# **Основы CSS: блочная модель**

Один из лучших способов внести изменения во внешний вид своего сайта — похимичить с CSS, с помощью которого можно поменять всё: от цветов сайта до шрифтов и фоновых изображений.

В этой статье для начинающих мы разбёремся, что такое блочная модель и как она работает. Сначала мы пройдёмся по теоретической части, а затем применим её на практике.

Блоки — они повсюду!

Первое, что нужно понять для эффективного использования CSS,  — всё, что вы видите на странице, состоит из прямоугольников. Каждый сайт представляет собой ничего более, чем набор прямоугольных блоков, расположенных рядом, сверху, снизу и внутри друг друга.

Заголовок? Прямоугольник. Боковая панель? Прямоугольник. Изображение? Тоже прямоугольник. Само собой, обычно мы называем их не прямоугольниками, а HTML-элементами, которые бывают разных типов.

Строчные и блочные элементы

Есть два типа HTML-элементов: строчные и блочные. Между ними есть разительные отличия, которые также влияют на использование блочной модели CSS.

Одной из особенностей блочных элементов является то, что они занимают всю площадь контейнера, в котором находятся. Если вы не указали иное с помощью CSS, они растягиваются, чтобы занять максимум доступного места, сдвигая другие элементы под ними.

Этот цитатный блок — пример блочного элемента. Как вы видите по его фону, он занимает всю ширину контейнера, а не растягивается только до тех мест, где есть текст.

Кроме того, блочные элементы могут содержать другие блочные или строчные элементы и автоматически регулируют свою высоту, чтобы уместить своё содержимое. В их число входят заголовки, контейнеры, списки и другие элементы.

Если вы пользуетесь WordPress, то вы, скорее всего, очень хорошо знакомы со строчными элементами, даже если вы об этом не знаете. Когда вы при написании текста выделяете его жирным или курсивом, WordPress добавляет <strong>...</strong> и <em>...</em> вокруг него, чтобы добиться нужного эффекта.

В отличие от блочных, строчные элементы занимают по горизонтали столько места, сколько им необходимо. Также они не сдвигают на новую строку и под себя другие элементы.

Так или иначе, из-за такого поведения большая часть блочной модели CSS применима только к блочным элементам. Скоро вы поймёте, почему.

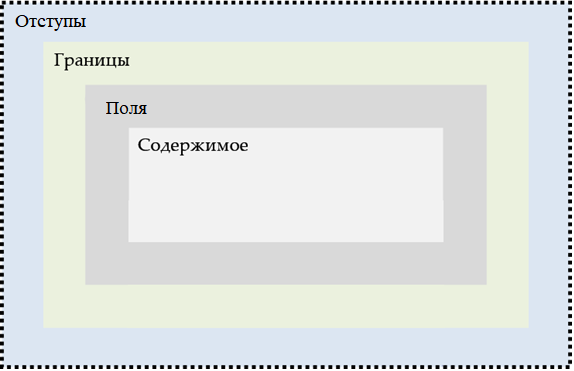
Так что это за блочная модель?

Возможно, вы об этом не знаете, но у каждого HTML-элемента есть блок, который его окружает. Этот блок состоит из нескольких слоёв, которыми можно независимо управлять с помощью CSS. Это позволяет разместить элементы относительно друг друга и оформить их разными способами.

Вот из чего состоят слои:

* Ширина — ширина площади содержимого элемента. Для блочных элементов значение по умолчанию равно 100%. У строчных элементов ширина зависит от содержимого.
* Высота — определяет высоту элемента. Как правило, она зависит от внутреннего содержимого, но при желании можно указать конкретную высоту. Опять же, это работает только с блочными элементами.
* Границы — границы есть у каждого элемента, даже если вы их не видите. У них может быть разный размер, цвет и оформление.
* Поля — они определяют расстояние между границей элемента и его содержимым. Их можно использовать, например, для того, чтобы текст внутри элемента оставался читаемым.
* Отступы — они определяют расстояние между границей элемента и тем, что его окружает.

Вот вам, собственно, и блочная модель CSS. Она становится ещё понятнее, если взглянуть на эту диаграмму:



Так как у каждого из слоёв есть соответствующее CSS-свойство, блочная модель представляет основные способы определения размера, положения и внешнего вида HTML-элемента. Как следствие, вы получаете возможность сильно изменить содержимое страницы. Рассмотрим несколько примеров, чтобы понять, как это работает.

Примеры использования блочной модели

Создадим простую страницу с блочным элементом:

<!DOCTYPE html>

<html>

<head>

<title>Example Site for CSS Box Model</title>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

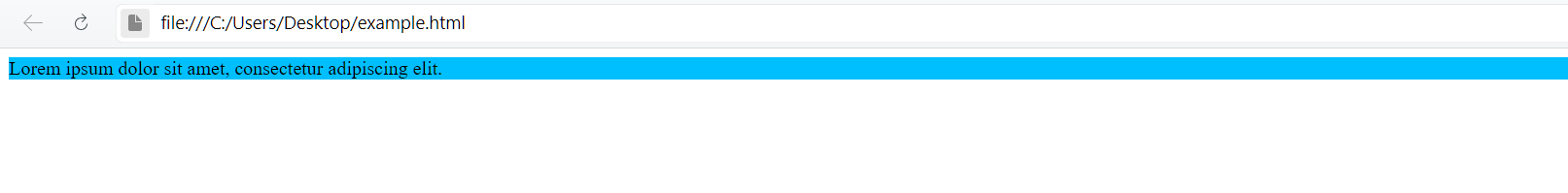
<body>

<div class="example-element">Lorem ipsum dolor sit amet, consectetur adipiscing elit.</div>

</body>

</html>

Это просто HTML-документ с текстом внутри контейнера div. В браузере он выглядит так:



Для блока мы определили цвета фона, чтобы вы могли видеть, как он растягивается по экрану. Это позволит лучше понять то, что будет происходить дальше. Вот соответствующий CSS:

.example-element {

background-color: deepskyblue;

}

Давайте внесём несколько изменений.

Изменяем ширину

Первое, что мы сделаем, — определим ширину. Как было упомянуто ранее, сейчас её значение равно 100%. Чтобы изменить его, нам нужно взаимодействовать со свойством width.

Для него можно установить разные значения. Мы можем использовать фиксированную ширину вроде пикселей или пропорциональное значение вроде процентов. Последний вариант особенно важен для штук вроде адаптивного дизайна. Однако для простоты обойдёмся пикселями.

Добавляем новое свойство в стили:

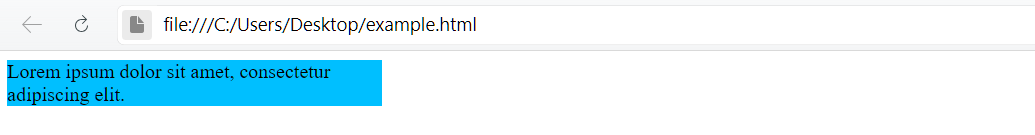
.example-element {

background-color: deepskyblue;

width: 300px;

}

После перезагрузки страницы вы должны увидеть следующее:



Вы сразу заметите, как теперь ограничено растягивание div вправо. Также, как видите, его высота автоматически изменяется, чтобы уместить внутренний текст. Давайте ещё увеличим её.

Увеличиваем высоту элемента через CSS

Хотя содержимое вполне себе умещается в пределах прямоугольника, порой есть веские причины ещё увеличить высоту. Например, что, если мы захотим, чтобы элемент был квадратом? Это вполне достижимо с помощью следующего свойства в нашем списке — height.

С высотой мы можем делать всё то же самое, что и с шириной: устанавливать фиксированное или пропорциональное значение. Добавляем в стили:

.example-element {

background-color: deepskyblue;

height: 300px;

width: 300px;

}

В итоге страница выглядит так:



Как видите, элемент принял нужную нам форму, так как мы установили одинаковые значения для высоты и ширины.

Знакомимся с границами

Теперь изменим границы. Как упоминалось, границы уже есть, просто мы их не видим. Изменим это с помощью свойства border.

Важно знать, что это свойство принимает три значения: width, style и color. Width отвечает за толщину границы (обычно в пикселях), style может принимать значения solid, dahsed, dotted и т.д., а в color можно прописать цвет как словом вроде «red», так и hex-значением цвета.

В принципе, достаточно просто определить толщину границ. Тем не менее, в большинстве случаев результат будет лучше при определении всех трёх значений. Меняем CSS:

.example-element {

background-color: deepskyblue;

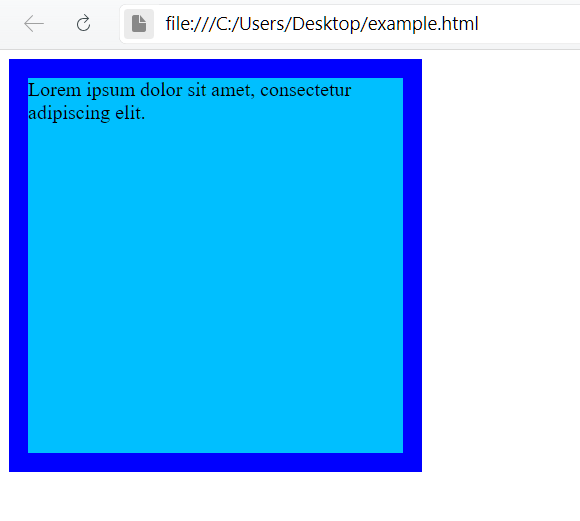
border: 15px solid blue;

height: 300px;

width: 300px;

}

И получаем результат:



Выглядит здорово, не так ли 🙂

Также обратите внимание, что для разных сторон можно установить разные типы границ с помощью свойств border-top, border-right, border-bottom и border-left.

Добавляем Поля

Одна из проблем, которая выделяется при взгляде на наш пример, заключается в том, что текст начинается прямо с границы элемента. Из-за этого его сложно читать, да и выглядит не очень приятно. К счастью, свойство padding позволяет это исправить:

.example-element {

background-color: deepskyblue;

border: 15px solid blue;

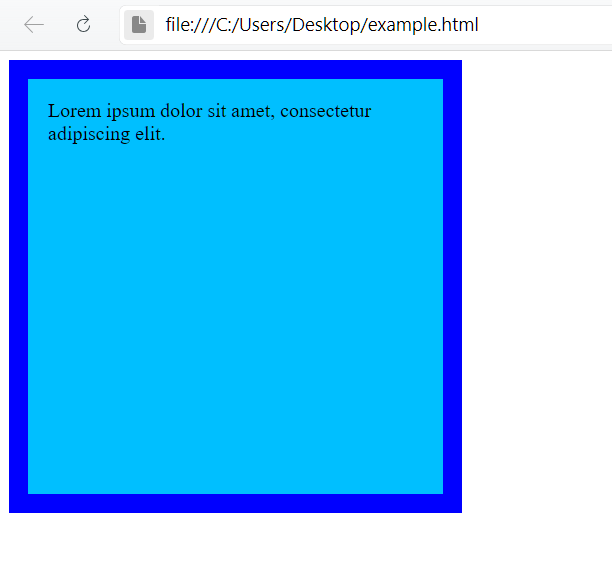
height: 300px;

padding: 16px;

width: 300px;

}

Вот как это отразится на результате:



Так же, как и с границами, можно настроить поля отдельно для каждой стороны элемента. Для этого используются свойства padding-top, padding-right, padding-bottom и padding-left.

Можно не прописывать отдельно все эти свойства, а описать все поля одной строкой вроде padding: 10px 5px 15px 10px;. Здесь значения соответствуют полям сверху, справа, снизу и слева соответственно.

Поля — одно из немногих свойств в этом списке, которые можно применить к строчным элементам. Однако вы рискуете вторгнуться в пределы верхних и нижних полей других элементов, так что аккуратнее с этим.

Добавляем Отступы

Вот мы и добрались до отступов, которые определяют расстояние между элементами.

Добавим margin в стили:

.example-element {

background-color: deepskyblue;

border: 15px solid blue;

height: 300px;

margin: 25px;

padding: 16px;

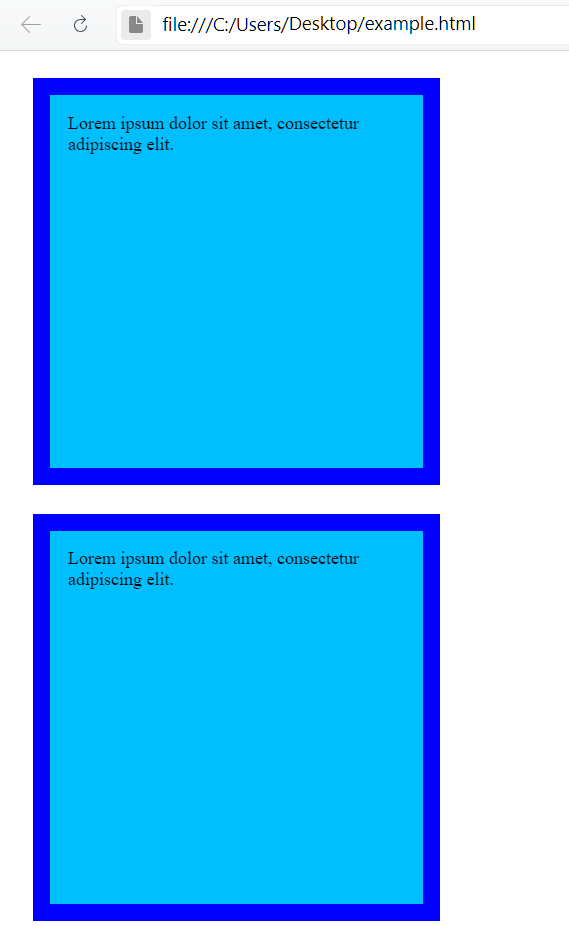
width: 300px;

}

Выглядит это так:



Как видите, в результате элементы отодвинулись от краёв страницы. Однако отступы для элемента определены со всех сторон, а не только сбоку и сверху. Скопируем div, чтобы это можно было увидеть:



Сейчас у нас одинаковые отступы со всех сторон. Однако для разных сторон можно установить свои значения. Это работает так же, как и с полями: margin-top, margin-left и т. д или однострочное сокращение.

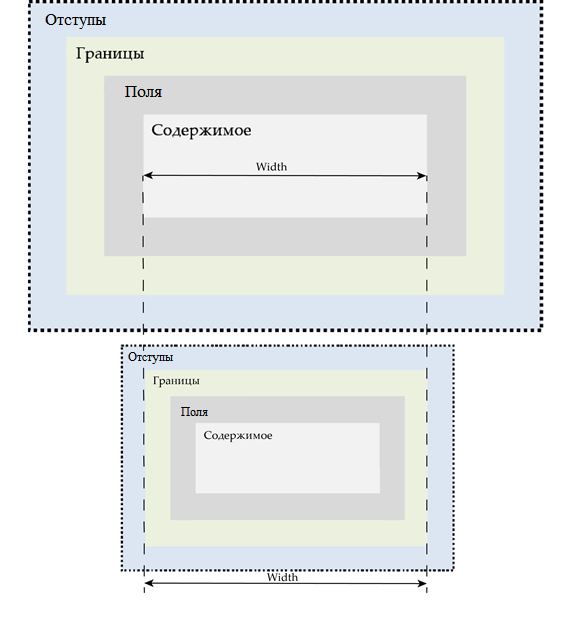
Отступы частично применимы к строчным элементам, но только к боковым сторонам.

Немного о размерах элементов

Когда речь заходит о размерах элементов, стоит принять во внимание, что учитываются все части блочной модели. Например, если вы установили ширину равной 200 px, это значение считается только для области с содержимым.

Любые границы, отступы и поля также вносят свой вклад в горизонтальный размер элемента. То же касается и высоты. В нашем случае также учитываются все слои блочной модели.

Есть способы обхода такого поведения вроде установки свойства элемента box-sizing равным border-box. В этом случае ширина будет включать в себя границы и всё содержимое подстроит под свои размеры, чтобы уместиться в этих рамках.



Тем не менее, отступы всё ещё влияют на общий размер элемента. Держите это в голове, когда будете работать над разметкой страницы.

Подводим итог

Вот вы и изучили основы блочной модели CSS. Эти знания помогут вам лучше понять, как работают сайты. С блочной моделью проще изменить внешний вид страницы и её структуру как душе угодно.

Также это отличная отправная точка для дальнейшего изучения CSS. Как только вы в совершенстве овладеете этим материалом, вам будет гораздо проще изучать новые способы изменения внешнего вида сайта. А там уже и до своих тем рукой подать.