

JS – Дмитрий Валак

MDN

[Общий раздел по HTML](#) с руководствами и остальными ссылками

[Элементы HTML](#)

[Глобальные атрибуты](#)

[Руководство по HTML-формам](#)

Flex

[Использование flex-контейнеров CSS](#) (устарела)

[Основные понятия Flexbox](#) (новая)

Grid

[CSS Grid Layout](#)

Ссылки и ресурсы

Большая объяснялка от w3

<https://www.w3schools.com/>

Валидатор

<https://validator.w3.org/>

Сайт Дмитрия Валака

brainscloud.ru/

Все свойства CSS

html5book.ru/css-css3/

gbusokolgora@mail.ru

Хороший ресурс по свойствам CSS

flaviocopes.com

Спец. символы смотреть тут

w3schools.com

Тут можно посмотреть, какие теги и свойства поддерживаются различными браузерами и нужно ли писать префиксы:

caniuse.com

Сервис гугла со шрифтами

fonts.google.com

Ещё сайт со шрифтам

webfonts.pro

Здесь можно удалить фон у фотографий

<https://www.remove.bg/ru>

Здесь можно пожать и оптимизировать картинки на сайт

<https://tinypng.com/>

Иконки платные и бесплатные

<https://www.iconfinder.com/>

Ещё иконки

<https://fontawesome.com/>

Изображения-заглушки прямо по ссылке

<https://placeholder.com/>

Все подобные сервисы работают одинаково. Вы формируете url вида:

```

```

```

```

Подробнее в статье [тут](#).

Список свойств, с которыми можно использовать transition

flaviocopes.com

Слайдер Slick

[Slick](#)

При помощи CSS сгенерировать треугольник

[css-triangle-generator](#)

At-rules

Это то, что я называл медиазапросами. [MDN](#).

Added: new page 'text.html'

Вопросы вопросыки

Как убрать подчёркивание у ссылок?

С помощью text-decoration: none.

Как делать подчёркивание при наведении на ссылку?

Псевдокласс :hover + text-decoration: none

Как переносить строчные формы на новые строки?

Достаточно заключить их в контейнер div.

Работа с формами

Чтобы разделить элементы формы (ваше имя, e-мейл, номер телефона...), нужно заключить каждый из них в отдельный div и прописать margin. Тогда они станут блочными и отлипнут друг от друга. Ещё им прописывается display: block.

button обычно не заключается в отдельный div, как input.

Чтобы сделать строчный элемент блочным и растянуть его, можно использовать display: block; вместе с width.

С элементом формы select не работает padding, поэтому чтобы подогнать по размеру выпадающий список под другие элементы формы, надо увеличивать параметры width и height. Толщина границы border также не влияет на его размер, её тоже нужно суммировать при расчёте width и height.

Как сменить текст чекбокса (лейбл), когда он отмечен?

```
.checkbox:checked + .label {  
  color: black;  
}
```

Если в форме несколько инпутов

Если в форме есть несколько инпутов и им задана определённая ширина в макете, то не надо прописывать ширину каждому инпуту. Вместо этого ширину надо задать всей форме, а инпутам прописать 100%. Вместо указания высоты height для инпутов, им нужно указать padding.

```
.form {
  width: 100%;
  max-width: 280px;
}

.input {
  display: block;
  width: 100%;
  padding: 12px 18px;

  font-family: inherit;  потому что сам не унаследует
  font-size: 1rem;
}
```

Как поменять цвет текста placeholder для инпутов?

С помощью псевдокласса:

```
.input::placeholder {
  color: #fff;
}
```

Класс form-group и отступы между инпутами

Валак рекомендует оборачивать элементы формы в отдельный div с классом .form-group и уже этому классу задавать отступы:

```
<!--Форма в HTML-->
<form class="form" action="/" method="post">
  <div class="form-group">
    <input class="input" type="text" placeholder="Name">
  </div>
  <div class="form-group">
    <input class="input" type="email" placeholder="Email">
  </div>
  <div class="form-group">
    <input class="input" type="tel" placeholder="Phone">
  </div>
</form>
```

```
CSS
.form-group {
  margin-bottom: 20px;
}
```

Как «нарисовать» полосу под элементом (подчёркивание) используя after?

```
.header-nav-link.active {
  position: relative;
}

.header-nav-link.active::after {
  content: "";
  width: 100%;
  height: 2px;

  background-color: #fff;
}
```

```

position: absolute;
left: 0;
bottom: -5px;
}

.nav__link::after {
  content: "";
  display: block;
  width: 100%;
  height: 3px;

  background-color: #fce38a;

  position: absolute;
  top: 100%;
  left: 0;
  z-index: 1;
}

```

Ещё один способ без использования position:

```

.intro__title::after {
  content: "";
  display: block;
  width: 60px;
  height: 3px;
  margin: 0 auto; это выравнивание по середине. Поднимать и опускать тоже через margin
  background-color: #fff;
}

```

Что нужно сделать, чтобы header всегда висел вверху страницы при скроле?

```

width: 100%;
position: fixed;
top: 0;
left: 0;
right: 0;
z-index: 1000;

```

Что делать, если в диве лежит другой див и не даёт скруглить углы (перекрывает их)?

```

overflow: hidden;

```

Как можно позиционировать элементы?

Используя свойства position, transform и background: top 15px left 15px.

Как выровнять div по середине страницы?

С относительным позиционированием:

```

.block {
  width: 100px;
  position: relative;
  margin: 0 auto;
}

```

С абсолютным позиционированием:

```
.block {
  width:100px;
  position: absolute;
  margin:0 auto;
  left:0;
  right:0;
}
```

Если использовать margin: auto, то обязательно надо указать размер блока, или ничего не получится:

```
.block {
  width: 300px;
  margin: 200px auto;
}
```

С использованием позиционирования:

```
.block {
  position: absolute;
  right: 50%;
  top: 50%;
  transform: translateY(-50%) translateX(50%);
}
```

```
position: absolute;
right: 50%;
margin-right: -<ширина элемента / 2>
```

Если отступ указать в процентах, отсчёт будет вестись от верхней точки элемента, а не его центра. Поэтому если указать позиционирование 50%, то по середине будет не сам элемент, а только его верхняя точка.

Чтобы решить эту проблему, надо указать отрицательное значение margin-top, равное половине высоты элемента, он сдвинется вверх. Это один из способов решения проблемы «верхней точки».

Если элемент блочный и находится внутри flex:

```
div {
  display: flex;
  justify-content: center;
  align-items: center;
}
```

Как выровнять по середине один элемент внутри другого?

```
position: absolute;
right: 0;
top: 50%;
transform: translateY(-50%);
```

Как прижать элемент к низу контейнера?

```
margin-top: auto;
```

Как прижать элемент к стенке контейнера (без position)?

```
margin-left: auto;
```

Как добавить изображение на фон текста?

Через background no-repeat center

Какое основное правило при использовании тега section?

Внутри обязательно должен быть заголовок любого уровня.

Как выбрать все элементы списка или контейнера, кроме последнего? Например, чтобы применить к ним margin, но последнему отступ не нужен.

```
.list__item:not(:last-child) {  
    margin-bottom: 35px;  
}
```

Что сделать, чтобы элементы внутри flexbox переносились на новую строку при заполнении строки?

```
.div {  
    display: flex;  
    flex-wrap: wrap;  
}
```

Как сделать плавный переход для градиента?

Transition не работает с градиентом. Для плавного перехода можно сделать маску, которая по умолчанию прозрачная, а при наведении плавно становится непрозрачной.

(Вероятно, вместо непрозрачности маски можно использовать маску с градиентом, только с более тёмными цветами. Получается оригинальный градиент-фон будет исчезать, а новый – появляться).

```
.btn {  
    display: inline-block;  
    cursor: pointer;  
    background: #333;  
    position: relative;  
    overflow: hidden;  
}  
  
.btn::before {  
    content: "";  
    width: 100%;  
    height: 100%;  
    background-color: #000;  
    opacity: 0;  
    position: absolute;  
    top: 0;  
    left: 0;  
    z-index: 1;  
    transition: opacity .2s linear;  
}
```

В span берётся текст кнопки, чтобы он был над маской и маска его цвет не меняла

```
.btn span {  
    position: relative;  
    z-index: 2;  
}  
  
.btn:hover::before {  
    opacity: .1;  
}
```

Что сделать, чтобы элементы не реагировали на наведение курсора (например, не активировались ссылки)?

```
.text:hover .img {
    visibility: visible;
}
```

#? Как это работает?

В примере ниже при наведении на текст будет показываться картинка:

```
.text:hover .img {
    visibility: visible;
}
```

Неизвестные свойства

align-content

align-items

Ошибки

- Для intro прописал высоту в пикселях, а не 100vh
- ссылкам в навигации надо прописывать display: inline-block



При вёрстке мого я вручную двигал h1 и h2 заголовки. Вместо этого надо использовать флекс для всей секции целиком:

```
.intro {
    display: flex;
    flex-direction: column;
    justify-content: center;
    width: 100%;
    height: 100vh;
    background: url('../img/intro-bg.jpg') center no-repeat;
    background-size: cover;
}
```

Кнопкам слайдера в Intro вручную прописывал размер. Вместо этого можно указать width: 23%

Эти кнопки прижимаются к низу секции так:

```
<!-- Intro -->
<div class="intro">
  <div class="container">
    <div class="intro__inner">
      <h2 class="intro__suptitle">Creative Template</h2>
      <h1 class="intro__title">Welcome to Mogo</h1>

      <a class="btn" href="#">Learn more</a>
    </div>
  </div>

  <div class="slider">
    <div class="container">
      <div class="slider__inner">
        <div class="slider__item active"><span class="slider__num">01</span> intro</div>
        <div class="slider__item"><span class="slider__num">02</span> work</div>
```

```

        <div class="slider__item"><span class="slider__num">03</span> about</div>
        <div class="slider__item"><span class="slider__num">04</span> contacts</div>
    </div>
</div>
</div>
</div>

.intro {
    display: flex;
    flex-direction: column;
    justify-content: center;
    width: 100%;
    height: 100vh;
    background: url('../img/intro-bg.jpg') center no-repeat;
    background-size: cover;
}

.slider {
    width: 100%;
    position: absolute;
    bottom: 0;
    left: 0;
    z-index: 1;
}

.slider__inner {
    display: flex;
    justify-content: space-between;
}

```

Секции часто повторяются, текст в них расположен одинаково. Вместо того каждый раз снова создавать каждую секцию, надо сделать общий класс и его вставлять везде.

Валак часто вместо фиксированных размеров элементов на странице (типа плитки с фотками) использует процентный размер и помещает это во флекс.

Основы

HTML (HyperText Markup Language) - язык гипертекстовой разметки

- Браузер читает разметку и показывает понятный для людей вид страниц
- HTML файлы имеют расширение **.html** или **.htm**
- Разметка состоит из набора **тегов и их атрибутов**
- Набор тегов создает иерархическое дерево. Теги можно вкладывать друг в друга, создаётся дерево.
- HTML файл читается браузером сверху вниз

Теги

Виды тегов: одинарные и парные

Одинарный тег: <название тега>

Парный тег: <название тега>...</название тега>

Теги можно вкладывать друг в друга

Основные типы тегов:

Блочные - занимают всю ширину и начинаются с новой строки

Строчные - занимают только то пространство, которое им необходимо и находятся на одной строке, пока не закончится пространство.

Атрибуты — это дополнительные параметры для тегов.

Пример: <тег атрибут= "..." >...</тег>

Существуют теги с обязательными атрибутами. Например, путь к изображению.

Глобальные атрибуты

[Global attributes](#) на mdn

Глобальные атрибуты — это атрибуты общие для всех HTML элементов.

Они могут быть указаны для любых [элементов HTML](#), даже для тех, которые не указаны в стандарте. Это значит, что все нестандартные элементы должны допускать эти атрибуты.

В дополнение к основным для HTML, также существуют следующие глобальные атрибуты:

1. Атрибуты обработчиков событий «on...»: onclick, ondrag и другие.
2. xml:lang и xml:base, которые унаследованы от XHTML и сохранены в целях совместимости.
3. [aria-*](#) атрибуты, используемые для улучшения доступности.

Перечень **основных** глобальных атрибутов:

accesskey	даёт подсказку для создания комбинации клавиш для текущего элемента.
class	список разделённых пробелами классов элемента.
contenteditable	нужно ли предоставить пользователю возможность редактировать элемент.
contextmenu	id элемента <code><menu></code> , который следует использовать в качестве контекстного меню
data-*	атрибуты пользовательских данных для обмена инфо между HTML и DOM

ЧаВо

Стандартная CSS стилизация для body

Можно сделать документацию небольшую и пронумеровать стили, если стилей много

```
/*
  1 - header
  2 - content
  3 - article
  4 - footer
*/

body {
  margin: 0;

  font-family: Arial, sans-serif;
  font-size: 15px;
  color: #121212;

  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

*,
*::before,
*::after {
  box-sizing: border-box;
}

h1, h2, h3, h4, h5, h6 {
  margin: 0;
}

ul, ol {
  margin: 0;
```

```

padding: 0;
list-style: none;
}

.img {
display: block;
max-width: 100%;
height: auto;
}

.container {
width: 100%;
max-width: 1230px;
margin: 0 auto;
padding: 0 15px;
background-color: #eee;
}

<a class="btn" href="#">Кнопка</a>
.btn {
display: inline-block;
padding: 6px 12px;

background-color: #36b717;

font-size: 15px;
color: #fff;
text-decoration: none;
}

```

```

группировка {
/* Отступы и размеры */
width: 400px;
max-width: 500px;
padding: 1rem;
margin: 20px auto;

/* шрифты */
font-family: Arial, sans-serif;
font-size: 1rem;
color: #121212;

/* внешнее оформление */
background-color: #eee;
box-shadow: 0 0 10px rgba(0, 0, 0, .5);

/* позиционирование */
position: absolute;
top: 0;
left: 0;
z-index: 1;

/* плавный переход отдельной группой */
transition: opacity 1s linear;
}

```

Выравнивание по центру

line-height

Позволяет выравнивать текст. Надо установить высоту строки (line-height) равной высоте height элемента, в котором она находится.

```
<div class="div">
  <p>text</p>
</div>

.div {
  width: 200px;
  height: 200px;
  background-color: pink;
  line-height: 200px;
  text-align: center;
}
```

С изображением такая тема не прокатывает. Чтобы прокатывало, надо дописать одно свойство для изображения. Рядом с изображением можно писать текст, всё будет ок:

```
<div class="div">
  
</div>

.div {
  width: 200px;
  height: 200px;
  background-color: pink;
  text-align: center;
  line-height: 200px;
}

.div img {
  vertical-align: middle;
}
```



position

Этот вариант надо использовать либо вместе с margin (с точными значениями смещения в px или в %, что неудобно), либо с transform.

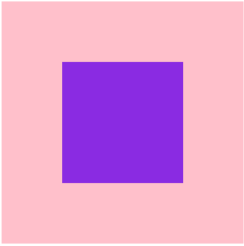
```
<div class="outer">
  <div class="inner"></div>
</div>

.outer {
  width: 200px;
  height: 200px;
  background-color: pink;
  position: relative;
}

.inner {
  width: 100px;
  height: 100px;
  background-color: blueviolet;
  position: absolute;
```

```
top: 50%;
left: 50%;

margin: -25% -25%;
transform: translate3d(-50%, -50%, 0);
}
```



table

Элементы надо превратить в таблицу через свойства «display: table» для родительского и «display: table-cell» для дочернего элемента. Я не буду прописывать пример целиком, потому что дочерний элемент просто растягивается на 100% родительского и перекрывает его.

flex

Самый простой и понятный способ. В родительский элемент надо прописать свойства «display» «justify-content» и «align-items»

```
<div class="outer">
  <div class="inner"></div>
</div>

.outer {
  width: 200px;
  height: 200px;
  background-color: pink;

  display: flex;
  justify-content: center;
  align-items: center;
}

.inner {
  width: 100px;
  height: 100px;
  background-color: blueviolet;
}
```

margin

Так можно выравнивать по горизонтали только блочный элемент (display: block), у которого явно задана ширина (иначе он растянется на 100% и отступать будет не от чего).

По высоте выравнивание только через margin не работает.

```
<div class="outer">
  <div class="inner"></div>
</div>

.outer {
  width: 200px;
  height: 200px;
  background-color: pink;
}
```

```
.inner {
  width: 100px;
  height: 100px;
  background-color: blueviolet;
  margin: 0 auto;
}
```

Можно использовать margin вместе с flex, тогда сработает выравнивание и по вертикали, и по горизонтали:

```
.outer {
  ...
  display: flex;
}

.inner {
  ...
  margin: auto;
}
```

Мой пример с блоками

```
JS
let parent = document.body.querySelector('.parent');

parent.addEventListener('click', (ev) => {
  let target = ev.target.closest('.square');

  if (!target) {
    return;
  }
  if (!target.classList.contains('square')) {
    return;
  }
  target.style.backgroundColor = 'red';
});
```

```
HTML
<div class="parent">
  <div class="square first">
    <div class="inner">first</div>
  </div>
  <div class="square second">
    <div class="inner">second</div>
  </div>
</div>
```

```
CSS
* {
  box-sizing: border-box;
}

.parent {
  width: 400px;
  height: 200px;
  background-color: pink;
  display: flex;
  justify-content: space-evenly;
  align-items: center;
}

.first, .second {
  display: flex;
  width: 150px;
  height: 150px;
```

```
background-color: plum;
justify-content: center;
align-items: center;
}

.inner {
display: flex;
width: 100px;
height: 30px;
justify-content: center;
background-color: yellow;
align-items: center;

font-family: monospace;
font-size: 20px;
text-transform: uppercase;
}
```

HTML

HTML basics

Базовая структура HTML документа

Главный документ должен называться index.html и лежать в корне проекта, а не во внутренней папке.

DOCTYPE – декларация типа документа
html – основной контейнер
head – мета информация, не рендерится
body – вся разметка. То, что видно на странице.

lang – указать актуальный язык
charset – кодировка страницы
link – подключение файла с чем-то
title – заголовок страницы

Скелет страницы:

```
<!DOCTYPE html>

<html lang="en">

  <head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="style.css">
    <title>Test</title>
  </head>

  <body>
    <div class="container"></div>
    <script src="index.js"></script>
  </body>

</html>
```

Мета-данные и внешние ресурсы

`<link>`

Ссылка на внешний ресурс. Чаще всего – для ссылки на stylesheets и для создания иконок.

href

путь до ресурса

rel

relationship, сообщает как указанный элемент связан с содержащим его документом. В справочнике «*типы ссылок*» есть много различных видов отношений. Распространённые типы ссылок:

rel="stylesheet"

rel="icon"

rel="preload"

type

используется для определения типа связываемого контента. Значение атрибута должно быть типом MIME, такое как text/html, text/css и т.д.

подключение шрифтов

от google

<meta>

charset

атрибут задаёт кодировку символов. Рекомендуется использовать UTF-8.

name

Используется для SEO. Описание, тип данных, ключевые слова. Например:

description

keywords

author

content

Описывается значение данных из атрибута name. Для keywords указываются через запятую.

viewport

даёт подсказки о размере изначального размера viewport. Используется для адаптивной вёрстки.

<style>

Лучше не использовать

Примеры:

Meta

```
<meta charset="UTF-8">
```

SEO

```
<meta name="description" content="Test web page">
```

```
<meta name="keywords" content="html, webdev, keyword">
```

```
<meta name="author" content="Glad Valakas">
```

Additional approach to add favicon

```
<link rel="apple-touch-icon" href="apple-touch-icon.png">
```

```
<link rel="apple-touch-icon" sizes="72x72" href="apple-touch-icon-72x72.png">
```

```
<link rel="apple-touch-icon" sizes="114x114" href="apple-touch-icon-114x114.png">
```

Use latest version of IE

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

Адаптивная и мобильная вёрстка

```
<meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=yes">
```

name="viewport"

Это одно из многих значений мета-информации.

```
content = "width=device-width"
```

Ширина сайта равняется ширине экрана устройства, с которого он открыт.

```
content = "initial-scale=1"
```

Говорим браузеру, чтобы по умолчанию он не увеличивал (не зумил) страницу.

Описание сайта и картинка при копировании ссылки

<https://ruogp.me/>

<https://pr-cy.ru/news/p/5407-open-graph-cto-eto-i-pochemu-kazhdyy-dolzhen-nastroit>

Это description:

🏠 **Анша Абдуль** – сакральное приветствие адептов.

📍 [memepedia.ru](#) > [abduloverovanie-religion/](#) ▼

Такие, как "Анша Абдуль" (сакральное приветствие), "Кара вечная" (удел всех грешников), "маслята" (грешники и неугодные) и т.д. В основном же искусство Абдуловерования сосредоточено на образе самого Абдулова. [Читать ещё](#) >

HTML структура сайта и семантические теги

Это логические блочные элементы, которые заменяют div с соответствующим классом, например «header» вместо «div class="header"», чтобы поисковые роботы могли лучше ориентироваться.

В официальной документации можно посмотреть эти теги [здесь](#).

Теги расположения

<header>	шапка
<main>	основное содержимое страницы
<footer>	подвал сайта
<aside>	боковая панель, sidebar

Теги содержания

<nav>	навигация по сайту
<content>	
<section>	в секции д.б. заголовок любого уровня
<article>	статья, запись, новость. д.б. заголовок любого уровня

```
<time datetime="ггг-мм-дд чч:мм"> </time>
```

```
<address> контактная информация
```




Валидация, семантика, доступность

Валидация – это проверка документа специальной программой на соответствие установленным веб-стандартам и для обнаружения ошибок. Эти стандарты называются спецификациями, разработанными консорциумом World Wide Web (w3c).

Сначала определяется тип документа, который указывается в doctype.

Затем проверяется html код на правильность и отсутствие ошибок, в т.ч. правильность имён тегов и их вложенности.

Валидный HTML-код, валидная разметка — это HTML-код, который написан в соответствии с определёнными стандартами. Валидатор от W3C [тут](#).

Основные 4 типа проверки:

1. Проверка синтаксиса
2. вложенности тегов
3. DTD – document type definition
4. проверка на наличие посторонних элементов или тегов.

Семантика – использование правильных тегов, описывающих содержимое контента внутри себя. Семантический тег – это тег, который носит смысловое объяснение. По тегу должно быть понятно, какой контент внутри него находится.

Правило для определения <article>, <section> и <div>:

Можете дать имя разделу и вынести этот раздел на другой сайт? — <article>

Можете дать имя разделу, но вынести на другой сайт не можете? — <section>

Не можете дать имя? Получается что-то наподобие «новости и фотогалерея» или «правая колонка»? — <div>

<article>

- Значение: независимая, отделяемая смысловая единица, например комментарий, твит, статья, виджет ВК и так далее.
- Особенности: желателен заголовок внутри.
- Типовые ошибки: путают с тегами <section> и <div>.

<section>

- Значение: смысловой раздел документа. Неотделяемый, в отличие от <article>.
- Нужен заголовок внутри.
- Типовые ошибки: путают с тегами <article> и <div>.

<aside>

- Значение: побочный, косвенный для страницы контент.
- Особенности: может иметь свой заголовок. Может встречаться несколько раз на странице.
- Типовые ошибки: считать <aside> тегом для «боковой панели» и размечать этим тегом основной контент, который связан с окружающими его элементами.

<nav>

- Значение: навигационный раздел со ссылками на другие страницы или другие части страниц.
- Особенности: используется для основной навигации, а не для всех групп ссылок. Основной является навигация или нет — на усмотрение верстальщика. Например, меню в подвале сайта можно не оборачивать в <nav>. В подвале обычно появляется краткий список ссылок (например, ссылка на главную, копирайт и условия) — это не является основной навигацией, семантически для такой информации предназначен <footer> сам по себе.
- Типовые ошибки: многие считают, что в <nav> может быть только список навигационных ссылок, но [согласно спецификации](#) там может быть навигация в любой форме.

<header>

- Значение: вводная часть смыслового раздела или всего сайта, обычно содержит подсказки и навигацию. Чаще всего повторяется на всех страницах сайта.
- Особенности: этих элементов может быть несколько на странице.
- Типовые ошибки: использовать только как шапку сайта.

<main>

- Значение: основное, не повторяющееся на других страницах, содержание страницы.
- Особенности: должен быть один на странице, исходя из определения.
- Типовые ошибки: включать в этот тег то, что повторяется на других страницах (навигацию, копирайты и так далее).

<footer>

- Значение: заключительная часть смыслового раздела или всего сайта, обычно содержит информацию об авторах, список литературы, копирайт и так далее. Чаще всего повторяется на всех страницах сайта.
- Особенности: этих элементов может быть несколько на странице. Тег <footer> не обязан находиться в конце раздела.
- Типовые ошибки: использовать только как подвал сайта.

Глобальные атрибуты

Это атрибуты, которые можно применить ко всем тегам.

Глобальные атрибуты

class

список разделённых пробелами классов элемента.

id

Уникальный идентификатор элемента.

data-*

пользовательские данные

contenteditable

Булевый, нужно ли предоставить пользователю возможность редактировать элемент.

spellcheck

Булевый. Может ли содержимое элемента быть проверено на наличие орфографических ошибок.

title

Содержит текст, предоставляющий консультативную информацию об элементе. Показываться пользователю в виде всплывающей подсказки.

dir

направление текста в элементе

lang

Участвует в определении языка элемента, языка написания нередатируемых элементов или языка, на котором должны быть написаны редактируемые элементы. Например, разные кавычки в цитатах в немецком и английском языках.

hidden

скрыть элемент со страницы. В DOM он остаётся.

style

инлайновые стили

tabindex

может ли элемент получать фокус, участвует ли он в последовательной навигации с клавиатуры, и если да, то в какой позиции. Значения:

отрицательное число – означает, что элемент фокусируемый, но он не может получить фокус посредством последовательной навигации с клавиатуры.

0 – означает, что элемент фокусируемый и может получить фокус посредством последовательной навигации с клавиатуры, но порядок его следования определяется платформой.

положительное значение – означает, что элемент фокусируемый и может получить фокус посредством последовательной навигации с клавиатуры. Порядок его следования определяется значением атрибута – последовательно возрастающего числа tabindex. В случае, когда несколько элементов имеют одинаковое значение атрибута tabindex, порядок их следования при навигации определяется их местом в документе.

Атрибуты обработчиков событий

onabort, onautocomplete, onautocompleteerror, onblur, oncancel, oncanplay, oncanplaythrough, onchange, onclick, onclose, oncontextmenu, oncuechange, ondblclick, ondrag, ondragend, ondragenter, ondragexit, ondragleave, ondragover, ondragstart, ondrop, ondurationchange, onemptied, onended, onerror, onfocus, oninput, oninvalid, onkeydown, onkeypress, onkeyup, onload, onloadeddata, onloadedmetadata, onloadstart, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onmousewheel, onpause, onplay, onplaying, onprogress, onratechange, onreset, onresize, onscroll, onseeked, onseeking, onselect, onshow, onsort, onstalled, onsubmit, onsuspend, ontimeupdate, ontoggle, onvolumechange, onwaiting

div, span

Текст

У текста больше всего тегов. Тут будут самые основные.

```
<h1> Заголовок самый важный </h1>
<h6> Заголовок менее важный </h6>

<p> параграф с большим куском текста </p>

<hr> горизонтальная линия
<br> перенос строки

<!-- Bold & Italic, Typing, Deleted text -->
<b>Bold text</b>
<i>Italic text</i>
<tt>Typing text</tt>
<s>Deleted text</s>
```

```
<u>подчёркнутый текст</u>

<!-- С семантическим акцентом -->
<strong>Bold accent text</strong>
<em>Italic accent text</em>
<s>Deleted text</s>
<ins>Inserted</ins> <!-- нижнее подчёркивание -->

<small>Small</small>
<big>Big</big>

<!-- Quote, Block of Quote -->
<q> цитата </q>
<blockquote>
  <p>
    Очень большая цитата
    С сохранением семантики, потому что есть <p>
  </p>
</blockquote>

<!-- инфо об авторе -->
<address> Glad Valakas, youtube maker </address>

<!-- Time or date information -->
<time>1989-01-26</time>
<time datetime="1989-01-26">My Birthday</time>

<!-- Моноширинный и преформатированный -->
<code> const getDate = () => new Date </code>
<pre> преформатированный текст </pre>

<!-- Subscripted и Superscripted -->
<sub>Subscripted</sub>
<sup>Superscripted</sup>
```

kbd

Обозначение горячих клавиш, название символов клавиатуры. Имеет оформление по умолчанию используя шрифт monospace.

Горячие клавиши <kbd> ctrl+f </kbd>

abbr

Аббревиатура с возможностью показать её расшифровку

Визуально ничего не меняется, это нужно для поисковых роботов. Поэтому рекомендуется указывать атрибут title и записать в него расшифровку. В этом случае появятся внешние изменения: текст будет подчёркнутым, при наведении появится всплывающее окно с расшифровкой из title.

[<abbr>](#)

You can use <abbr title="Cascading Style Sheets">CSS</abbr> to style HTML

cite

Источник цитаты. Можно использовать вкупе с blockquote.

dfn

Definition. Выделение определения или термина, которое встречается впервые на странице. Имеет оформление курсивом по умолчанию.

mark

выделить текст с помощью жёлтой подсветки

Списки

Элементы

ul не нумерованный
ol нумерованный
li элемент списка

Атрибуты

type="" тип маркера списка
start="" с какого числа начинается нумерация
reversed нумерация в обратном порядке

Вложенный список

```
<ol>
  <li>элемент</li>
  <li>элемент
    <ul>
      <li>элемент 2</li>
    </ul>
  </li>
</ol>
```

Список определений

dl Description List, список описаний
dt Description Term, термин
dd Description Details, определение термина

```
<dl>
  <dt>реторсия</dt>
  <dd>ответные ограничительные меры</dd>

  <dt>залог</dt>
  <dd>способ обеспечения обязательств</dd>
</dl>
```

Картинки

Простая

```
<img>
src=""      путь
alt=""      обязательный атрибут

title=""    всплывёт надпись при наведении на изображение (глобальный атрибут)

max-width=""
max-height=""
```

Картинка с подписью

так описание будет котироваться поисковыми роботами
<figure>

Контейнер, внутри которого `` и подпись к картинке в `<figcaption>`.
Обычно `<figure>` это рисунок, иллюстрация, диаграмма, фрагмент кода.

`<figcaption>`

Подпись

```
<figure>
  
  
  <figcaption>Описание для изображения</figcaption>
</figure>
```

Контейнер `picture`

`<picture>`

Контейнер для одного или более элементов `<source>` и одного элемента `` для различных размеров экрана.

```
<picture>
  <source media="(min-width: 1024px)" srcset="https://picsum.photos/600/600">
  <source media="(min-width: 360px)" srcset="https://picsum.photos/100/100">
  
</picture>
```

`<source>`

указывает несколько медиа-ресурсов для элементов `<picture>`, `<video>` и `<audio>`. Он обычно используется для обслуживания одного и того же медиа-контента в нескольких форматах, поддерживаемых различными браузерами.

`media=""` Определяет медиавыражение , согласно которому будет выводиться изображение.

Резиновая картинка

```
.img {
  max-width: 100%;
  height: auto;
  display: block;
}
```

Таблица

Вся стилизация таблицы перенесена на сторону CSS.

table	контейнер
tr	table row, строка
td	table data, ячейка

Объединение ячеек по вертикали и горизонтали

```
<td colspan="2">
<td rowspan="2">
```

```
<table>
  <tr>
    <td>Front-end</td>
  </tr>
</table>
```

`<caption>`

определяет название (заголовок) таблицы. Он всегда должен быть первым вложенным элементом тега `<table>`. CSS `caption-side` и `text-align`.

<thead>

defines a set of rows defining the head of the columns of the table. Текст будет выравниваться по центру.

Внутри используются:

```
<tr>
<th>
```

<tbody>

Внутри используются:

```
<tr>
<td>
```

<tfoot>

defines a set of rows summarizing the columns of the table

Внутри используются:

```
<tr>
<th>
```

Ссылки

<a>

anchor, ссылка для перемещения на другую страницу или якорь к элементу на текущей странице.

href=""

href="#" ссылка-якорь, указывается id элемента, к которому надо промотать

target="_self" открыть в текущей вкладке

target="_blank" открыть в новой вкладке

target="_parent" загружает документ в родительской вкладке.

target="_top" загружает документ в окне высшего уровня.

download скачать файл, а не открыть в браузере

href="tel:+37123455234"

href="mailto:info@web.com"

href="skype:info@web.com"

Ссылка-изображение

```
<a href="#">
```

```
  <img src="" alt="">
```

```
</a>
```

Кнопка

<button>

```
<button name="button">Тык!</button>
```

HTMLButtonElement dom

type

type="button"

не имеет поведения по умолчанию, надо самому повесить скрипты.

type="submit"

отправить данные формы на сервер. Это значение по умолчанию, в т.ч. когда значения нет или значение неправильное.

```
type="reset"
очистить форму?
```

```
type="menu"
открывает всплывающее меню defined via its designated <menu> element.
```

Атрибуты

form	указать id формы, с которой связана кнопка
name	имя кнопки, которое отправляется с формой
value	начальное значение кнопки
autofocus	автоматическая фокусировка например, на крестике «закрыть» в подсказке
disabled	заблокировать нажатие на кнопку

Главным преимуществом HTML-элемента [<button>](#) в сравнении с элементом [<input>](#) в том, что [<input>](#) может принимать только простой текст, а [<button>](#) позволяет использовать весь HTML для создания более стилизованного текста внутри.

Специальные символы

Это символы рубля, евро, математические символы и другие, которых нет на клавиатуре.

Чтобы их использовать, надо поставить амперсент «&» и ввести код символа или сущность.

Сущность – это то, что прописывается словами, а не кодом: &euro. Сущность есть не у всех символов, поэтому какие-то символы могут быть представлены только кодом.

В конце ставится «;»

Коды специальных символов и сущности надо искать в справочниках. Например, тут: w3schools.com

```
<p> 260.00 &euro;</p>
```

```
€ &euro;
© &copy;
₽ &#8381;
```

Видео и аудио

Видео

Видео элемент может содержать один или несколько источников видео. Чтобы указать источник видео, необходимо использовать атрибут **src** или элемент `<source>`.

Можно указывать несколько источников с разными форматами, браузер сам выберет тот, который поддерживает. Самые универсальные – mp3 и mp4.

[<video>](#)

[HTMLVideoElement](#)

```
<video src="https://demo.mp4" controls width="700"></video>
```

```
<video width="320" height="240" poster="">
  <source src="example.ogg" type="video/ogg">
  <source src="example.mp4" type="video/mp4">
  Your browser does not support the video tag.
</video>
```

src

Адрес видео. Вместо этого атрибута, внутрь можно поместить элемент `source`.

type

тип файла. Это атрибут элемента `source`, а не `video`.

poster

URL-адрес обложки

height**width**

Высота и ширина области отображения видео в пикселях. Если не указать размер видео, то оно развернётся в соответствии с собственным размером.

controls

Если этот атрибут присутствует, тогда браузер отобразит элементы управления.

autoplay

Логический атрибут. Если указан, то видео начнёт воспроизводиться автоматически.

loop

Логический. Зациклить видео.

muted

Логический, аудио воспроизводиться не будет.

preload

`none`: видео не будет предварительно загружаться.

`metadata`: предварительно загружаются только метаданные видео (например, длина).

`auto`: видео сразу загружается.

пустая строка: синоним `auto`.

Аудио

Используется для встраивания звукового контента в документ. Может содержать один или более источников аудио, браузер выберет один наиболее подходящий.

[`<audio>`](#)

[HTMLMediaElement](#)

```
<audio controls>
  <source src="example.mp3" type="audio/mpeg">
  Your browser does not support the audio tag.
</audio>
```

src

Адрес аудио. Вместо этого атрибута, внутрь можно поместить элемент `source`.

type

тип файла. Нет на странице с видео такого атрибута.

controls

Если этот атрибут присутствует, тогда браузер отобразит элементы управления.

autoplay

Логический атрибут. Если указан, то аудио начнёт воспроизводиться автоматически.

loop

Логический. Зациклить видео.

muted

Логический, аудио воспроизводиться не будет.

preload

`none`: не будет предварительно загружаться.

`metadata`: предварительно загружаются только метаданные (например, длина).

auto: аудио сразу загружается.
пустая строка: синоним auto.

crossorigin

Этот атрибут указывает, следует ли использовать CORS при загрузке мультимедиа

source

указывает несколько медиа-ресурсов для элементов [picture](#), [video](#) и [audio](#). Это пустой элемент. Он обычно используется для обслуживания одного и того же медиа-контента в нескольких форматах, поддерживаемых различными браузерами.

[<source>](#)

[HTMLSourceElement](#)

Создание источника для элементов audio, video и picture. Браузер подберет более подходящий.

```
<video width="720" controls poster="/files/poster.jpg">  
  <source src="./files/video.mp4" type="video/mp4">  
</video>
```

```
<audio controls>  
  <source src="files/file.mp3" type="audio/mpeg">  
  <source src="files/file.ogg" type="audio/ogg">  
</audio>
```

src

адрес медиа-ресурсов. Значение этого атрибута игнорируется, когда `<source>` размещён внутри `<picture>`.

type

MIME-тип ресурса, опционально содержащий параметр codecs.

Если атрибут type не указан, то он запрашивается с сервера и проверяется, может ли user agent его обрабатывать. Если он не может быть обработан, проверяется следующий `<source>`

Более подробную информацию о типах файлов можно почитать на сайте [iana.org](#).

srcset

хзч. Внутри тега `picture`. Список из одного или нескольких изображений, из которых браузер подберет наиболее подходящее. Указывается путь к файлу, а так же один из двух дескрипторов: w или x.

sizes

Список размеров изображений для разных размеров страниц. Он состоит из разделённых запятыми медиавыражений со значениями ширины изображения.

media

Определяет медиавыражение, согласно которому будет выводиться изображение. Работает только в `<picture>`.

`</>`

Разное

Условные комментарии

Браузер IE поддерживает специальную технологию определения версии под названием «условные комментарии». Синтаксис их применения следующий.

```
<!--[if условие]> невидимый HTML-код <![endif]-->  
<![if условие]> видимый HTML-код <![endif]>
```

canvas

Создаёт область для рисования с помощью JS. Какая-то мутная тема.

[canvas](#)

[Canvas API](#)

time

Служит для выделения даты и времени. Позволяет выводить понятную дату для людей, а так же для поисковых роботов благодаря атрибутам. Используется, например, для даты публикации новостей, комментариев и тд.

Опубликовано: <time datetime="2019-04-29 19:00">3 дня назад</time>

datetime="yyyy-mm-dd hh:mm"

Это атрибут для машин в строгом формате

address

Указание контактной информации. По умолчанию выделяется курсивом.

```
<address>
  <p>Пишите нам на email: info@company.ru</p>
</address>
```

HTML Форма

Элементы для создания форм

[руководство по формам](#)
[хороший перечень](#)

<form>	атрибуты action и method обязательны
<button>	
<fieldset>	группировка нескольких элементов формы
<legend>	заголовок, вставленный в рамку формы или fieldset
<label>	метка для элемента формы напротив инпут-полей
<textarea>	
<select>	содержит меню опций внутри
<optgroup>	группирует опции внутри элемента select
<datalist>	работает как т9 с options внутри
<option>	пункт для <select>, <optgroup> или <datalist>
<output>	элемент вывода чего-либо
<progress>	
<meter>	
<input>	для получения данных от пользователя
type	по умолчанию text. Подробно в этой таблице
text	однострочное текстовое поле

password	текстовое поле со скрытыми символами
number	только числовые значения. Надо для мобильных устройств.
email	для редактирования электронной почты, надо для мобильных устройств.
tel	ввод номера телефона. Надо для мобильных устройств.
submit	Кнопка для отправления формы
reset	Кнопка сброса формы
image	Кнопка отправки формы в виде изображения
checkbox	Флаг, выбор несколько опций. Использовать с label
radio	Можно выбрать только одно значение из нескольких. Группу через общее имя
file	выбрать и прикрепить файл к форме
date	Элемент для ввода даты: год, месяц и день.
datetime	Элемент для ввода даты и времени: час, минута, секунда и доля секунды.
range	Полоса с бегунком для установки значения.
color	интерфейс выбора цвета (в текстовом значении)

form

Контейнер, элементы и поля в котором обрабатываются как единое целое. Это значит, что все данные из всех полей обрабатываются и отправляются одним запросом.

Представляет (собой) раздел документа, содержащий интерактивные элементы управления, которые позволяют пользователю отправлять информацию на веб-сервер.

`<form>`

[HTMLFormElement](#)

action

URI-адрес программы, которая обрабатывает информацию переданную через форму. Это значение может быть переписано с помощью атрибута `formaction` на `<button>` или `<input>` элементе.

method

HTTP метод, который браузер использует, для отправки формы. Возможные значения:

`post`: данные из формы включаются в тело формы и посылаются на сервер.

`get`: данные из формы добавляются к URI атрибута `action`, их разделяет '?', и полученный URI посылается на сервер.

name

Имя формы. Оно должно быть уникальным и не пустым.

autocomplete

Могут ли элементы управления автоматически быть дописаны браузером. Браузер может автоматически дополнить значения, основанные на значениях, которые пользователь уже вводил, в течение предыдущего использования формы. Возможные значения: `off` и `on`.

novalidate

Отключение проверки введенных значений. По умолчанию эта проверка есть.

button

`<button>`

`<button name="button">Тык!</button>`

[HTMLButtonElement](#) `dom`

type

`type="button"`

не имеет поведения по умолчанию, надо самому повесить скрипты.

`type="submit"`

отправить данные формы на сервер. Это значение по умолчанию, в т.ч. когда значения нет или значение неправильное.

`type="reset"`

очистить форму?

`type="menu"`

открывает всплывающее меню defined via its designated `<menu>` element.

Атрибуты

<code>form</code>	указать id формы, с которой связана кнопка
<code>name</code>	имя кнопки, которое отправляется с формой
<code>value</code>	начальное значение кнопки
<code>autofocus</code>	автоматическая фокусировка например, на крестике «закрыть» в подсказке
<code>disabled</code>	заблокировать нажатие на кнопку

Главным преимуществом HTML-элемента [<button>](#) в сравнении с элементом [<input>](#) в том, что [<input>](#) может принимать только простой текст, а [<button>](#) позволяет использовать весь HTML для создания более стилизованного текста внутри.

[datalist т9](#)

Когда будешь вводить что-то в инпут, будут подставляться готовые варианты из опций, как в т9.

Содержит набор опций `<option>`, доступных для выбора. Выбранное значение будет установлено для элемента `<input>` с атрибутом `list`.

[<datalist>](#)

[HTMLDataListElement](#)

```
<input id="input_id" list="datalist_id" name="capital">
```

```
<datalist id="datalist_id">
  <option>moscow</option>
  <option>london</option>
  <option>minsk</option>
</datalist>
```

[legend](#)

Заголовок, вставленный в рамку формы или fieldset.

[<legend>](#)

[HTMLLegendElement](#)

```
<form>
  <fieldset>
    <legend>Favorite monster</legend>
    <p>Kraken</p>
    <p>Sasquatch</p>
  </fieldset>
</form>
```

fieldset

Используется для группировки нескольких элементов формы. Это прямоугольная обводка группы элементов, чтобы их было удобнее видеть.

Используется вместе с legend.

[<fieldset>](#)

[HTMLFieldSetElement](#)

```
<form>
  <fieldset>
    <legend>Favorite monster</legend>
    <p>Kraken</p>
    <p>Sasquatch</p>
  </fieldset>
</form>
```

optgroup

Группирует опции, находящиеся внутри элемента <select>.

Используется с <option>

[<optgroup>](#)

[HTMLOptGroupElement](#)

```
<select>
  <optgroup label="Группа 1">
    <option>Опция 1.1</option>
  </optgroup>

  <optgroup label="Группа 2">
    <option>Опция 2.1</option>
  </optgroup>

  <optgroup label="Группа 3" disabled>
    <option>Опция 3.1</option>
  </optgroup>
</select>
```

label

Имя группы, которое будет отображено браузером в выпадающем списке. Этот атрибут обязателен.

disabled

работает на всю группу

option

используется для определения пункта списка контейнера <select>, элемента <optgroup>, или элемента <datalist>.

[<option>](#)

[HTMLOptionElement](#)

```
<select name="select">
  <option value="value1">Значение 1</option>
  <option value="value2" selected>Значение 2</option>
  <option value="value3">Значение 3</option>
</select>
```

disabled

Boolean, опция недоступна для выделения.

label

Если label не указан, то его значение совпадает с текстовым содержанием элемента <option>.

value

value идёт на сервер, а текст внутри option – это UI. Если value отсутствует, значение берётся из текста.

selected

Boolean атрибут отображает то, что опция изначально выделена.

label

Метка для элемента формы. Правильно использовать вместе с кнопками и напротив инпут-полей. При клике на лейбл, фокус переходит на относящийся к ней элемент.

<label> можно связать с элементом управления, поместив элемент внутри <label> или используя атрибут for.

[<label>](#)

[HTMLLabelElement](#)

```
<label>Click me <input type="text"></label>
```

```
<label for="username">Click me</label>
<input type="text" id="username">
```

for

ID labelable-элемента

text area

Поле из нескольких строчек для написания больших текстов.

[<textarea>](#)

[HTMLTextAreaElement](#)

```
<textarea name="textarea" rows="10" cols="50">
Write something here
</textarea>
```

form

Форма, с которой textarea связана.

name**placeholder**

неосязаемый текст-заменитель

cols

число, кол-во символов в ширину. По умолчанию 20.

rows

кол-во видимых линий

maxlength**minlength****disabled**

Логическое значение

autofocus**required**

поле обязательно должно быть заполнено

CSS

resize: none;

textarea:invalid {}

```
textarea:valid {}
```

select

Выпадающий список, меню опций.

Используется с <option> или <optgroup>

[<select>](#)

[HTMLSelectElement](#)

```
<select name="select">
  <option value="value1">Значение 1</option>
  <option value="value2" selected>Значение 2</option>
  <option value="value3">Значение 3</option>
</select>
```

name

form

указывает к какой форме относится <select>

multiple

логический атрибут, возможен выбор нескольких опций в списке.

required

логический атрибут, обязательно должна быть выбрана опция

size

Если элемент управления представлен как прокручиваемый список, этот атрибут указывает количество строк в списке, которые должны быть видны за раз. По умолчанию - 0.

disabled

meter

Графики-полоски для отображения прогресса, как прогресс-бар.

[<meter>](#)

[HTMLMeterElement](#)

min минимальное значение диапазона. Если не определён, то 0.
max максимальное значение диапазона. Если не определён, то 1.

low
optimum
high

value непосредственно величина значения

```
<meter
  id="meter_id"
  min="0" max="100"
  low="33" high="66" optimum="80"
  value="50">at 50/100</meter>
```

progress

Индикатор выполнения.

[<progress>](#)

HTMLProgressElement

```
<label for="progress_id">File progress:</label>

<progress id="progress_id" max="100" value="70"> 70% </progress>

max      максимальное значение
value    текущее значение
```

input

Множество разных значений. Типы инпутов задаются атрибутом type.

Используется для создания интерактивных элементов управления в веб-формах для получения данных от пользователя.

[<input>](#)

HTMLInputElement

Не все типы инпутов работают одинаково в разных браузерах. Например, в ie не работает color и date, надо использовать полифилы. Проверять в [can-i-use](#).

Некоторые инпуты не стилизуются. Date, color, range и т.д. Эти элементы приходится эмулировать, потому что они по разному отображаются в разных браузерах. Но их не часто используют.

name
placeholder
readonly
required
autofocus
disabled
size

value
Значение, которое отправится на сервер.

autocomplete
атрибут указывает, разрешено ли автоматическое заполнение поля браузером.

maxlength
Используется с text, email, search, password, tel, or url.

max
min
Используются с numeric или date-time.

size
размер инпута в пикселях или в знаках.

multiple
Boolean, может ли пользователь вводить несколько значений. Применяется, если для атрибута type задано email или file.

checked
Boolean для type = radio or checkbox.

form
Связывает элемент с формой. Нужно указать в качестве значения id формы.

formaction
Ссылка на программу, которая обработает отправленную информацию элемента, если это submit button или image. Перезаписывает (переопределяет?) [action](#) атрибут связанной формы.

formmethod

post или get, тоже переопределяет.

list + type=text

В list указывается id элемента <datalist>, в котором находится список предопределённых значений. Работет как т9. Работает только с текстовыми type. См. пример ниже.

pattern

A regular expression that the control's value is checked against.

Используется с text, search, tel, url или email.

type

если не указан, то по умолчанию используется text.

Ссылки на страницы, где расписаны возможности инпутов подробно, в [этой](#) таблице.

Текстовые данные

text

Однострочное текстовое поле. Переносы строк автоматически удаляются из входного значения.

pattern

password

текстовое поле со скрытыми символами. Используется с:

minlength

maxlength

number

Введение только числовых значений. Надо для мобильных устройств. Появляются стрелки, которыми можно проматывать значения.

Дополнительные атрибуты:

max

min

step на сколько меняется значение при клике на стрелку

email

Поле для редактирования адреса электронной почты, надо для мобильных устройств. Перед отправкой проверяется, что входное значение содержит либо пустую строку, либо один адрес электронной почты. Т.е. фишка в автоматической валидации.

Соответствуют CSS псевдоклассам :valid and :invalid.

Используется с атрбутом multiple и pattern.

tel

ввод номера телефона. Разрывы строк автоматически удаляются. Надо для мобильных устройств.

Можно использовать:

pattern, maxlength

CSS :valid, :invalid

Кнопки

submit

Кнопка для отправления формы.

value изменить текст в кнопке

reset

Кнопка сброса содержимого формы в состояние по умолчанию.

value изменить текст в кнопке

image

Кнопка отправки формы в виде изображения.

src, alt

height, width

checkbox

Флаг, выбор несколько опций. Использовать с тегом label.

value	определяет значение, которое будет передано на сервер
checked	указать, должен ли флажок быть выставлен
indeterminate	флажок находится в неопределённом состоянии

radio

Можно выбрать только одно значение из нескольких. Чтобы создать группу кнопок, у них должно быть общее имя и разные value. Использовать с тегом label.

value	определяет значение, которое будет передано на сервер
checked	указать, должен ли быть выбран вариант

```
<input type="radio" name="r_name" value="1">  
<input type="radio" name="r_name" value="2">  
<input type="radio" name="r_name" value="3">
```

Дата и время

date

Элемент для ввода даты: год, месяц и день.

datetime

Элемент для ввода даты и времени: час, минута, секунда и доля секунды.

datetime-local

для ввода даты и времени

time

Элемент для ввода значения времени без часового пояса.

month

для ввода месяца и года без часового пояса

week

для ввода даты, содержащей число неделя-год и номер недели без часового пояса.

Визуальные

range

Полоса с бегунком для установки значения. Визуально при движении ползунка данные нигде не отражаются, но их отображение можно настроить в JS.

min: 0
max: 100
value:
step: 1

color

интерфейс выбора цвета (в текстовом значении)

Разное

url

Поле для редактирования URI. Введённое значение должно содержать либо пустую строку, либо допустимый абсолютный URL. В противном случае значение не будет принято. Переводы строк, лидирующие и завершающие пробельные символы будут автоматически удалены. Можно использовать такие атрибуты как pattern или maxlength, чтобы ограничить вводимые значения. Псевдоклассы CSS :valid and :invalid.

search

текстовое поле для поисковых запросов. Нужно для браузера и поисковиков. Разрывы автоматически удаляются.

hidden

Элемент, который не отображается, но чьё значение будет отправлено на сервер. Это промежуточные данные, которые пользователю не надо показывать и пользователь не должен их изменять. Могут быть нужны для скриптов.

value скрытое дополнительное значение.

file

позволяет пользователю выбрать файл. Используется вместе с атрибутом accept.
accept=""

определить типы файлов, которые могут быть выбраны. Атрибут определяет типы файлов, которые сервер может принять. В противном случае файл игнорируется. Значение должно быть списком уникальных спецификаторов типов содержания, разделённым запятыми.
multiple возможность прикреплять сразу несколько файлов.

list + type=text

В list указывается id элемента <datalist>, в котором находится список предопределённых значений. Работет как t9. Работает только с текстовыми type.

```
<input type="text" list="list_id">
<datalist id="list_id">
  <option>опция 1</option>
  <option>опция 2</option>
</datalist>
```

</>

CSS

CSS Basics

CSS (Cascading Style Sheets) - каскадные таблицы стилей

Обычно файлы называются style.css

Служит для внешнего оформления HTML документа (шрифт, цвет, фон, тип и тд.)

CSS файлы имеют расширение .css

CSS файл состоит из набора селекторов, свойств и значений к ним. Создаёшь селектор, присваиваешь элементу, прописываешь свойства и значения.

CSS файл читается браузером сверху вниз. Правила, которые написаны последними, будут перекрывать предыдущие.

Дочерние элементы наследуют некоторые свойства их родителей

Некоторые теги имеют оформление по умолчанию, которое придает им браузер

Постоянно забываю

```
pointer-events: none;
text-transform: uppercase;
.input::placeholder
overflow: hidden;
```

Если что-то не срабатывает (например, pointer-events), то можно попробовать поставить display: inline-block.

Вендорные префиксы

Это приставка к CSS свойству, чтобы обеспечить его поддержку в браузерах, где это свойство ещё не добавлено на постоянной основе. Новые свойства долго внедряются в браузеры. Со временем необходимость добавлять это свойства отпала.

Т.е. свойство введено в спецификацию CSS, но в браузере оно либо на стадии разработки, либо в стадии тестирования. Либо это экспериментальное или нестандартное свойство. [Vendor Prefix](#).

-moz-	Firefox
-webkit-	Safari, Chrome
-o-	Opera
-ms-	Internet Explorer

Способы подключения CSS

Подключение файла

```
<link rel="stylesheet" href="style.css">
```

Импорт из одного CSS-файла в другой

```
@import "main.css"
```

Внутренние стили

В теге <style> непосредственно в head

Инлайновые стили

В атрибуте style у элемента

Специфичность селекторов

Специфичность - это способ, с помощью которого браузеры определяют, какие значения свойств CSS наиболее соответствуют элементу и, следовательно, будут применены. [MDN](#)

Взаимное расположение элементов в дереве документа не влияет на специфичность.

Стили непосредственно соответствующих элементов всегда предпочитают унаследованным стилям.

Универсальный селектор (*), комбинаторы (+, >, ~) и отрицающий псевдокласс :not() не влияют на специфичность. Однако селекторы, объявленные *внутри* :not(), влияют.

Для того, чтобы посчитать вес селектора, достаточно просчитать все входящие в него элементы (тупа математически).

1	10	100	1000
element	class	id	inline
pseudo-element	attribute		
	pseudo-class		

important в css

Модификатор технически не имеет специфичность, но он переопределяет всё ранее указанное.

important в html

Сильнее, чем important в CSS.

```
.title {  
  color: green !important;  
}
```

CSS селекторы

CSS-селектор — это часть CSS-правила, которая позволяет указать, к какому элементу (элементам) применить стиль. [MDN](#)

В качестве селекторов используются:

Простые селекторы

tag

```
.class
#id
*

a[attr=value]    по атрибуту

:hover           псевдокласс описывает состояние элемента
::псевдоэлемент
```

Комбинаторы

```
h1, h2, h3      - применить правило к нескольким селекторам

div + div        - выбрать одного соседа (брата) справа
div ~ div        - выбрать всех соседей (братьев) справа

se1 > se2        - выбрать прямых потомков
se1 se2          - выбрать всех потомков
```

Class

Если наименование класса состоит из нескольких слов, они записываются через-дефис.
Можно присваивать тегу сразу несколько классов через пробел.

```
.class1.class2
```

Выбрать такой элемент, который обладает одновременно двумя классами

* звёздочка

Звездочка (*) - универсальный селектор для CSS, соответствует любому тегу. [MDN](#)

В CSS 3 звездочка (*) может использоваться в комбинации с пространством имён:

```
ns | * - вхождения всех элементов в пространстве имён ns #?
  * | * - находит все элементы
  | * - ищет все элементы без объявленного пространства имён
```

[] Селекторы атрибутов

Вообще, это селекторы CSS, но они также используются при поиске тех или иных элементов в DOM через `querySelector`. Можно задавать условия поиска.

[MDN](#)

```
[attr]
```

Обозначает элемент с атрибутом по имени `attr`.

```
[attr=value]
```

Обозначает элемент с именем атрибута `attr` и значением **в точности** сопадающим с `value`.

```
[attr~=value]
```

Обозначает элемент с именем атрибута `attr` значением которого является **набор слов** разделенных пробелами, одно из которых в точности равно `value`

```
[attr|=value]
```

Обозначает элемент с именем атрибута `attr`. Его значение при этом может быть **или** в точности равно `"value"` или может начинаться с `"value"` со сразу же следующим `"-"` (U+002D). Это может быть использовано когда язык описывается с подкомом.

[attr^=value]

Обозначает элемент с именем атрибута attr значение которого **начинается** с "value"

[attr\$=value]

Обозначает элемент с именем атрибута attr чье значение **заканчивается** на "value"

[attr*=value]

Обозначает элемент с именем атрибута attr чье значение **содержит** по крайней мере одно вхождение строки "value" как подстроки.

Можно комбинировать селекторы, указывая их один за другим.

Выбрать все ссылки, href которых содержит «://», но которые не начинаются с «http://internal.com»:

```
selector = 'a[href*="//"]:not([href^="http://internal.com"])'
```

: Псевдоклассы

Также называются «Селекторы состояния элементов». [MDN](#)

Список стандартных псевдоклассов в документации.

Как и с обычными классами, можно совмещать вместе в одном селекторе любое число псевдоклассов.

```
selector:pseudo-class {  
  property: value;  
}
```

Некоторые псевдоклассы:

a:link все элементы, которые являются ссылками на странице
a:hover ссылки, на которые наведён курсор
a:focus ссылки, на которых сфокусировались с клавиатуры
a:active ссылки, на которые сейчас кликнули
a:visited ссылки, по которым уже переходили

:root выбирает корневой элемент HTML-документа

ul li:nth-child(3n)
каждый третий li

ul li:nth-child(n+2)
Все со второго

ul li:nth-child(odd)
обратиться ко всем нечётным элементам

ul li:nth-child(even)
обратиться к чётным элементам

ul li:first-child
только первый элемент

:last-child
только последний элемент

:first-of-type
:last-of-type
:nth-child

:only-child

выбирает элемент, который является единственным дочерним для своего родительского

:only-of-type

выбирает элемент, который является единственным элементом данного типа.

:empty

выбирает пустые элементы.

:not()

Отрицательный CSS псевдо-класс, :not(X) - функция, принимающая простой селектор X в качестве аргумента. Он находит элементы, не соответствующие селектору. X не должен содержать других отрицательных селекторов.

[MDN](#)

```
:not(selector) { style properties }
```

```
li:not(first-child)
```

выберутся все элементы списка, которые не являются первым ребёнком.

Выбрать все ссылки, href которых содержит «://», но которые не начинаются с «http://internal.com»:

```
selector = 'a[href*="//"]:not([href^="http://internal.com"] )'
```

:: Псевдоэлементы

Псевдоэлемент в CSS позволяет стилизовать определённую часть выбранного элемента.

Список псевдоэлементов в документации [MDN](#).

В селекторе можно использовать только один псевдоэлемент. Он должен находиться после простых селекторов в выражении.

```
selector::pseudo-element {  
  property: value;  
}
```

Некоторые псевдоэлементы:

```
::first-letter выделяет первую букву текста  
::first-line выделяет первую строку текста  
::before  
::after
```

Наследование стилей

Дочерние элементы наследуют некоторые стили родительских элементов, если такие стили не были заданы.

В основном наследуется стиль оформления текста.

Не наследуются рамки, позиционирование, отступы.

Блочная модель

Все элементы делятся на 3 типа:

- строчные
- блочные
- смесь

Строчные элементы подстраиваются под контент, в котором находятся.

Блочные занимают всю ширину экрана, занимают отдельную строку.

Если к строчным добавить `margin` или `padding`, то отступы появятся только по сторонам, но не по высоте.

Марджины схлопываются.

С помощью `margin-right` и `left = auto` выравнивают все сайты по центру. Для этого после `body` всё содержимое упаковывается в `div` и к нему применяется `margin auto`. Ещё можно указать фиксированную ширину `width`.

Display – свойство, которое управляет типом тега.

```
display: inline
display: block
display: inline-block
```

```
display: flex
display: grid
```

Box-sizing – свойство, которое рассчитывает суммарный размер элемента.

```
box-sizing: content-box
box-sizing: border-box
```

Обычно его прописывают в самом начале сразу для всех элементов:

```
*, *:before, *:after {
  box-sizing: border-box;
}
```

Другие свойства по теме

```
margin
padding
```

```
border: 1px solid #333;
border-collapse: collapse
```

 схлопывание границ у таблиц

```
outline: 1px solid red;  обводка элемента. Не влияет на размер блока
outline-offset: 2px;     отступ между обводкой outline и элементом
```

Позиционирование

Изначально все элементы находятся в основном потоке документа (на одном уровне).

Свойство `position` вырывает элемент из освного потока и формирует новый.

position

Задаёт позиционирование элемента в документе, используется вместе с `top`, `right`, `bottom`, `left` и `z-index`.

Существует 5 основных типов позиционирования:

```
position: static;  элемент в обычном потоке документа, свойства не действуют
position: relative; элемент остается в потоке, но его можно сдвигать относительно его позиции
```

```
position: absolute;   позиционируется относительно ближайшего родителя с не-static
position: fixed;      позиционирование относительно окна браузера.
position: sticky;
```

top, right, bottom, left, z-index

На сколько пикселей элемент сдвигается от указанной стороны.

Отступы задаются в пикселях или процентах.

Отступ считается от ближайшего предка с position: relative или от body.

```
position: absolute;

top: 50px;
left: 50px;

bottom: 70px;
right: 70px;

z-index: 2
```

Если отступ указать в процентах, отсчёт будет вестись от верхней точки элемента, а не его центра. Поэтому если указать позиционирование 50%, то по середине будет не сам элемент, а только его верхняя точка.

Чтобы решить эту проблему, надо указать отрицательное значение margin-top, равное половине высоты элемента, он сдвинется вверх. Это один из способов решения проблемы «верхней точки».

Размеры элемента

Если задать размеры жёстко, то элемент не будет «резиновым», он не будет растягиваться или сжиматься под контент.

Max-значения – это максимальный размер, больше которого элемент быть не может, но он будет сжиматься.

Min-значения – наоборот.

```
width
height

max-width
max-height

min-width
min-height
```

Единицы

Абсолютные единицы
px

Относительные
rem root, размеры считаются от базового размера в html
em текущий размер шрифта
% рассчитывается от размера родительского элемента

Размер окна браузера
vh
vw

Вычисление
width: calc(10px + 200px + 1rem)

Переполнение блока

overflow определяет, как блок будет вести себя при переполнении.

Сокращенная запись, которая объединяет два свойства overflow-x и overflow-y

overflow: visible;	
overflow: hidden;	скрывает то, что выходит за границы контейнера
overflow: scroll;	прокрутка видна всегда
overflow: auto;	прокрутка появится автоматом
overflow: hidden auto;	два параметра задают разные значения для X и Y осей
overflow-x	
overflow-y	

Маска

Маска — это когда затемняется весь экран, становится неактивным и обычно на нём всплывает модальное окно.

```
.html
<div class="mask"></div>
<div class="modal"></div>

.css
.mask {
  width: 100vw;
  height: 100vh;
  background-color: blue;
  opacity: .2;
  z-index: 100;
  position: absolute; fixed
}

.modal {
  width: 200px;
  height: 200px;
  background-color: tomato;

  position: fixed;
  top: 50%;
  left: 50%;
  z-index: 101;
}
```

Текст

font	размер шрифта, шрифт и другие параметры.
font-family	тип шрифта
font-size	размер
font-weight	жирный
font-style	наклонный
color	цвет шрифта
line-height	межстрочный интервал
letter-spacing	расстояние между буквами
word-spacing	расстояние между словами
vertical-align	выравнивание инлайн-элемента по вертикали
text-align	выравнивание инлайн-элемента по горизонту

text-decoration	подчёркивание и зачёркивание
text-transform	регистр букв
text-indent	отступ красная строка

font

font: bold italic 15px/1.5 Arial, sans-serif;

Жирный шрифт

font-weight: bold;	жирный размер
font-weight: normal;	обычный размер
font-weight: initial;	значение по умолчанию
font-weight: inherit;	наследовать от родителя

Числовые значения

font-weight: 100-900; 400 – норма

Выравнивание по горизонту

text-align: right;	
text-align: left;	
text-align: center;	
text-align: justify;	автоматом по ширине блока

Выравнивание по вертикали

vertical-align: top;	
vertical-align: center;	
vertical-align: middle;	
vertical-align: bottom;	
vertical-align: baseline, sub, super, text-top, text-bottom	

Подчёркивание

text-decoration: underline;	подчёркнутый снизу
text-decoration: overline;	линия сверху
text-decoration: line-through;	зачёркнутый текст
text-decoration: underline overline line-through;	

text-decoration-color	цвет
text-decoration-line	начертание линии
text-decoration-style	где подчёркивать
text-decoration: dotted underline red;	

Регистр

text-transform: uppercase;	все буквы к верхнему регистру
text-transform: lowercase;	все буквы к нижнему регистру
text-transform: capitalize;	сделать первую букву каждого слова заглавной

Безопасные шрифты – те, которые есть на каждом компьютере.

Сглаживание шрифтов

Некоторые шрифты и браузеры поддерживают сглаживание. Оно включается вот так:

```
body {  
  -webkit-font-smoothing: antialiased;  
  -moz-osx-font-smoothing: grayscale;  
}
```

Текстовые эффекты и многоколоночный текст

Обезка текста с многоточием

`overflow: hidden;`

`text-overflow: ellipsis;`

Совместное использование обрежет слово, которое не помещается в границы блока, а в конце поставит многоточие.

Запрет переносов

`white-space: nowrap;`

Длинный текст не будет переноситься на новую строку, если не помещается в контейнере по ширине.

Перенос по слогам

`word-wrap: break-word;`

Если слово не помещается в строку, будет перенесена только его часть, а не всё слово целиком.

writing-mode

устанавливает горизонтальное или вертикальное положение текста.

`vertical-rl`

Текст располагается вертикально от верха к низу, прижимается к правому краю контейнера.

`vertical-lr`

Текст располагается вертикально от верха к низу, прижимается к левому краю контейнера.

Несколько колонок

Если в контейнере есть несколько блоков (например, `p`), их можно разбить на колонки.

```
.container {  
  column-count: 3;  
}
```

column-count

разбивает содержимое элемента на заданное число столбцов.

column-rule

Устанавливает ширину, стиль и цвет линии, находящейся между колонками в мультиколоночной вёрстке. По принимаемым свойствам полностью аналогична `border`. Краткая форма записи для:

`column-rule-width`

`column-rule-style`

`column-rule-color`

column-width

Ширина колонки. Если установлено, кол-во колонок определяется автоматом, несмотря на `column-count`.

column-gap

Расстояние между колонками.

Подключение шрифтов

Сервис гугла со шрифтами: fonts.google.com

Локальные шрифты

Для сайтов подходят шрифты в формате `.ttf`, `.woff` и `.woff2`. Другие форматы, на самом деле, тоже подходят.

Шрифты надо скопировать в папку «font» в проекте.

Далее всё делать в файле через директиву `@font-face`:

```
@font-face {  
  font-family: 'Chocolates';  
  src: url('../fonts/TTChocolates-Regular.woff') format('woff'),  
       url('../fonts/TTChocolates-Regular.woff2') format('woff2');
```

```
}  
  
@font-face {  
  font-family: 'Chocolates';  
  src: url('./fonts/TTChocolates-Bold.woff') format('woff'),  
       url('/fonts/TTChocolates-Bold.woff2') format('woff2');  
  font-weight: 700;  
}
```

font-family

Наименование может быть любым, его можно написать от балды.

src

Путь к файлу со шрифтом относительно CSS файла.

Через запятую указываются несколько форматов. Если браузер формат не поддерживает, он выберет нужный.

```
font-weight: 700;  
font-style: italic;
```

Можно указать эти свойства, и когда на практике соответствующие стили будут использоваться, шрифты по указанным адресам будут подгружаться.

Границы

border

Граница участвует в блочной модели

```
border: 1px solid red
```

border-width	ширина границы
border-style	8 типов
border-color	цвет границы

border-right	left top bottom
border-rigth-color	

```
border-collapse: collapse
```

 схлопывание границ у таблиц

outline

Outline – дополнительная рамка вокруг элемента, но она не участвует в блочной модели. Нельзя обращаться отдельно к каждой стороне. Есть ещё одно свойство: outline-offset. Это отступ обводки от элемента.

Рекомендуется использовать вместе с кнопками и полями ввода, чтобы при нажатии на tab было понятно, где сейчас фокус.

```
outline-width  
outline-style  
outline-color
```

```
outline: 1px solid red;  
outline-offset: 2px;
```

Задаёт отступ между обводкой outline и элементом

Тени

```
text-shadow: 1px 2px 5px red
```

```
text-shadow: 1px 2px 5px red, 6px 7px 8px blue inset;
```

горизонтальное смещение

вертикальное смещение

размытость

цвет

`inset` – тень отрисовывается внутри элемента

```
box-shadow: 1px 2px 5px red;
```

```
box-shadow: 1px 2px 5px red, 6px 7px 8px blue inset;
```

множественные тени через запятую

Скругление углов

```
border-radius: 50px;
```

```
border-radius: 50px 40px 30px 20px;
```

по часовой стрелке от верхнего левого угла

```
border-top-left-radius: 50px;
```

```
border-radius: 50px/40px;
```

радиус по горизонтали

радиус по вертикали

Списки, css

`list-style-type`

Тип маркера для нумерованного списка

```
list-style-type: disc;
```

чёрная точка, по умолчанию

```
list-style-type: circle;
```

круг

```
list-style-type: square;
```

квадрат

```
list-style-type: decimal
```

нумерованный список

```
list-style-type: decimal-leading-zero
```

нумерованный список из двух цифр: 01, 02

```
list-style-type: upper-alpha;
```

буквы вместо цифр, верхний регистр

```
list-style-type: lower-alpha;
```

буквы вместо цифр, нижний регистр

```
list-style-type: upper-roman;
```

римские цифры

```
list-style-type: lower-roman;
```

```
list-style-type: none;
```

`list-style-position`

позиционирование маркера относительно списка

```
list-style-position: outside;
```

маркеры за границей списка, по умолчанию

```
list-style-position: inside;
```

маркер внутри границы списка, физически перед буквами

`list-style-image`

нельзя управлять размерами значка и его позицией.

```
list-style-image: url();
```

использовать свою иконку

Цвета

```
color: словом  
color: #hex
```

```
color: rgb(225, 0, 0)  
color: rgba(225, 0, 0, 0.3)
```

```
color: hsl(0, 100%, 50%)  
color: hsla(0, 100%, 50%, )
```

hue, тон
saturation, насыщенность
lightness, светлота

Видимость элементов

Элементы можно скрывать с помощью свойств:

```
visibility: hidden;  
физически скрыт, но место остаётся занятым.  
Для поисковых систем доступен.  
visibility: collapse;  
Элемент скрыт, место под ним занимает другими элементами.
```

```
display: none;  
Элемент скрыт со страницы, как если бы его вообще не было в HTML.  
Для поисковиков не доступен.
```

HTMLElement.hidden
Атрибут и DOM-свойство, технически работает как style="display:none", но применение проще.

```
opacity: 0;  
Элемент просто становится прозрачным, занимает место  
Можно сделать эффект плавного появления с помощью transition.
```

```
color: transparent  
костыль
```

visibility

Скрывает или показывает элемент без изменения разметки документа. На практике с ним редко работают.

```
visibility: visible;  
visibility: hidden;  
visibility: collapse;
```

```
visibility: visible;  
элемент видно, по умолчанию
```

```
visibility: collapse;  
Элемент скрыт, место под ним занимает другими элементами.
```

```
visibility: hidden;  
физически скрыт, но место остаётся занятым.
```


Если применить `:hover` на элементе, то он не сработает, но если навести на другой элемент на странице, то он появится. При появлении к изображению будут применены все прописанные ему в CSS правила.

```
.text:hover .img {  
    visibility: visible;  
}
```

HTMLElement.hidden

Свойство является Boolean типом данных, который принимает значение `true`, если содержимое скрыто, в противном случае значение будет `false`.

Атрибут и DOM-свойство «`hidden`» указывает на то, видим ли мы элемент или нет.

Технически, работает так же, как `display:none`, но его применение проще.

```
isHidden = HTMLElement.hidden;  
HTMLElement.hidden = true | false;
```

Мы можем использовать его в HTML или назначать при помощи JavaScript:

```
<div>Оба тега DIV внизу невидимы</div>  
  
<div hidden>С атрибутом "hidden"</div>  
  
<div id="elem">С назначенным JavaScript свойством "hidden"</div>  
  
<script>  
    elem.hidden = true;  
</script>
```

Мигающий элемент:

```
<div id="elem">Мигающий элемент</div>  
<script>  
    setInterval(() => elem.hidden = !elem.hidden, 1000);  
</script>
```

Фон

background

```
background: #fff url("images/bg.jpg") center no-repeat;  
background: #fff url("images/bg.jpg") center no-repeat, background: #fff url("images/bg.jpg") center no-repeat;  
позволяет добавлять неограниченное кол-во фоновых изображений
```

```
background-color  
background-image: url();
```

```
background-size: cover;  
background-repeat: no-repeat;  
background-position  
background-origin  
background-attachment  
background-clip
```

```
background-size: cover;  
растягивается по ширине и высоте блока, но не искажается (обрезается)  
background-size: contain;
```

изображение сохраняет пропорции, моститься.

`background-size: 100px 200px; auto, %`

будет моститься

`background-size: auto;`

оригинальный размер изображения

`background-repeat: no-repeat;`

фоновое изображение не будет моститься

`background-repeat: repeat-x;`

`background-repeat: repeat-y;`

`background-position: x y; px, %`

`background-position: center center;`

`background-position: right bottom;`

`background-attachment: fixed;`

фоновое изображение не будет скролиться, всегда в поле зрения

`background-attachment: scroll;`

`background-attachment: local;`

если элемент имеет механизм прокрутки, фон прокручивается с содержимым элемента

`background-clip: border-box;`

Фон будет расположен под границей border элемента

`background-clip: content-box;`

Фон элемента будет расположен строго под содержимым элемента

`background-clip: padding-box;`

Фон не будет размещен под границей border элемента

Данное значение экспериментальное

`background-origin`

Градиент

`background-image: linear-gradient(45deg, red, blue);`

`background-image: linear-gradient(45deg, red, blue, green);`

`background-image: linear-gradient(45deg, red 10%, blue 50%, green 70%);`

угол наклона

начальный цвет

конечный цвет

сколько угодно цветов

процент – опорная точка, с которой начнётся переход

`background: linear-gradient(to top right, red, blue);`

to left

to bottom

to right

to top

`background-image: linear-gradient(45deg, red 10%, blue 10%, blue 90%, green 90%, green 100%);`

полосы вместо градиента

`background-image: repeating-linear-gradient(45deg, red 10px, blue 10px, blue 20px);`

повторять градиент

`background-image: radial-gradient(circle, #f16e12, #9c1717);`

`background-image: radial-gradient(circle closest-side, #f16e12, #9c1717);`

circle, ellipse

closest-side	распространение из центра к ближайшему краю, всегда помещается в границы
closest-corner	распространяется к углам
farthest-side	распространение до дальней стороны
farthest-corner	

Фильтры

позволяет применять к элементу графические эффекты типа размытие и смещение цвета. Фильтры обычно используются с изображениями, фонами и рамками. [MDN](#).

```
filter: url(resources.svg);
filter: blur(5px);
filter: brightness(0.4);
filter: contrast(200%);
filter: drop-shadow(16px 16px 20px blue);
filter: grayscale(50%);
filter: hue-rotate(90deg);
filter: invert(75%);
filter: opacity(25%);
filter: saturate(30%);
filter: sepia(60%);

/* Применение нескольких фильтров */
filter: contrast(175%) brightness(3%);
```

Трансформации

Это вращение элемента, изменение размеров, наклоны, сдвиги.

Мультифункциональные значения

```
transform: translateX(10px) rotate(10deg) translateY(5px);
transform: perspective(500px) translate(10px, 0, 20px) rotateY(3deg);
```

Вращение

```
transform: rotate(120%);
transform: rotate(0.5turn);
transform: rotate3d(1, 2.0, 3.0, 10deg);
transform: rotateX(10deg);
transform: rotateY(10deg);
transform: rotateZ(10deg);
```

Масштабирование

```
transform: scale(0.5);
transform: scale(2, 0.5);
transform: scale3d(2.5, 1.2, 0.3);
transform: scaleX(2);
transform: scaleY(0.5);
transform: scaleZ(0.3);
```

Смещение

```
transform: translate(12px, 50%);
transform: translate3d(12px, 50%, 3em);
transform: translateX(2em);
transform: translateY(3in);
transform: translateZ(2px);
```

Наклон

```
transform: skewX(30deg);
transform: skew(30deg, 20deg);
transform: skewY(1.07rad);

transform: matrix(1.0, 2.0, 3.0, 4.0, 5.0, 6.0);
transform: matrix3d(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
```

Перспектива

```
transform: perspective(17px);
```

Чтобы вращение отображалось реалистично, надо использовать свойство **perspective**. Оно прописывается для контейнера, в котором находится вращающийся объект, а не самому объекту. Обычно ставится 300 или 500px

Опорная точка

```
transform-origin: right bottom;
right, bottom, left, top, px
```

Плавный переход

Список свойств, с которыми можно использовать transition: [flaviocopes.com](https://flaviocopes.com/transition/)

Не работает с градиентами.

Краткая запись

```
transition: property duration [function] [delay]
```

```
transition: opacity 0.2s linear 0.4s;
transition: all 0.2s linear 0.4s;
```

Свойства отдельно

transition-property	к какому свойству применять плавность
transition-duration	длительность перехода свойств
transition-timing-function	функция расчёта времени
transition-delay	задержка перед изменением свойств

Стандартные функции (есть другие, см. [здесь](#)):

linear, ease, ease-in, ease-out, ease-in-out, step-start, step-end

Несколько свойств подряд

```
transition:
  opacity 0.2s linear 0.4s,
  background 0.2s linear;

{
transition-property: transform, background-color;
transition-duration: 500ms, 2s;
transition-timing-delay: 0, 500ms;
} для двух разных свойств установлено разное время
```

Анимации

Руководство в MDN: [Использование CSS-анимации](#)

Разница между переходом transition:

Переход позволяет анимировать переход одного набора свойств в другие.

Анимация позволяет анимировать более двух переходов наборов свойств. Анимацию можно зациклить, остановить. При создании анимации создаются ключевые кадры. Каждый кадр содержит набор свойств, которые должны изменяться в соответствующем промежутке времени.

Анимация создаётся в 2 этапа:

1. Определить анимацию в [@keyframes](#);
2. Применить её к элементу.

Все свойства анимации можно записать через одно свойство `animation`.

Можно применять несколько анимаций к элементу, их имена и свойства записываются через запятую.

Определение анимации

```
@keyframes animationName {
  from {
    /* исходные значения свойства */
    background-color: aqua;
  }

  50% {
    /* свойства, к которым анимация придёт */
    background-color: red;
  }

  to {
    /* свойства, к которым анимация придёт */
    background-color: green;
  }
}
```

Применение к элементу

`animation-name`

Имя анимации, которая будет использоваться с элементом. Каждое имя является правилом `@keyframes`.

Одиночная анимация: `name1`;

Несколько анимаций: `name1, name2`;

`animation-duration`

Определяет время, в течение которого должен пройти один цикл анимации.

Задаётся в `s` или `ms`. `0s` – анимация не выполняется.

`animation-timing-function`

Настраивает ускорение анимации. Значений много, см. документацию. Некоторые значения:

`linear`;

`ease`; `ease-in`; `ease-out`; `ease-in-out`;

`step-start`; `step-end`;

`animation-direction`

Направление проигрывания и повторы.

Одиночная анимация: `normal`; `reverse`; `alternate`; `alternate-reverse`;

Несколько анимаций: `normal, reverse`; `alternate, reverse, normal`;

`animation-iteration-count`

количество повторений анимации. `infinite` для бесконечного повторения.

`animation-fill-mode`

Настраивает значения, используемые анимацией, до и после исполнения.

`none`; сбросить на первый кадр

`forwards`; оставить последний кадр

`backwards`;

`both`;

animation-play-state

Позволяет приостановить и возобновить анимацию. Можно повесить на :hover

Одна анимация: running; paused;

Несколько анимаций: paused, running, running;

animation-delay

Время задержки перед началом анимации.

Примеры:

```
.container {
  margin: 150px auto;
  width: 200px;
  height: 200px;
  animation-name: animationColor, anumationSize;
  animation-iteration-count: infinite;
  animation-duration: 1s, 2s;
  animation-fill-mode: forwards, forwards;
  animation-direction: alternate, alternate;
  animation-timing-function: linear;
}

.container:hover {
  animation-play-state: paused;
}

@keyframes animationColor {
  from {
    background-color: aqua;
  }
  50% {
    background-color: red;
  }
  to {
    background-color: green;
  }
}

@keyframes anumationSize {
  from {
    transform: scale(1);
    transform: rotate(0deg);
  }

  to {
    transform: scale(2);
    transform: rotate(90deg);
  }
}
```

Таблицы

Текст в tCaprion и thead – жирный и выравнивается по центру.

Ширина колонок выставляется автоматом в зависимости от ширины самого длинного слова.

Отступы, скругление углов, градиенты – это всё тоже актуально.

```
table: {
```

max-width	таблица и ячейки не будут растягиваться на всю страницу по ширине
table-layout	управляет размерами ячеек. См. документацию по ссылке.

Границы

<code>border</code>	у таблицы и ячеек появятся границы.
<code>border-collapse</code>	схлопнуть границы, будут выглядеть нормально.
<code>border-spacing</code>	ширина двойных границ таблицы, когда включен <code>border-collapse: separate</code>
<code>empty cells</code>	отображение ячеек без контента, когда включен <code>border-collapse: separate</code>

Заголовок таблицы

<code>caption-side</code>	Положение заголовка <code>caption</code> в таблице. Главные значения: <code>top</code> <code>bottom</code>
---------------------------	--

Выравнивание текста

<code>text-align</code>	горизонтальное выравнивание
<code>vertical-align</code>	вертикальное выравнивание, много значений.

Курсор

`pointer-events`

Определяет то, как элементы будут реагировать на курсор или прикосновение к сенсорному экрану.

`none`

повешенные в JS обработчики перестанут работать. Для курсора элемент невидим. Если прописать это для текста, то текст же не получится выделить. А изображение – сохранить.

`cursor`

Определяет, как будет выглядеть курсор, когда наведён над элементом. Значений много, в т.ч. использование кастомных изображений и координат, см. документацию.

Примеры значений можно посмотреть на csscursor.info.

Некоторые значения:

<code>auto</code>	курсor будет вести себя как обозначено браузером
<code>none</code>	убирает курсор при взаимодействии с элементом
<code>default</code>	обычный курсор
<code>pointer</code>	меняет вид курсора на <code>pointer</code>

Плейсхолдер

`::placeholder`

```
::placeholder {  
  color: blue;  
  font-size: 1.5em;  
}
```

`input type="date"`

`input type="time"`

невозможно стилизовать

Скролл

`::-webkit-scrollbar`

only available in Blink- and WebKit-based browsers: Chrome, Edge, Opera, Safari.

<code>::-webkit-scrollbar-track</code>	стилизация подложки скрола (по которому двигается ползунок)
<code>::-webkit-scrollbar-thumb</code>	стилизация ползунка

scrollbar-color

Спрайты

Набор картинок на прозрачном фоне – это css спрайт. Все картинки загружаются сразу вместе, в результате не надо делать много запросов на сервер.

Отображение каждой картинки происходит с помощью background-position.

```
button {
width: 200px;
height: 200px;
outline: gray;
background-color: transparent;
background-image: url(sprite.png);
background-position: top;
background-size: cover;
}

button:hover {
background-position: center;
}

button:active {
background-position: bottom;
}
```

Запрет зума при двойном клике

```
touch-action: manipulation
```

CSS Медиазапросы

Медиазапрос – это директива или правило, которое позволяет применять различные свойства в зависимости от разрешения страницы, ориентации устройства, плотности пикселей, типа устройства. Всё это можно комбинировать.

Чаще всего используются для:

- настройки кол-ва колонок
- параметров ширины
- сжатия пустых пространств
- размера шрифтов
- изменения вида навигации
- скрытия и отображения контента на разных устройствах

@media

Использование медиавыражений

```
@media (max-height: 350px) {
@media screen, print {
```

Комплексные выражения

Используются операторы not, and, и only.

```
@media (min-width) and (max-width) {
```



```
@media (max-height: 350px) and (orientation: landscape) {  
@media speech and (aspect-ratio: 11/5) {  
@media not screen and (color), print and (color) {
```

width height

```
width: 360px  
min-width: 35rem  
max-width: 50rem
```

Медиазапросы для меньших размеров надо прописывать после больших.

Обычно запросы делаются для следующих величин: 1200px, 991px, 767px, 575px.

aspect-ratio

```
min-aspect-ratio: 8/5  
max-aspect-ratio: 3/2  
aspect-ratio: 1/1
```

orientation

```
landscape  
portrait
```

resolution

```
resolution: 150dpi  
min-resolution: 72dpi  
max-resolution: 300dpi
```

screen	для цветных компьютерных экранов
print	для принтеров
speech	предназначен для синтезаторов речи
all	подходит для всех устройств

Meta

Чтобы всё адаптировалось на экране телефонов, надо указывать такой мета-тег:

```
<meta name="viewport" content="width=device-width, initial-scale=1">  
<meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=yes">
```

name="viewport"

Это одно из многих значений мета-информации.

content = "width=device-width"

Ширина сайта равняется ширине экрана устройства, с которого он открыт.

content = "initial-scale=1"

Говорим браузеру, чтобы по умолчанию он не увеличивал (не зумил) страницу.

Мы говорим браузеру, чтобы сайт принимал ширину устройства (width=device-width) и не зумил страницу (initial-scale=1).

CSS Flexbox

Flexible Box Layout Module (Flexbox) представляет собой способ компоновки элементов, в основе которого лежит идея оси. Другими словами все элементы можно располагать вдоль основной и поперечной осей, или вертикально и горизонтально. Технология позволяет буквально в пару свойств создать гибкий лэйаут, или гибкие UI элементы.

Дочерние элементы внутри flex-блока называются flex item.

Решает задачи:

- контроль размеров и порядка элементов
- их горизонтальное и вертикальное выравнивание
- управление пустым пространством
- размещение элементов друг относительно друга.

Основные понятия Flexbox

При работе с flexbox нужно мыслить с точки зрения двух осей – главной оси и побочной оси.

Разбираемся с обёртыванием Flex элементов – годно!

CSS Flexible Box Layout

CSS Properties – все свойства

Создание промежутков между элементами

flex

Создать flex-контейнер

display: flex

display: inline-flex блок

Свойства контейнера

flex-direction управление главной осью

row

row-reverse

column

column-reverse

flex-wrap

возможность переносить элементы на новую строку при переполнении контейнера

nowrap

wrap

wrap-reverse

flex-flow

flex-direction и flex-wrap

Свойства для элементов

flex-basis определяет размер доступного пространства для элемента

auto

Если элементам задана ширина, то она и будет в flex-basis: auto. Если ширина не задана - используется размер контента. Если размер задан в flex-basis: 100px, то basis имеет приоритет перед width и height.

flex-grow

определяет как много свободного пространства может быть назначено элементу

0.1 займёт часть свободного пространства

1 займёт всё свободное пространство. Можно дать каждому элементу, поровну разделят

2 займёт больше места, чем элемент с коэффициентом 1

flex-shrink

определяет фактор сжатия flex-элемента, когда контейнер переполнен.

1 меньше сжатие

2 больше сжатие

flex

сокращённая запись для grow, shrink, basis

flex: 2 2 10%

Выравнивание и распределение элементов

justify-content выравнивание элементов по горизонтали (основная ось). Примеры [здесь](#).

center по центру строки

flex-start прижать к левому краю

flex-end	прижать к правому краю
space-between	элементы равномерно распределяются по всей строке, крайние прижимаются к краю.
space-around	элементы равномерно распределяются по всей строке, крайние не прижаты к краю.
space-evenly	расстояние между любыми элементами, а также у краёв одинаковое.
align-items	выравнивание элементов по вертикали (вспомогательная ось). Примеры здесь .
flex-start	элементы выравниваются в начале вертикальной оси контейнера (вверху)
flex-end	элементы выравниваются в конце вертикальной оси контейнера (внизу)
center	по центру
baseline	флексy выравниваются по их базовой линии
stretch	элементы растягиваются на всю высоту, если им не задан height
align-content	то же, что и align-items, но для выравнивания многострочного контента (с wrap).
center	по центру строки
flex-start	прижать к началу оси
flex-end	прижать к концу оси
space-between	элементы равномерно распределяются по всей строке, крайние прижимаются к краю.
space-around	элементы равномерно распределяются по всей строке, крайние не прижаты к краю.
space-evenly	расстояние между любыми элементами, а также у краёв одинаковое.
align-self	выравнивание индивидуального элемента, переписывает значение align-items
order	элементы располагаются в восходящем порядке по их значению order
-1, 1, 2	

</>

Flexbox

Хорошая статья про флекс [здесь](#).

MDN

[Использование flex-контейнеров CSS](#) (устарела)

[Основные понятия Flexbox](#) (новая) – годно!

[Разбираемся с обёртыванием Flex элементов](#) – годно!

[CSS Flexible Box Layout](#)

[CSS Properties](#) – все свойства

[Создание промежутков между элементами](#)

align-content	распределение пространства (?) по вертикали
align-items	выравнивание элементов по вертикали (как justify-content)
align-self	
justify-content	как распределить пространство между и вокруг элементов
order	порядок размещения flex элементов в контейнере
flex	flex-grow, flex-shrink и flex-basis
flex-basis	размер доступного пространства элемента в flexbox, как width и height(? клик)
flex-direction	как flex-элементы располагаются в контейнере: колонна-линия, normal-reversed
flex-flow	flex-direction и flex-wrap
flex-grow	сколько свободного места в контейнере должно быть назначено текущему элементу
flex-shrink	определяет фактор сжатия flex-элемента
flex-wrap	возможность переноса элементов на новую строку (клик)
flex	flex-grow, flex-shrink и flex-basis
flex-grow	возможность элементов расти
flex-shrink	возможность элементов сжиматься
flex-basis	размер доступного пространства элемента в flexbox

display: flex

Свойство для родительского контейнера, чтобы включить флекс.

Можно прописывать для табличного элемента ul.

display: flex

Flex – это значение, которое определяет, как будут располагаться дочерние элементы этого блока.

Чтобы сделать сайдбар, нужно создать родительский div и бросить туда два (или больше) контейнера: aside и main. Родительскому div в CSS прописать свойство display: flex.

Получится вот так:

Sidebar	Main	Sidebar
---------	------	---------

```
.html
<div class="content">

    <aside class="sidebar">Sidebar</aside>
    <main class="main">content</main>
    <aside class="sidebar">Sidebar</aside>
</div>
```

```
.css
.content {
    display: flex;
}
```

flex

Сокращенная запись для свойств

flex-grow (рост элемента),

flex-shrink (сжатие элемента),

flex-basis

и применяется к элементам контейнера flexbox.

```
flex: 1 1 100px;
```

flex: 1;

Если указан один параметр без единиц измерения - определяет значение для flex-grow, при этом flex-basis будет равен 0. В итоге получим элементы одинаковой ширины.

flex: 50%;

Если указан один параметр с единицами измерения - определяет значение для flex-basis, при этом flex-grow будет равен 1

flex: 0 40%;

Если указано два параметра, первый без единиц измерения, второй с единицами измерения - определяет значение для flex-grow и flex-basis

flex: 1 1;

Если указано два параметра и оба без единиц измерения - определяет значение для flex-grow и flex-shrink

flex: 1 1 50%;

Если указаны все три параметра, определяет flex-grow, flex-shrink, flex-basis

flex-shrink

Устанавливает коэффициент сжатия элемента контейнера flexbox

```
flex-shrink: 1;
```

flex-shrink: 0;

Элемент не будет уменьшаться, он сохранит нужную ему ширину и не будет переносить содержимое, если размеры элемента не заданы. Его родные братья сократятся, чтобы освободить место для целевого элемента. Существует вероятность переполнения содержимого контейнера flexbox, если для него не задан flex-wrap: wrap;

flex-shrink: 1;

Элемент будет сжиматься при необходимости

`flex-shrink: 2;`

Значение `flex-shrink` является относительным, его поведение зависит от значения родственных элементов контейнера `flexbox`.

Если в контейнере находятся три элемента, один из них пытается занять много пространства, у второго `flex-shrink: 1`; а у третьего `flex-shrink: 2`; Необходимое, первому элементу, пространство будет взято от элемента с `flex-shrink: 2`; в два раза больше чем от элемента с `flex-shrink: 1`;

`flex-basis`

Задаёт начальный размер элемента в контейнере `flexbox`.

Похоже, что это то же самое, что и `width` и `height`. Данное свойство имеет приоритет перед `width` и `height` свойствами.

Если используется свойство «`flex-direction: row`», то `flex-basis` отвечает за ширину элемента, если «`flex-direction: column`» - за высоту элемента.

`flex-wrap`

Определяет возможность переносить элементы контейнера `flexbox` на новую строку.

Если они не помещаются по ширине, то они не будут утробовываться, а будут перенесены на новую строку.

```
flex-wrap: wrap;           разрешает перенос
flex-wrap: wrap-reverse;   разрешает перенос, если они не помещаются
```

`flex-wrap: nowrap;`

Не разрешает перенос элементов на новую строку. Элементы будут сжаты, если не помещаются в строку и если их сжатие разрешено

`flex-wrap: wrap;`

Разрешает перенос элементов на новую строку, если они не помещаются в текущей строке. Новая строка будет добавлена после предыдущей строки

`flex-wrap: wrap-reverse;`

Разрешает перенос элементов на новую строку, если они не помещаются в текущей строке. Новая строка будет добавлена перед предыдущей строкой

`justify-content`

Выравнивание `flex` элементов.

Определяет, как элементы `flexbox` / `grid` выравниваются в соответствии с главной осью внутри контейнера.

Прописывается в родительском элементе.

```
justify-content: flex-start;
justify-content: flex-end;

justify-content: center;
justify-content: start;
justify-content: end;
justify-content: stretch;

justify-content: space-between;
justify-content: space-around;
justify-content: space-evenly;
```

Свойств много, они с примерами описаны на [htmlbase](https://htmlbase.ru/justify-content/).

С их помощью можно автоматом выравнивать элементы по разным принципам.

`justify-content: flex-start;`



`justify-content: flex-end;`



`justify-content: center;`

Элементы flexbox / grid центрированы вдоль главной оси контейнера



`justify-content: space-between;`

Оставшееся пространство распределяется между элементами flexbox / grid



`justify-content: space-around;`

Оставшееся пространство распределяется вокруг элементов flexbox / grid. Пустое пространство до первого и после последнего элемента равно половине пространства между каждым элементом.



Пример – надо создать шапку страницы, на которой указать наименование с левой стороны, а почту – с правой. Это можно сделать с помощью 3-х контейнеров: родительского и дочерних `aside` с свойством `justify-content`. Отличие предыдущих методов в том, что тут в родителе только 2 контейнера, которым автоматически присваивается необходимая ширина и которые разносятся по сторонам. Внутри нет третьего контейнера, который занимал бы необходимое место между ними.

Получится вот так:

Website	(пусто)	E-mail@ya.ru
---------	---------	--------------

```
.html
<header class="header">

  <div>Website</div>
  <div>E-mail@ya.ru</div>

</header>
```

```
.css
.header {
  display: flex;
  justify-content: space-between;
}
```

Если `div`-ам присвоить класс и указать для них одинаковую ширину (`width: 50%`), то текст в них съедет к началу контейнера. Получится так:

Website		E-mail@ya.ru
---------	--	--------------

align-items

Во Flexbox свойство `align-items` контролирует выравнивание элементов по вертикали. В grid свойство `align-items` контролирует выравнивание элементов на оси блока в пределах их области сетки.

```
align-items: center;
align-items: flex-start;
align-items: flex-end;

align-items: stretch;
```

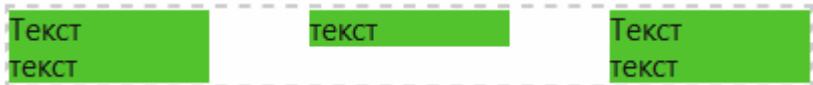
`align-items: stretch;`

Элементы flexbox / grid равномерно распределятся по всей ширине родительского контейнера и растягиваются по высоте



`align-items: flex-start;`

Элементы flexbox выровнены только по ширине родительского контейнера и прижаты к верху



`align-items: flex-end;`

Элементы flexbox выровнены только по ширине родительского контейнера и прижаты к низу



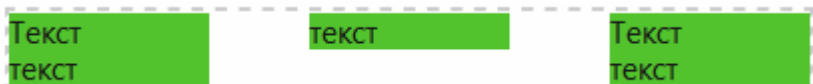
`align-items: center;`

Элементы flexbox / grid выровнены в центре поперечной оси и не растягиваются



`align-items: start;`

Элементы grid выровнены в начале поперечной оси ???



`align-items: end;`

Элементы grid выровнены в конце поперечной оси ???



flex-direction

Определяет способ размещения элементов в контейнере flexbox, задает основную ось и задает направление элементов.

Можно менять направление выстраивания контейнеров.

`flex-direction: row;`

По умолчанию flex выстраивается по горизонтальной оси слева направо:

1	2	3
---	---	---

flex-direction: row-reverse;

3	2	1
---	---	---

flex-direction: column;

flex-direction: column-reverse;

1
2
3

flex-grow

Коэффициент роста элемента контейнера flexbox. Задаёт сколько оставшегося пространства в контейнере flexbox должно быть назначено этому элементу.

Это удобно, когда контейнерам с разным содержимым надо сделать одинаковый width. Можно не ебать мозги вычислением и просто поставить значение 1.

```
flex-grow: 1;
```

flex-grow: 0;

Элемент не будет расти, если есть свободное место. Он будет использовать только то пространство, которое ему необходимо

текст текст текст	текст текст	текст
-------------------	-------------	-------

flex-grow: 1;

Элемент заполнит оставшееся пространство, если ни один другой элемент flexbox не имеет значения flex-grow. Если для всех элементов контейнера flexbox задано flex-grow: 1; то все элементы распределят равномерно оставшееся пространство контейнера

текст текст текст	текст текст	текст
-------------------	-------------	-------

flex-grow: 2;

Значение flex-grow является относительным, его поведение зависит от значения родственных элементов контейнера flexbox.

Если в контейнере находятся два элемента, у одного flex-grow: 1; а у другого flex-grow: 2; то оставшееся в контейнере место распределится пропорционально этим значениям. Элемент с flex-grow: 2; получит в два раза больше оставшегося пространства.

текст текст	текст текст
-------------	-------------

order

Устанавливает порядок размещения элемента в контейнере flex или grid.

Элементы в контейнере обычно сортируются в том порядке, как они записаны в html. Если каждому из них указать своё значение order, то они будут сортироваться в соответствии с этим значением, а не как они записаны в html. Можно использовать положительные и отрицательные целые числа, а так же ноль.

Если order указан не для всех элементов flex-контейнера, то элементы сортируются по порядку их положения в HTML коде, а затем по значению указанному в order.

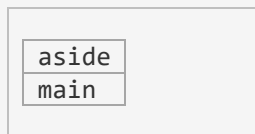
```
order: 1;
```

Примеры

На канале WebDev в [этом](#) видео очень хорошие примеры.

По умолчанию, если в контейнер закинуть другие блочные элементы, то они расположатся один за другим:

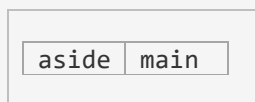
```
<div>
  <aside>
  <main>
</div>
```



display: flex

Если родительскому контейнеру прописать свойство «display: flex», то внутренние блочные элементы будут располагаться по горизонтали. Можно докидывать сколько угодно внутренних блоков, можно создавать любую структуру:

```
<div>
  <aside>
  <main>
</div>
```



Если в одной колонке будет много текста, то вторая (и остальные, если их несколько) автоматом растянутся.

Размеры внутренних блоков в сумме должны равняться ширине родительского контейнера. Свойство ширины width можно указывать в пикселях или в процентах.

Вместо «width» для задания ширины можно использовать свойство «flex-basis» в процентах.

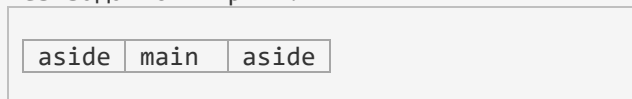
```
div {
  display: flex;
  width: 600px;
}

aside {
  width: 200px;
  width: 30%;
}

main {
  width: 400px;
  width: 70%;
}
```

Если колонкам не указывать ширину, то по умолчанию они будут занимать столько места, сколько им нужно (текст, паддинги):

Без заданной ширины:



justify-content

По умолчанию, элементы внутри контейнера стоят непосредственно рядом друг с другом:

Website	info@mail.com
---------	---------------

Свойство «justify-content» определяет, как элементы внутри flexbox будут выравниваться внутри контейнера. Есть много значений, см. справочник.

```
.header {  
  justify-content: space-between;  
}
```

Website	info@mail.com
---------	---------------

flex-grow

Если остаётся свободное место между элементами, то flex-grow добавит ширины самим элементам пропорционально их исходным размерам. Это нужно, чтобы не ебать мозги и не высчитывать width для нескольких элементов, чтобы они пропорционально смотрелись в контейнере.

align-items

Определяет, как элементы будут выравниваться по вертикали. Например, если в одном контейнере – 3 строки, а в другом – одна, они будут висеть на разном уровне:

aside	main	aside
	main	
	main	

Если прописать свойство «align-items: center», то они выровняются по центру:

aside	main	aside
	main	
	main	

flex-direction

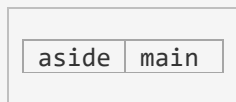
элементы во флексе могут располагаться не только как колонки по ширине. Они также могут располагаться один за другим: «flex-direction: row». Это похоже на стандартное расположение без флекса, но к ним можно добавлять все специальные свойства флекса. Например, выравнивать по ширине.

aside
main
aside

order

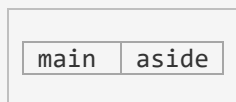
По умолчанию, элементы внутри флекса располагаются так, как они записаны в html:

```
<div>
  <aside>
  <main>
</div>
```



Этот порядок элементов внутри флекса можно менять с помощью свойства «order». Свойство применяется к самим элементам. В примере прописано свойство только одному контейнеру, но на практике надо прописывать порядок всем, потому что они могут переставиться неправильно: элемент без свойства «order» будет идти первым, а остальные с «order» будут идти после него.

```
aside {
  order:2;
}
```



Димон говорит, что порядок меняют для мобильных версий сайта, когда логотип первым не надо ставить.

Старое css

resize

Задаёт возможность менять размеры элемента.

Часто используется для тега `textarea`. Если его прописать соответствующим образом, то окошко с вводом текста нельзя будет растягивать за угол.

```
resize    возможность менять размеры элемента. Чаще используется с textarea
none
vertical
horizontal
both
```

`resize: none;`

Запрещает изменять размеры элемента

`resize: vertical;`

Разрешает изменять размер элемента по вертикали

`resize: horizontal;`

Разрешает изменять размер элемента по горизонтали

`resize: both;`

Разрешает изменять размер элемента по вертикали и горизонтали

Float

float

Делает элемент "плавающим". Размещает элемент с левой или правой стороны его контейнера. Следующие элементы будут обтекать плавающий элемент с другой стороны. Для очистки обтеканий используется свойство clear.

Данное свойство используется все реже. Раньше его часто использовали для создания структуры сайта, но потом на смену пришли flexbox.

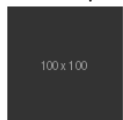
```
float: none;  
float: left;  
float: right;
```

float: none;

Убирает обтекание у элемента

float: left;

Размещает элемент с левой стороны его контейнера. Следующие элементы, в том числе и текст, будут обтекать его с правой стороны

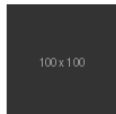


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Eos, fugit facere praesentium inventore. Itaque, minima, similique, soluta, quisquam vel quis cum at iusto dolorum minus ipsa debitis hic veritatis. Illo. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Eos, fugit facere praesentium inventore.

float: right;

Размещает элемент с правой стороны его контейнера. Следующие элементы, в том числе и текст, будут обтекать его с левой стороны

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Eos, fugit facere praesentium inventore. Itaque, minima, similique, soluta, quisquam vel quis cum at iusto dolorum minus ipsa debitis hic veritatis. Illo. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Eos, fugit facere praesentium inventore praesentium inventore.



clear

Запрет обтеканий (float). Перемещает элемент под плавающий элемент. Может применяться как к плавающим элементам, так и к обычным.

```
clear: none;  
clear: left;  
clear: right;  
clear: both;
```

Какая-то хуйня для очистки обтеканий

```
.clearfix:before,  
.clearfix:after {  
  content: "";  
  display: table;  
  width: 100%;  
}
```

```
.clearfix:after {  
  clear: both;  
}
```

clear: none;

Очистки обтеканий нет

clear: left;

Элемент, который идет после элемента с float: left; будет перемещен на новую строку



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Eos, fugit facere praesentium inventore.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sunt, ducimus quis aperiam perspiciatis adipisci esse blanditiis id nostrum consequuntur reprehenderit. Quasi ut magnam laudantium perspiciatis beatae odio quibusdam cum nesciunt.

clear: right;

Элемент, который идет после элемента с float: right; будет перемещен на новую строку

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Eos, fugit facere praesentium inventore.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sunt, ducimus quis aperiam perspiciatis adipisci esse blanditiis id nostrum consequuntur reprehenderit. Quasi ut magnam laudantium perspiciatis beatae odio quibusdam cum nesciunt.

clear: both;

Очистка всех обтеканий



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Eos, fugit facere praesentium inventore.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sunt, ducimus quis aperiam perspiciatis adipisci esse blanditiis id nostrum consequuntur reprehenderit. Quasi ut magnam laudantium perspiciatis beatae odio quibusdam cum nesciunt.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Eos, fugit facere praesentium inventore.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sunt, ducimus quis aperiam perspiciatis adipisci esse blanditiis id nostrum consequuntur reprehenderit. Quasi ut magnam laudantium perspiciatis beatae odio quibusdam cum nesciunt.

CSS выравнивание по центру

align-items

Во Flexbox – выравнивание элементов по поперечной оси.

В grid – выравнивание элементов на оси блока в пределах их области сетки.

```
align-items: center;
```

align-items: stretch;

Элементы flexbox / grid растянутся по всей поперечной оси

align-items: flex-start;

Элементы flexbox выровнены в начале поперечной оси

align-items: flex-end;

Элементы flexbox выровнены в конце поперечной оси

align-items: center;

Элементы flexbox / grid выровнены в центре поперечной оси

align-items: start;

Элементы grid выровнены в начале поперечной оси

align-items: end;

Элементы grid выровнены в конце поперечной оси

text-align

Задаёт выравнивание текста

```
text-align: center;
```

text-align: left;

Выравнивание текста по левому краю

text-align: right;

Выравнивание текста по правому краю

text-align: center;

Выравнивание текста по центру

text-align: justify;

Выравнивание текста по левому и правому краям за исключением последней строки

text-align: start;

Аналогично значению left. Если текст идет слева направо, то текст будет выравниваться по левому краю. Если текст идет справа налево, то текст будет выравниваться по правому краю

text-align: end;

Аналогично значению right. Если текст идет слева направо, то текст будет выравниваться по правому краю. Если текст идет справа налево, то текст будет выравниваться по левому краю

align-content

Определяет, как каждая строка выравнивается внутри контейнера flexbox вдоль поперечной оси или внутри контейнера grid вдоль оси блока. Это применимо, только если присутствует [flex-wrap](#) со значением wrap и если есть несколько строк элементов flexbox / grid.

Куча значений, см. документацию.

```
align-content: center;
```

vertical-align

Выравнивает элемент inline / table-cell по вертикали относительно своего родителя, окружающего текста или ячейки таблицы. Этот стиль работает только с таблицами и инлайн элементами.

```
vertical-align: center;
```

vertical-align: top;

Возможные значения: baseline, sub, super, text-top, text-bottom, middle, top, center, bottom

vertical-align: center;

vertical-align: bottom;

CSS разное

display

Здесь про display вообще, а не про flex.

Меняет поведение элемента в потоке. Можно сделать из строчного элемента блочный.

А также меняет работу некоторых CSS свойств в соответствии с выбранным типом.

```
display: block;
```

```
display: inline;

display: flex;
display: inline-block;

display: none;
```

`display: block;`

`display: inline;`

Задают тип элемента и его поведение в потоке.

`table, flex, grid`

Значения, которые определяют как будут располагаться дочерние элементы блока.

`display: inline-block;`

Гибридный тип элемента, который в потоке ведет себя как строчный, но может принимать CSS свойства для блочных элементов (`width`, `height` и тд.)

`display: none;`

элемент не будет отображаться на странице вообще. Скрытие элемента. Браузер будет вести себя так, будто этого элемента не было.

visibility

Предназначен для отображения или скрытия элемента, включая рамку вокруг него и фон. При скрытии элемента, хотя он и становится не виден, место, которое элемент занимает, остается за ним.

```
visibility: visible | hidden | collapse | inherit
```

`visible`

Отображает элемент как видимый.

`hidden`

Элемент становится полностью прозрачным и продолжает участвовать в форматировании страницы.

`collapse`

Если это значение применяется не к строкам или колонкам таблицы, то результат его использования будет таким же, как `hidden`. В случае использования `collapse` для содержимого ячеек таблиц, то они реагируют, словно к ним было добавлено `display: none`. Иными словами, заданные строки и колонки убираются, а таблица перестраивается по новой. Это значение не поддерживается браузером Internet Explorer.

`inherit`

Наследует значение родителя.

object-fit

определяет, как содержимое заменяемого элемента, такого как `img`, `video`, `embed`, `iframe`, должно заполнять контейнер относительно его высоты и ширины.

Работает схоже как свойство `background-size` для фоновых изображений.

В уроке использовалось для создания слайдера с фотками.

```
object-fit: cover;
```

`object-fit: fill;`

Элемент полностью заполняет область объекта не соблюдая пропорции

object-fit: contain;

Элемент заполняет объект так, чтобы подстроится под область внутри блока пропорционально собственным параметрам

object-fit: cover;

Элемент меняет свой размер таким образом, чтобы сохранять свои пропорции при заполнении блока

object-fit: none;

Элемент не изменяет свой размер с целью заполнить пространство блока

object-fit: scale-down;

Элемент изменяет размер, сравнивая разницу между none и contain, для того, чтобы найти наименьший конкретный размер объекта

object-position

определяет позицию заполняющего элемента при использовании object-fit

```
object-fit: cover;
```

object-position: right top;

Используя ключевые слова, такие как top, right, bottom, left. Первый параметр - позиция по оси X, второй - позиция по оси Y

object-position: 50% 50%;

Используя проценты, где 0% по оси X - левый край блока, а 100% - правый и 0% по оси Y - верхний край блока, а 100% - нижний. Первый параметр - позиция по оси X, второй - позиция по оси Y

object-position: 75px 30px;

Используя единицы измерения такие как px, em, rem, где начальная точка координат левый верхний угол блока. Первый параметр - позиция по оси X, второй - позиция по оси Y

</>

? полоска под элементом через ::before

Вот так можно «нарисовать» полоску под элементом (используя after) или над элементом (before):

```
h1::after {
  content: "";
  display: block;
  width: 50px;
  height: 3px;

  background-color: blue;
}
```

Ещё один способ без использования position:

```
.intro__title::after {
  content: "";
  display: block;
  width: 60px;
  height: 3px;
  margin: 0 auto;
  background-color: #fff;
}
```

::first-letter

применяет стили к первой букве первой строки блочного элемента, но только если нету другого предшествующего содержимого (такого как изображения или инлайн таблицы).

Знаки препинания до или после первой буквы также считаются этой первой буквой.

```
p:first-letter {  
  color: red;  
}
```

::first-line

::cue

::selection

::slotted

::backdrop

::placeholder

Float (кратко)

Если изображению прописать float, то текст его будет обтекать:

```
img { float: right }
```

Хорошее видео у webdev [здесь](#).

Лучше смотреть видео.

Схлопывание родителя.

Clear: both;

Clear: fix;

Если внутри одного контейнера разместить 2 дива, по умолчанию они стоят друг под другом.

Если одному прописать float, то он сожмётся по величине контента (или по заданной величине), а второй див будет его обтекать (справа и снизу, если он больше по размеру).

```
<div class="container">  
  <div class="item red">1</div>  
  <div class="item green">2</div>  
</div>
```

```
.container {  
  width: 400px;  
  height: 200px;  
  padding: 20px;  
  background-color: papayawhip;  
}
```

```
.item {  
  font-size: 50px;  
  margin: 0 auto;  
}
```

```
.red {  
  background-color: red;  
  float: left;  
}
```

```
.green {  
  background-color: green;  
  height: 2em;  
}
```

CSS learn.js

Единицы измерения: px, em, rem

Пиксель px – это базовая единица измерения.

Количество пикселей задаётся в настройках разрешения экрана. Все значения браузер в итоге пересчитает в пиксели. Пиксели могут быть дробными: 16.5px. При окончательном отображении дробные пиксели округляются и становятся целыми.

1em – текущий размер шрифта.

Можно брать любые пропорции от текущего шрифта: 2em, 0.5em и т.п. Размеры в em – относительные, они определяются по текущему контексту. Если и у родителя, и у потомка размер 1.5em, то у у потомка он будет больше, потому что принимает за единицу размер родителя.

rem

Задаёт размер относительно размера шрифта, указанного для элемента <html>.

Элементы, размер которых задан в rem, не зависят друг от друга и от контекста – и этим похожи на px, а с другой стороны они все заданы относительно размера шрифта <html>.

Единица rem не поддерживается в IE8-.

Проценты %

Относительные единицы. Как правило, процент будет от значения свойства родителя с тем же названием, но не всегда:

- при установке свойства margin-left, процент берётся от *ширины* родительского блока, а не от его margin-left.
- при установке свойства line-height в %, процент берётся от текущего *размера шрифта*, а вовсе не от line-height родителя.
- для width/height обычно процент от ширины/высоты родителя, но при position:fixed, процент берётся от ширины/высоты *окна*.

Относительно экрана: vw, vh, vmin, vmax

Эти значения были созданы, в первую очередь, для поддержки мобильных устройств. Их основное преимущество – в том, что любые размеры, которые в них заданы, автоматически масштабируются при изменении размеров окна.

vw – 1% ширины окна

vh – 1% высоты окна

vmin – наименьшее из (vw, vh), в IE9 обозначается vm

vmax – наибольшее из (vw, vh)

Что понимать под размером шрифта

Она обычно чуть больше, чем расстояние от верха самой большой буквы до низа самой маленькой. То есть, предполагается, что в эту высоту помещается любая буква или их сочетание. Но при этом «хвосты» букв, таких как p, g могут заходить за это значение, то есть вылезать снизу. Поэтому обычно высоту строки делают чуть больше, чем размер шрифта.

display

Свойство display (CSS) определяет тип отображения (display type) элемента, имеющий два основных свойства, определяющих генерацию боксов (бокс - это прямоугольная область, являющаяся изображением элемента):

- внешний тип отображения определяет расположение самого бокса в схеме потока (flow layout);
- внутренний тип отображения определяет расположение дочерних элементов бокса.

Значений много, вот основные:

```
display: none;

display: block;
display: inline;
display: inline-block;

display: flex;
display: grid;
display: table;
```

none

Самое простое значение. Элемент не показывается, вообще. Как будто его и нет.

block

Блок стремится расшириться на всю доступную ширину. Можно указать ширину и высоту явно.

Блочные элементы располагаются один над другим, вертикально (если нет особых свойств позиционирования, например float). Блоки прилегают друг к другу вплотную, если у них нет margin.

Это значение многие элементы имеют по умолчанию: <div>, заголовок <h1>, параграф <p>.

inline

Элементы располагаются на той же строке, последовательно.

Ширина и высота элемента определяются по содержимому. Поменять их нельзя.

Например, инлайн-элементы по умолчанию: , <a>.

inline-block

Это значение – означает элемент, который продолжает находиться в строке (inline), но при этом может иметь важные свойства блока. Это значение display используют, чтобы отобразить в одну строку блочные элементы, в том числе разных размеров. Свойство vertical-align позволяет выровнять такие элементы внутри внешнего блока.

Как и инлайн-элемент:

- Располагается в строке.
- Размер устанавливается по содержимому.

Во всём остальном – это блок, то есть:

- Элемент всегда прямоугольный.
- Работают свойства width/height.

table-*

Современные браузеры (IE8+) позволяют описывать таблицу любыми элементами.

Для таблицы целиком table, для строки – table-row, для ячейки – table-cell и т.д.

С точки зрения современного CSS, обычные <table>, <tr>, <td> и т.д. – это просто элементы с предопределёнными значениями display.

Внутри ячеек table-cell свойство vertical-align выравнивает содержимое по вертикали. Это можно использовать для центрирования.

flex-box

Flexbox позволяет удобно управлять дочерними и родительскими элементами на странице, располагая их в необходимом порядке.

float

<https://learn.javascript.ru/float>

position

Позволяет сдвигать элемент со своего обычного места. Основные значения:

```
position: static
position: relative
position: absolute
position: fixed
```

position: static

производится по умолчанию, в том случае, если свойство position не указано. Такая запись встречается редко и используется для переопределения других значений position.

position: relative

Относительное позиционирование сдвигает элемент относительно его обычного положения. Необходимо указать элементу CSS-свойство position: relative и координаты left/right/top/bottom.

position: absolute

Абсолютное позиционирование делает две вещи:

1. Элемент исчезает с того места, где он должен быть и позиционируется заново. Остальные элементы, располагаются так, как будто этого элемента никогда не было.
2. Координаты top/bottom/left/right для нового местоположения отсчитываются от ближайшего позиционированного родителя, т.е. родителя с позиционированием, отличным от static. Если такого родителя нет – то относительно документа.

Кроме того:

1. Ширина элемента с position: absolute устанавливается по содержимому.
2. Элемент получает display:block, который перекрывает почти все возможные display.
3. В абсолютно позиционированном элементе можно одновременно задавать противоположные границы left/right, top/bottom. Браузер растянет такой элемент до границ.

Важное отличие от relative: так как элемент удаляется со своего обычного места, то элементы под ним сдвигаются, занимая освободившееся пространство.

Иногда бывает нужно поменять элементу position на absolute, но так, чтобы элементы вокруг не сдвигались. Как правило, это делают, меняя соседей – добавляют margin/padding или вставляют в документ пустой элемент с такими же размерами.

position: fixed

Позиционирует объект точно так же, как absolute, но относительно window.

Когда страницу прокручивают, фиксированный элемент остаётся на своём месте и не прокручивается вместе со страницей.

Центрирование горизонтальное и вертикальное

Горизонтальное

text-align

Для центрирования инлайновых элементов – достаточно поставить родителю text-align: center

Для центрирования блока это уже не подойдёт, свойство просто не подействует.

margin: auto

Блок по горизонтали центрируется через margin: auto

Значение margin-left:auto/margin-right:auto заставляет браузер выделять под margin всё доступное сбоку пространство. А если и то и другое auto, то слева и справа будет одинаковый отступ, таким образом элемент окажется в середине.

Вертикальное

Вертикальное центрирование изначально не было предусмотрено в спецификации CSS и по сей день вызывает ряд проблем. Есть три основных решения.

position:absolute + margin

Центрируемый элемент позиционируем абсолютно и опускаем до середины по вертикали при помощи `top:50%` и смещения `margin` на половину ширины элемента.

Одна строка: `line-height`

Вертикально отцентрировать одну строку в элементе с известной высотой `height` можно, указав эту высоту в свойстве `line-height`. Это работает, но лишь до тех пор, пока строка одна, а если содержимое вдруг переносится на другую строку, то начинает выглядеть довольно уродливо.

Таблица с `vertical-align`

В таблицах свойство `vertical-align` указывает расположение содержимого ячейки. С `vertical-align: middle` содержимое находится по центру. Таким образом, можно обернуть нужный элемент в таблицу размера `width:100%;height:100%` с одной ячейкой, у которой указать `vertical-align:middle`, и он будет отцентрирован.

Но мы рассмотрим более красивый способ, который поддерживается во всех современных браузерах, и в IE8+. В них не обязательно делать таблицу, так как доступно значение `display:table-cell`. Для элемента с таким `display` используются те же алгоритмы вычисления ширины и центрирования, что и в TD. И, в том числе, работает `vertical-align`. Этот способ замечателен тем, что он не требует знания высоты элементов.

```
<div style="display: table-cell; vertical-align: middle; ... >
```

Можно и в процентах, завернув «псевдоячейку» в элемент с `display:table`, которому и поставим ширину:

```
<div style="display: table; width: 100%">
  <div style="display: table-cell; vertical-align: middle; height: 100px; border: 1px solid blue">
    <button>Кнопка<br>с любой высотой<br>и шириной</button>
  </div>
</div>
```

Центрирование в строке с `vertical-align`

Для инлайн-элементов (`display:inline/inline-block`), включая картинки, свойство `vertical-align` центрирует сам инлайн-элемент в окружающем его тексте.

```
vertical-align:
baseline      по умолчанию
middle        по середине
sub           вровень с <sub>
super         вровень с <sup>
text-top      верхняя граница вровень с текстом
text-bottom   нижняя граница вровень с текстом
```

[Центрирование с `vertical-align` без таблиц](#)

Какая-то ёбань, см. по ссылке.

С использованием модели flexbox

```
.outer {
  display: flex;
  justify-content: center; /*Центрирование по горизонтали*/
  align-items: center;     /*Центрирование по вертикали */
}
```

Свойства font-size и line-height

font-size – *размер шрифта*, в частности, определяющий высоту букв.

line-height – *высота строки*.

Размер шрифта обычно равен расстоянию от самой верхней границы букв до самой нижней, исключая «нижние хвосты» букв, таких как p, g. При размере строки, равном font-size, строка не будет размером точно «под букву». В зависимости от шрифта, «хвосты» букв при этом могут вылезать, правые буквы Ё и р в примере выше пересекаются как раз поэтому.

Обычно размер строки делают чуть больше, чем шрифт. По умолчанию в браузерах используется специальное значение line-height: normal. Как правило, оно будет в диапазоне 1.1 - 1.25, но стандарт не гарантирует этого, он говорит лишь, что оно должно быть «разумным» (дословно – англ. reasonable).

Множитель для line-height

Значение line-height можно указать при помощи px или em, но гораздо лучше – задать его числом.

Значение-число интерпретируется как множитель относительно размера шрифта. Например, значение с множителем line-height: 2 при font-size: 16px будет аналогично line-height: 32px (=16px*2).

Установить font-size и line-height можно **одновременно**.

Соответствующий синтаксис выглядит так:

минимум

```
font: 20px/1.5 Arial,sans-serif;
```

максимум

```
font: italic bold 20px/1.5 Arial,sans-serif;
```

Свойство white-space

Свойство outline

Свойство box-sizing

Без сортировки

Трансформация

Transform – свойство, которое отвечает за трансформацию.

Можно указывать сразу несколько трансформаций в одном свойстве transform: наклон, сдвиг, поворот и масштаб.

С помощью **transform-origin** можно указать точку, вокруг которой будет вращаться объект. По умолчанию это центр.

Можно делать анимацию с помощью :hover и transition:

```
.square {  
  transform: rotate(45deg) translate(10px);  
  transition: .2s linear;  
}  
  
.square:hover {  
  transform: rotate(0) translate(10px);
```

```
}
```

Важно, что если надо изменить какой-то один параметр трансформации (например, поворот), а остальные оставить как было, то в `:hover` их всё равно надо прописать, иначе их значения будут сбрасываться.

Чтобы вращение отображалось реалистично, надо использовать свойство **perspective**. Оно прописывается для контейнера, в котором находится вращающийся объект, а не самому объекту:

```
<div class="composition">
  <div class="square"></div>
</div>

.composition {
  perspective: 500px;
}

.square {
  transform: rotateX(45deg);
  transform-origin: center;
  transition: .2s linear;
}

.square:hover {
  transform: rotateX(0);
}
```

transform

Позволяет увеличивать / уменьшать, вращать, искажать и передвигать элементы страницы. Атрибутов очень много, лучше смотреть документацию.

```
transform: scale(1);
transform: rotate(45deg);
transform: skewX(25deg);
transform: translate(10px, 10%);

transform: rotate(45deg) translate(10px) skewX(10deg) scale(0.5);

transform: translate3d(10px 10px 10px);

transform: rotate3d(1, 1, 1, 45deg) translate3d(0, 0, 30px);
```

transform: translate3d

позволяет указать сразу 3 координаты x, y и z для смещения элемента. Это то же самое, что прописать `translateX`, `translateY`, `translateZ` отдельно.

transform: rotate3d

```
transform: rotate3d(1, 1, 1, 45deg);
```

Вращает элемент в трехмерном пространстве. Первые три параметра - множители вращения для осей X, Y, Z, которые будут умножены на значение четвертого параметра. В на [MDN](#) написано, что их значения могут быть между 0 и 1. Изменение этих векторов повлияет на дальнейшее поведение объекта: он будет передвигаться по изменённым осям.

`rotate3d(1, 1, 1, 45deg);` - вращает элемент по осям X, Y, Z на 45 градусов

`rotate3d(0, 1, 1, 45deg);` - Вращает элемент по осям Y, Z на 45 градусов

`rotate3d(2, 1, 1, 45deg);` - Вращает элемент по осям Y, Z на 45 градусов и по оси X на 90 градусов ($45 \cdot 2$)

transform-origin

Свойство transform-origin устанавливает координаты точки, относительно которой будет происходить трансформация элемента. По умолчанию координата точки установлена по центру трансформируемого элемента.

```
transform-origin: center;
transform-origin: left top;
transform-origin: 10px 10px;
transform-origin: 10% 10%;
```

transform-origin: center center;

Первый параметр - координата точки трансформации по **оси X**, второй - координата точки трансформации по **оси Y**. Если первое и второе совпадают, можно указать только одно значение, которое применится к двум осям.

Доступные значения по **оси X** - *left, center, right*

Доступные значения по **оси Y** - *top, center, bottom*

transform-origin: left top; transform-origin: 50px 50px;

Можно указывать координаты точки используя единицы измерения **px, em, rem** и другие. Первый параметр - координата точки по **оси X**, второй по **оси Y**. Начальные координаты берутся от верхнего левого угла элемента и равны 0, 0.

transform-origin: 100% 50%;

Можно указывать координаты точки используя %. Первый параметр - координата точки по **оси X**, второй по **оси Y**. Начальные координаты берутся от верхнего левого угла элемента и равны 0, 0.

transform-style

определяет положение дочернего элемента в 3D-пространстве или в той же плоскости, что и родительский элемент. Используется когда нужно трансформировать элементы внутри трансформируемого элемента, например при создании 3D куба.

```
transform-style: flat;
transform-style: preserve-3d;
```

transform-style: flat;

Дочерний элемент лежит в той же плоскости, что и родительский.

transform-style: preserve-3d;

Дочерний элемент имеет свое собственное 3D-пространство.

perspective

создает эффект перспективы и добавляет глубину для 3D трансформируемых элементов.

Оптимальные значения – от 500px до 600px. Прописывается для контейнера, в котором находятся трансформируемые элементы, но не для самих элементов.

```
<div class="composition">
  <div class="square"></div>
</div>

.composition {
  perspective: 500px;
}
```

perspective-origin

Задаёт координаты точки, куда смотрит наблюдатель. Это свойство работает совместно со свойством perspective и определяет точку сходимости линий при перспективе.

Применяется к трансформируемым элементам.

```
perspective-origin: <позиция>
```

```
/* One-value syntax */
```

```
perspective-origin: x-position;
```

```
/* Two-value syntax */
```

```
perspective-origin: x-position y-position;
```

```
/* When both x-position and y-position are keywords, the following is also valid */
```

```
perspective-origin: y-position x-position;
```

backface-visibility

Определяет, видно ли заднюю грань элемента, когда он повернут к пользователю. Применяется при работе с трансформацией, например при создании вращающихся 3D карточек, где при вращении нужно чтобы пользователь видел только одну грань карточки.

```
backface-visibility: hidden;
```

```
backface-visibility: visible;
```

```
backface-visibility: visible;
```

Пользователь видит все грани элемента, даже если повернут задней гранью к пользователю.

```
backface-visibility: hidden;
```

Пользователь не видит все грани элемента. Если элемент повернут к пользователю своей задней гранью, то он будет скрыт.

...

Примеры реализации

Навигация

```
.html
<nav class="nav">
  <a class="nav-link" href="#">Nav link</a>
  <a class="nav-link" href="#">Nav link</a>
  <a class="nav-link" href="#">Nav link</a>
</nav>
```

```
.css
.nav {
  display: flex;
  justify-content: center;
  width: 700px;
  margin: 50px auto;
  position: relative;
}

.nav:before {
  content: "";
  width: 100%;
  height: 100%;
  background-color: #2133c1;
  position: absolute;
  top: 0;
  left: 0;
  z-index: -1;
  transform: skewX(-45deg);
}

.nav-link {
  flex-grow: 1;
  padding: 15px 35px;
  position: relative;
  color: #fff;
  text-decoration: none;
  text-align: center;
}

.nav-link:before {
  content: "";
  width: 100%;
  height: 100%;
  position: absolute;
  top: 0;
  left: 0;
  z-index: -1;
  transform: skewX(-45deg);
  transition: background .2s linear;
}

.nav-link:hover:before {
  background-color: #0d1c8d;
}
```

.nav

display: flex — чтобы ссылки выстраивались в ряд, а не друг под другом.

position: relative — нужно для позиционирования псевдоэлемента.

.nav:before

position: absolute – чтобы этот элемент был ровно за родительским элементом.

transform: skewX(-45deg) – всё ради этого эффекта. Родительский элемент не трансформируется, только дочерний псевдоэлемент, служащий для него фоном.

.nav-link

flex-grow: 1 – расстояние между элементами будет автоматом заполнено «воздухом».

position: relative – нужно для позиционирования дочернего элемента.

.nav-link:before

position: absolute – чтобы разместить элемент прямо за родителем

transform: skewX(-45deg) – всё ради этого эффекта.

Карточка

```
.html
<div class="card-wrapper">

  <div class="card">
    <div class="card-front">
      
    </div>
    <div class="card-back">
      <div class="card-text">Text</div>
    </div>
  </div>
</div>
```

```
.css

.card-wrapper {
  width: 300px;
  height: 450px;
  margin: 100px auto;
  perspective: 500px;
}

.card {
  width: 100%;
  height: 100%;
  box-shadow: 0 10px 30px rgba(0, 0, 0, .2);
  border-radius: 10px;
  transition: transform .2s linear;
  transform-style: preserve-3d;
}

.card-wrapper:hover .card {
  transform: rotateY(180deg);
}

.card-front,
.card-back {
  width: 100%;
  height: 100%;
  position: absolute;
  backface-visibility: hidden;
}

.card-back {
  transform: rotateY(-180deg);
}

.card-image {
  display: block;
  border-radius: 10px;
}
```

.card-wrapper

Остаётся неподвижным, поэтому не глючит, когда на него наводишь курсор. Определяет размеры всей карты.
perspective – придание объёма при вращении. Прописывается родительскому элементу.

.card

Этот контейнер будет вращаться при наведении на обёртку.

transition – для плавности поворота.

transform-style – чтобы этот контейнер вращался вместе со своими наследниками, а не в отрыве от них.

.card-wrapper:~hover .card

При наведении на обёртку будет вращаться его наследник: карточка.

.card-front,

.card-back

position: absolute – чтобы контейнеры, которые записаны в HTML друг под другом, выбились из потока и располагались внутри контейнера-карты. Если этого не сделать, то они могут оказаться ниже карточки, за её пределами.

backface-visibility – чтобы не было видно оборотную сторону.

.card-back

Со старта должна быть развёрнута на 180, потому что это оборотная сторона.

Масштабирование элемента

.html

```
<div class="card">

  <div class="card-header">
    
  </div>

  <div class="card-content">
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>
  </div>

</div>
```

.css

```
.card {
  width: 300px;
  margin: 50px auto;
  background-color: #fff;
  box-shadow: 0 10px 30px rgba(0, 0, 0, 0.3);
}

.card:~hover .card-header img {
  opacity: 0.3;
  transform: scale(1.5);
}
```

```
.card-header {
  height: 200px;
  position: relative;
  overflow: hidden;
  background-color: #000;
}
```

```
.card-header img {
  display: block;
  position: absolute;
  top: 0;
  left: 0;
  z-index: 1;
}
```

```

    transition: transform 2s linear, opacity 1s linear;
}

.card-content {
    padding: 20px;
}

```

.card: hover .card-header img

Формат записи типа: «куда навести, чтобы сработало» => «к чему применять стили»

overflow: hidden;

Картинка будет увеличиваться в размерах. Чтобы она не вылезла за границы своей рамки-хедера, прописывается это.

transition: transform 2s linear, opacity 1s linear;

Прописывается одновременно для двух свойств.

Аватар и карточка

```

.html
<div class="card">

    <div class="card-content">
        <h3 class="card-title">Card title</h3>
        <div class="card-text"> <p>Text</p> </div>
    </div>
</div>

.css
.card {
    width: 300px;
    margin: 100px auto;
    position: relative;
    background-color: #fff;
    box-shadow: 0 1px 3px rgba(0, 0, 0, .1);
    border-radius: 10px;
    transition: transform .2s linear, box-shadow .2s linear;
}

.card: hover {
    transform: translateY(-10px);
    box-shadow: 0 5px 20px rgba(0, 0, 0, .1);
}

.card-avatar {
    width: 150px;
    height: 150px;
    border-radius: 50%;
    position: absolute;
    left: 50%;
    transform: translate3d(-50%, -50%, 0);
}

.card-content {
    padding: 100px 20px 20px;
}

.card-title {
    margin: 0 0 15px;
}

```

```
font-size: 24px;
color: #d02424;
}

.card-text {
font-size: 14px;
}
```

.card

position: relative

потому что внутри будут элементы со значением absolute.

.card:hover

transform: translateY(-10px);

Этот эффект смещения будет приподнимать всю карточку.

.card-avatar {

position: absolute;

Аватар должен быть выбит из общего потока, чтобы поднять его наполовину вверх и разместить в середине контейнера.

left: 50%;

transform: translate3d(-50%, -50%, 0);

Смещение на нужную позицию происходит с двумя этими параметрами.

.card-content

padding: 100px 20px 20px;

Ничего интересного, кроме такого паддинга, чтобы отделить контент от аватара.

Куб 3д

```
.html
<div class="scene">

  <div class="coub">
    <div class="coub-part front">1</div>
    <div class="coub-part back">2</div>
    <div class="coub-part left">3</div>
    <div class="coub-part right">4</div>
    <div class="coub-part bottom">5</div>
    <div class="coub-part top">6</div>
  </div>
</div>

.CSS
.scene {
  perspective: 500px;
  perspective-origin: center;
}

.coub {
  width: 200px;
  height: 200px;
  margin: 150px auto;

  transform: rotateX(-5deg) rotateY(-30deg);
  transform-style: preserve-3d;
}
```

```

.coub-part {
  width: 100%;
  height: 100%;

  font-size: 50px;
  color: #fff;
  text-align: center;
  line-height: 200px;

  opacity: .8;
  position: absolute;
}

.coub-part.front {
  background-color: #30c924;
  transform: translateZ(100px);
}

.coub-part.back {
  background-color: #30c924;
  transform: translateZ(-100px) rotateY(-180deg);
}

.coub-part.left {
  background-color: #2d4dd6;
  transform: rotateY(-90deg) translateZ(100px);
}

.coub-part.right {
  background-color: #2d4dd6;
  transform: rotateY(90deg) translateZ(100px);
}

.coub-part.bottom {
  background-color: #cc1a1a;
  transform: rotateX(-90deg) translateZ(100px);
}

.coub-part.top {
  background-color: #cc1a1a;
  transform: rotateX(90deg) translateZ(100px);
}

```

Анимация

@keyframes

At-правило, которое устанавливает ключевые кадры при анимации элемента. Анимация представляет собой плавный переход стиливых свойств от одного ключевого кадра к другому. Вычисление промежуточных значений между такими кадрами берёт на себя браузер.

[MDN](#).

Внутри стилей для элемента прописывается свойство **animation-name**. Это будет идентификатор, с помощью которого браузер понимает, к чему и как применять анимацию.

Также, в правилах элемента прорисовываются свойства анимации. Например, будет ли она циклична, с какого момента будет проигрываться, будет ли у неё задержка и т.д.

Минимально необходимые свойства для запуска анимации: **animation-name** и **animation-duration**.

Внутри **@keyframes** указывается только то, в какой момент анимации и как будут меняться CSS правила.

В примере ниже указано, что на старте у квадрата будут нолевые показатели правил **transform** и **background**.

К 50% анимации появится **border-radius**.

К 100% произойдут соответствующие **transform**-изменения и исчезнет **border-radius**. Если на каком-то проценте не прописано свойство, значит его нет.

Если квадрату прописать цвет в 0% анимации, то квадрат в него окрасится только при старте анимации. Если для анимации установлена задержка, то квадрата будет белым до старта анимации. Поэтому цвет прописан и в стилях, и в 0% анимации. Это можно решить через **animation-fill-mode**, но сейчас не об этом.

В примере я также указал проценты, но можно указать ключевые слова from (начало) и to (конец).

```
.html
<div class="square"></div>

.css
.square {
  width: 200px;
  height: 200px;
  margin: 20px;

  background-color: #de3232;

  animation-name: translateX;
  animation-duration: 10s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
  animation-timing-function: linear;
  animation-delay: 2s;
  animation-play-state: running;
}

@keyframes translateX {
  0% {
    transform: translateX(0) translateY(0) rotate(0deg) scale(1);
    background-color: #de3232;
  }
  50% {
    border-radius: 50%;
  }
  100% {
    transform: translateX(500px) translateY(100px) rotate(180deg) scale(0.5);
    background-color: #0a40b9;
  }
}
```

animation-fill-mode

Определяет, как нужно применять стили к объекту анимации до и после ее выполнения

```
animation-fill-mode: backwards;
animation-fill-mode: both;
```

animation-fill-mode: none;

Стили анимации не будут применены к элементу до и после ее выполнения

animation-fill-mode: forwards;

По окончании анимации элемент сохранит стили последнего ключевого кадра

animation-fill-mode: backwards;

Стили анимации первого кадра будут применены к элементу до ее запуска

animation-fill-mode: both;

Стили к элементу применяются до и после воспроизведения анимации, как если бы значения forwards и backwards были заданы одновременно

animation-fill-mode: both, forwards;

Применяет свойство animation-fill-mode для разных анимаций элемента, если их указано несколько

Геометрические фигуры

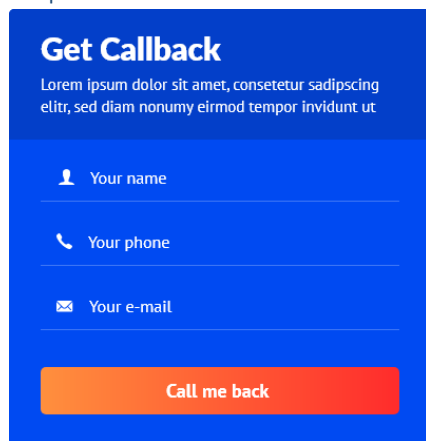
<https://www.youtube.com/watch?v=lrj55RYH254>

https://www.youtube.com/watch?v=3v_5BZK_L0

<https://www.youtube.com/watch?v=Z7J6xtyimSs>

Примеры реализации разных элементов

Форма



The image shows a blue rectangular form with a white header area. The header contains the title 'Get Callback' in bold white text, followed by a line of placeholder text: 'Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut'. Below the header, there are three input fields, each with a small icon (person, phone, and envelope) and a placeholder text ('Your name', 'Your phone', 'Your e-mail'). At the bottom of the form is a large orange button with the text 'Call me back' in white.

```
.html
<div class="request-form request-form--intro">

  <div class="request-form__header">
    <h3 class="request-form__title">Get Callback</h3>
    <div class="request-form__text">Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
nonumy eirmod tempor invidunt ut</div>
  </div>

  <div class="request-form__content">
    <form class="" action="/" method="post">
      <div class="form__group">
        <input class="input input--user" type="text" placeholder="Your name">
      </div>
      <div class="form__group">
        <input class="input input--phone" type="tel" placeholder="Your phone">
      </div>
      <div class="form__group">
        <input class="input input--email" type="email" placeholder="Your e-mail">
      </div>

      <button class="btn btn--block btn--orange" type="submit">Call me back</button>

    </form>
  </div>
</div>
<!-- / Request Form -->
```

```
.CSS
/* Request-form */
.request-form {
  width: 100%;
  max-width: 400px;
  background-color: #004AF2;
  border-radius: 5px;
  overflow: hidden;
```

```

}

.request-form__header {
    padding: 20px 30px;
    background-color: #033FC9;
}

.request-form__title {
    font-family: 'Lato', sans-serif;
    font-size: 30px;
    font-weight: 900;
    color: #fff;
}

.request-form__text {
    font-size: 15px;
    color: #fff;
    line-height: 1.4;
}

.request-form__content {
    padding: 30px;
}

.request-form--intro {
    position: relative;
    bottom: -30px;
}

/* Form */
.form__group {
    margin-bottom: 25px;
}

.input {
    display: block;
    width: 100%;
    padding-bottom: 12px;
    padding-left: 50px;
    background: none;
    border: none;
    border-bottom: 1px solid rgba(255, 255, 255, .25);
    font-family: 'PT Sans', sans-serif;
    font-size: 16px;
    color: #fff;
    transition: border-color .1s linear;
}

.input::placeholder {
    color: #fff;
}

.input:focus {
    outline: none;
    border-bottom: 1px solid #fff;
}

.input--user {
    background: url("../images/user-icon.svg") left 15px top 5px no-repeat;
}

.input--phone {
    background: url("../images/phone-icon.svg") left 15px top 5px no-repeat;
}

.input--email {
    background: url("../images/mail-icon.svg") left 15px top 5px no-repeat;
}

/* BTN */
.btn {
    display: inline-block;

```

```

vertical-align: top;
padding: 11px 20px;
font-family: 'PT Sans', sans-serif;
font-size: 18px;
line-height: 1.1;
color: #fff;
font-weight: 700;
text-align: center;
cursor: pointer;
background: #333;
border: 0;
border-radius: 5px;
}

.btn--orange {
  background-image: linear-gradient(to right, #FF903E, #FF2C2C);
}

.btn--block {
  display: block;
  width: 100%;
}

```

Пунктирная линия

```

.benefits-list {
  width: 100%;
  max-width: 370px;

  background-image: linear-gradient(to bottom, #fff 40%, #A5A5A5 40%);
  background-size: 2px 35px;
  background-repeat: repeat-y;
  background-position: left 12px top;
}

```

Двухцветная полоса

```

.mission {
  position: relative;
}

.mission::before,
.mission::after {
  content: "";
  display: block;
  width: 41.66666%;
  height: 2px;
  position: absolute;
  bottom: 0;
  z-index: 1;
}

.mission::before {
  background-color: #004AF2;
  right: 50%;
}

.mission::after {
  background-color: #FC2C2B;
  left: 50%;
}

```

Модальные окна

Стили для модальных окон прописываются в самом конце, перед закрывающим `/body`.

При появлении модального окна экран закрывается тёмной **маской**.

```
.modal {
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, .9);
  position: fixed;
  top: 0;
  left: 0;
  z-index: 2000;
}
```

Скролл у страницы тоже нужно убрать. Делается это через класс `body.no-scroll`, который будет присваиваться для `body` через JS.

```
body.no-scroll {
  overflow: hidden;
}
```

`.modal` – это маска на весь экран.

`.modal__inner` – ячейка, которая идёт на всю высоту модального окна.

`.modal__content` – наполнение окна.

```
.html
<!-- Modals -->
<div class="modal">

  <div class="modal__inner"></div>
  <button class="modal__close"></button>

</div>

.CSS
.modal {
  /* display: none */
  display: block;
  width: 100%;
  height: 100%;
  padding: 30px 15px;
  overflow-y: auto;
  background-color: rgba(0, 0, 0, .9);
  position: fixed;
  top: 0;
  left: 0;
  z-index: 2000;
}

.modal__inner {
  display: flex;
  justify-content: center;
  align-items: center; /* чтобы на всю высоту не растягивалось */
  min-height: 100%;
}

.modal__content {
  width: 100%;
  max-width: 400px;
  padding: 25px;
```

```

position: relative;
background-color: #fff;
}

.modal__close {
width: 20px;
height: 20px;
padding: 0;

background: none;    /* убрать фон за крестом */
border: none;
cursor: pointer;

position: absolute;
top: -20px;
right: -30px;
z-index: 1;
}

.modal__close img {
display: block;      /* чтобы не было лишних отступов */
}

```

display: block;

flex-direction: column;

На практике наверное не понадобится, но для уверенности, что если у маски появятся ещё дочерние элементы, то они будут располагаться вертикально.

position: relative;

В .modal__content это нужно для того, чтобы свободно позиционировать закрывающую кнопку-крест.

overflow-y: auto;

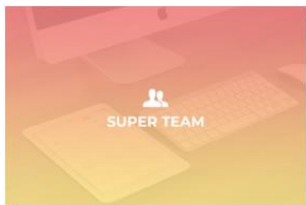
Если модальное окно не будет влезать по размерам в экран, то у него появится скролл, чтобы его прокрутить и посмотреть полностью.

Итоговая реализация:

.html

.css

Mogo плитки



```

.html
<div class="about">

  <div class="about__item">
    <div class="about__img">
      
    </div>
    <div class="about__text">
      
      <p>super team</p>
    </div>
  </div>

```

```

    </div>
</div>

<div class="about__item">
  <div class="about__img">
    
  </div>
  <div class="about__text">
    
    <p>super headphones</p>
  </div>
</div>

<div class="about__item">
  <div class="about__img">
    
  </div>
  <div class="about__text">
    
    <p>super cactus</p>
  </div>
</div>
</div>

```

```

.CSS
.about {
  margin-top: 80px;
  display: flex;
  justify-content: space-between;
}

.about__item {
  width: 380px;
  position: relative;
  background-color: #95e1d3;
}

.about__img {
  background: linear-gradient(to bottom, #f38181, #fce38a);
  transition: transform .2s linear;
}

.about__img img {
  display: block;
  transition: opacity .1s linear;
}

.about__text {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  width: 100%;
  height: 100%;

  font-size: 18px;
  font-weight: 700;
  color: #fff;

  opacity: 0;

  position: absolute;
  top: 0;
  left: 0;
  z-index: 2;

  transition: .2s linear;
}

.about__item:hover .about__img {
  transform: translate3d(-10px, -9px, 0);
}

```

```
}

.about__item:hover .about__img img {
  opacity: .1;
}

.about__item:hover .about__text {
  opacity: 1;
  transform: translate3d(-10px, -9px, 0);
}
```

Формы

[Руководство по HTML-формам](#)

[Отправка данных формы](#)

Все элементы формы [здесь](#).

Часто используемые с формами HTML-структуры

оборачивать лейбл и виджет формы в элемент `<div>` — это общепринятая практика.

Элемент `<p>` также часто используется, как и HTML-списки.

В добавок к элементу `<fieldset>` часто используют HTML-заголовки (например, `<h1>`, `<h2>`) и `<section>` для структурирования сложных форм.

Элементы формы

<code><form></code>	раздел документа, содержащий интерактивные элементы управления
<code><button></code>	самостоятельная кнопка
<code><datalist></code>	содержит набор опций, доступных для выбора. значение будет установлено для <code>input</code> .
<code><fieldset></code>	используется для группировки нескольких элементов управления без веб-форм.
<code><input></code>	для создания интерактивных элементов управления для получения данных пользователя
<code><label></code>	подпись к элементу пользовательского интерфейса
<code><legend></code>	заголовок содержания родительского элемента <code>fieldset</code>
<code><meter></code>	
<code><optgroup></code>	позволяет группировать опции, находящиеся внутри элемента <code>select</code>
<code><option></code>	для определения пункта контейнера <code>select</code> , элемента <code>optgroup</code> , или элемента <code>datalist</code>
<code><output></code>	
<code><progress></code>	полоса выполнения задачи, как в компьютерных играх
<code><select></code>	элемент управления который содержит меню опций
<code><textarea></code>	

`<form>`

[Элемент `<form>`](#)

в стандартной практике для формы принято указывать атрибуты `action` и `method`:

action – определяет адрес, куда должны быть посланы данные после отправки формы

method – указывает, какой HTTP-метод будет использован при передаче данных: "get" или "post"

См. [Отправка данных формы](#)

<input>

Используется для создания интерактивных элементов управления в веб-формах для получения данных от пользователя. Доступен широкий выбор входных данных.

[Атрибуты](#) на MDN.

Примеры инпутов и их свойств: [<input>: The Input \(Form Input\) element](#)

Атрибутов очень много, нет никакого смысла их сюда выписывать. Укажу несколько ходовых
Селектор для правил **placeholder**: `.input::placeholder`

text

Присваивается по умолчанию. Самый обыкновенный текст в одну строчку. Может быть пустой строкой.

Additional attributes

`value` – строка, вбитая в инпут

```
<form>
  <div>
    <label for="uname">Choose a username: </label>
    <input type="text" id="uname" name="name">
  </div>
  <div>
    <button>Submit</button>
  </div>
</form>
```

tel

Для ввода номера телефона. В отличие от email, введённое значение **не проверяется** автоматически по определённому формату.

Additional attributes

```
<label for="phone">Enter your phone number:</label>

<input type="tel" id="phone" name="phone"
  pattern="[0-9]{3}-[0-9]{3}-[0-9]{4}"
  required>

<small>Format: 123-456-7890</small>
```

email

Введённое значение – это value. Оно проверяется (валидация) автоматом. CSS правила для них: [:valid](#) и [:invalid](#).

Валидацию пройдут:

- пустая строка ""
- правильно указанный адрес
- несколько адресов, если есть атрибут [multiple](#)

Дополнительные атрибуты [тут](#).

```
type='email'
```

`value` значение, которое введёт пользователь. Отображается в поле ввода.

`name`

`placeholder` ну понятно

`list`

`maxlength`

`minlength`

`multiple`

`pattern`

`readonly`

[size](#)

password

Безопасный ввод пароля.

```
<input id="userPassword" type="password">
```

[inputmode](#)

Если надо вводить только цифры или только буквы

[required](#)

Если заполнение поля пароля обязательно

[autocomplete](#)

поддержка автозаполнения

button

Обычная кнопка в форме, которая может размещаться в любом месте и на которую можно повесить разный функционал: функционала по умолчанию, как у `<input type="submit">` и `<input type="reset">`, нет.

Чтобы кнопка что-то делала, надо прописать код на JS.

```
<input type="button" value="Click Me">
```

[value](#)

Строка, которая будет «написана» на кнопке

[disabled](#)

Глобальный атрибут для отключения кнопки.

file

Позволяет пользователю выбрать файлы с локального диска и загрузить на сервер через [form submission](#) или управлять через JS с помощью [File API](#).

Additional attributes

```
<input type="file" accept="image/*,.pdf">
```

[accept](#)

Какой тип файла разрешено подгрузить. Можно посмотреть тут: [unique file type specifiers](#)

[capture](#)

Источник захвата. Не понял, что это.

[files](#)

Не понял: [FileList](#)

[multiple](#)

Булевый тип. Позволяет загружать сразу несколько файлов.

checkbox

В отличие от радиоаппонков, чекбоксы не группируются и позволяют выбирать несколько вариантов. Т.к. они маленькие по размеру, их правильно связывать с <label>.

```
type='checkbox'  
value - значение, которое будет отдано этим элементом  
checked - указать, должен ли флажок быть выставлен  
name
```

После отправки формы, будет создана пара name = value. Например:

```
<input type="checkbox" id="subscribeNews" name="subscribe" value="newsletter">
```

Группировка чекбоксов:

У группы чекбоксов стоит одинаковый атрибут name. После отправки на сервер, будет создана строка «interest=coding&interest=music». Не понимаю, в какую переменную она записывается и как выглядит. В документации написано, что эту строку надо распарсить.

```
<fieldset>  
  <legend>Choose your interests</legend>  
  <div>  
    <input type="checkbox" id="coding" name="interest" value="coding" checked>  
    <label for="coding">Coding</label>  
  </div>  
  <div>  
    <input type="checkbox" id="music" name="interest" value="music">  
    <label for="music">Music</label>  
  </div>  
</fieldset>
```

radio

Кнопка-переключатель, позволяет выбрать только одно значение из множества с одинаковым значением name.

```
<input type="radio">  
  
value - значение, которое будет отдано этим элементом  
name - для группы кнопок должно быть одинаковое  
checked - поставить выбор по умолчанию
```

Пример:

```
<form>  
  <p>Please select your preferred contact method:</p>  
  <div>  
  
    <input type="radio" id="contactChoice1"  
      name="contact" value="email">  
    <label for="contactChoice1">Email</label>  
  
    <input type="radio" id="contactChoice2"  
      name="contact" value="phone">  
    <label for="contactChoice2">Phone</label>  
  
    <input type="radio" id="contactChoice3"  
      name="contact" value="mail">  
    <label for="contactChoice3">Mail</label>  
  
  </div>
```

```
<div>
  <button type="submit">Submit</button>
</div>
</form>
```

<label>

Подпись к элементу пользовательского интерфейса.

При клике на лейбл, фокус перейдёт на связанный с ним элемент (всегда?).

По умолчанию это inline элемент, поэтому в css можно указать: display: block;

<label> можно связать с элементом управления, поместив элемент управления внутри элемента <label> или используя атрибут [for](#).

DOM: [HTMLLabelElement](#)

размещение элемента внутри

```
<label>
  Click me
  <input type="text">
</label>
```

через id

```
<label for="username">Click me</label>
<input type="text" id="username">
```

for

id [labelable](#)-элемента, который находится в том же документе, что и элемент label.

form

форма, с которым связан label. Это позволяет размещать элементы label в любом месте документа, а не только как потомки их элементов формы.

<button>

Кнопка, которая может быть использована в формах или в любом другом месте документа, который требует простой, стандартной кнопки.

DOM: [HTMLButtonElement](#)

Документация сообщает, что btn легче стилизовать, чем кнопки в input. Поскольку тег парный, можно прямо в кнопке использовать , , , использовать псевдо-элементы :after и :before. В то время как <input> принимает только текст как атрибут value.

Некоторые атрибуты:

```
<button name="button">Тык!</button>
```

type

submit – отправит данные на сервер

reset – очистит форму

button – нет предзаданного поведения

menu – откроет всплывающее меню через назначенный элемент [<menu>](#)

name

The name of the button, which is submitted with the form data.

value

The initial value of the button – хз что это.

form

Атрибут должен хранить значение `id` элемента `<form>`, с которым связана кнопка. При этом `btn` можно разместить в любом месте документа. Если данный атрибут не установлен, то `<button>` будет связан с родительским `<form>`

autofocus

Понятно.

disabled

Булевый атрибут, указывающий, что пользователь не может взаимодействовать с кнопкой. Если атрибут не установлен, то кнопка наследует его от элемента-контейнера, в котором она расположена.

formaction

Ссылка на обработчик формы. Если атрибут определён – он переопределит атрибут [action](#) у формы-родителя.

formnovalidate

Булевый атрибут. Указывает, что данные формы не будут валидироваться при отправке. Если этот атрибут определён, он переопределяет атрибут [novalidate](#) у формы-родителя.

formtarget

formenctype

formmethod

`<datalist>`

Элемент содержит набор опций ([<option>](#)), доступных для выбора. Выбранное значение будет установлено для элемента `<input>`, с атрибутом [list](#).

DOM-интерфейс [HTMLDataListElement](#)

```
<label for="myBrowser">Choose a browser from this list:</label>
<input list="browsers" id="myBrowser" name="myBrowser" />
<datalist id="browsers">
  <option value="Chrome">
  <option value="Firefox">
  <option value="Internet Explorer">
  <option value="Opera">
  <option value="Safari">
  <option value="Microsoft Edge">
</datalist>
```

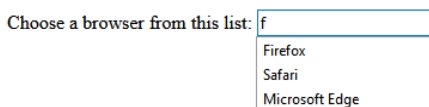
`<fieldset>`

Используется для группировки нескольких элементов управления без веб-форм.

DOM: [HTMLFieldSetElement](#)

Заголовок для `<fieldset>` устанавливается первым [<legend>](#) внутри него. Если задан `<legend>`, он будет помещён поверх верхней границы.

Странная штука. Надо начать что-то вводить, чтобы стали предлагаться совпадающие варианты:



```
<form action="#">
  <fieldset>
    <legend>Simple fieldset</legend>
    <input type="radio" id="radio">
    <label for="radio">Spirit of radio</label>
```

```
</fieldset>
</form>
```

name

Имя, связанное с группой.

form

id элемента <form>, с которой нужно связать <fieldset>, даже если он находится вне формы.

disabled

Если установлен, все элементы управления формой в <fieldset> будут отключены. Это значит, что их нельзя изменять, но можно отправить через форму <form>, в отличие от атрибута disabled на элементах управления формой. Они не будут реагировать на браузерные события, такие как клики мышью или события focus. По умолчанию, браузер отображает такие элементы управления в сером цвете. Обратите внимание, что элементы формы внутри элемента <legend> не будут отключены.

<legend>

Это заголовок содержания родительского элемента <fieldset>, поэтому является его дочерним элементом.

Этот элемент включает только [универсальные атрибуты](#)

DOM: [HTMLLegendElement](#)

```
<form action="#">
  <fieldset>
    <legend>Simple fieldset</legend>
    <input type="radio" id="radio">
    <label for="radio">Spirit of radio</label>
  </fieldset>
</form>
```

<meter>

Прикольная штука, как полоса загрузки в играх. См. документацию.

DOM: [HTMLMeterElement](#)

value

Текущее числовое значение. Он должен быть между минимальным и максимальным значением, если они указаны. Если не указан или имеет неверное значение, то равно 0.

in

Нижняя числовая граница измеряемого диапазона.

max

Верхняя числовая граница измеряемого диапазона.

low

Верхняя числовая граница нижнего предела измеряемого диапазона.

high

Нижняя числовая граница верхнего предела измеряемого диапазона.

optimum

Этот атрибут указывает оптимальное числовое значение. При использовании с атрибутами low и high, он указывает какая часть диапазона является предпочтительной.

form

Этот атрибут связывает элемент с элементом form, частью которого является элемент meter. Например, meter может отображать диапазон, соответствующий элементу input с type number.

<select>

Элемент формы, который содержит меню опций на выбор.

Используется совместно с [<option>](#) и [<optgroup>](#).

DOM: [HTMLSelectElement](#)

```
<select name="select">
  <option value="value1">Значение 1</option>
  <option value="value2" selected>Значение 2</option>
  <option value="value3">Значение 3</option>
</select>
```

autofocus

понятно

disabled

Этот логический атрибут указывает что пользователь не может взаимодействовать с элементом управления.

form

Указывает к какой конкретно форме относится элемент <select>

multiple

логический атрибут указывает что возможен выбор нескольких опций в списке. Если не указан, то только одна опция может быть выбрана.

name

используется для указания имени элемента управления.

required

логический атрибут указывает что обязательно должна быть выбрана опция и которая содержит не пустую строку.

size

Если элемент управления представлен как прокручиваемый список, этот атрибут указывает количество строк в списке, которые должны быть видны за раз.

<optgroup>

Позволяет группировать опции, находящиеся внутри элемента [<select>](#).

Внутри содержит одну или более [<option>](#)

DOM: [HTMLOptGroupElement](#)

label (обязателен)

Имя группы, которое будет отображено браузером в выпадающем списке. Этот атрибут обязателен.

disabled

опции, находящиеся внутри элемента станут недоступными для выбора. Часто браузеры отображают эти опции серым цветом и игнорируют срабатывающие на них события, такие как события мыши или события получения фокуса.

<select>

```
<optgroup label="Группа 1">
  <option>Опция 1.1</option>
</optgroup>
```

```
<optgroup label="Группа 2">
  <option>Опция 2.1</option>
  <option>Опция 2.2</option>
</optgroup>
```

```
<optgroup label="Группа 3" disabled>
  <option>Опция 3.1</option>
  <option>Опция 3.2</option>
```

```
<option>Опция 3.3</option>
</optgroup>
</select>
```

<option>

для определения пункта контейнера select, элемента optgroup, или элемента datalist.

Используется совместно с [<select>](#), [<optgroup>](#) и [<datalist>](#).

DOM: [HTMLOptionElement](#)

value

значение, отправляемое формой, если выбрана данная опция. Если атрибут отсутствует, значение берётся из текстового содержания элемента<option>.

disabled

Если этот Boolean атрибут установлен, опция недоступна для выделения.

label

Этот атрибут - текст ярлыка, отображающий значение(смысл, описание) опции.

selected

опция изначально будет выделена. Если элемент <option> принадлежит элементу <select>, чей атрибут [multiple](#) не установлен, только одна-единственная <option> элемента [<select>](#) может иметь атрибут selected.

<progress>

отображает индикатор, показывающий ход выполнения задачи, обычно отображаемый в виде прогресс бара (индикатора выполнения). Как загрузка в играх.

DOM: [HTMLProgressElement](#)

```
<progress value="70" max="100">70 %</progress>
```

max

должен быть положительным, также, возможно применение числа с плавающей точкой. Значение по умолчанию 1.

value

Этот атрибут указывает какая часть задачи была выполнена. Это может быть число с плавающей точкой от 0 до max, или между 0 и 1, если max не указан.

<textarea>

DOM: [HTMLTextAreaElement](#)

autocomplete

autofocus

cols

rows

disabled

form

maxlength


```
minlength
name
placeholder
readonly
required
spellcheck
wrap
```

Формы

Руководство на сайте мозилы [здесь](#).

Все элементы формы [здесь](#).

Как правило, формы или некоторые элементы форм не наследуют шрифты и их параметры, поэтому всё надо указать снова.

См. также:

```
resize
min-height
max-height
min-width
max-width
transition
```

form

Создание формы. Используется для отправки значений полей формы (input, textarea, select, button).

Документация [здесь](#).

target=""

Указывает как открыть страницу обработчик при отправке формы, указанную в атрибуте action=""

_blank - открывает страницу обработчик в новой вкладке

_self - открывает страницу обработчик в той же вкладке (по умолчанию)

_parent - открывает страницу обработчик в родительской вкладке. Если нет родителя, параметр будет вести себя как _self

_top - открывает страницу обработчик в окне высшего уровня. Если родителя нет, опция ведёт себя как _self

enctype=""

Определяет MIME тип передаваемой информации. Работает только если method="post"

application/x-www-form-urlencoded - значение по умолчанию. Подходит для форм без загружаемых файлов.

text/plain - Отправка данных в виде обычного текста

multipart/form-data - используется для отправки форм с загружаемыми файлами <input type="file">

input

Создание полей формы для заполнения.

- два обязательных атрибута: type и name.

- если надо увеличить размер input-окошка, то можно прописать ему padding.

- селектор для правил placeholder: .input::placeholder

```
<form>
<input type="text" name="username">
</form>
```

value=""

Значение внутри поля по умолчанию. В отличие от placeholder, его надо будет удалить руками. Если я правильно понял, это то значение, которое отправится на сервер.

placeholder=""

Текст внутри поля, который отображается, когда поле пустое. Обычно текст подсказывает, что должно быть внесено в него. Этот текст пропадает, когда начинаешь что-то писать.

Прописать CSS правила для placeholder можно с помощью селектора `.input::placeholder`

radio

На выбор предлагается несколько кнопок, но выбрать нужно только одну. Атрибут `name=""` определяет группу, к которой относится группа кнопок radio. Так комп сможет понять, выбрал ты какой-то один вариант в группе кнопок или нет.

В примере группа кнопок прописана в `name` и называется «choose_male». Лейблы приклеены для удобства выбора, чтобы не тыкать в маленький кружок:

```
<div class="form-group">
  <label for="male">Муж</label>
  <input type="radio" name="choose_male" id="male">

  <label for="female">Жен</label>
  <input type="radio" name="choose_male" id="female">
</div>
```

Radio и checkbox:

Если для элемента указать `checked`, то этот вариант будет выбран по умолчанию.

file

создает кнопку загрузки файла

```
<form>
<div class="form-group">
  <label for="photo">Фото</label>
  <input type="file" name="upload_photo" id="photo">
</div>
</form>
```

textarea

Создает текстовое поле для ввода большого количества информации. Используется при работе с формами. Не наследует шрифты и их параметры, поэтому всё надо указать снова.

```
<form>
<input type="text" name="username">
</form>
```

Прописать CSS правила для placeholder можно с помощью селектора `.input::placeholder`

cols=""

rows=""

Определяет ширину и высоту текстового поля. Указывается целое число, значение по умолчанию 20.

Рекомендуется использовать стилизацию через CSS для обозначения размеров поля.

select

Выпадающий список для выбора опций. Используется с тегом option при создании форм.

С select не работает padding, поэтому чтобы подогнать выпадающий список под другие элементы формы, надо увеличивать параметры width.

```
<form action="/" method="post">
  <select name="country">
    <option value="russia">Россия</option>
    <option value="latvia">Латвия</option>
  </select>
</form>
```

value=""

Это то значение, которое принадлежит выбранному варианту и отправится на сервер.

(у <option>)

name=""

Задаёт уникальное имя выпадающего списка для последующей обработки на сервере.

required

Указывает что поле обязательно для заполнения. Браузерная проверка.

disabled

Поле недоступно для выбора значений.

autofocus

Устанавливает фокус в данное поле по умолчанию при загрузке страницы.

multiple

Дает возможность выбрать несколько вариантов. Чтобы выбрать несколько вариантов зажмите клавишу Ctrl и щелкайте по вариантам

size=""

Задаёт сколько вариантов для выбора будет видно в окне до появления скrolла. Доступен, если используется атрибут multiple

tabindex=""

Указывает последовательность перехода по полям формы при нажатии на кнопку tab. Указывается целое положительное либо отрицательное число.

option

Элемент выпадающего списка select.

```
<form action="/" method="post">
  <select name="country">
    <option value="russia">Россия</option>
    <option value="latvia">Латвия</option>
  </select>
</form>
```

value=""

Это то значение, которое принадлежит выбранному варианту и отправится на сервер.

button

Создание кнопки. Используется для работы с формами либо для создания кнопок для вызова javascript событий. Имеет стилизацию по умолчанию.

```
<button type="submit">Нажми сюда</button>
```

type=""

button обычная кнопка, при нажатии на которую ничего не произойдет

reset служит для сброса значений всех полей формы

submit служит для отправки формы. Форма отправляется по адресу, который указан в <form action="/">

button ничего не будет происходить

name=""

Уникальное имя кнопки для обработки ее значения на стороне сервера

value=""

Значение, которое вы хотите передать при отправке формы

disabled

Делает кнопку неактивной. При нажатии на кнопку ничего не произойдет. Не принимает никаких значений. Нужно для программирования форм. Например, статус disabled сохраняется до тех пор, пока не заполнены обязательные поля формы.

Можно использовать с псевдоклассом : disabled.

fieldset

Создание группы полей формы. Будет нарисована форма в форме. Используется вместе с тегом legend.

```
<fieldset>
  <legend>Личные данные</legend>
  <input type="text" name="name" placeholder="Имя"></input>
  <input type="text" name="sname" placeholder="Фамилия"></input>
</fieldset>
```

name=""

Имя группы

disabled

Блокирует все элементы формы, которые находятся в данной группе.

legend

Заголовок для группы формы (fieldset)

Вместо legend можно использовать какой-нибудь заголовок типа h3.

Таблицы

```
.html
<table class="table">

  <thead>
    <tr>
      <th>#</th>
      <th>Имя</th>
    </tr>
  </thead>
```

```

<tbody>
  <tr>
    <td>1</td>
    <td>Балакас</td>
  </tr>
</tbody>

<tfoot>
  <tr>
    <td colspan="2">Bcero: 2</td>
  </tr>
</tfoot>
</table>

```

```

.CSS
.table {
  border: 1px solid #333;
  width: 50%;
  border-collapse: collapse;
  table-layout: fixed;
}

.table th {
  padding: 5px;
  text-align: left;
  color: gray;
}

.table td {
  border: 1px solid #333;
  padding: 5px;
}

.table tfoot td {
  text-align: right;
}

```

HTML таблиц

<table>

<thead>

<th>

<tbody>

<tfoot>

<td>

<table>

Создаёт контейнер-таблицу. Дальнейшие теги должны быть внутри table.

<tr>

table row.

Создать строку. Внутри строк создаются ячейки td.

<td>

Создать ячейку. Тег должен находиться внутри строк tr или tfoot.

```
<td colspan="2"> </td>    слить несколько ячеек по ширине
```

<thead>

Как правило, самая верхняя строка – это названия столбцов ниже. Это называется шапкой таблицы. Внутри этого тега-строки должны быть ячейки th.

<th>

Ячейка в шапке. Стиль по умолчанию – жирный шрифт и выравнивание по центру.

<tbody>

Строки с данными. Внутри помещаются tr и td.

<tfoot>

Подвал. В нём могут быть какие-то подсчёты из таблицы и т.д.

Ячейка в подвале стандартная – td.

CSS для таблиц

border	границы для таблиц.
border-collapse	схлопывание границ
padding	отступ для текста внутри ячеек.
width	ширина таблицы или ячейки
text-align	Горизонтальное выравнивание текста внутри таблицы
vertical-align	Вертикальное выравнивание inline / table-cell элементов в блоке.
color	задать цвет текста в ячейках
table-layout	Определяет ширину ячеек. Применяется к тегу <table>.
display: flex	можно прописывать для ul

border

границы для таблиц.

Таблица и ячейки внутри неё по-умолчанию отображаются в браузере без видимых границ.

Универсальное свойство border позволяет одновременно установить толщину, стиль и цвет границы вокруг элемента.

Значения могут идти в любом порядке, разделяясь пробелом, браузер сам определит, какое из них соответствует нужному свойству. Для установки границы только на определённых сторонах элемента, воспользуйтесь свойствами [border-top](#), [border-bottom](#), [border-left](#), [border-right](#).

```
border: border-width border-style border-color;
border: 1px solid #333;

border-style: solid    линия
border-style: dashed   чёрточки
border-style: dotted   точки

border-collapse: collapse  схлопывание границ
```

border-collapse

схлопывание границ, чтобы они не выглядели объёмными.

padding

отступ от границ внутри ячеек.

```
.table td {
```

```
padding: 5px;           отступ со всех сторон
padding: 5px 10px;      отступ вверх-низ, право-лево
padding: 5px 10px 5px 10px;
```

```
padding: 5%;
```

```
padding-right
padding-left
padding-top
padding-bottom
```

Поле называется расстояние от внутреннего края рамки элемента до прямоугольника, ограничивающего его содержимое.

Разрешается использовать несколько значений, разделяя их пробелом:

1: поля установлены одновременно с каждого края элемента.

2: первое — от верхнего и нижнего краёв, второе — от левого и правого.

3: первое — от верхнего края, второе — одновременно от левого и правого краёв, третье — от нижнего.

4: поочерёдно от верхнего, правого, нижнего и левого краёв (по часовой стрелке).

width

ширина таблицы или ячейки

По умолчанию ширина и высота таблицы определяется содержимым её ячеек.

```
width: 100%   растянуть на всю ширину блока-контейнера
width: 700px  ну соответственно
```

text-align

Горизонтальное выравнивание текста внутри таблицы

```
text-align: right;
text-align: left;
text-align: center;
```

vertical-align

Вертикальное выравнивание inline / table-cell элементов в блоке.

Обычно прописывается для td

```
vertical-align: center;
vertical-align: top;
vertical-align: bottom;
vertical-align: middle;
```

color

задать цвет текста в ячейках

table-layout

Устанавливает алгоритм расчета размеров ячеек таблицы. Применяется к тегу <table>.

```
.table {
table-layout: fixed;
table-layout: auto;
}
```

table-layout: auto;

Размер ячеек таблицы задается автоматически в зависимости от содержимого ячеек

table-layout: fixed;

Задаёт одинаковый размер всех ячеек таблицы

Адаптивная вёрстка

Адаптивные (резиновые) блоки

Обычно блокам прописывается параметр width. Если ширина окна (например, в телефоне) меньше этого значения, то появляется полоса прокрутки, чтобы этот блок проматывать вправо и влево.

В адаптивной вёрстке такие блоки надо делать «резиновыми», чтобы они подстраивались под разные окна, поэтому блокам прописываются параметры ширины так:

```
.container {  
    max-width: 800px;  
}  
  
.container {  
    width: 100%  
    max-width: 800px;  
}
```

В таком случае блок будет не шире допустимого значения на широких экранах, а на маленьких будет автоматом сжиматься под размер окна.

Аналогично прописываются изображения.

CSS flex-wrap

(это копия из flex)

Определяет возможность переносить элементы контейнера flexbox на новую строку.

Если они не помещаются по ширине, то они не будут утробовываться, а будут перенесены на новую строку.

```
flex-wrap: wrap;           разрешает перенос  
flex-wrap: wrap-reverse;   разрешает перенос, если они не помещаются
```

flex-wrap: nowrap;

Не разрешает перенос элементов на новую строку. Элементы будут сжаты, если не помещаются в строку и если их сжатие разрешено

flex-wrap: wrap;

Разрешает перенос элементов на новую строку, если они не помещаются в текущей строке. Новая строка будет добавлена после предыдущей строки

flex-wrap: wrap-reverse;

Разрешает перенос элементов на новую строку, если они не помещаются в текущей строке. Новая строка будет добавлена перед предыдущей строкой

CSS flex-basis

Задаёт начальный размер элемента в контейнере flexbox. Если используется «flex-direction: row», то flex-basis отвечает за ширину элемента, если используется «flex-direction: column» то flex-basis отвечает за высоту элемента.

Данное свойство имеет приоритет, если используется вместе с width и height свойствами.

```
flex-basis: 20%;
```


CSS flex-grow

Коэффициент роста элемента контейнера flexbox. Задаёт сколько оставшегося пространства в контейнере flexbox должно быть назначено этому элементу.

flex-grow: 0;	не будет расти
flex-grow: 1;	будет расти
flex-grow: 2;	

flex-grow: 0;

Элемент не будет расти, если есть свободное место. Он будет использовать только то пространство, которое ему необходимо

flex-grow: 1;

Элемент заполнит оставшееся пространство, если ни один другой элемент flexbox не имеет значения flex-grow. Если для всех элементов контейнера flexbox задано flex-grow: 1; то все элементы распределят равномерно оставшееся пространство контейнера

flex-grow: 2;

Значение flex-grow является относительным, его поведение зависит от значения родственных элементов контейнера flexbox.

Если в контейнере находятся два элемента, у одного flex-grow: 1; а у другого flex-grow: 2; то оставшееся в контейнере место распределится пропорционально этим значениям. Элемент с flex-grow: 2; получит в два раза больше оставшегося пространства.

Змейка

canvas

Создание области для рисования динамической графики как 2D так и 3D используя javascript.

Элемент предоставляет пустую графическую зону, на которой специальные JavaScript APIs могут рисовать (такие как Canvas 2D или WebGL).

[MDN](#), [HTMLCanvasElement](#) и [Руководство по Canvas](#)

```
<canvas width="200" height="100" id="draw" style="background-color: #c5def1;">
  Ваш браузер не поддерживает canvas.
</canvas>
```

width=""

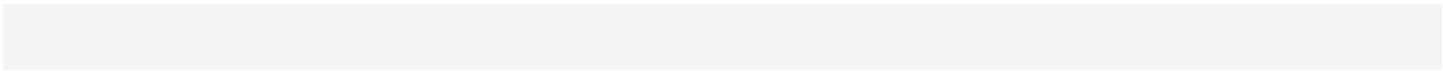
Ширина холста, у большинства браузеров по умолчанию 300px

height=""

Высота холста, у большинства браузеров по умолчанию 150px

Чистые таблицы





.html

.css