HTML&CSS

*<!-- в HTML-коде комментарий окружён такими символами -->*

*<!-- <p>невидимый пользователю абзац</p> -->*

*/\* в CSS — такими \*/*

*/\* p {*

*color: white;*

*} \*/*

Ссылки

Если в элемент ссылки добавить атрибут target со значением \_blank (blank, «чистый»), то другая страница будет показана в новой вкладке

**<p><a href="https://www.noisli.com" target="\_blank">Плейлист</a> для беспечного занятия кодингом.</p>**

Несколько атрибутов надо писать через пробел в открывающем теге элемента. Их порядок значения не имеет:

<a href="http://info.cern.ch" target="\_blank">Сайт о первом веб-сайте</a>

*<!-- то же самое -->*

<a target="\_blank" href="http://info.cern.ch">Сайт о первом веб-сайте</a>

rel = “noopener noreferrer”

Изображения

Теперь найдите два отличия:

Скопировать кодHTML

*<!-- ссылка -->*

<a href=""></a>

*<!-- изображение -->*

<img src="">

Первое отличие — путь к картинке указывается атрибутом src

Второе – отсутствие у img закрывающего тега.

Alt – описание, width – размер (ширина).

**<img src="https://pictures.s3.yandex.net/frontend-developer/free-course/mountains.jpg" alt="картинка на обложке" width="230">**

В head можно задать favicon — иконку для вкладки браузера. Для этого используется тег <link>. Он содержит ссылку на внешний файл, указание типа и предназначения подключаемого файла.

<link rel="icon" href="fav.ico" type="image">

*<!-- rel — назначение (от англ. relation, отношение); здесь это иконка -->*

*<!-- type — тип файла. здесь это картинка -->*

*<!-- href — уже знакомый атрибут адреса -->*

Элемент link дает возможность подключить не только иконку, но и файлы других типов и другого назначения, например — таблицы стилей.

Добавьте вашему HTML-проекту иконку для вкладки браузера.

Вместе с тегом <link rel="icon" type="image"> укажите для иконки атрибут href, чтобы иконка загружалась с адреса <https://code.s3.yandex.net/web-code/playlist-favicon.ico>

**<link rel="icon" type="image" href="https://code.s3.yandex.net/web-code/playlist-favicon.ico">**

Чтобы придать элементу индивидуальный стиль, в тег встраивается атрибут style. Ему присваивается название свойства (что именно менять) и значение этого свойства (как это менять).

**<h1 style="color: springgreen; font-size: 72px;">Твин Пикс</h1>**

Отступы

Отступы можно задавать отдельно для верхней, правой, нижней или левой сторон. Тогда имя свойства обозначается не просто margin, а соответственно margin-top, margin-right, margin-bottom или margin-left.

Если написать margin: 20px, отступы в 20 пикселей появятся со всех четырёх сторон, а если margin-left: 20px — только слева.

Профессиональные разработчики убирают начальные настройки, чтобы контролировать отступы и размеры. В самом начале style.css создайте необычный селектор с перечислением: body, h1 { }. Внутри правила укажите свойство margin: 0.

Пора разобраться с самым верхним отступом. Почему мы ещё не написали margin-top для элемента h1?

Есть проблема: margin-top первого дочернего элемента будет отодвигать не элемент от верхней границы родителя, а самого родителя от вышестоящего элемента.

Иными словами, дело обстоит так:

Скопировать кодHTML

<div>

*<!-- margin-top для h1 сдвинет вниз весь div -->*

<h1></h1>

*<!-- margin-top для p отодвинет p от h1 -->*

<p></p>

</div>

Вместо того, чтобы делать отступ наружу, отступим от границы элемента h1 внутрь. Для элемента h1 обнулите margin-top и назначьте padding-top со значением 34px.

Селектору div пропишите margin: auto;. Это незнакомое заклинание поместит div по центру по горизонтали. Так как div — не текст, то text-align здесь не сработает.

Профессиональные разработчики убирают начальные настройки, чтобы контролировать отступы и размеры. В самом начале style.css создайте необычный селектор с перечислением: body, h1 { }. Внутри правила укажите свойство margin: 0.

# Размеры в процентах и долях

Если задать элементу width: 50%, то он посмотрит на ширину родительского элемента, разделит пополам и применит к себе. То же самое и с высотой.

Элементы h1–h6, p*,* div и header по умолчанию занимают 100% ширины своего родителя. Ширина элемента body по умолчанию равна видимой ширине окна браузера. Добавим обезьян 🙈🙉🙊 для привлечения особого внимания к этому факту.

**2. Сравнение с окном просмотра — в долях.**

Вы можете встретить сайты, на которых секции точь-в-точь соответствуют размерам браузера. Точнее, окна браузера, в котором вы просматриваете сайт.

По-английски окно просмотра в браузере называется viewport. Его ширина измеряется в единицах vw, а высота в единицах vh. Полная ширина окна равна 100vw, полная высота окна — 100vh.

Например, если установить элементу свойство height: 50vh, его высота станет равна половине высоты окна просмотра; 100vh — всей высоте, 40vh — 40% высоты.

Цвета в HTML

RGB-палитра описывает цвета комбинацией красного (R), зеленого (G) и синего (B).

/\* черный цвет \*/ rgb(0, 0, 0);

/\* белый цвет \*/ rgb(255, 255, 255);

/\* небесно-голубой цвет \*/ rgb(0, 150, 200);



div { */\* цвет фона \*/* background-color: maroon; */\* фоновое изображение \*/* background-image: url(https://yastatic.net/s3/auth2/\_/msoffice.d1ada09f.svg); }

HEX-colors — это другой стандарт. Принцип тот же, что в стандарте RGB, только числа от 0 до 255 заменяют на соответствующую им комбинацию двух символов из ряда 0 1 2 3 4 5 6 7 8 9 A B C D E F. По два символа на красный, зелёный, синий — достаточно шести символов.

Браузер узнает запись в стандарте HEX-colors по символу хеш — *#*. Так, жёлтый цвет вместо rgb(255, 255, 0) обозначается #FFFF00.

Размеры фонового изображения определяет background-size:

/\* cover (англ. «накрыть») сожмёт или растянет картинку \*/

/\* так, чтобы занимала всю поверхность \*/

/\* сохранит пропорции \*/

background-size: cover;

/\* contain (англ. «содержать») сделает так, чтобы вся картинка уместилась \*/

/\* сохранит пропорции \*/

background-size: contain;

/\* можно указать размер в пикселях \*/

/\* первое значение — ширина, второе — высота \*/

background-size: 220px 400px;

/\* так отменяется повтор изображения \*/

background-repeat: no-repeat;

/\* изображение повторяется только по вертикали \*/

background-repeat: repeat-y;

Положением фона управляет background-position:

/\* по центру — сразу по горизонтали и по вертикали \*/

background-position: center;

/\* по горизонтали к правому краю, по вертикали — к нижнему \*/

background-position: right bottom;

/\* можно указать сдвиг в пикселях \*/

/\* по горизонтали сдвиг на 150px вправо от левого верхнего угла блока \*/

/\* по вертикали — на 50px вверх \*/

background-position: 150px -50px;

Прозрачность

/\* rgb без альфа-канала, просто цвет \*/

rgb(115, 170, 200);

/\* rgb c альфа-каналом, цвет + прозрачность 50% \*/

/\* вместо rgb пишут rgba \*/

rgba(115, 170, 200, 0.5);

Десятичная дробь в CSS и многих других языках пишется с точкой — не с запятой, как в школьной тетради. Также рекомендуем писать дробь без нуля.

/\* один и тот же цвет \*/

rgba(115, 170, 200, 0.5);

rgba(115, 170, 200, .5);

**Коробка в коробке**

Созданный div станет прозрачной прослойкой между картинкой и заголовком. Для этого перенесите h1 внутрь <div></div>.

<div> <h1>Четыре правила вёрстки</h1></div>

div {

width: 250px;

height: 250px;

background-color: rgba(255, 255, 255, .3);

Поскольку для div родительский элемент — это header, можно передать прозрачному квадрату размеры всего экрана. Установите значение 100% для ширины и высоты div. Цвет квадрата поменяйте на полупрозрачный чёрный: rgba(0, 0, 0, .5).

div {

width: 100%;

height: 100%;

background-color: rgba(0, 0, 0, .5);

# Наследование

Стиль родительских элементов оказывает влияние на стиль дочерних.

Скопировать кодHTML

*<!-- родительский -->*

<div>

*<!-- дочерний -->*

<p>Задача веб-дизайна — сделать не только красиво.</p>

</div>

Например, если абзацу p не задан конкретный размер шрифта, цвет текста или выравнивание, абзац будет «прислушиваться» к родителю, то есть к стилю элемента div.

И если для div определить свойство font-size: 40px, высота символов во вложенном элементе p тоже станет 40px. Установите для div выравнивание по центру text-align: center — и текст абзаца p выровняется по центру.

Это называется наследованием и работает с большинством CSS-свойств. Но если дочернему элементу установить собственное конкретное значение, то свойство наследоваться не будет.

Странице нужно задать новый шрифт. Чтобы не писать свойства для каждого элемента по отдельности, можно сказать что-то сразу всему документу. Для этого обращаются к элементу body, ведь он — корневой родительский элемент для всех видимых элементов на HTML-странице.

Сразу после правила для body, h1 создайте отдельный селектор body и присвойте ему свойство font-family: 'Helvetica', 'Arial', sans-serif.

Оно задаст всей странице шрифт Helvetica. Если такого шрифта нет на устройстве пользователя, браузер попробует использовать Arial. Если нет и Arial — браузер задействует системный шрифт без засечек.

Типографика

Уберите заданные по умолчанию отступы и размеры — скоро вы создадите их сами, без влияния базовых настроек браузера.

Сбросьте всё сразу для нескольких элементов. В селектор body, h1 через запятую добавьте h2, h3, h4, p, a.

body, h1, h2, h3, h4, p, a {

margin: 0;

}

Для сброса размеров текста воспользуйтесь комбинацией font-size: 100%; font-weight: normal. Первое свойство установит единый для всех элементов размер шрифта. Второе свойство задаст толщину шрифта со значением normal

body, h1, h2, h3, h4, p, a {

margin: 0;

font-size: 100%;

font-weight: normal;

}

Текст имеет много настроек, в CSS можно разбить их на две группы. За сам шрифт отвечает группа font, за поведение текстовых элементов — группа text.

Из группы font популярны свойства:

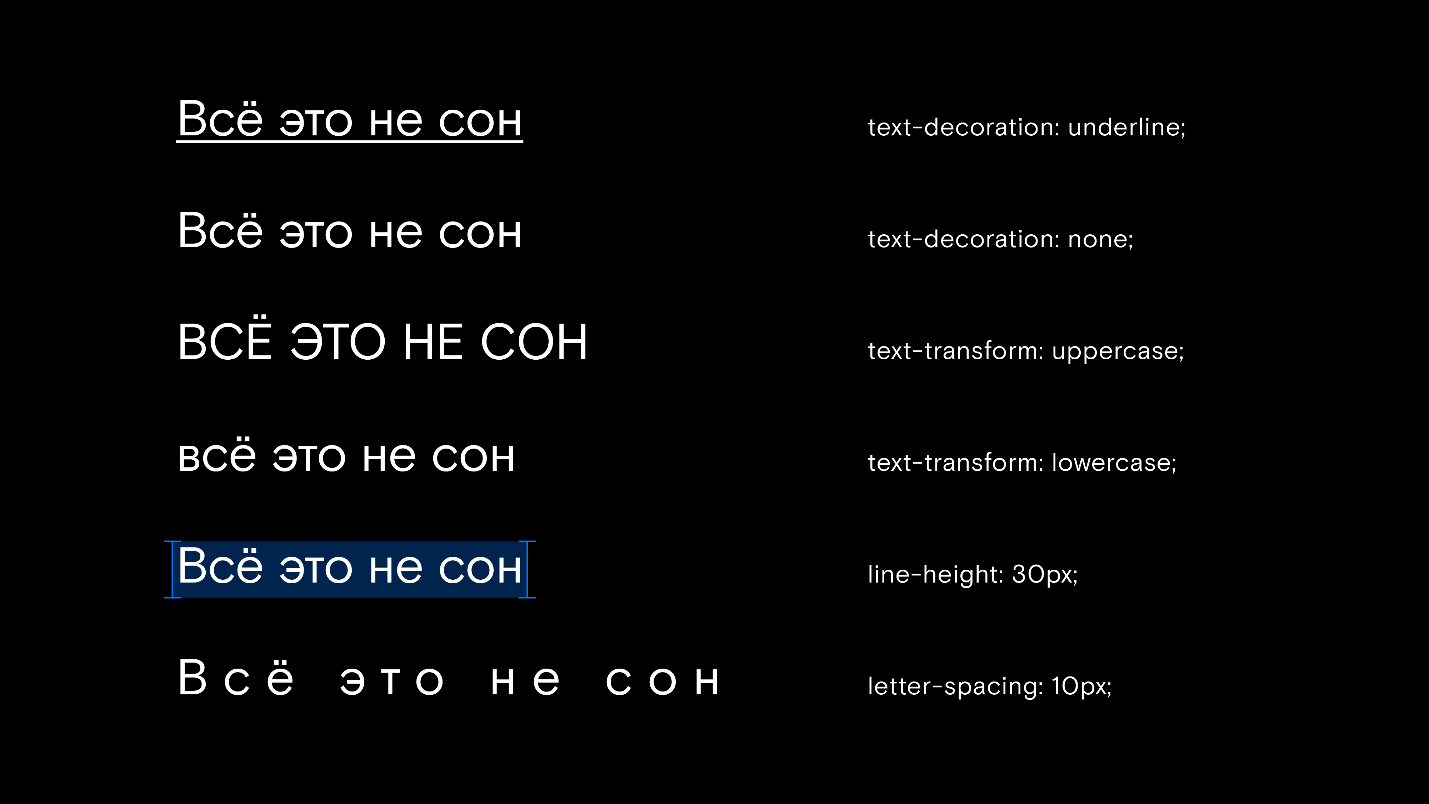
font-size */\* размер шрифта \*/*

font-weight */\* «вес» шрифта, толщина (отношение ширины штриха к высоте буквы) \*/*

font-family */\* семейство шрифта \*/*

font-style */\* стиль шрифта. например, italic — курсивный шрифт \*/*

Из группы text вам знакомо свойство text-align — выравнивание. Часто используют правило text-decoration: underline (подчеркнутый текст) или text-decoration: none (убирает подчёркивание; например, заданное по умолчанию для гиперссылок). Регистром букв управляет text-transform.



Толщину шрифта можно указывать словами, например, bold или normal. Более точная настройка осуществляется числами, вернее, сотнями в диапазоне от 100 (очень тонкое начертание) до 900 (ультра-массивное начертание).

Присвойте заголовку font-weight значение 500, чтобы получить что-то среднее между bold и normal.

Ещё для текста можно установить высоту строки свойством line-height — это называется интерлиньяж. Свойство letter-spacing определяет расстояние между символами в тексте.

*<!-- в HTML-коде комментарий окружён такими символами -->*

*<!-- <p>невидимый пользователю абзац</p> -->*

*/\* в CSS — такими \*/*

*/\* p {*

*color: white;*

*} \*/*

Для сброса размеров текста воспользуйтесь комбинацией font-size: 100%; font-weight: normal. Первое свойство установит единый для всех элементов размер шрифта. Второе свойство задаст толщину шрифта со значением normal.

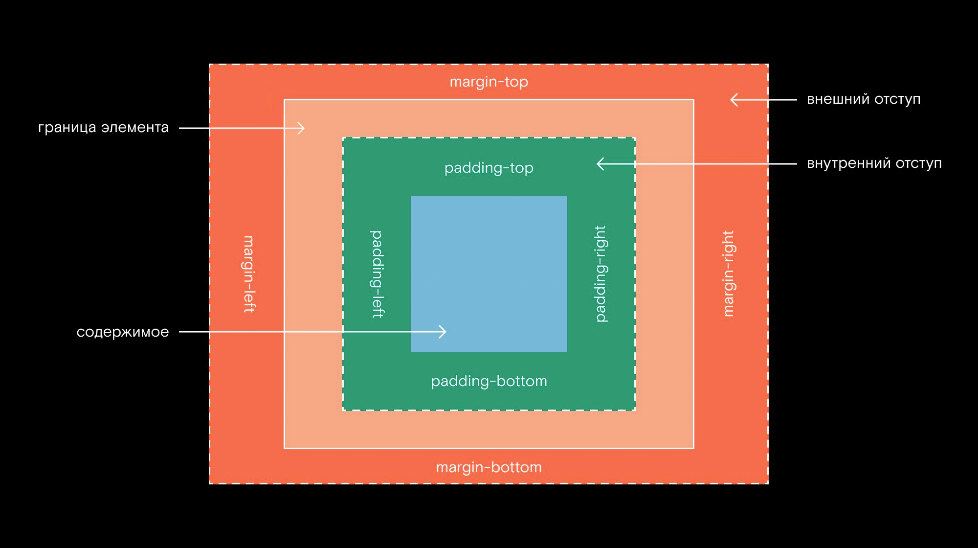
# Поток и блочная модель

Порядок расположения элементов на странице называют поток. Браузер читает код файла сверху вниз и так же отрисовывает страницу. Поэтому говорят, что элементы следуют друг за другом в потоке.

Знакомые вам div, section, header, h1–h6 и p, располагаясь в потоке, по умолчанию занимают всю ширину своего родителя. Такие элементы называются блочными.

Модель любого блочного элемента:

* содержимое с размерами width и height;
* внешние отступы, они же поля: margin;
* внутренние отступы padding;
* границы, которых в коде вы пока не встречали, но ещё встретите.



.card {

width: 350px;

background-color: #e3f3ff;

margin-bottom: 60px;

padding-left: 45px;

padding-right: 45px;

padding-top: 45px;

padding-bottom: 70px;

Короткая запись свойств

Устанавливать отступы для одного элемента с разных сторон можно короче. CSS позволяет записывать свойство в одну строку без уточнений стороны.

/\* направление расстановки — по часовой стрелке, начиная сверху \*/

/\* 4 значения \*/

/\* для каждой стороны – своё значение \*/

padding: 20px 15px 30px 15px;

/\* 3 значения \*/

/\* верх — 10px, по бокам — 20px, низ — 30px \*/

padding: 10px 20px 30px;

/\* 2 значения \*/

/\* верх и низ — 10px, по бокам — 20px \*/

padding: 10px 20px;

/\* 1 значение \*/

/\* 10px со всех сторон \*/

padding: 10px;

Работает и для margin, и для padding.

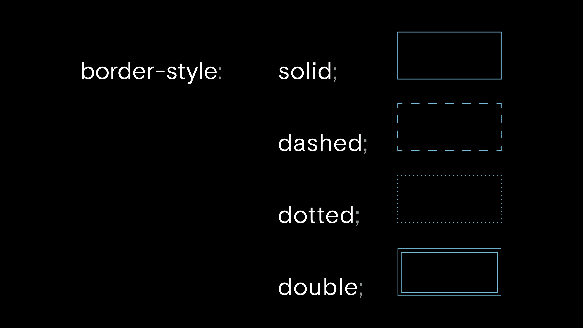
Границы

Граница элемента находится между его margin и padding. В стилях границы определяют свойствами группы border:

border-color: #000; */\* цвет границы \*/*

border-width: 1px; */\* толщина границы в px \*/*

border-style: solid; */\* начертание границы (см. картинку) \*/*



Опять же — вместо написания трёх свойств по отдельности, разработчики применяют краткую форму. Значения color, width и style указывают внутри одного свойства через пробел:

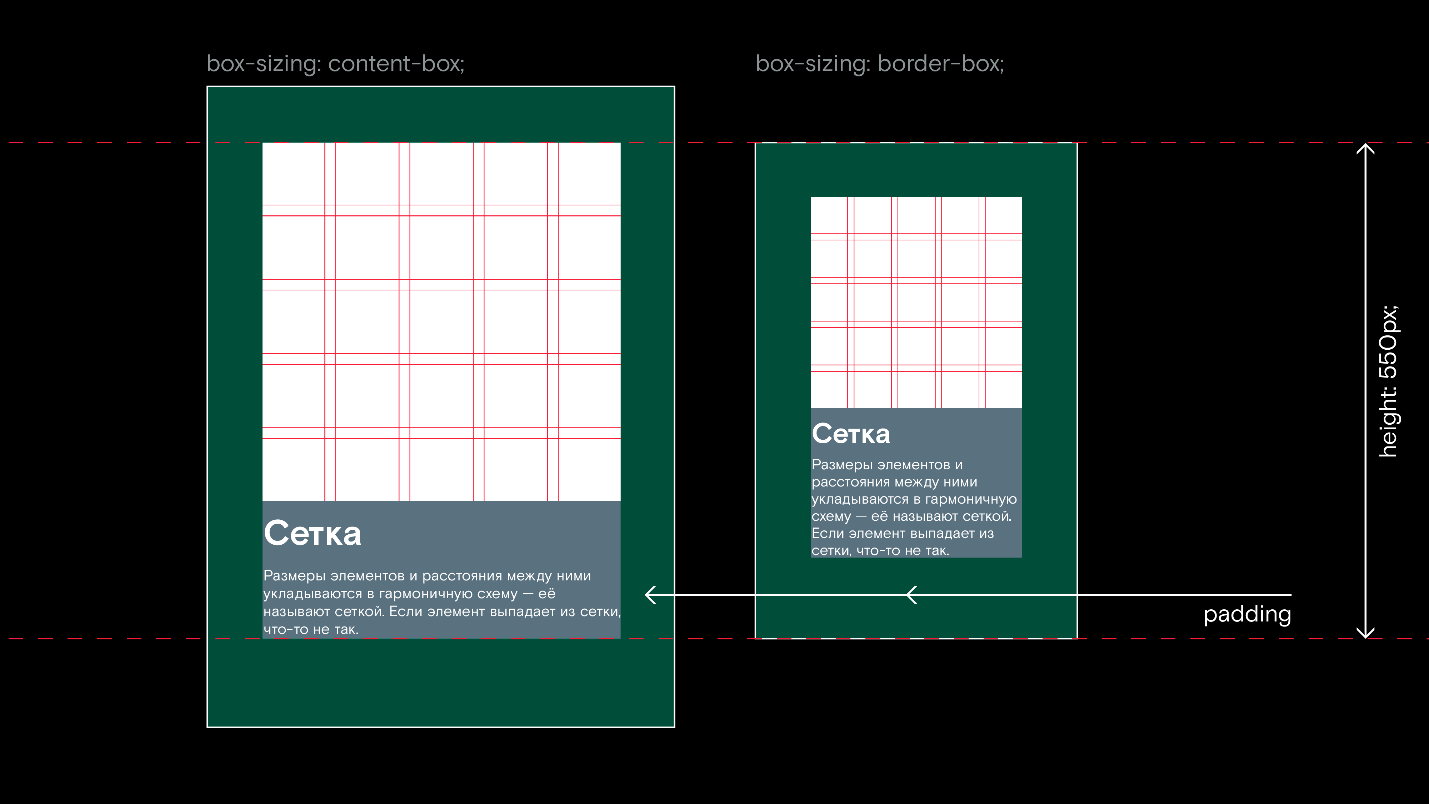
*/\* непрерывная граница черного цвета толщиной 3px \*/*

border: 3px solid #000;

Внешний и внутренний фокус элемента

Обратите внимание на смысловую нестыковку. Ранее вы указали ширину 350px для карточки — как на макете. Но после манипуляций с внутренними отступами и границей, которые увеличили элемент, его ширина уже не та.

Выход есть. Например, брать калькулятор и высчитывать, насколько уменьшить значение width для класса card, чтобы в сумме с отступами и границами вышло 350.



Долго, сложно и чревато ошибками. Чтобы не считать вручную, есть свойство box-sizing — оно определяет поведение границ и внутренних отступов. По умолчанию для всех элементов действует правило box-sizing: content-box, границы и внутренний отступ расширяют элемент.

Установив для box-sizing значение border-box, вы измените фокус элемента: границы и внутренний отступ отрисуются внутрь. Общая ширина будет равна значению width. Довольно интуитивное поведение, поэтому нормальная практика — задать его сразу для всех элементов на странице.

Чтобы задать фокус не только для .card и не печатать box-sizing вручную для всех элементов — познакомьтесь с селектором \*. Он передаёт свойства напрямую каждому элементу на странице. Так как \* очень общий селектор, создайте правило \* { } сразу после сброса стилей, перед body { }.

\* {

box-sizing: border-box;

}

Блочные + строчные

Что делать, когда блоки с определенными размерами должны следовать друг за другом по горизонтали и не занимать всю строку? Можно задать элементам комбинированный тип — блочно-строчный. С одной стороны, они не занимают собой всю горизонталь, с другой, восприимчивы к указанию размеров через CSS. Например, так ведут себя элементы img.

Переопределяется тип свойством display:

*/\* display по-английски «отображение» \*/*

display: block; */\* сделает элемент блочным \*/*

display: inline; */\* сделает элемент строчным \*/*

display: inline-block; */\* сделает элемент блочно-строчным \*/*

Благодаря работе с display становится возможной сетка элементов из текста второй карточки.

И строчные, и блочно-строчные элементы ведут себя как слова в предложении. А между словами стоят пробелы, ширина которых зависит от размера шрифта.

Поэтому карточки встают в строку, но не вплотную, хотя отступов между ними вы не задавали. Уберите непредвиденный зазор — присвойте родительскому элементу *section* класс content и задайте ему свойство font-size: 0.

Селектор .content { } рекомендуем разместить под .overlay { } и перед .card { }.

Это необязательно, ведь код работает и так. Но если у вас будет от 200-300 строк кода, то размещать селекторы где попало — плохой тон; придётся тратить массу времени на поиски нужной строчки.

Гораздо лучше держать стили для одного компонента рядом. По принципу «от общего — к частному», когда правила для дочерних лежат ниже правил для родителей.

# margin: auto;

Напомним, карточки — блочные элементы, и поэтому занимают всю родительскую ширину даже несмотря на установленное width: 350px. Теперь мысленный эксперимент: как бы вы разместили карточки посередине занятой ими строки?

Свойство text-align не поможет — оно действует только на текст. Можно на глаз задать количество пикселей для margin-left первой карточки, пока правила не окажутся в центре. Но такой способ сработает только на вашем компьютере, а для экрана другого размера потребуется другое число пикселей.

С центрированием блочных элементов работает особое значение auto. Оно автоматически устанавливает максимально возможный отступ по горизонтали. Если задать свойствам margin-left и margin-right значение auto, получатся максимальные отступы с обеих сторон, и элемент встанет по центру своего родителя.

Поставьте элемент section в центр экрана, задайте ему внешний отступ со значением auto слева и справа. Сделайте это всего одним свойством, отступам сверху и снизу укажите значение 0.

.content {

font-size: 0;

width: 790px;

margin: 0 auto;

# Тени

Тень — полезный инструмент дизайна. За её создание отвечает свойство box-shadow.

div

{ box-shadow: -2px 2px 5px #FD6969;

*/\* сдвинута на 2px влево, на 2px вниз \*/*

*/\* радиус размытия — 5px \*/*

*/\* цвет красноватый, #FD6969 \*/* }

Последовательность значений, которые описывают тень:

* горизонтальный сдвиг (отрицательное значение помещает тень левее элемента, положительное — правее);
* вертикальный сдвиг (отрицательное значение помещает тень выше элемента, положительное — ниже);
* радиус размытия (чем значение больше, тем тень шире и бледней);
* цвет (задаётся так же, как цвет текста или фона).

Иногда тень не нужно сдвигать, достаточно только размытия. В этом случае указывают, что первые два значения равны 0.

Тень можно создавать не только для границ элемента, но и для текста. В этом случае используют свойство text-shadow. Оно работает аналогично.

Центрировать по вертикали

Способ есть. Для уже изученного свойства display помимо block, inline и inline-block есть особое значение flex:

display: flex;

*/\* \*flex\* переводится как «гибкий» \*/*

.footer {

display: flex;

height: 300px;

background-color: rgb(0,0,0);

}

.footer-author {

color: rgb(250,250,250);

font-size: 24px;

margin: auto;

**Элементы в потоке**

Все HTML-элементы следуют друг за другом: либо находятся на одной строке с соседом, либо начинают новую строку. Это зависит от типа элемента — условно одного из двух:

1. **Строчные элементы.** Занимают ровно столько места, сколько нужно для их контента, располагаются на одной строке с другими строчными элементами и не реагируют на изменение ширины и высоты.

Так ссылка внутри параграфа находится на одной строке с текстом этого параграфа:

Скопировать кодHTML

<p>В этом абзаце есть <a href="http://ya.ru">ссылка</a></p>

1. **Блочные элементы.** По умолчанию занимают всю ширину своего родителя и располагаются на новой строке.

Например, абзацы, контейнеры div и заголовки.

Скопировать кодHTML

*<!-- Все эти элементы займут ширину родителя целиком и будут начинаться с новой строки -->*

<h1>Самый главный заголовок</h1>

<p>Это абзац текста.</p>

<div></div>

Свойство display определяет поведение элементов. Строчные могут стать блочными и наоборот:

Скопировать кодCSS

a {

display: block; */\* теперь ссылка занимает всю ширину родителя и расположена на новой строке \*/*

}

p {

display: inline; */\* абзац займёт ровно те размеры, которые нужны для контента внутри, и окажется на одной строке с другими строчными элементами \*/*

}

Когда блочный элемент становится строчным, ему нельзя задать ширину и высоту. Чтобы он вёл себя как строчный, но не игнорировал свойства width и height, придумали комбинированный тип — блочно-строчный:

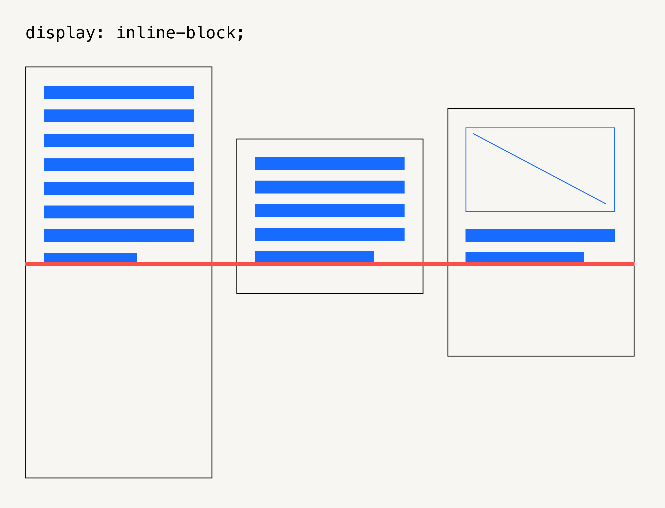
Скопировать кодCSS

p {

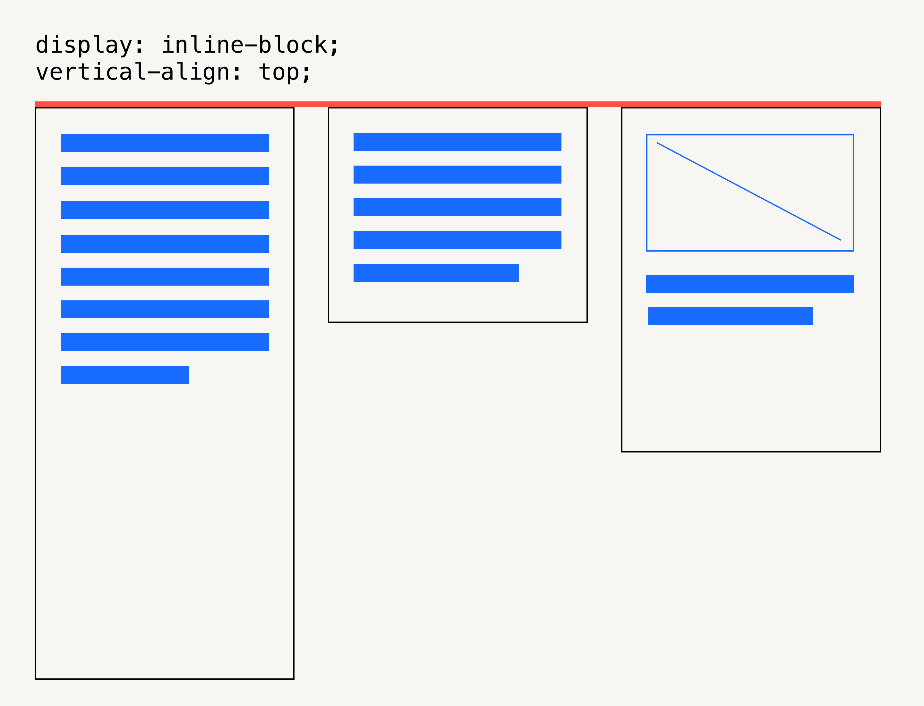
display: inline-block; */\* абзац окажется на одной строке со строчными и блочно-строчными элементами, но ему можно задать ширину и высоту \*/*

}

Блочно-строчные элементы выравниваются по последней строке текста внутри них:



Поэтому вертикальным выравниванием нужно управлять, например, задавать выравнивание по верхней строчке:



Если два блочно-строчных элемента описаны в коде друг под другом, в документе их будет разделять пробел.

Скопировать кодHTML

<p>Первый абзац</p>

<p>Второй абзац</p>

Для этих двух абзацев можно написать такой CSS:

Скопировать кодCSS

p {

display: inline-block;

vertical-align: top;

}

Абзацы встанут в одну строку, но между ними появится пробел.

# Селекторы

Чтобы стилизовать элементы, их нужно выбрать. Для этого существуют селекторы. С их помощью можно сразу выбрать целую группу элементов (параграфы, ссылки, классы) и задать всей этой группе определённые параметры: размер, шрифт, цвет и что только не.

Самые распространённые селекторы:

1. По имени тега:

Скопировать кодCSS

*/\* выбираем все элементы с тегом p на странице \*/*

p {

color: red;

}

1. По имени класса:

Скопировать кодCSS

*/\* выбираем все элементы с классом my-class \*/*

.my-class {

color: red;

}

1. Селекторы вложенности. Их используют, когда нужно выбрать все дочерние элементы по отношению к родительскому элементу:

Скопировать кодCSS

*/\* все p внутри div \*/*

div p {

color: red;

}

*/\* все элементы с классом my-class внутри div \*/*

div .my-class {

color: red;

}

1. Комбинированные селекторы. Их используют, когда нужно выбрать специфичный элемент, например, одновременно с двумя классами или ссылку с определенным параметром:

Скопировать кодCSS

*/\* ссылка с классом my-link \*/*

a.my-link {

color: red;

}

*/\* элемент с классами my-button и my-button\_large одновременно \*/*

.my-button.my-button\_large {

color: red;

}

Для повседневной работы этого хватит. Если хотите разобраться во всех тонкостях выбора элементов, рекомендуем игру для изучения селекторов: <https://flukeout.github.io/> (на английском).

}

# Класс

class — атрибут, который помогает различать элементы. Его можно присвоить нескольким элементам и обращаться к ним по имени класса. Это удобно при работе со стилями и скриптами.

Скопировать кодHTML

<p class="my-paragraph">Параграф с классом my-paragraph</p>

Одному элементу можно присвоить несколько классов. Один класс может задавать параметры всех кнопок, а другой покрасить каждую в свой цвет — в зависимости от контекста использования.

### HTML

Скопировать кодHTML

*<!-- Кнопка отправки формы: -->*

<button class="project-button project-button\_type\_submit">Отправить</button>

### CSS

Скопировать кодCSS

*/\* Все кнопки будут 100 на 30 пикселей, со скруглёнными углами и надписью заглавными буквами \*/*

.project-button {

width: 100px;

height: 30px;

border-radius: 4px;

text-transform: uppercase;

}

*/\* Кнопка отправки формы будет синего цвета стали с белой надписью \*/*

.project-button\_type\_submit {

background-color: steelblue;

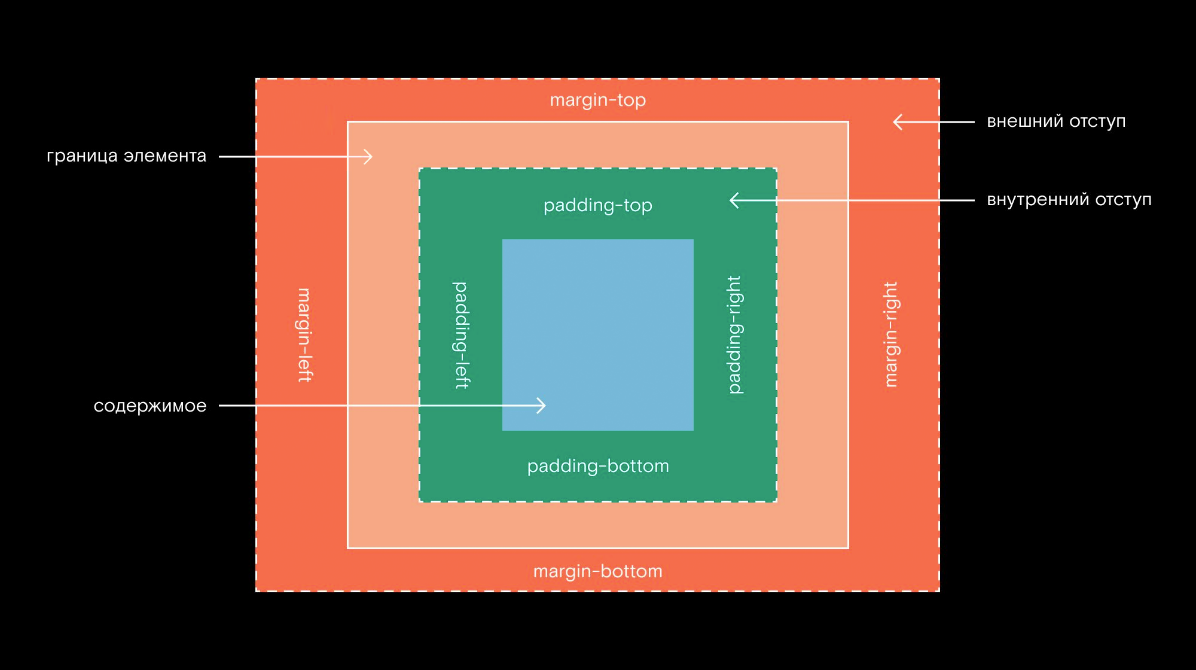
color: white;

}

Принято стилизовать элементы по имени класса, а не тега. Так с ними удобнее работать в CSS.

**Блочная модель документа**

Блок в HTML состоит из нескольких частей: размеров, границы, внешних и внутренних отступов. Это сочетание называют блочной моделью документа. Чаще всего её описывают изображением:



**Размеры блока** задают шириной и высотой. Можно использовать абсолютные (пиксели) и относительные (сотые доли) величины:

* % — от размеров родительского блока,
* vh — от высоты окна браузера,
* vw — от ширины окна браузера.

Скопировать кодCSS

div {

width: 300px;

height: 200px;

}

**Граница** — группа свойств со множеством параметров. Основные:

border-width— толщина границы,

border-style— стиль границы,

border-color — её цвет.

Для краткости эти свойства пишут в одну строку — друг за другом, через пробел.

Скопировать кодCSS

div {

border: 1px solid red; */\* сплошная граница в 1 пиксель красного цвета \*/*

}

**Внешние отступы** margin, или поля — пространство между элементом и его соседями:

Скопировать кодCSS

div {

margin-top: 20px; */\* расстояние до верхнего соседа \*/*

margin-right: 20px; */\* расстояние до правого соседа \*/*

margin-bottom: 20px; */\* расстояние до соседа снизу \*/*

margin-left: 20px; */\* расстояние до соседа слева \*/*

}

*/\* Сокращённая запись того же самого \*/*

div {

margin: 20px;

}

Если блочному элементу установить margin-left и margin-right в значении auto, блок центрируется по горизонтали. Для вертикального выравнивания это не сработает.

Скопировать кодCSS

div {

margin-left: auto;

margin-right: auto;

}

*/\* Сокращённая запись \*/*

div {

margin: 0 auto;

}

Внутренний отступ padding отодвигает контент внутри блока от границы. У padding нет специального значения auto.

Скопировать кодCSS

div {

padding-top: 20px; */\* расстояние от верхней границы до содержимого \*/*

padding-right: 20px; */\* расстояние от правой границы до содержимого \*/*

padding-bottom: 20px; */\* расстояние от нижней границы до содержимого \*/*

padding-left: 20px; */\* расстояние от левой границы до содержимого \*/*

}

*/\* Сокращённая запись \*/*

div {

padding: 20px;

}

Когда мы описываем границу или внутренний отступ, размер блока увеличивается. Добавляется толщина границы, а внутренние отступы «раздувают» блок, чтобы создать пространство вокруг контента.

Такое поведение меняют свойством box-sizing (со значением content-box по умолчанию). Если изменить значение на border-box, границы будут отрисованы внутри блока, а внутренние отступы не повлияют на его размеры:

Скопировать кодCSS

div {

box-sizing: border-box; */\* задали box-sizing \*/*

width: 300px; */\* ширина блока \*/*

height: 200px; */\* высота блока \*/*

border: 2px solid red; */\* граница не влияет на размер \*/*

padding: 20px; */\* отступ не растягивает блок \*/*

}