Для комментирования строк используются специальные символы:

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

*/\* в CSS — такими \*/*

**Относительный путь** – начинается с точки (“***./***” или “***../***”), указывает путь относительно текущего места.

**Абсолютный путь** – начинается со слеша “***/***”, указывает путь от корня проекта.

***Cascading Style Sheets*** – каскадные таблицы стилей отвечают за внешность веб-страницы. Чтобы придать элементу индивидуальный стиль, в тег встраивается атрибут ***style***. Ему присваивается название свойства и значение свойства. Например, атрибут ***color*** позволяет задать цвет.

*<h1 style=”color: springgreen; font-size: 72px;”>Текст</h1>*

Стили можно задавать как в самом *HTML*-документе, так и в отдельном файле. В первом случае внутри элемента *<head></head>* создается элемент *<style></style>* и заполняется селекторами. Во втором случае внутри элемента *<head></head>* создается ссылка на файл с *CSS*. Атрибут ***rel*** задает назначение, ***href*** – адрес.

Для подключения внешних файлов к CSS используется директива импорта ***@import***:

*@import url(fonts.css)*

Стиль родительских элементов оказывает влияние на стиль дочерних (наследуется, если дочернему элементу не задано конкретное значение).

**Каскад** – это механизм работы браузера, по которому порядок следования правил в *CSS* имеет значение. Когда к одному элементу применяются несколько правил, используется то, которое идет в *CSS* последним (ниже остальных по списку).

Если к одному и тому же элементу подходит сразу несколько *CSS*-правил, включается алгоритм **специфичность** – браузер взвешивает каждый селектор и применяет стили того, что тяжелее. Для расчета веса селектора используется следующая таблица:

* Инлайн-стили в *HTML* – вес 1000.
* Идентификаторы – вес 100.
* Классы, атрибуты – вес 10.
* Псевдоклассы – вес как у самого тяжелого селектора в скобках.
* Теги и псевдоэлементы – вес 1.

Ключевое слово ***!important*** позволяет перебить любой вес других *CSS*-правил. Перебить его можно только таким же ключевым словом.

*div {*

*color: tomato !important;*

*}*

Рекомендуется описывать *CSS* стили элементов от общего к частному в порядке их следования в *HTML*.

Самый частый лайфхак – указывать старый вариант, который поддерживают все браузеры, а за ним писать новую версию, которая может не работать на некоторых браузерах.

Для придания индивидуальности каждому элементу отдельно можно использовать атрибут ***class***. **Классы** – это уникальные селекторы, на которые ссылаются из *HTML*-кода.

*// в html название класса указывается в кавычках атрибута class*

*<h1 class="white-text">Откуда к нам пришла зима</h1>*

*<p class="white-text">Не знаешь ты, никто не знает</p>*

*// в стилях селектор класса начинается с точки*

*.white-text {*

*color: #fff;*

*}*

Одному элементу можно задать несколько классов, записав их через пробел.

*<p class="****text special-text****">Bonobo, Rhye — Break Apart</p>*

Для изменения стиля сразу во многих элементах можно использовать **селектор**.

*h2 { // селектор по тегу h2*

*font-size: 32px;*

*}*

*.article-title { // селектор по классу class=”article-title”*

*font-size: 32px;*

*}*

*#second { // селектор по идентификатору id=”second”*

*font-size: 32px;*

*}*

*.article-title, .text-footer { // перечисление селекторов по классам*

*font-size: 32px;*

*}*

*div p { // все p внутри div*

*margin-left: 20px //* ***вложенный*** *селектор потомка (любой уровень вложенности)*

*}*

*.app .best-choice { // все элементы с class=”best-choice” внутри class=”app”*

*margin-left: 20px //* ***вложенный*** *селектор потомка (любой уровень вложенности)*

*}*

*.app.best-choice { // элементы с одномоментным объявлением class=”.app .best-choice”*

*font-style: italic; //* ***комбинированный*** *селектор потомка (любой уровень вложенности)*

*}*

*.app > .best-choice { // все элементы с class=”best-choice”, являющиеся прямыми потомками class=”app”*

*margin-left: 20px //* ***вложенный*** *дочерний селектор (вложенность первого уровня)*

*}*

*.app + .best-choice { // элемент class=”best-choice” стоит сразу после элемента class=”app” в HTML*

*margin-left: 20px // селектор* ***смежных элементов***

*}*

*.app ~ .best-choice { // применится к class=”best-choice”, если он на одном уровне с class=”app” в HTML*

*margin-left: 20px // селектор* ***соседей*** *(одинаковый уровень вложенности)*

*}*

*[disabled] { // селектор по атрибуту disabled*

*font-size: 32px;*

*}*

*[href=”#”] { // селектор по значению атрибута href, которое равно #*

*font-size: 32px; // другие селекторы по значению атрибута можно найти* [*тут*](https://doka.guide/css/attribute-selector/)

*}*

**Специальный селектор** для удаления значений по умолчанию:

*body, h1, h2, h3, h4, h5, h6, p, a {*

*margin: 0;*

*font-size: 100%;*

*font-weight: normal;*

*}*

Сразу после сброса стилей, можно задать общий для всех элементов **универсальный селектор** ***\****, а также для всех используемых псевдоэлементов:

*\*,*

*\*::before,*

*\*::after {*

*box-sizing: border-box;*

*}*

Порядок расположения элементов на странице называется **потоком**. Браузер читает код файла сверху вниз и также отрисовывает страницу, поэтому говорят, что элементы следуют друг за другом в потоке.

Элементы *div*, *section*, *header*, *h1-h6*, *p*, располагаясь в потоке, занимают 100% ширины своего родителя, поэтому называются **блочными**. Значение элемента *body* равно видимой ширине окна браузера.

Кроме блочных элементов (*div, section, body, header, footer*), в *HTML* есть **строчные**. Строчные элементы (*span*) занимают ровно столько места, сколько контента в них содержится. Если они идут подряд, то по умолчанию все расположены на одной строке. Они игнорируют указание размеров через стили.

Также существует комбинированный тип – **строчно-блочный**. С одной стороны, они не занимают собой всю горизонталь, с другой, восприимчивы к указанию размеров через *CSS*. Если они не помещаются на одной строке, то переносятся на следующую. Например, так ведут себя элементы *img*.

Свойство ***display*** позволяет переопределить тип отображения элемента:

*display: block; // сделает элемент блочным*

*display: inline; // сделает элемент строчным*

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

*display: flex; // сделает flex-элементом*

Для того чтобы убрать зазор между строчно-блочными элементами, для родительского элемента нужно задать нулевой размер шрифта:

*font-size: 0*;

Строчно-блочные элементы выравниваются по последней строке текста внутри них. Вертикальным выравниванием можно управлять с помощью свойства ***vertical-align***. Задать размер можно с помощью свойств ***inline-size*** и ***block-size***.

Не рекомендуется указывать фиксированные размеры ***width*** и ***height***. Вместо них лучше использовать либо относительные единицы, либо ***min-width***, ***min-height***, ***max-width*** и ***max-height***.

Окно просмотра измеряется в единицах ***vw*** (*view width*) и ***vh*** (*view height*) относительно размеров окна браузера. При значении ***width: 100vw*** появится полоса скролла. При значении ***width: 100%*** полосы скролла не будет.

В некоторых случаях проценты не будут работать. Для высоты они работают только во флекс-контейнере или когда высота родителя задана явно.

Лайфхак для настройки размеров на мобильных устройствах:

*html {*

*height: 100%;*

*}*

*body {*

*height: 100%;*

*}*

Значения ***vmin*** и ***vmax*** вычисляются от наименьшего/наибольшего значений высоты и ширины браузера.

Значение ***vi*** равняется 1% размера окна просмотра по направлению оси выстраивания строчных элементов.

Значение ***vb*** равняется 1% размера окна просмотра по направлению оси выстраивания блочных элементов.

На мобильных устройствах не учитываются размеры управляющих элементов, поэтому появляется скроллинг. В связи с этим придумали новые единицы:

* Единицы для обозначения большого окна просмотра имеют префикс ***lv*** (*lvw*, *lvh*, *lvi*, *lvb*, *lvmin*, *lvmax*).
* Единицы для обозначения малого окна просмотра имеют префикс ***sv*** (*svw*, *svh*, *svi*, *svb*, *svmin*, *svmax*).
* Динамические единицы для обозначения окна просмотра имеют префикс ***dv*** (*dvw*, *dvh*, *dvi*, *dvb*, *dvmin*, *dvmax*).

Задавать цвет можно не только английскими словами (147 штук), но и палитрами ***RGB*** или ***HEX-colors***. Не важно строчные или заглавные буквы используются. Если буквы повторяются, их можно сократить в два раза.

*rgb(255, 255, 0) // желтый*

*#FFFF00 // желтый*

*#FF0 // желтый*

Задать прозрачность можно с помощью **альфа-канала**. Величина прозрачности меняется от 1 до 0 (от непрозрачного к прозрачному). Дробь можно писать без нуля.

*rgba(255, 255, 0, 0.2) // желтый с 80% прозрачности*

За отображение фона отвечают свойства из группы ***background***:

* ***background-color*** позволяет задать фоновый цвет.Например:

*background-color: rgb(255, 255, 0);*

* ***background-image*** позволяет задать фоновую картинку. Например:

*background-image: url(https://yastatic.net/s3/auth2/\_/msoffice.d1ada09f.svg);*

* ***background-position*** позволяет задать положение фона. Например:

*background-position: center; // по центру и горизонтально и вертикально*

*background-position: right bottom; // прижать к нижнему правому углу*

*background-position: 150px -50px; // сдвиг на 150px вправо и на 50px вверх*

* ***background-size*** позволяет задать размер картинки фиксировано или в пропорциях. Например:

*background-size: cover; // сожмёт или растянет картинку, сохранив пропорции*

*background-size: contain; // картинка занимает всю доступную площадь, остальное - обрезается*

*background-size: 220px 400px; // задаст фикс размер ширина + высота*

* ***background-repeat*** позволяет отменить повторение изображения, если оно меньше поверхности блока. Например:

*background-repeat: no-repeat; // не повторяться*

*background-repeat: repeat; // с повтором*

*background-repeat: repeat-y; // с повтором только по вертикали*

* ***background-clip*** позволяет сделать трафарет из блока. Например:

*background-clip: text; // вырежем текст внутри блока*

*-webkit-background-clip: text;*

*color: transparent; // сделаем текст прозрачным*

*background-image: url(./image.jpg); // через дырки от теста будет видна картинка*

Свойству *background-image* можно задать несколько градиентов через запятую.

Свойство ***linear-gradient*** позволяет задать линейный градиент для *background-image*. Изменение цвета будет происходить по линии его направления (по умолчанию сверху вниз). Задать направление градиента можно с помощью угла или ключевых слов ***to left***, ***to top***, ***to right*** и ***to bottom***.

*background-image: linear-gradient(150deg, #0078FF, #C2E3E3, #126DDC);*

С помощью градиента можно задавать начальные и конечные точки перехода между цветами, и таким образом создавать монотонные участки с одним цветом. Если эти точки совпадают, то переход будет резким.

*// градиент будет идти с 80px до 120px и с 180px до 220px*

*background-image: linear-gradient(#80d994 0px 80px, #6dc1fd 120px 180px, #5f0dee 220px 300px);*

Свойство ***radial-gradient*** позволяет задать радиальный градиент для *background-image*. Изменение цвета будет происходить от центра в форме эллипса. По умолчанию центр градиента находится в середине блока, но его можно сместить с помощью ключевых слов ***at***, ***at left***, ***at top***, ***at right*** и ***at bottom***.

Для изменения формы градиента можно задать ему ширину и высоту через пиксели или ключевые слова ***closest-side***, ***farthest-side***, ***closest-corner***, ***farthest-corner***. Ключевое слово ***circle*** позволяет превратить его в круг.

*// сместить центр на 40px правее левого края блока и на 50px ниже верхнего края блока*

*background-image: radial-gradient(circle, at 40px 50px, #0078FF, #126DDC);*

*// высота 200px и ширина 100px*

*background-image: radial-gradient(100px 200px, at 40px 50px, #0078FF, #126DDC);*

Радиальный градиент, как и линейный, поддерживает начальные и конечные точки перехода.

Свойство ***conic-gradient*** позволяет задать конический градиент для *background-image*. По умолчанию конический градиент начинается и заканчивается вертикальной линией наверху. Для более плавного перехода рекомендовано начинать и заканчивать градиент одним и тем же цветом. Сместить линию начала градиента можно с помощью ключевого слова ***from***. По умолчанию центр градиента находится в середине блока, но его можно сместить с помощью ключевых слов ***at***, ***at left***, ***at top***, ***at right*** и ***at bottom***.

*// сместить центр к левому краю и повернуть ось начала на 30 градусов*

*background-image: conic-gradient(from 30deg at left, #6dc1fd, #feafe8, #6dc1fd);*

Конический градиент, как и линейный, поддерживает начальные и конечные точки перехода.

Свойства ***repeating-linear-gradient***, ***repeating-radial-gradient*** и ***repeating-conic-gradient*** позволяют задать повторяющийся градиент для *background-image*. Градиент повторяется до тех пор, пока не заполнит весь элемент. Синтаксис такой же, как и у обычных градиентов.

*// заполнит весь блок полосками по 30px*

*background-image: repeating-linear-gradient(#feafe8, #6dc1fd 30px);*

Свойство ***mix-blend-mode*** позволяет смешивать отдельные элементы. Плохая совместимость с *Safari*. Оно может принимать значения:

* ***normal*** – цвета не смешиваются, а накладываются поверх друг друга.
* ***multiply*** – цвет верхнего слоя умножается на цвет нижнего.
* ***screen*** – изображения накладываются друг на друга как полупрозрачные.
* ***overlay*** – комбинирует режим *screen* для светлых участков и режим *multiply* для темных.
* ***darken*** – для итогового слоя берутся самые темные участки.
* ***lighten*** – для итогового слоя берутся самые светлые участки.
* ***color*** – оттенок и насыщенность верхнего элемента применяются к нижнему.

Свойство ***isolation*** позволяет задать уровень изоляции элементов для *mix-blend-mode*.

Свойство ***background-blend-mode*** позволяет объединять несколько фонов у одного элемента. Каждому слою можно прописать свой режим смешивания. Использует те же значения, что и *mix-blend-mode*. Слой *background-color* всегда считается самым нижним, но обойти это ограничение можно с помощью градиентов и переходом цвета в самого себя.

*.element {*

*background-image: url(images/picture1.jpg), url(images/picture2.png);*

*background-blend-mode: overlay, saturation;*

*}*

Единица измерения ***em*** опирается на размер буквы **М** в любом шрифте. В *CSS* размер шрифта в единицах ***em*** считается от размера шрифта родительского элемента (аналогично процентам).

Размер шрифта в единицах ***rem*** считается от размера корневого элемента *html*. По умолчанию это ***16px***.

*.parent {*

*font-size: 24px;*

*}*

*.child {*

*font-size: 2em; // размер равен 48px*

*font-size: 2rem; // размер равен 32px*

*}*

При использовании ***em*** для других свойств, кроме шрифта, размеры будут считаться от размера шрифта этого элемента.

*.parent {*

*font-size: 20px;*

*}*

*.element {*

*font-size: 1.5em; // 30px*

*border: 0.5em solid #6dc1fd; // 15px*

*}*

Единица измерения ***ch*** опирается на ширину символа **0** (ноль) в шрифте. Для моноширных шрифтов ширина будет равна ширине любого символа.

Единица измерения ***ex*** опирается на высоту строчной буквы **x** в шрифте.

Единица измерения ***lh*** опирается на размер межстрочного интервала родительского элемента. Единица измерения ***rlh*** опирается на размер межстрочного интервала корневого элемента *html*. Не поддерживаются всеми браузерами.

***serif*** – системный шрифт с засечками

***sans-serif*** – системный шрифт без засечек

***monospace*** – шрифт с одинаковой шириной символов

***cursive*** – наклонный системный шрифт

***fantasy*** – декоративный системный шрифт

За отображение шрифта отвечают свойства из группы ***font***:

* ***font-size*** позволяет задать размер шрифта. При указании размера шрифта в процентах, они высчитываются от родительского элемента. Например:

*font-size: 24px;*

*font-size: larger; // на размер больше родительского элемента*

*font-size: smaller; // на размер меньше родительского элемента*

* ***font-weight*** или ***wght*** позволяет задать толщину текста (жирный). Можно задавать как словами, так и числами в диапазоне от 100 до 900. Например:

*font-weight: normal;*

*font-weight: 400;*

* ***font-family*** позволяет задать семейство шрифта в приоритетном порядке. Например:

*font-family: 'Helvetica', 'Arial', sans-serif;*

* ***font-style*** или ***slnt*** позволяет задать угол наклона шрифта. Например:

*font-style: italic;*

* ***font-stretch*** или ***wdht*** позволяет задать ширину символа. Например:

*font-stretch: 120%;*

* ***font-optical-sizing*** или ***opsz*** позволяет задать оптический размер. Может принимать только значения ***auto*** и ***none***. Например:

*font-optical-sizing: auto;*

Шорткат ***font-variation-settings*** позволяет объединить параметры *font-weight*, *font-stretch*, *font-style* и *font-optical-sizing* в одну запись. Достаточно указать сокращенное свойство. При отсутствии свойства оно сбрасывается на значение по умолчанию.

*font-variation-settings: 'wght' 850, 'wdth' 110, 'opsz' 50, 'slnt' -7;*

***serif*** – системный шрифт с засечками

***sans-serif*** – системный шрифт без засечек

***monospace*** – шрифт с одинаковой шириной символов

***cursive*** – наклонный системный шрифт

***fantasy*** – декоративный системный шрифт

Для подключения нового шрифта, сначала нужно его декларировать (указать откуда его брать и что с ним делать). Это можно сделать с помощью директивы ***@font-face*** в начале *CSS*-файла:

*@font-face {*

*src: local(‘Best font ever’), // путь к локальному файлу*

*url(https://pictures.s3.yandex.net/fonts/Ibmplexserif.woff); // путь к файлу в сети*

*// подключение альтернативных форматов шрифтов для разных браузеров*

*src: url(https://pictures.s3.yandex.net/fonts/Ibmplexserif.woff) format(‘woff’),*

*url(https://pictures.s3.yandex.net/fonts/Ibmplexserif.woff2) format(‘woff2’);*

*font-family: 'Best Font Ever'; // сами задаем имя шрифту*

*font-display: swap;*

*}*

*// различные начертания шрифта подключаются отдельно*

*@font-face {*

*src: url(https://pictures.s3.yandex.net/fonts/Ibmplexserif.woff);*

*font-family: 'Best Font Ever';*

*font-style: italic;*

*font-weight: bold;*

*}*

*<link rel="stylesheet" href="./styles/fonts.css" /> // подключаем шрифты*

*<link rel="stylesheet" href="./styles/style.css" /> // подключаем стили*

*div {*

*// Если шрифт из файла не загрузился, использовать Arial или Helvetica,*

*// на крайний случай — любой шрифт без засечек в системе пользователя*

*font-family: 'Best Font Ever', 'Arial', 'Helvetica', sans-serif;*

*font-weight: bold;*

*font-style: italic;*

*}*

Пока скачивается файл *'Best Font Ever'* по ссылке из интернета, браузер отрисовывает весь текст доступным шрифтом *Arial* и лишь после скачивания начинает заменять шрифт текста. За логику замены шрифта отвечает свойство ***font-display***:

*font-display: block;* // отрисует прозрачным альтернативным шрифтом текст, дождётся загрузки и

// перерисует видимым скачанным шрифтом весь текст.

*font-display: swap;* // сначала покажет альтернативный шрифт, дождётся загрузки и заменит

// отображение текста. Может возникнуть мигание, но текст будет виден сразу.

Ссылка на самую большую библиотеку бесплатных шрифтов <https://fonts.google.com/> .

**Вариативные шрифты** – это один файл со всеми вариациями одного семейства шрифта (*Thin*, *Bold*, *Italic* и т.д.). Его преимущества заключаются в быстрой загрузке, малом весе (как 2 обычных шрифта) и возможности тонкой настройки (у стандартных шрифтов шаг жирности 100). Для получения информации о возможностях конкретного вариативного шрифта можно воспользоваться сервисом <https://wakamaifondue.com/>.

Вариативные шрифты подключаются также, как и обычные, но дополнительно нужно указывать специальный формат ***supports variations*** и формат для старых браузеров ***woff2-variations***. Для тонкой настройки можно задать минимальное и максимальное отклонение.

*@font-face {*

*font-display: swap;*

*font-family: 'Roboto-Flex-Variable';*

*src:*

*local('Roboto-Flex-Variable'),*

*url('../fonts/Roboto-Flex-Variable.woff2') format('woff2 supports variations'),*

*url('../fonts/Roboto-Flex-Variable.woff2') format('woff2-variations');*

*font-weight: 400 700; // жирность от 400 до 700*

*font-stretch: 100% 120%; // ширина символов от 100 до 120 процентов*

*}*

Для гарантированного отображения шрифтов, лучше всего применять стандартный шрифт, а вариативный загружать с помощью директивы ***@supports*** только при поддержки браузером.

*body {*

*font-family: 'Roboto', sans-serif;*

*}*

*@supports (font-variation-settings: normal) {*

*body {*

*font-family: 'Roboto-Flex-Variable', sans-serif;*

*}*

*}*

Ссылка на самую большую библиотеку бесплатных вариативных шрифтов <https://v-fonts.com/> .

Свойства ***overflow-x*** и ***overflow-y*** задают **родительскому блоку**, что делать с его непомещающимся содержимым по двум направлениям. Свойство ***overflow*** позволяет объединить оба направления:

*overflow: hidden scroll; // hidden по оси X и scroll по оси Y*

*overflow: visible; // значение по умолчанию, раздувает блок и отображает весь контент*

*overflow: auto; // включает скролл, если случилось переполнение*

*overflow: scroll; // включает скролл всегда*

*overflow: hidden; // скрывает лишний контент*

*overflow: clip; // обрезает лишний контент и запрещает скролл*

За поведение текстовых элементов отвечают свойства из группы ***text***:

* ***text-align*** позволяет выравнивать текст.Например:

*text-align: center; // выровнять по центру*

* ***text-decoration*** позволяет задать подчеркивание текста. Например:

*text-decoration: underline; // подчеркнуть текст*

*text-decoration: line-through; // перечеркнутый текст*

*text-decoration: none; // убирает подчеркивание по умолчанию для гиперссылок*

* ***text-transform*** позволяет задать регистр букв текста. Например:

*text-transform: uppercase; // перевести в верхний регистр*

*text-transform: lowercase; // перевести в нижний регистр*

*text-transform: capitalize; // перевести все первые буквы в словах верхний регистр*

* ***line-height*** позволяет задать высоту строки текста. Например:

*line-height: normal; // размер шрифта font-family умноженному на 1,2*

*line-height: 1,5; // размер шрифта font-family умноженному на 1,5*

*line-height: 3px; // длина в пикселях и других условных единицах*

*line-height: 120%; // за 100% берется размер* ***шрифта font-family****, а не родительского элемента*

* ***letter-spacing*** позволяет задать расстояние между символами в тексте. Например:

*letter-spacing: normal; // размер задается аналогично line-height*

* ***text-shadow*** позволяет задать тень для текста. Для одного элемента можно задать сразу несколько теней через запятую.

*// сдвинута вправо на 2px и вниз на 4px с размытием 3px, цвет тени – черный с прозрачностью 0,5*

*text-shadow: 2px 4px 3px rgb(0 0 0 / 0.5);*

* ***text-stroke*** позволяет задать обводку для текста. Экспериментальное свойство.

*@supports((text-stroke: 1px #000000) or (-webkit-text-stroke: 1px #000000)) {*

*p {*

*-webkit-text-stroke: 1px ##000000;*

*text-stroke: 1px #000000;*

*text-shadow: none;*

*}*

*}*

* ***white-space*** задает браузеру, как обрабатывать пробелы в тексте. Например:

*white-space: normal; // текст переносится по пробелам и достижении границы родителя*

*white-space: nowrap; // текст перестает переноситься по пробелам и выводится в одну строку*

*white-space: pre; // сохраняется форматирование, границы элемента игнорируются*

*white-space: pre-wrap; // сохраняется форматирование, границы элемента сохраняются*

*// сохраняется форматирование, но пробелы схлопываются, границы элемента сохраняются*

*white-space: pre-line;*

* ***text-overflow*** задает поведение для лишнего текста. Работает только вместе с ***white-space: nowrap*** и ***overflow: hidden***. Например:

*text-overflow: clip; // обрежет лишний текст по краю*

*text-overflow: ellipsis; // появится многоточие*

* ***overflow-wrap*** задает поведение для переноса длинных слов. Например:

*overflow-wrap: normal; // перенесет слово целиком либо выйдет за границу родителя*

*overflow-wrap: break-word; // разобьет слово в любой части*

*overflow-wrap: anywhere; // разобьет слово в любой части с учетом правил мягкого переноса (слоги)*



Свойство ***margin*** создает пустое пространство между элементом и его соседями (внешние отступы). По умолчанию задается для всех сторон. Для задания отступа с конкретной стороны используются соответствующие свойства ***margin-top***, ***margin-right***, ***margin-bottom*** и ***margin-left***.

Если задать свойствам *margin-left* и *margin-right* значения ***auto***, то элемент получит максимальные отступы с обоих сторон и встанет по центру своего родителя.

Если задать *margin-top* для самого первого элемента в коробке *<div>*, то сдвинется не первый элемент, а сама коробка. Именно поэтому используется ***padding***.

Свойство ***padding*** позволяет отступить от границы текущего элемента внутрь (внутренние отступы). Для задания отступа с конкретной стороны используются соответствующие свойства ***padding-top***, ***padding-right***, ***padding-bottom*** и ***padding-left***.

Установить отступы *margin* или *padding* для элемента можно через короткую запись:

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

*padding: 20px 15px 30px 15px;*

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

*padding: 10px 20px 30px;*

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

*padding: 10px 20px;*

*// 10px со всех сторон*

*padding: 10px;*

**У отступов в процентах** есть важная особенность – их размер вычисляется **относительно ширины** родительского элемента, а не размеров самого элемента. Причем неважно направление отступа.

В примере ниже, мы задаем соотношение сторон 16:9 для фрейма с видео, находящегося внутри резинового контейнера.



**Выпадение полей** – нестандартное поведение блока (кроме *flex* и *grid* блоков), при котором отступы его дочерних элементов смещают сам родительский блок. Это связано с тем, что у родительского блока не заданы границы, внутренние отступы или *overflow*. Достаточно добавить что-то одно для исправления такого поведения, либо сделать блок *flex* или *grid* контейнером.



**Схлопывание полей** – нестандартное поведение блока (кроме *flex* и *grid* блоков), при котором стыкуются соседние блоки и меньший отступ поглощается большим (игнорируется). С боковыми полями схлопывание работает так же.



Когда называют размер элемента, то имеют в виду размер содержимого до границ включительно, *margin* сюда не входит.

Свойство ***box-sizing*** определяет поведение границ и внутренних отступов. По умолчанию для всех элементов действует правило *box-sizing: content-box* (границы и внутренний отступ расширяют элемент). Установив *box-sizing: border-box*, границы и внутренний отступ отрисуется внутрь.

Свойство ***box-shadow*** отвечает за создание тени элемента. Необязательное значение ***inset*** позволяет изменить тень на блик внутри элемента. Для одного элемента можно задать сразу несколько теней через запятую.

*div {*

*box-shadow: inset -2px 2px 20px 5px #FD6969;*

*// сдвинута влево на 2px и вниз на 2px с размытием 5px, цвет тени – красноватый, #FD6969*

*// радиус распространения тени 20px (опционально)*

*}*

Свойство ***outline*** отвечает за создание обводки вокруг элемента.

Свойство ***outline-offset*** отвечает за отступ обводки вокруг элемента.

*div {*

*outline: green solid 3px; // сплошная обводка зеленого цвета толщиной 3 пикселя*

*outline-offset: 2px; // отступ обводки в 2 пикселя*

*}*

За отображение границы элемента отвечают свойства из группы ***border***:

* ***border-color*** позволяет задать цвет границы.Например:

*border-color: #000;*

* ***border-width*** позволяет задать толщину границы элемента. Например:

*border-width: 1px;*

* ***border-style*** позволяет задать стиль границы элемента. Например:

*border-style: solid;*



Установить свойства *border* для элемента можно через короткую запись:

*border: 3px solid #000;*

С помощью свойств из группы ***border-image*** в качестве рамки можно задать изображение:

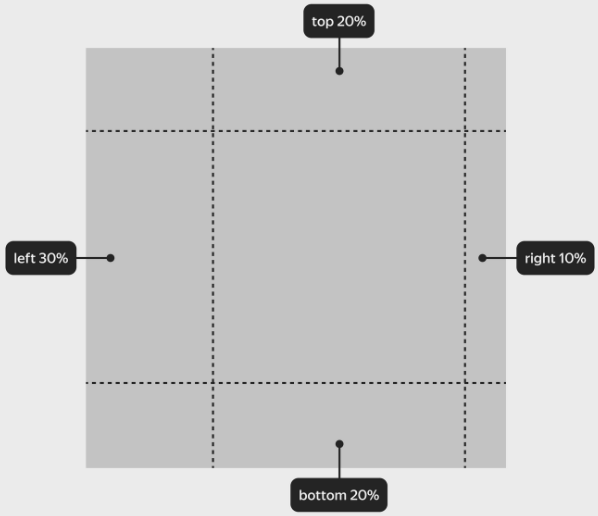
* ***border-image-source*** позволяет задать источник картинки.Например:

*border-image-source: url("image.jpg");*

* ***border-image-slice*** позволяет управлять боковыми размерами рамок. Например:

*border-image-slice: 20% 10% 20% 30%; // в центре картинки не будет*

*border-image-slice: 20% 10% 20% 30% fill; // в центре будет картинка от рамки на фоне контента*



* ***border-image-repeat*** позволяет задать, как браузер будет заполнять пространство между уголками рамки. Можно задать до двух значений (первое для горизонтальной оси, второе – для вертикальной оси). Например:

*border-image-repeat: stretch; // растянуть или сжать с сохранением пропорций и приклеить к углам*

*border-image-repeat: repeat; // повторять фрагменты с обрезанием с сохранением пропорций*

*border-image-repeat: round; // повторять фрагменты без обрезанием и без сохранения пропорций*

*// повторять фрагменты с сохранением пропорций, остается пустое место*

*border-image-repeat: space;*

* ***border-image-width*** позволяет задать толщину рамки, аналогично *border-width*. Например:

*border-image-width: 30px 20%; // сверху и снизу по 30 пикселей, а с боков – 20%*

* ***border-image-outset*** позволяет выйти за края рамки и даже блока без смещения соседних элементов. Например:

*border-image-outset: 30px 30px;*

Шорткат ***border-image*** объединяет все свойства и позволяет использовать сокращенную запись:

*// source slice / width / outset repeat*

*border-image: url("border.png") 27 33 / 20px 30px / 1.5rem round;*

Шорткат ***transform*** позволяет управлять изменением состоянием объекта и объединяет несколько функций:

* ***scale()*** – отвечает за изменение масштаба элемента без влияния на соседей. Она объединяет функции ***scaleX()***, ***scaleY()*** и ***scaleZ()***. В качестве аргумента может принимать одно (сразу для строчной оси X и блочной оси Y), два (первое – для оси X, второе – для оси Y) или три значения (первое – для оси X, второе – для оси Y, третье – для оси Z).

*.card:hover {*

*transform: scale(1.1); // задаем увеличение карточки на 10% при наведении*

*}*

* ***rotate()*** – отвечает за поворот элемента. В качестве аргумента может принимать градусы *deg*, повороты *turn*, радианы *rad* и грады *grad*. Если потребуется 3D-поворот, то можно указать, к какой конкретно оси относится заданное значение.

*.card:hover .card-image {*

*transform: rotate(5deg); // задаем поворот картинки карточки на 5 градусов при наведении*

*rotate: x 5deg; // альтернативная запись*

*}*

* ***translate()*** – отвечает за смещение элемента. Она объединяет функции ***translateX()***, ***translateY()*** и ***translateZ()***. В качестве аргумента может принимать одно (только для горизонтальной оси X), два (первое – для оси X, второе – для вертикальной оси Y) или три значения (первое – для оси X, второе – для оси Y, третье – для оси Z).

*.card-content {*

*transform: translate(0, 100%); // сдвигаем текст карточки вниз за экран*

*transform: translateY(100%); // альтернативная запись*

*}*

*.card:hover .card-content {*

*transform: translate(0, 0); // убираем сдвиг текста карточки при наведении*

*transition: transform 0.5s 0.2s // добавляем плавности переходу*

*}*

* ***skew()*** – отвечает за наклон элемента. Она объединяет функции ***skewX()*** и ***skewY()***. В качестве аргумента может принимать одно (смещает верхнюю часть относительно нижней) или два значения (первое смещает верхнюю часть относительно нижней, второе – смещает левую сторону относительно правой).
* ***matrix()*** – позволяет записать любую 2D-трансформацию в виде матрицы. Для удобства рекомендуется использовать [генератор матриц](https://angrytools.com/css-generator/transform/).
* ***perspective()*** – отвечает за эмуляцию позиции наблюдателя относительно экрана (как далеко человек сидит от монитора). Она позволяет добавить 3D-объем элементам.

*perspective: 200px; // отдельная запись*

*transform: perspective(200px);*

Функции трансформации можно указывать последовательно в одном шорткате, но результат может отличаться в зависимости от порядка их следования. Чтобы избежать сложения функций в шорткате и их влияния друг на друга, рекомендовано записывать их как отдельные свойства.

*transform: rotate(15deg) scale(1.1) translateY(20%);*

*// альтернативная запись*

*rotate: 15deg;*

*scale: 1.1;*

*translateY: 20%;*

Свойство ***backface-visibility*** позволяет управлять видимостью задней стороны элемента.

*backface-visibility: hidden; // скрыли задник*

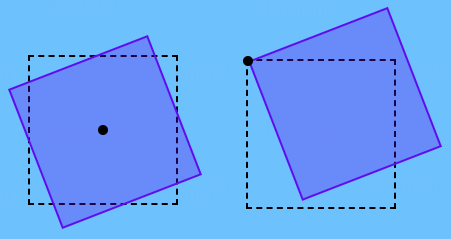
Свойство ***perspective-origin*** позволяет задать точку фокуса перспективы на экране (по умолчанию она в центре).

*perspective-origin: 100% 40px; // отступ слева 100% экрана и сверху 40px*

**Точка трансформации** – это координаты, вокруг которых происходит трансформация. По умолчанию, точка трансформации находится в центре элемента, а все координаты в вебе считаются от верхнего левого края.

Свойство ***transform-origin*** позволяет перемещать точку трансформации. В качестве аргумента может принимать до трех числовых значений или ключевых слов (*center*, *top*, *left*, *right*, *bottom*). Для оси Z нет ключевых слов.

Для примера, у правого куба задано свойство *transform-origin: 0 0;*



Свойства *opacity*, *transform* и отдельные функции трансформации, которые не меняют геометрию элемента и на находящихся рядом соседей и родителей, являются дешевыми, быстрыми в обработке и практически не нагружают браузер. В процессе анимации эти свойства затрагивают только слой с текущим элементом и не заставляют браузер перерисовывать всю страницу.

Если же этих свойств недостаточно, то для выделения элемента в отдельный слой используется свойство ***will-change***. В качестве значения оно принимает свойство, которое планируется изменять. По умолчанию используется значение *auto* и браузер сам пытается догадаться, нужно ли выносить элемент на отдельный слой.

*.link {*

*will-change: transform;*

*}*

*.link:hover {*

*transform: translateX(10px);*

*}*

Свойство ***transform-style*** позволяет объявить элемент как 2D или 3D. Для этого используются значения ***flat*** и ***preserve-3d*** соответственно.

Свойство ***translate3d*** работает аналогично *translate*, но принимает сразу три значения для всех осей.

*transform: translate3d(30px, 40px, -50px);*

Свойство ***scale3d*** работает аналогично *scale*, но принимает сразу три значения для всех осей.

*transform: scale3d(1.2, 1, 1.5);*

Свойство ***rotate3d*** работает аналогично *rotate*, но принимает сразу три значения для всех осей.

*transform: rotate3d(7, 10, 5, 45deg); // задали вектор и угол поворота*

Свойство ***matrix3d*** работает аналогично *matrix*, но принимает сразу 16 значений для матрицы 4\*4. Для удобства рекомендуется использовать [генератор матриц](https://angrytools.com/css-generator/transform/).

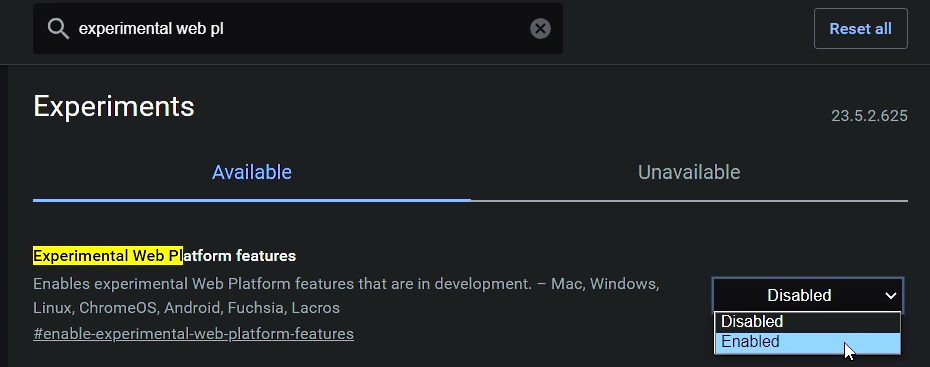
**Вендорные префиксы** – это приставки перед свойствами, селекторами, функциями и другими сущностями CSS, которые гарантируют, что экспериментальная версия не будет конфликтовать с итоговой реализацией или другими браузерами. Нельзя перечислять сущности с префиксами через запятую.

Префикс ***-webkit-*** используется для *Safari* (движок *WebKit*), *Chrome*, *Yandex Browser* и *Opera* (движок *Blink*).

Префикс ***-moz-*** используется для *Firefox* (движок *Quantum*).

Префикс ***-ms-*** используется для *Internet Explorer* и старого *Microsoft Edge 12-18* (движок *Trident*).

Включить поддержку экспериментальных свойств можно в настройках браузера.



Проверить поддержку CSS свойств можно на сайте <https://caniuse.com/>

Автоматизировать проверку необходимости в префиксе можно с помощью инструментов, задав целевые версии браузеров. Например, [Autoprefixer](https://github.com/postcss/autoprefixer) или [Browserlist](https://github.com/browserslist/browserslist).

Директива ***@supports*** проверяет поддержку браузером указанных свойств и селекторов перед их применением.

*@supports (color: red) {*

*color: red;*

*}*

**Кроссбраузерность** – это одинаковая работа и корректное отображение сайта во всех браузерах вне зависимости от платформы. Сейчас существует около 60 браузеров. Рекомендована поддержка браузеров с долей пользователей больше 2,5-5%.

Протестировать работу сайта в различных комбинациях “устройство + операционная система + браузер” можно на сервисах [BrowserStack](https://www.browserstack.com/) и [Lambdatest](https://www.lambdatest.com/).

Протестировать работу на мобильном устройстве можно также подключив его по USB к пк.