**Шаблоны HTML страницы с размерами**

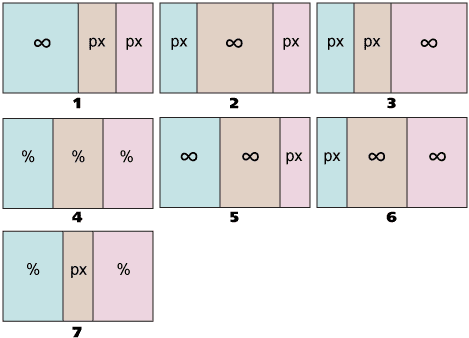
**Резиновый трёхколоночный макет**

10.08.2011

Влад Мержевич

Все остальные макеты (*не только резиновые 3-х колоночные*) нужно выбирать, нажимая на ссылки вверху-справа, на странице - http://htmlbook.ru/samlayout/tipovye-makety/rezinovyi-trekhkolonochnyi-maket

Резиновый макет с тремя колонками, пожалуй, самый гибкий и настраиваемый из существующих макетов. Сочетание процентов и пикселов для указания ширины колонок позволяет делать разные макеты, выбирая их под свои задачи. На рис. 5.17 представлены варианты трёхколоночных макетов, для удобства они пронумерованы.



*Рис. 5.17. Трёхколоночные макеты*

Здесь символ процентов (%) означает, что ширина колонки задана в процентах от ширины макета, px — ширина колонки указана в пикселах, а знак бесконечности (∞), что колонка занимает оставшееся пространство. Несмотря на обилие разных макетов, принципы их построения остаются одинаковыми и включают два основных способа: позиционирование и плавающие элементы. Также можно воспользоваться таблицами для создания колонок одинаковой высоты.

Будут показаны 3-и типа написания шаблонов с использованием:

- плавающих элементов,

- позиционирования,

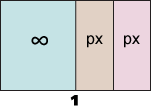
- таблицы в качестве колонок.

**Использование плавающих элементов**

Плавающие элементы это универсальный метод построения разнообразных трёхколоночных макетов. В отличие от позиционирования(*ниже*) он позволяют не заботиться о проблеме подвала(*футера*), поскольку подвал всегда будет располагаться на своём месте под самой высокой колонкой.

Построение колонок происходит с помощью свойства float в сочетании со свойствами margin и width. В зависимости от выбранного макета меняется и порядок колонок в коде, вначале всегда следуют слои, к которым добавляется float.

**Макет № 1. Резиновая первая колонка**



Ширина второй и третьей колонки указывается в пикселах, а их положение через свойство float со значением right. Для первой колонки также требуется задать свойство margin-right со значением равным суммарной ширине остальных колонок (пример 5.22).

**Пример 5.22. Макет № 1**

XHTML 1.0CSS 2.1IECrOpSaFx

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

**<html** xmlns="http://www.w3.org/1999/xhtml"**>**

**<head>**

**<meta** http-equiv="Content-Type" content="text/html; charset=utf-8" **/>**

**<title>**Макет 1**</title>**

**<style** type="text/css"**>**

.header, .footer { background: #D5BAE4; }

.layout {

overflow: hidden; /\* Отменяем обтекание \*/

}

.col1 {

background: #C7E3E4; /\* Цвет фона \*/

margin-right: 300px; /\* Сдвиг влево на ширину колонок 2 и 3 \*/

}

.col2 {

background: #E0D2C7; width: 200px;

float: right; /\* Обтекание слева \*/

}

.col3 {

background: #ECD5DE; width: 100px;

float: right; /\* Обтекание слева \*/

}

**</style>**

**</head>**

**<body>**

**<div** class="header"**>**Шапка сайта**</div>**

**<div** class="layout"**>**

**<div** class="col3"**>**Колонка 3**</div>**

**<div** class="col2"**>**Колонка 2**</div>**

**<div** class="col1"**>**Колонка 1**</div>**

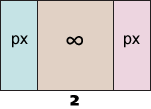
**</div>**

**<div** class="footer"**>**Подвал**</div>**

**</body>**

**</html>**

**Макет № 2. Резиновая средняя колонка**



Ширина первой и третьей колонки задана в пикселах, а их положение — через свойство float со значением left для первой колонки и right для третьей. Средняя колонка, чтобы она сохраняла свой вид, содержит универсальное свойство margin, задающее отступ слева и справа (пример 5.23).

**Пример 5.23. Макет № 2**

XHTML 1.0CSS 2.1IECrOpSaFx

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

**<html** xmlns="http://www.w3.org/1999/xhtml"**>**

**<head>**

**<meta** http-equiv="Content-Type" content="text/html; charset=utf-8" **/>**

**<title>**Макет 2**</title>**

**<style** type="text/css"**>**

.header, .footer { background: #D5BAE4; }

.layout { overflow: hidden; }

.col1 { background: #C7E3E4; float: left; width: 200px; }

.col2 { background: #E0D2C7;

margin: 0 100px 0 200px; /\* Отступ справа и слева \*/

}

.col3 { background: #ECD5DE; width: 100px; float: right; }

**</style>**

**</head>**

**<body>**

**<div** class="header"**>**Шапка сайта**</div>**

**<div** class="layout"**>**

**<div** class="col1"**>**Колонка 1**</div>**

**<div** class="col3"**>**Колонка 3**</div>**

**<div** class="col2"**>**Колонка 2**</div>**

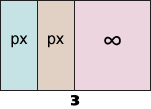
**</div>**

**<div** class="footer"**>**Подвал**</div>**

**</body>**

**</html>**

**Макет № 3. Резиновая третья колонка**



Положение первой и второй колонки указывается через свойство float со значением left, для третьей колонки нужно установить отступ слева (margin-left) на суммарную ширину остальных колонок (пример 5.24).

**Пример 5.24. Макет № 3**

XHTML 1.0CSS 2.1IECrOpSaFx

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

**<html** xmlns="http://www.w3.org/1999/xhtml"**>**

**<head>**

**<meta** http-equiv="Content-Type" content="text/html; charset=utf-8" **/>**

**<title>**Макет 3**</title>**

**<style** type="text/css"**>**

.header, .footer { background: #D5BAE4; }

.layout { overflow: hidden; }

.col1 { background: #C7E3E4; float: left; width: 100px; }

.col2 { background: #E0D2C7; float: left; width: 200px; }

.col3 { background: #ECD5DE;

margin-left: 300px; /\* Отступ слева на ширину колонок 1 и 2 \*/

}

**</style>**

**</head>**

**<body>**

**<div** class="header"**>**Шапка сайта**</div>**

**<div** class="layout"**>**

**<div** class="col1"**>**Колонка 1**</div>**

**<div** class="col2"**>**Колонка 2**</div>**

**<div** class="col3"**>**Колонка 3**</div>**

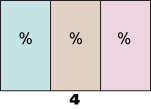
**</div>**

**<div** class="footer"**>**Подвал**</div>**

**</body>**

**</html>**

**Макет № 4. Ширина всех колонок задана в процентах**



Для построения этого макета подойдёт множество вариантов, основанных на предыдущих макетах, только вместо пикселов следует указать проценты. Ещё один способ показан в примере 5.25, где используется только свойство float и width.

**Пример 5.25. Макет № 4**

XHTML 1.0CSS 2.1IECrOpSaFx

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

**<html** xmlns="http://www.w3.org/1999/xhtml"**>**

**<head>**

**<meta** http-equiv="Content-Type" content="text/html; charset=utf-8" **/>**

**<title>**Макет 4**</title>**

**<style** type="text/css"**>**

.header, .footer { background: #D5BAE4; }

.layout { overflow: hidden; }

.layout **DIV** { float: left; }

.col1 { background: #C7E3E4; width: 20%; }

.col2 { background: #E0D2C7; width: 60%; }

.col3 { background: #ECD5DE; width: 20%; }

**</style>**

**</head>**

**<body>**

**<div** class="header"**>**Шапка сайта**</div>**

**<div** class="layout"**>**

**<div** class="col1"**>**Колонка 1**</div>**

**<div** class="col2"**>**Колонка 2**</div>**

**<div** class="col3"**>**Колонка 3**</div>**

**</div>**

**<div** class="footer"**>**Подвал**</div>**

**</body>**

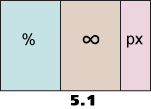
**</html>**

**Макет № 5. Ширина двух колонок резиновая**

На рис. 5.19 было продемонстрировано, что макет № 5, у которого две резиновые колонки, можно трактовать по-разному.

1. Ширина первой колонки указана в процентах от ширины макета, третьей колонки в пикселах, а средняя колонка занимает оставшееся место.
2. Ширина второй колонки указана в процентах от ширины макета, третьей колонки в пикселах, а первая колонка занимает оставшееся место.
3. Суммарная ширина двух резиновых колонок принимается за 100%, и ширина колонок указывается в процентах от этой величины.

**Ширина средней колонки вычисляемая**



Здесь ширина первой колонки указывается в процентах, а её положение через свойство float со значением left, для третьей колонки ширина задана в пикселах, а значение свойства float как right. Для средней колонки остаётся только установить отступы слева и справа на ширину колонок (пример 5.26).

**Пример 5.26. Макет 5.1**

XHTML 1.0CSS 2.1IECrOpSaFx

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

**<html** xmlns="http://www.w3.org/1999/xhtml"**>**

**<head>**

**<meta** http-equiv="Content-Type" content="text/html; charset=utf-8" **/>**

**<title>**Макет 5.1**</title>**

**<style** type="text/css"**>**

.header, .footer { background: #D5BAE4; }

.layout { overflow: hidden; }

.col1 { background: #C7E3E4; width: 40%; float: left; }

.col2 { background: #E0D2C7; margin: 0 200px 0 40%; }

.col3 { background: #ECD5DE; width: 200px; float: right; }

**</style>**

**</head>**

**<body>**

**<div** class="header"**>**Шапка сайта**</div>**

**<div** class="layout"**>**

**<div** class="col1"**>**Колонка 1**</div>**

**<div** class="col3"**>**Колонка 3**</div>**

**<div** class="col2"**>**Колонка 2**</div>**

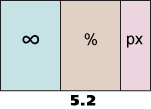
**</div>**

**<div** class="footer"**>**Подвал**</div>**

**</body>**

**</html>**

**Ширина первой колонки вычисляемая**



Это наиболее хитрый макет, поскольку для первой колонки её ширина напрямую не указывается. Но чтобы ограничить контент необходимо для свойства margin-right указать значение совмещающее проценты и пикселы. В CSS2, как уже говорилось, подобное задать нельзя, поэтому добавим внутрь слоя col1 ещё один слой с именем wrap и добавим отступ каждому из этих слоёв. Одному в процентах, другому в пикселах (пример 5.27).

**Пример 5.27. Макет 5.2**

XHTML 1.0CSS 2.1IECrOpSaFx

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

**<html** xmlns="http://www.w3.org/1999/xhtml"**>**

**<head>**

**<meta** http-equiv="Content-Type" content="text/html; charset=utf-8" **/>**

**<title>**Макет 5.2**</title>**

**<style** type="text/css"**>**

.header, .footer { background: #D5BAE4; }

.layout { overflow: hidden; }

.col1 { margin-right: 40%; }

.col1 .wrap { margin-right: 200px; background: #C7E3E4; }

.col2 { background: #E0D2C7; width: 40%; float: right; }

.col3 { background: #ECD5DE; width: 200px; float: right; }

**</style>**

**</head>**

**<body>**

**<div** class="header"**>**Шапка сайта**</div>**

**<div** class="layout"**>**

**<div** class="col3"**>**Колонка 3**</div>**

**<div** class="col2"**>**Колонка 2**</div>**

**<div** class="col1"**>**

**<div** class="wrap"**>**

Колонка 1

**</div>**

**</div>**

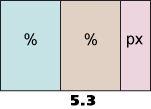
**</div>**

**<div** class="footer"**>**Подвал**</div>**

**</body>**

**</html>**

**Ширина двух колонок в процентах**



В этом макете общая ширина резиновых колонок принимается за 100%, поэтому нам потребуется дополнительный слой, относительно которого будет отсчитываться ширина внутренних колонок. Этот слой с именем rubber совместно со слоем col3 работает как двухколоночный макет, а внутренние слои col1 и col2 выстраиваются по горизонтали за счёт применения свойства float (пример 5.28).

**Пример 5.28. Макет 5.3**

XHTML 1.0CSS 2.1IECrOpSaFx

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

**<html** xmlns="http://www.w3.org/1999/xhtml"**>**

**<head>**

**<meta** http-equiv="Content-Type" content="text/html; charset=utf-8" **/>**

**<title>**Макет 5.3**</title>**

**<style** type="text/css"**>**

.header, .footer { background: #D5BAE4; }

.layout { overflow: hidden; }

.rubber { margin-right: 200px; }

.col1 { background: #C7E3E4; width: 60%; float: left; }

.col2 { background: #E0D2C7; width: 40%; float: left; }

.col3 { background: #ECD5DE; width: 200px; float: right; }

**</style>**

**</head>**

**<body>**

**<div** class="header"**>**Шапка сайта**</div>**

**<div** class="layout"**>**

**<div** class="col3"**>**Колонка 3**</div>**

**<div** class="rubber"**>**

**<div** class="col2"**>**Колонка 2**</div>**

**<div** class="col1"**>**Колонка 1**</div>**

**</div>**

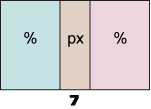
**</div>**

**<div** class="footer"**>**Подвал**</div>**

**</body>**

**</html>**

Макеты 6, 7 и их разновидности строятся по тому же принципу, что и макет № 5, за исключением макета № 7, в котором ширина левой и правой колонки резиновая и равна между собой.



Вначале готовим основу, указываем порядок слоёв в HTML-коде.

**<div** class="layout"**>**

**<div** class="col1"**>**Колонка 1**</div>**

**<div** class="col3"**>**Колонка 3**</div>**

**<div** class="col2"**>**Колонка 2**</div>**

**</div>**

Далее устанавливаем ширину всех колонок и их метод обтекания.

.col1 { width: 50%; float: left; }

.col2 { width: 200px; float: left; }

.col3 { width: 50%; float: right; }

Это ещё не всё, слои пока никаких колонок не формируют и выстраиваются совершенно не подходящим для нас способом. Требуется сместить вторую колонку влево на половину её ширины (margin-left: -100px) и заставить третью колонку встать на своё место. Для этого надо увеличить её ширину так, чтобы она была равна или больше, чем 50%+100px (половина второй колонки). Лучше всего подойдёт свойство margin-left с отрицательным значением, после этого колонки будут созданы. Есть ещё некоторые нюансы. Крайние колонки состыкованы между собой, что хорошо заметно, когда их высота превышает высоту средней колонки (рис. 5.21).

Состыкованные колонки

*Рис. 5.21. Состыкованные колонки*

К тому же в левой колонке текст справа, а в правой колонке текст слева скрывается под фоном центральной колонки. Это будет не заметно, если высота этой колонки большая, но это не всегда возможно. Чтобы преодолеть указанные недостатки, необходимо вложить внутрь крайних колонок ещё по одному слою (в примере он называется wrap) и уже для них задать цвет фона колонки, необходимые поля и отступы. Окончательный код приведён в примере 5.29.

**Пример 5.29. Макет № 7**

XHTML 1.0CSS 2.1IECrOpSaFx

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

**<html** xmlns="http://www.w3.org/1999/xhtml"**>**

**<head>**

**<meta** http-equiv="Content-Type" content="text/html; charset=utf-8" **/>**

**<title>**Макет 7**</title>**

**<style** type="text/css"**>**

.header, .footer { background: #D5BAE4; }

.layout { overflow: hidden; }

.col1 { width: 50%; float: left; }

.col1 .wrap {

background: #C7E3E4;

margin-right: 100px; /\* Сдвигаем влево на половину ширины колонки 2 \*/

padding: 10px; /\* Поля вокруг текста \*/

}

.col2 { background: #E0D2C7; width: 200px; float: left;

margin-left: -100px; }

.col3 { width: 50%; float: right; margin-left: -100px; }

.col3 .wrap {

background: #ECD5DE;

margin-left: 100px; /\* Сдвигаем вправо на половину ширины колонки 2 \*/

padding: 10px;

}

**</style>**

**</head>**

**<body>**

**<div** class="header"**>**Шапка сайта**</div>**

**<div** class="layout"**>**

**<div** class="col1"**><div** class="wrap"**>**Колонка 1**</div></div>**

**<div** class="col3"**><div** class="wrap"**>**Колонка 3**</div></div>**

**<div** class="col2"**>**Колонка 2**</div>**

**</div>**

**<div** class="footer"**>**Подвал**</div>**

**</body>**

**</html>**

**Использование позиционирования**

Для управления положением слоёв относительно родительского элемента необходимо для родителя установить свойство position со значением relative, а дочерним элементам, которые формируют колонки, значение absolute. Структура кода для первых четырёх макетов будет одинаковой и представлена в примере 5.13.

**Пример 5.13. Две колонки в пикселах или три в процентах**

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

**<html** xmlns="http://www.w3.org/1999/xhtml"**>**

**<head>**

**<meta** http-equiv="Content-Type" content="text/html; charset=utf-8" **/>**

**<title>**Три колонки**</title>**

**</head>**

**<body>**

**<div** class="header"**>**Шапка сайта**</div>**

**<div** class="layout"**>**

**<div** class="col1"**>**Колонка 1**</div>**

**<div** class="col2"**>**Колонка 2**</div>**

**<div** class="col3"**>**Колонка 3**</div>**

**</div>**

**</body>**

**</html>**

Здесь колонки пронумерованы по порядку, т.е. колонка 1 располагается слева, колонка 2 по центру, а колонка 3 справа.

У позиционирования есть определённый недостаток, который ограничивает применение этого метода — при добавлении подвала он будет скрыт под колонками. Это связано с тем, что у слоя layout нет высоты как таковой, поэтому он не «толкает» слой с подвалом вниз. Если подвал непременно требуется на странице, следует воспользоваться методом построения колонок, основанным на плавающих элементах. Ещё этот метод не работает в IE6 из-за наличия в нём ошибок.

Для макета № 1, в котором ширина первой колонки резиновая, а остальных фиксирована, стиль будет следующим (пример 5.14).

**Пример 5.14. Макет № 1**

.header { background: #D5BAE4; }

.layout { position: relative; /\* Относительное позиционирование \*/ }

.layout **DIV** { position: absolute; /\* Абсолютное позиционирование \*/ }

.col1 {

background: #C7E3E4; /\* Цвет фона \*/

left: 0; /\* Положение левого края \*/

right: 300px; /\* Положение правого края \*/

}

.col2 {

background: #E0D2C7;

width: 200px; /\* Ширина колонки \*/

right: 100px; /\* Сдвигаем влево на ширину колонки 3 \*/

}

.col3 { background: #ECD5DE; width: 100px; right: 0; }

У колонок с заданной шириной стоит свойство width, а их положение слева или справа задаётся соответственно свойством left или right. Резиновая ширина оставшейся колонки строится после одновременного добавления left и right, значения которых совпадают с шириной фиксированных колонок.

Макеты № 2 (пример 5.15) и № 3 (пример 5.16) построены на том же принципе.

**Пример 5.15. Макет № 2**

.header { background: #D5BAE4; }

.layout { position: relative; }

.layout **DIV** { position: absolute; }

.col1 { background: #C7E3E4; width: 100px; }

.col2 { background: #E0D2C7; left: 100px; right: 200px;}

.col3 { background: #ECD5DE; width: 200px; right: 0; }

**Пример 5.16. Макет № 3**

.header { background: #D5BAE4; }

.layout { position: relative;}

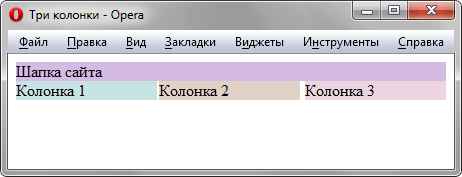
.layout **DIV** { position: absolute; }

.col1 { background: #C7E3E4; width: 100px; }

.col2 { background: #E0D2C7; width: 200px; left: 100px; }

.col3 { background: #ECD5DE; left: 300px; right: 0; }

Макет № 4, в котором ширина всех колонок задана в процентах имеет некоторые нюансы. Если требуется одинаковая ширина всех колонок, её можно задать дробно (33.33%), но браузер Opera не умеет работать с дробными значениями процентов, поэтому в нём между колонками появятся «дыры» (рис. 5.18).



*Рис. 5.18. Ширина колонок, заданная в дробных процентах*

В подобных ситуациях следует перейти на неравные доли, к примеру, 33%, 34%, 33%, как показано в примере 5.17.

**Пример 5.17. Макет № 4**

.header { background: #D5BAE4; }

.layout { position: relative;}

.layout **DIV** { position: absolute; }

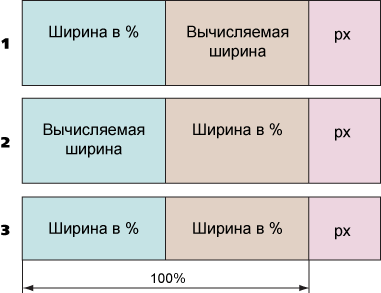
.col1 { background: #C7E3E4; width: 33%; }

.col2 { background: #E0D2C7; left: 33%; width: 34%; }

.col3 { background: #ECD5DE; right: 0; width: 33%; }

В некоторых браузерах возможно появление небольшого промежутка между колонками. Решается использованием не целых, а дробных значений процентов, т.е. 33.3% вместо 33%.

Оставшиеся макеты, в которых две колонки из трёх резиновые, представляют особую группу, потому что их можно трактовать по-разному. Так, ширина одной колонки указывается в пикселах, другой в процентах от ширины макета, а ширина оставшейся вычисляется автоматически.  На рис. 5.19 показаны разные подходы к вычислению ширины колонок на примере макета № 5.



*Рис. 5.19. Ширина двух резиновых колонок*

В первом варианте ширина первой колонки задана в процентах от ширины макета, ширина второй колонки вычисляется автоматически, а третья колонка имеет фиксированную ширину в пикселах. Во втором варианте колонки меняются между собой, и автоматически вычисляется ширина первой колонки. Третий вариант предполагает, что общая ширина резиновых колонок принимается за 100% и ширина первой и второй колонки вычисляется от неё.

Первый и второй вариант легко реализуется аналогично коду с двумя колонками в пикселах. Только вместо ширины в px указываем %. В примере 5.18 приведён стиль макета № 5 с первой колонкой заданной в процентах.

**Пример 5.18. Макет № 5. Ширина второй колонки вычисляемая**

.header { background: #D5BAE4; }

.layout { position: relative;}

.layout **DIV** { position: absolute; }

.col1 { background: #C7E3E4; width: 50%; }

.col2 { background: #E0D2C7; left: 50%; right: 200px; }

.col3 { background: #ECD5DE; right: 0; width: 200px; }

Стиль макета № 5 с вычисляемой первой колонкой показан в примере 5.19.

**Пример 5.19. Макет № 5. Ширина первой колонки вычисляемая**

.header { background: #D5BAE4; }

.layout { position: relative;}

.layout **DIV** { position: absolute; }

.col1 { background: #C7E3E4; left: 0; right: 200px; margin-right: 50%; }

.col2 { background: #E0D2C7; width: 50%; right: 200px; }

.col3 { background: #ECD5DE; width: 200px; right: 0; }

Первой колонке нельзя задать ограничение справа через свойство right, поскольку значение будет равно 50%+200px, в CSS2 таких вычисляемых значений нет. Поэтому применим фокус — ограничим первый слой справа шириной 200px через right и сдвинем его влево на ширину второй колонки 50% с помощью margin-right. Слой у нас абсолютно позиционированный, поэтому такие махинации никак на ширине не скажутся.

Третий вариант с двумя резиновыми колонками требует наличия дополнительного слоя, назовём его rubber, относительно которого будет задаваться ширина колонок (пример 5.20).

**Пример 5.20. Макет № 5. Ширина двух колонок в процентах**

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

**<html** xmlns="http://www.w3.org/1999/xhtml"**>**

**<head>**

**<meta** http-equiv="Content-Type" content="text/html; charset=utf-8" **/>**

**<title>**Три колонки**</title>**

**<style** type="text/css"**>**

.header { background: #D5BAE4; }

.layout { position: relative;}

.layout **DIV** { position: absolute; }

.rubber { left: 0; right: 200px; }

.col1 { background: #C7E3E4; width: 60%; }

.col2 { background: #E0D2C7; width: 40%; left: 60%; }

.col3 { background: #ECD5DE; width: 200px; right: 0; }

**</style>**

**</head>**

**<body>**

**<div** class="header"**>**Шапка сайта**</div>**

**<div** class="layout"**>**

**<div** class="rubber"**>**

**<div** class="col1"**>**Колонка 1**</div>**

**<div** class="col2"**>**Колонка 2**</div>**

**</div>**

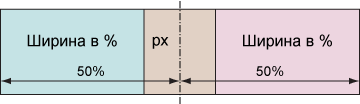
**<div** class="col3"**>**Колонка 3**</div>**

**</div>**

**</body>**

**</html>**

Построение макета № 6 аналогично макету № 5, поэтому останавливаться на нём не будем. Что касается макета № 7, то для него существует вариант, который может вызвать трудность. Это касается того случая, когда требуется сделать ширину левой и правой колонки вычисляемыми и равными между собой. Для этого ширину первой и третьей колонки следует сделать равной 50% (рис. 5.20).



*Рис. 5.20. Макет № 7 с равными по ширине колонками*

Чтобы осталось место под вторую колонку, воспользуемся свойством margin-right для первой колонки и margin-left для третьей. Значением этих свойств будет половина ширины второй колонки. Так, если она равна 200px, то получится margin-right: 100px (пример 5.21).

**Пример 5.21. Макет № 7. Ширина резиновых колонок одинакова**

.header { background: #D5BAE4; }

.layout { position: relative;}

.layout **DIV** { position: absolute; }

.col1 { background: #C7E3E4; left: 0; right: 50%; margin-right: 100px; }

.col2 { background: #E0D2C7; width: 200px; left: 50%; margin-left: -100px; }

.col3 { background: #ECD5DE; left: 50%; right: 0; margin-left: 100px; }

Напрямую свойством width пользоваться нельзя, поскольку добавление margin только увеличит ширину, а не уменьшит, как нам требуется. Поэтому ширина формируется одновременно заданными свойствами left и right. С позиционированием второй колонки возникают сложности в указании значения left или right, поскольку оно будет равно 50%-200px. Так что устанавливаем положение левого края на 50% (left: 50%), а затем колонку целиком сдвигаем влево на половину её ширины через свойство margin-left (margin-left: -100px).

Поскольку ширина некоторых колонок вычисляется автоматически, рекомендуется сделать ограничение по минимальной ширине макета, добавив свойство min-width для селектора **BODY**. Тогда ширина колонок не станет уменьшаться при достижении заданного предела.

**BODY** {

min-width: 800px; /\* Минимальная ширина макета \*/

}

Значение min-width зависит от макета и содержимого страницы и обычно подбирается опытным путём.

**Поля и границы в трёхколоночном макете**

В приведённые примеры макетов намеренно не добавлялось свойство padding, поскольку код служит лишь основой макета, и вы сами должны в конкретных случаях решать, нужны поля в колонках или нет. Добавление полей и границ к блочному элементу увеличивает его ширину, что надо учитывать при вёрстке. Однако ширина вырастает, только если для слоя установлено значение width. Рассмотрим несколько примеров, где ширина будет меняться, а где нет.

**div** { /\* padding влияет на ширину \*/

width: 200px;

padding: 10px;

}

**div** { /\* padding не влияет на ширину \*/

position: absolute;

left: 20px; right: 20px;

padding: 10px;

}

**div** { /\* padding влияет на ширину \*/

float: left;

width: 50%;

padding: 10px;

}

**div** { /\* padding не влияет на ширину \*/

margin-right: 50%;

padding: 10px;

}

В тех колонках, где padding или border требуется, но их добавление приведёт к «ломке» макета, можно воспользоваться вложенным слоем и установить необходимые свойства для него.

**<div** style="width: 200px"**>**

**<div** style="padding: 10px; border: 1px solid #000"**>**

Колонка

**</div>**

**</div>**

В этом примере ширина слоя заданная как 200 пикселов меняться не будет, но поля и рамка будут добавлены.

**Таблица в качестве колонок**

Таблицу удобно использовать для простого и быстрого создания колонок одинаковой высоты. Такая быстрота достигается несколькими вещами. Во-первых, структура кода для любого макета остаётся одинаковой, колонки в коде, как в случае с float, свой порядок никогда не меняют. Во-вторых, ширина ячеек вычисляется автоматически исходя из их содержимого, поэтому достаточно указать ширину нужных колонок, а оставшиеся подстроятся под общую ширину таблицы. В примере 5.30 показан макет № 1 сделанный на таблице.

**Пример 5.30. Макет № 1**

XHTML 1.0CSS 2.1IECrOpSaFx

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

**<html** xmlns="http://www.w3.org/1999/xhtml"**>**

**<head>**

**<meta** http-equiv="Content-Type" content="text/html; charset=utf-8" **/>**

**<title>**Макет 1**</title>**

**<style** type="text/css"**>**

.header, .footer { background: #D5BAE4; }

.layout {

width: 100%; /\* Ширина таблицы \*/

}

.layout **TD** {

padding: 5px; /\* Поля в ячейках \*/

vertical-align: top; /\* Выравнивание по верхнему краю \*/

}

.col1 { background: #C7E3E4; }

.col2 { background: #E0D2C7; width: 200px; }

.col3 { background: #ECD5DE; width: 100px; }

**</style>**

**</head>**

**<body>**

**<div** class="header"**>**Шапка сайта**</div>**

**<table** class="layout" cellspacing="0"**>**

**<tr>**

**<td** class="col1"**>**Колонка 1**</td>**

**<td** class="col2"**>**Колонка 2**</td>**

**<td** class="col3"**>**Колонка 3**</td>**

**</tr>**

**</table>**

**<div** class="footer"**>**Подвал**</div>**

**</body>**

**</html>**

Добавление свойства padding к селектору **TD** отменяет действие атрибута cellpadding тега **<table>** и добавляет поля к содержимому ячеек. При этом сама ширина колонок никак не изменится.

Для остальных макетов далее показан только стиль, HTML-код останется таким же, как в примере 5.30.

**Макет № 2**

.header, .footer { background: #D5BAE4; }

.layout { width: 100%;}

.layout **TD** { padding: 5px; vertical-align: top; }

.col1 { background: #C7E3E4; width: 100px; }

.col2 { background: #E0D2C7; }

.col3 { background: #ECD5DE; width: 200px; }

**Макет № 3**

.header, .footer { background: #D5BAE4; }

.layout { width: 100%;}

.layout **TD** { padding: 5px; vertical-align: top; }

.col1 { background: #C7E3E4; width: 100px; }

.col2 { background: #E0D2C7; width: 200px; }

.col3 { background: #ECD5DE; }

**Макет № 4**

.header, .footer { background: #D5BAE4; }

.layout { width: 100%;}

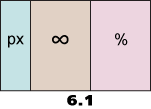
.layout **TD** { padding: 5px; vertical-align: top; }

.col1 { background: #C7E3E4; width: 33%; }

.col2 { background: #E0D2C7; width: 34%; }

.col3 { background: #ECD5DE; width: 33%; }

В макете 6.1 ширина первой колонки установлена в пикселах, третьей в процентах, а средняя колонка занимает оставшееся место.



**Макет № 6.1**

.header, .footer { background: #D5BAE4; }

.layout { width: 100%;}

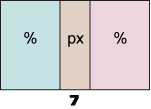
.layout **TD** { padding: 5px; vertical-align: top; }

.col1 { background: #C7E3E4; width: 200px; }

.col2 { background: #E0D2C7; }

.col3 { background: #ECD5DE; width: 40%; }

Остальные макеты с помощью таблицы строятся аналогично, меняется только значение свойства width. Для макета № 7, в котором ширина крайних колонок резиновая и равна между собой имеются некоторые хитрости, так что на нём остановимся подробнее.



Ширина для крайних колонок устанавливается равной 50%, а для средней колонки она задаётся в пикселах. Чтобы все размеры соблюдались к таблице необходимо добавить свойство table-layout со значением fixed.

.layout { width: 100%; table-layout: fixed; }

.col1 { width: 50%; }

.col2 { width: 200px; }

.col3 { width: 50%; }

Хотя суммарная ширина ячеек превышает ширину таблицы, никакого расширения и смещения таблицы не произойдет, как это случилось бы с блочными элементами. Как в таких случаях говорится, это не баг, это фича!

В примере 5.31 приведён полный код макета № 7.

**Пример 5.31. Макет № 7**

XHTML 1.0CSS 2.1IECrOpSaFx

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

**<html** xmlns="http://www.w3.org/1999/xhtml"**>**

**<head>**

**<meta** http-equiv="Content-Type" content="text/html; charset=utf-8" **/>**

**<title>**Макет 7**</title>**

**<style** type="text/css"**>**

**BODY** { margin: 0; background: #6D775B; }

.layout { width: 100%; table-layout: fixed; }

.layout **TD** { padding: 7px; vertical-align: top; }

.col1 { background: #6D775B; width: 50%; }

.col2 { background: #F5E8D0; width: 300px; }

.col3 { background: #6D775B; width: 50%; text-align: right; }

**</style>**

**</head>**

**<body>**

**<table** class="layout" cellspacing="0"**>**

**<tr>**

**<td** class="col1"**><img** src="images/tenniel1.png" alt="" **/></td>**

**<td** class="col2"**>**

**<img** src="images/jabberwock.jpg" alt="Бармаглот" **/>**

**<p>**Он встал под дерево и ждет,**<br />**

И вдруг граахнул гром.**<br />**

Летит ужасный бармаглот, **<br />**

И пылкает огнем. **</p>**

**<p>**Раз-два, раз-два! Горит трава,

Взы-взы — стрижает меч,**<br />**

Ува! Ува! И голова **<br />**

Барабардает с плеч.**</p>**

**<p>**Льюис Кэрролл, перевод Дины Орловой, рисунки Джона Тенниела**</p>**

**</td>**

**<td** class="col3"**><img** src="images/tenniel2.png" alt="" **/></td>**

**</tr>**

**</table>**

**</body>**

**</html>**