Для работы рекомендуется использовать приложение *VSCode* и *node.js*. В *VSCode* рекомендуется поставить следующие расширения:

* *Code Runner* для быстрого запуска
* *LiveServer* для автоперезагрузки страницы
* *Auto Rename Tag* для автозамены закрывающегося тега при редактирования открывающегося
* *W3C Web Validator* для проверки кодстайла
* *Prettier* для автоформатирования кода
* *EditorConfig* для замены табуляции на два пробела
* *PostCSS Sorting* для задания правила сортировки свойств (запуск через *F1 -> PostCSS Sorting: Run*)

Также рекомендуется включить автосохранение документа, настроить файл *User Settings (JSON)* или скопировать настройки из приложенного файла ***settings.json***.

***NodeJS*** – это среда, которая позволяет запускать программы на JavaScrip не в вашем браузере, а на вашем компьютере. Скачать можно на <https://nodejs.org/>

***NPM*** – это менеджер пакетов для *NodeJS* и устанавливается в комплекте с ним.

Настроим автоматизацию проверок внешнего вида кода с помощью пакетов *NodeJS*:

1. Перейдем в папку проекта и создадим новый проект командой

*npm init*

1. Добавим в исключения *.gitignore* папку ***node\_modules/***
2. Установим пакеты ***Prettier*** и ***Stylelint***.

*npm i prettier*

*npm i stylelint*

1. Скопируем конфигурационный файл ***.prettierrc*** вместе с содержимым.
2. Скопируем конфигурационный файл ***stylelint.config.js*** вместе с содержимым.
3. Скачиваем плагин ***Stylelint*** для *VSCode*.
4. Добавляем в файл ***package.json*** блок со скриптами

*"scripts": {*

*"test": "echo 'Привет!'",*

*"check": "prettier --check './\*\*/\*.{html,css,js}'",*

*"format": "prettier --write './\*\*/\*.{html,css,js}'",*

*"lint": "stylelint '\*\*/\*.css'",*

*"lintfix": "stylelint '\*\*/\*.css' --fix",*

*"checkAll": "npm run check && npm run lint",*

*"fixAll": "npm run format && npm run lintfix"*

*},*

Теперь можно запускать в терминале команды:

*npm run checkAll // проверить весь код обоими плагинами*

*npm run fixAll // исправить весь код обоими плагинами*

Для сравнения макета и свёрстанного проекта используются несколько приемов:

1. Добавить макет в разметку, сделать полупрозрачным и поменять цвета.

*.pixel-perfect-image {*

*position: fixed;*

*top: 0;*

*left: 50%;*

*transform: translateX(-50%);*

*opacity: 0.5;*

*filter: sepia(0.6);*

*}*

1. Использовать плагин *PerfectPixel*.

***Hyper Text Marker Language*** – язык *HTML*, используется для создания каркаса сайта.

*HTML*-элемент состоит из открывающего и закрывающего тегов, а также размещенного между ними содержания.

**Атрибуты** (или **свойства**) сообщают дополнительные сведения об элементе. Атрибут встраивается только в открывающий тег. Атрибуты пишутся подряд через пробел, порядок не имеет значение. Значение атрибута приводят в ‘одинарных’ или “двойных” кавычках (без разницы). Для *HTML* рекомендуется использовать двойные кавычки, а для *CSS* – одинарные.

Все возможные атрибуты элементов можно посмотреть на <https://doka.guide/>

Для комментирования строк используются специальные символы:

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

*/\* в CSS — такими \*/*

***<!DOCTYPE html>*** - одиночный тег обозначающий, что перед нами *html*-страница.

***<head lang=”ru”> </head>*** - теги метаданных страницы. Необязательный универсальный атрибут ***lang*** подсказывает поисковым системам, скринридерам и расширениям перевода на каком языке написана страница (в теге *html*) или отдельные блоки текста. В качестве значений используется двухбуквенный код языка, который можно посмотреть [тут](https://www.iana.org/assignments/language-subtag-registry/language-subtag-registry).

***<meta charset=”UTF-8” />*** - одиночный тег для подключения кодировки UTF-8.

***<meta name="viewport" content="width=device-width, initial-scale=1" />*** - одиночный тег, который сообщает браузеру две инструкции. Первая ***width=device-width***, что видимая пользователю часть занимает всю ширину экрана. Вторая ***initial-scale=1***, что страница не масштабируется при загрузке.

***<meta name="description" content="Описание страницы" />*** - описание страницы для поисковиков.

Формат ***Open Graph*** (или ***OG***) позволяет сформировать превью страницы в социальных сетях при передаче ссылки на неё. Подробнее можно почитать [тут](https://ogp.me/).

***<meta property="og:title" content="Можно мягче" />***

***<meta property="og:url" content=”https://maybesofter.com/” />***

***<meta property="og:image" content="/images/og.jpg" />***

***<meta property="twitter:card" content="summary\_large\_image" />*** *// у twitter есть свои теги*

**Фавикон** – это иконка вкладки страницы.

***<link rel=”icon” href=”favicon.ico” sizes=”any” />*** - одиночный тег для подключения фавикона в формате ***ICO*** (32х32 пикселей) для старых браузеров.

***<link rel=”icon” href=”icon.svg” type=”image/svg+xml” />*** - одиночный тег для подключения фавикона в формате ***SVG*** для актуальных браузеров.

***<link rel=”apple-touch-icon” href=”apple-touch-icon.png” />*** - одиночный тег для подключения фавикона в формате ***PNG*** (512х512 пикселей) для иконки смартфонов и планшетов, если пользователь захочет добавить его на экран.

***<title></title>*** - теги названия вкладки страницы.

***<link rel=”stylesheet” href=”./fonts/fonts.css” />*** - одиночный тег для подключения шрифтов.

***<link rel=”stylesheet” href=”./styles/global.css” />*** - одиночный тег для подключения глобальных таблиц стилей.

***<link rel=”stylesheet” href=”./ styles/styles.css” />*** - одиночный тег для подключения таблиц стилей.

***<h1></h1>*** - теги заголовка. Всего их 6 размеров, *<h6>* самый маленький.

***<p></p>*** - теги абзаца. В них заключается основная часть текста.

***<a></a>*** - теги ссылки-якоря (anchor). Необязательный атрибут ***href*** хранит адрес ссылки.

*<a href=”https://takiedela.ru/”>Перейти по ссылке</a>*

Необязательный атрибут ***target*** открывает ссылку в этой вкладке или в новой.

*<a target=”\_blank” href=”https://takiedela.ru/”>Перейти по ссылке</a>*  *// в новой вкладке*

***<img>*** - тег изображения. Создается только одним открывающимся тегом с обязательным атрибутом ***src***.

*<img src=”images/mobile.jpg”>*

Необязательный атрибут ***alt*** отображает текст, который пользователь видит, когда картинка не загружается:

*<img src=”images/mobile.jpg” alt=”здесь должна быть картинка, но что-то пошло не так…”>*

Необязательный атрибут ***width*** позволяет задать ширину картинки в пикселях. Это также позволит избежать рывков страницы при медленной загрузке изображений.

*<img src=”images/mobile.jpg” width=230>*

**Векторная графика** – это изображения формата *svg*, которые состоят из линий. Их можно растягивать без потери качества.

**Растровая графика** – это изображения формата *jpeg*, *png*, которые состоят из пикселей. При растягивании качество теряется, становятся видны пиксели.

***<q></q>*** - теги цитирования фразы прямо внутри текста. Заключают текст в кавычки, вид которых зависит от языка документа.

***<blockquote></blockquote>*** - теги для блочного цитирования фразы. Необязательный атрибут ***cite*** позволяет указать автора цитаты.

*<blockquote cite="https://praktikum.yandex.ru">Достаточно начать и не бросить.</blockquote>*

***<body></body>*** - теги тела страницы (видимой части).

***<div></div>*** - универсальные теги элемента-блока (абстрактная пустая коробка), позволяющие объединять другие элементы. Расположен в *body*.

***<header></header>*** - теги элемента-блока шапки страницы, позволяющая объединять другие элементы (меню, логотип, заголовок страницы, вводная информация). Расположен в *body* и занимает всю ширину страницы.

***<section></section>*** - теги элемента-блока страницы по аналогии с *div*, но лучше читается поисковиками. Подходят для разделения контента на большие логические блоки. Расположен в *body*.

***<footer></footer>*** - теги элемента-блока подвала страницы, позволяющая объединять другие элементы (дополнительная навигация, копирайт, ссылки на партнеров, другая доп.информация). Расположен в *body*.

***<nav></nav>*** - теги элемента-блока навигации по сайту. В него принято оборачивать ссылки навигации по сайту или отдельной странице. Расположен в *body*.

***<main></ main >*** - теги элемента-блока основного содержимого страницы. На многостраничных сайтах, этот контейнер содержит уникальный контент одной страницы. Расположен в *body*.

***<aside></ aside >*** - теги элемента-блока дополнительной информации, часто оформляется как боковая колонка. Расположен в *body*.

***<article></ article>*** - теги элемента-блока, не привязанного к окружающему его контенту. Такой блок можно убрать со страницы и вставить в любое другое место сайта.

В *HTML* существуют два вида списков – маркированный и нумерованный.

***<ul></ul>*** - теги маркированного списка, оформленного маркерами в виде точек.

***<ol></ol>*** - теги нумерованного списка.

***<li></li>*** - теги элементов любого списка.

Необязательное *CSS*-свойство ***list-style*** отвечает за управление маркерами и нумерацией списков. Например, список можно оформить так:

*<ul style=” list-style: none”> // данный стиль вообще скрывает маркер*

*<li>*

*<div>Я на своём месте =)</div>*

*</li>*

*</ul>*

***<button></ button>*** - теги простой кнопки.

***<table></table>*** - теги элементов таблицы.

***<caption></caption>*** - теги краткого описания таблицы. Идет сразу после *<table>*.

***<tr></tr>*** - теги задания строки таблицы.

***<th></th>*** - теги задания ячейки первой строки таблицы.

***<td></td>*** - теги задания ячейки остальных строк таблицы.

***<thead></thead>*** - теги элемента-блока шапки таблицы.

***<tbody></tbody>*** - теги элемента-блока тела таблицы. Можно использовать несколько в одной таблице.

***<tfoot></tfoot>*** - теги элемента-блока подвала таблицы.

Необязательные атрибуты ***rowspan*** и ***colspan*** помогают растянуть одну ячейку на несколько по вертикали и горизонтали соответственно. Соседние ячейки при этом смещаются.

Например, таблицу можно оформить так:

*<table>*

*<caption>*

*Три романа с большим количеством букв*

*</caption>*

*<thead>*

*<tr> // первая строка с заголовками*

*<th>Название книги</th>*

*<th>Автор</th>*

*<th>Количество слов</th>*

*</tr>*

*</thead>*

*<tbody>*

*<tr> // вторая и последующие строки с данными таблицы*

*<td>Келидар</td>*

*<td>Махмуд Довлатабади</td>*

*<td>950 000</td>*

*</tr>*

*</tbody>*

*<tfoot>*

*<tr>*

*<td colspan="2">Среднее количество слов</td> // растянуть ячейку на две в ширину*

*<td>1502009.3</td>*

*</tr>*

*</tfoot>*

*</table>*

К таблице также можно применять стили. Например:

*td, th {*

*border: 1px solid #000; // нарисовать рамку вокруг каждой ячейки*

*}*

*table {*

*background-color: #fff; // задать фоновый цвет таблицы*

*border-collapse: collapse; // убрать расстояние между рамками ячеек таблицы*

*}*

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

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

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

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

*@import url(fonts.css)*

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

*h2 { // селектор по тегу h2*

*font-size: 32px;*

*}*

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

*font-size: 32px;*

*}*

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

*font-size: 32px;*

*}*

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

*font-size: 32px;*

*}*

*div p { // все p внутри div*

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

*}*

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

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

*}*

*.app.best-choice { // элементы с class=”app” и class=”best-choice” одновременно*

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

*}*

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

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

*}*

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

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

*}*

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

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

*}*

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

*font-size: 32px;*

*}*

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

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

*}*

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

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

*margin: 0;*

*font-size: 100%;*

*font-weight: normal;*

*}*

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

*\*,*

*\*::before,*

*\*::after {*

*box-sizing: border-box;*

*}*

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

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

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

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

*// в стилях селектор класса начинается с точки*

*.white-text {*

*color: #fff;*

*}*

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

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

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

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

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

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

*div {*

*color: tomato !important;*

*}*

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

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

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

*rgb(255, 255, 0) // желтый*

*#FFFF00 // желтый*

*#FF0 // желтый*

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

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

**Якорные ссылки** – элемент навигации на сайте, позволяющие переместиться к заданному блоку с атрибутом идентификатором ***id***. Значение атрибута не может начинаться с цифр или спец.символов, и не должно содержать пробелов. Для идентификаторов можно задавать *CSS* селекторы стилей, но не рекомендуется из-за сложности поддержки.

Если якорная ссылка и элемент, на который она ссылается, находятся на одной странице, то в качестве адреса ссылки нужно прописывать ***#имя\_идентификатора***. Чтобы сослаться на элемент с любой другой страницы, нужно указать ***полный\_адрес#имя\_идентификатора***.

*<section class="section about" id="about">*

*<a href="#about">Что это</a>*

*<a href="https://atychiphobia.com/article#about">Якорная ссылка с другой страницы</a>*

Для плавной прокрутки страницы при переходе по якорной ссылке необходимо в *CSS* стилях создать селектор *html* с атрибутом ***scroll-behavior*** и значением *smooth*:

*html {*

*scroll-behavior: smooth;*

*}*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

* ***font-size*** позволяет задать размер шрифта.Например:

*font-size: 24px;*

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

*font-weight: normal;*

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

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

* ***font-style*** позволяет задать стиль шрифта. Например:

*font-style: italic;*

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

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

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

***cursive*** – наклонный системный шрифт

***fantasy*** – декоративный системный шрифт

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

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

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

*@font-face {*

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

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

*// подключение альтернативных форматов шрифтов для разных браузеров*

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

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

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

*font-display: swap;*

*}*

*// различные начертания шрифта подключаются отдельно*

*@font-face {*

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

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

*font-style: italic;*

*font-weight: bold;*

*}*

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

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

*div {*

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

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

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

*font-weight: bold;*

*font-style: italic;*

*}*

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

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

// перерисует видимым скачанным шрифтом весь текст.

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

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

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

Указывать размер элементов можно в процентном соотношении. Это особенно важно при кроссплатформенной верстке – ведь *300px* на мониторе и на смартфоне занимают совершенно разное соотношение экрана. Элементы *h1-h6*, *p*, *div* и *header* занимают 100% ширины своего родителя. Значение элемента *body* равно видимой ширине окна браузера.

Окно просмотра измеряется в единицах ***vw*** (*view width*) и ***vh*** (*view height*) относительно размеров окна браузера, или в %. Как правило, на мобильных устройствах не учитываются размеры управляющих элементов, поэтому появляется скроллинг.

Лайфхак для настройки размеров на мобильных устройствах:

*html {*

*height: 100%;*

*}*

*body {*

*height: 100%;*

*}*

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

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

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

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

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

*overflow: scroll; // включает скролл всегда*

*overflow: hidden; // скрывает лишний контент*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*line-height: 120%; // за 100% берется размер шрифта font-family*

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

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

* ***text-shadow*** позволяет задать тень для текста.
* ***white-space*** задает браузеру, как обрабатывать пробелы в тексте. Например:

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

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

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

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

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

*white-space: pre-line;*

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

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

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

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

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

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

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

Рассмотрим пример, когда при наведении на элемент одного класса будет меняться прозрачность элементов другого класса. Этого можно достичь с помощью свойства ***hover***.

*.error {*

*font-size: 120px;*

*}*

*.message {*

*opacity: 0; // сделать прозрачным*

*}*

*.error:hover ~ .message {*

*opacity: 1; // сделать видимым*

*}*

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

Элементы *div*, *section*, *header*, *h1-h6*, *p*, располагаясь в потоке, занимают всю ширину своего родителя, поэтому называются **блочными**.



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

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

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

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

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

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

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

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

*padding: 10px 20px 30px;*

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

*padding: 10px 20px;*

*// 10px со всех сторон*

*padding: 10px;*

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

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



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

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

*border-color: #000;*

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

*border-width: 1px;*

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

*border-style: solid;*



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

*border: 3px solid #000;*

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

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

Свойство ***box-shadow*** отвечает за создание тени элемента.

*div {*

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

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

*// радиус размытия — 5px*

*// цвет красноватый, #FD6969*

*}*

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

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

*div {*

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

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

*}*

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

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

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

*display: block; // сделает элемент блочным*

*display: inline; // сделает элемент строчным*

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

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

Для того чтобы убрать зазор между блочно-строчными элементами, для коробки нужно задать:

*font-size: 0*;

Строчно-блочные элементы выравниваются по последней строке текста внутри них. Вертикальным выравниванием можно управлять с помощью свойства ***vertical-align***.

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



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



**Флексбокс** — это *CSS*-механизм, созданный специально для быстрого и удобного распределения блоков, их выравнивания и распределения свободного места между ними. Свойство ***display*** со значением ***flex*** превращает элемент во флекс-контейнер, а все его вложенные элементы – во флекс-элементы.

В то время как *display: block/inline/inline-block* определяет поведение конкретного компонента, *display: flex* задает **контекст** для всех вложенных флекс-элементов – особые правила поведения внутри себя (это удобнее). Он делает блоки “резиновыми” – элементы могут сжиматься и растягиваться по заданным правилам, занимая нужное пространство, или автоматически выстраиваться в несколько строк или столбцов, занимая всё предоставленное место.

Уровень вложенности неограничен – дочерние элементы сами могут быть флекс-контейнерами. Настройки флекс-контейнеров более высокого уровня не наследуются.



При преобразовании элемента во флексбокс-контейнер, внутри него появляется две оси – основная и поперечная.

По умолчанию все флекс-элементы выстраиваются в одну строку.

Свойство ***flex-direction*** позволяет задать направление основной оси:

*flex-direction: row; // слева направо (по умолчанию)*

*flex-direction: column; // сверху вниз*

*flex-direction: row-reverse; // справа налево*

*flex-direction: column-reverse; // снизу вверх*

Свойство ***justify-content*** позволяет распределить свободное пространство на основной оси между флекс-элементами:

*justify-content: start; // сначала элементы, потом оставшееся пустое пространство*

*justify-content: end; // сначала пустое пространство, потом элементы*

*justify-content: center; // элементы по середине, пустое пространство поровну с двух сторон*

*// первый элемент прижат к началу контейнера, последний – к концу,*

*// а остальные расставлены между ними с одинаковыми промежутками*

*justify-content: space-between;*

*// все пустое пространство делится на промежутки. Элементы получают отступы по обе стороны,*

*// равные этому промежутку. Получается между элементами отступ в два промежутка,*

*// а перед первым и после последнего элемента отступ в один промежуток*

*justify-content: space-around;*

*justify-content: space-evenly; // все отступы равной величины*

По умолчанию все флекс-элементы растягиваются по поперечной оси до размера наибольшего элемента (в том числе изображения).

Свойство ***align-items*** позволяет задать логику выравнивания флекс-элементов по поперечной оси:

*align-items: stretch // элементы растянуты от края до края (по умолчанию)*

*align-items: start // элементы прижаты к началу*

*align-items: end // элементы прижаты к концу*

*align-items: center // элементы по середине*

*align-items: baseline // элементы выравниваются по базовой линии текста*

**Базовая линия текста** – воображаемая линия, проходящая по нижнему краю знаков шрифта первой строки.



Свойство ***flex-wrap*** позволяет переносить элементы флекс-контейнера, которые в него не влезают:

*flex-wrap: nowrap // элементы не переносятся (по умолчанию)*

*flex-wrap: wrap // элементы переносятся на следующую строку сверху вниз*

*flex-wrap: wrap-reverse // элементы переносятся на следующую строку снизу вверх*

Свойство ***row-gap*** позволяет задать расстояние между строками элементов флекс-контейнера. Свойство ***column-gap*** позволяет задать расстояние между столбцами элементов флекс-контейнера. Оба эти значения можно записать кратко через свойство ***gap***. Такие отступы являются фиксированными и не меняются в результате сжатия или растяжения. Его удобно использовать для выравнивания карточек.

*row-gap: 15px; // задать расстояние между строками*

*column-gap: 30px; // задать расстояние между столбцами*

*gap: 25px 15px; // задать расстояние между строками 25px и столбцами 15px*

*gap: 25px; // задать расстояние между строками и столбцами по 25px*

Свойство ***align-content*** позволяет задать логику выравнивания элементов флекс-контейнера по поперечной оси при переносе на новую строку. Для его работы должен быть включен ***flex-wrap*** и свободное пространство.

*// строки растянуты одинаково так, чтобы занять*

*// все доступное пространство родителя (по умолчанию)*

*align-content: stretch;*

*align-content: start; // все строки располагаются у начала родителя*

*align -content: end; // все строки располагаются у конца родителя*

*align -content: center; // все строки располагаются по центру родителя*

*// первая строка прижата к началу контейнера, последняя – к концу,*

*// а остальные расставлены между ними с одинаковыми промежутками*

*align -content: space-between;*

*// все пустое пространство делится на промежутки. Строки получают отступы по обе стороны,*

*// равные этому промежутку. Получается между строками отступ в два промежутка,*

*// а перед первой и после последней строки отступ в один промежуток*

*align -content: space-around;*

*align -content: space-evenly; // все отступы равной величины*

Свойство ***order*** позволяет задать вес элементу флекс-контейнера. Элементы располагаются в порядке увеличения веса. По умолчанию вес равен нулю. Следует использовать его только в случаях, когда отсутствует доступ в *HTML*.

Свойство ***align-self*** позволяет задать конкретному элементу флекс-контейнера логику выравнивания по поперечной оси. Может принимать те же значения, что и ***align-items***.

Свойство ***flex-grow*** флекс-элемента указывает, может ли он вырастать при наличии свободного места, игнорируя размер контента. По умолчанию значение равно нулю (расширяться нельзя).

Если всем флекс-элементам задать значения 1, то свободное пространство равномерно распределится между ними. Но стоит одному флекс-элементу задать значение 2, то он заберет в 2 раза больше свободного пространства, чем флекс-элементы со значением 1.

Например, с помощью этого свойства закрепим ***footer*** внизу страницы, независимо от высоты контента:

*body {*

*min-height: 100%; // задаем размер равный всей видимой части браузера*

*display: flex; // преобразуем во флекс-контейнер*

*flex-direction: column; // задаем направление сверху вниз*

*}*

*.main {*

*flex-grow: 1; // разрешаем занимать все свободное пространство*

*}*

Свойство ***flex-basis*** флекс-элемента позволяет задать размер элемента (в относительных или абсолютных значениях) до того, как на него начнет действовать расширение. Например, в первой строке поместилось три карточки, а на второй – только две, которые нужно растянуть на всю ширину:

*.content {*

*display: flex; // преобразуем во флекс-контейнер*

*flex-wrap: wrap; // разрешаем перенос*

*}*

*.card {*

*flex-basis: 300px; // задаем минимальную ширину*

*flex-grow: 1; // разрешаем занимать все свободное пространство*

*}*

Свойство ***flex-shrink*** флекс-элемента указывает, может ли он сжиматься при нехватке свободного места, игнорируя размер контента. По умолчанию значение равно нулю (сжиматься нельзя, размер равен значению *flex-basis*). Работает аналогично ***flex-grow*** – чем больше значение, тем быстрее сжимается.

Свойство ***flex*** объединяет все три свойства ***flex-grow***, ***flex-shrink***, ***flex-basis***.

*flex: 0 1 auto;*

**Поток** – это связка элементов и их зависимость друг от друга. Взаимное расположение элементов в потоке называется **позиционированием**. За него отвечает свойство ***position***. По умолчанию для всех элементов используется **статическое позиционирование** и свойства смещения игнорируются:

* Элемент находится в потоке. Соседние элементы знают, где он расположен, и реагируют на его смещение.
* Элемент перемещается визуально только за счет отступов. Так как отступы – часть блочной модели, то с точки зрения браузера мы двигаем не элемент, а меняем его размеры за счет отступов.

*position: static;*

**Относительное позиционирование** позволяет оставить элемент в потоке и смещаться относительно самого себя (технически для остальных элементов место забронировано, но смещено визуально).

*div {*

*position: relative;*

*top: 20px; // сместить элемент вниз*

*right: 40px; // сместить элемент влево (игнор)*

*bottom: 10px; // сместить элемент вверх (игнор)*

*left: 10px; // сместить элемент вправо*

*}*

Если *left* и *right* противоречат друг другу, то *right* игнорируется.

Если *top* и *bottom* противоречат друг другу, то *bottom* игнорируется.

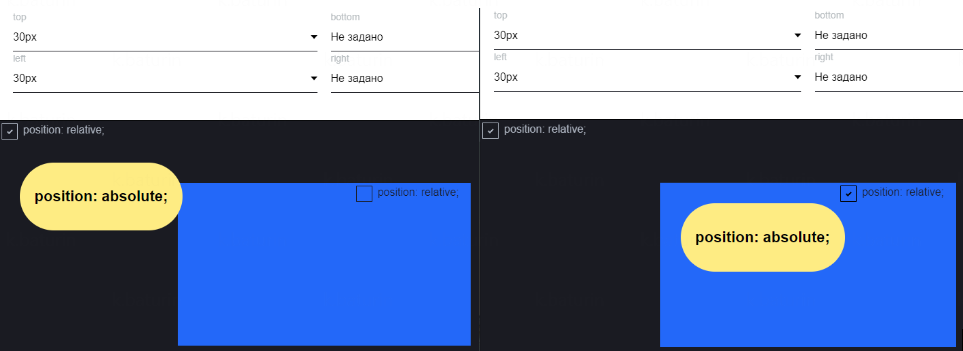
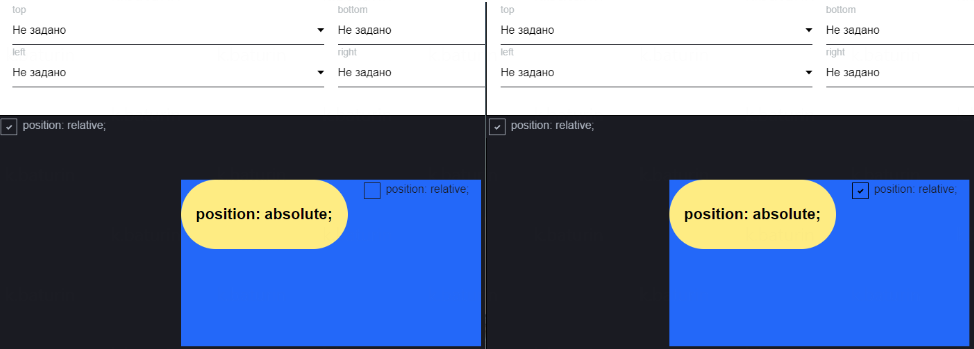
Свойство ***inset*** объединяет в себе все четыре свойства ***top***, ***right***, ***bottom*** и ***left***, по аналогии с ***margin***.

**Абсолютное позиционирование** позволяет убрать элемент из потока и располагает его относительно границ родителя.

Если родительский элемент спозиционирован (значение *position* отлично от *static*), то элемент с абсолютным позиционированием будет размещен согласно свойствам смещения относительно границ родителя. Например, картинка займет всю площадь родителя.

Если у родительского элемента не менялось статическое позиционирование, то элемент с абсолютным позиционированием попробует найти другие родительские элементы с нестатическим позиционированием. Если не найдет, то расположится относительно границ *body*.

*position: absolute;*



Пространство веб-страницы не двумерное, а трехмерное. Ось *Z* направлена перпендикулярно экрану, из глубины монитора по направлению к пользователю. Когда при позиционировании мы вырываем элементы из потока, они могут перекрывать друг друга. Для управления близостью элементов используется свойство ***z-index***. Чем больше целое число, тем ближе оно к пользователю. По умолчанию равно нулю.

Свойство близости не действует на элементы со статическим позиционированием, однако если элемент находится во *flex*- или *grid*-контейнере, то оно сработает.

**Фиксированное позиционирование** позволяет убрать элемент из потока и располагает его относительно окна браузера. Игнорирование противоречащих свойств смещения меняется на приоритетность (например, свойства *left* и *right* позволяют растянуть шапку на весь экран). Подходит для фиксированной шапки сайта или всплывающего окна (***popup***).

*position: fixed;*

**Липкое позиционирование** позволяет задать момент, в который элемент будет вырван из потока и “прилипнет” к окну браузера при прокрутке страницы. Такой элемент “отлипнет” по достижению границ родительского элемента. Обязательно требует свойство смещения.

*position: sticky;*

*top: 20px;*

**Грид** — это *CSS*-механизм, созданный специально для быстрого и удобного распределения элементов на веб-страницах. Свойство ***display*** со значением ***grid*** превращает элемент в грид-контейнер, а все его вложенные элементы – в грид-элементы.

Флексбоксы ориентированы на контент и используются для мест, где нужно выравнивание в строке или колонке. Гриды же ориентированы на структуру и используются для создания общей сетки страницы. Они взаимозаменяемы, но в разных ситуациях удобнее использовать тот или иной инструмент.

**Грид-контейнер** – это набор невидимых вертикальных и горизонтальных **грид-линий**, которые пересекаются и создают области, в которых могут располагаться грид-элементы. Грид-линии имеют нумерацию, начиная с единицы, а также могут принимать отрицательные значения.

По умолчанию грид-контейнер состоит из одного столбца и одной строки, которые занимают весь его размер. При добавлении элементов, для каждого из них создастся новая строка, но элементы все еще будут занимать один столбец.

Свойство ***display*** со значением ***inline-grid*** превращает элемент в грид-контейнер, который будет вести себя как строчный элемент и занимать только то пространство, которое необходимо, чтобы разместить грид-элементы.

Свойство ***grid-template-columns*** позволяет создавать вертикальные грид-линии внутри грид-контейнера. Свойство ***grid-template-rows*** позволяет создавать горизонтальные грид-линии внутри грид-контейнера. Значения могут задаваться в пикселях, процентах или рассчитываться автоматически. Между этими невидимыми грид-линиями будет сформированы **грид-ячейки**.

*.container {*

*display: grid;*

*grid-template-columns: 190px 200px 270px; // создаем три колонки*

*grid-template-rows: repeat(3, 150px); // создаем три столбца через repeat*

*}*

По умолчанию распределение элементов происходит слева направо и сверху вниз. Если элементов больше, чем созданных нами ячеек, то будут сформированы неявные ячейки со значениями *auto*.



Свойство ***row-gap*** позволяет задать расстояние между строками грид-ячеек. Свойство ***column-gap*** позволяет задать расстояние между столбцами грид-ячеек. Оба эти значения можно записать кратко через свойство ***gap***. Такие отступы являются фиксированными и не меняются в результате сжатия или растяжения.

**Фракция**, или ***fr*** – единица доступного пространства, которое можно разделить на сколько угодно частей. Она придумана специально для гридов и её можно комбинировать с фиксированными размерами. Элемент размером 2*fr* будет занимать вдвое больше свободного пространства, чем 1*fr*, а 3*fr* – втрое. Если других фракций нет, забирает все свободное пространство.

*.container {*

*width: 600px;*

*display: grid;*

*grid-template-columns: 100px 2fr 3fr; // получим 3 ячейки с размерами 100px, 200px и 300px*

*}*

По умолчанию каждый грид-элемент занимает одну грид-ячейку. Изменить занимаемый размер грид-элемента можно с помощью свойств ***grid-column-start*** и ***grid-column-end*** для вертикальных грид-линий, ***grid-row-start*** и ***grid-row-end*** для горизонтальных грид-линий. Свойства ***grid-column*** и ***grid-row*** позволяют использовать короткую запись. С помощью этих свойств можно накладывать грид-элементы друг на друга.

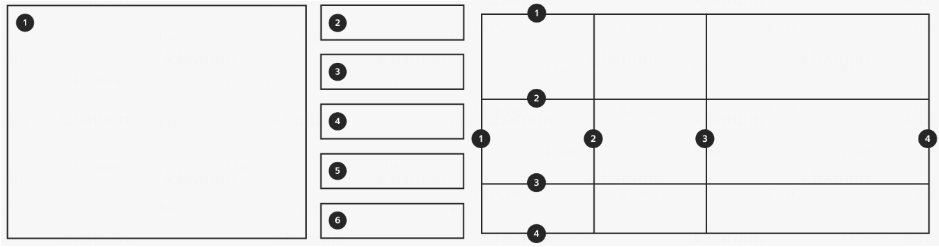
*.block\_size\_big {*

*grid-column-start: 1; // начинается с 1 линии*

*grid-column-end: 3; // заканчивается до 3 линии*

*grid-row: 1/6; // сокращенная запись*

*}*



Грид-линиям можно задавать имена, при этом цифровые названия останутся доступными. Имя указывается в квадратных скобках и может быть любым, кроме ***span***.

*grid-template-rows: [aside-start] 300px [aside-end]; // именуем линии*

*grid-row: aside-start / 4; // задаем размер элемента через имя*

Ключевым словом ***span*** указывают, какое количество строк или столбцов должен занимать элемент до или после какой-то линии.

*.block {*

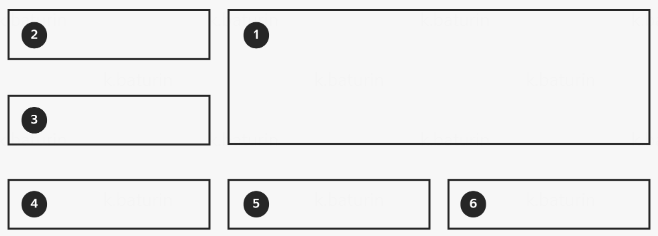
*grid-column-start: 2; // элемент начнется со 2 колонки*

*grid-column-end: span 2; // элемент займет 2 колонки*

*grid-row-start: span 2; // элемент займет 2 строки*

*grid-row-end: 3; // элемент закончится до 3 строки*

*}*



**Грид-область** – это прямоугольная область, которая состоит из одной или нескольких грид-ячеек. Задать имя такой области позволяет свойство ***grid-area***.

*.header {*

*grid-area: header;*

*}*

Свойство ***grid-template-areas*** позволяет разделить грид-контейнер на области. Можно задать только одну область для каждого имени, разъединенные области создать не получится. Каждую ячейку сетки необходимо заполнить, иначе свойство не сработает. Если какую-то область нужно оставить пустой, вместе имени ставится точка.

*.container {*

*display: grid;*

*grid-template-columns: repeat(3, 1fr); // задаем столбцы*

*grid-template-rows: repeat(4, 1fr); // задаем строки*

*grid-template-areas: // распределяем области*

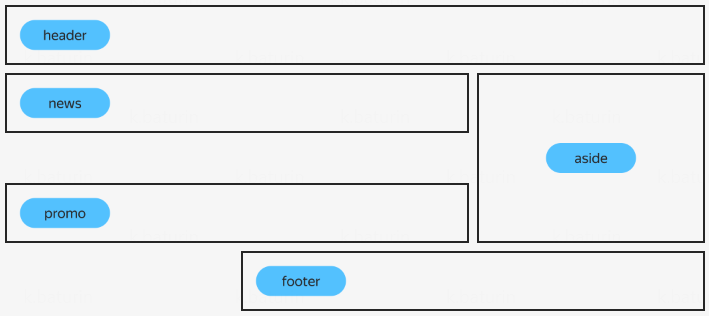
*"header header header"*

*"news news aside"*

*"promo promo aside"*

*". footer footer";*

*}*



Свойства ***align-content*** и ***justify-content*** позволяют выровнять сетку в грид-контейнере по высоте и ширине соответственно. Доступны значения *stretch*, *flex-start*, *flex-end*, *center*, *space-between*, *space-around* и *space-evenly*. Свойство ***place-content*** позволяет кратко записать оба эти свойства.

*.container {*

*align-content: center;*

*justify-content: space-between;*

*place-content: center space-between; // то же самое*

*place-content: center; // если оба значения совпадают*

*}*

Свойства ***align-items*** и ***justify-items*** позволяют выровнять элементы во всех вложенных грид-областях по высоте и ширине соответственно. Доступны значения *stretch*, *start*, *end* и *center*. Свойство ***place-items*** позволяет кратко записать оба эти свойства.

*.container {*

*align-items: center;*

*justify-items: space-between;*

*place-items: center space-between; // то же самое*

*place-items: center; // если оба значения совпадают*

*}*

Свойства ***align-self*** и ***justify-self*** позволяют выровнять элементы в одной конкретной грид-области по высоте и ширине соответственно.

**Фрейм** – это окошко, в котором видно другую страницу сайта. Для его создания нужны теги ***iframe***. Технически фреймы ведут себя в потоке как *div*.

*<iframe src=”https://ya.ru/” frameborder=”0”></iframe>*

Атрибут ***frameborder*** со значением 0 позволяет отключить серую рамку фрейма.

Свойство ***boarder*** позволяет модифицировать или отключить серую рамку фрейма.

***Application Programming Interface*** (или ***API***) – это интерфейс взаимодействия между сайтом и сторонними программами и серверами.

Рассмотрим пример с подключением *API Youtube* через фрейм. Для этого используется ссылка вида *https://www.yotube.com/embed/идентификатор\_ролика*.

*<iframe*

*src="https://www.youtube.com/embed/RhlQvbvMg-0?start=30&mute=true" // без звука и с 30й секунды*

*title="YouTube video player"*

*frameborder="0"*

*allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share"*

*allowfullscreen*

*></iframe*>

Рассмотрим пример с подключением Яндекс.Карт через фрейм. Для этого воспользуемся специальным конструктором *https://yandex.ru/map-constructor/*.

*<iframe*

*src="https://yandex.ru/map-widget/v1/?um=constructor%3A6fc6359a9fea00ada5a4032946d1df7ac9ad308d11f5e45b29cf78e2bdf6a738&amp;source=constructor"*

*width="100%"*

*height="500"*

*frameborder="0"*

*></iframe>*

Рассмотрим пример подключения своего видео без использования сторонних сервисов. Для этого нам нужны теги ***video***. Не рекомендуется позиционировать видео относительно элемента html, т.к. на некоторых размерах экранов будет неправильно высчитывать ширину в 100%, и вырвать видео из потока ***position:relative***.

Атрибут ***controls*** позволяет отобразить кнопки управления видео.

Атрибут ***muted*** позволяет выключить звук видео.

Атрибут ***autoplay*** позволяет включить автовоспроизведение видео. Поддерживается не всеми браузерами и работает в связке с ***muted***.

Атрибут ***loop*** позволяет включить зациклить видео по кругу.

Атрибут ***poster*** позволяет установить картинку, которую будет видеть пользователь, пока не запустит воспроизведение или пока видео загружается.

*<video src="video.mp4" controls muted autoplay loop poster=”poster.png”></video>*

Для подключения нескольких видео используется одиночный тег ***source***. В первую очередь нужно подключать более современные типы файлов, которые указываются в атрибуте ***type***. Браузер возьмет первый файл, который сможет проиграть, и не будет скачивать остальные файлы.

*<video controls muted loop poster=”poster.png”>*

*<source src="video.webm" type="video/webm" />*

*<source src="video.mp4" type="video/mp4" />*

*</video>*

Свойство ***object-fit*** позволяет задать как видео будет подстраиваться под заданные размеры.

*object-fit: fill; // по умолчанию, вписывается без сохранения пропорций*

*object-fit: contain; // вписывается под заданные размеры без нарушения пропорций*

*object-fit: cover; // вписывается без нарушения пропорций, обрезая все лишнее*

*object-fit: none; // отображается без изменений пропорций или размеров*

*object-fit: scale-down; // браузер сам выбирает между none и contain*

Свойство ***object-position*** позволяет выровнять изображение по осям *X* и *Y* после подгонки по размерам.

*object-position: top right; // сначала по X, потом по Y*

Рассмотрим пример подключения своего аудио без использования сторонних сервисов. Для этого нам нужны теги ***audio***. Они имеют такие же атрибуты как у тегов ***video***.

*<audio>*

*<source src="sound.mp3" type="audio/mp3" />*

*<source src="sound.ogg" type="audio/ogg" />*

*</audio>*

Как сделать свой видеоплеер можно почитать [тут](https://habr.com/ru/company/microsoft/blog/127295/).

**Псевдоклассы** – это инструмент, который позволяет элементам реагировать на действия пользователя и давать обратную связь. Например, псевдокласс ***:hover*** выбирает элемент, на который наведен курсор мыши. Селектор псевдокласса ***.article-link:hover*** выберет на странице все элементы с классом ***.article-link*** и применит к ним стили из правила, когда на них будет наведен курсор.

*button {*

*border: 2px solid red;*

*color: white;*

*background-color: black;*

*}*

*button:hover { // перекрашивает кнопку при наведении*

*border: 2px solid black;*

*background-color: red;*

*}*

Псевдокласс ***:link*** стилизует все ссылки, по которым еще не переходили. Его редко используют, потому что селекторы ссылки ***a*** и ***a:link*** абсолютно идентичны, и стили применяются к одним и тем же элементам.

Псевдокласс ***:visited*** стилизует все ссылки, по которым уже переходили. В целях безопасности возможности стилизации посещенных ссылок ограничены следующими свойствами: ***color***, ***background-color***, ***border-color***, ***outline-color***, ***column-rule-color***, ***fill***, ***stroke***. Также браузер будет игнорировать прозрачные (***transparent***) и полупрозрачные цвета (***rgb* с альфа-каналом**).

Псевдокласс ***:active*** стилизует элементы в тот миг, когда на них уже нажали, но еще не отпустили клавишу мыши. Это промежуточное состояние называется **активным**.

Для исключения конфликтов рекомендовано писать псевдоклассы для элемента в следующем порядке:

*link -> visited -> hover -> active*

Псевдокласс ***:focus*** стилизует элементы, на которые навелись любым способом (например, с помощью *Tab* клавиатуры или после клика по ним). Как правило он изменяет обводку (***outline***) элемента.

*button:focus {*

*outline: 15px solid lightpink;*

*}*

Псевдокласс ***:focus-visible*** стилизует элементы, на которые навелись только с клавиатуры. Потребуется сбросить стили для ***:focus***.

*button:focus {*

*outline: none;*

*}*

*button:focus-visible {*

*border-color: rebeccapurple;*

*color: palevioletred;*

*}*

Псевдокласс ***:focus-within*** стилизует элементы, которые либо сами находятся в фокусе (на которые навелись любым способом), либо имеют дочерние элементы в фокусе.

*form:focus-within { // при фокусировки любого элемента формы, форма сменит цвет*

*background-color: #c4c4c4;*

*}*

Псевдоклассы ***:nth-child()*** и ***:nth-last-child()*** стилизуют каждый ***n***-й элемент с начала и конца соответственно. Внутри скобок указывается либо формула, либо номер, либо ключевые слова (например, ***even*** для четных и ***odd*** для нечетных). Подсчет элементов ведется внутри одного блока родителя. Визуализировать формулы можно [тут](https://css-tricks.com/examples/nth-child-tester/).

*li:nth-child(3n) { // перекрасим каждый 3й элемент списка*

*color: #fd24af;*

*}*

*li:nth-child(5) { // подчеркнем 5й элемент списка*

*text-decoration: underline;*

*}*

*.card:nth-child(even) { //* заменить правый отступ на левый у каждого четного элемента

*margin-right: 0;*

*margin-left: 80px;*

*}*

Псевдоклассы ***:first-child*** и ***:last-child*** являются производными ***:nth-child()*** и стилизуют первый и последний элементы соответственно внутри одного блока родителя.

Псевдокласс ***:nth-of-type()*** и его производные работают аналогично ***:nth-child()***, но считают только однотипные элементы.

*button:first-of-type { // перекрасить цвет первой кнопки внутри родителя*

*color: #fd24af;*

*}*

Псевдокласс ***:not()*** используется для исключения селектора из выборки. Внутри скобок могут быть и другие псевдоклассы, перечисленные через запятую.

*.text:not(h2) { // изменить цвет всех элементов, кроме заголовков h2*

*color: #fd24af;*

*}*

*p:not(:last-of-type) { // задать отступы всем абзацам, кроме последнего*

*margin-bottom: 25px;*

*}*

Псевдокласс ***:has()*** позволяет изменить родительский элемент указанного селектора. Сработает для любого уровня вложенности.

*.card:has(.label) { // если у card есть пометка label, то card меняет цвет рамки*

*border-color: #5f0dee;*

*}*

**Псевдоэлементы** – это инструмент, который позволяет выбрать только часть какого-то элемента. Например, псевдоэлемент ***::first-line*** выберет только первую строку текста, какой бы длины она ни была и как бы ни менялась. Селектор псевдоэлемента ***.article-text::first-line*** выберет на странице все элементы с классом ***.article-text*** и применит стили из правила только к их первой строке текста.

Псевдоэлементы ***::before*** и ***::after*** позволяют встраивать блоки перед и после элемента соответственно с помощью свойства ***content***. Не забываем сбрасывать ***box-sizing*** для них через общий селектор ***\****. Скринридеры не увидят эти изменения. Основная задача этих псевдоэлементов – избавиться от пустых или бессмысленных блоков. Не работает с тегами, вместо которых что-то рисуется.

*p::before { // добавим текст перед абзацем*

*content: "Заметка! ";*

*color: #fd24af;*

*}*

*p::after { // добавим текст после абзаца*

*content: "!";*

*color: #fd24af;*

*}*

*.header::before { // добавим затемняющий слой поверх картинки и перед заголовком*

*content: ‘ ’;*

*position: absolute;*

*z-index: -1;*

*inset: 0;*

*background-color: rgb(0 0 0 / 0.5);*

*}*

Свойство ***white-space*** позволяет редактировать текст внутри свойства ***content***. Например, переносить на новую строку.

*.page\_\_title:hover::before { // всплывающая подсказка при наведении на текст*

*content: 'На самом деле нет.\AВажнее &mdash; научиться ошибаться.';*

*white-space: pre;*

*}*

Псевдоэлемент ***::marker*** используется для создания самодельного маркера в списках. На текущий момент поддерживает свойства ***content***, ***white-space***, ***text-transform***, ***letter-spacing***, ***color*** и все свойства семейства ***font***. Раньше эту задачу решали при помощи ***::before***.

*li {*

*display: flex;*

*align-items: center;*

*}*

*li::before { // создать кастомный маркер и поместить перед элементом списка*

*content: “”;*

*display: inline-block;*

*width: 15px;*

*height: 15px;*

*margin-inline-end: 10px;*

*border-radius: 50%;*

*background-color: red;*

*}*

*li:nth-child(2)::before { // сменить цвет маркера*

*background-color: orange;*

*}*

*li::marker { // аналогично через marker*

*color: #23054a;*

*font-weight: bold;*

*font-size: 2rem;*

*}*

Псевдоэлемент ***::selection*** позволяет менять стили текста при выделении. На текущий момент поддерживает свойства ***color***, ***background-color***, ***text-decoration*** и ***text-shadow***.

*p::selection { // покрасить цвет и фон при выделении текста*

*color: #6fc1fd;*

*background-color: black;*

*}*

Псевдоэлемент ***::placeholder*** позволяет менять стили текста подсказок в полях ввода.

*input::placeholder {*

*color: rgb(0 0 0 / 60%);*

*}*

**Доступность** – это возможность использования интерфейса людьми с различными особенностями. Во многих странах доступность – это часть законодательства, касающегося цифровых сервисов. Подробнее можно почитать [тут](https://www.w3.org/Translations/WCAG20-ru/). Основные принципы доступности интерфейсов:

* **Воспринимаемость** – интерфейс должны суметь воспринять любые люди. Например, важно дублировать источники информации: показывать ошибку не только красным цветом, но и текстом.
* **Управляемость** – нужно думать об альтернативных сценариях управления интерфейсом. Например, не только мышкой, но и клавиатурой.
* **Понятность** – не самое простое правило, ведь даже ребёнок видит мир не так, как взрослый. Например, можно делать единообразную навигацию по сайту.
* **Надёжность** – нужно, чтобы интерфейсы работали с разными технологиями, которые упрощают людям использование интернета. Например, вся страница должна корректно считываться скринридерами.

Ключевые особенности доступности интерфейсов:

* Наличие альтернативных способов донесения информации (субтитры в видео, озвучка кнопок, описание изображений).
* Резкость анимации (отсутствие скримеров, быстрых мерцаний блоков для эпилептиков).
* Читаемость (размер шрифта менее 16 пикселей трудночитаем для людей с дислексией).
* Контрастность, или разница в яркости и цвете, которая делает объект различимым. Для его проверки можно использовать плагины для *Figma* [*Contrast*](https://www.figma.com/community/plugin/748533339900865323) или [*A11y Color Contrast Checker*](https://www.figma.com/community/plugin/733159460536249875/A11y---Color-Contrast-Checker).

Включить эмулятор особенностей зрения можно в инструментах разработчика с помощью команд:

*“Ctrl + Shift + P” -> “Rendering” -> “Emulate vision deficiencies”*

Для корректной работы скринридеров браузер строит из *HTML*-кода **дерево доступности** (*accessibility tree*). Каждый элемент описывается через набор параметров:

* Роль элемента. Например, у заголовка роль “*heading*”, у блока *nav* роль “*navigation*” и т.д. Полный список можно найти [тут](https://doka.guide/a11y/).
* Имя элемента, которое не всегда есть. Например, у заголовков или кнопок оно берется из текста внутри, но у секций или абзацев его нет.

Список популярных скринридеров:

* Для *Windows* – [*JAWS*](https://www.freedomscientific.com/products/software/jaws/) (платный), [*NVDA*](https://www.nvaccess.org/about-nvda/) (бесплатный) и [*Narrator*](https://support.microsoft.com/ru-ru/windows/%D0%BF%D0%BE%D0%BB%D0%BD%D0%BE%D0%B5-%D1%80%D1%83%D0%BA%D0%BE%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE-%D0%BF%D0%BE-%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8E-%D1%8D%D0%BA%D1%80%D0%B0%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE-%D0%B4%D0%B8%D0%BA%D1%82%D0%BE%D1%80%D0%B0-e4397a0d-ef4f-b386-d8ae-c172f109bdb1) (бесплатный и предустановленный).
* Для *MacOS* и *iOS* – [*VoiceOver*](https://www.apple.com/ru/accessibility/vision/) (предустановлен).
* Для *Android* – [*TalkBack*](https://support.google.com/accessibility/android/topic/3529932) (предустановлен).
* Для *Linux* – [*Orca*](https://help.gnome.org/users/orca/stable/) (предустановлен).

Появление и пропадание важных для понимания смысла элементов по наведению мыши – это **антипаттерн**. На мобильных устройствах нет мыши.

Для сокрытия элементов из дерева доступности используют следующие способы:

*<button hidden>Кнопка Шрёдингера</button> // атрибут hidden в html*

*.hidden-block { // свойство display в css*

*display: none;*

*}*

*.another-hidden-block { // свойство visibility в css*

*visibility: hidden;*

*}*

Атрибут ***hidden*** и свойство ***display*** не только скрывают элемент, но и вырывают его из потока элементов.

В качестве альтернативы можно скрыть элемент визуально, но оставить его в дереве доступности:

*.visually-hidden {*

*position: absolute; // вырвем элемент из потока*

*width: 1px; // сделаем элемент очень маленьким, но чтобы не пропал совсем*

*height: 1px;*

*margin: -1px; // уберем элемент из поля зрения, сдвинув на размер самого себя*

*padding: 0; // обнулим отступы, которые есть у некоторых элементов по умолчанию*

*border: 0; // уберем границу, которая может быть по умолчанию у элементов*

*clip: rect(0 0 0 0); // обрежем видимую область до нулевого значения*

*overflow: hidden; // на всякий случай сокроем видимое за пределами блока*

*}*

*<button class="header\_\_menu-button">*

*<span class="visually-hidden">Открыть меню</span> // сокроем описание кнопки*

*</button>*

Изображение, которые влияют на смысл страницы, называются **контентными**, а те, что оформляют смысл, называются **декоративными**. Контентные изображения должны быть в *HTML*, чтобы их озвучил скринридер. Декоративные изображения не должны озвучиваться скринридером (например, внешний вид кнопки), поэтому они должны быть в *CSS* или скрыты другими способами. Например:

* первая реализация вставляет контентное фото в качестве фона без озвучки скринридером

*<div class="card\_\_image-wrapper dalai-lama-photo">*

*<div class="card\_\_description">*

*<h2 class="card\_\_title">Далай Лама</h2>*

*<p class="photo-copyright">Фото: агентство Reuters</p>*

*</div>*

*</div>*

* вторая реализация вставляет контентное фото картинкой с озвучкой скринридера

*<div class="card\_\_image-wrapper">*

*<img class="card\_\_image"*

*src="./images/dalai-lama.jpg"*

*alt="Далай Лама в монашеской одежде, показывает двумя пальцами символ мира."*

*/>*

*<div class="card\_\_description">*

*<h2 class="card\_\_title">Далай Лама</h2>*

*<p class="photo-copyright">Фото: агентство Reuters</p>*

*</div>*

*</div>*

***Accessible Rich Internet Applications***, или ***ARIA*** – это доступные многофункциональные интернет-приложения, или дополнительные инструменты, которые помогают сделать интерфейс более доступным. Полный список можно посмотреть [тут](https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Attributes).

Атрибут ***role*** управляет именами элементов.

Атрибут ***aria-hidden*** управляет видимость элементов в дереве доступности.

Атрибут ***aria-label*** позволяет задать имена-метки для элементов.

Атрибут ***aria-description*** позволяет задать описание для элементов.

*<header class="header" aria-label="Шапка сайта">*

Атрибут ***aria-labelledby*** задает имя через ссылку на *id*.

Атрибут ***aria-describedby*** задает описание через ссылку на *id*.

*<article class="card"*

*aria-labelledby="hazaiya-heading"*

*aria-describedby="hazaiya-description"*

*>*

*<blockquote class="quote">*

*<p class="quote-text">*

*«Счастлив тот, кто вошел в сокровищницу внутри себя и увидел звезду»*

*</p>*

*<div class="qoute-wrapper">*

*<p class="quote-author" id="hazaiya-heading">Иосиф Хаззайя</p>*

*<p class="quote-description" id="hazaiya-description">*

*Монах, писатель, богослов, восточно-сирийский мистик Ассирийской Церкви.*

*</p>*

*</div>*

*</blockquote>*

*</article>*

Атрибут ***aria-controls*** указывает на *id* элемента, которым управляет кнопка.

Атрибут ***aria-expanded*** сообщает скринридеру статус элемента (свернут *false* или раскрыт *true*).

Атрибут ***aria-selected*** подсказывает через скринридер, выбрана ли в текущий момент опция табуляции (*true*) или нет (*false*).

*<button class="header\_\_menu-button"*

*aria-controls="main-menu"*

*aria-expanded="false"*

*>*

Свойство уменьшения движения ***prefers-reduced-motion*** позволяет устранить резкий переход (мигание) цвета.

*@media (prefers-reduced-motion) {*

*button {*

*transition: .5s; // меняем плавность перехода*

*}*

*button:hover {*

*background-color: tomato;*

*color: white;*

*}*

*}*

Свойство повышенной контрастности ***prefers-contrast*** позволяет увеличить разницу цветов при их смене.

*@media (prefers-contrast) {*

*button {*

*background-color: black;*

*}*

*button:hover {*

*background-color: white;*

*color: black;*

*border: 2px solid black;*

*}*

*}*

**Форма** – это компонент страницы для получения информации от пользователя. Для её создания используются теги ***form***.

Для группировки нескольких полей формы, связанных по смыслу, используются филдсеты с тегами ***fieldset***. Например, первый блок отведен под сбор контактной информации, а второй – под информацию о доставке. Рекомендуется в каждом филдсете указывать его описания для скринридера с помощью тегов ***legend***.

Для получения информации из полей ввода формы используются элементы с одиночным тегом ***input***.

Атрибут ***type*** позволяют задать логику для ввода данных с помощью элементов ***input*** и ***button***. Он может принимать значения *text*, *number*, *email*, *password*, *tel*, *url*, *color*, *range*, *date*, *datetime-local*, *month*, *week*, *time*, *radio*, *checkbox*, *button*, *submit*, *file*, *reset* и другие.

Атрибуты ***min***, ***max*** и ***step*** позволяют задать минимальное и максимальное значения для числового типа полей ввода данных, а также размер шага. По умолчанию они имеют значения от 0 до 100 с шагом 1.

Атрибуты ***minlength*** и ***maxlength*** позволяют задать минимальное и максимальное количество символов для текстового типа полей ввода данных.

Атрибут ***placeholder*** позволяет задавать подсказки в полях ввода.

Для создания кнопки в форме используются теги ***button***. Они работают аналогично ***<input type=”button” />***, но позволяет разместить контент, текст или изображение.

*<form>*

*<fieldset>*

*<legend>Ваши контактные данные</legend>*

*<label for="email" class="visually-hidden">Почта</label>*

*<input type="email" placeholder="me@example.com" id="email" />*

*<label for="password" class="visually-hidden">Пароль</label>*

*<input type="text" placeholder="Пароль" id="password" />*

*</fieldset>*

*<fieldset>*

*<legend>Время и адрес доставки</legend>*

*</fieldset>*

*<button class="button" type="submit">Войти</button>*

*</form>*

Атрибут ***name*** задает имя поля ввода, которые передаются на сервер.

Атрибут ***value*** хранит данные поля ввода, которые передаются на сервер.

*<form>*

*<label for="email">Почта</label>*

*<input type="email" name="email" id="email" />*

*<label for="password">Пароль</label>*

*<input type="password" name="password" id="password" />*

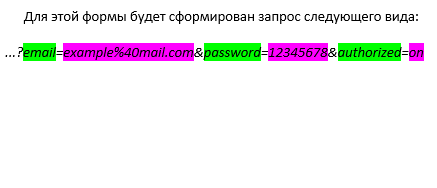
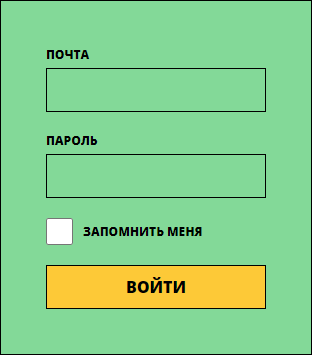
*<label for="authorized" class="agree">*

*<input type="checkbox" name="authorized" id="authorized" /> Запомнить меня*

*</label>*

*<button>Войти</button>*

*</form>*



Атрибут ***action*** для ***form*** отвечает за отправку данных формы на сервер или внешний сервис. Его значением могут быть:

* Пустое значение, заданное по умолчанию.
* Путь до скрипта.
* URL до сервиса, предоставляющего готовый скрипт (форм-процессорами). Например, [таким](http://formspark.io/).

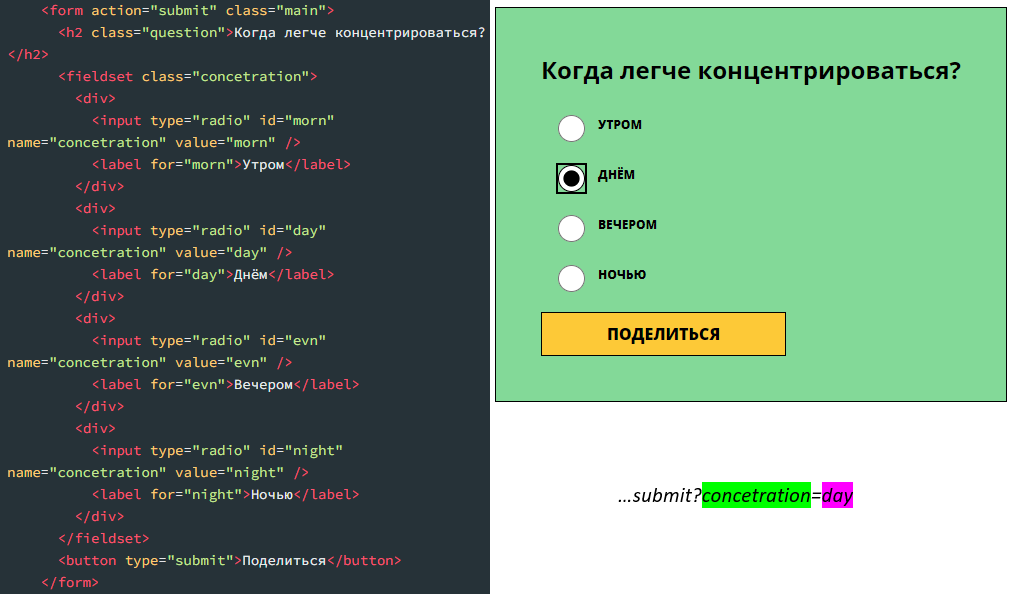
*<form action=""></form>*

Главным отличием значения ***radio*** от ***checkbox*** для атрибута ***type*** является отсутствие возможности множественного выбора. В первом случае, значение переменной ***name*** просто перезаписывается. Во втором же – дублируется при отправке запроса.

Атрибут **checked** внутри тега ***input*** позволяет отмечать дефолтные варианты при загрузке страницы.

Атрибут **required** внутри тега ***input*** позволяет отмечать обязательные к заполнению поля.

Атрибут **disabled** позволяет сделать одно или несколько полей недоступными для выбора.



Для ввода многострочного текста в форме лучше использовать теги ***textarea***. Размер шрифта влияет на размер поля.

Атрибут ***rows*** позволяет задать количество строк, которые помещаются в текстовом поле. По умолчанию равно 2.

Атрибут ***cols*** позволяет задать сколько символов поместится в каждой строке.

Свойство ***resize*** управляет возможностью изменения размера формы ввода текста. По умолчанию можно изменить размер, потянув за правый нижний угол. Свойства ***width*** и ***height*** при применении ***resize*** игнорируют ***rows*** и ***cols***.

*resize: none; // запретить любое изменение*

*resize: vertical; // запретить изменение по вертикали*

*resize: horizontal; // запретить изменение по горизонтали*

Для создания поля с выпадающим списком используются теги ***select***. Варианты же заключены между тегами ***option***. Для объединения опций в группы можно использовать тег ***optgroup***, но не стоит вкладывать один ***optgroup*** в другой.

*<select>*

*<optgroup label="Выберите тотем дня" id="totem">*

*<option value="Тигр из мадагаскара">Тигр из мадагаскара</option>*

*<option value="Бешеная каракатица">Бешеная каракатица</option>*

*</optgroup>*

*</select>*

С помощью тегов ***label*** (ярлык) мы сообщаем пользователю, что именно вводить в каждое поле. Для более тесной связки ярлыка и поля ввода используются атрибуты ***for*** и ***id***. Сами по себе поля ввода не имеют семантики, а ярлыки позволяют скринридерам озвучивать их. Ярлыки также можно скрывать с помощью *visually-hidden*. Например:

*<label class="name"> // первый вариант*

*<span>Имя</span>*

*<input type="text" />*

*</label>*

*<label for="surname">Фамилия</label> // второй вариант*

*<input type="text" id="surname" /> // позволяет расположить ярлык и поле в разных местах*

Поля ввода – это интерактивный элемент. Нельзя портить его фокусное состояние и ухудшать доступность. Не убирайте *outline*, а если меняете его, меняйте на контрастный.

Атрибут **autocomplete** позволяет управлять автозаполнением полей формы, основываясь на ***type*** полей ввода. По умолчанию включено настройками *html*, а также политиками браузеров. Он также позволяет задать значение автозаполнения для конкретного поля с тегами ***input***, ***select*** и ***textarea***.

*<form autocomplete="off">*

*<label for="email" class="visually-hidden">Почта</label>*

*<input*

*type="text"*

*placeholder="me@example.com"*

*id="email"*

*name="email"*

*autocomplete="email"*

*/>*

*</form>*

Атрибут ***inputmode*** отвечает за адаптацию виртуальной клавиатуры под тип данных. Он во многом пересекается с атрибутом ***type***. Например, для ***<input type="number">*** у ***inputmode*** есть значения ***numeric*** (только цифры) и ***decimal*** (цифры и разделитель дробных чисел, адаптированный под язык раскладки). Расширить набор значений можно с атрибутом ***pattern***. Подробнее можно почитать [тут](https://doka.guide/html/pattern/).

По умолчанию форма проверяет соответствие введенного в поле текста и его ожидаемым значением. Обычно вместо браузерной валидации используют проверку на *JavaScript*.

Например, тип *email* ожидает почту с собакой *@*. Выделить корректное и некорректное значение можно с помощью псевдоклассов ***:valid*** и ***:invalid***.

*input:invalid {*

*background-color: #fd24af;*

*}*

*form:valid {*

*outline: 2px solid #5f0dee;*

*}*

Например, для проверки числовых значений в диапазонах, указанных с помощью атрибутов min и max , есть псевдоклассы ***:in-range*** и ***:out-of-range***.

*input[type='number']:out-of-range {*

*background-color: #fd24af;*

*}*

*input[type='number']:in-range {*

*background-color: #80d994;*

*}*

Рассмотрим стили форм ввода, используемые по умолчанию:

* Поля ввода и ярлыки являются строчно-блочными элементами. Изменить это можно с помощью свойства ***display***.
* Текстовые стили и цвет вводимого в поле ввода текста не наследуется от родителя, поэтому всем полям ввода нужно задавать наследование явно с помощью специального значения ***inherit***.
* Полю ввода по умолчанию задан белый фон. Изменить это можно с помощью свойства ***background-color***.
* Полю ввода по умолчанию задана рамка. Изменить это можно с помощью свойства ***border***.
* Поле ввода всегда небольшое по ширине. Изменить это можно задав ***width: 100%*** от родителя.
* У поля ввода по умолчанию отсутствуют отступы и текст может прилипать к краям.
* По умолчанию поле ввода окружается рамкой, когда находится в фокусе. Изменить поведение можно, использовав псевдоклассы ***:focus*** и ***:focus-visible***.

*input {*

*font-family: inherit;*

*font-size: inherit;*

*color: inherit;*

*background-color: transparent;*

*padding: 10px 15px;*

*border: 1px solid #000;*

*}*

*input:focus {*

*outline: none;*

*}*

*input:focus-visible {*

*outline: 1px solid -webkit-focus-ring-color; // стандартный цвет браузера пользователя*

*}*

По умолчанию поля ввода, в которые нельзя ввести текст (***checkbox***, ***radio***, ***range***, ***select***), стилизируются каждым браузером по-своему. Для изменения стилизации чекбоксов и радиокнопки используется следующий порядок действий:

1. Пишем правильную разметку
2. Скрываем стандартный элемент
3. Рисуем на его месте новый элемент
4. Пишем сложносочиненный селектор, чтобы все работало

*<form>*

*<fieldset>*

*<label>*

*<input class=”visually-hidden” type="radio" name="answer" />*

*<span></span>*

*Да*

*</label>*

*<label>*

*<input class=”visually-hidden” type="radio" name="answer" />*

*<span></span>*

*Нет*

*</label>*

*</fieldset>*

*<fieldset>*

*<label>*

*<input class=”visually-hidden” type="checkbox" name="tv-show" />*

*<span></span>*

*Новости*

*</label>*

*<label>*

*<input class=”visually-hidden” type=" checkbox " name="tv-show" />*

*<span></span>*

*Фантастика*

*</label>*

*</fieldset>*

*</form>*

*.visually-hidden { // css-свойство для скрытия стандартного элемента*

*position: absolute;*

*clip: rect(0 0 0 0); // свойство clip только для старых браузеров*

*clip-path: inset(50%); // свойство clip-path превращает элемент в невидимую глазу точку*

*width: 1px;*

*height: 1px;*

*overflow: hidden;*

*white-space: nowrap;*

*}*

*label span { // рисуем новый элемент checkbox*

*width: 20px;*

*height: 20px;*

*background-color: #feafe8;*

*}*

*input[type='radio'] + span { // рисуем новый элемент radio*

*border-radius: 50%;*

*}*

*input[type='radio']:checked + span { // стилизуем новый элемент radio в статусе checked*

*border: 3px solid #fff;*

*}*

*input[type='checkbox']:checked + span { // стилизуем новый элемент chackbox в статусе checked*

*background-image: url(./images/check.svg);*

*background-repeat: no-repeat;*

*background-size: contain;*

*background-position: center;*

*}*

*input[type='radio']:focus-visible + span, // стилизуем оба новых элемента в статусе focus-visible*

*input[type='checkbox']:focus-visible + span {*

*box-shadow: 0 0 0 5px rgb(254 175 232 / 0.5);*

*}*



Рассмотрим реализацию стилизации выпадающего списка с помощью *html* и *css*:

*select {*

*// заставляем браузер забыть стандартные стили этого элемента*

*-webkit-appearance: none;*

*-moz-appearance: none;*

*appearance: none;*

*// задаем свои стили*

*padding: 10px 15px;*

*border: 1px solid #000;*

*border-radius: 0;*

*background-color: transparent;*

*font-family: inherit;*

*font-size: 15px;*

*font-weight: 800;*

*text-transform: uppercase;*

*// добавим кнопку со стрелочкой*

*background-image: url(../images/up-down.svg);*

*background-size: 25px;*

*background-repeat: no-repeat;*

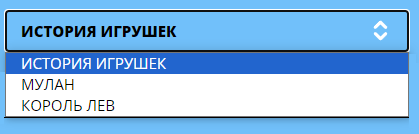
*background-position: center right 15px;*

*}*

*select:focus-visible { // стилизуем элемент в статусе focus-visible*

*box-shadow: 0 0 0 5px rgb(254 175 232 / 0.5);*

*}*



Рассмотрим реализацию стилизации элемента *range*:

*input[type="range"] {*

*// заставляем браузер забыть стандартные стили этого элемента*

*-webkit-appearance: none;*

*-moz-appearance: none;*

*appearance: none;*

*// задаем свои стили*

*width: 100%;*

*background-color: transparent;*

*}*

*input[type="range"]:focus { // стилизуем элемент в фокусе*

*outline: 2px solid #000;*

*outline-offset: 2px;*

*}*

*// задаем стилизацию ползунка для браузеров на Chromium*

*input[type="range"]::-webkit-slider-thumb {*

*// повторно сбрасываем стили*

*-webkit-appearance: none;*

*-moz-appearance: none;*

*appearance: none;*

*// задаем свои стили*

*border: 1px solid #000;*

*height: 16px;*

*width: 26px;*

*border-radius: 0;*

*background-color: #fff;*

*cursor: pointer; // курсор в виде руки*

*margin-top: -3px;*

*}*

*// задаем стилизацию полоски для браузеров на Chromium*

*input[type="range"]::-webkit-slider-runnable-track {*

*width: 100%;*

*height: 10px;*

*cursor: pointer;*

*background-color: #000;*

*border-radius: 0;*

*border: none;*

*}*

*// задаем стилизацию ползунка для Firefox*

*input[type="range"]::-moz-range-thumb {*

*box-sizing: border-box;*

*border: 1px solid #000;*

*height: 16px;*

*width: 26px;*

*border-radius: 0;*

*background-color: #fff;*

*cursor: pointer; // курсор в виде руки*

*}*

*// задаем стилизацию полоски для Firefox*

*input[type="range"]::-moz-range-track {*

*width: 100%;*

*height: 10px;*

*cursor: pointer;*

*background-color: #000;*

*border-radius: 0;*

*border: none;*

*}*

