CSS Kaskadowe Arkusze Styli

dr inż. Grzegorz Rogus

Czym jest CSS

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

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

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

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 biegle korzystać CSS trzeba…

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

div.first p -> SELEKTOR – GDZIE ZASTOSOWAC

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  }
```

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 Akusze 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

CSS (Kaskadowe Akusze Stylu)

DEMO -> background.html

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; }
```

Klasy i identyfikatory

Jak klasy i identyfikatory pomagają w definicji stylu strony ?

Możemy za pomocą klasy lub identyfikatora konkretnie wybrać element(y) dla którego mają być nadane specyficzne własności i cechy.

Selektor klasy:

```
selektor.nazwaklasy { cecha1: wartość;
                      cecha2: wartość;
                      ...;
                    }
```

Separator klasy

Selektor identyfikatora:

```
selektor#nazwaklasy { cecha1: wartość;
                      cecha2: wartość;
                      ...;
                    }
```

Separator identyfikatora

Przy okazji – król znaczników div

- Bloki wydzielamy za pomocą znaczników ** ... **. Fragment dokumentu objęty blokiem można w swobodny sposób formatować za pomocą stylów.
- Wydzielane bloki mogą zawierać w sobie tytuły, akapity, wykazy, a także inne bloki.
- Dzięki temu nadają się do wydzielania większych, logicznych fragmentów dokumentów i nadawania im specyficznego formatowania za pomocą stylów.

Identyfikacja bloku – specyfikacja klasy bloku

```
.komentarz
{
  font-family: verdana, arial, sanserif;
  font-weight: bold;
  color: blue;
}

<div class="komentarz">
  <h1>Uwaga</h1>
  <p>
    To jest tekst komentarza
   </p>
</div>
```

Uwaga

To jest tekst komentarza

	Własności
Fonty	font-family font-size font-weight font-style font-variant text-decoration @font-face
Wielkość	width height
Układ	position left, right float, clear
Grafika	background-image background-repeat background-position

	Własności
Barwy	color background-color
Przestrzeń i odstępy	margin padding margin-left, margin-right, margin-top, margin-bottom padding-left, padding-right, padding-top, padding-bottom
Obramowanie	border-width border-style border-color border (ustawia grubość, styl i barwę za jednym zamachem)
Wyrównanie tekstu	text-align text-indent word-spacing letter-spacing line-height white-space

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><img src="breeze.jpg"></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 potomków (selektory kontekstowe)

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>

<img src="breezes.jpg"></p>

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

<h2>Tytuł</h2>

<p>Oraz za wyjątkiem paragrafów które następują
po nagłówkach</p>

<p>Kolejny paragraf w tym tekście już powinien
takie wcięcie posiadać</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

Oraz za wyjątkiem paragrafów które następują po nagłówkach

Kolejny paragraf w tym tekście już powinien takie wcięcie posiadać

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 element E

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

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

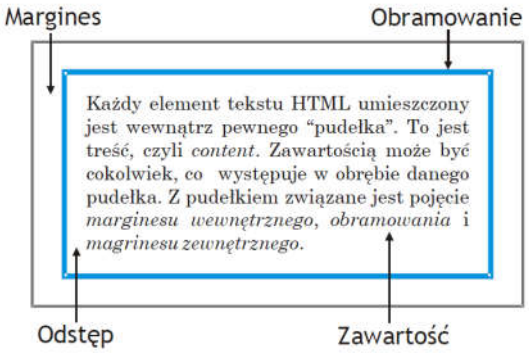
```
p::first-line {font-style:italic; }
```

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

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

BOX MODEL

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



BOX MODEL

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 &mdash; box model</h1>
```

Model pojemnika — box model

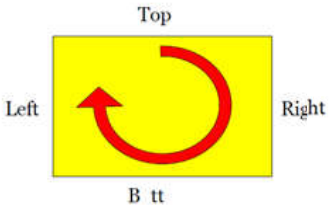
```
h1
{
  background-color: skyblue;

  padding-left: 20px;
  padding-top: 15px;
  padding-bottom: 15px;
  padding-right: 20px;
}

...

<h1>Model pojemnika &mdash; box model</h1>
```

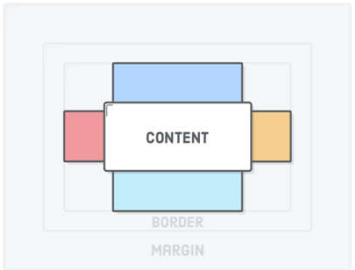
Model pojemnika — box model

ZASADA WSKAZÓWEK ZEGARA

Jak zapamiętać przypisanie wartości do określonych boków pojemnika?

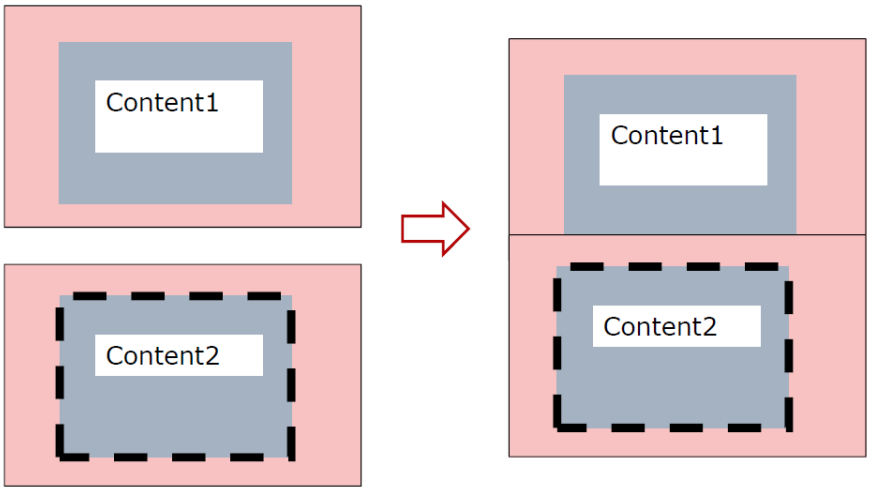
- ***Reguła stopera*** lub ***zegara wskazującego 12-stą***,
- ***Reguła trouble*** – TopRightBottomLeft.






```
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

Rodzaje pojemników

Dwa rodzaje pojemników:

- *Blokowe* (ang. *block boxes*), generowane np. przez znaczniki *p, div, table.*
- *Wewnątrzliniowe* (ang. *inline boxes*), generowane np. przez znaczniki *b, i, span.*

Elementy blokowe i wewnątrzliniowe

```
<div>
  Tekst pierwszy
  Tekst drugi
  <p>Tekst <em>trzeci</em> w paragrafie</p>
  Tekst czwarty
</div>
```

Tekst pierwszy Tekst drugi

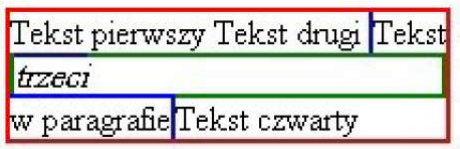
Tekst *trzeci* w paragrafie

Tekst czwarty

Rodzaje pojemników, zmiana sposobu wyświetlania

Można zmieniać sposób traktowania pojemników wykorzystując właściwość *display*.

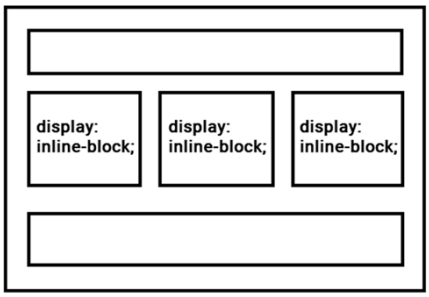
```
body { display: inline; }
p { display: inline; }
em { display: block;}
```




```
p { display: inline; }
```

właściwości display

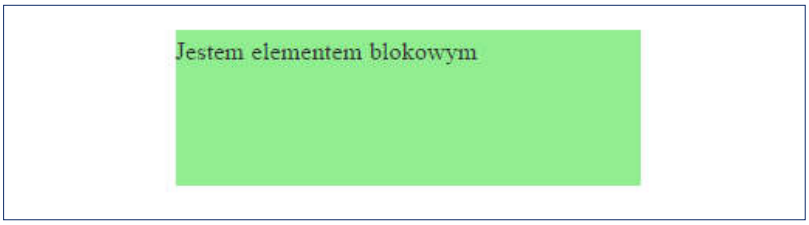
- **display: none** – za pomocą którego możemy ukryć nasz element.
- **display: inline** – sprawia, że element będzie wyświetlany jako element liniowy.
- **display: block** – element zostanie wyświetlony jako element blokowy.
- **display: inline-block** – łączy cechy elementów liniowych i blokowych. Za jego pomocą możemy sprawić, że będzie możliwe nadanie naszemu elementowi wymiarów i wszystkich marginesów, a także nie nastąpi po nim przejście do kolejnej linii.
- **display: grid -** macierz
- **display: flex** - vector



Jak wyśrodkować elementy w pionie i poziomie?

```html
<div class="container">
  <div class="center-me">
    Jestem elementem blokowym
  </div>
</div>
```

```css
.center-me {
  background: lightgreen;
  height: 100px;
  width: 300px;
  margin: 0 auto; /* Środkuję blokowy element */
}
```



PRZEPŁYW DOKUMENTU

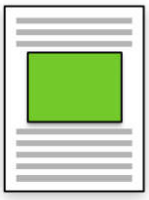
Elementy HTML w dokumencie układają się na kilka sposobów:

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 określane jest pozycją statyczną:

position: static;



PRZEPŁYW DOKUMENTU

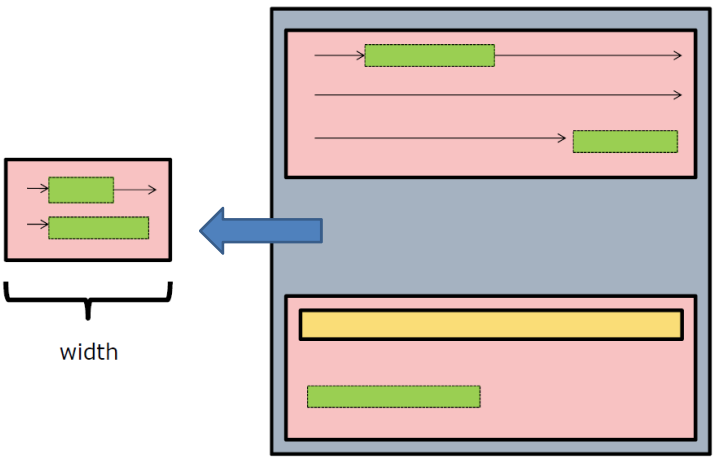
Wyjęcie z przepływu treści

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

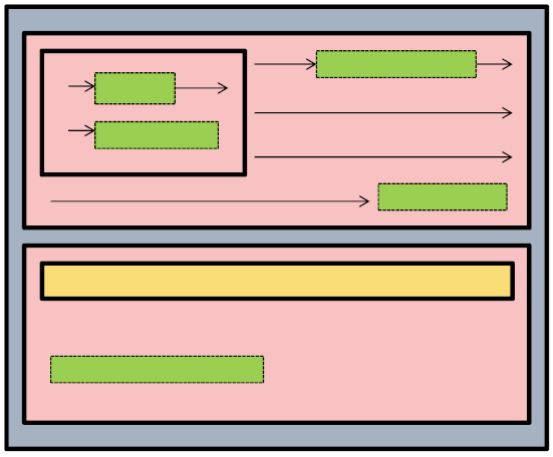
Obiekty wyjęte z biegu dokumentu najczęściej umieszczane są w innym miejscu strony przez **pozycjonowanie absolutne** lub **float**.

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 – wyjecie z normalnego przepływu

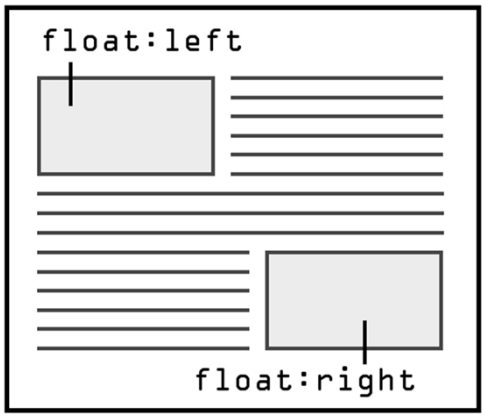


Przepływ dokumentu



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.



Pozycjonowanie pływające - przykład

Element jest przesuwany do lewej lub prawej strony bieżącej linii a zawartość go *opływa*. Niech taki blok nazywa się umownie *pływający*.

```
div#block
{
  padding: 5px;
  margin: 1em;
  . . .
}

div#fl_blk
{
  float : right;
  width : 30%;
  padding: 3px;
  margin: 5px;
  . . .
}

<div id="block">
  <div id="fl_blk">Blok pływający</div>
  Mechanizm stylów ma: odseparować zawartość dokumentu od sposobu
  jego formatowania . . .
</div>
```

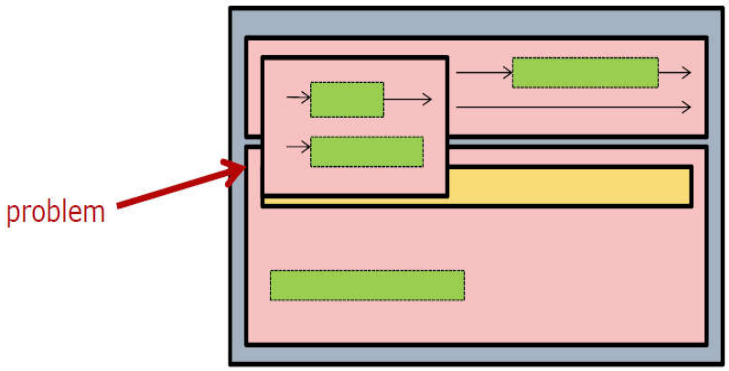
Blok pływający

Mechanizm stylów ma: odseparować zawartość dokumentu od sposobu jego formatowania, zapobiec niekontrolowanemu rozszerzaniu zestawu znaczników języka HTML o znaczniki formatujące, dodawane w sposób arbitralny przez producentów przeglądarek.

Zasady pozycjonowania pływającego

- **Float może być tylko po lewej lub prawej**. Nie ma możliwości umieszczenia na środku strony obiektu opływanego przez tekst.
- Środkowanie obiektów bez opływania przez tekst robi się za pomocą margin:auto.
- Float zawsze trzeba używać w połączeniu z elementami mającymi regułę **clear** albo **overflow**, inaczej efekty jego działania będą trudne do przewidzenia.
- tekst i elementy z display: inline zawsze je opływają,
- są wyjęte z normalnego biegu dokumentu — nie muszą być na ekranie w takiej kolejności, w jakiej są w dokumencie oraz mogą wystawać poza dolną krawędź obiektu, w którym się znajdują,
- tworzą swój własny bieg dokumentu, przez co kilka elementów z float nigdy nie będzie się nakładało (w przeciwieństwie do pozycjonowania absolutnego), tylko ułożą się obok siebie.

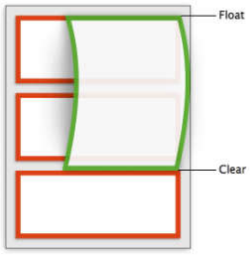
Przepływ dokumentu - opływanie



ZAPOBIEGANIE OPŁYWANIU (CLEAR)

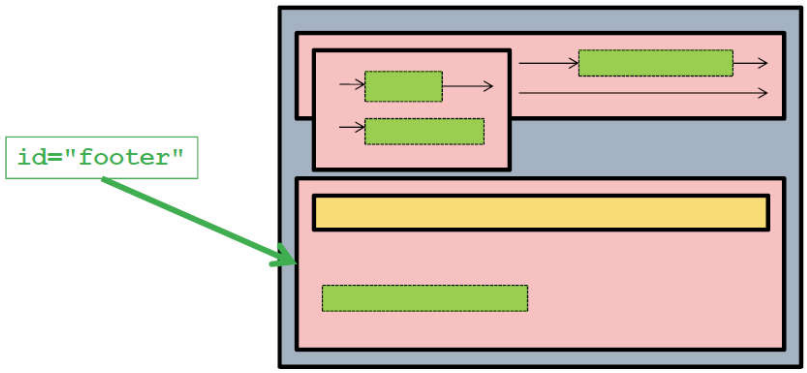
Przerywanie przez clear

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 jakiemuś elementowi za elementem z float nadać właściwość clear — powoduje ona ponowne „złączenie" float z biegiem dokumentu.

Wykorzystanie właściwości clear




```
#footer { clear: left; }
```

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.

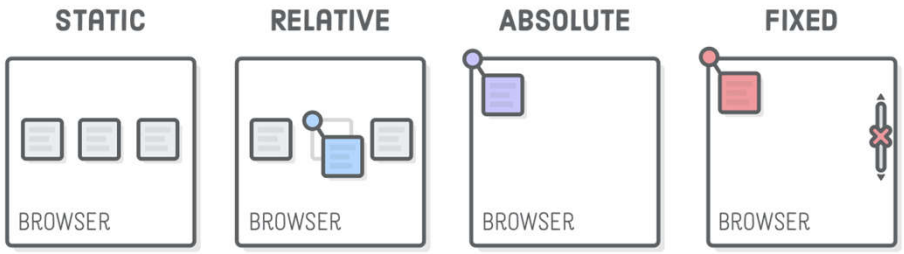
- visible - domyślne
- hidden - zakrywanie nie mieszczącej się zawartości w zadeklarowanych ramach
- scroll - przewijanie
- auto - w razie konieczności pojawią się przewijanie

Nadwiar tekstu w tym bloku zostanie przycięty.

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

Pełny tekst będzie można

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 umieszczania, 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.:

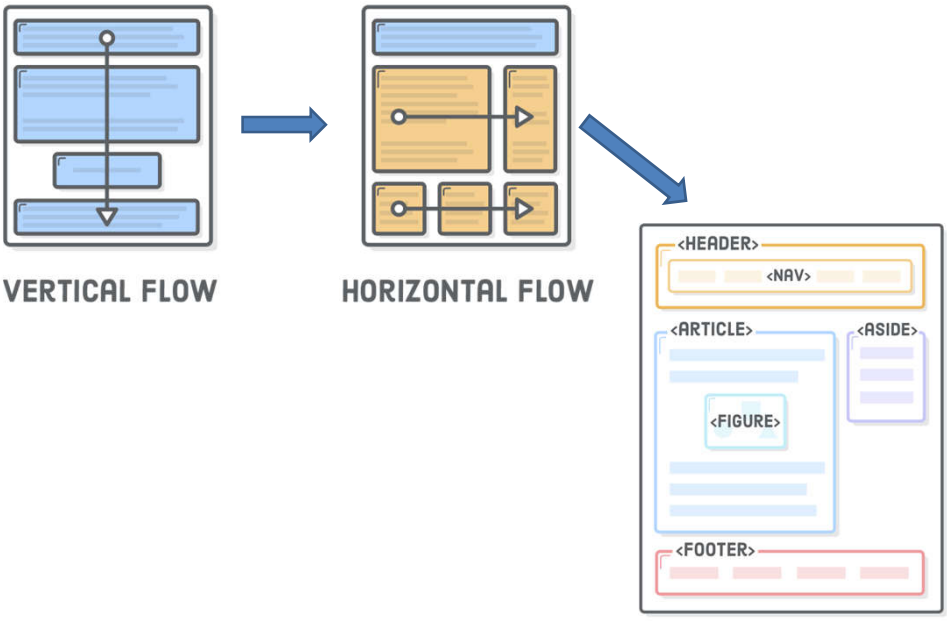
  div { position:relative; left:30px; }

  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; }

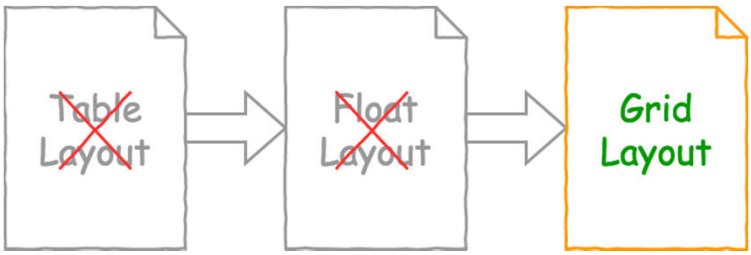
  ustawi element w odległości 30px od krawędzi elementu nadrzędnego.

Tworzenie Layout (Układ strony)



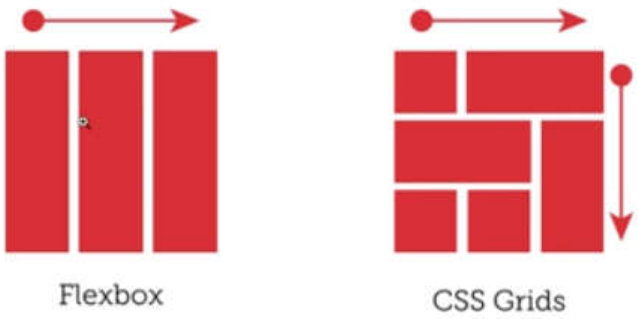
Tworzenie layout

- Floats
- Inline-block
- ~~display: table~~
- Absolute & Relative pozycjonowanie
- Frameworki …



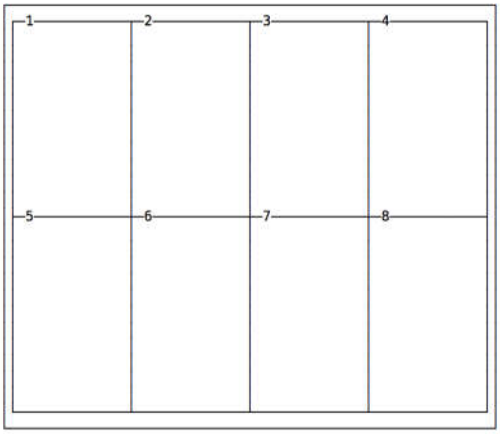
Nowsze sposoby

- Flexbox https://drafts.csswg.org/css-flexbox/
- CSS Grid Layout https://drafts.csswg.org/css-grid/



CSS grid

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



1

```
.cards {
  display: grid;


}
```

```
.cards {
  display: grid;
  grid-template-columns: 250px 250px 250px;
  grid-template-rows: 200px 200px 200px;

}
```

2

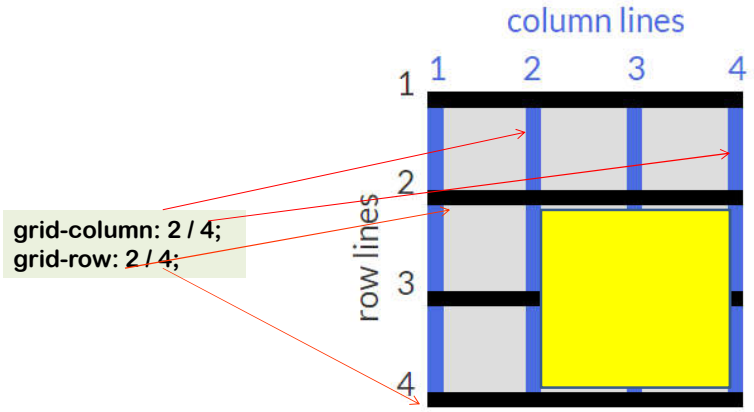
```
.cards {
  display: grid;
  grid-template-columns: 250px 250px 250px;
  grid-template-rows: 200px 200px 200px;
  grid-gap: 20px;
}
```

3

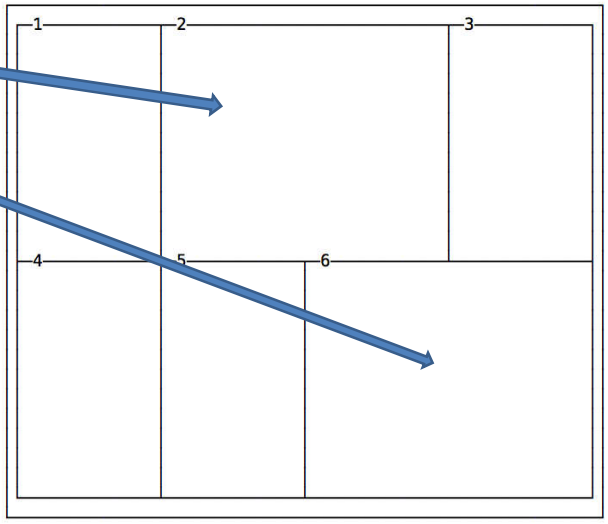
```
.cards {
  display: grid;
  grid-template-columns: 1fr 1fr 2fr;
  grid-template-rows: 200px 200px 200px;
  grid-gap: 20px;
}
```

4

Łączenie komórek

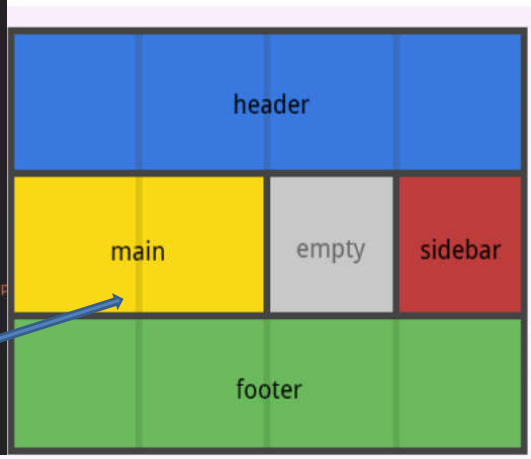



```
.container {
  display: grid;
  grid-template-columns: 200px 200px 200px 200px;
  grid-template-rows: 300px 300px;
}
.item1 {
  grid-column-start: 2;
  grid-column-end: 4;
}
.item6 {
  grid-column-start: 3;
  grid-column-end: 5;
}
```



GRID – budowa układu strony

```
.item-a {
  grid-area: header;
}
.item-b {
  grid-area: main;
}
.item-c {
  grid-area: sidebar;
}
.item-d {
  grid-area: footer;
}

.container {
  grid-template-columns: 50px 50px 50px 50p
  grid-template-rows: auto;
  grid-template-areas:
    "header header header header"
    "main main . sidebar"
    "footer footer footer footer";
}
```




```
.wrapper {
  display:grid;
  grid-template-columns: 1fr 3fr;
  grid-template-rows: auto;
  grid-gap: 20px;
  grid-template-areas:
   "aa aa"
   "sidebar content"
   "stopka stopka";
}
header {
  background-color: red;
  grid-area: aa;
  padding: 20px;}
article {
  background-color: blue;
  grid-area: content;
  padding: 20px;}
aside {
  background-color: yellow;
  grid-area: sidebar; }
footer {
  background-color: green;
  grid-area: stopka; }
```

```
<div class="wrapper">
   <header>
      To jest naglowek
   </header>
   <article>
      <h1> Przyklad szablonu typu grid </h1>
      <p> Zawartosc strony glownej</p>
   </article>
   <aside>
     <ul>  Lista reklam
       <li> reklama 1 </li>  <li> reklama 2 </li>  <li> reklama 3 </li>
     </ul>
   </aside>
   <footer>  To jest stopka  create by GR - WdAI</footer>
</div>
```

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**