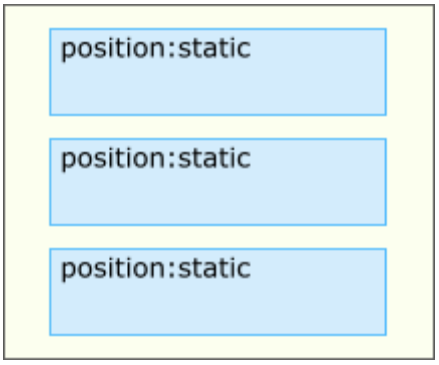
ЗАНЯТИЕ 4. ВЕРСТКА: ПОЗИЦИОНИРОВАНИЕ В CSS. СВОЙСТВА POSITION,FLOAT,DISPLAY

Позиционирование – один из наиболее значимых пунктов для блочной верстки. Задать положение блока можно задать с помощью css-свойств: position, float, display.

Для начала рассмотрим свойство position.
Существуют четыре способа позиционирования блоков:

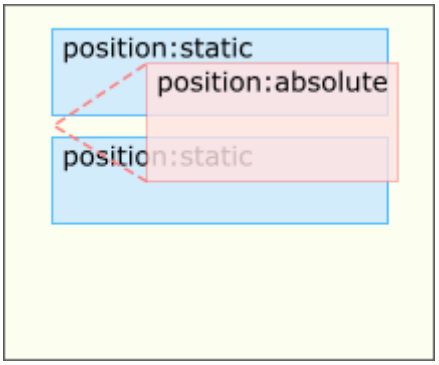
STATIC

Это способ по умолчанию, можно сказать, отсутствие какого бы то ни было специального позиционирования, а просто выкладывание боксов одного за другим сверху вниз. Этот порядок как раз и есть прямой поток.



ABSOLUTE

Блок с абсолютным позиционированием располагается по заданным координатам, а из того места, где он должен был бы быть, он удаляется, и в этом месте сразу начинают раскладываться следующие боксы. Говорят, что он "исключается из потока".

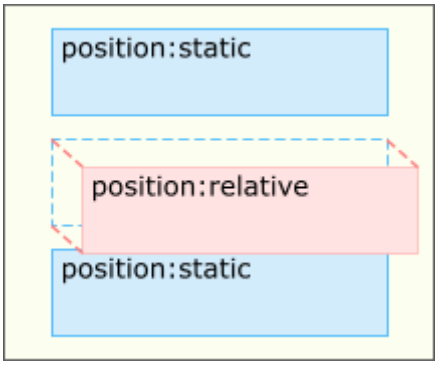


FIXED

Ведет себя так же, как absolute, но при этом он не скролится вместе с остальной страницей.

RELATIVE

Такой блок можно сдвинуть относительно того места, где он был бы в потоке, но при этом из потока он не исключается, а продолжает занимать там свое место. То есть сдвигается со своего места он только визуально, а положение всех блоков вокруг него никак не меняется.



А теперь немного детальнее и на конкретных примерах:

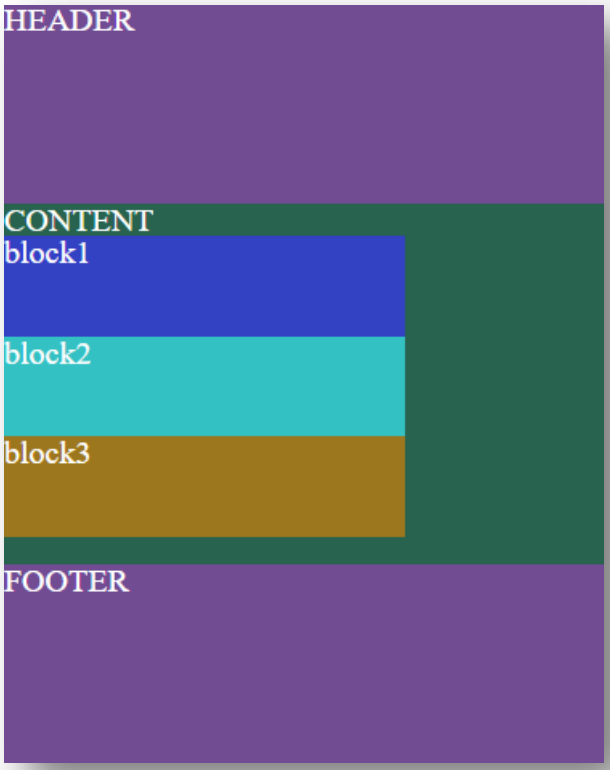
STATIC

Значение по умолчанию. Элементы отображаются последовательно один за другим в том порядке, в котором они определены в html-документе. Собственно, в этом нет ничего сложного, этот тип позиционирования устанавливается сразу неявно для всех блоков. Напишем ряд блоков в html-файле, обозначим размеры и цвета в css-документе (представлен частично):

```html
<!doctype html>
<html lang="ru">
    <html>

        <head>
            <meta charset="utf-8">

            <link href="reset.css" rel="stylesheet">
            <link href="style.css" rel="stylesheet">

            <title>Позиционируем блоки</title>
        </head>

    <body>

        <section class = "header">
            HEADER
        </section>

        <section class = "content">
            CONTENT
                <div class="block1">block1</div>
                <div class="block2">block2</div>
                <div class="block3">block3</div>
        </section>

        <section class = "footer">
            FOOTER
        </section>

    </body>


</html>
```

```css
.header
{
    width: 300px;
    height: 100px;
    background: #724c93;
    color: white;
}

.content
{
    width: 300px;
    height: 180px;
    background: #28634f;
    color: white;
}

.block1
{
    width: 200px;
    height: 50px;
    background: #3341c3;
    color: white;
}
```

В конечном итоге получим такое отображение в браузере. Все блоки расположены в таком же порядке, как и в html-документе:



ABSOLUTE

- Можно задавать только для блочных (display: block) и строчно-блочных (display: inline-block) элементов (поговорим чуть позже о типах элементов).
- Позиция элемента смещается относительно родительского элемента вверх, вправо, вниз или влево (зависит от того, какое задано значение: top / right / bottom / left), при этом нарушается порядок отображения элементов. Элементы, следующие за ним, могут попасть под него.
- На положение также влияет значение свойства position родительского элемента.

  Так, если у родителя значение position установлено как static или родителя нет, то отсчет положения ведется от верхнего края окна браузера. Если у родителя значение position задано как relative / fixed / absolute, то отсчет положения элемента ведется от края родительского элемента.

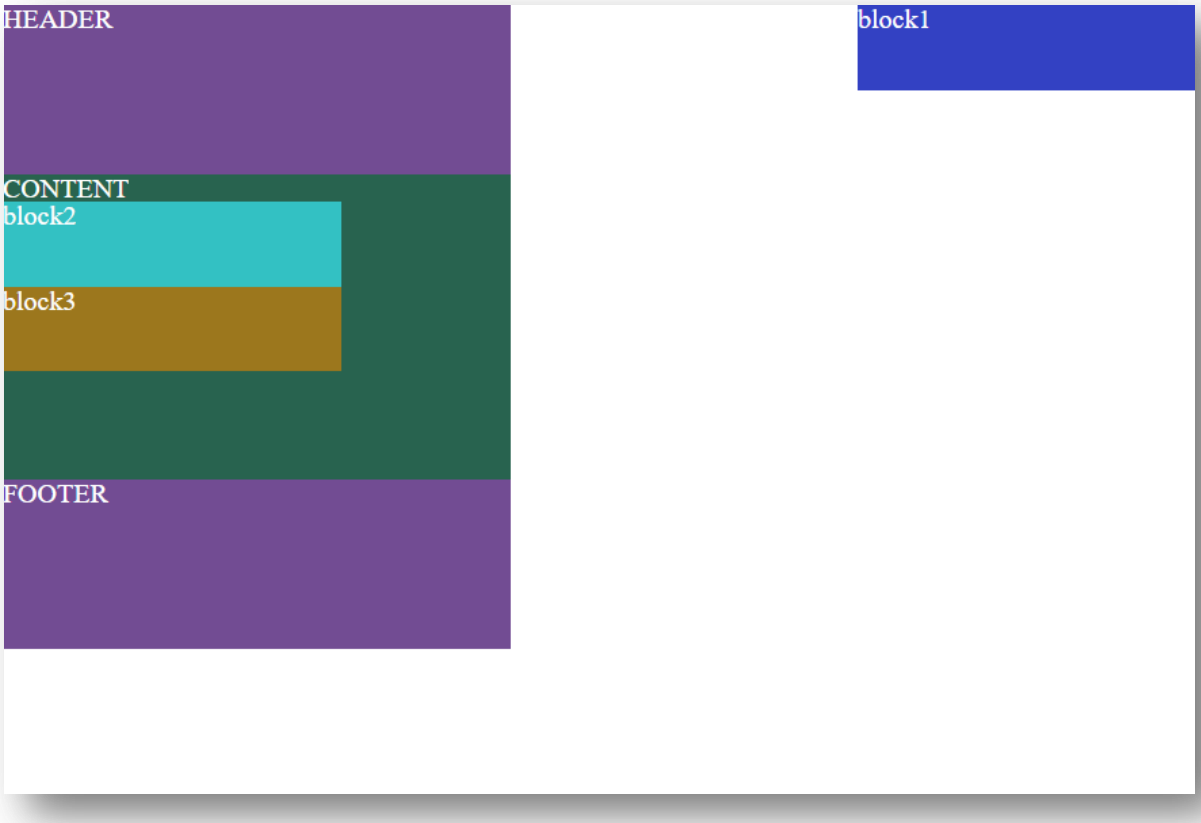
- Можно вкладывать один блок с position: absolute в другой блок с position: absolute.

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

Давайте, для примера, переместим блок block1 в верхний, правый угол страницы, задав ему определенные свойства:

```
.block1
{
    position: absolute;
    top:0;
    right: 0;
    width: 200px;
    height: 50px;
    background: #3341c3;
    color: white;
}
```

В итоге получится следующее:



Обратите внимание, что на этот раз, поскольку блок block1 был удален из документа, оставшиеся элементы на странице расположились по-другому: block2 и block3 переместились выше, на место удаленного блока. А сам блок block1, расположился точно в правом, верхнему углу страницы.

FIXED

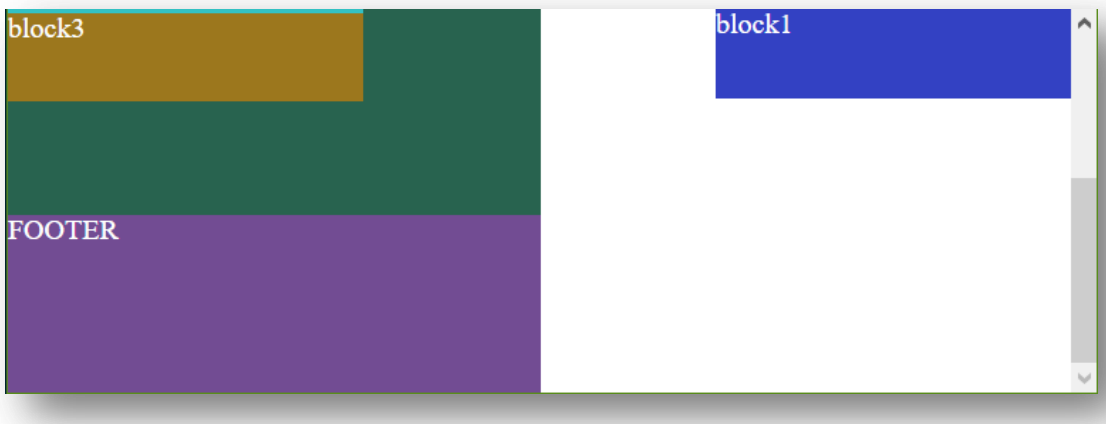
Фиксированное позиционирование (position: fixed), является подразделом абсолютного позиционирования. Единственное его отличие в том, что он всегда находится в видимой области экрана, и не двигается во время прокрутки страницы. В этом отношении, он немного похож на фиксированное фоновое изображение.

Элемент можно сместить вверх, вправо, вниз или влево (в зависимости от заданного значения: top / right / bottom / left).

Попробуем проделать это с нашим block1:

```
.block1
{
    position: fixed;
    top:0;
    right: 0;
    width: 200px;
    height: 50px;
    background: #3341c3;
    color: white;
}
```

Даже при скроллинге позиция остается неизменной:

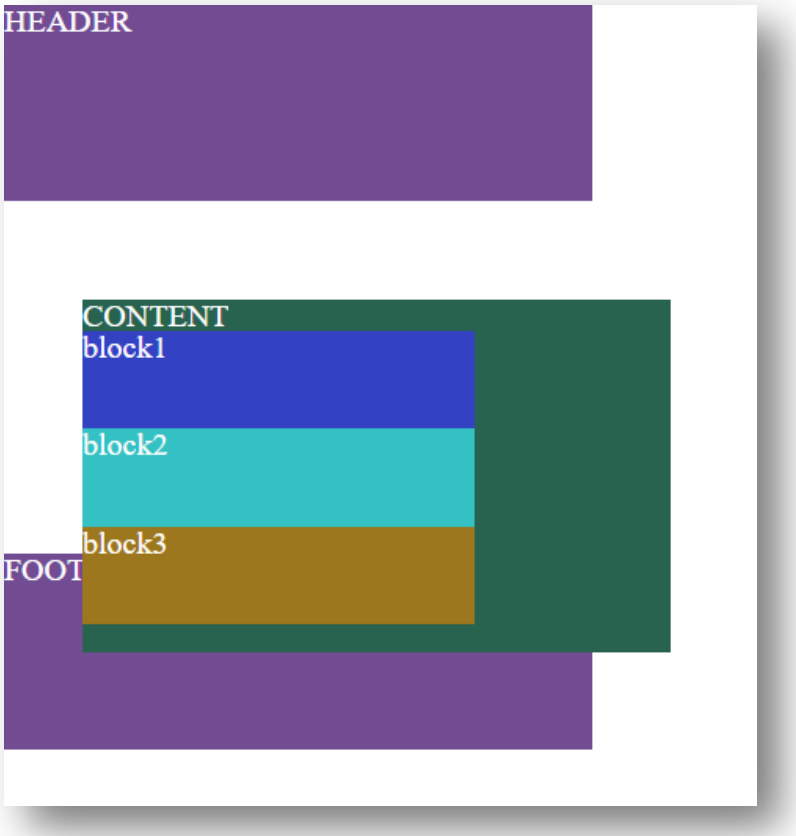


RELATIVE

Позволяет смещать элемент относительно его текущего положения в структуре html-документа вверх, вправо, вниз или влево (зависит от того, какое задано значение: top / right / bottom / left). Остальные элементы размещены на странице относительно его первоначального положения, без смещения.

Давайте для примера сместим content на 50 пикселей вниз, и на 40 пикселей влево:

```
.content
{
    position: relative;
    top: 50px;
    left: 40px;
    width: 300px;
    height: 180px;
    background: #28634f;
    color: white;
}
```



Разумеется, будь все так просто – мы бы не посвящали позиционированию целое занятие. Сложность представляет собой сочетание свойств, рассмотренных выше – необходимо знать как расположатся блоки с применением различных типов позиционирования.

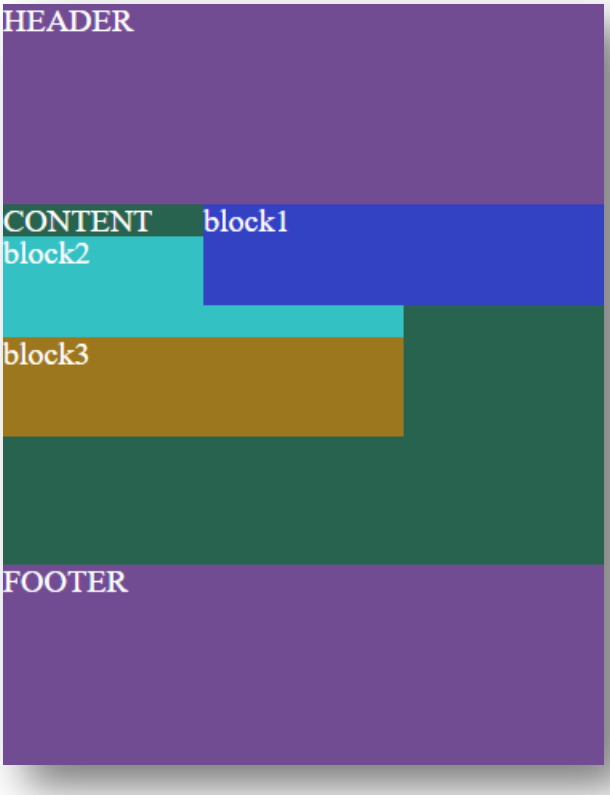
Итак, разберемся, как же они ведут себя в таких случаях на нескольких примерах.

POSITION: RELATIVE + POSITION: ABSOLUTE

Довольно часто нам придется кстати такое сочетание. Оно используется в том случае, когда нужно позиционировать элемент в пределах другого.

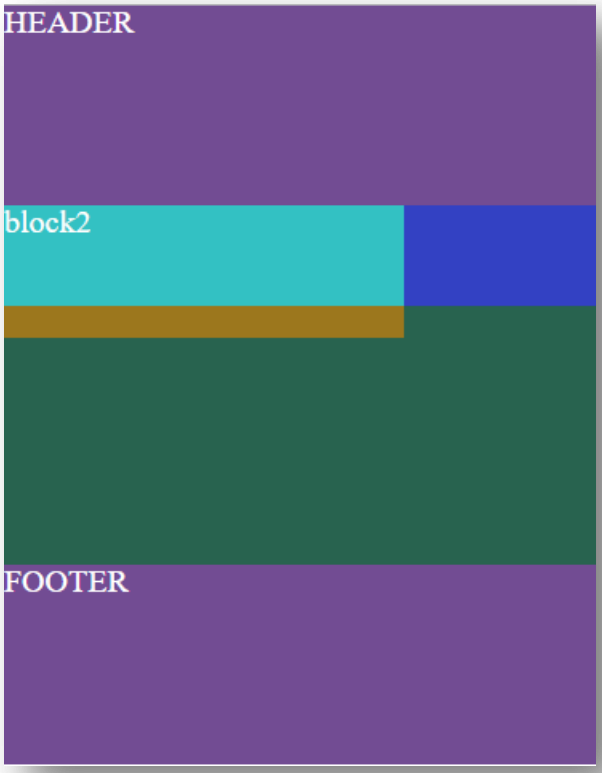
Назначив блоку content относительное позиционирование (position: relative), мы сможем позиционировать любые дочерние элементы, относительно его границ. Давайте разместим block1, в верхнем левом углу блока content:

```
.content
{
    position: relative;
    width: 300px;
    height: 180px;
    background: #28634f;
    color: white;
}

.block1
{
    position: absolute;
    top:0;
    right: 0;
    width: 200px;
    height: 50px;
    background: #3341c3;
    color: white;
}
```



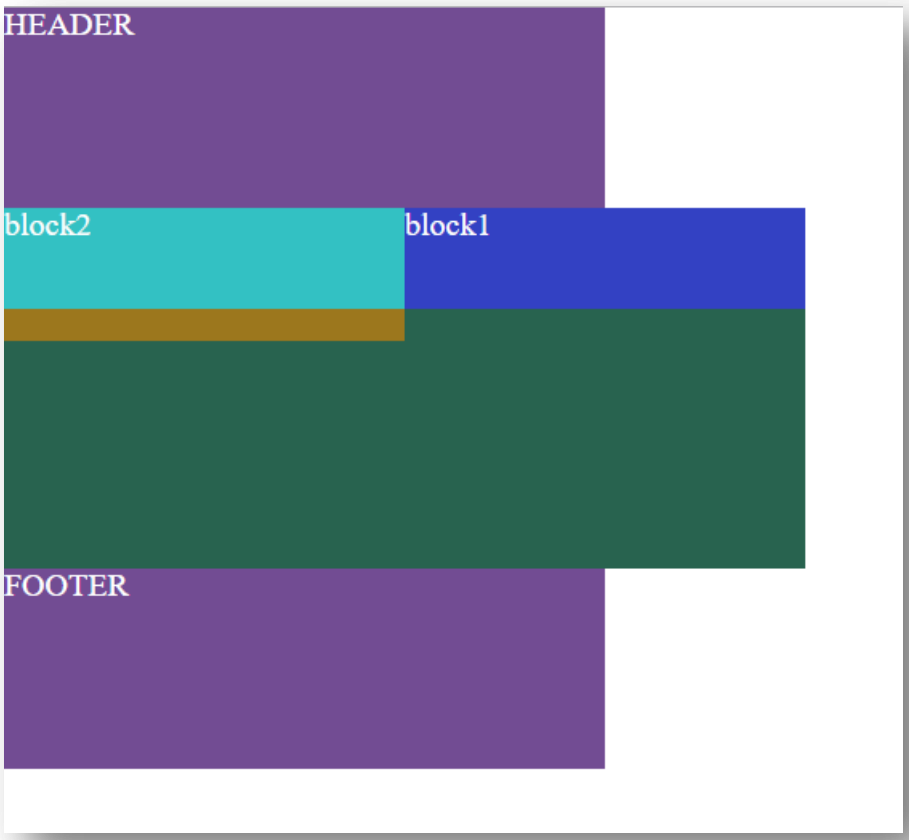
Далее можно попробовать сделать две колонки, с помощью относительного и абсолютного позиционирования:

```css
.content
{
    position: relative;
    width: 300px;
    height: 180px;
    background: #28634f;
    color: white;
}

.block1
{
    position: absolute;
    top:0;
    right: 0;
    width: 200px;
    height: 50px;
    background: #3341c3;
    color: white;
}

.block2
{
    position: absolute;
    top:0;
    left:0;
    width: 200px;
    height: 50px;
    background: #33c1c3;
    color: white;
}
```



Казалось бы, все верно и блоки должны отобразиться в две колонки, но почему не отображается название второго блока? Присмотритесь внимательнее к размерам родительского блока content и к размерам block1 и block2. Проблема в том, что ширина родительского блока не вмещает дочерних (у content width:300px, а сумма значений block1,2 дает 400px). Возможное решение – увеличить ширину родительского блока:

```css
.content
{
    position: relative;
    width: 400px;
    height: 180px;
    background: #28634f;
    color: white;
}
```



Также распространенная проблема при таком типе позиционирования – перекрытие одного блока другим. Наиболее вероятно вы просто забыли указать высоту блокам.

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

FLOAT (ПЛАВАЮЩИЕ БЛОКИ)

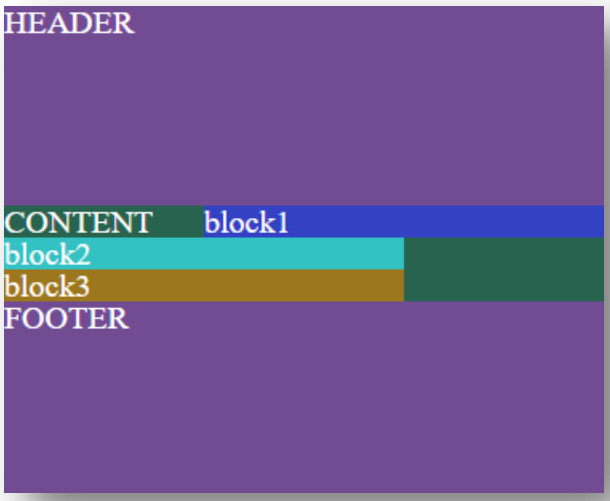
Назначив блоку float, мы максимально возможно оттолкнем его к правому (или левому) краю, а следующий за блоком текст, будет обтекать его. Обычно такой прием используется для картинок, но мы будем использовать его и для более сложных задач.

- Плавающие блоки определяются свойством float. Возможны три варианта: left - блок прижимается к левому краю, остальные элементы обтекают его с правой стороны.

- right - блок прижимается к правому краю, остальные элементы обтекают его с левой стороны.
- none - блок не перемещается и позиционируется согласно свойству position.

Установим свойство float:right для block1:

```
.content
{
    width: 300px;
    background: #28634f;
    color: white;
}

.block1
{
    float: right;
    width: 200px;
    background: #3341c3;
    color: white;
}
```

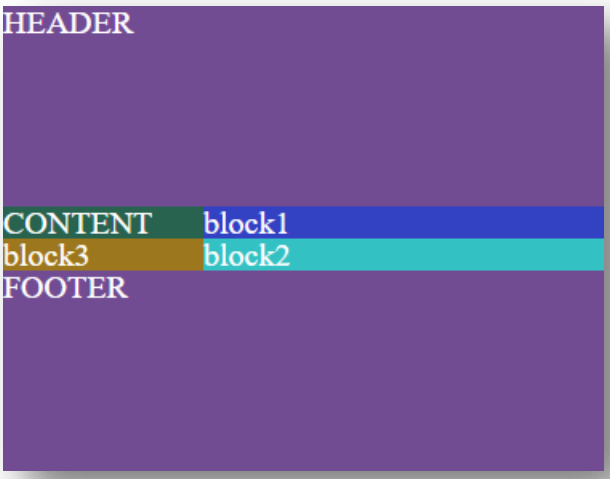


Как видим, блок прижался к правому краю, другие же обтекают его. Несмотря на то, что блок content является родительским для block1 – он теперь располагается слева от него.

Давайте посмотрим, что будет, если назначить и для block2 свойство float:right:

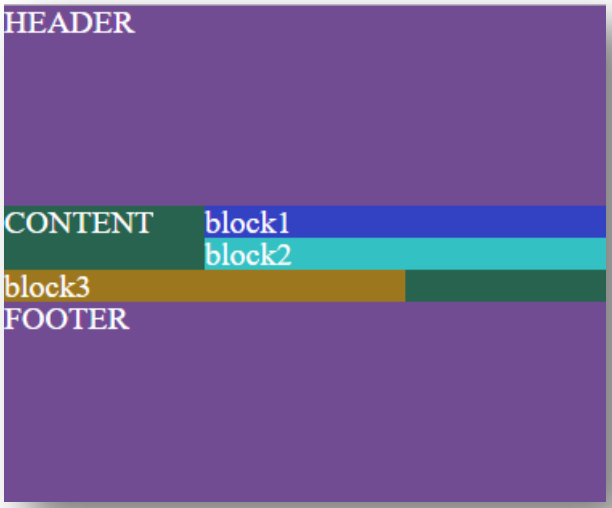
```
.block1
{
    float: right;
    width: 200px;
    background: #3341c3;
    color: white;
}

.block2
{
    float: right;
    width: 200px;
    background: #33c1c3;
    color: white;
}
```

Block2 также прижался к правому краю, однако возникла проблема: block3 частично перекрывается block1 и block2.



Решается проблема довольно просто – так называемой очисткой float. Для этого необходимо назначить свойство clear:both; для того блока, который оказался перекрыт. В нашем случае – для block 3.

```css
.block1
{
    float: right;
    width: 200px;
    background: #3341c3;
    color: white;
}

.block2
{
    float: right;
    width: 200px;
    background: #33c1c3;
    color: white;
}

.block3
{
    width: 200px;
    background: #9c771d;
    color: white;
    clear: both;
}
```



Теперь block3 снова на своем месте, а block1 и block2 прижаты к правому краю.

DISPLAY

Далее поговорим о еще одном важном свойстве: display. Свойство display имеет много разных значений. Обычно, используются только три из них: none, inline и block, потому что когда-то браузеры другие не поддерживали. Поэтому рассмотрим эти распространенные варианты.

- Каждый элемент имеет значение отображения по умолчанию в зависимости от того, к какому типу относится данный элемент.
- Для большинства элементов, значения отображения по умолчанию, как правило, будут block или inline.
- В оригинале, блочный элемент часто еще называют элементом блочного уровня(block-level element).У строчного же элемента нет альтернативного названия.

ЗНАЧЕНИЕ NONE

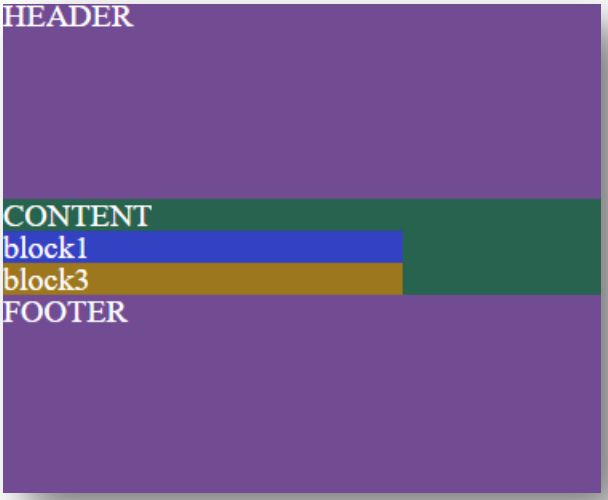
Некоторые специализированные элементы, такие как script используют это значение по умолчанию. Оно широко используется с JavaScript, чтобы скрывать и показывать элементы вместо того, чтобы удалять и воссоздавать их снова.

Оно отличается от visibility. При задании свойству display значения none страница будет отображаться словно элемент не существует. visibility: hidden; просто скроет элемент, но элемент по прежнему будет продолжать занимать место, как если бы он был полностью виден.

Установим для block2 значение display:none :

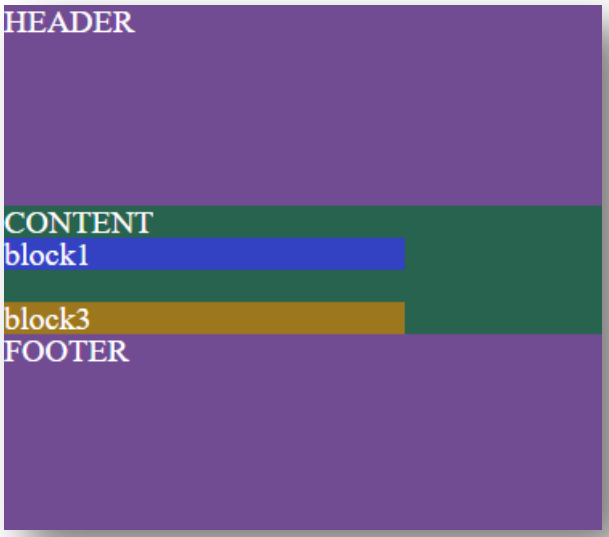
```
.block2
{
    display: none;
    width: 200px;
    background: #33c1c3;
    color: white;
}
```

Block2 исчез, остальные блоки расположились так, будто его и вовсе нет.



Посмотрим что же будет происходить при установлении visibility:hidden :

```
.block2
{
    visibility: hidden;
    width: 200px;
    background: #33c1c3;
    color: white;
}
```



ЗНАЧЕНИЕ BLOCK

- Блочные элементы располагаются один над другим, вертикально (если нет особых свойств позиционирования, например float).
- Блок стремится расшириться на всю доступную ширину. Можно указать ширину и высоту явно.

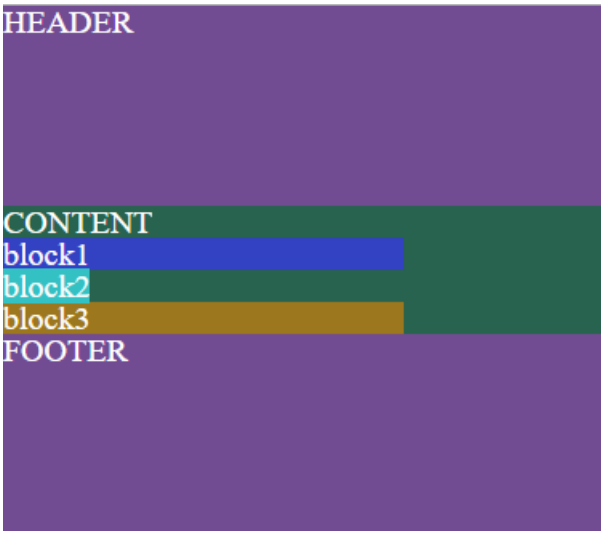
- Это значение display многие элементы имеют по умолчанию: , заголовок <h1>.
- Другие распространенные блочные элементы это p и form, а также новые блочные элементы из HTML5, такие как header, footer, section, и прочие.
- Блоки прилегают друг к другу вплотную, если у них нет margin.

ЗНАЧЕНИЕ INLINE

- Элементы располагаются на той же строке, последовательно.
- Ширина и высота элемента определяются по содержимому. Поменять их нельзя.
- Инлайновые элементы по умолчанию: , <a>.
- Содержимое инлайн-элемента может переноситься на другую строку.

Давайте назначим свойство display:inline для block2. Как видно – ширина и высота определились по содержимому, даже если поставить ширину и высоту у блока – ситуация не поменяется.

```
.block2
{
    display: inline;
    width: 200px;
    background: #33c1c3;
    color: white;
}
```



Посмотрим что будет, если установить данное свойство для всех трех блоков:

```css
.block1
{
    display: inline;
    width: 200px;
    background: #3341c3;
    color: white;
}

.block2
{
    display: inline;
    width: 200px;
    background: #33c1c3;
    color: white;
}

.block3
{
    display: inline;
    width: 200px;
    background: #9c771d;
    color: white;
}
```