Что такое CSS

CSS (Cascading Style Sheets) - каскадные таблицы стилей.

Стиль - набор параметров, задающий внешнее представление объекта. Например, пусть мы хотим, чтобы все заголовки первого уровня (теги <h1>) на одной странице имели красный цвет, размер - 24 и были написаны курсивом, а на другой странице были бы синего цвета, размера - 12. Наш заголовок - это объект, а цвет, размер и начертание - это параметры. Просто параметры нашего объекта для разных страниц разные, т.е. они отличаются стилем.

Каждый элемент на странице может иметь свой стиль (параграфы, заголовки, линии, текст...). Набор стилей всех элементов называют таблицей стилей. Если для одного элемента задано несколько стилей (как в примере с заголовками), то применяется каскадирование, которое определяет приоритет того или иного стиля.

CSS, как и любой язык, имеет свой синтаксис. В нем нет ни элементов, ни параметров, ни тегов. В нем есть правила. Правило состоит из селектора и блока объявления стилей, заключенного в фигурные скобки:

Сам блок объявления стилей состоит из свойств и их значений, разделенных точкой с запятой:

Селекторами называются имена стилей, в которых указаны параметры форматирования.

 пример селектора

p.title { font-family: Arial, Helvetica, sans-serif; font-size: 18px; color: maroon; font-weight: bold }

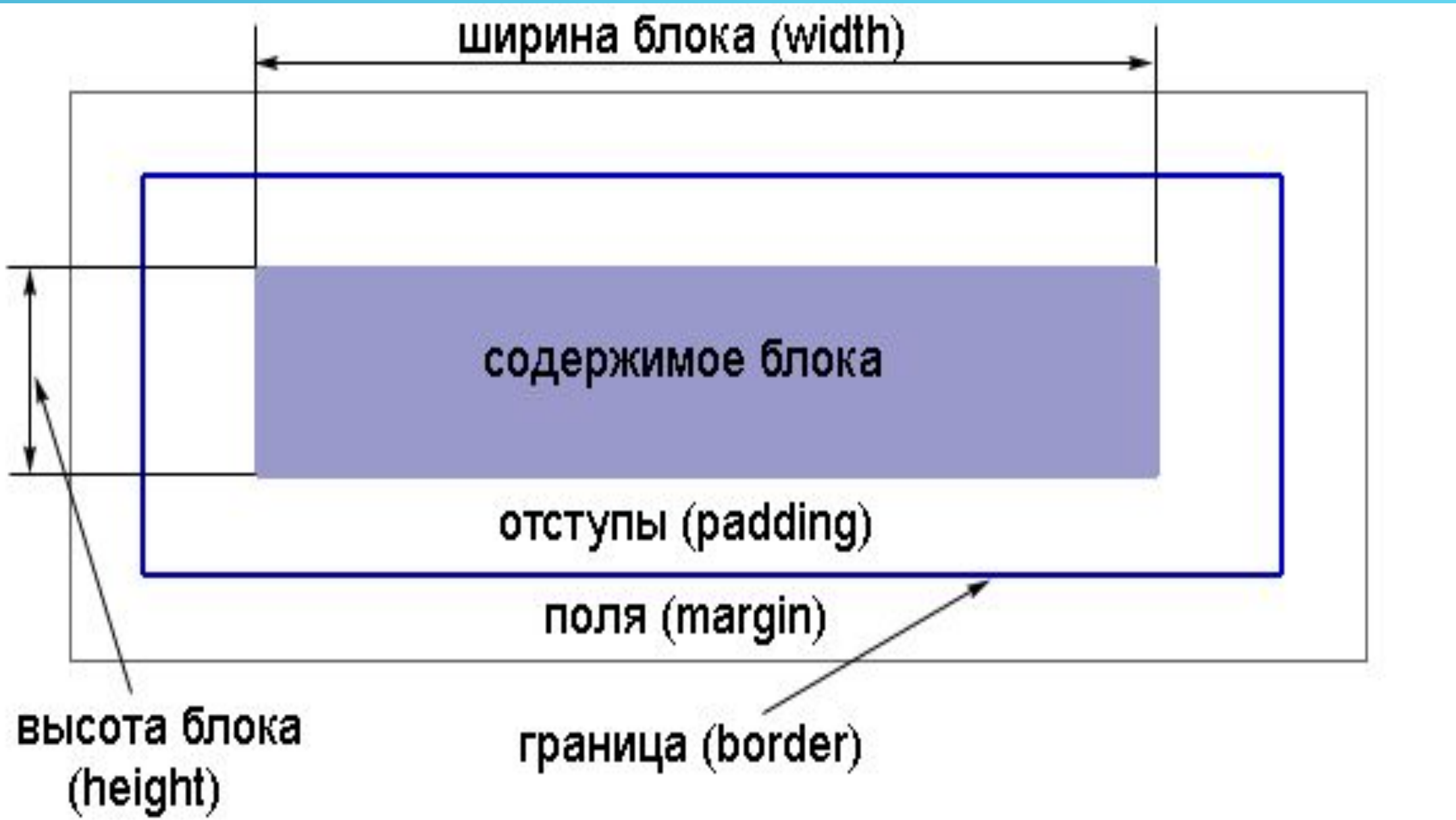
 пример селектора

p#title { font-family: Arial, Helvetica, sans-serif; font-size: 18px; color: maroon; font-weight: bold }

В примере используется селектор с именем (id) title для которого задано определение стиля. Определение состоит из свойства и его значения разделенных двоеточием. Определения разделяются точкой с запятой.

В HTML все элементы можно разделить на два типа: блочные и строчные.
Блочные элементы визуально создают самостоятельную структурную единицу - блок. К ним можно отнести, например, элементы H1-H6, P, DIV. Такие элементы отделяются от других абзацными отступами.
Строчные элементы выводятся линейной строкой. К ним можно отнести, например, элементы I, B, U, S и другие.

В CSS модель документа представляется блоком. Каждый элемент в дереве элементов документа представляет собой самостоятельный блок. Причем, есть блоки, структурно отделенные от остальных, а есть строчные блоки, которые могут находиться внутри структурных блоков.
Блок имеет прямоугольную форму:


ширина блока (width)
содержимое блока
отступы (padding)
поля (margin)
высота блока
(height)
граница (border)

У блока есть содержимое, например, для элемента p - это текст. Вокруг содержимого есть отступы (padding), они служат для того, чтобы текст не примыкал вплотную к границе блока. Фон отступов такой же, как и у содержимого.
Затем идет граница блока (border), которая может быть как видимой, так и невидимой.
Также блок имеет поля (margin), которые задают дополнительное свободное пространство вокруг блока. Фон полей прозрачный, т.е. сквозь него просвечивает фон родительского элемента.
Размер блока, т.е. его ширина (width) и высота (height), определяются содержимым. И это надо запомнить: поля и отступы не учитываются в размере блока.

Элементы могут быть блочными и строчными По умолчанию для каждого элемента его вид определен, так элементы div и p являются блочными, а и <a> - строчными. Но иногда это необходимо изменить, для этого используется свойство display. Это свойство множество значений.

Но рассмотрим три самых распространенных:

display:block, display:inline, display:none

display:block

Это значение делает элемент блочным. Предположим, мы решили сделать вертикальное меню. Для этого на html-странице мы напишем следующий код:

```
<!doctype html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" href="css/all.css" type="text/css"/>
<script src="http://code.jquery.com/jquery-1.8.3.js"></script>
<script type="text/javascript" src="js/main.js"></script>
<title>Title</title>
</head>
```

```
<body>
<div id="menu">
<a ref="index.html">Главная</a>
<a href="about_us.html">О нас</a>
<a href="contact.html">Контакты</a>
</div>
<a href="№">Кнопка</a>
 </body>
```

Зададим на странице style.css стиль для наших ссылок, причем только для ссылок, которые находятся в div-е с id="menu" (тогда другие ссылки на странице, если они будут, останутся без изменения или им можно будет задать другой стиль):

```
 #menu{
 width:200px;
 background:yellow;
}
#menu a{
 color:#2b3845;
 text-decoration:none;
}
#menu a:hover{
 color:#92A9BF;
 background:blue;
}
```

Элемент a является строчным, поэтому наши ссылки расположились в одну строку и их размер зависит от текста. Но мы хотели сделать вертикальное меню, для этого мы и добавим нашим ссылкам свойство display:block:

```
#menu a{ display:block;}
```

Обратите внимание, так как теперь наши ссылки стали блочными элементами, то и размер у них стал одинаковый, равный ширине родительского div-а.

display:inline

Это значение делает элемент строчным. Предположим, мы хотим разместить на странице параграф, который должен начинаться с заголовка. На html-странице мы напишем следующий код:

```
<h5>Заголовок.</h5> <p>Текст параграфа</p>
```

Добавим на страницу style.css стили для наших элементов:

```
h5, p{ display:inline;}
```

display:none

Это значение убирает элемент со страницы. Очень часто используется для формирования раскрывающихся меню сайтов, например, с помощью языка javascript. Вы, наверно, встречали такие меню, где при щелчке по пункту меню раскрывается список подпунктов. Вот в таких меню и используется значение display:none.

Надо сказать, что в CSS есть еще одно свойство на первый взгляд похожее на display:none. Это свойство, отвечающее за видимость блока - visibility. Оно может принимать два значения: visible (отображать) и hidden (сделать невидимым).

Различия здесь следующее: display:none скрывает элемент, как будто его и не было, а visibility:hidden делает элемент прозрачным.

Понятнее будет на примере. У нас есть пять параграфов:

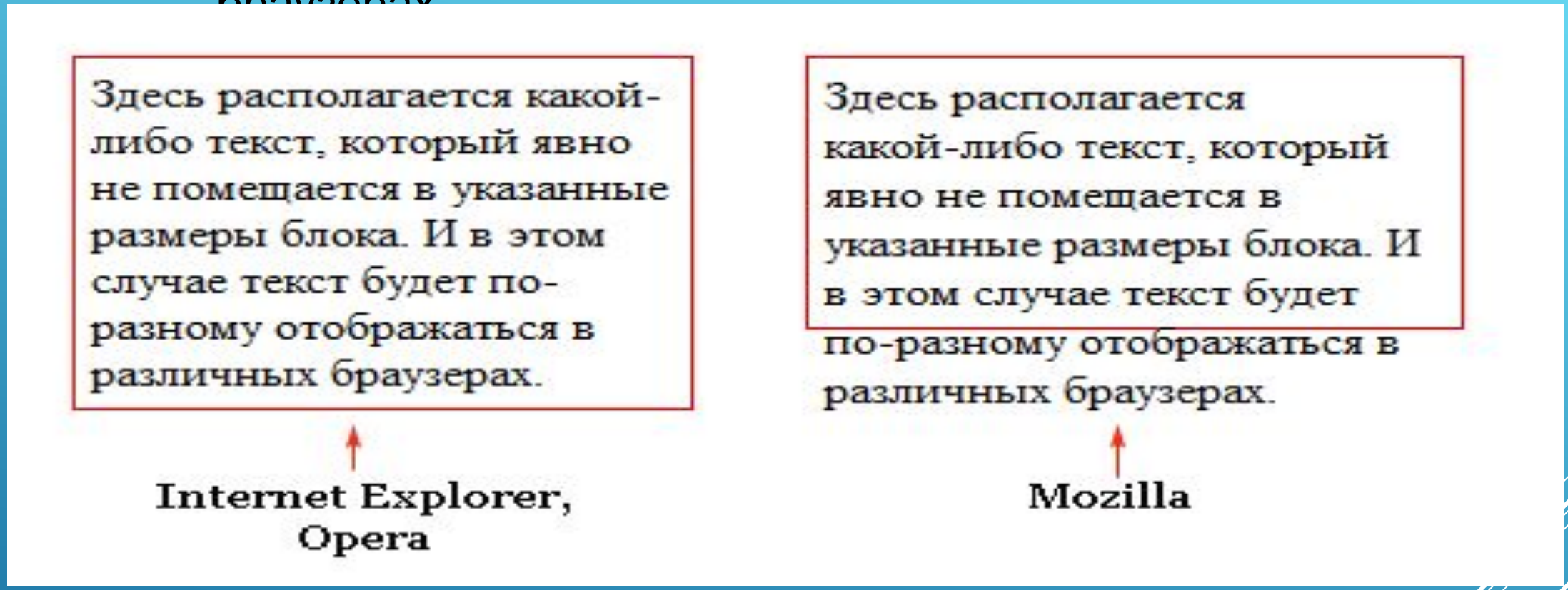
```
<body>
<p>Параграф 1</p>
<p id="p2">Параграф 2</p>
<p>Параграф 3</p>
<p id="p4">Параграф 4</p>
<p>Параграф 5</p>
 </body>
```

Давайте зададим для второго параграфа свойство display:none, а для четвертого - свойство visibility:hidden:

```
#p2{ display:none;}
#p4{ visibility:hidden;}
```

Как видите, второй параграф отсутствует, а четвертый - невидим, но место под него оставлено. В этом и заключается разница. Свойство visibility так же, как и свойство display, чаще всего применяется совместно с javascript. (чтобы показать или скрыть необходимые элементы)

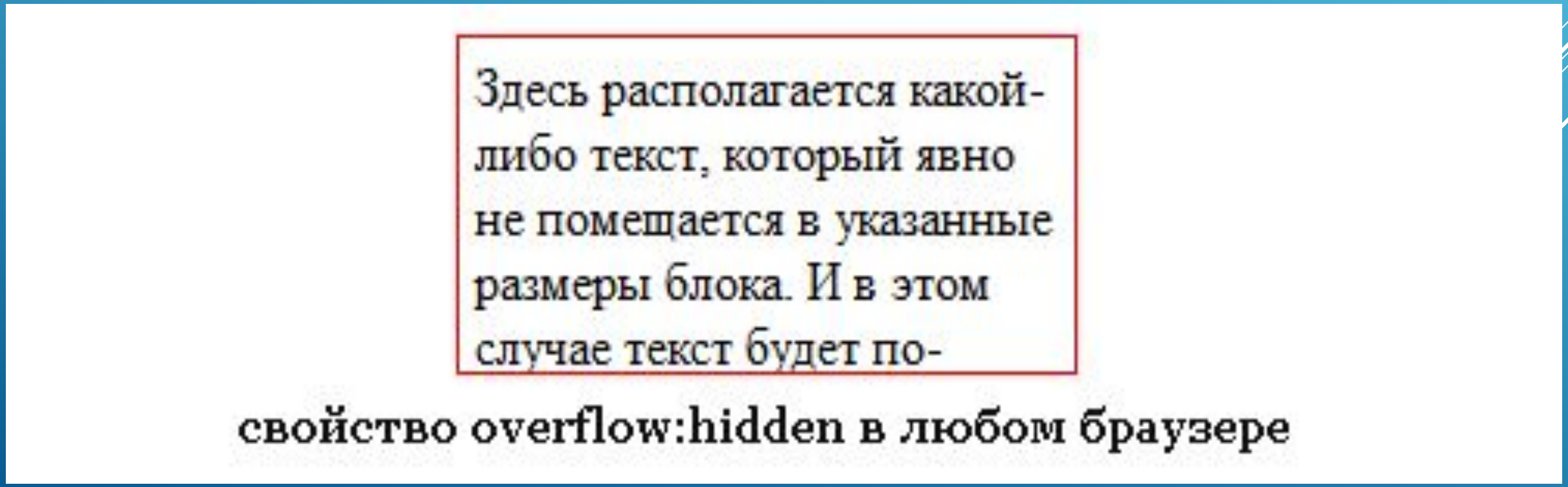
Отображение содержимого блоков в разных браузерах



Как вы думаете, что будет, если блочному элементу задать меньшие размеры, чем его содержимое? Ваш ответ будет зависеть от того, в каком браузере вы работаете. Предположим мы задали блоку размер 200x100 пикселов и поместили в него текст, явно превышающий эти размеры. Выше приведен рисунок, где отображено поведение различных браузеров в такой ситуации

Как видите, одни браузеры растягивают блоки, чтобы вместить содержимое, а другие перекрывают блок содержимым. Конечно, лучшего всего контролировать размеры объектов, но это не всегда возможно. Как же быть тогда? Воспользоваться свойством overflow. Это свойство может принимать четыре значения:

visible - если содержимое превышает размеры блока, оно все-равно останется на экране. Результат будет такой же, как на рисунке выше.



свойство overflow:hidden в любом браузере

scroll - блок будет снабжен полосами прокрутки, причем как горизонтальной, так и вертикальной (смотря какую прокрутку мы зададим по X или У оси).

auto - блок будет снабжен только теми полосами прокрутки, которые необходимы.

Border , Padding , Margin

```
<body>
    <p>Текст в параграфе.</p>
    <p>Текст в параграфе.</p>
</body>
```

Универсальное свойство border позволяет одновременно установить толщину, стиль и цвет границы вокруг элемента. Значения могут идти в любом порядке, разделяясь пробелом, браузер сам определит, какое из них соответствует нужному свойству. Для установки границы только на определенных сторонах элемента, воспользуйтесь
свойствами border-top, border-bottom, border-left,border-right.

Значение border-width определяет толщину границы. Для управления ее видом предоставляется несколько значений border-style.

Для того, чтобы увидеть отступы, поля и границы, зададим границу

```
p{  border:1px solid red;}
```

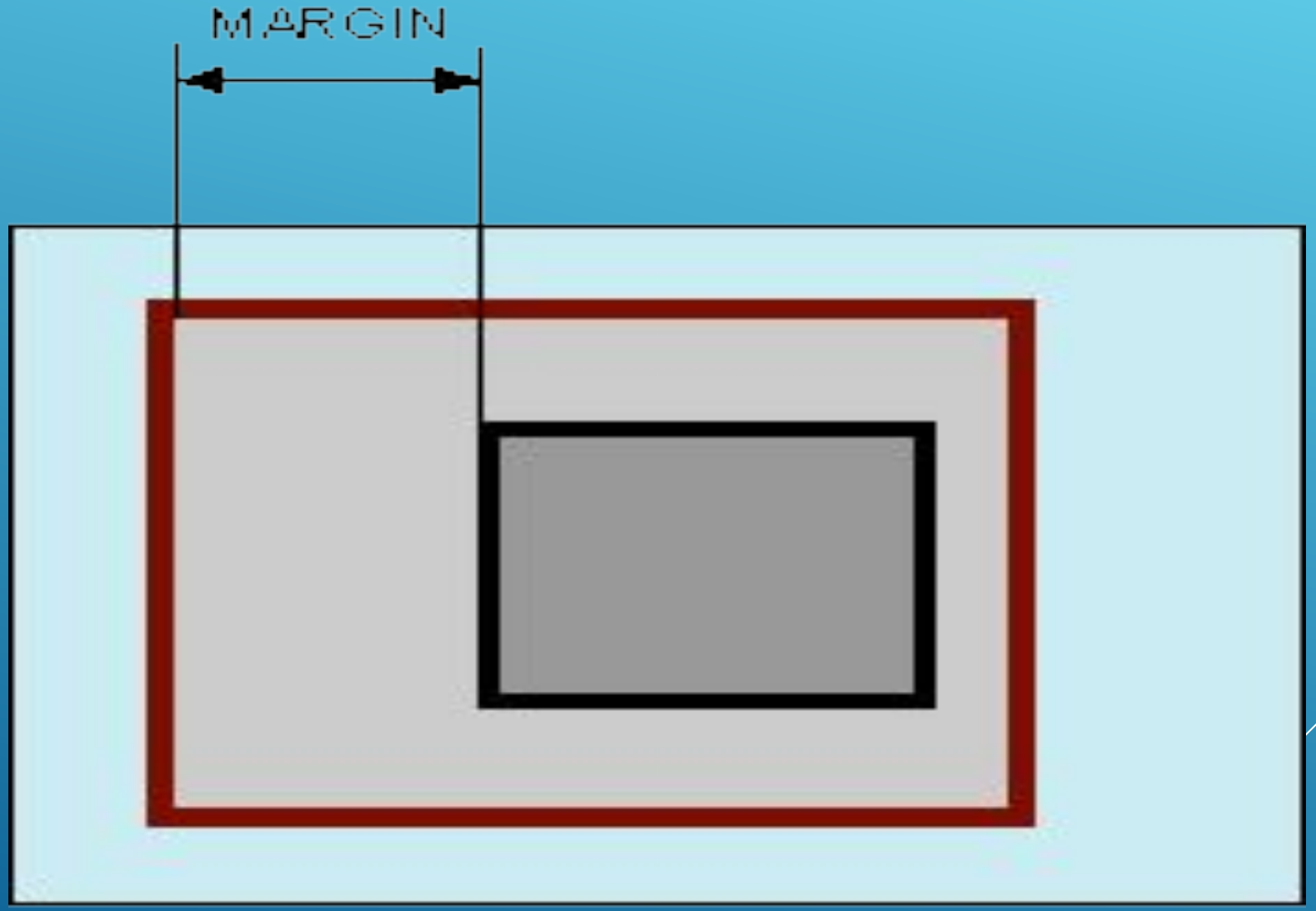
Текст в параграфе.

Текст в параграфе.

Отступы от границы задаются свойством padding, оно позволяет задать величину внутреннего отступа сразу для всех сторон элемента или определить ее только для указанных сторон.

Добавляем к нашему свойству p{ padding:10px;}

Далее задаем внешние отступы – Margin . Отступом является пространство от границы текущего элемента до внутренней границы его родительского элемента



Если у элемента нет родителя, отступом будет расстояние от края элемента до края окна браузера с учетом того, что у самого окна по умолчанию тоже установлены отступы. Чтобы от них избавиться, следует устанавливать значение margin для селектора <body> равное нулю.

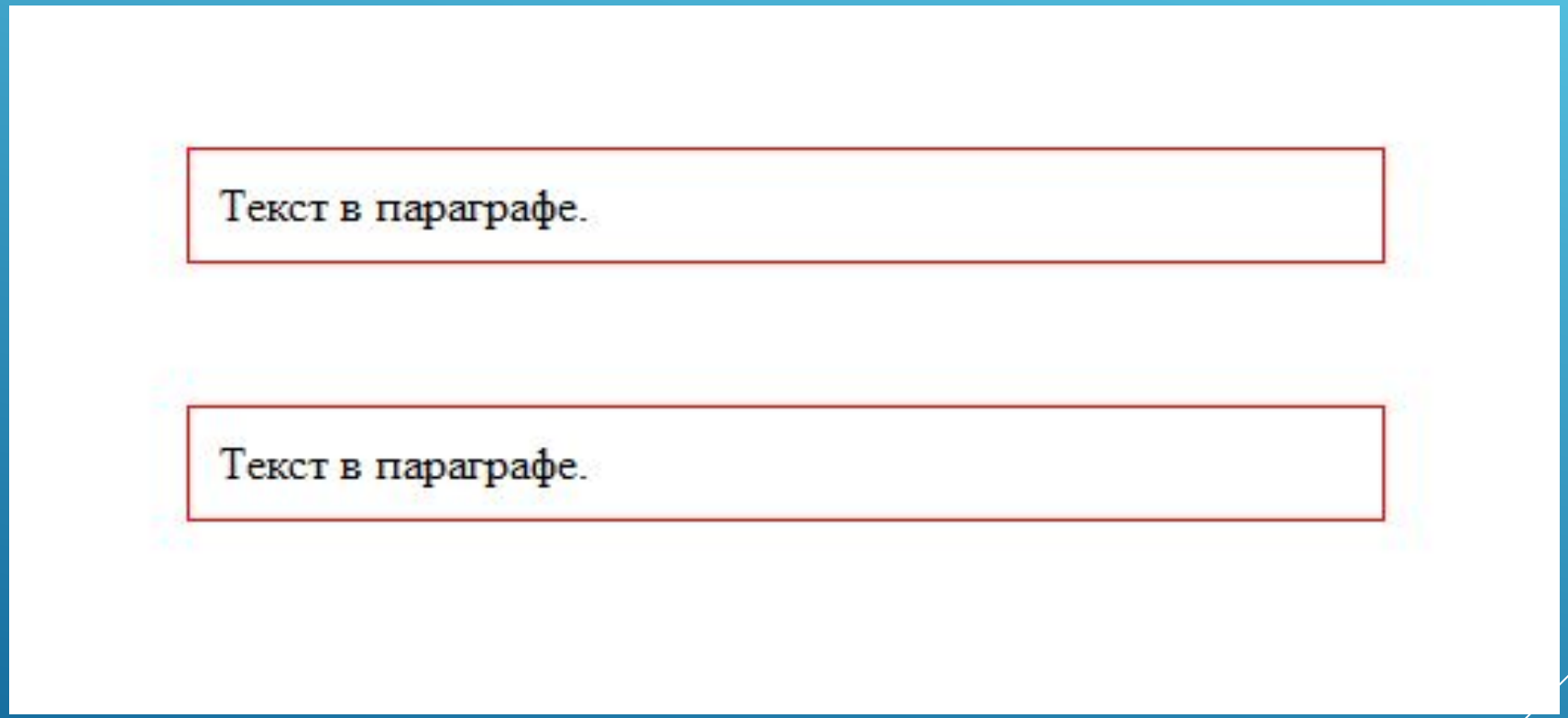
Свойство margin позволяет задать величину отступа сразу для всех сторон элемента или определить ее только для указанных сторон.

Величину отступов можно указывать в пикселах (px), процентах (%) или других допустимых для CSS единицах. Значение может быть как положительным, так и отрицательным числом.

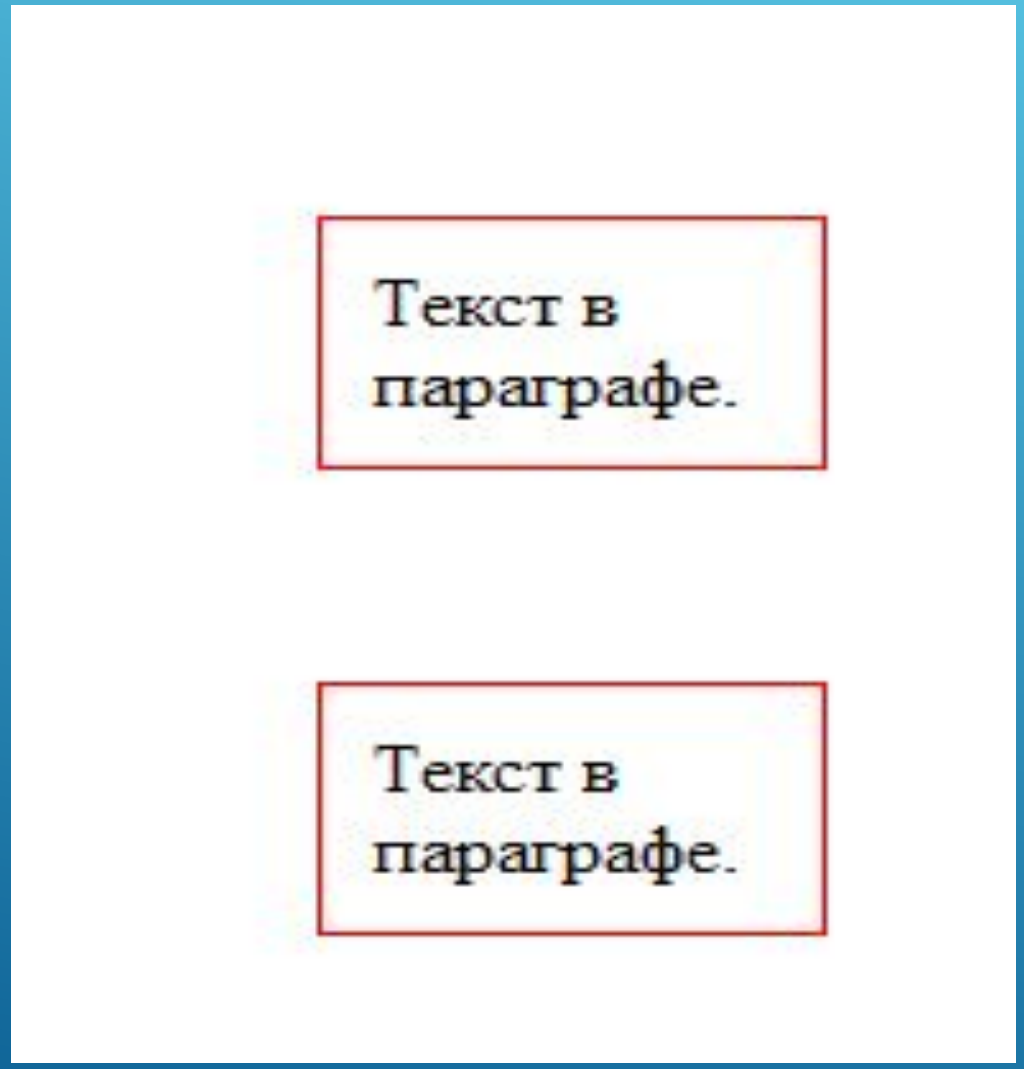
Auto Указывает, что размер отступов будет автоматически рассчитан браузером.

Inherit Наследует значение родителя.

И снова добавляем свойства к нашему параграфу p{ margin:50px;}



А теперь добавляем ширину и высоту нашим блокам p{ width:100px; height:50px;}



Плавающие блоки

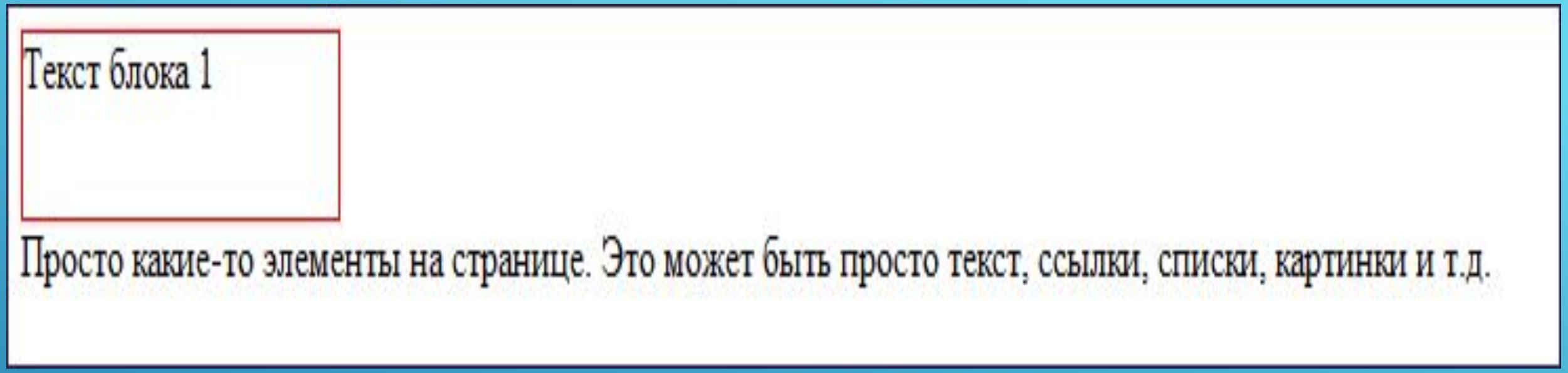
Эти блоки нельзя позиционировать с точностью до пиксела, как в предыдущих схемах, но именно эта схема позиционирования очень распространена. Без плавающих блоков обходится редкий сайт, а уж сделать "резиновую" верстку сайта без них и вовсе невозможно.
Плавающие блоки определяются свойством float, который определяет будет ли блок плавающим и в какую сторону он будет перемещаться. Возможны три варианта:

left - блок прижимается к левому краю, остальные элементы обтекают его с правой стороны.

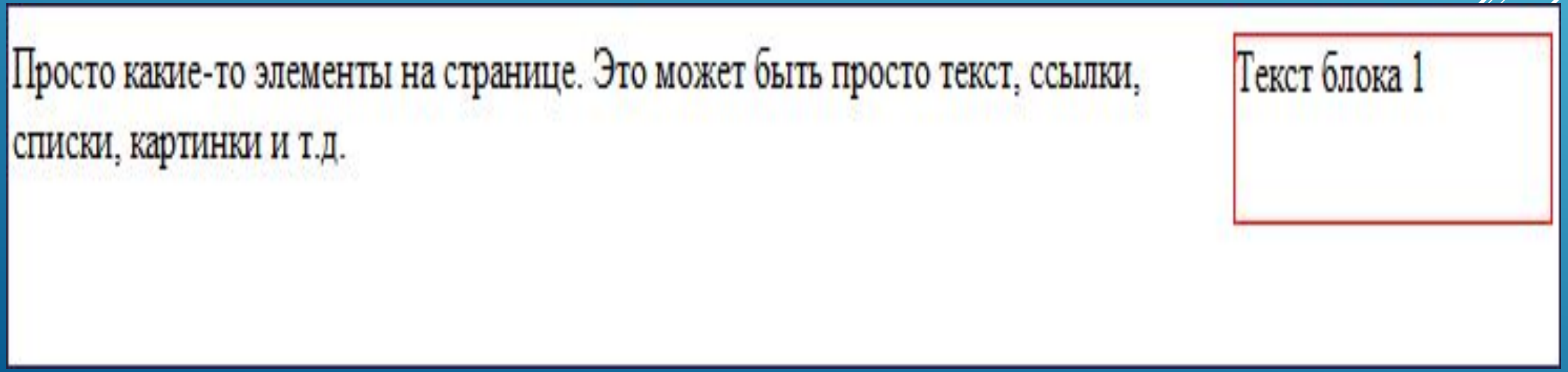
right - блок прижимается к правому краю, остальные элементы обтекают его с левой стороны.

none - блок не перемещается и позиционируется согласно свойству position.

```
<body>
<div id="blok1">Текст блока 1</div>
Просто какие-то элементы на странице. Это может быть просто
текст, ссылки, списки, картинки и т.д.
</body>
#blok1{
border:1px solid red;
width:150px;
height:50px;
}
```



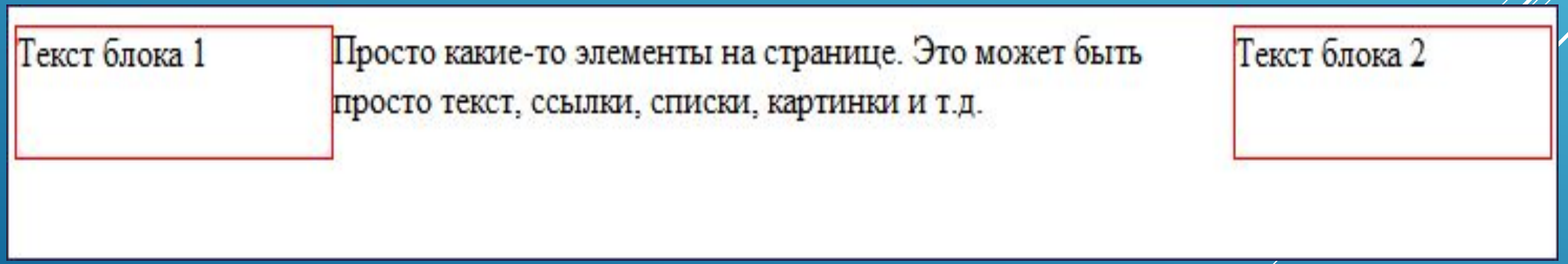
#blok1{float:right;}



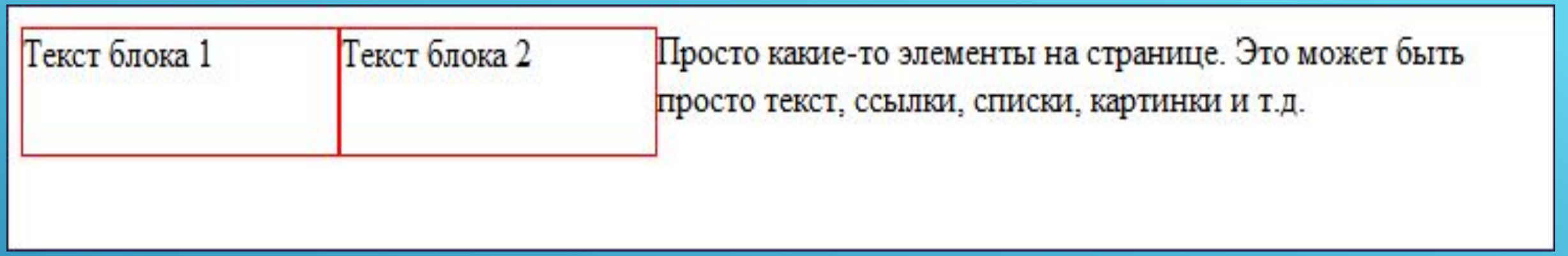
Добавим еще один такой же блок но с обратным обтеканием

```
<body>
    <div id="blok1">Текст блока 1</div>
    <div id="blok2">Текст блока 2</div>
    Просто какие-то элементы на странице. Это может быть просто
    текст, ссылки, списки, картинки и т.д.
</body>
#blok1{
border:1px solid red;
width:150px;
height:50px;
float:left;
}
```

```
#blok2{
border:1px solid red;
width:150px;
height:50px;
float:right;
}
```



Теперь поменяйте их обтекание на одинаковое значение float:left;



А что делать, если мы хотим, чтобы блоки были прижаты к правому краю, но располагались бы один под другим. Для этого существует свойство clear, которое определяет, какие стороны плавающего блока не могут соседствовать с другими плавающими блоками.

У этого свойства может быть задано одно из четырех значений:

left - блок должен располагаться ниже всех левосторонних блоков.

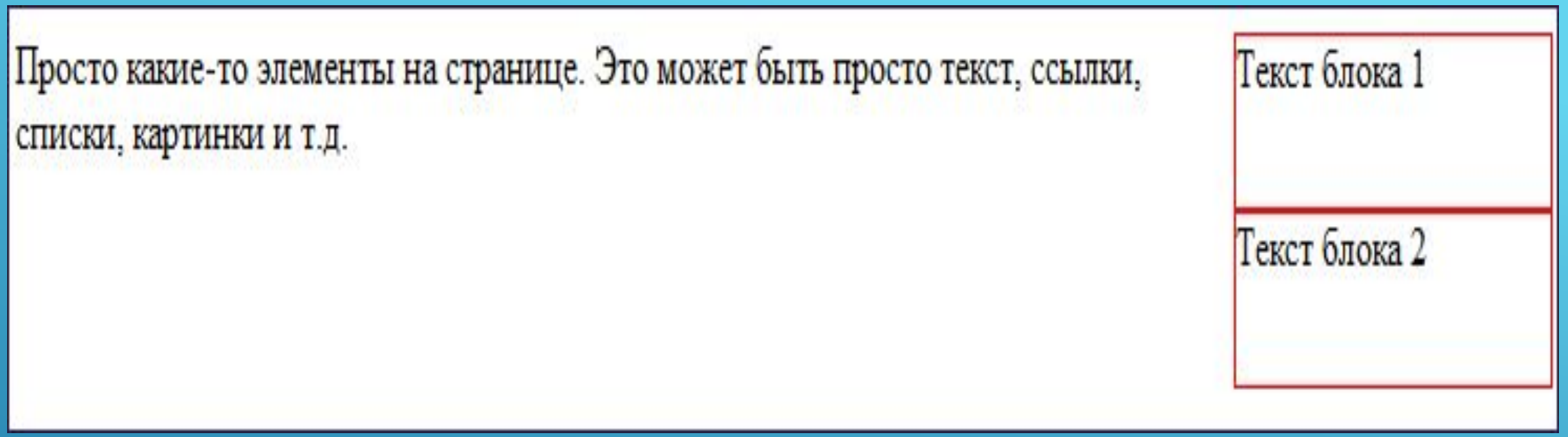
right - блок должен располагаться ниже всех правосторонних блоков.

both - блок должен располагаться ниже всех плавающих блоков.

none - никаких ограничений нет, это значение по умолчанию.

```
#blok1{
border:1px solid red;
width:150px;
height:50px;
float:right;
}

#blok2{
border:1px solid red;
width:150px;
height:50px;
float:right;
clear:right;
}
```



ДЗ

https://learn.javascript.ru/display

httphttp://http://htmlbookhttp://htmlbook.http://htmlbook.ruhttp://htmlbook.ru/http://htmlbook.ru/html

httphttp://http://htmlbookhttp://htm

ПОЗИЦИОНИРОВАНИЕ

Если визуально разделить нашу страницу на прямоугольные блоки, то мы получим четыре блока: шапка сайта, меню, контент и низ сайта. Таким образом, мы имеем

четыре div-а

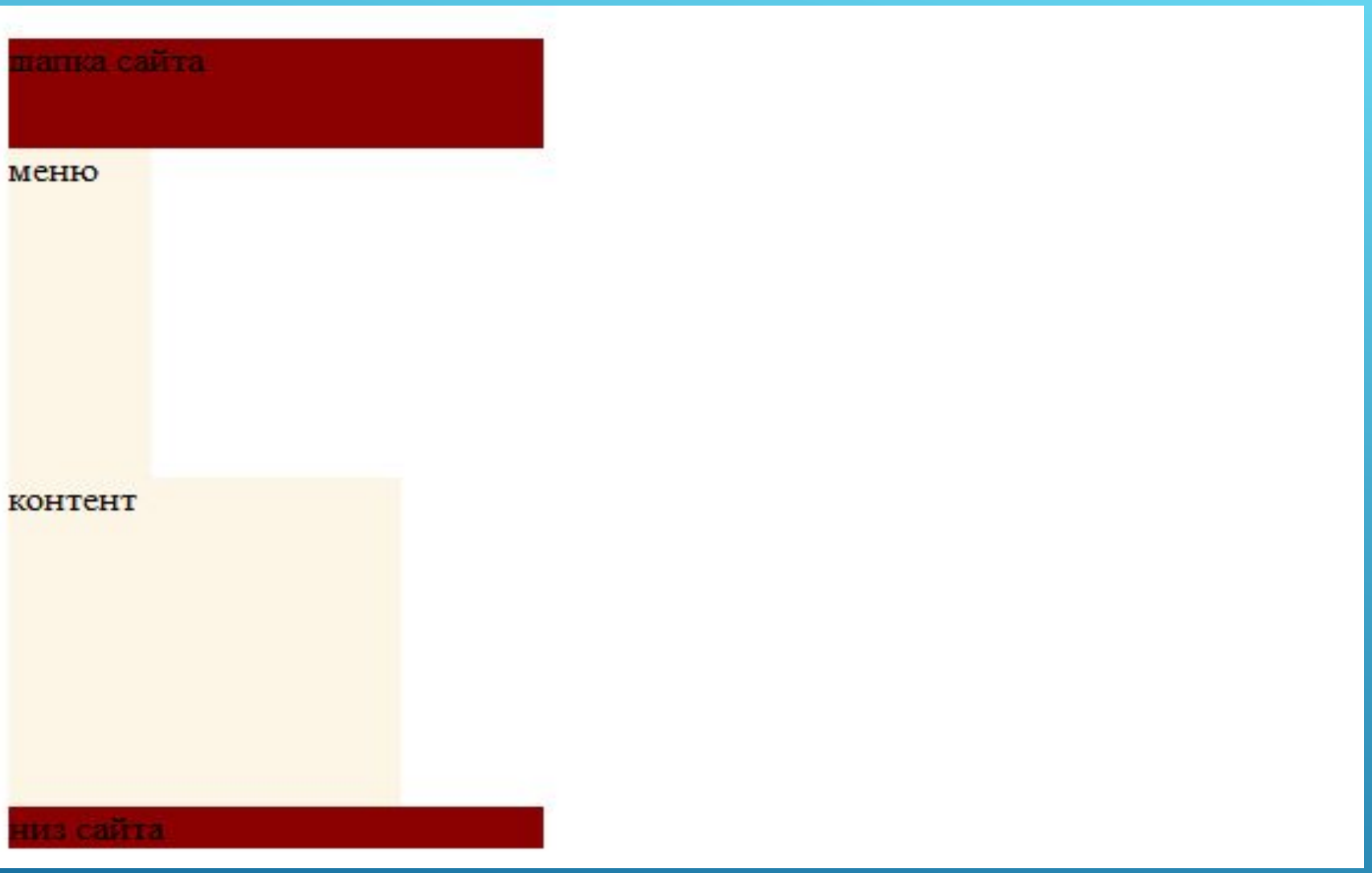
```
<body>
<div id="header">шапка сайта</div>
<div id="menu">меню</div>
<div id="content">контент</div>
<div id="footer">низ сайта</div>
  </body>
```

```
#header{
background:darkred;
width:715px;
height:100px;
}

#menu{
background:oldlace;
width:190px;
height:300px;
}
```

```
#content{
background:oldlace;
width:525px;
height:300px;
}

#footer{
background:darkred;
width:715px;
height:30px;
}
```



Такое позиционирование элементов называется позиционированием в нормальном потоке. Это значит, что все элементы отображаются в окне браузера сверху вниз, по вертикали, в том порядке, в каком они следуют друг за другом в html-коде.

В CSS же нам предоставляются и другие схемы позиционирования: абсолютное позиционирование, относительное позиционирование, плавающая

Для определения схемы позиционирования используется свойство position, оно может принимать четыре значения, соответствующие выбранной схеме позиционирования:

static - блок позиционируется в нормальном потоке. Это значение по умолчанию.

relative - относительное позиционирование (относительно нормального потока).

absolute - абсолютное позиционирование

fixed - фиксированное позиционирование (фиксируется относительно области просмотра).

Абсолютное позиционирование

При этой схеме позиционирования расположение блока на странице не зависит от того, в каком месте html-кода расположен этот блок. Расположение каждого блока задается указанием, в каком месте экрана отобразить данный блок. Для этого существуют четыре свойства:

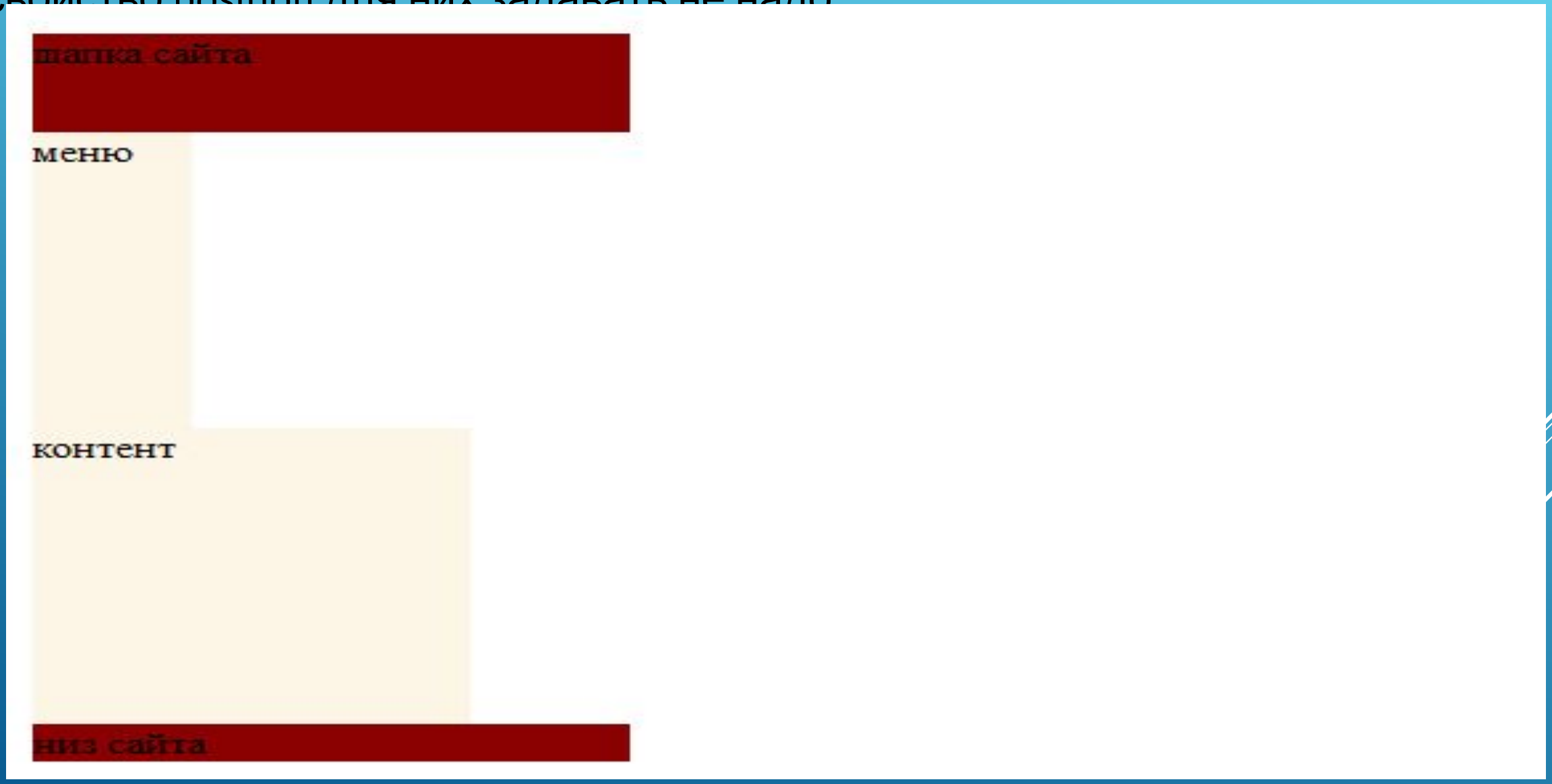
left - указывает на сколько надо сместить блок относительно левого края окна.

right - указывает на сколько надо сместить блок относительно правого края окна.

top - указывает на сколько надо сместить блок относительно верхнего края окна.

bottom - указывает на сколько надо сместить блок относительно нижнего края окна.

Вернемся к нашему примеру. Наши
блоки header, menu и footer позиционируются в нормальном потоке, поэтому
свойство position для них задавать не надо



А вот блок content нужно расположить в другом месте, поэтому для него мы укажем свойство position:absolute и зададим смещение: от левого края окна на ширину блока menu, т.е. на 190 пикселей, а от верхнего края окна на высоту блока header, т. е. на 100 пикселей.

```
#content{
position:absolute;
left: 190px;
top:100px;
}
```

По умолчанию для элемента body определены поля, а мы их не учитывали при задании свойств смещения. Чтобы решить эту проблему, достаточно задать для body свойство margin:0px, т.е. явно указать размер полей (в нашем примере - их отсутствие) и свойство кот. Будет указывать на родительское отношение к позиционированию !!! Добавим это в таблицу стилей: body{margin:0px; position:relative }


шапка сайта
меню
контент
низ сайта

Фиксированные блоки

при фиксированном позиционировании блок фиксируется относительно области просмотра. У фиксированных блоков есть один существенный недостаток: они не поддерживаются браузерами Internet Explorer. А потому использовать их крайне аккуратно.

```
#blok{
position:fixed;
left:0px;
top:0px;
}
```

Блок с идентификатором "blok" будет при прокрутке страницы оставаться на месте.

Многослойность

Представьте себе множество листков бумаги, на каждом из которых что-то написано или нарисовано. Мы видим только содержание верхнего листа, но если мы его снимем, то увидим содержание следующего листа и т.д.

Также и в CSS, мы можем создать несколько слоев, на каждом разместить необходимые элементы и пронумеровать слои с помощью свойства z-index. Чем больше номер, тем выше слой находится в стопке слоев. Например, если сделать 6 слоев, то пользователь сначала увидит именно слой z-index:6.

Чтобы было визуально понятно давайте изменим нашему блоку контент цвет фона

```
#menu {Position: relative; z-index:1;}

#content{ Left;0;background: green; z-index:2}
```

```
<!doctype html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" href="css/all.css" type="text/css"/>
<script src="http://code.jquery.com/jquery-1.8.3.js"></script>
<script type="text/javascript" src="js/main.js"></script>
<title>Title</title>
</head>
<body> текст </body>
```

```
body{
background: blue;
}
```

Теперь, давайте сделаем текст на странице белым цветом:

```
body{
background: blue;
color: white;
}
```

Теперь сделаем цвета заголовков красным (для h1) и желтым (для h2):

```
body{
background: blue;
color: white;
}
h1{color:red;}
h2{color:yellow;}
```

Группировка селекторов

Если блоки объявления стилей для нескольких селекторов совпадают (например, мы хотим, чтобы заголовки первых трех уровней были зеленого цвета), то их можно сгруппировать. Для этого селекторы, к которым будет применяться один стиль, нужно перечислить через запятую. Пример:

```
h1, h2, h3{  color:green;}
```

Если кроме цвета, мы хотим задать нашим заголовкам размер. Тогда мы можем просто дописать в нашу таблицу стилей:

```
h1, h2, h3{  color:green;}

h1{  font-size:18px;}

h2{  font-size:16px;}

h3{  font-size:14px;}
```

У наших заголовков будет указанный размер, но все они будут зеленого цвета.

Color

в компьютерном дизайне чаще всего используют цветовую модель RGB. В ее основе лежат три цвета: red - красный, green - зеленый, blue - синий. С помощью смешивания этих трех цветов можно получить почти весь видимый спектр. Для указания RGB-цвета в HTML используется шестнадцатеричная система счисления. Начинается код цвета со спецсимвола # (что и указывает на шестнадцатеричный код), а далее следует шесть символов. Первые два отвечают за количество красного оттенка, третий и четвертый за насыщенность зеленого, а пятый и шестой - синего. Так, #FF0000 - красный цвет, #00FF00 - зеленый, #0000FF - синий, а #FFFFFF - белый.

Значениями свойства color могут быть именные цвета (red, blue...), шестнадцатеричные коды цветов (#FF0000) и десятичные коды цвета в модели RGB (rgb(255, 0, 0)). Подробнее о цвете читайте на странице цвета для web. Итак, задать цвет текста для элемента можно тремя способами:

```
body{  color:green;}

h1{  color:#FF0000;}
```

Свойство color является наследуемым.

```
<body>
<h1>Заголовок</h1>
<p>Здесь текст параграфа.</p>
Здесь просто текст.
 </body>
body{  color:green;}
h1{  color:red;}
p{  color:white;}
```

Фон - background

На самом деле это группа свойств, так или иначе связанная с фоном. При помощи CSS фон можно задать не только странице, но и заголовку, и абзацу, и любому другому элементу. Пусть у нас есть html-страница с таким кодом:

```
<body>
  Здесь содержимое документа
</body>
```

Рассмотрим группу свойств background:

background-color - задает цвет фона. По умолчанию не наследуется, но его можно сделать наследуемым, если в качестве значения указать значение inherit.

```
body{
background-color:#243CED;
color:yellow;
}
```

background-image - задает фоновое изображение. Значением свойства является URL графического файла. Формат задания следующий: сначала идет обозначение функции url, а затем в круглых скобках указывается путь к файлу. Путь к файлу указывается относительно таблицы стилей.

```
body{
  background-image:url(picture.gif);
  background-color:#243CED;
  color:yellow;
}
```

background-repeat - задает возможность повторения фонового изображения. В качестве фонового изображения может выступать как цельное изображение (например, шапка сайта), так и маленькое изображение, которое должно замостить собой все пространство элемента. Данное свойство как раз и указывает, повторять ли изображение и, если да, то как именно повторять. Возможны 4 варианта:

repeat - повторять изображение по горизонтали и вертикали.

repeat-x - повторять изображение только по горизонтали.

repeat-y - повторять изображение только по вертикали.

no-repeat - не повторять изображение.

По умолчанию используется значение repeat

```
body{
  background-image:url(picture.gif);
  background-repeat:no-repeat;
  background-color:#243CED;
  color:yellow;
}
```

background-position - задает расположение элемента относительно окна браузера. Значения можно задавать в процентах, в единицах длины и при помощи ключевых слов.

```
body{
  background-image:url(picture.gif);
  background-repeat:no-repeat;
  background-color:#243CED;
  background-position:10% 30%;
  color:yellow;
}
```

Сокращенная запись свойства background

В CSS для многих свойств существует сокращенная запись. В этом случае значения всех свойств перечисляются через пробел в произвольном порядке.

```
body{
  background:url(picture.gif) no-repeat #33CCFF center top;
  color:yellow;
}
```