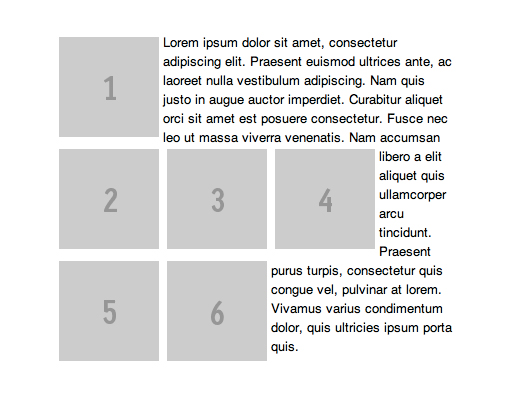
Подробно о свойстве float

[Автор оригинала: Joshua Johnson](http://designshack.net/articles/css/everything-you-never-knew-about-css-floats/" \o "Автор оригинала: Joshua Johnson" \t "_blank)

* [Разработка веб-сайтов](https://habr.com/ru/hub/webdev/),
* [CSS](https://habr.com/ru/hub/css/)
* Перевод

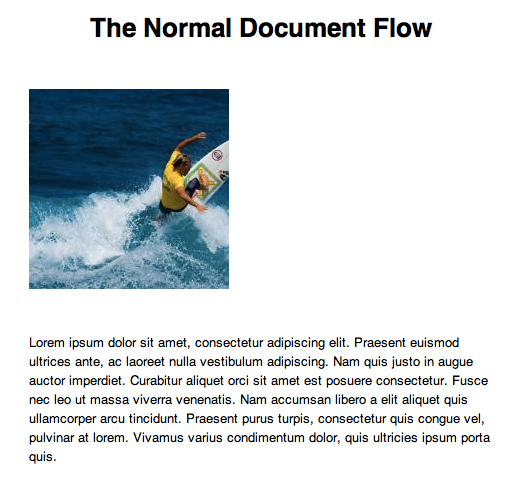
Правильное использование CSS свойства float может стать непростой задачей даже для опытного верстальщика. В этой статье собраны варианты применения float, а также некоторые ошибки, с наглядными примерами.  


Что такое float?

Некоторые элементы в CSS являются блочными и поэтому начинаются с новой строки. Например, если расположить два абзаца с тегами P, то они будут находиться друг под другом. Другие же элементы являются «строчными», т.е. отображаются на странице в одну строку.  
  
Один из способов переназначить элементам тип обтекания — это использование свойства float. Классический пример — использование float для выравнивания картинки по левому или правому краю. Вот простой HTML-код картинки и абзаца:

<img src="http://lorempixum.com/200/200/" />

<p>Lorem ipsum...</p>

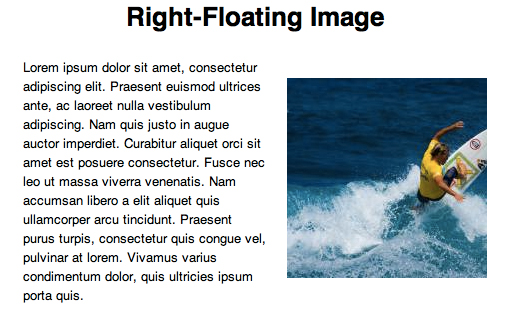
Они отображаются с новой строки:  
  
  
  
Добавляем немного CSS картинке:

img {

float: right;

margin: 20px;

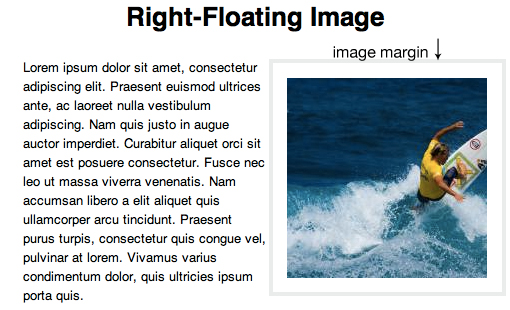
}

Получается выравнивание по правому краю:  
  
  
  
Если текста больше, то абзац будет обтекать картинку:  
  
  
  
Допустим, нам нужно сделать отступ в 20 пикселей между картинкой и текстом. Вот такая конструкция работать не будет:

p {margin: 20px;}

Правильно вот так:

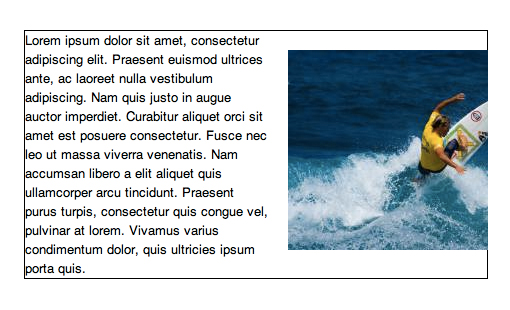
img {margin: 20px;}

  
  
Почему же не работает отступ для абзаца? Чтобы разобраться, добавим рамку:

p {

border: solid 1px black;

}

Результат может вас удивить:  
  
  
  
Оказывается, картинка находится внутри абзаца! Поэтому и не работает свойство margin в первом случае. Чтобы исправить это, можно применить float:left к абзацу и указываем абсолютную ширину:

img {

float: right;

margin: 20px;

}

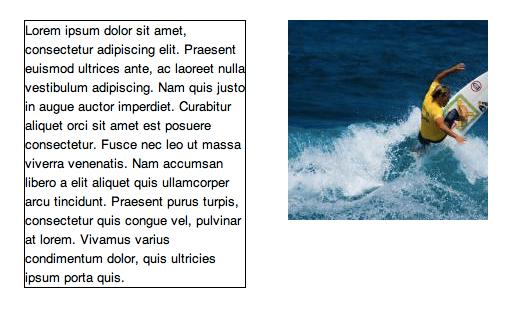
p {

float: left;

width: 220px;

margin: 20px;

}



Странные правила float

Перейдем к более сложным вариантам использования float: к правилам, регулирующим плавающие объекты. Это бывает необходимо при верстке галереи изображений. Например:

<ul>

<li><img src="1450823466601083332032"/></li>

<li><img src="http://placehold.it/100x150&text=2"/></li>

<li><img src="1450823466394721548724"/></li>

<li><img src="1450823466376067888878"/></li>

<li><img src="1450823466279266032276"/></li>

<li><img src="http://placehold.it/100x150&text=6"/></li>

<li><img src="http://placehold.it/100x100&text=7"/></li>

</ul>

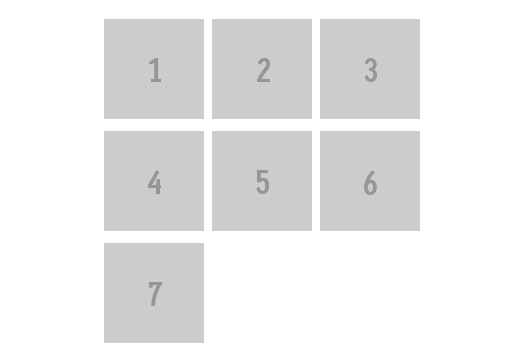
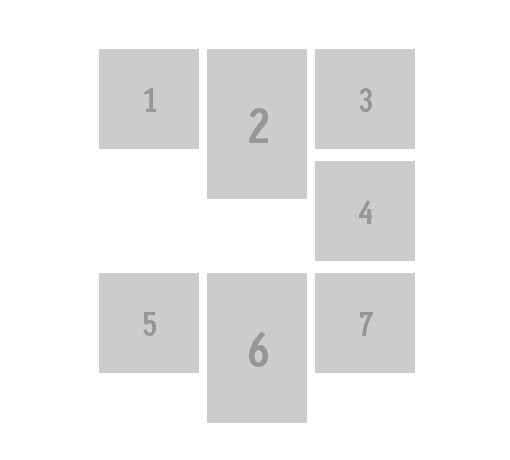
По умолчанию каждый элемент списка появится с новой строки. Если применить к каждому float:left, изображения встанут в один ряд с переносом строки:

li {

float: left;

margin: 4px;

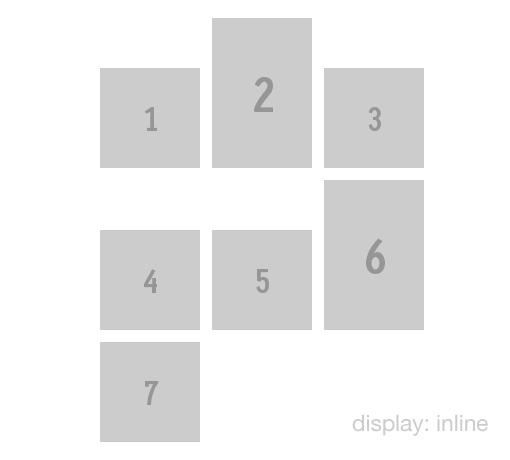
}

  
  
Но что, если изображения разной высоты?  
  
  
  
Если мы добавим элементам списка отображение в одну строку, получится немного симпатичнее:

li {

display: inline;

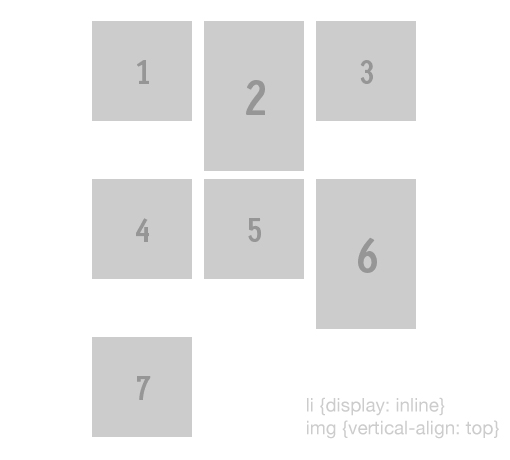
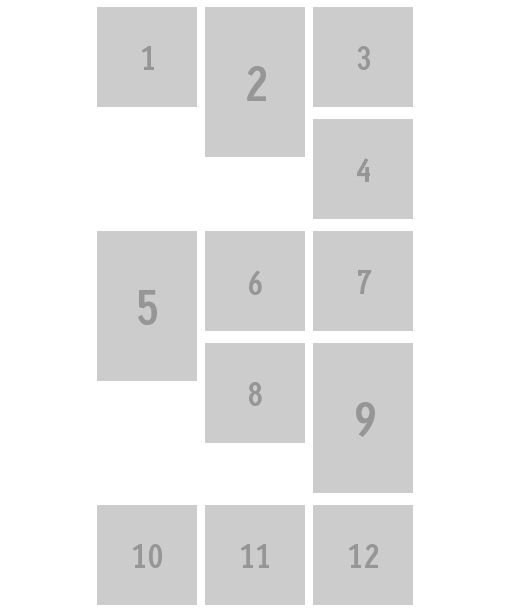
}

  
  
А теперь выравниваем по вертикали:

img {

vertical-align: top;

}

  
  
Следует помнить, что в случае большей высоты картинок, остальные изображения обтекают только одно предыдущее, например:  
  
  
  
Пример изменения очередности элементов — например, у нас есть список элементов по порядку:

<ul>

<li><img src="1450823466601083332032"/></li>

<li><img src="http://placehold.it/100x100&text=2"/></li>

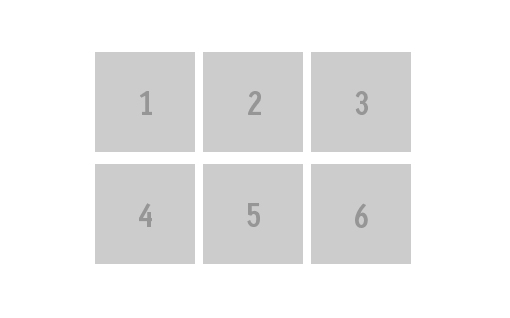
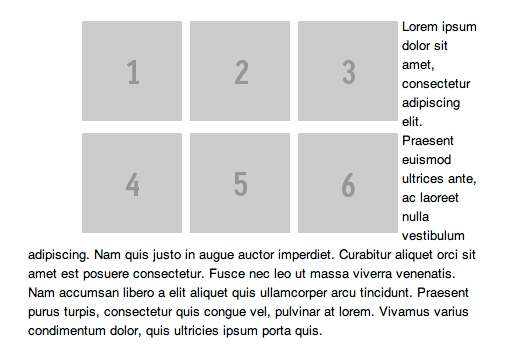
<li><img src="1450823466394721548724"/></li>

<li><img src="1450823466376067888878"/></li>

<li><img src="1450823466279266032276"/></li>

<li><img src="http://placehold.it/100x100&text=6"/></li>

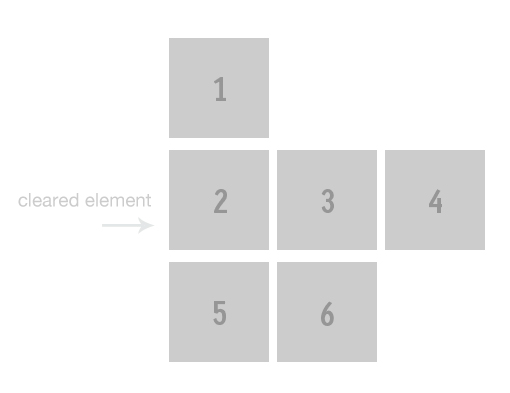
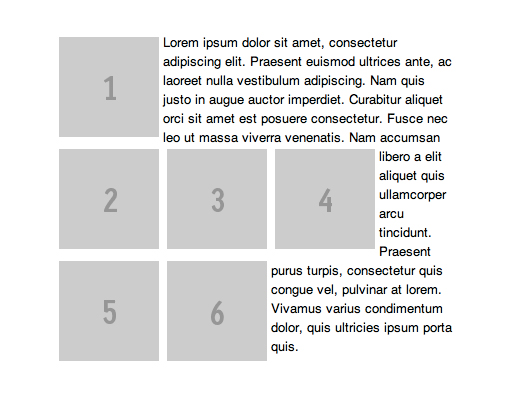
</ul>

Если мы хотим расположить их в обратном порядке, просто применяем float:right вместо float:left, и не придется менять порядок в HTML:  
  
  
  
С помощью float удобно группировать элементы на странице, но большой проблемой становится то, что последующие элементы (текст или блок) также получают свойство обтекания. Например, у нас есть блок картинок:  
  
  
  
Текст под ним начинает обтекать весь блок:  
  
  
  
Чтобы избежать этого, необходимо использовать свойство clear. Если применим его ко второму изображению:

ul li:nth-child(2) {

clear: left;

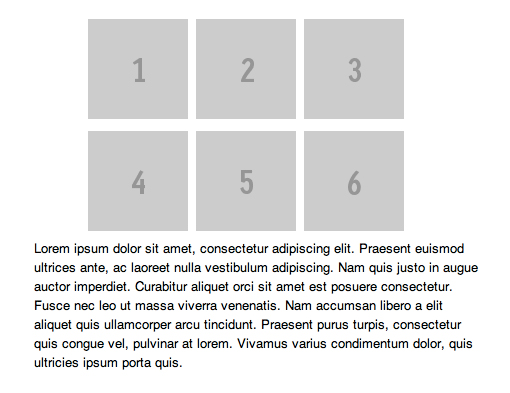
}

Получим вот, что:  
  
  
  
В этом случае остальные изображения продолжают наследовать float:left. Соответсвенно текст будет отображаться коряво:  
  
  
  
Нужно применить clear:both к абзацу:

p {

clear: both;

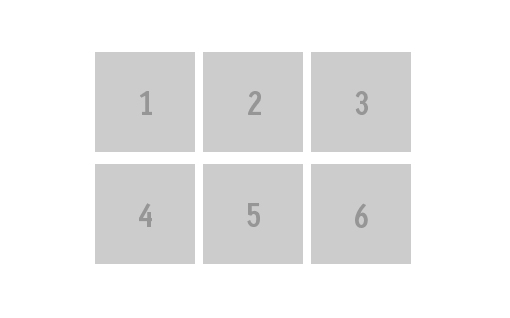
}

Наша проблема решена:  
  
  
  
Теперь предположим, что нам необходимо установить фон для галереи из предыдущих примеров. Если бы элементы были не плавающими, то получилось бы вот, что:

ul {

background: gray;

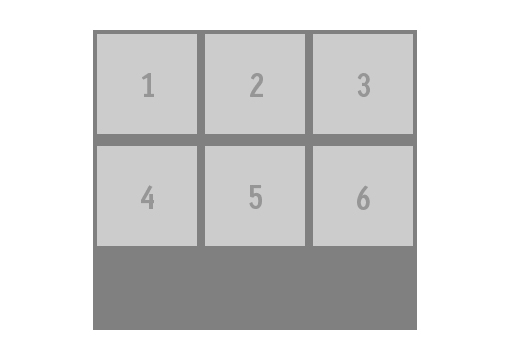
}

  
  
Но если элементам списка применить float:left, фон совсем пропадает:  
  
  
  
Если мы сначала устанавливаем высоту для UL:

ul {

height: 300px;

}

  
  
Это тоже не решает проблему, т.к. размеры фона заданы абсолютно. Нам поможет класс clearfix, который будет применен к div на том же уровне, что и элементы UL.

.clearfix {

clear: both;

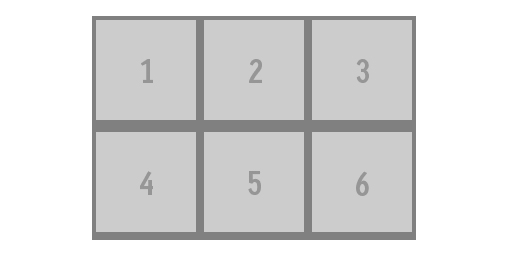
}

Существует еще одно решение, с использованием overflow:

ul {

overflow: auto;

}



Девять правил float-элементов:

1. Плавающие элементы не могут выходить за край своего контейнера-родителя.
2. Каждый плавающий элемент будет отображаться справа или ниже от предыдущего при float:left, либо слева и ниже, при float:right.
3. Блок с float:left не может быть правее, чем блок с float:right.
4. Плавающий элемент не может выходить за пределы верхней границы своего контейнера.
5. Плавающий элемент не может располагаться выше, чем родительский блок или предыдущий плавающий элемент.
6. Плавающий элемент не может располагаться выше, чем предыдущая строка inline-элементов
7. Плавающий блок должен быть расположен как можно выше.
8. Один плавающий элемент, следующий за другим, не может выходить за пределы своего контейнера — происходит перенос на следующую строку.
9. Блок с float:left должен быть расположен как можно левее, а с float:right — как можно правее.