CSS-позиционирование

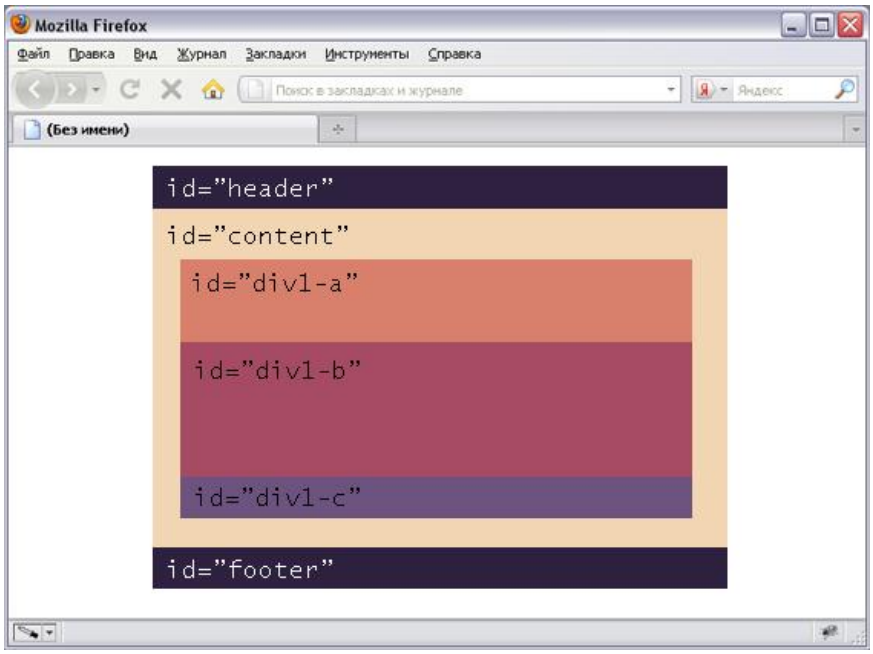
Позиционирование — одно из ключевых понятий в блочной верстке. Разобравшись с ним, вам многое станет понятно, а верстка из шаманства превратится в осмысленный процесс. Итак, речь в статье пойдет о CSS-свойствах *position* и *float*.

position:static

По умолчанию все элементы на странице имеют статическое позиционирование (position: static), это означает, что элемент не позиционирован, и появляется в документе на своем обычном месте, то есть в том же порядке, как и в html-разметке.

Нет необходимости специально назначать это свойство какому-либо элементу, если только вам не требуется изменить ранее установленное позиционирование на дефолтное.

```
#content{
 position: static;
}
```

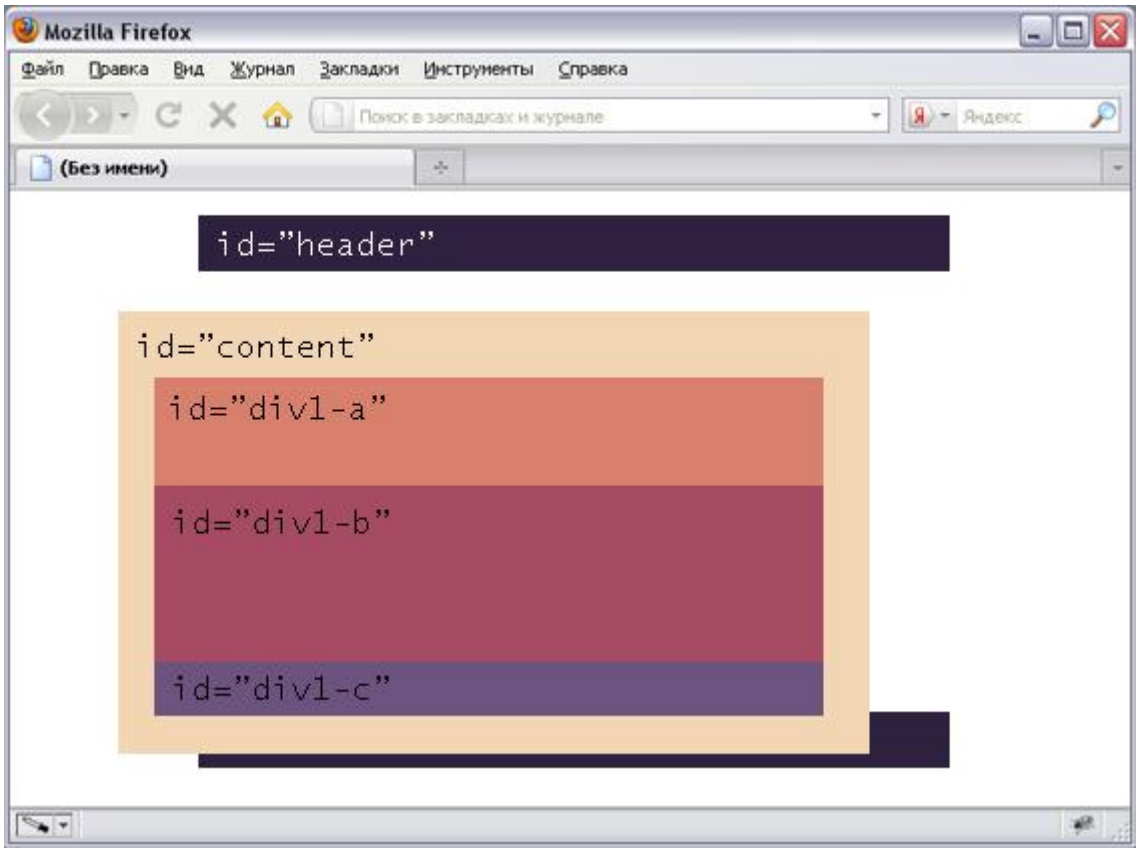


position:relative

Относительное позиционирование (position: relative) позволяет вам использовать свойства: top, bottom, left и right, для расположения элемента относительно того места, где бы он появился при обычном позиционировании.

Давайте переместим #content на 20 пикселей вниз, и на 40 пикселей влево:

```
#content{
	position: relative;
	top: 20px;
	left: -40px;
}
```



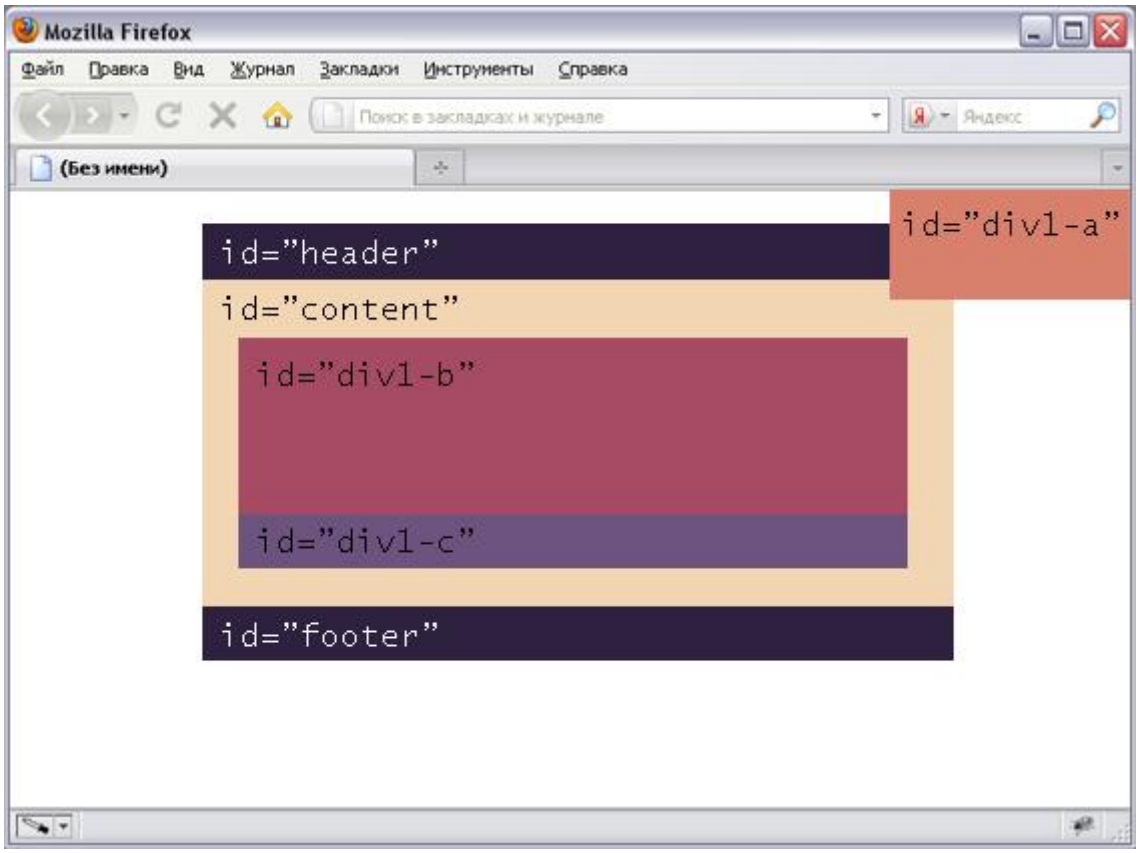
Обратите внимание, что на том месте, где бы должен был находится наш блок #content, теперь образовалось пустое пространство. Следующий за блоком #content, блок #footer не переместился ниже, потому что, #content по-прежнему занимает свое место в документе, несмотря на то, что мы передвинули его.

position: absolute

При абсолютном позиционировании (position: absolute), элемент удаляется из документа, и появляется там, где вы ему скажете.

Давайте, для примера, переместим блок #div-1a в верхний, правый угол страницы:

```
#div-1a {
	position:absolute;
	top:0;
	right:0;
	width:200px;
}
```



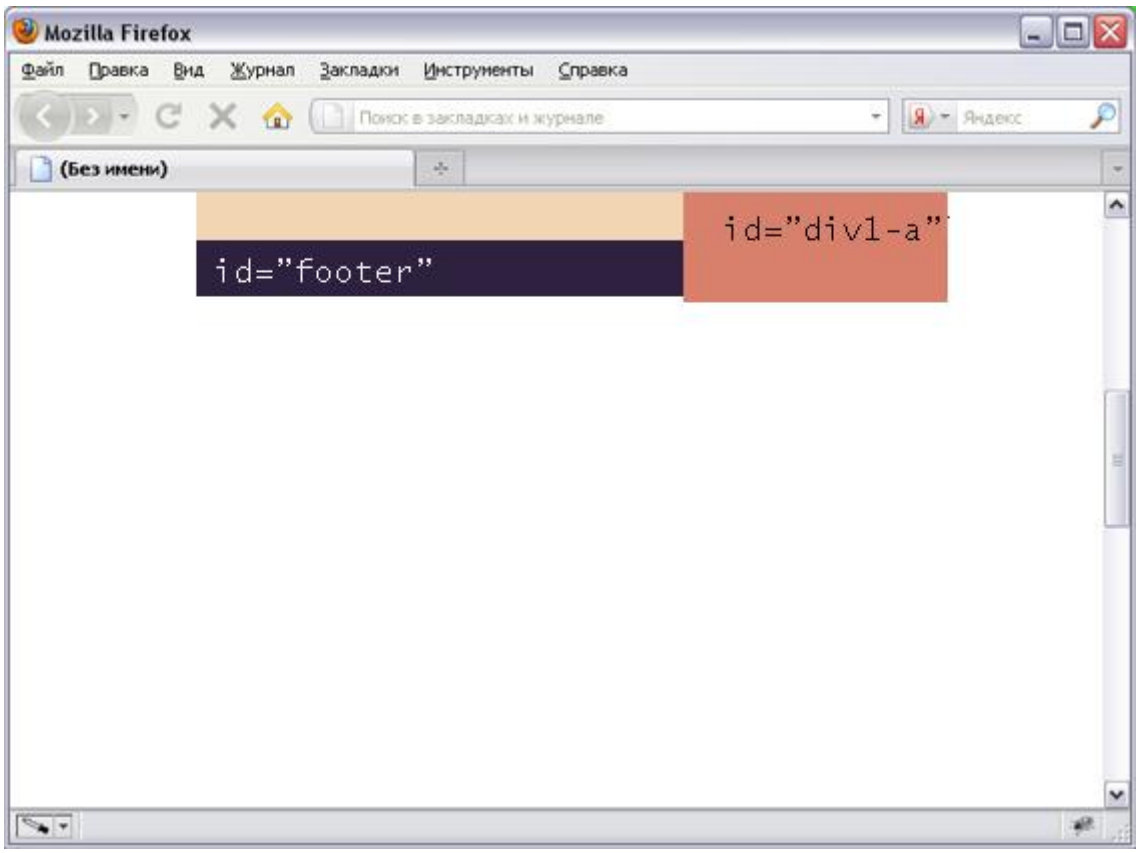
Обратите внимание, что на этот раз, поскольку блок #div-1a был удален из документа, оставшиеся элементы на странице расположились по-другому: #div-1b, #div-1c и #footer переместились выше, на место удаленного блока. А сам блок #div-1a, расположился точно в правом, верхнему углу страницы.

Таким образом, мы можем позиционировать любой элемент относительно страницы, однако этого недостаточно. На самом деле, нам необходимо позиционировать #div-1a относительно родительского блока #content. И на этом этапе, относительное позиционирование вновь вступает в игру.

position: fixed

Фиксированное позиционирование (position: fixed), является подразделом абсолютного позиционирования. Единственное его отличие в том, что он всегда находится в видимой области экрана, и не двигается во время прокрутки страницы. В этом отношении, он немного похож на фиксированное фоновое изображение.

```
#div-1a {
	position:fixed;
	top:0;
	right:0;
	width:200px;
}
```



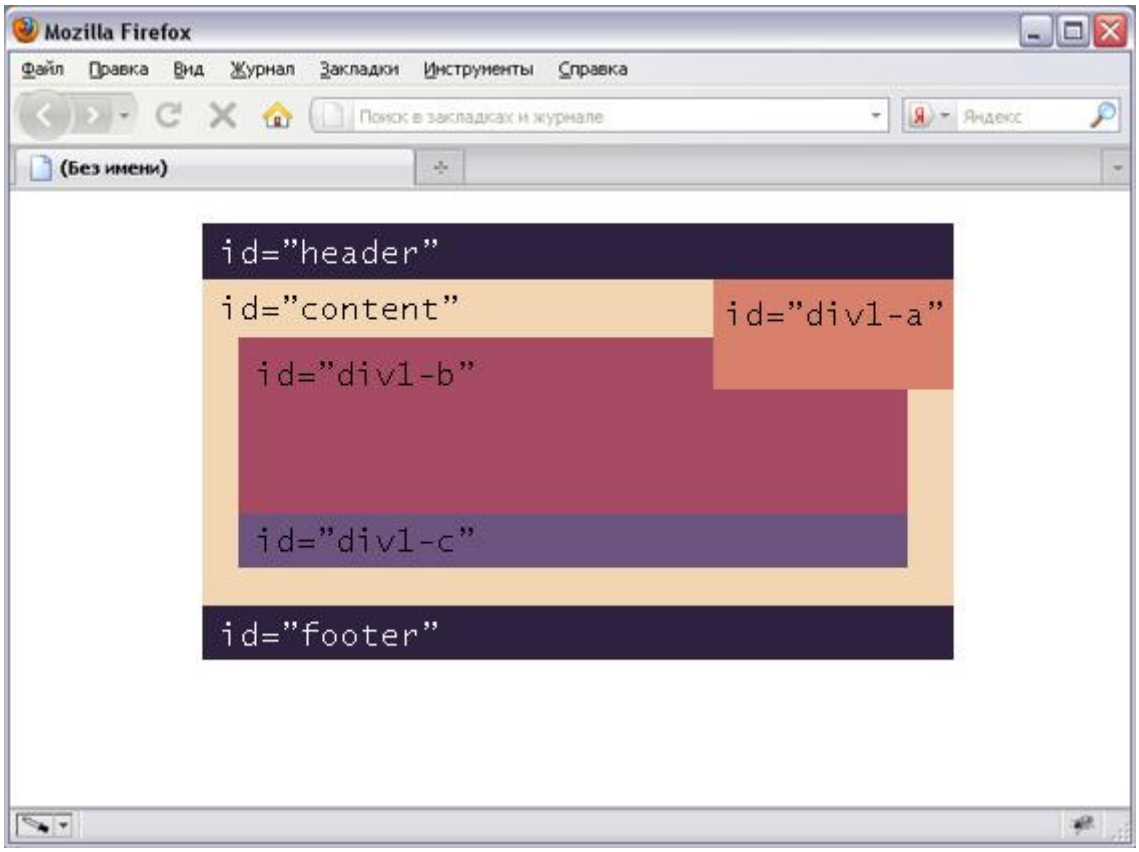
В IE с position: fixed не все так гладко, как бы нам хотелось, но существует множество способов обойти эти ограничения.

position:relative + position:absolute

Назначив блоку #content относительное позиционирование (position: relative), мы сможем позиционировать любые дочерние элементы, относительно его границ. Давайте разместим блок #div-1a, в верхнем правом углу блока #content.

```
#content {
	position:relative;
}

#div-1a {
	position:absolute;
	top:0;
```

```
        right:0;
        width:200px;
}
```

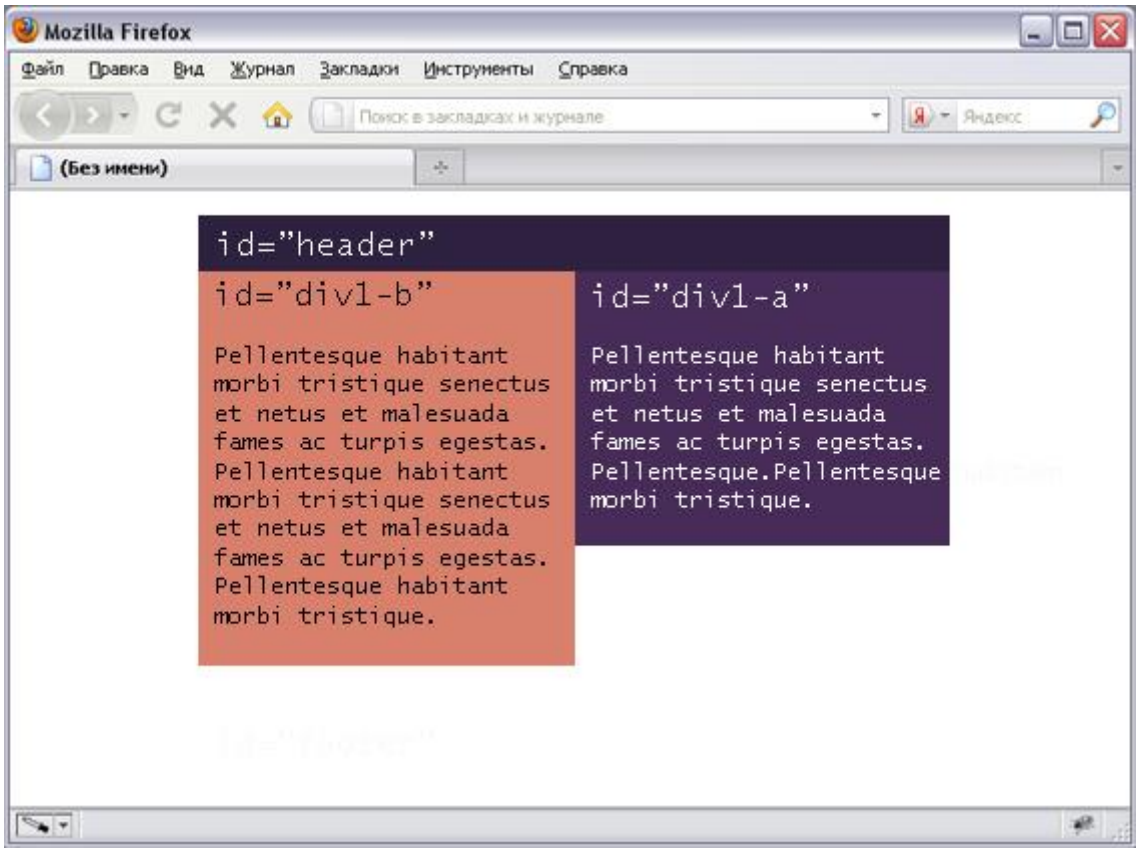


Две колонки

Вооружившись знаниями из предыдущих шагов, можно попробовать сделать две колонки, с помощью относительного и абсолютного позиционирования.

```
#content {
        position:relative;
}

#div-1a {
        position:absolute;
        top:0;
```

```
        right:0;
        width:200px;
}

#div-1b {
        position:absolute;
        top:0;
        left:0;
        width:200px;
}
```



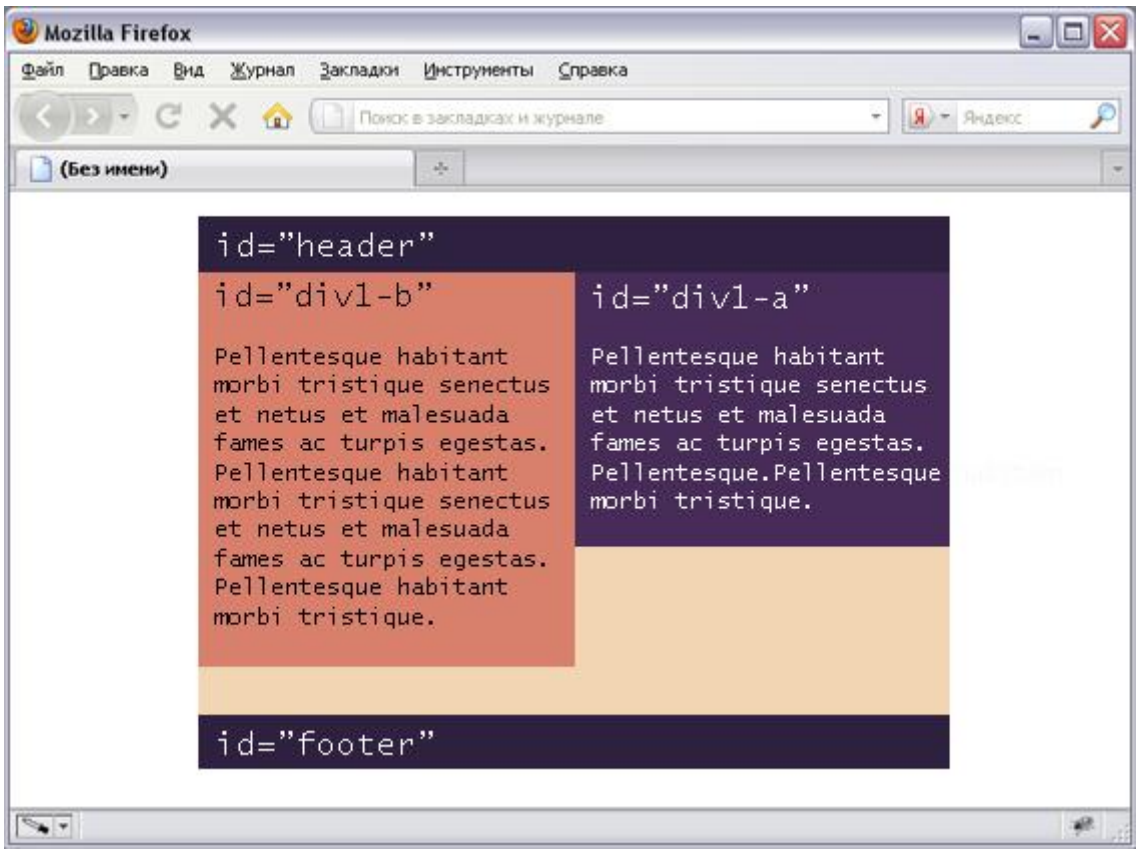
Одним из преимуществ абсолютного позиционирования, является возможность размещать элементы в произвольном порядке, независимо от того, как именно они расположены в разметке. В приведенном выше примере, блок #div-1b расположен перед блоком #div-1a.

А сейчас у вас должен был возникнуть вопрос: “А куда же делись остальные элементы из нашего примера?”. Они скрылись под абсолютно расположенными блоками. К счастью, есть возможность это исправить.

Две колонки с фиксированной высотой

Одно из решений – задать фиксированную высоту контейнеру, содержащему колонки.

```
#content {
	position:relative;
	height: 450px;
}

#div-1a {
	position:absolute;
	top:0;
	right:0;
	width:200px;
}

#div-1b {
	position:absolute;
	top:0;
	left:0;
	width:200px;
}
```



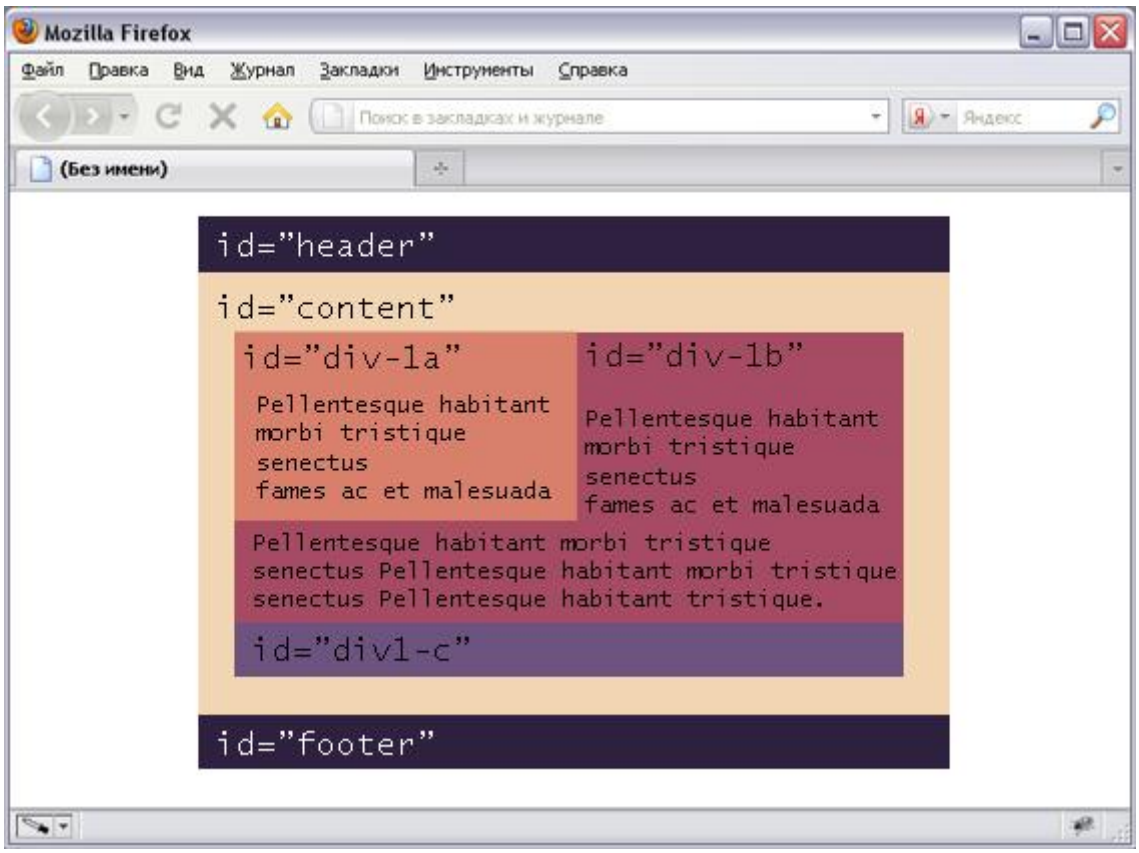
Решение не очень подходящее, поскольку мы никогда не знаем заранее, какого размера текст, будет расположен в колонках, и какой шрифт будет использован.

Float

Для колонок с переменной высотой, абсолютное позиционирование не подходит, поэтому давайте рассмотрим другой вариант.

Назначив блоку float, мы максимально возможно оттолкнем его к правому (или левому) краю, а следующий за блоком текст, будет обтекать его. Обычно такой прием используется для картинок, но мы будем использовать его для более сложной задачи, поскольку это единственный инструмент, имеющийся в нашем распоряжении.

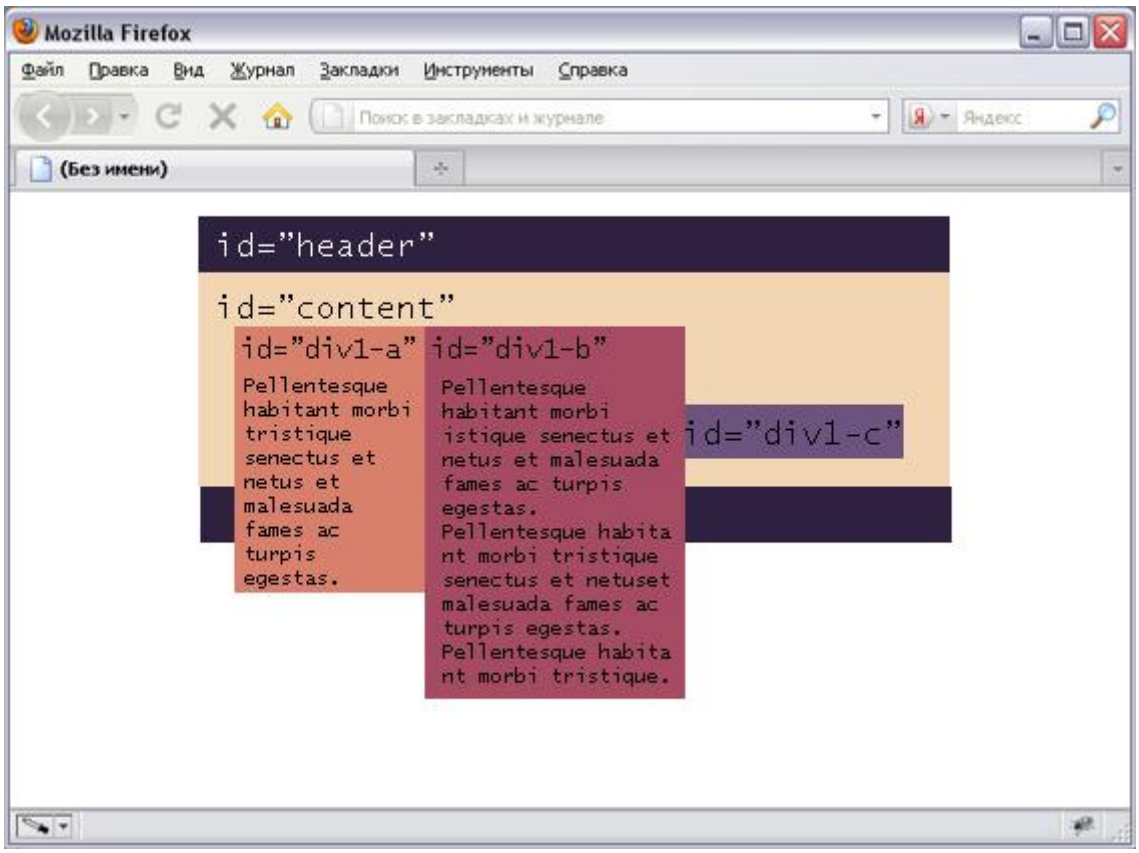
```
#div-1a {
      float:left;
      width:200px;
}
```



“Плавающие” колонки

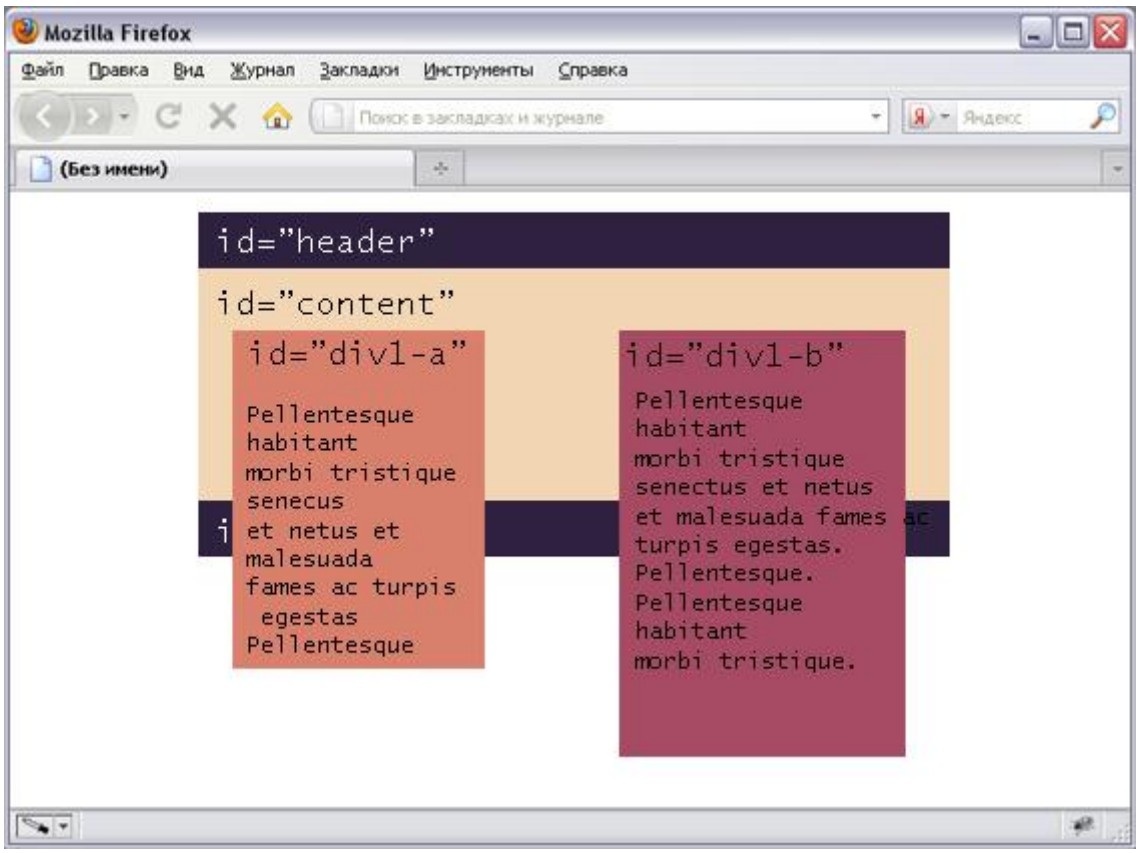
Если назначить первому блоку float: left, а затем второму float: left, каждый из блоков прижмется к левому краю, и мы получим две колонки, с переменной высотой.

```
#div-1a {
      float:left;
      width:150px;
}

#div-1b {
      float:left;
      width:150px;
}
```



Также, можно назначить колонкам противоположное значение float, в этом случае, они распределятся по краям контейнера.

```
#div-1a {
      float:right;
      width:150px;
}

#div-1b {
      float:left;
      width:150px;
}
```



Но теперь у нас появилась другая проблема – колонки выходят за пределы родительского контейнера, тем самым ломая всю верстку. Эта самая распространенная проблема начинающих верстальщиков, хотя решается она довольно просто.

Очистка float

Чистку флоатов можно делать двумя способами. Если после колонок идет еще один блок, достаточно назначить ему clear: both.

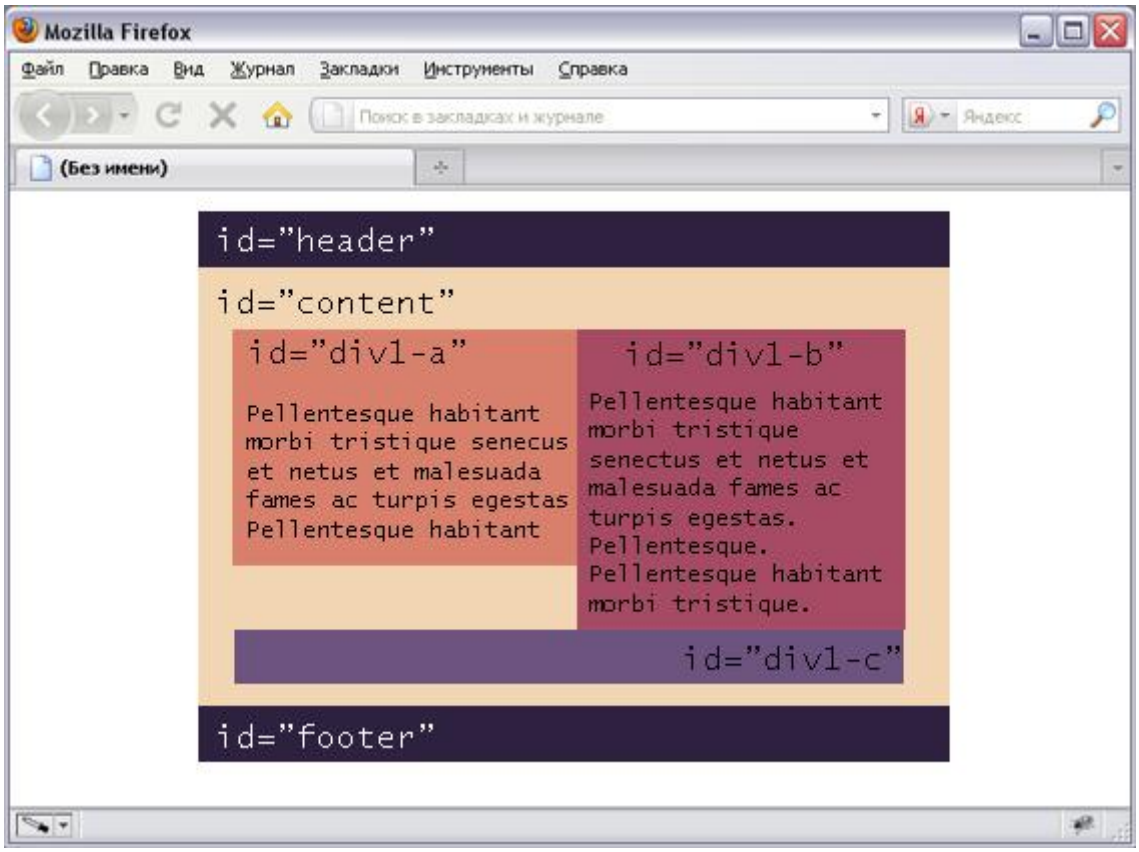
```
#div-1a {
	float:left;
	width:190px;
}
#div-1b {
	float:left;
	width:190px;
```

```
}
#div-1c {
	clear:both;
}
```

Или же назначить родительскому контейнеру свойство overflow: hidden

```
#content {
	overflow:hidden;
}
```

В любом случае, результат будет один и тот же.



Практическое задание

1. Разместите элемент с помощью абсолютного позиционирования (например, заголовок или футер).
2. Разместите элемент с помощью фиксированного позиционирования (например, футер или кнопку «наверх»).
3. Используйте комбинацию относительного и абсолютного позиционирования.
4. Реализуйте N колонок с помощью свойства float.
5. Продемонстрируйте применение свойства clear, если это возможно.