## Блочная модель документа

|  |
| --- |
| [Блочные элементы](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/1) |
| [Строчные элементы](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/2) |
| [Ширина и высота](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/3) |
| [Внутренние отступы, свойство padding](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/4) |
| [Внешние отступы, свойство margin](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/5) |
| [Рамки](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/6) |
| [Стандартная блочная модель](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/7) |
| [Первое испытание](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/8) |
| [«Схлопывание» внешних отступов](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/9) |
| [«Выпадание» внешних отступов](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/10) |
| [Как отцентровать элемент?](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/11) |
| [Блочная модель и строчные элементы](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/12) |
| [Ширина 100% и ширина по умолчанию](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/13) |
| [Проблемы обычной блочной модели](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/14) |
| [Изменяем блочную модель, свойство box-sizing](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/15) |
| [Второе испытание](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/16) |
| [Управление типом элемента, свойство display](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/17) |
| [display: inline-block](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/18) |
| [display: table](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/19) |
| [display: table-row](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/20) |
| [display: table-cell](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/21) |
| [display: none](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/22) |
| [Последнее испытание](https://npoed.htmlacademy.ru/htmlcss110/course/11/run/23) |

# Блочные элементы [1/23]

Элементы HTML страницы обычно делятся на блочные и строчные.

Блочные элементы можно представлять как прямоугольные области на странице. Они имеют следующие особенности:

1. До и после блочного элемента существует перенос строки.
2. Блочным элементам можно задавать ширину, высоту, внутренние и внешние отступы.
3. Занимают всё доступное пространство по горизонтали.

К блочным элементам относятся такие теги как: <p>, <h1>, <h2>, <ul> и так далее.

Еще один важный блочный тег — это тег <div>, который обозначает просто «блок» или «прямоугольный контейнер». Этот тег чаще всего используется для создания сеток.

Существуют и более сложные типы элементов, им будет посвящена вторая часть раздела.

# Строчные элементы [2/23]

Строчные элементы располагаются друг за другом в одной строке, при необходимости строка переносится. Особенности строчных элементов:

1. До и после строчного элемента отсутствуют переносы строки.
2. Ширина и высота строчного элемента зависит только от его содержания, задать размеры с помощью CSS нельзя.
3. Можно задавать только горизонтальные отступы.

К строчным элементам относятся такие теги как: <a>, <strong>, <em>, <span> и так далее.

Строчные элементы предназначены для оформления текста на уровне небольших фраз и отдельных слов. Блочные же элементы предназначены для разметки крупных блоков текста (заголовки, абзацы, списки) и создания сетки.

# Ширина и высота [3/23]

Ширина и высота элементов задаются с помощью свойств width и height соответственно.

По умолчанию блочные элементы занимают всю доступную ширину, которая равна ширине родительского контейнера или окна браузера.

Высота по умолчанию блочных элементов зависит от их содержимого. Если задать блочному элементу ширину и высоту так, что содержимое элемента не будет в него помещаться, то оно как бы «выпадет» из него.

Строчные элементы не реагируют на задание ширины и высоты в CSS.

Задать ширину блоку можно, например, так:

.selector {

width: 100px;

height: 100px;

}

Вернуть значения по умолчанию можно с помощью специального значения auto:

.selector {

width: auto;

height: auto;

}

# Внутренние отступы, свойство padding [4/23]

Свойство padding задает внутренние отступы элемента — отступы от внешней границы элемента до его содержания. Эти отступы еще иногда называют полями.

Существует несколько способов записи свойства padding.

padding: 10px;

padding: 5px 10px;

padding: 5px 10px 15px;

padding: 5px 10px 15px 20px;

1. Одинаковые отступы со всех сторон.
2. Отступы сверху и снизу 5px, справа и слева 10px.
3. Отступ сверху 5px, слева и справа 10px, снизу 15px.
4. Разные отступы со всех сторон, в порядке верхний, правый, нижний, левый.

Также можно задавать отступы для разных сторон с помощью свойств padding-left, padding-right, padding-top, padding-bottom.

Для строчных элементов лучше не задавать вертикальных отступов, т.к. они ведут себя непредсказуемо.

В задании внутренние отступы блоков сделаны прозрачными.

# Внешние отступы, свойство margin [5/23]

Свойство margin задает внешние отступы элемента — отступы от внешней границы элемента до границ родительского элемента или до соседних элементов.

Способы записи свойства margin аналогичны свойству padding.

margin: 10px;

margin: 5px 10px;

margin: 5px 10px 15px;

margin: 5px 10px 15px 20px;

1. Одинаковые отступы со всех сторон.
2. Сверху и снизу 5px, справа и слева 10px.
3. Сверху 5px, слева и справа 10px, снизу 15px.
4. Верхний, правый, нижний, левый отступы соответственно.

Можно задавать отступы для разных сторон с помощью свойств margin-left, margin-right, margin-top, margin-bottom.

Строчные элементы реагируют только на горизонтальные отступы.

# Рамки [6/23]

Рамка задаётся с помощью свойства border, которое состоит из трёх компонентов:

1. ширина рамки;
2. стиль рамки;
3. цвет.

Например:

selector{

border: 5px solid red;

}

Это правило задаёт красную сплошную рамку толщиной 5px.

Задавать рамку можно одним свойством border, а можно и с помощью отдельных свойств border-width, border-style, border-color. Например:

selector{

border-width: 5px;

border-style: solid;

border-color: red;

}

Это правило задаёт такую же рамку, как и в примере выше.

Несколько самых распространенных стилей рамок:

* solid — сплошная;
* dashed — пунктирная;
* dotted — точками.

Рамку можно задавать и строчным, и блочным элементам.

# Стандартная блочная модель [7/23]

Стандартная блочная модель отвечает на основной вопрос:

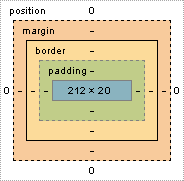
Сколько же в итоге места будет занимать элемент?

Ответ следующий:

Область, занимаемая блочным элементом, складывается из его ширины и высоты содержания, внутренних и внешних отступов, ширины рамок .

Для строчных элементов есть свои особенности, которые мы опишем позже.

Схема блочной модели:



* ширина/высота содержания — свойства width и height (синий прямоугольник на схеме);
* внутренние отступы — свойство padding;
* рамки — свойство border;
* внешние отступы — свойство margin.

# Первое испытание [8/23]

Используйте изученные свойства блочной модели для прохождения испытания, а именно:

* ширину и высоту;
* отступы;
* рамки.

Ничего кроме размеров блоков, отступов и рамок менять не надо.

Подобрать правильные размеры вам поможет сетка на фоне. Размеры маленькой ячейки 20 на 20 пикселей, большой ячейки 100 на 100 пикселей.

Внутренние отступы, в отличие от заданий, здесь непрозрачные.

# «Схлопывание» внешних отступов [9/23]

Во второй части раздела разберем некоторые важные тонкости блочной модели. И начнем с внешних отступов (или маргинов).

Вертикальный отступ между двумя соседними элементами равен максимальному отступу между ними. Если отступ одного элемента равен 20px, а второго 40px, то отступ между ними будет 40px.

Этот эффект называется эффектом «схлопывания» внешних отступов или «схлопывания» маргинов.

Горизонтальные отступы между элементами просто складываются. Например, горизонтальный отступ между двумя элементами с отступами 30px будет равен 60px.

# «Выпадание» внешних отступов [10/23]

«Выпадание» — это еще один эффект, связанный с вертикальными внешними отступами. Если внутри родительского блока расположить блок и задать ему отступ сверху, то внутренний блок прижмется к верхнему краю родительского, а у родительского элемента появится отступ сверху. Т.е. верхний отступ внутреннего элемента «выпадает» из родительского элемента.

Если у родительского элемента тоже был задан внешний отступ, то выберется максимальный отступ между собственным и «выпавшим» .

Чтобы избавиться от эффекта выпадания, можно задать родительскому элементу внутренний отступ (паддинг) сверху или добавить рамку сверху.

Внешние и внутренние отступы всегда складываются.

# Как отцентровать элемент? [11/23]

Чтобы отцентровать блочный элемент, нужно выполнить следующие действия:

1. Задать элементу ширину, которая меньше ширины родительского контейнера.
2. Задать для внешних отступов справа и слева значение auto.

Примеры кода:

selector {

width: 100px;

margin: 0 auto;

}

selector {

width: 100px;

margin-left: auto;

margin-right: auto;

}

# Блочная модель и строчные элементы [12/23]

Еще раз акцентируем внимание на особенности поведения строчных элементов в блочной модели:

1. Не реагируют на CSS-свойства width и height.
2. Частично реагируют на margin, воспринимая только горизонтальные отступы.
3. Частично реагируют на padding, воспринимая только горизонтальные отступы.
4. При задании вертикальных padding визуально увеличиваются, но без увеличения занимаемого места (не отталкивают другие элементы).
5. Воспринимают рамки. Аналогично padding рамки сверху и снизу не увеличивают занимаемое элементом место.

После выполнения задания обратите внимание на то, что расстояние между строк в абзацах не увеличилось, хотя элементы визуально стали выше.

# Ширина 100% и ширина по умолчанию [13/23]

CSS-свойство width задаёт не общую ширину блока, а только ширину содержания. Общая ширина блока затем складывается из трёх компонентов: ширины содержания, внутренних отступов и ширины рамок слева и справа.

Поведение элемента может зависеть от того, как именно вы зададите его ширину.

**Первый вариант**. Вариант по умолчанию, когда ширина не задается, соответствует значению width:auto;. В этом случае блок занимает всю ширину родительского блока. Если у блока есть внутренние отступы или рамки, то его ширина содержания автоматически уменьшается, а общая ширина остается равной ширине родителя.

**Второй вариант**. Когда ширина блока задана явно, например, width:100%;. В этом случае ширина содержания блока равна ширине родительского блока. Если блоку добавить внутренние отступы и рамки, то его общая ширина становится больше ширины родителя.

В этом задании вы увидите, как эти эффекты работают на практике. Обратите внимание, что ширины блоков будут рассчитываться автоматически.

# Проблемы обычной блочной модели [14/23]

В чём проблема описанного механизма расчета ширины?. Ведь можно просто не задавать ширину, используя auto, либо просто подбирать размеры блоков и отступов как на макете.

Проблемы начинаются, когда сетка резиновая и элементы в ней должны тянуться. Самый простой пример: форма, в которой поля должны занимать всю ширину контейнера, но при этом иметь фиксированные внутренние отступы, чтобы текст не прилипал к краям.

Для такой формы можно задать такие стили:

input[type="text"] {

width: 100%;

padding: 0 10px;

}

Ширина 100% заставит поля быть такой же ширины, как родительский блок, а паддинг добавит отступы. Но, как вы уже догадались, за счет отступов поле станет шире своего контейнера. Попрактикуемся.

Ширина полей ввода определяется значением атрибута size. Ширина width: auto для полей ввода рассчитывается из значения size по умолчанию и не растягивает поля на всю ширину контейнера.

# Изменяем блочную модель, свойство box-sizing [15/23]

Проблему с шириной тянущихся полей решить сложно. Идеальным вариантом было бы изменение алгоритма расчета ширины элемента, чтобы свойство width задавало не ширину содержания, а общую ширину.

К счастью, такая возможность была добавлена в CSS3 с помощью свойства box-sizing, которое уже поддерживается большинством современных браузеров.

Это свойство имеет два значения:

1. content-box — значение по умолчанию, соответствует стандартной блочной модели.
2. border-box — изменяет режим расчета ширины элемента на описанный выше.

Данное свойство лучше задавать с использованием префиксов, т.к. оно достаточно новое:

-webkit-box-sizing: border-box;

-moz-box-sizing: border-box;

box-sizing: border-box;

Полезная статья про box-sizing на русском: [Используйте свойство box-sizing](http://simonenko.su/32197993404/use-css3-box-sizing)

# Второе испытание [16/23]

Используйте полученные знания о тонкостях блочной модели, чтобы пройти это испытание.

Обратите внимание, что HTML-код заблокирован, а в нём некоторые CSS-свойства заданы в атрибуте style. Это обозначает, что вы не можете их переопределить и придется выкручиваться по-другому.

Если будете использовать box-sizing, то задавайте его, как в предыдущем задании: сначала свойства с префиксами, затем свойство без префикса.

# Управление типом элемента, свойство display [17/23]

Тип элемента не является чем-то вечным и неизменным, его можно изменять с помощью CSS. За это отвечает свойство display.

С его помощью, например, можно сделать абзацы и заголовки строчными, а спаны и стронги блочными элементами.

У свойства display много значений. С двумя из них вы уже заочно познакомились. Свойство display:block обозначает блочный элемент, а display:inline — строчный.

В третьей части раздела мы познакомимся с некоторыми более продвинутыми значениями свойства display.

# display: inline-block [18/23]

Иногда возникает необходимость расположить в ряд несколько элементов с заданными размерами. Строчные элементы для этого не подходят, т.к. не воспринимают размеры. Блочные элементы тоже не подходят, т.к. до и после них существует перенос строки. Конечно, блочные элементы можно приспособить для такой задачи, используя дополнительные свойства (которые будут разбираться в разделе про сетки).

Но более простой способ — использовать блочно-строчные элементы. В HTML нет тегов, которые по умолчанию вели бы себя как блочно-строчные, но любой элемент можно переключить в данный режим, задав ему свойство display со значением inline-block.

Особенности блочно-строчных элементов:

* им можно задавать размеры, рамки и отступы, как и блочным элементам;
* их ширина по умолчанию зависит от содержания, а не растягивается на всю ширину контейнера;
* они не порождают принудительных переносов строк, поэтому могут располагаться на одной строке, пока помещаются в родительский контейнер;
* элементы в одной строке выравниваются вертикально подобно строчным элементам.

# display: table [19/23]

Дальнейшие несколько заданий будут посвящены табличным типам элементов.

Табличные сетки были очень популярны на заре веба. Действительно, у табличных сеток есть преимущества, которые очень сложно получить с помощью блоков. Например, столбцы одинаковой высоты.

Однако, семантически таблицы не предназначены для разметки сеток и постепенно от них отказались. Но память об их удобстве жила.

Позднее было разработано семейство значений свойства display, которое позволяет задать табличное поведение любым элементам.

Первое значение — display:table задает элементу тип таблица. Особенности табличных элементов:

1. можно задавать ширину, высоту, рамки, отступы;
2. по умолчанию ширина зависит от содержания;
3. переносы строки до и после элемента.

Табличные элементы похожи на блочные за исключением ширины по умолчанию.

# display: table-row [20/23]

Как вы догадались, мы строим таблицу на дивах с помощью новых значений свойства display.

Любая таблица содержит элементы строка таблицы, внутри которых должны содержаться элементы ячейка таблицы. Соответствие тегов таблицы значениям display:

<table> — display:table;

<tr> — display:table-row;

<td> — display:table-cell;

В этом задании вы создадите строки таблицы с помощью display:table-row;. Строка — необычный элемент. Она является контейнером для ячеек и практически не имеет собственного отображения. Для нее можно только задавать цвет фона.

# display: table-row [20/23]

Как вы догадались, мы строим таблицу на дивах с помощью новых значений свойства display.

Любая таблица содержит элементы строка таблицы, внутри которых должны содержаться элементы ячейка таблицы. Соответствие тегов таблицы значениям display:

<table> — display:table;

<tr> — display:table-row;

<td> — display:table-cell;

В этом задании вы создадите строки таблицы с помощью display:table-row;. Строка — необычный элемент. Она является контейнером для ячеек и практически не имеет собственного отображения. Для нее можно только задавать цвет фона.

# display: table-cell [21/23]

С помощью значения table-cell свойства display можно задать элементу тип ячейка таблицы. Конечно, лучше чтобы элементы-ячейки находились внутри элементов-строк, которые находятся внутри элементов-таблиц.

Вы можете просто задать элементу тип ячейка таблицы, не добавляя вокруг него дополнительных элементов-строк и таблиц. В этом случае браузер создаст дополнительные анонимные элементы строки и таблицы. Неудобство заключается в том, что вы не сможете ими управлять.

К таблицам, созданным с помощью CSS, можно применять те же свойства, что и к обычным таблицам. Например, задавать отступы между ячейками или режим схлопывания границ.

# display: none [22/23]

Значение none свойства display используется очень часто. С его помощью можно скрыть элемент, как будто его и не было. Скрытый элемент не отображается и не занимает места на странице.

Данное свойство применяется при создании выпадающих меню, динамических галерей, переключающихся вкладок и много где еще.

Есть еще одно CSS-свойство, которое используется для сокрытия элементов. Это свойство visibility со значением hidden. Оно «прячет» элемент — он становится невидимым, но занимает место на странице.

# Последнее испытание [23/23]

Чтобы пройти это испытание, примените знания о продвинутых типах элементов, полученные в третьей части раздела.

Изменять нужно только свойства, относящиеся к блочной модели, и одно свойство, связанное с отображением рамок таблицы.