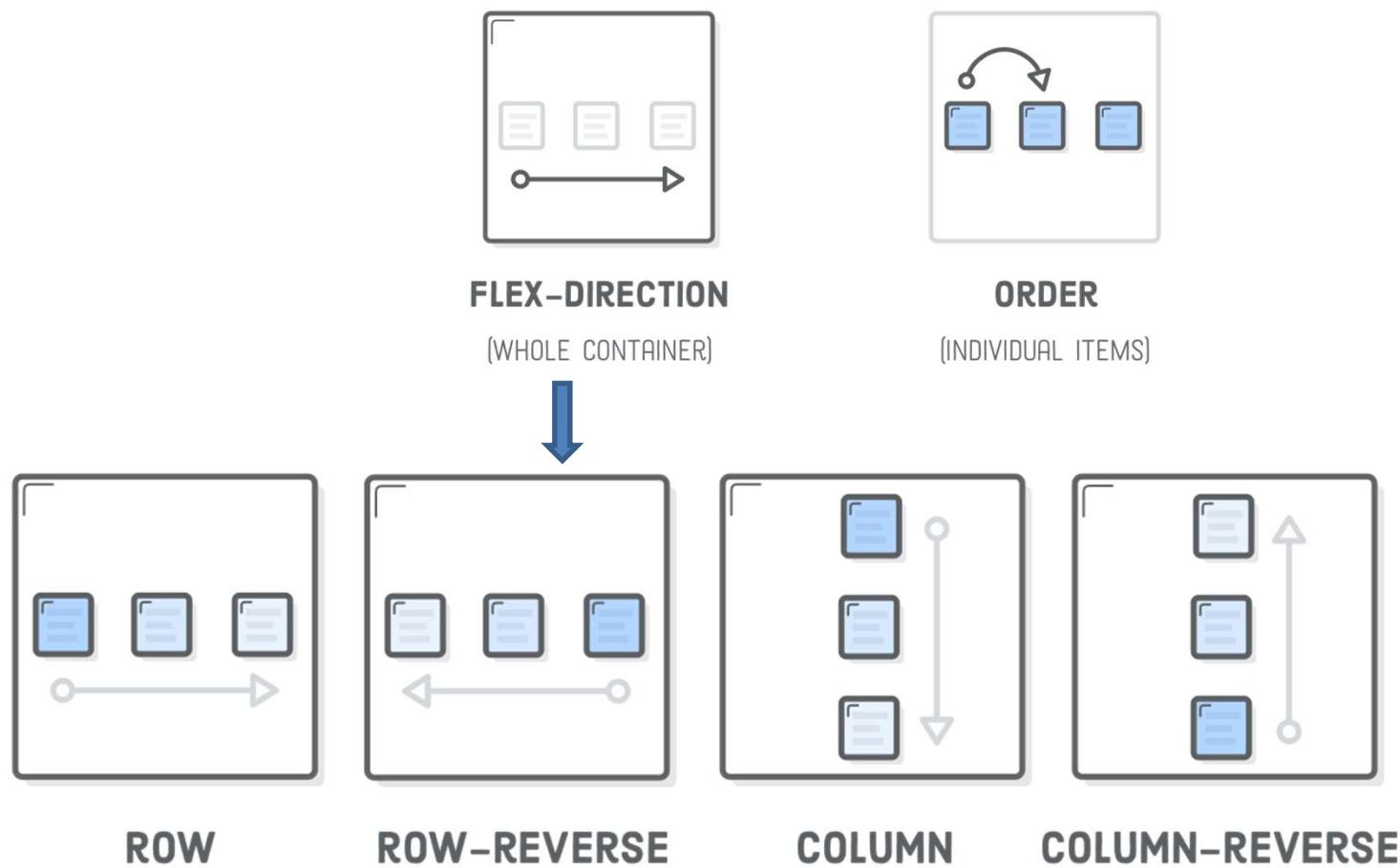


ORDER



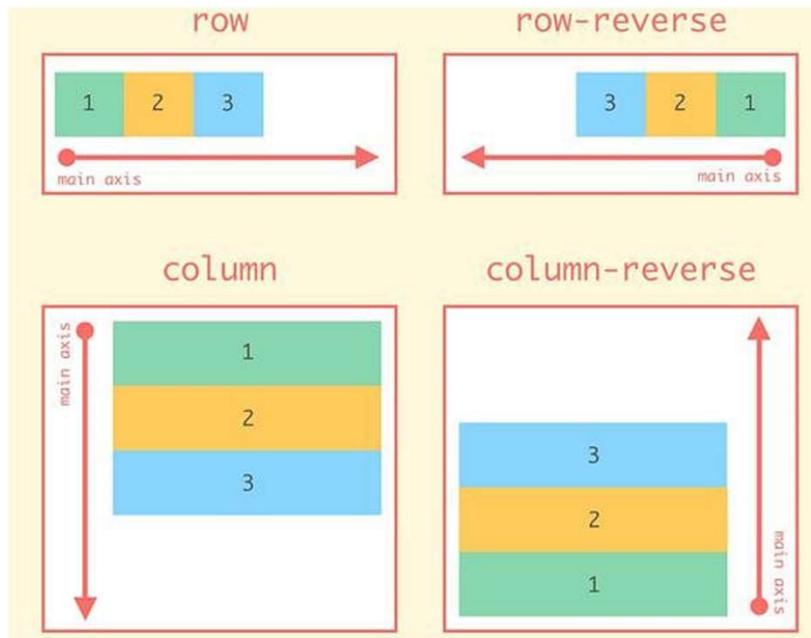
SIZE

flex container order

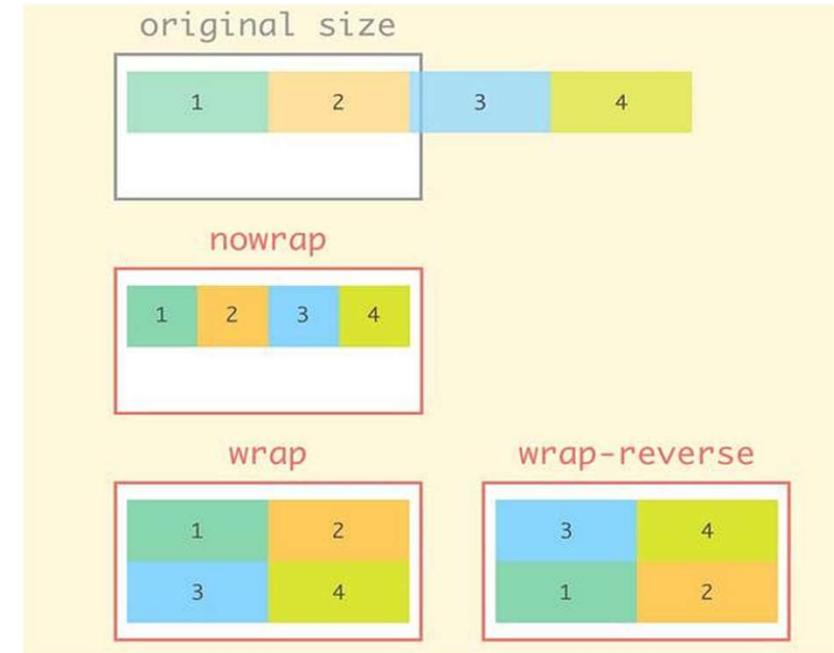


FLEXBOX – właściwości

flex-direction

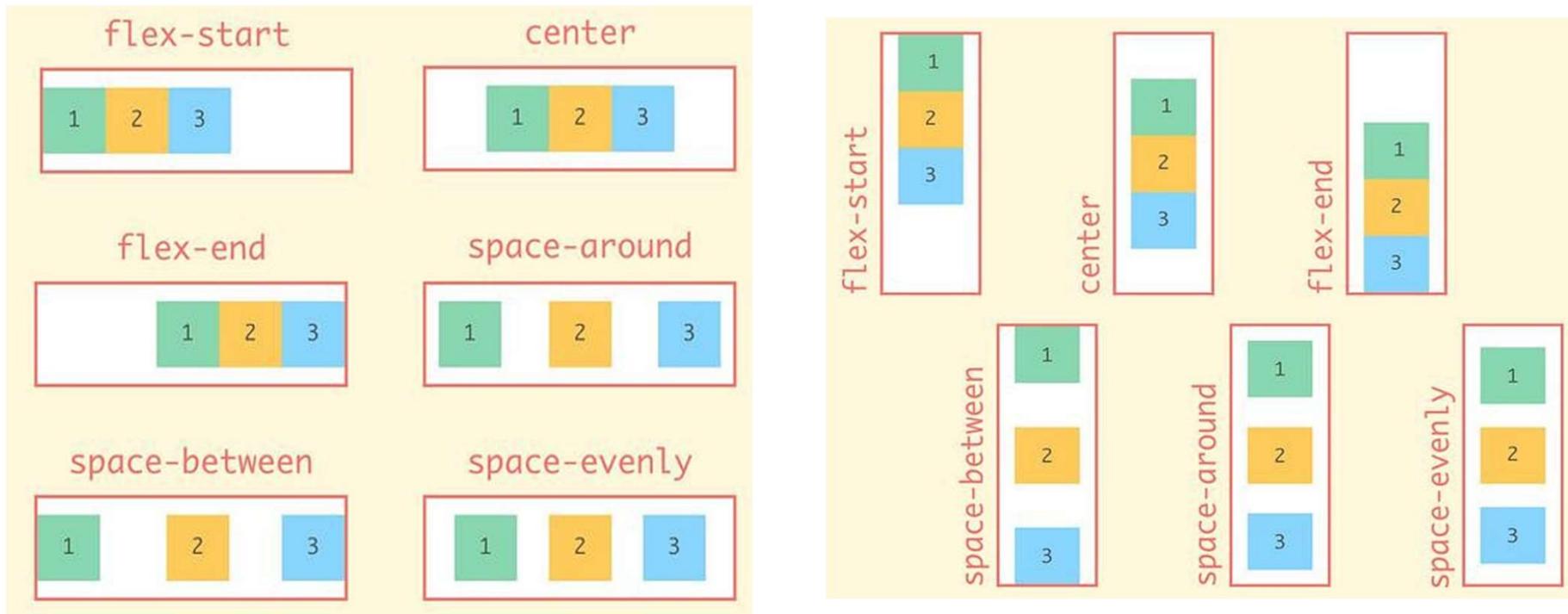


flex-wrap

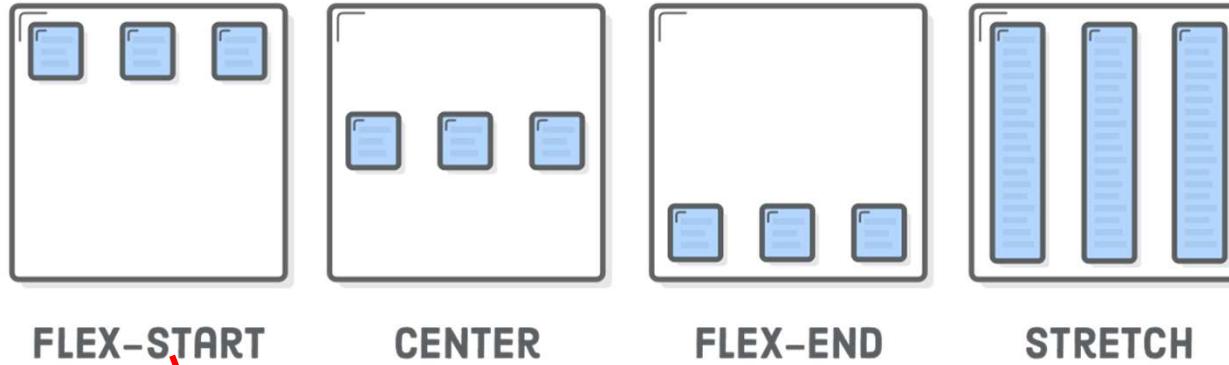


FLEXBOX – właściwości

justify-content



aligning (vertical) a flex item



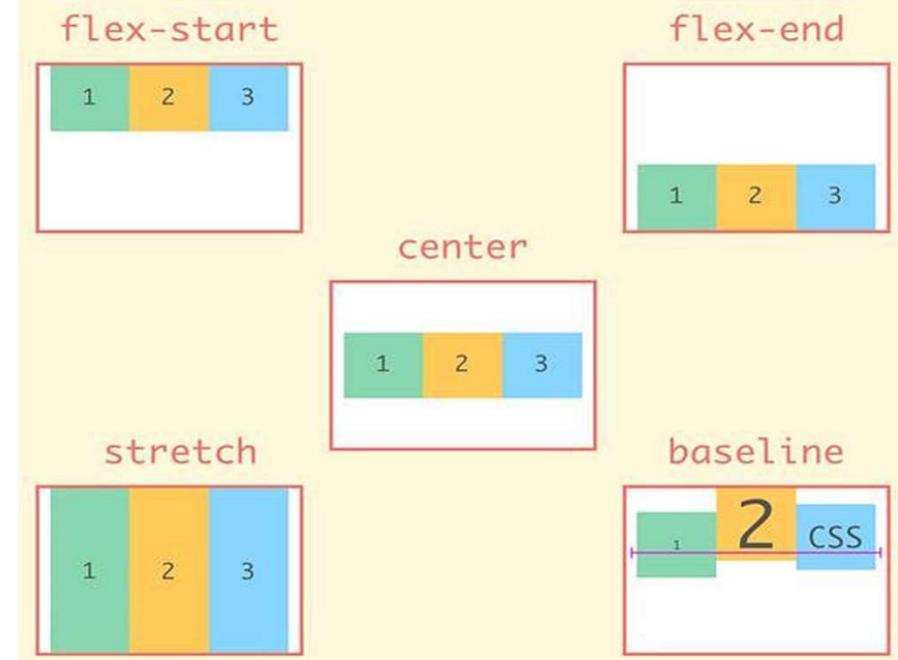
FLEX-START

CENTER

FLEX-END

STRETCH

```
#flex-container {  
display: flex;  
align-items: flex-start;  
}
```

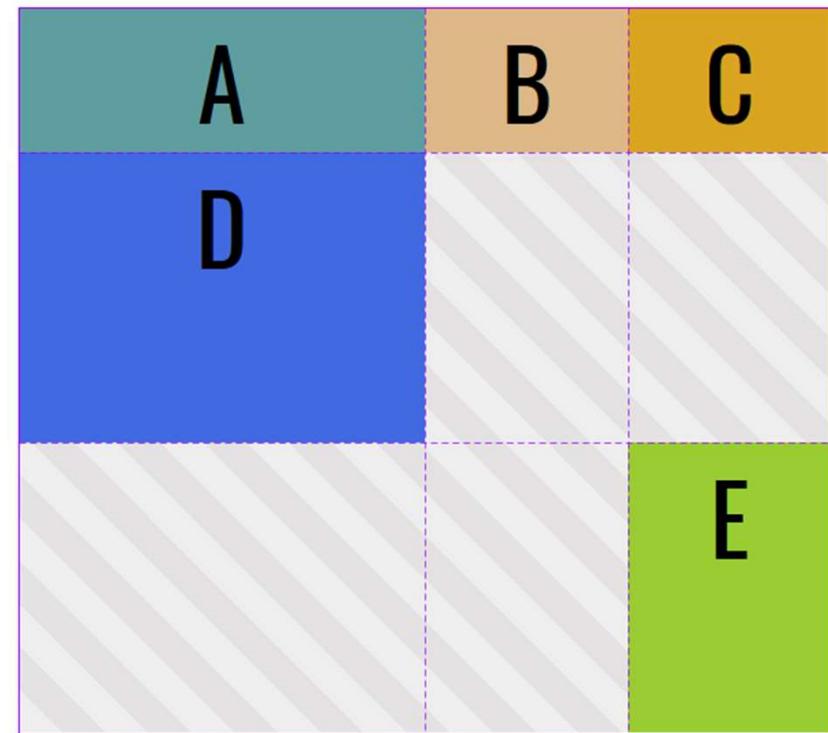


O co chodzi z CSS Grid

Idea, która stoi za CSS Grid, to oprzeć layout o strukturę **siatki**.

Siatka składa się z kolumn i wierszy (jest **druwymiarowa**).

Najmniejszą funkcjonalną częścią siatki jest **komórka**.



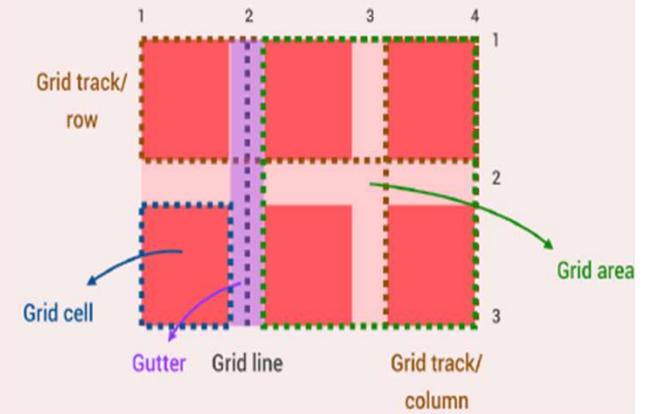
Grid – koncepcja i właściwości

CONTAINER

- 1 grid-template-rows
grid-template-columns
grid-template-areas
 - 2 grid-row-gap
grid-column-gap
 - 3 justify-items
align-items
justify-content
align-content
 - 4 grid-auto-rows
grid-auto-columns
grid-auto-flow
- grid-template
- grid-gap

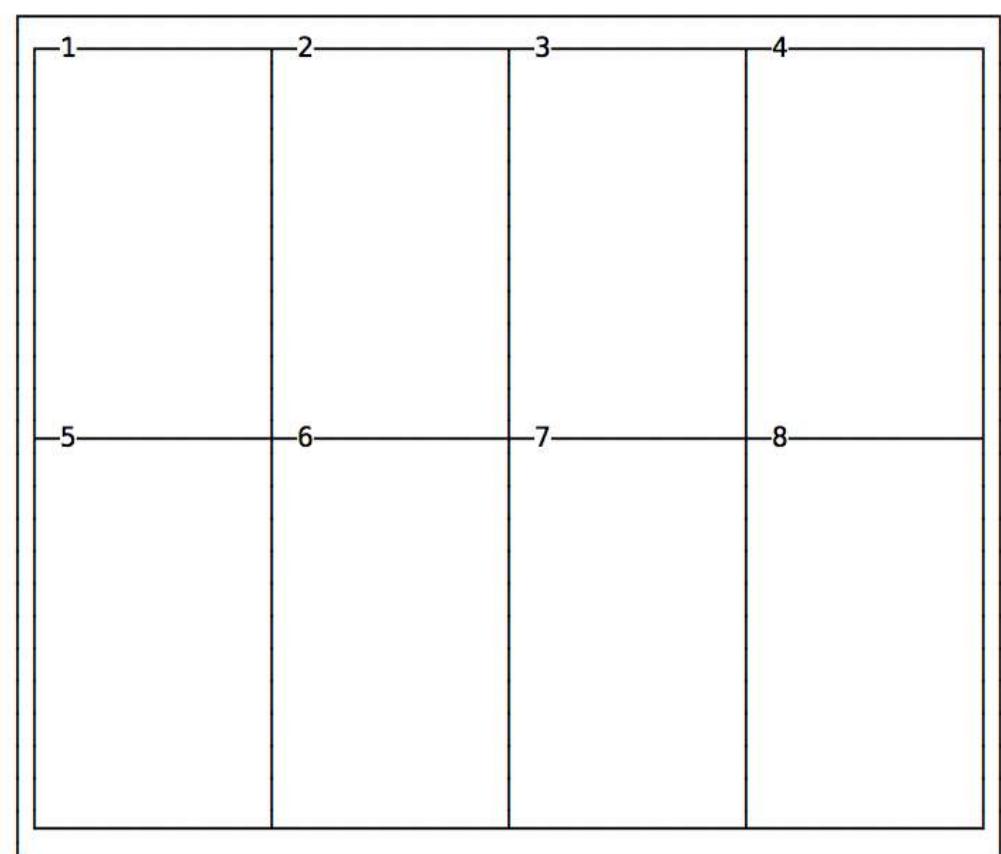
ITEM

- 1 grid-row-start
grid-row-end
grid-column-start
grid-column-end
 - 2 justify-self
align-self
 - 3 order
- grid-row
- grid-column
- grid-area



CSS grid

```
.container {  
    display: grid;  
    grid-template-columns: 200px 200px 200px 200px;  
    grid-template-rows: 300px 300px;  
}
```



CSS3 – co nowego

- **CSS3 Borders - Cienie i Zaokrąglenia**
- **CSS3 Text Effects**
- **CSS3 Backgrounds**
- **CSS3 Fonts**
- **CSS3 2D Transforms**
- **CSS3 3D Transforms**
- **CSS3 Transitions**
- **CSS3 Animations**
- **CSS3 Układ elastyczny**
- **CSS3 Zaawansowane filtry**

CSS3 Transformacje 2D i 3D

Transformacje 2D

translate()

rotate()

scale()

skew()

matrix()

Multiple Transforms dla jednego elementu

```
<style type="text/css">

#submenu {
    background-color: #eee;
    -webkit-transition: all 1s ease-in-out;
    -moz-transition: all 1s ease-in-out;
    -o-transition: all 1s ease-in-out;
    -ms-transition: all 1s ease-in-out;
    transition: all 1s ease-in-out;
}

#submenu:hover {
    background-color: #fc3;
    -webkit-transform: rotate(360deg) scale(2);
    -moz-transform: rotate(360deg) scale(2);
    -o-transform: rotate(360deg) scale(2);
    -ms-transform: rotate(360deg) scale(2);
    transform: rotate(360deg) scale(2);
}

</style>
```

CSS3 Transitions

Transitions (Przemiana)

Definiuje przemianę (przejście) obiektu

Użycie: transition: własność czas-trwania rozkład-w-czasie opóźnienie.

[Algorytm użycia transition w CSS:](#)

1. Zdefiniować dla obiektu styl zwykłego
2. Zdeklarować końcowy stan dla obiektu np., dla hover
3. Do stanu zwykłego dodać funkcję transition przy użyciu właściwości : transition-property, transition-duration, transition-timing-function, and transition-delay.

[lista właściwości jakie mogą być mogą być przemieniane](#)

- background-color and background-position ■ border-color, border-spacing, and border-width
- bottom, top, left, and right ■ clip ■ color ■ crop ■ font-size and font-weight ■ height and width
- letter-spacing ■ line-height ■ margin ■ max-height, max-width, min-height, and min-width
- opacity ■ outline-color, outline-offset, and outline-width ■ padding ■ text-indent ■ text-shadow
- vertical-align ■ visibility ■ word-spacing ■ z-index

Dla leniwych : <http://www.css3-generator.de/transform.html>



**Responsive
Web
Design**

Składowe nowoczesnej aplikacji webowej

czyli jak pisać dobry kod HTML i CSS ... i budować dobre strony internetowe

PROJEKTOWANIE
RESPONSYWNE

- elastyczny układ (layouts)
- media queries
- responsywne zdjęcia
- właściwe jednostki
- desktop-first a mobile-first

JAKOŚĆ
KODU

Kod skalowany i łatwy do zarządzania

- czystość formy
- łatwy do zrozumienia
- łatwość rozbudowy
- wielokrotnego użycia
- organizacja struktury plików
- nazewnictwo klas i struktur
- struktura HTML

WYDAJNOŚĆ

- mniej zapytań HTTP
- mniej kodu
- kompresja kodu
- użycie preprocessora CSS
- mniej obrazów
- kompresja obrazów

Problem - wygląd strony na różnych nośnikach



Rozwiązańia

- **Adaptive Design** – serwer zwraca spreparowaną stronę na podstawie informacji o urządzeniu klienckim.
Jeden adres URL.
- **Separate Mobile Site (.m)**- oddzielny adres URL dla stron mobilnych
- **Responsive Web Design (RWD)**



Nowoczesne aplikacje internetowe - FrontEnd

RWD Responsive Web Design [Reaktywne Projektowanie Stron] - jest to nowoczesne podejście do projektowania stron internetowych, w którym programista zapewnia, że niezależnie od tego na jakie urządzenie strona zostanie podana wyświetli się ona prawidłowo. W tym kontekście programista zapewnia dostosowanie się wyglądu strony, rozmiaru oraz układu strony do rozmiaru okna przeglądarki w którym będzie wyświetlana.

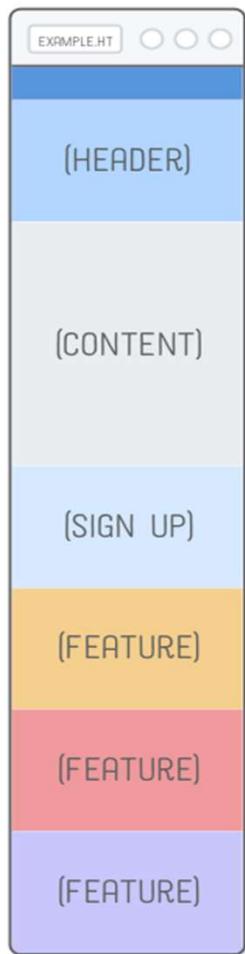
Jak to osiągnąć ?

Na przykład:



Idea RWD

MOBILE



TABLET



DESKTOP



Responsive Design

- Definiowany przez 3 najważniejsze elementy
 - Flexible grid-based layout – układ siatkowy
 - style warunkowe oparte o tzw. Media queries (CSS3)
 - Zdjęcia o elastycznym rozmiarze (Images resize)
- www.alistapart.com/articles/responsive-web-design/

The screenshot shows the header and top portion of the article page. On the left is the A List Apart logo, featuring a large stylized 'A' inside a laurel wreath, with the words 'LIST apart' below it and 'FOR PEOPLE WHO MAKE WEBSITES' at the bottom. To the right of the logo is a small black bird icon. The main title 'Responsive Web Design' is in red text, followed by the author's name 'by ETHAN MARCOTTE'. Below the title is a horizontal dotted line. At the bottom of the screenshot, there is a note about publication in categories like CSS, Layout, User Interface Design, Mobile, Mobile Design, and Mobile Development, along with links to 'Discuss this article' and 'Share this article'.

MAY 25, 2010

Responsive Web Design

by ETHAN MARCOTTE

Published in: CSS, Layout, User Interface Design, Mobile, Mobile Design, Mobile Development

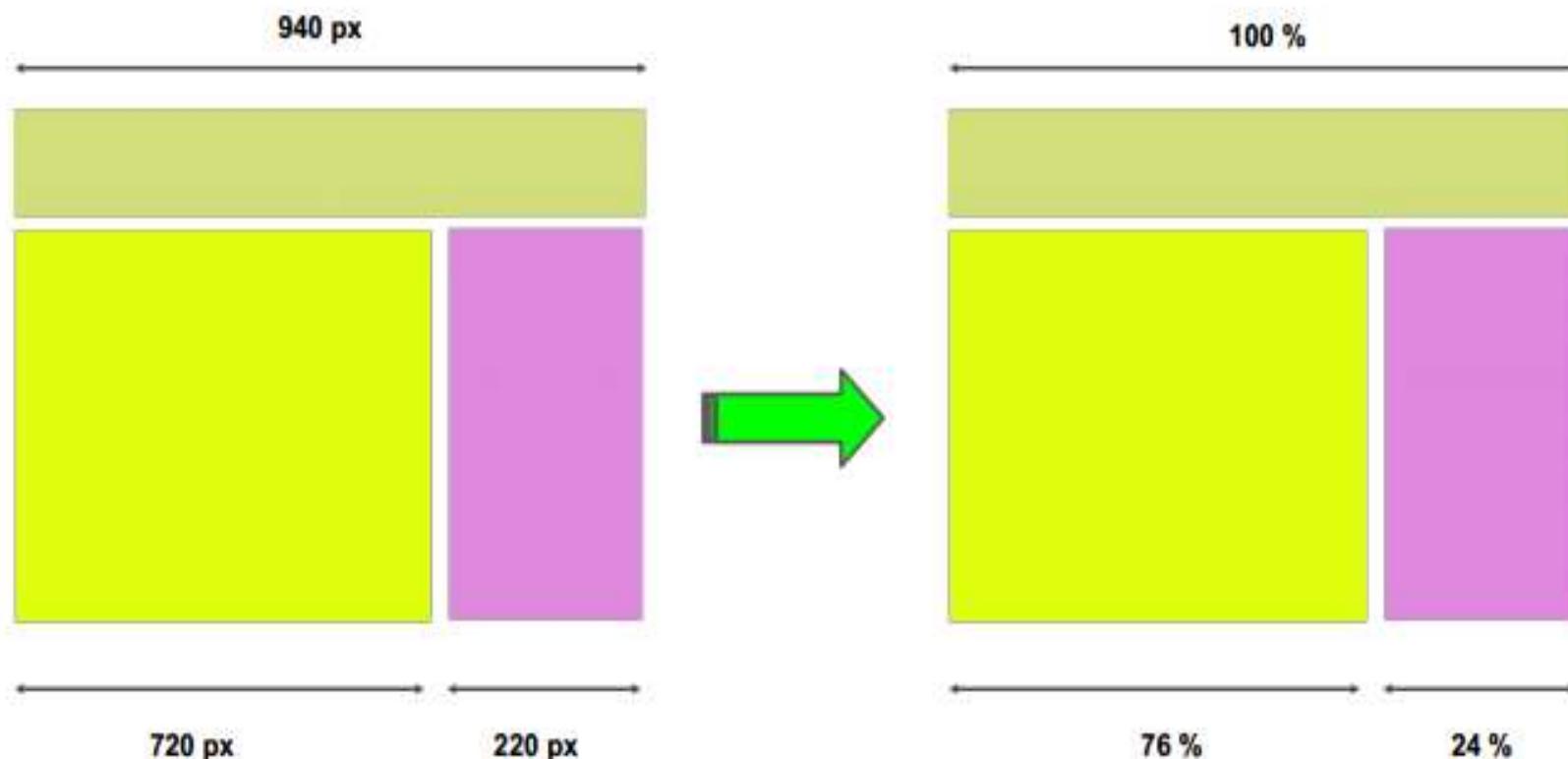
[Discuss this article »](#) [Share this article »](#)

1. Flexible grid-based layout

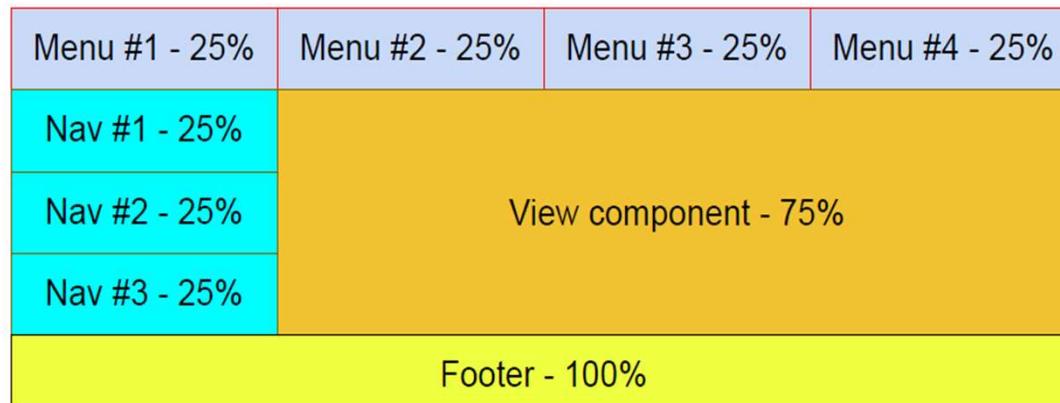
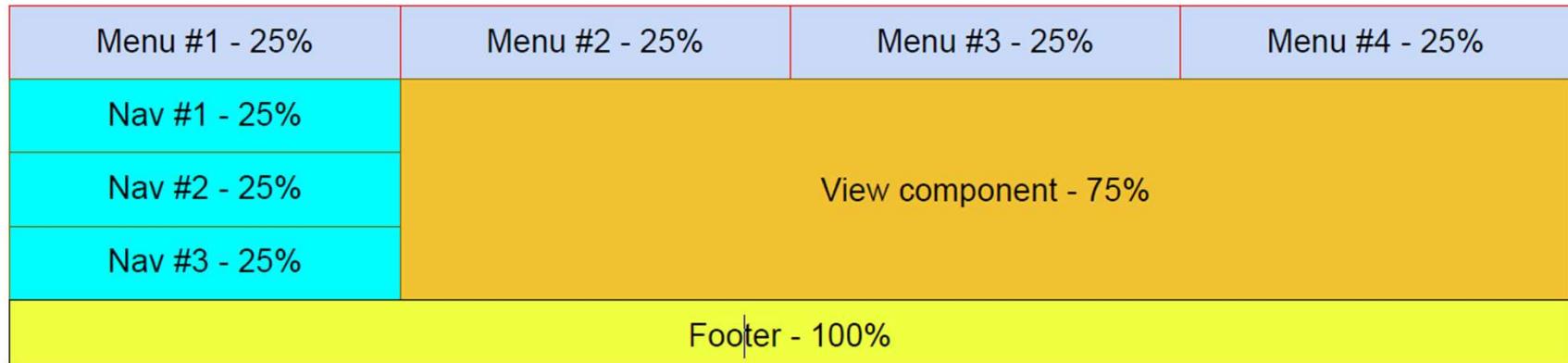
Chcąc zaprojektować "elastyczną" stronę dopasowującą się do całego ekranu powinniśmy zrezygnować z jednostek bezwzględnych wyrażonych np. w px i zastąpić je jednostkami względnymi w procentach %.

Pozwoli to uzyskać dopasowany układ do różnych szerokości ekranów.

Atrybuty *max-width* powinny być używane często zwłaszcza w przypadku elementów, które mogą nam "rozpychać layout" tj. obrazów, osadzonych odtwarzaczy video etc.



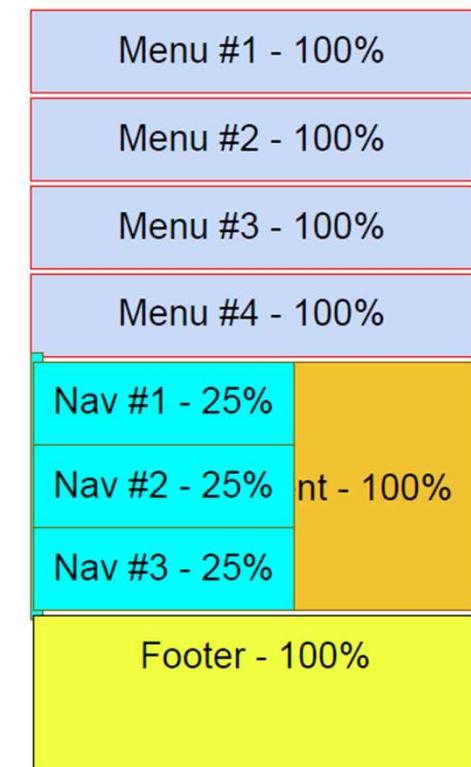
Układ elastyczny - przykład



Ale czasami sam układ elastyczny nie wystarczy

Reguły CSS:

```
@media only screen and (min-width:768px) {  
/* layout do tabletów i desktopów */ }  
  
@media only screen and (max-width: 767px)  
{ /* smartfony */ }  
  
@media only screen and (max-width:767px)  
and (orientation: portrait) {  
/* smartfony orientacja portrait */ }
```

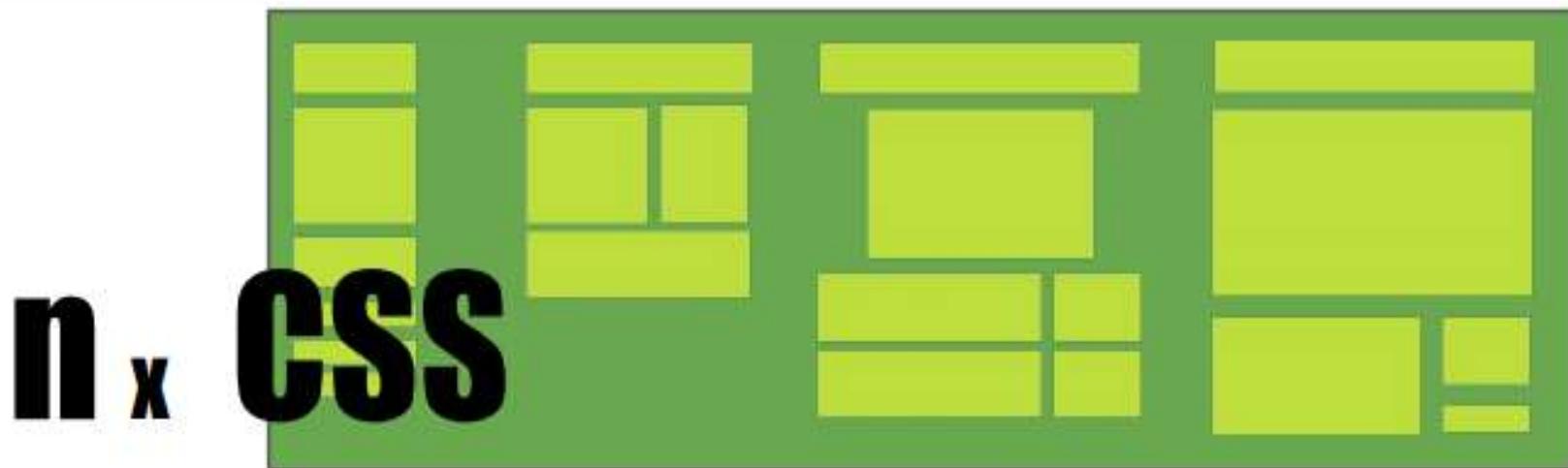


2. Style warunkowe

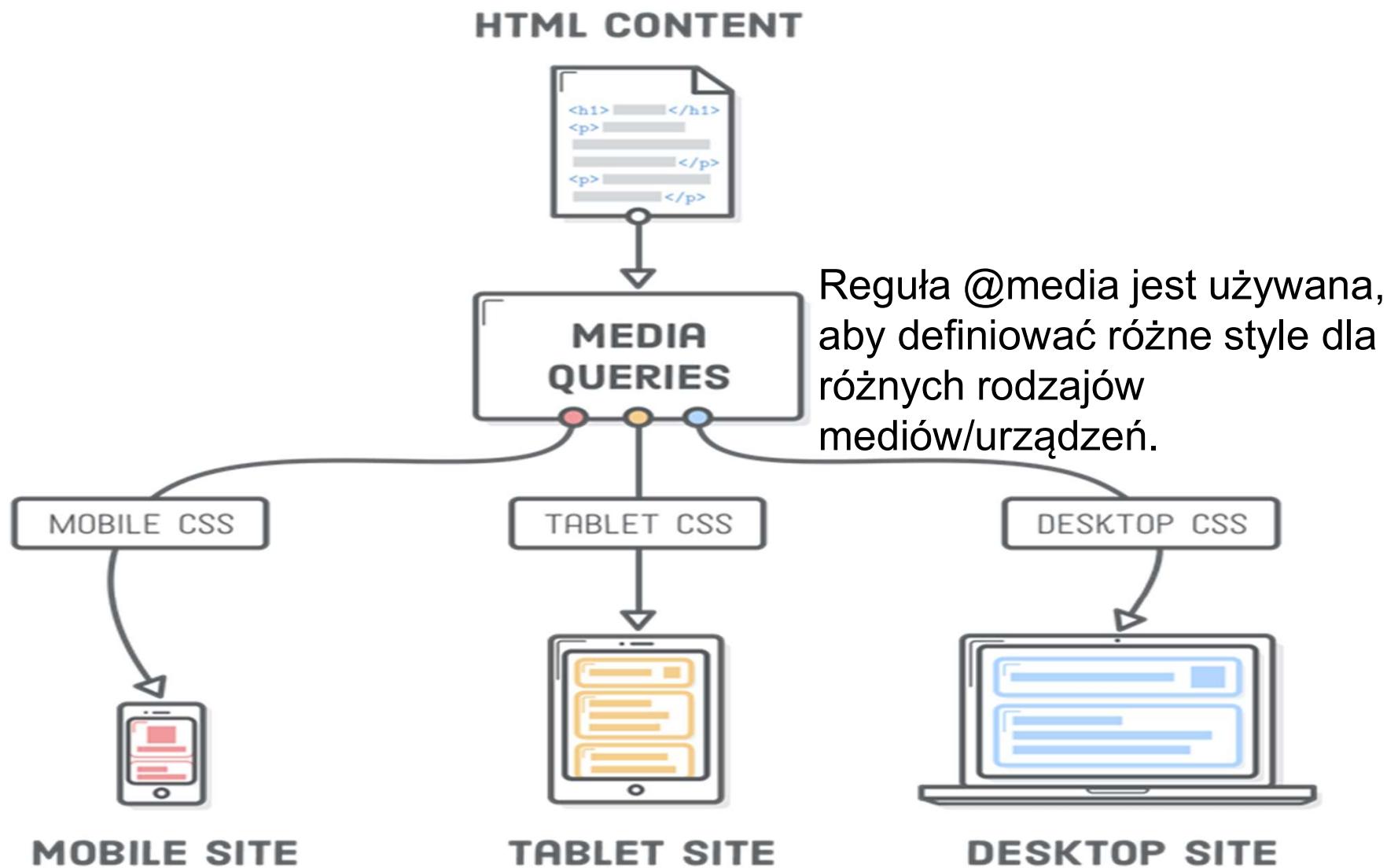


ZASADA:

W ogólności zasada jaką powinna przyświecać tworzeniu stron zgodnie z metodologią RWD jest utworzenie jednego pliku HTML, a dla formatowania jego wyglądu wiele “layoutów” w CSS które będą zawierać odpowiednie reguły wyświetlania strony w zależności od rozdzielczości ekranu.



RWD



Zapytania mediów - media queries

- Technika CSS wprowadzona w CSS3.
- Używa reguły @media żeby dołączyć jakiś blok właściwości CSS jeśli zdefiniowany warunek jest prawdziwy.

Przykłady:

- jeśli okno przeglądarki jest mniejsze niż 500 pikseli to kolor tła zmieni się na jasnoniebieski

```
@media only screen and (max-width: 500px) {  
    body {  
        background-color: lightblue;  
    }  
}
```

- kiedy ekran (okno przeglądarki) stanie się mniejsze niż 768 pikseli, każda kolumna będzie miała szerokość 100%:

```
@media only screen and (max-width: 768px) {  
    [class*="col-"] { /* dla mobile phones: */  
        width: 100%;  
    }  
}
```

Media Query – informacje

Zapytania mediów sprawdzają możliwości urządzenia, mogą być użyte do sprawdzenia wielu rzeczy, np.:

- szerokości i wysokości okna roboczego,
- szerokości i wysokości ekranu urządzenia
- orientacji (czy tablet/telefon jest poziomo czy pionowo?)
- rozdzielczości i wielu innych.

Zapytanie mediów standardowo składa się z:

- **typu lub grupy** medium,
- słowa kluczowego **and**
- **cechy** medium umieszczonej w nawiasie.

Operatory logiczne

- AND
- przecinek
- NOT

REGUŁY MEDIA QUERIES

```
@media warunek1, warunek2, (...), warunek-n {  
    /* kod css dla urządzeń spełniających podane warunki */  
}  
  
@media screen {  
    * { font-family: sans-serif }  
}  
  
@media screen and (min-width: 400px) and (max-width: 700px)  
{ ... }  
  
@media not screen and (color) {  
    /* kod css dla urządzeń nie posiadających kolorowego ekranu */  
}
```

Inne przykłady

```
@media (pointer: coarse) and (min-width: 600px) {  
  ...  
}  
@media (min-width: 900px), (orientation: landscape) {  
  ...  
}  
@media not (orientation: landscape) {  
  ...  
}
```

Typy mediów

- **all** - dla wszystkich typów urządzeń
- **print** - używane dla drukarek (dla podglądu wydruku)
- **screen** - dla ekranów komputera, tabletu, smartfonów itp.
- **speech** - dla czytników ekranu, które czytają stronę
- **aural** - **nieaktualne**, dla syntezatorów mowy i dźwięku
- **braille** - **nieaktualne**, dla urządzeń przeznaczonych dla niewidomych
- **embossed** - **nieaktualne**, dla drukarek brailla
- **handheld** - **nieaktualne**, dla bezprzewodowych urządzeń ręcznych
- **projection** - **nieaktualne**, dla prezentacji projektorowych
- **tty** - **nieaktualne**, dla dalekopisów, terminali z ograniczonymi możliwościami wyświetlania
- **tv** - **nieaktualne**, dla telewizora

MEDIA QUERIES

Reguły media queries bazują na cechach nośnika, takich jak:

- Szerokość: width, min-width, max-width
- Wysokość: height, min-height, max-height
- Szerokość urządzenia: device-width, min-device-width, max-device-width
- Wysokość urządzenia: device-height, min-device-height, max-device-height
- Orientacja: orientation
- Proporcje obrazu: aspect-ratio, min-aspect-ratio, max-aspect-ratio
- Proporcje obrazu urządzenia: device-aspect-ratio, min-device-aspect-ratio, max-device-aspect-ratio
- Głębia koloru: color, min-color, max-color
- Paleta kolorów: color-index, min-color-index, max-color-index
- Urządzenia monochromatyczne: monochrome, min-monochrome, max-monochrome
- Rozdzielcość: resolution, min-resolution, max-resolution
- Technika wyświetlania: scan
- Siatka: grid

Sposoby implementacji media queries?

Trzy sposoby implementacji:

1. Użycie importu stylów zewnętrznych w pliku CSS - @import.

```
@import url(style600.css) screen and (min-width: 600px);
```

2. użycie Media Queries bezpośrednio w arkuszu stylów

```
@media screen and (min-width: 600px){ #section{ width: 550px; } }
```

3. Użycie query jako atrybutu w stylu zewnętrznym

```
<link rel="stylesheet" type="text/css" media="screen and (max-device-width: 600px)" href="style600.css" />
```

REGUŁA MEDIA QUERIES

```
<head>
    <meta name="viewport" content="width=device-width, initial-
scale=1">
</head>
```

- width=device-width – ustawia szerokość strony adekwatną do szerokości wyświetlacza urządzenia
- initial-scale=1 – ustawia poziom powiększenia podczas pierwszego uruchomienia strony.

Meta-tag *viewport* zapewnia poprawne dostosowanie szerokości oraz brak powiększenia. Jeżeli zapomnisz go umieścić, strona będzie prezentować się źle.

**Wyłączenie zoomowania dla urządzeń mobilnych:
zasada mobile-first**

```
<meta name = 'viewport' content='width=device-width, initial-scale=1,
maximum-scale = 1'
```

Desktop-First vs Mobile-First

Desktop-First

```
.example-section {  
display: flex;  
width: 60%;  
}  
.example-section__text {  
font-size: 32px;  
}  
@media (max-width: 768px) {  
.example-section {  
flex-direction: column;  
width: 80%;  
}
```

Mobile-First

```
.example-section {  
  display: flex;  
  flex-direction: column;  
  width: 80%;  
}  
.example-section__text {  
  font-size: 24px;  
}  
@media (min-width: 768px) {  
.example-section {  
  flex-direction: row;  
}
```

Jak ustalić breakpointy?

- Podział popularnych urządzeń na kilka grup
- BP zgodne z 'wymaganiami' designu

Typowe ustawienia

Label	Layout Width
Smartphones	480px and below
Portrait Tablets	480px to 768px
Landscape Tablets	768px to 940px
Default	940px and up
Large Screens	1210px and up

PUNKTY PRZEŁAMANIA (RWD BREAKPOINTS)

```
/*===== Mobile First Method =====*/
/* Custom, iPhone Retina */
@media only screen and (min-width : 320px) { ... }

/* Extra Small Devices, Phones */
@media only screen and (min-width : 480px) { ... }

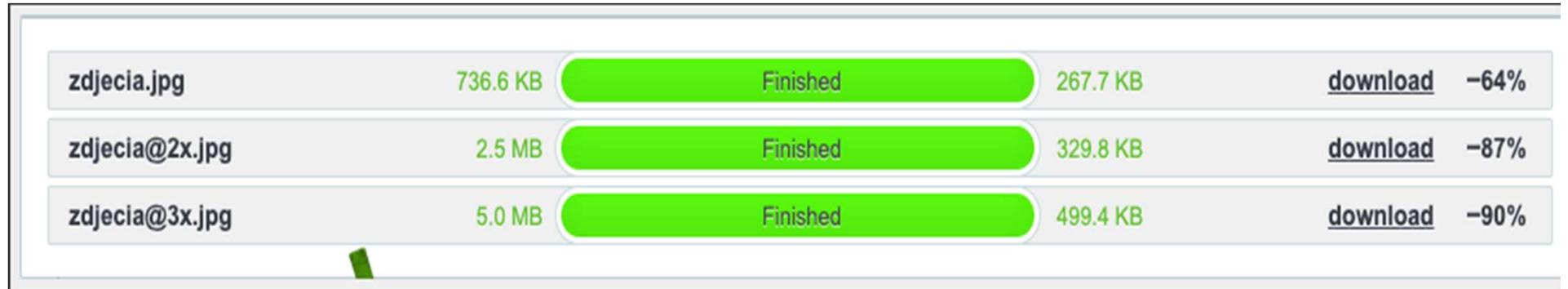
/* Small Devices, Tablets */
@media only screen and (min-width : 768px) { ... }

/* Medium Devices, Desktops */
@media only screen and (min-width : 992px) { ... }

/* Large Devices, Wide Screens */
@media only screen and (min-width : 1200px) { ... }
```

Optymalizacja zdjęć/grafik

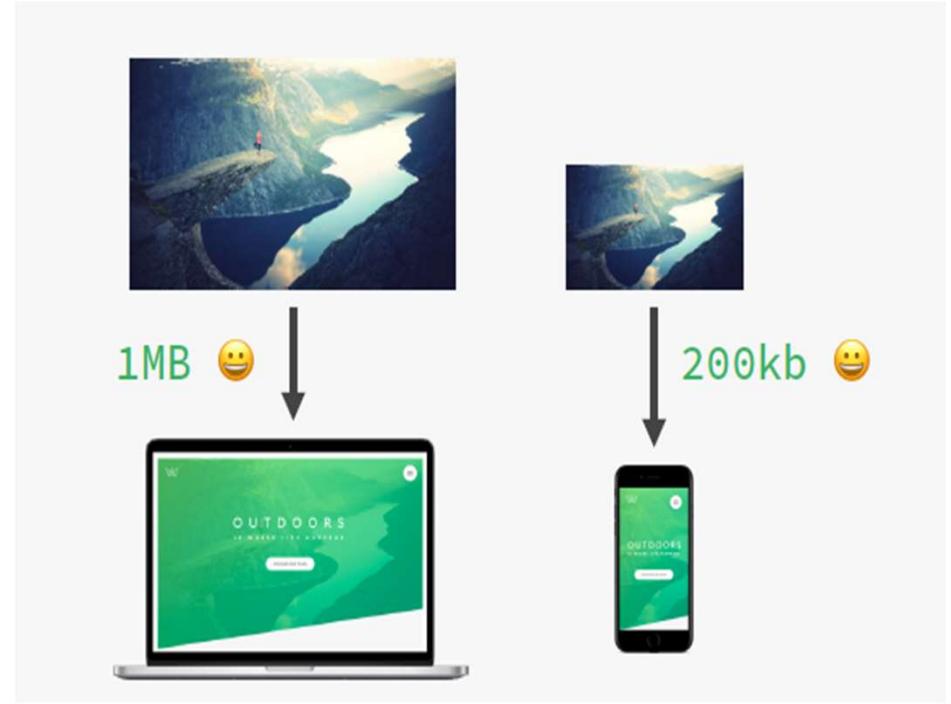
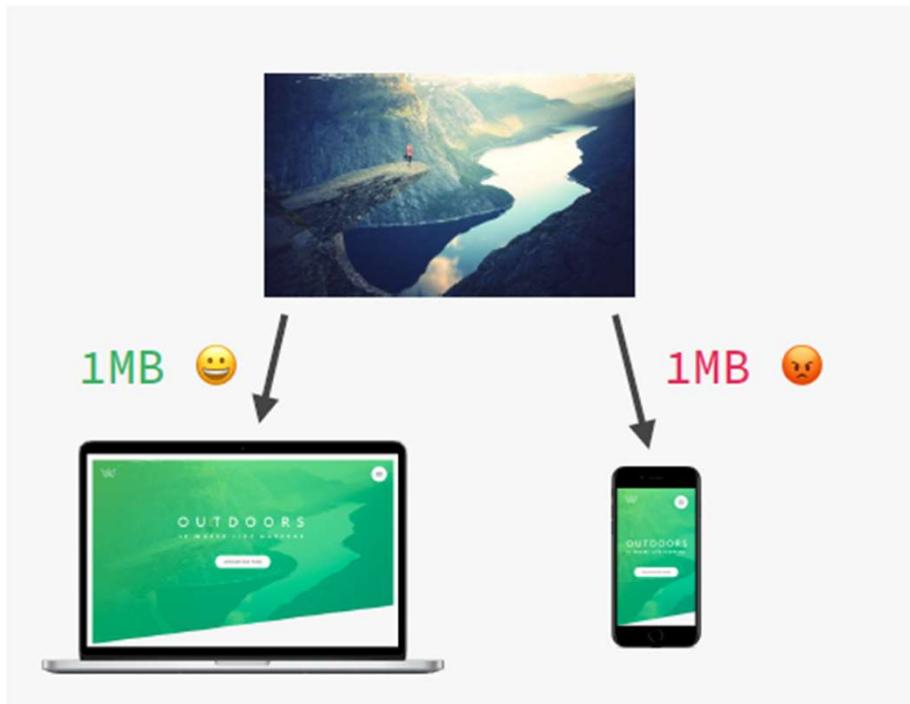
- Kompresja plików



- Różne pliki dla różnych ekranów
- Skalowanie zdjęć i filmów video w celu dopasowania do rozmiaru obszaru na ekranie

```
img { width: 100%; height: auto; }  
video { width: 100%; height: auto; }
```

Różne pliki dla różnych ekranów



<https://responsivebreakpoints.com/>

Zalecenia

- Używaj obrazów, które najlepiej pasują do cech wyświetlacza. Weź pod uwagę rozmiar ekranu, rozdzielcość urządzenia i układ strony.
- Gdy chcesz określić różne obrazy wyświetlane w zależności od cech urządzenia (tzn. dostosować grafikę), użyj elementu **picture**.
- Użyj atrybutu **srcset** i deskryptora x w elemencie **img**, by przy wyborze obrazu podpowiedzieć przeglądarce, której rozdzielcości najlepiej użyć.

Grafika w zależności od gęstości pikseli

```

```

Grafika w zależności od rozmiaru ekranu

```

```

sizes są podobne do zapytań o media, opisujące, ile miejsca zajmuje obraz w rzutni.

- jeśli rzutnia jest większa niż 1200px, obraz ma dokładnie 800px
- jeśli rzutnia ma wielkość między 1200px a 600px, obraz zajmuje 48% rzutni
- jeśli rzutnia jest mniejsza niż 640px obraz zajmuje 580 px

srcset po prostu mówi przeglądarce, jakie obrazy mamy dostępne i jakie są ich rozmiary.

- img/foto.jpg ma szerokość 480 pikseli,
- img/foto2.jpg ma szerokość 700 pikseli,

Grafika w zależności od 'wizji artystycznej'

```
<picture>
<source media="(max-width: 799px)"
srcset="img/foto.jpg">
<source media="(min-width: 800px)"
srcset="img/foto2.jpg">

</picture>
```