

Elementy pływające

Najstarsza i najprostsza metoda, która pozwalała na dodawanie do stron elementów częściowo responsywnych.

Obecnie **nie polecam** – chyba, że do wykonywania prostych elementów.



Elementy pływające

- ✓ Element pływający pozostaje poza standardowym układem elementów na stronie.
- ✓ Elementy znajdujące się za elementem pływającym przesuваются się do góry, układając się obok niego, jeśli jest tam wystarczająco dużo miejsca.
- ✓ Własność `clear` pozwala określić, czy elementy znajdujące się za elementem pływającym mają przesuwać się do góry czy nie.
 - Jeśli na przykład mamy dwa akapity i chcemy, aby tylko pierwszy z nich pojawił się obok elementu pływającego, możemy ten drugi zatrzymać pod elementem I pływającym za pomocą własności `clear`.



Własność **float**

- ✓ Jednym z zastosowań własności **float** jest otaczanie obrazów tekstem.

Zacniemy od przypomnienia zasad otaczania obrazów tekstem.

`img {float:left; margin:0 4px 4px 0;}`

Powyższa reguła spycha obraz na lewa dzięki czemu tekst będzie znajdował się z jego prawej strony.

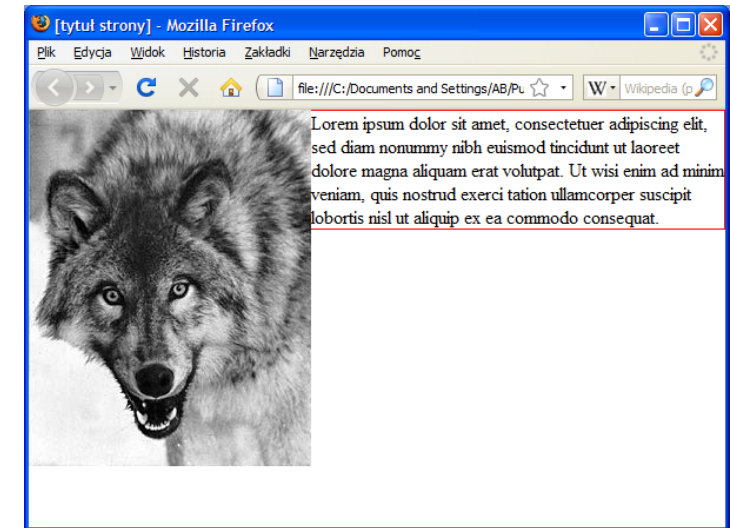
Aby własność float zadziałała poprawnie, kod XHTML musi wyglądać następująco:

``
`<p>...tekst akapitu...</p>`

Własność **float**



- ✓ Innymi słowy, używając własności **float**, żądamy, aby element został przesunięty jak najdalej w lewo (lub prawo w przypadku deklaracji **float: right**) w obrębie zawierającego go elementu — w tym przypadku body.
- ✓ Akapit (z przykładu na poprzednim slajdzie) nie traktuje elementu pływającego jako bloku znajdującego się przed nim i dlatego również przesuwa się do lewego górnego rogu swojego rodzica. Jednak jego treść (tekst) zawija się wokół pływającego obrazu.

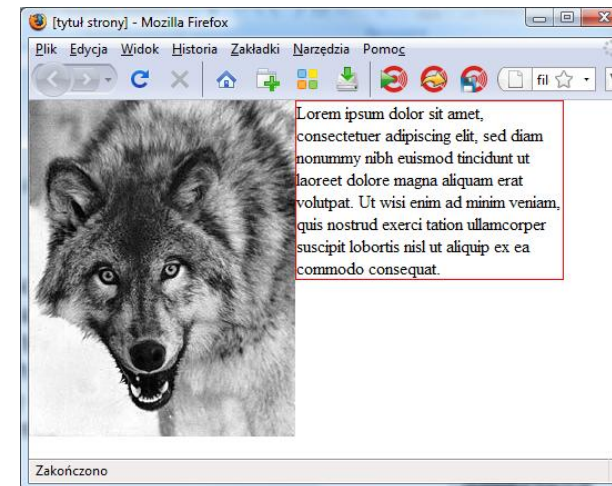


Własność **float**



- ✓ Zastosowanie własności float zarówno do obrazu, jak i akapitu (o ustalonych szerokościach) powoduje, że tekst przestaje owijać się wokół obrazu. Jest to jedna z głównych zasad tworzenia układów kolumnowych przy użyciu elementów pływających. Elementy ustawiają się obok siebie jak kolumny, jeśli mają ustaloną szerokość i jest dla nich wystarczająco miejsca

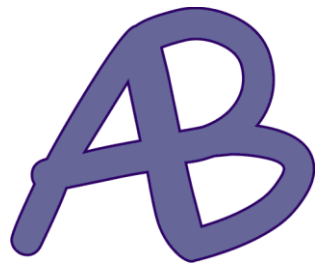
img {float:left; margin:0 4px 4px 0;}





Własność **clear**

- ✓ Z własnością **float** zazwyczaj współwystępuje własność **clear**. Jeżeli jeden element jest pływający, inny — jeśli starczy dla niego miejsca — ustawi się obok niego. Czasami jednak nie chcemy, aby tak się stało. Wolimy, aby ten drugi element pozostał pod elementem pływającym.
- ✓ Rozwiązaniem w tym przypadku jest dodanie niepływającego elementu do kodu HTML i ustawienie jego własności **clear** w celu zatrzymania ostatniego elementu na dole.



Pozycjonowanie elementów



Własność **position**

- ✓ W CSS własność **position** pozwala zdefiniować punkt odniesienia, względem którego element ma być pozycjonowany na stronie.
- ✓ Własność **position**: może przyjmować jedną z czterech wartości:
 - **static,**
 - **absolute,**
 - **fixed ,**
 - **relative.**

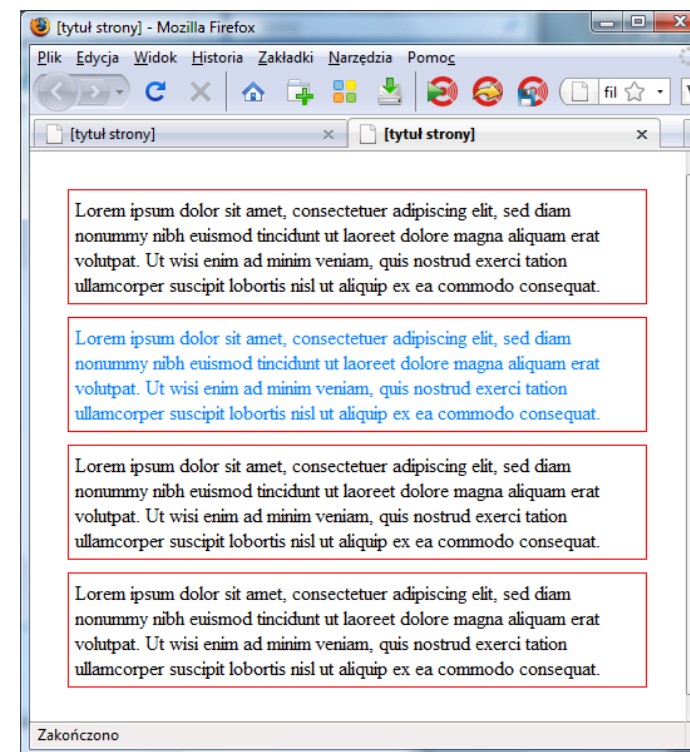
Domyślna jest pierwsza z wymienionych.

Pozycjonowanie statyczne



Pozycjonowanie statyczne polega na ułożeniu elementów jeden pod drugim.

Odległość między nimi jest równa ich domyślnym marginesom.



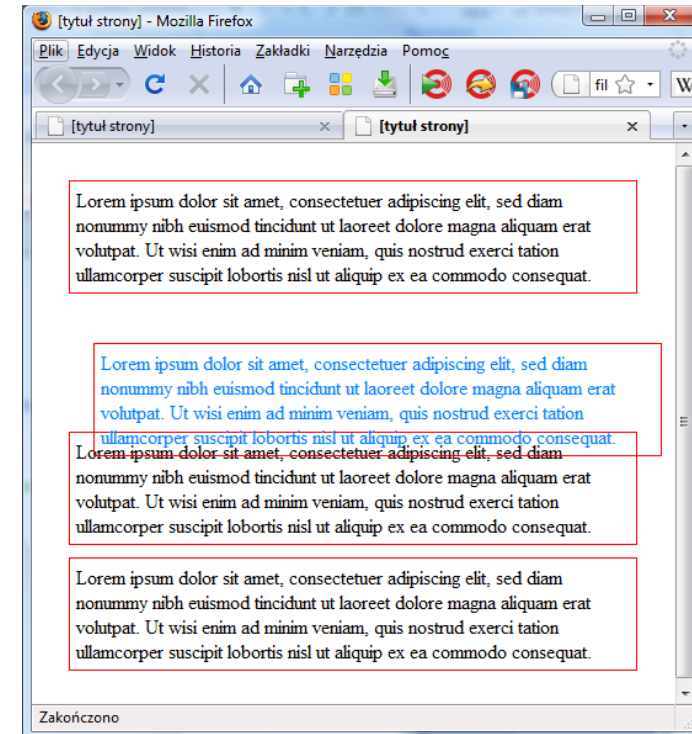
Pozycjonowanie względne



Ustawiamy własność **position** wyróżnionego kolorem akapitu na wartość **relative**.

Dzięki temu możemy przesuwać ten element względem jego domyślnego położenia za pomocą własności **top**, **right**, **bottom** i **left**.

Zazwyczaj wystarczą tylko wartości **top** i **left**.



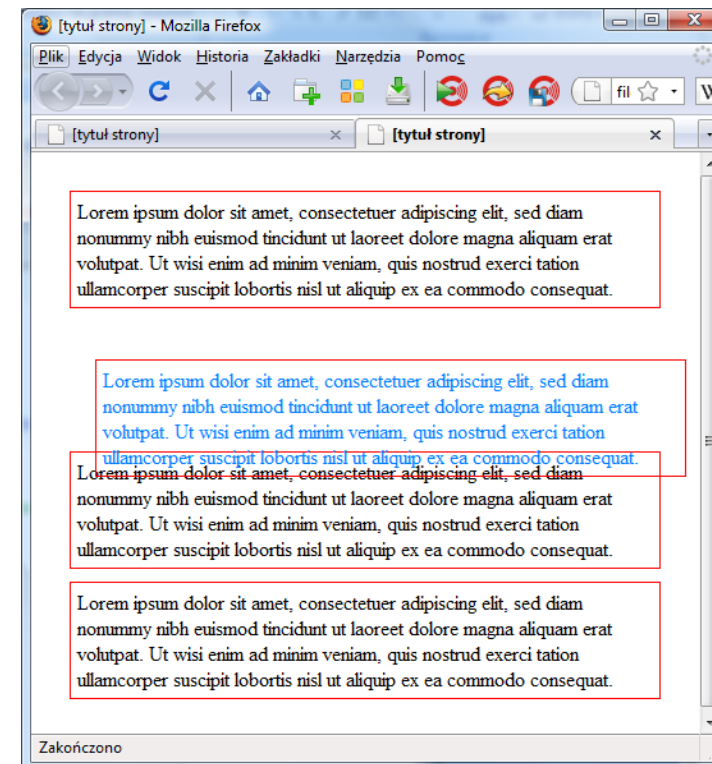
```
position: relative;  
top: 30px;  
left: 20px;
```

Pozycjonowanie względne



Lewy górny róg akapitu został przesunięty o 30 pikseli w dół i 20 pikseli w prawo. Jak widać, mimo że trzeci akapit został przesunięty, nic więcej na stronie się nie zmieniło. Miejsce zajmowane przez ten akapit, kiedy był pozycjonowany statycznie, nie zostało zwolnione. Podobnie pozostałe elementy — nadal są na swoich pierwotnych miejscach.

Należy pamiętać, że przesuwając element w ten sposób, trzeba wcześniej wygospodarować dla niego miejsce.



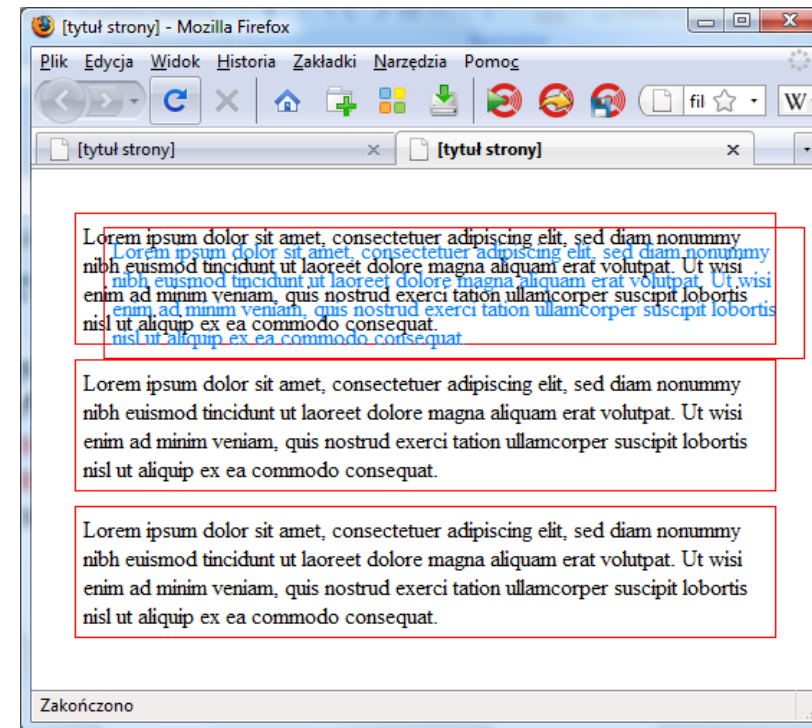


Pozycjonowanie bezwzględne

Pozycjonowanie bezwzględne pozwala całkowicie wytrącić element z normalnego układu dokumentu.

Jak widać na rysunku, miejsce wcześniej zajmowane przez drugi akapit zostało zajęte przez kolejny. Element pozycjonowany bezwzględnie jest całkowicie niezależny od innych otaczających go w kodzie HTML elementów.

Jego położenie jest obliczane względem elementu viewportu.



```
position: absolute;  
top: 30px;  
left: 40px; }
```



Pozycjonowanie bezwzględne

W tym miejscu musimy zapoznać się z pojęciem **kontekstu pozycjonowania**.

- ✓ Domyślnym kontekstem pozycjonowania elementu pozycjonowanego bezwzględnie jest element **Document Containing Bloc**, w uproszeniu ma on rozmiar **Viewportu**, czyli widocznej części strony.
- ✓ Element spozycjonowany bezwzględnie wychodzi poza swój kontener.
- ✓ Jak widać na rysunku z poprzedniego slajdu, własności `top` i `left` przesunęły akapit względem viewportu, zamiast względem jego domyślnej pozycji w dokumencie.
- ✓ Ponieważ kontekstem pozycjonowania elementu pozycjonowanego bezwzględnie jest viewport, pozycjonowany element przesuwa się w miarę przewijania strony, aby cały czas pozostać w tym samym miejscu. Element `body` także przesuwa się w miarę przewijania strony



Kontekst pozycjonowania

Pozycjonowanie kontekstowe polega na przesuwaniu danego elementu względem innego za pomocą własności **top**, **right**, **left** i **bottom**. Ten drugi element jest właśnie **kontekstem pozycjonowania**.

Kontekstem pozycjonowania jest pierwszy element rodzic który nie posiada własności **position: static**; Ponieważ jest to domyślna wartość Kontekstem pozycjonowania staje się zwykle element `<body>`

Kontekstem pozycjonowania może być jednak dowolny element będący przodkiem innego elementu, jeśli jego własności **position** nada się wartość **relative**.



Kontekst pozycjonowania

Przyjrzyjmy się poniższemu kodowi:

HTML:

```
<div class="zewnetrzny">  
  <div class="wewnetrzny">Lorem ipsum .... </div>  
</div>
```

CSS:

```
.zewnetrzny {  
  width:250px;  
  margin:50px 40px;  
  border-top: 10px solid red;  
}  
.wewnetrzny {  
  top:10px;  
  left:20px;  
  background:#0ff;  
}
```

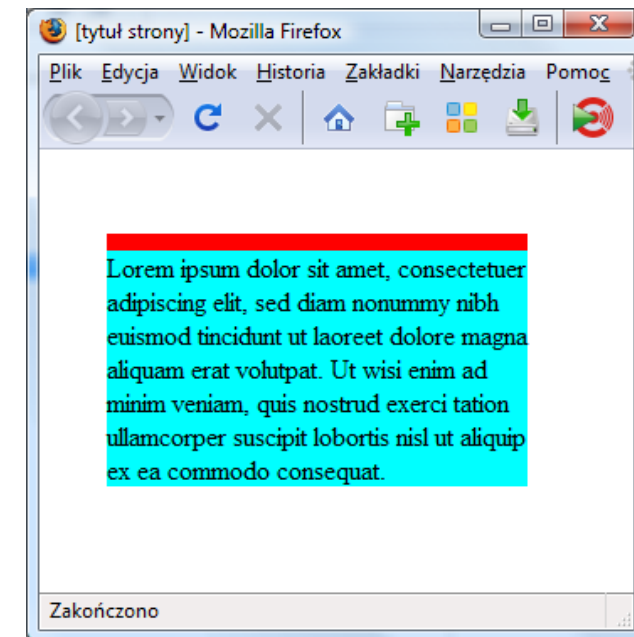
Kontekst pozycjonowania



- ✓ Nasuwa się pytanie: dlaczego wewnętrzny element div nie jest odsunięty o 10 pikseli w dół i 20 pikseli w lewo względem zewnętrznego elementu div?
- ✓ Zamiast tego górne lewe rogi obu tych elementów znajdują się w tym samym punkcie.

Powodem jest pozycjonowanie statyczne obu elementów. Oznacza to że elementy te wpasowują się w domyślny rozkład elementów na stronie.

- ✓ Ponieważ zewnętrzny element nie ma żadnej treści, element wewnętrzny zaczyna się w tym samym miejscu.
- ✓ Własności top, right, bottom i left działają tylko jeśli element jest pozycjonowany względnie, bezwzględnie lub w sposób stały

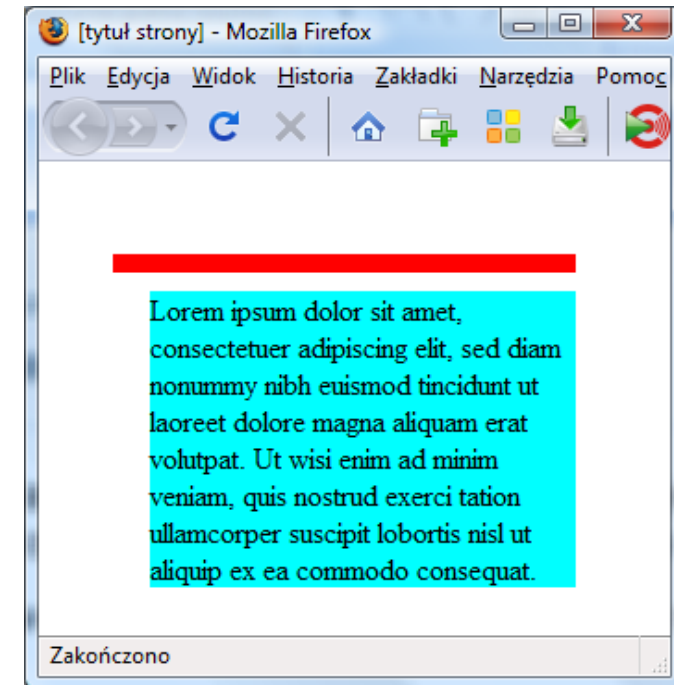


Kontekst pozycjonowania



Aby zewnętrzny kontener div stał się kontekstem pozycjonowania dla kontenera wewnętrznego należy:

1. Ustawić **position: absolute** dla kontenera wewnętrznego
2. Ustawić **position: relative** dla kontenera zewnętrznego





Pozycjonowanie stałe

Pozycjonowanie stałe (**fixed**) jest podobne do bezwzględnego. Różnica polega na tym, że element jest pozycjonowany w odniesieniu do okna przeglądarki. Dzięki temu dany element nie przesuwa się w miarę przewijania strony.

Element pozycjonowany w ten sposób nie przesuwa się podczas scrollowania elementu body.



Pozycjonowanie sticky

position : sticky jest miksem pomiędzy ustawieniem fixed i relative. Element jest traktowany jako relative, czyli skroluje razem ze stroną dopóki pewna wartość wysokości strony nie zostanie przekroczona. Od tego momentu traktowany jest jako element jako pozycjonowany względem viewportu - fixed

Pozycjonowanie sticky



Aktualne wsparcie dla sticky

IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
			4-22													
			23-36													
	12-15	2-25	37-51	3.1-6	10-38	3.2-5.1										
	16-18	26-31	52-55	6.1-7	39-41	6-7.1										
	79-90	32-58	56-90	7.1-12.1	42-77	8-12.5										
6-10	91-100	59-99	91-100	13-15.3	78-85	13-15.3		2.1-4.4.4	12-12.1				4-5.4			
11	101	100	101	15.4	86	15.4	all	101	64	101	100	12.12	6.2-15.0	10.4	7.12	2.5
		101-102	102-104	TP	87											

<https://caniuse.com>



Kontekst **display**

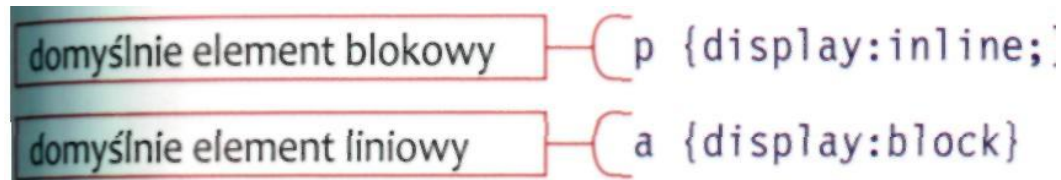
Poza własnością **position** każdy element ma także własność **display**. Mimo iż własność ta może przyjmować kilka wartości, najczęściej używane są dwie: **block** i **inline**. Tym, którzy przespali poprzednie wykłady, przypominam różnicę między elementami blokowymi (**block**) a liniowymi (**inline**):

- ✓ Elementy blokowe, na przykład akapity, nagłówki czy listy, układają się jeden nad drugim w oknie przeglądarki.
- ✓ Elementy liniowe, na przykład **a** oraz **span**, układają się jeden obok drugiego w oknie przeglądarki. Przechodzą do nowej linii dopiero wówczas, gdy w aktualnej nie ma dla nich wystarczająco dużo miejsca.

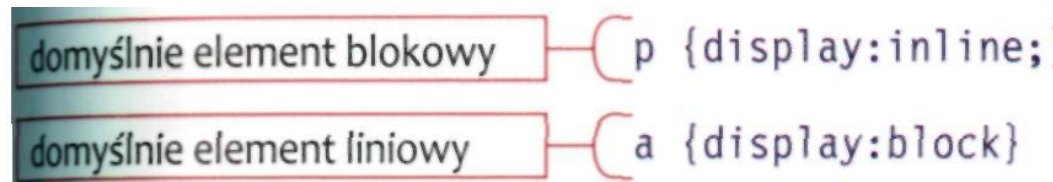


Kontekst **display**

Możliwość zamiany elementów blokowych w liniowe i odwrotnie, jak poniżej:



Możliwość zamiany elementów blokowych w liniowe i odwrotnie, jak poniżej:





Kontekst **display**

Jeszcze jedną wartością własności **display**, o której warto wspomnieć, jest **none**.

- ✓ Powoduje ona, że element i wszystkie zagnieżdżone w nim elementy stają się niewidoczne na stronie.
- ✓ Miejsce normalnie zajmowane przez ten element nie jest wtedy przez niego zajmowane. Wygląda to tak, jakby kod HTML tego elementu w ogóle nie istniał (jest jeszcze własność **visibility**, której wartość **hidden** powoduje, że element jest niewidoczny, ale nadal zajmuje przeznaczone dla niego miejsce).



Literatura:

W prezentacji wykorzystano fragmenty i przykłady z książki:

- Wyke-Smith Ch.; *CSS Witryny szyte na miarę*. Helion, Gliwice 2008.