Для комментирования строк используются специальные символы:

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

*/\* в CSS — такими \*/*

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

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

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

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

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

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

*@import url(fonts.css)*

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

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

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

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

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

*div {*

*color: tomato !important;*

*}*

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

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

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

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

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

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

*// в стилях селектор класса начинается с точки*

*.white-text {*

*color: #fff;*

*}*

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

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

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

*h2 { // селектор по тегу h2*

*font-size: 32px;*

*}*

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

*font-size: 32px;*

*}*

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

*font-size: 32px;*

*}*

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

*font-size: 32px;*

*}*

*div p { // все p внутри div*

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

*}*

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

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

*}*

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

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

*}*

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

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

*}*

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

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

*}*

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

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

*}*

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

*font-size: 32px;*

*}*

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

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

*}*

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

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

*margin: 0;*

*font-size: 100%;*

*font-weight: normal;*

*}*

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

*\*,*

*\*::before,*

*\*::after {*

*box-sizing: border-box;*

*}*

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

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

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

Также существует комбинированный тип – **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 родительского элемента нужно задать нулевой размер шрифта:

*font-size: 0*;

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

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

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

Лайфхак для настройки размеров на мобильных устройствах:

*html {*

*height: 100%;*

*}*

*body {*

*height: 100%;*

*}*

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

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

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

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

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

**При резиновой верстке** элементы растягиваются: пропорционально масштабируются под размеры окна, но не меняют свое положение и внешний вид. Размеры задаются в относительные единицы измерения *%*, *vw*, *vh*.

**При адаптивной верстке** внешний вид сайта меняется, а элементы перестраиваются в зависимости от размера браузера. Пишутся *CSS*-правила под разные условия (размеры окна).

**При отзывчивой верстке** совмещается резиновая и адаптивная верстка. Благодаря резиновой верстке переход из одного состояния в другое происходит плавно, а благодаря адаптивной верстке в определенных точках перестраивается внешний вид.

При любом из подходов для сайта верстают несколько состояний: для телефона, планшета, ноутбука, десктопа. Используются подходы ***desktop first*** и ***mobile first***.

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

*rgb(255, 255, 0) // желтый*

*#FFFF00 // желтый*

*#FF0 // желтый*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*.parent {*

*font-size: 24px;*

*}*

*.child {*

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

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

*}*

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

*.parent {*

*font-size: 20px;*

*}*

*.element {*

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

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

*}*

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

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

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

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

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

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

***cursive*** – наклонный системный шрифт

***fantasy*** – декоративный системный шрифт

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

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

*font-size: 24px;*

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

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

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

*font-weight: normal;*

*font-weight: 400;*

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

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

* ***font-style*** 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

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/> .

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

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

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

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

*overflow: scroll; // включает скролл всегда*

*overflow: hidden; // скрывает лишний контент*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

*white-space: pre-line;*

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

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

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

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

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

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

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



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

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

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

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

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

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

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

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

*padding: 10px 20px 30px;*

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

*padding: 10px 20px;*

*// 10px со всех сторон*

*padding: 10px;*

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

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



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



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



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*

*}*

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;*

*}*

*}*

Для хранения значений в *CSS* используются **переменные**. Название переменных начинается с двух дефисов в начале. Имя переменной может содержать только английские буквы, символ дефиса и цифры. Браузер чувствителен к регистру в именах переменных. На переменные распространяются правила каскада.

Формально для создания переменных используется селектор ***:root***. Однако их можно объявлять в любом месте.

Для использования в коде переменных понадобится функций ***var()***.

*:root {*

*--main-text-color: #000; // создаем переменную*

*}*

*body {*

*color: var(--main-text-color); // используем переменную*

*}*

Если при вызове переменной она не определена, то браузер не применит его к элементу. Для этого принято использовать **значения по умолчанию**.

*button {*

*box-shadow: var(--offset, 10px) 15px 0 0 #feafe8; // значение по умолчанию 10px*

*}*

*html {*

*font-family: var(--fonts, Helvetica, Arial, sans-serif); // запасные шрифты по умолчанию*

*}*

Теперь, благодаря переменным, исчезла необходимость переписывать всю функцию целиком для изменения параметров. При совпадении набора селекторов переменная будет попадать в общее поле с функцией, независимо от порядка селекторов.

*<div class="container">*

*<h1 class="text title">Придёт серенький волчок…</h1>*

*<h2 class="text description">И укусит за бочок \*</h2>*

*<p class="text note">Не является публичной офертой</p>*

*</div>*

*.text {*

*color: rgb(0 0 0 / var(--opacity, 1);*

*}*

*.description {*

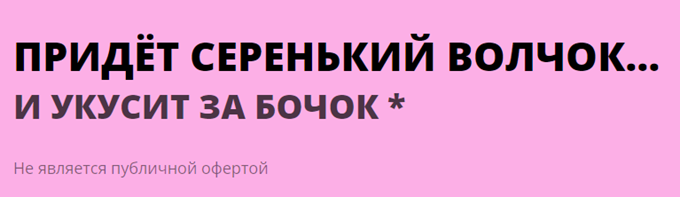
*--opacity: 0.8;*

*}*

*.note {*

*--opacity: 0.5;*

*}*



Рассмотрим пример реализации смены темной и светлых тем с помощью *CSS*-переменных.

*<body>*

*<div class="theme-toggler">*

*<label class="theme-toggler\_\_label">*

*Переключить тему*

*<input type="checkbox" class="theme-toggler\_\_control" />*

*</label>*

*</div>*

*</body>*

*body {*

*--main-text-color: #152028;*

*--main-background-color: #fff;*

*color: var(--main-text-color);*

*background-color: var(--main-background-color);*

*}*

*body:has(.theme-toggler\_\_control:checked) { // меняем значения переменных местами*

*--main-text-color: #fff;*

*--main-background-color: #152028;*

*}*

