ПОЗИЦИОНРИРОВАНИЕ ЭЛЕМЕНТОВ

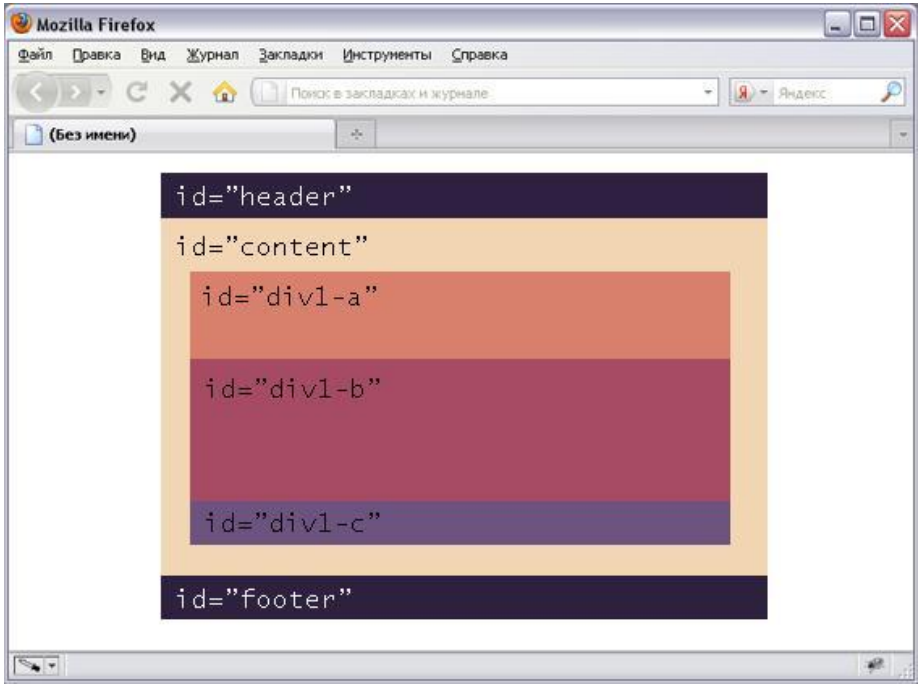
(Опять крутой подзаголовок)

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

1. 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.

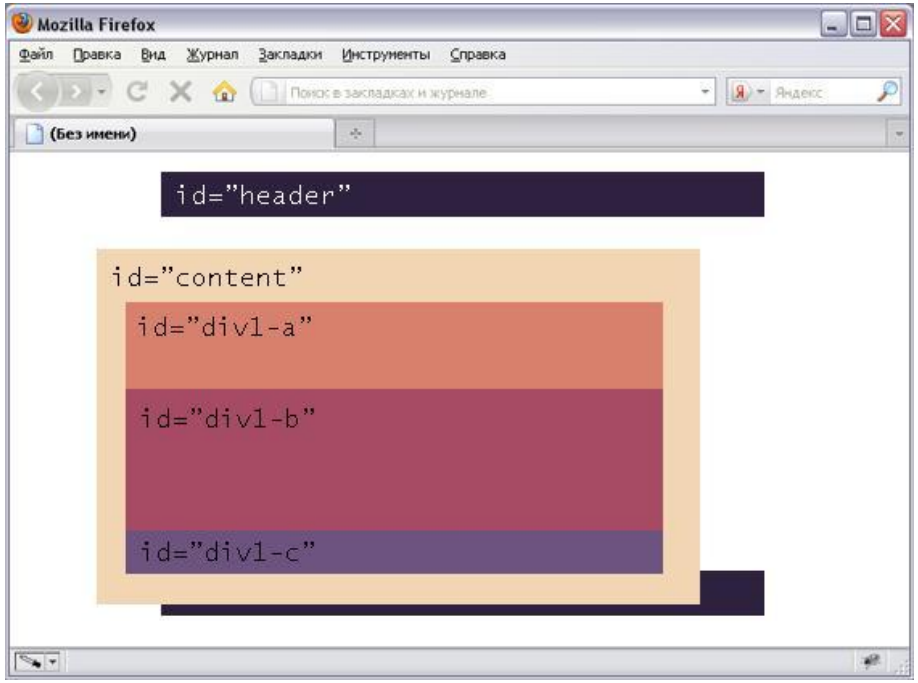
```
#content{
  position:
static;
}
```



2. 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;
}
```

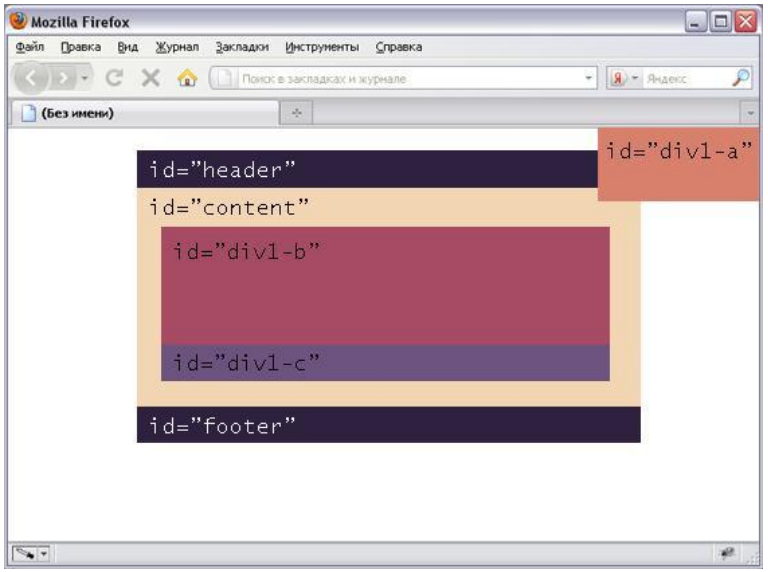


- Обратите внимание, что на том месте, где бы должен был находится наш блок #content, теперь образовалось пустое пространство. Следующий за блоком #content, блок #footer не переместился ниже, потому что, #content по-прежнему занимает свое место в документе, несмотря на то, что мы передвинули его.
- На данном этапе может показаться, что относительное позиционирование не так уж и полезно, но, не спешите с выводами, далее в статье, вы узнаете, для чего его можно использовать.

3. 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:

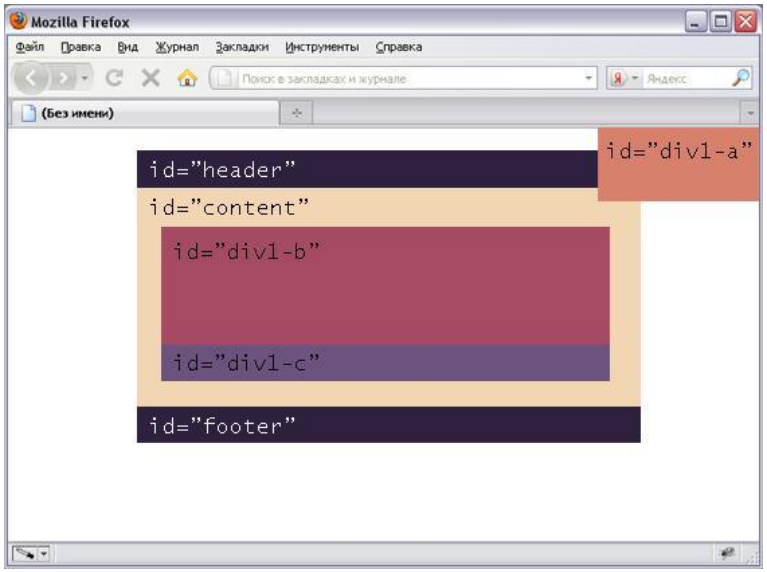
```
#div-1a {
  position:absolute;
  top:0;
  right:0;
  width:200px;
}
```



3. 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;
}
```

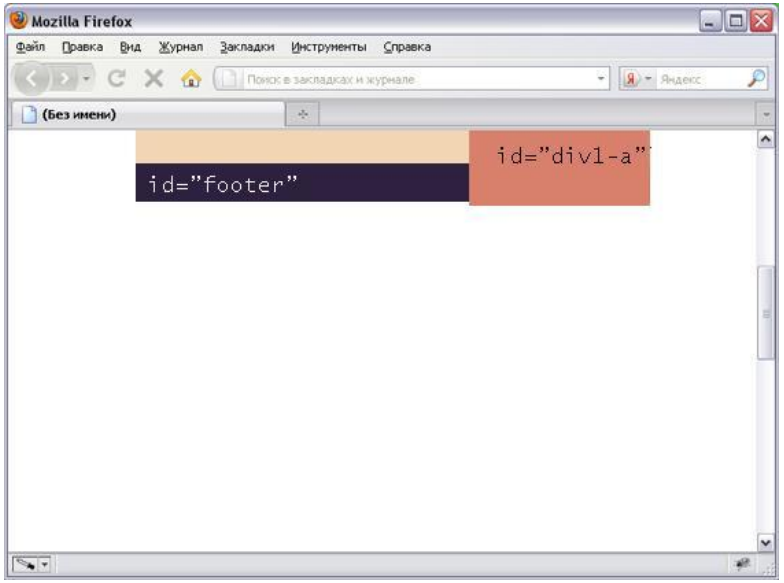


- Обратите внимание, что на этот раз, поскольку блок #div-1a был удален из документа, оставшиеся элементы на странице расположились по-другому: #div-1b, #div-1c и #footer переместились выше, на место удаленного блока. А сам блок #div-1a, расположился точно в правом, верхнему углу страницы.
- Таким образом, мы можем позиционировать любой элемент относительно страницы, однако этого не достаточно. На самом деле, нам необходимо позиционировать #div-1a относительно родительского блока #content. И на этом этапе, относительное позиционирование вновь вступает в игру.

4. 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;
}
```

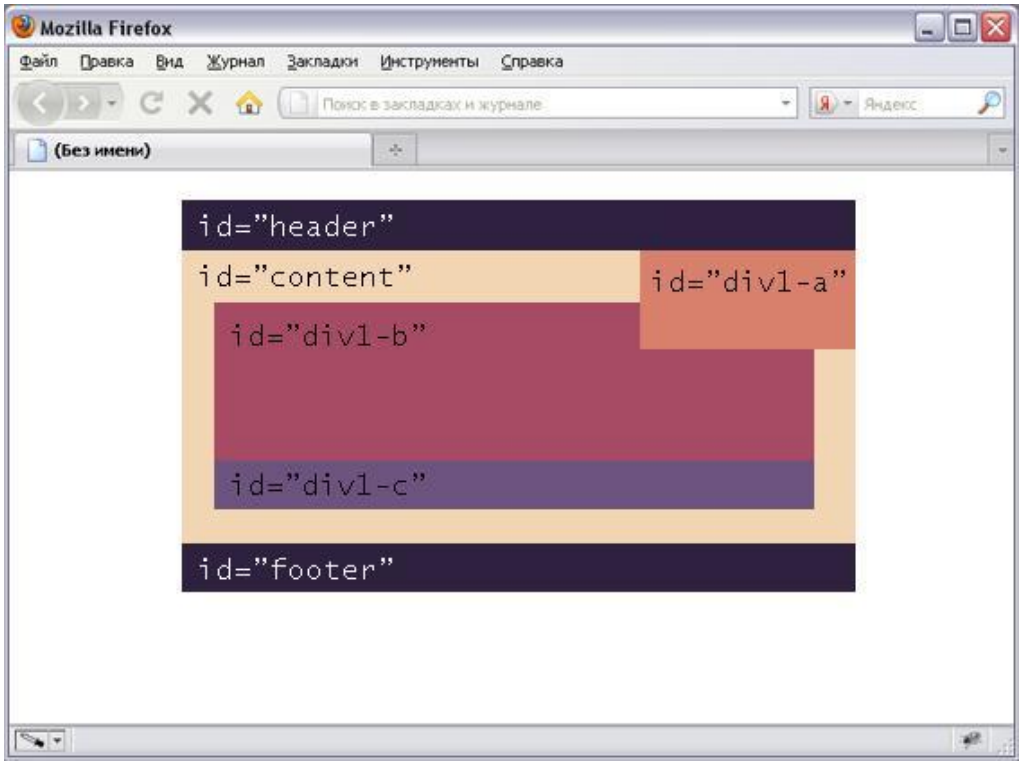


- В IE с position: fixed не все так гладко, как бы нам хотелось, но существует множество способов обойти эти ограничения.

5. 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

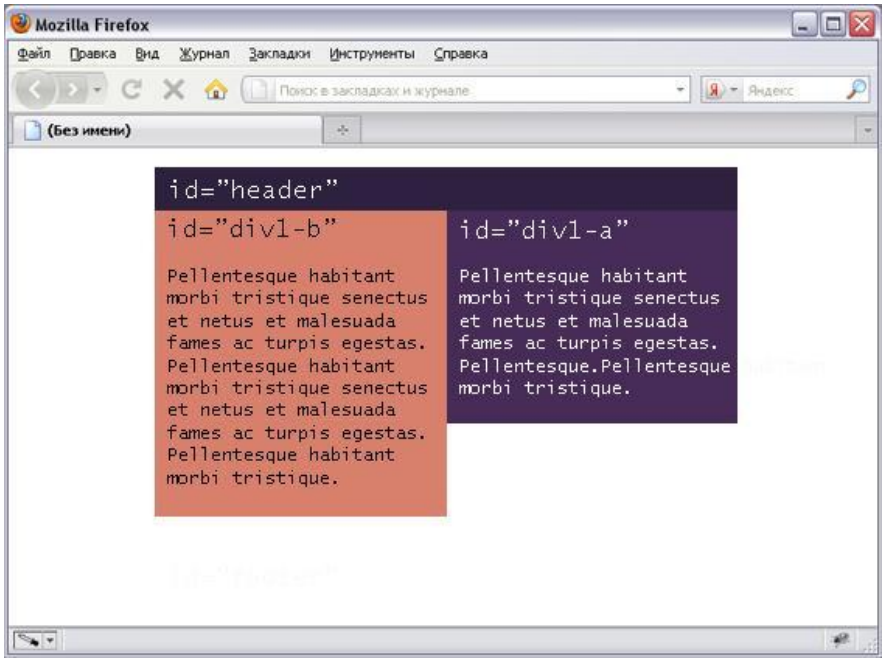
```
#content {
 position:relative;
}

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



6. 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;
}
```

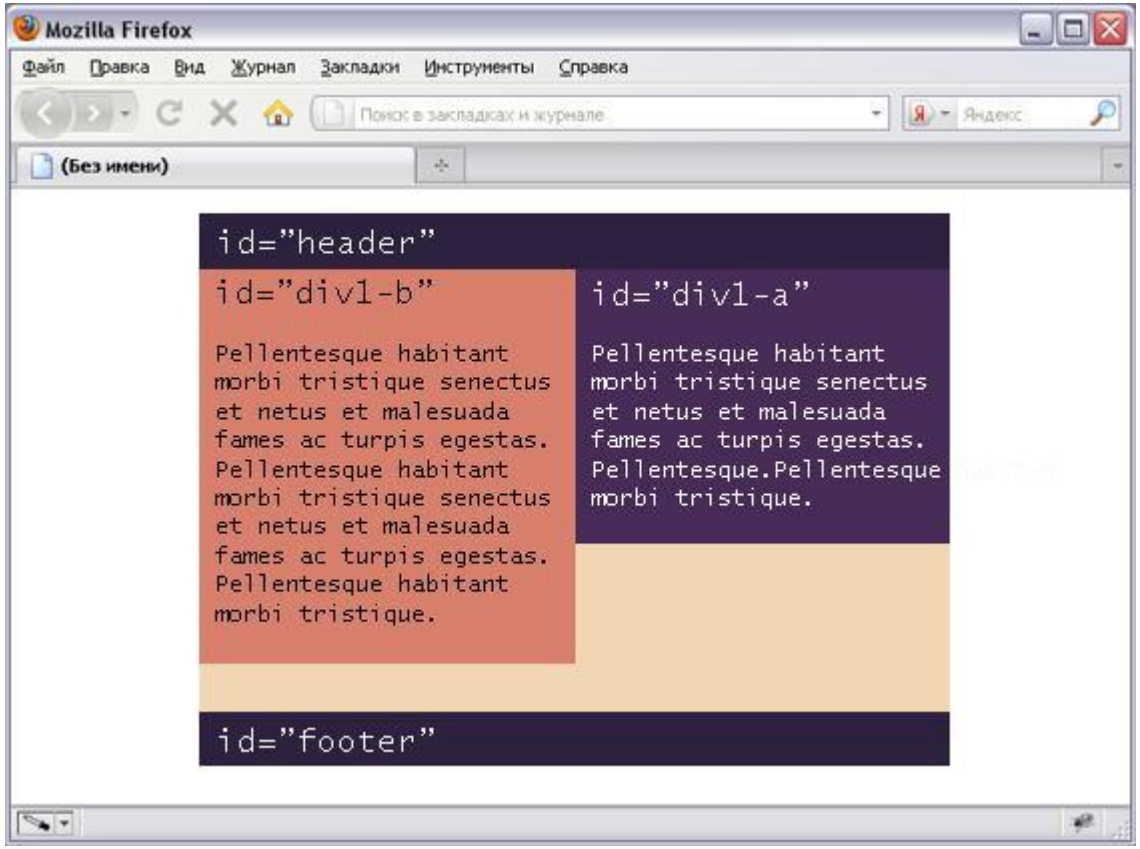


- Одним из преимуществ абсолютного позиционирования, является возможность размещать элементы в произвольном порядке, независимо от того, как именно они расположены в разметке. В приведенном выше примере, блок #div-1b расположен перед блоком #div-1a.
- А сейчас у вас должен был возникнуть вопрос: “А куда же делись остальные элементы из нашего примера?”. Они скрылись под абсолютно расположенными блоками. К счастью, есть возможность это исправить.

7. 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;
}
```

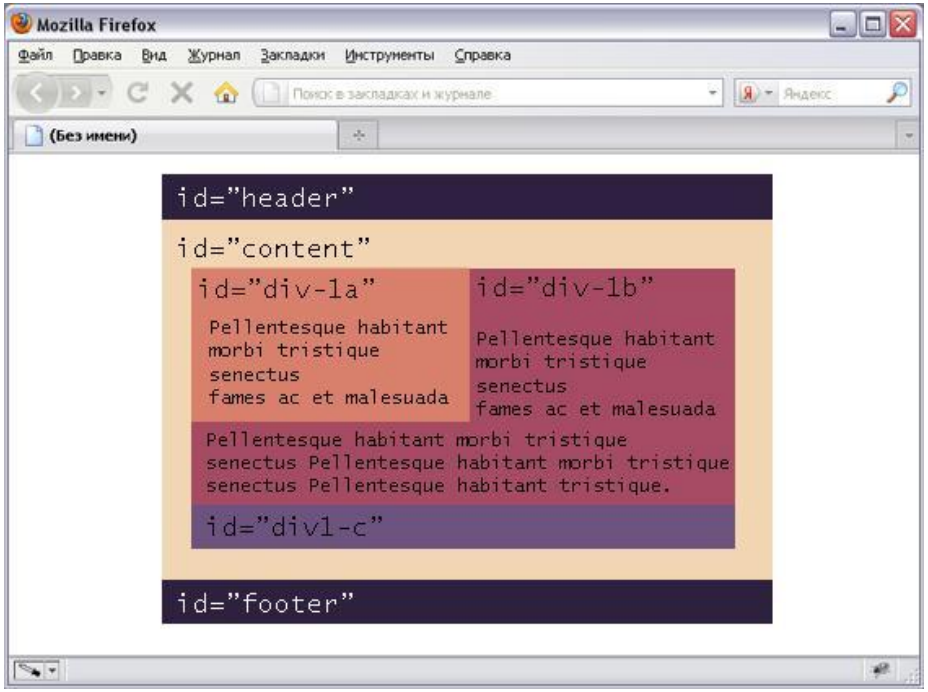


- Решение не очень подходящее, поскольку мы никогда не знаем заранее, какого размера текст, будет расположен в колонках, и какой шрифт будет использован.

8. 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.

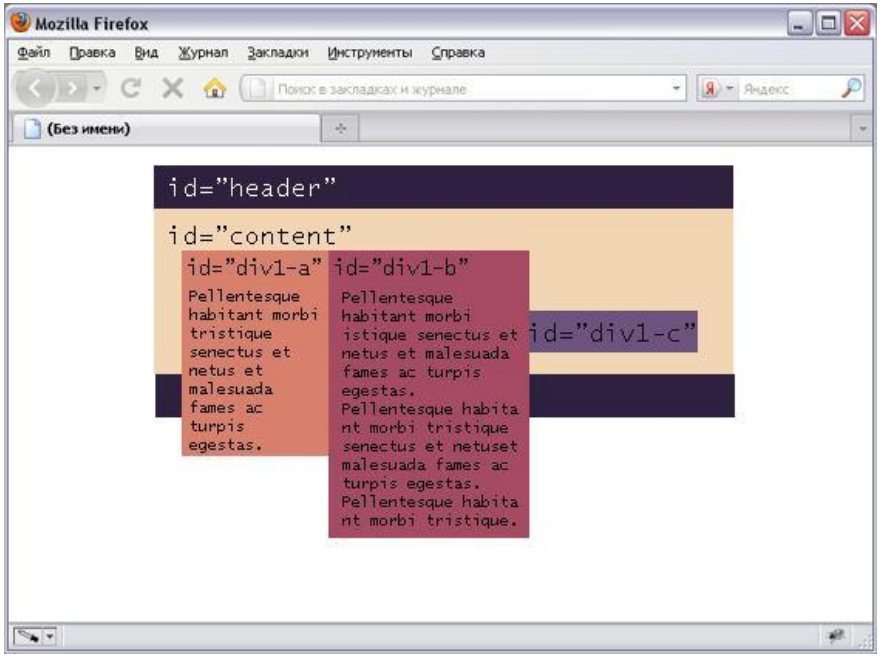
```
#div-1a {
 float:left;
 width:200px;
}
```



9. “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.

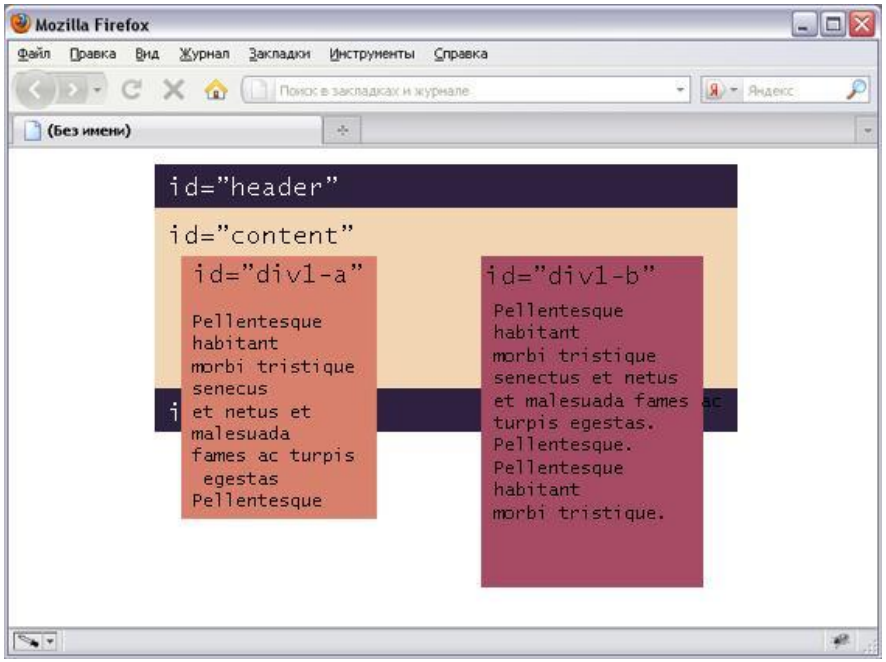
```
#div-1a {
 float:left;
 width:150px;
}

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



9. “Плавающие” колонки

- 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.

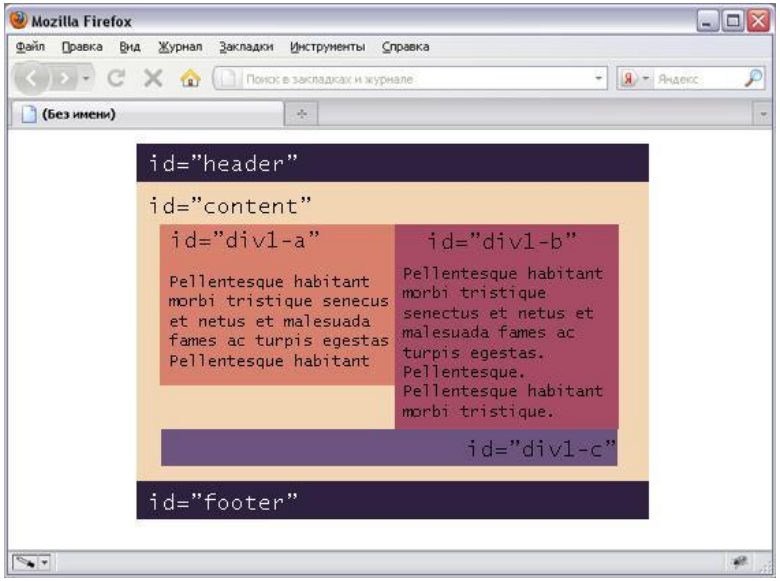
10. Очистка float

- 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;
}
```



Свойство стиля	Описание свойства	Значения	Значение по умолчанию
position	алгоритм позиционирования	static - обычный блок, заданные свойства **top** и **left** игнорируются; relative - свойства **top** и **left** смещают блок относительно его нормального положения в общем потоке данных; absolute - блок изымается из потока данных и позиционируется в соответствии со свойствами **top**, **left**, **right** или **bottom** относительно ближайшего по вложенности вмещающего его **позиционированного** контейнера; fixed - блок позиционируется как и в случае значения absolute, а потом фиксируется относительно некоторого объекта (например, области просмотра - для устройств без разбивки - в этом случае не смещается при прокрутке)	static
top	вертикальное смещение блока относительно верха контейнера	auto, *размер,* *процент* (относительно высоты контейнера)	auto
right	горизонтальное смещение блока относительно правой границы контейнера	auto, *размер,* *процент* (относительно ширины контейнера)	auto
bottom	вертикальное смещение блока относительно нижней границы контейнера	auto, *размер,* *процент* (относительно высоты контейнера)	auto
left	горизонтальное смещение блока относительно левой границы контейнера	auto, *размер,* *процент* (относительно ширины контейнера)	auto
float	задает смещение блока, вследствие чего блок будет обтекаться текстом контейнера	none, left, right	none
clear	задает управление потоком после смещенного блока	none, left, right, both	none
z-index	позиционный уровень блока; блок с большим свойством **z-index** располагается над блоком с меньшим свойством **z-index**	auto - свойство **z-index** присваивается в порядке описания элементов в исходном коде, *число*	auto
display	тип структурного блока, порождаемого HTML-элементом, которому приписано данное свойство стиля; говоря чуть более человеческим языком, это свойство указывает на поведение и , следовательно, отображение данного HTML-элемента. Например, значение display:inline заставляет элемент (насколько это возможно) встраиваться в строку данных; значение display:block выводит его с новой строки и позволяет задать элементу свойства блока; значение display:none исключает элемент из потока и вообще не показывает его и т.д.	inline, block, list-item, run-in, compact, marker, table, inline-table, table-row-group, table-header-group, table-footer-group, table-row, table-column-group, table-column, table-cell, table-caption, none	inline