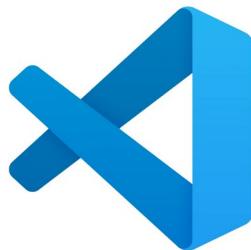


Учебник по работе с Visual Studio Code для верстки сайтов

Для начинающих веб-мастеров старшего поколения



Оглавление

Введение: Начнем с правильного инструмента

Для кого эта книга? (Для пенсионеров, которые хотят научиться верстать сайты с помощью современного и удобного инструмента)

Почему мы выбрали VS Code? Потому что он бесплатный, мощный и стал стандартом для веб-разработки.

Это не сложно! Интерфейс VS Code интуитивно понятен, а мы поможем разобраться.

Что вы узнаете: Как настроить VS Code для комфортной верстки и использовать его основные возможности.

Часть 1: Знакомство и первая настройка VS Code

Глава 1: Устанавливаем и запускаем VS Code

Альтернативные способы скачивания:

Через Microsoft Store (самый простой способ)

Через надежные российские зеркала или файлообменники

Попросить близких помочь со скачиванием

Процесс установки: "Далее, далее, готово"

Первый запуск: что мы видим на экране?

Глава 2: Настраиваем VS Code "под себя"

Увеличиваем шрифт редактора и интерфейса
Включаем тему "Для слабого зрения"
Настраиваем масштаб интерфейса
Сохраняем настройки

Глава 3: Знакомство с интерфейсом

Панель активности (слева)
Проводник файлов
Основная область редактирования
Панель состояния (внизу)
Боковая панель

Часть 2: Организация работы над сайтом

Глава 4: Первый проект - создаем папку для сайта

Как создать папку для проекта на Рабочем столе
Как открыть папку в VS Code
Создаем первую структуру файлов:
index.html
style.css
images (папка для картинок)

Глава 5: Проводник файлов - ваш главный помощник

Как создавать новые файлы и папки прямо в VS Code
Как переименовывать и удалять файлы
Быстрая навигация между файлами проекта

Часть 3: Основные возможности для верстки

Глава 6: Интеллектуальное редактирование

Подсветка синтаксиса - как в Notepad++
Автодополнение тегов - VS Code подсказывает продолжение
Автозакрытие тегов - программа сама добавляет закрывающие теги

Глава 7: Работа с вкладками и разделение экрана

Как открыть несколько файлов одновременно
Разделение экрана для работы с HTML и CSS одновременно

Горячие клавиши для переключения между вкладками

Глава 8: Поиск и замена

Простой поиск в одном файле

Поиск и замена во всем проекте

Как найти и исправить ошибку во всех файлах сразу

Часть 4: Полезные расширения для верстки

Глава 9: Магазин расширений

Как открыть магазин расширений

Поиск и установка нужных расширений

Безопасность: устанавливаем только проверенные расширения

Глава 10: Самые нужные расширения для начинающих

Live Server - просмотр сайта с автоматическим обновлением

Russian Language Pack - русский интерфейс

Auto Rename Tag - автоматическое переименование парных тегов

Глава 11: Emmet - верстка на скорости

Что такое Emmet и как он ускоряет работу

Основные команды Emmet для HTML

Практические примеры быстрой верстки

Часть 5: Отладка и проверка кода

Глава 12: Подсказки и обнаружение ошибок

Как VS Code подсвечивает ошибки

Всплывающие подсказки по HTML и CSS

Проверка валидности кода

Глава 13: Встроенный терминал

Что такое терминал и зачем он нужен

Как открыть терминал в VS Code

Простые команды для навигации по папкам

Часть 6: Советы и хитрости для ежедневной работы

Глава 14: Горячие клавиши для экономии времени

Сохранение файла
Копирование и вставка
Комментирование кода
Быстрое форматирование текста

Глава 15: Работа с цветами

Встроенный выбор цветов для CSS
Пипетка для копирования цветов
Палитра цветов с предпросмотром

Приложения

Приложение А: Шпаргалка по горячим клавишам

Приложение Б: Список полезных расширений

Приложение В: Частые проблемы и их решение

Приложение Г: Ресурсы для дальнейшего обучения

Оглавление

Введение: Начнем с правильного инструмента

Для кого эта книга? (Для пенсионеров, которые хотят научиться верстать сайты с помощью современного и удобного инструмента)

Уважаемый читатель!

Если вы держите в руках эту книгу, значит, вы — человек, который не смотрит на годы как на преграду, а видит в них возможность научиться чему-то новому. Эта книга написана специально для вас — для людей старшего поколения, которые хотят не просто «быть в теме», а активно осваивать современные технологии.

Эта книга для вас, если:

Вам интересно, как создаются сайты. Вы всегда задумывались, что стоит за страницами, которые вы видите в интернете, и хотите понять эту «магию» изнутри.

Вы хотите найти увлекательное хобби, которое развивает логику, память и творческое мышление. Верстка сайтов — это современное ремесло, очень похожее на сборку конструктора или рукоделие, только в цифровом пространстве.

Вам нужен практический результат. Вы мечтаете создать свою личную страничку-визитку, сайт для дачного кооператива, клуба по интересам или просто красивый фотоальбом о своей семье.

Вы цените порядок и структуру. Вам нравится, когда всё разложено по полочкам. Процесс верстки — это как раз про создание четкой и логичной структуры, которую потом можно красиво оформить.

Вы пробовали изучать программирование, но пугала сложность. Здесь мы начнем с самого понятного и наглядного — с создания внешнего вида страницы. Вы сразу будете видеть плоды своего труда в браузере.

Вам важно учиться в комфортном темпе, без спешки и стресса. Мы будем разбирать каждый шаг максимально подробно, повторять пройденное и никуда не торопиться.

Что ждет вас в этой книге?

Никакой лишней информации и сложной терминологии. Мы будем использовать простые аналогии из жизни. Например, HTML — это «каркас дома» (стены, окна, двери), а CSS — это его «отделка и дизайн» (обои, цвет стен, расстановка мебели).

Максимум практики и наглядности. Мы не будем грузить вас голой теорией. С самого начала вы начнете создавать реальные вещи: сначала простой текст, потом — заголовки, списки, изображения, и в итоге — целую страницу.

Помощь современного помощника — VS Code. Этот редактор не просто программа для набора кода. Он ваш умный помощник: он будет подсвечивать ошибки, подсказывать следующие шаги и автоматически дополнять код. Это как умная ручка, которая помогает писать без ошибок.

Акцент на понимание, а не на зубрежку. Мы объясним, почему нужно делать именно так, чтобы вы могли не просто копировать примеры, а понимать логику и создавать свое.

Главное наше послание к вам:

Не бойтесь! Современные инструменты, такие как VS Code, созданы для того, чтобы помогать, а не усложнять. Вы не останетесь один на один с непонятными символами. Программа будет вашим гидом, а эта книга — вашим терпеливым и понятливым учителем.

Давайте вместе откроем дверь в увлекательный мир веб-верстки!

Ваша мудрость, жизненный опыт и наше руководство — вот идеальная формула для успеха.

Почему мы выбрали VS Code? Потому что он бесплатный, мощный и стал стандартом для веб-разработки.

Дорогой друг!

Мы прекрасно понимаем, что когда начинаешь что-то новое, особенно в возрасте, хочется выбрать инструмент попроще. Возможно, вы слышали о стандартном «Блокноте» или других простых программах. Но мы сознательно предлагаем вам начать с **Visual Studio Code (VS Code)**, и вот почему это — лучший выбор для вас.

1. Он бесплатный — и это честно!

Никаких скрытых платежей. VS Code полностью бесплатен — это официальная политика компании Microsoft. Вам не нужно беспокоиться о взломах, пиратских версиях или внезапных требованиях оплаты.

Без риска для компьютера. Скачивая программу с официального сайта, вы защищаете свой компьютер от вирусов и вредоносных программ.

Экономия средств. Не нужно тратить пенсию на дорогое программное обеспечение. Все необходимое для профессиональной верстки у вас уже будет.

2. Он стал стандартом — учимся на лучшем!

Как все профессионалы. 75% веб-разработчиков во всем мире используют именно VS Code. Начиная с него, вы учитесь на том же инструменте, что и специалисты.

Огромное сообщество. Если у вас возникнет вопрос, тысячи людей уже сталкивались с такой же проблемой и помогут решить ее на форумах и в видеоуроках.

Актуальные знания. Все современные курсы и учебники ориентированы на VS Code. Вам не придется переучиваться с устаревших программ.

3. Он — ваш умный помощник, а не просто текстовый редактор

Подсказки как у автоответчика. VS Code будет предлагать продолжение ваших мыслей:

html

```
<!-- Начните писать <h1 — программа сама предложит </h1> -->
<h1>Заголовок</h1>
```

Подсветка ошибок. Если вы забудете закрыть кавычку или скобку, программа покрасит ошибку в красный цвет — невозможно не заметить!

Автоматическое форматирование. Приведет ваш код в порядок одним нажатием кнопки — все будет ровно и аккуратно.

4. Интуитивно понятный интерфейс

Похож на проводник Windows. Слева — папки вашего проекта, справа — область для редактирования, точно как в знакомом проводнике.

Легко увеличить шрифт. В два клика можно сделать текст достаточно крупным для комфортной работы.

Темы для глаз. Есть специальные темы с высоким контрастом и темные режимы, которые берегут зрение.

5. «Живой сервер» — волшебная кнопка для мгновенного результата

Видите изменения сразу. Напишете код — просто нажмите кнопку — и ваш сайт откроется в браузере, причем все изменения будут видны мгновенно, без перезагрузки страницы.

Работает на вашем компьютере. Не нужно подключаться к интернету, чтобы увидеть результат своей работы.

6. Русский язык и поддержка

Русскоязычный интерфейс. Установите официальное расширение «Russian Language Pack», и меню станет на родном языке.

Подробные уроки на YouTube. По VS Code существуют тысячи видеоуроков на русском языке, где каждый шаг показан максимально подробно.

Почему НЕ другие программы?

Обычный Блокнот: Слишком примитивен, нет подсветки ошибок, легко запутаться

Notepad++: Хорошая программа, но уже устаревает для веб-разработки

Другие редакторы: Часто платные или слишком сложные для начинающих

Наше обещание к вам:

Мы научим вас не просто «писать код», а комфортно работать в профессиональной среде. Да, сначала может показаться непривычно, но уже через 2-3 занятия вы поймете, насколько VS Code удобен и продуман.

Это как научиться водить сразу на автомобиле с АКПП — проще, комфортнее и безопаснее, чем осваивать старую механическую коробку передач!

Это не сложно! Интерфейс VS Code интуитивно понятен, а мы поможем разобраться.

Дорогой друг!

Когда вы впервые откроете Visual Studio Code, может показаться, что перед вами панель управления космическим кораблем — много кнопочек, непонятных меню и значков. Но не пугайтесь! Давайте вместе разберемся, и вы убедитесь, что всё здесь создано для вашего удобства.

1. «Знакомые черты» — вы уже это видели!

Посмотрите на интерфейс VS Code — он устроен так же логично, как и многие другие программы:

Слева — как «Проводник» в Windows — здесь вы видите все ваши файлы и папки. Точно так же, как вы находите фотографии в папке «Мои изображения», здесь вы найдете все файлы вашего сайта.

Сверху — знакомое меню — «Файл», «Правка», «Вид» — те же разделы, что и в Word или Блокноте.

В центре — обычный лист для письма — большая белая область, где вы будете набирать текст (код), совсем как в обычном текстовом редакторе.

2. Главные элементы, которые нужно знать (всего 4 штуки!)

Давайте запомним только самое необходимое — остальное пока можете смело игнорировать!

① САМОЕ ВАЖНОЕ: Панель деятельности (слева)

text

- [] Проводник — ваши файлы и папки
- [] Поиск — чтобы найти что-то в проекте
- [] Git — пока не нужно!
- [] Запуск — для тестирования
- [] Расширения — магазин дополнений

Смотрите сюда, если что-то не работает — VS Code часто показывает здесь подсказки об ошибках.

② Область редактирования (центр)

Здесь вы работаете с текстом — всё просто и привычно, как в обычном блокноте!

③ Панель вкладок (сверху)

Как в браузере — переключайтесь между открытыми файлами простым щелчком.

3. «Страшные» кнопки, которые на самом деле — ваши друзья

Вот что кажется сложным, но это — ваши помощники:

text

[] Terminal — это просто «командная строка», но она вам не понадобится сразу

[] Debug — для поиска ошибок, но VS Code и так подсвечивает их цветом

[] Settings — настройки, но мы настроим всё за вас

[] Extensions — «магазин приложений» для VS Code, где вы найдете полезные дополнения

4. Наша стратегия обучения — «Ничего лишнего!»

Мы будем открывать программу постепенно:

Первая неделя: Работаем только с «Проводником» и областью редактирования

Вторая неделя: Подключаем «Поиск» и учимся находить файлы

Третья неделя: Знакомимся с «Расширениями» и устанавливаем 2-3 полезные дополнения

5. Почему это ПРОЩЕ, чем кажется?

Цветовые подсказки — правильный код окрашивается в приятные цвета, ошибки — в красный. Не нужно быть экспертом, чтобы понять, где ошибка!

Всплывающие советы — наведите курсор на непонятное слово, и VS Code объяснит, что оно значит.

Автодополнение — программа сама предлагает варианты, как может продолжиться ваша фраза — как Т9 в телефоне!

6. Запомните главное:

Не пытайтесь выучить всё сразу — это невозможно и не нужно!

Сосредоточьтесь на том, что используете прямо сейчас — остальное подождет.

Ошибки — это нормально! VS Code поможет их найти и исправить.

Мы будем двигаться маленькими шагами — каждый день осваивая по одной новой кнопке.

Представьте, что VS Code — это ваш новый автомобиль.

Вам не нужно знать, как работает двигатель, чтобы ездить на нем.

Достаточно понять, где руль, педали газа и тормоза. Всему остальному научитесь постепенно, в процессе вождения!

Мы будем вашим инструктором по вождению этого «автомобиля».
Обещаем — через неделю вы будете чувствовать себя за рулем уверенно и комфортно!

Что вы узнаете: Как настроить VS Code для комфортной верстки и использовать его основные возможности.

Дорогой друг!

Давайте конкретно разберем, каким волшебным навыкам вы научитесь благодаря этой книге. Мы не будем грузить вас лишней информацией — только то, что действительно пригодится для создания сайтов.

Часть 1: Превращаем VS Code в удобный инструмент именно для вас

1. Настройки «для зоркого глаза»:

Увеличиваем шрифт — сделаем текст достаточно крупным, чтобы не напрягать зрение

Выбираем удобную цветовую схему — светлую или темную, в зависимости от времени суток и ваших предпочтений

Настраиваем межстрочный интервал — чтобы текст был хорошо читаемым

Включаем перенос строк — чтобы не бегать горизонтальным ползунком туда-сюда

2. Организация рабочего пространства:

Создаем папку проекта — учимся хранить все файлы сайта в одном месте, как в обычной папке на столе

Работаем с проводником — осваиваем навигацию по файлам вашего сайта

Создаем структуру папок — отдельно для изображений, отдельно для стилей

Пример настройки крупного шрифта:

```
json
// Это можно будет сделать через меню настроек
{
```

```
"editor.fontSize": 18,  
"editor.lineHeight": 1.5  
}
```

Часть 2: Основные возможности — ваш ежедневный инструментарий

3. Интеллектуальная работа с кодом:

Подсветка синтаксиса — разные элементы кода будут разного цвета, как в детской раскраске

Автодополнение тегов — программа сама предлагает варианты, как может продолжиться ваша мысль

Автозакрытие тегов — когда вы пишете `<h1>`, VS Code автоматически добавит `</h1>`

4. Эффективная навигация:

Работа с вкладками — как переключаться между HTML и CSS файлами

Разделение экрана — как одновременно видеть HTML и CSS код

Быстрый поиск — как найти нужное слово в большом файле

Часть 3: Полезные хитрости, которые экономят время

5. Горячие клавиши для самых частых действий:

`Ctrl+S` — сохранить файл (ваша самая важная привычка!)

`Ctrl+Z` — отменить последнее действие (если что-то испортили)

`Ctrl+F` — найти текст в файле

`Ctrl+/` — закомментировать строку

6. Расширения, которые сделают работу приятнее:

Live Server — одна кнопка — и ваш сайт открывается в браузере с автоматическим обновлением

Russian Language Pack — русский интерфейс для программы

Auto Rename Tag — автоматически переименовывает парные теги

Часть 4: Конкретные навыки для верстки

7. Для работы с HTML:

Шаблоны быстрой верстки с помощью Emmet
Проверка правильности закрытия тегов
Быстрая навигация по структуре HTML-документа

8. Для работы с CSS:

Встроенный выбор цветов с палитрой
Подсказки по CSS-свойствам
Проверка правильности написания свойств

9. Поиск и исправление ошибок:

Как читать подсказки об ошибках
Как находить незакрытые теги
Как исправлять опечатки в названиях свойств

Что вы будете уметь после прохождения курса:

Создавать и организовывать папки проектов

Быстро переключаться между файлами HTML и CSS

Использовать автодополнение для ускорения верстки

Находить и исправлять основные ошибки в коде

Сразу видеть результат своей работы в браузере

Настраивать программу под свои потребности

Самое главное: вы не просто узнаете о возможностях VS Code — вы научитесь применять их на практике. Каждая настройка, каждая кнопка будет рассмотрена на конкретных примерах из верстки реальных страниц.

Мы будем двигаться от простого к сложному:

Сначала научимся создавать файлы и папки
Потом настроим комфортный интерфейс

Затем освоим основные инструменты редактирования
И наконец — познакомимся с продвинутыми возможностями

**К концу обучения VS Code станет для вас таким же привычным
инструментом, как молоток для плотника или спицы для
вязальщицы!**

Часть 1: Знакомство и первая настройка VS Code

Глава 1: Устанавливаем и запускаем VS Code

Альтернативные способы скачивания:

Через Microsoft Store (самый простой способ)

Дорогой друг!

Если официальный сайт не открывается, не переживайте! У нас есть самый простой и безопасный способ установки — через **Microsoft Store**. Это как магазин приложений в вашем телефоне, только для компьютера. Здесь всё проверено и точно не содержит вирусов.

Пошаговая инструкция (с картинками в мыслях)

Шаг 1: Открываем Microsoft Store

Нажмите на кнопку «**Пуск**» в левом нижнем углу экрана (значок Windows). В появившемся меню найдите надпись «**Microsoft Store**» (обычно это значок с сумкой).

Если не можете найти — просто начните печатать на

клавиатуре «**Store**» — система сама найдет магазин

Совет: можете представить, что это как «Магазин приложений» на вашем телефоне, только для компьютера.

Шаг 2: Ищем Visual Studio Code

В открывшемся магазине в правом верхнем углу есть строка поиска (лупа).

Нажмите на нее и напечатайте: «**Visual Studio Code**»

Нажмите Enter или на значок поиска

Не пугайтесь! Вы увидите несколько программ с похожими названиями.

Нам нужна именно та, где написано «Visual Studio Code» и есть синий значок.

Шаг 3: Устанавливаем программу

На странице программы найдите большую синюю

кнопку «**Получить**» или «**Установить**»

Нажмите на эту кнопку — установка начнется автоматически

Ничего не нужно выбирать или настраивать! Программа установится сама

Это займет несколько минут. Вы увидите прогресс установки — можете попить чай, пока компьютер работает.

Шаг 4: Запускаем VS Code

После установки кнопка «Получить» изменится на «Запустить»

Нажмите ее — программа откроется!

Также вы всегда сможете найти VS Code в меню «Пуск» — просто начните печатать «Visual Studio»

Преимущества этого способа:

- Автоматические обновления** — программа сама будет обновляться до новой версии
- Безопасно** — все приложения в Microsoft Store проверены на вирусы
- Просто** — не нужно выбирать папку для установки или настраивать параметры
- Надежно** — если что-то пойдет не так, можно легко переустановить

Если возникли проблемы:

«У меня нет Microsoft Store»

Такое бывает на некоторых версиях Windows. Не переживайте! У нас есть другие способы.

«Не находит программу в поиске»

Проверьте, правильно ли вы напечатали название

Попробуйте ввести просто «VS Code»

«Требует учетную запись Microsoft»

Да, для работы магазина нужна учетная запись. Но она у вас скорее всего уже есть — это тот же логин и пароль, которые вы используете для входа в Windows.

Что будет после установки:

На рабочем столе может появиться ярлык VS Code (синий значок)

В меню «Пуск» появится программа в списке установленных

При первом запуске программа спросит о цветовой теме — выберите ту, которая вам больше нравится!

Запомните: это самый простой способ! Не нужно ничего скачивать вручную, не нужно распаковывать архивы — всего несколько кликов, и профессиональный редактор для верстки у вас на компьютере.

Если что-то получается — не стесняйтесь попросить помощи у близких или наших консультантов. Мы поможем!

Через надежные российские зеркала или файлообменники

Дорогой друг!

Если Microsoft Store по какой-то причине не работает, не отчайвайтесь! Существуют другие безопасные способы скачать программу. Давайте разберем их максимально подробно.

Что такое «зеркала» и почему они безопасны?

«**Зеркало**» — это точная копия официального сайта, расположенная на другом сервере. Представьте, что у вас есть любимая книга, и библиотека делает ее точную копию, чтобы читатели могли прочитать. Так же и с программами!

Почему это безопасно:

Это те же самые файлы, что и на официальном сайте
Их проверяют на наличие вирусов
Часто такие зеркала создают сами разработчики для разных стран

Пошаговая инструкция по скачиванию через зеркала

Шаг 1: Поиск надежного зеркала

Откройте любой поисковик (Яндекс, Google)
Ведите запрос: «**Visual Studio Code скачать Россия**» или «**VS Code зеркало**»
В результатах поиска обращайте внимание на:
Сайты крупных IT-компаний (например, RU-CENTER)
Образовательные порталы университетов
Крупные файловые хостинги

Проверенные варианты:

В поисковой строке браузера (в данном примере — в Яндексе пишем «официальный сайт Visual Studio Code»). После этого Яндекс нам покажет результаты поиска. Нас будет интересовать, как показано на скриншоте ниже, самый первый сайт.



Яндекс

официальный сайт Visual



официальный сайт Visual Studio Code

[поиск](#)[алиса](#)[картинки](#)[видео](#)[карты](#)[товары](#)[финансы](#)[квартиры](#)

✖ [Visual Studio Code - The open source AI code editor](#)

code.visualstudio.com

Visual Studio Code redefines AI-powered **coding** with GitHub Copilot for building and debugging modern web and cloud applications. **Visual Studio Code** is free and available on your favorite platform - Linux, macOS, and Windows.

[Скачать](#) · [Настройка](#)

■ [Visual Studio: IDE and Code Editor for Software Development](#)

visualstudio.microsoft.com

Visual Studio dev tools & services make app development easy for any developer, on any platform & language. Develop with our **code** editor or IDE anywhere for free.

[Visual Studio Express](#) · [Поддержка Visual Studio](#) · [Разработка](#)

[soft.ru](#) — российский портал программного обеспечения

[download.ru](#) — каталог программ для России

[mydiv.net](#) — популярный архив программ

Шаг 2: Скачивание правильной версии

На странице загрузки вы увидите несколько вариантов:

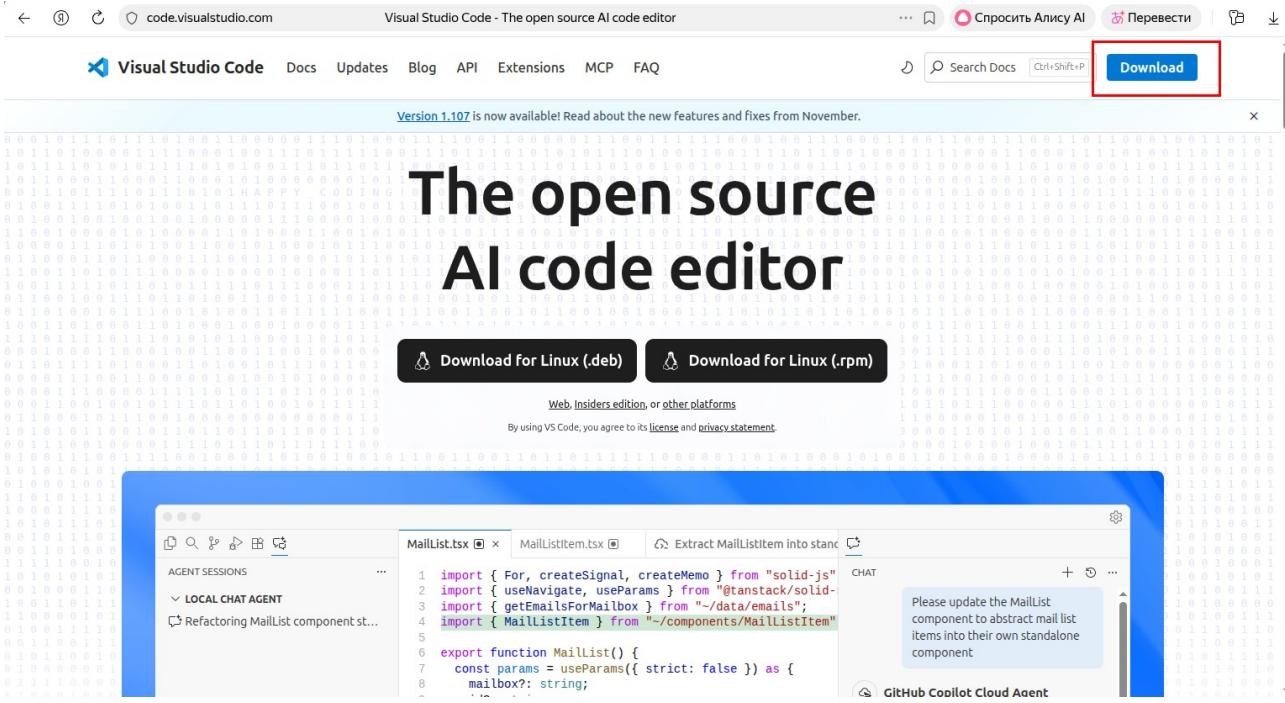
Windows — для обычных компьютеров и ноутбуков

User Installer — рекомендуем эту версию! (установщик для пользователя)

System Installer — для системных администраторов (не нужно)

64-bit — подходит для большинства современных компьютеров

Просто нажмите на «User Installer, 64-bit» — это правильный выбор в 99% случаев!



Шаг 3: Процесс скачивания

Нажмите на кнопку «Скачать» или «Download»

Файл будет называться примерно так: **VSCodeUserSetup-x64-1.XX.X.exe**

Запомните, куда сохраняется файл (обычно это папка «Загрузки»)

Скачивание через файлообменники

Популярные безопасные файлообменники:

Yandex Disk (Яндекс.Диск)

Mail.ru Cloud (Облако Mail.ru)

Dropbox (если доступен)

Как это работает:

Кто-то из пользователей уже скачал официальную версию и выложил в общее облако

Вы получаете готовую ссылку для скачивания

Часто скорость скачивания даже выше, чем с официального сайта

Поиск ссылок:

В поисковике: «**VS Code скачать Яндекс.Диск**»

На технических форумах (например, Habr)
В тематических группах в социальных сетях

Меры безопасности при скачивании

Красные флаги (чего стоит избегать):

Сайты, требующие отправки СМС для скачивания
Файлы с расширением .rar или .zip (должен быть .exe)
Слишком маленький размер файла (меньше 50 МБ)
Сайты с большим количеством навязчивой рекламы

Признаки безопасного файла:

Размер около 70-100 МБ
Расширение .exe
Название соответствует официальному (VSCodeUserSetup-x64-...)
Дата публикации не старше 1-2 месяцев

Процесс установки из скачанного файла

Найдите скачанный файл в папке «Загрузки»
Дважды щелкните по нему левой кнопкой мыши
Появится окно установки — нажмите «Да» (если спросит разрешение)
Следуйте простым шагам:
Примите лицензионное соглашение
Выберите папку для установки (оставьте по умолчанию)
Создайте значок на рабочем столе (рекомендуем!)
Нажмите «Установить»
Через 1-2 минуты установка завершится
Нажмите «Готово» и запустите программу

Если что-то пошло не так:

Антивирус ругается — это нормально! Добавьте программу в исключения. VS Code — безопасная программа.

Файл не запускается — возможно, скачался битый файл. Попробуйте скачать с другого зеркала.

Возникают ошибки при установке — закройте все программы и попробуйте еще раз.

Помните: тысячи людей в России успешно скачивают VS Code через зеркала. Это обычная практика, а не что-то рискованное!

Главное — скачивайте с крупных проверенных порталов, и у вас всё получится!

Попросить близких помочь со скачиванием

Дорогой друг!

Если компьютерные термины и процессы скачивания кажутся слишком сложными, есть простой и надежный способ — попросить помощи у близких. Это абсолютно нормально и правильно! Давайте разберем, как сделать это максимально эффективно.

Почему это хорошая идея?

Экономия нервов и времени — не нужно разбираться в том, что вызывает затруднения

Безопасность - молодое поколение лучше ориентируется в интернете и отличает надежные сайты от подозрительных

Укрепление связей - совместные занятия с внуками или детьми сближают

Быстрый результат - вместо часов самостоятельных попыток вы получите готовую программу за 15 минут

Как правильно попросить о помощи

Подготовьтесь заранее:

Узнайте название программы - запишите на бумажке: "**Visual Studio Code**"

Знайте цель - "Мне нужно установить программу для создания сайтов"

Выберите удобное время - когда у помощника будет 15-20 свободных минут

Примерный разговор:

"Сашенька, у меня есть просьба. Я начал(а) изучать создание сайтов, и мне нужно установить на компьютер программу Visual Studio Code. Можешь помочь мне ее скачать и установить? Я пока не очень разбираюсь в этом."

Что попросить сделать конкретно

Вариант 1: Полная помощь

"Пожалуйста, скачай и установи мне программу Visual Studio Code. Я хочу начать с ней работать."

Вариант 2: Помощь с скачиванием

"Помоги, пожалуйста, найти официальный сайт и скачать программу. А установку я попробую сам(а)."

Вариант 3: Дистанционная помощь

"Можешь найти надежную ссылку для скачивания и прислать мне ее в сообщении? Я попробую скачать сам(а)."

Что нужно объяснить помощнику

Название программы - Visual Studio Code (не путать с Visual Studio!)

Для чего вам - "для верстки сайтов, для обучения"

Варианты скачивания:

Через **Microsoft Store** (предпочтительно)

С официального сайта code.visualstudio.com

Через надежные российские зеркала

Контрольный список для помощника

Попросите проверить:

Установлена последняя версия программы

Создан ярлык на рабочем столе

Программа запускается и работает

Установлена русская языковая пачка (если нужно)

Шрифт в программе достаточно крупный

Если помощник тоже не очень разбирается

Можно показать ему эту инструкцию:

Откройте **Microsoft Store** в Windows
В поиске введите "**Visual Studio Code**"
Нажмите "**Получить**" или "**Установить**"
После установки найдите программу в меню "Пуск"
Создайте ярлык на рабочем столе (правой кнопкой мыши → "Закрепить на начальном экране" или "Создать ярлык")

Что делать после установки

Когда программа будет установлена:

Поблагодарите помощника - даже если это близкий человек

Попробуйте запустить программу сами - двойной щелчок по ярлыку на рабочем столе

Убедитесь, что все работает правильно

Запишите, куда установлена программа (обычно: C:\Users\ВашеИмя\AppData\Local\Programs\Microsoft VS Code)

Альтернативные варианты помощи

Попросить соседа - часто соседи с удовольствием помогают

Обратиться в компьютерный клуб - во многих городах есть бесплатные консультации

Воспользоваться услугой "Компьютерная помощь" - многие компании предоставляют такие услуги недорого

Попросить помощи у преподавателя - если вы занимаетесь на курсах

Важные моменты

Не стесняйтесь просить о помощи - это показывает вашу мудрость, а не некомпетентность

Лучше попросить помощи, чем бросить обучение из-за технических сложностей

Большинство людей с радостью помогут - особенно если видят ваш искренний интерес к обучению

Помните: Многие известные программисты начинали с помощи более опытных товарищей. Просить помощи — это нормальная часть процесса обучения в любом возрасте!

Главное — не останавливаться на первом же препятствии. Просить помощи — это разумно, а не стыдно!

Процесс установки: "Далее, далее, готово"

Дорогой друг!

Сейчас мы с вами установим Visual Studio Code. Это очень просто — практически как установка любой другой программы. Я опишу каждый шаг так подробно, что у вас не останется никаких вопросов!

Шаг 1: Запускаем установочный файл

Найдите файл, который вы скачали (обычно он лежит в папке «Загрузки»)

Файл будет называться примерно так: **VSCodeUserSetup-x64-1.XX.X.exe**

Дважды щелкните по нему левой кнопкой мыши

Появится окно с вопросом: «Разрешить этому приложению вносить изменения на вашем устройстве?»

Смело нажимайте «Да»

Шаг 2: Принимаем лицензионное соглашение

Появится окно установщика с заголовком «**Setup - Visual Studio Code**»:

Прочитайте (или просто пролистайте) лицензионное соглашение

Поставьте галочку «**I accept the agreement**» (Я принимаю условия соглашения)

Нажмите кнопку «**Next**» (Далее)

Не переживайте о тексте соглашения — это стандартный юридический документ для всех программ.

Шаг 3: Выбираем папку для установки

Вам предложат выбрать папку для установки:

Не меняйте ничего! Оставьте путь по умолчанию:

C:\Users\ВашеИмя\AppData\Local\Programs\Microsoft VS Code\

Просто нажмите «**Next**»

Почему не нужно менять? Программа сама выбирает правильное место, где ей будет работать лучше всего.

Шаг 4: Выбираем дополнительные задачи

Это важное окно! Здесь нужно поставить несколько галочек:

Обязательно отметьте:

- Create a desktop icon** (Создать значок на рабочем столе) — чтобы легко находить программу
- Add to PATH** (Добавить в PATH) — это поможет в будущем при установке расширений

Можно отметить по желанию:

- Register Code as an editor for supported file types** (Зарегистрировать Code как редактор для поддерживаемых файлов)
 - Add "Open with Code" action** (Добавить действие «Открыть с помощью Code»)
Проще говоря: поставьте все галочки — это сделает работу с программой удобнее!
- Нажмите «**Next**»

Шаг 5: Все готово к установке

Появится окно с заголовком «**Ready to install**» (Готов к установке):
Здесь просто показано, что вы выбрали
Ничего менять не нужно
Нажмайте «**Install**» (Установить)

Шаг 6: Ждем завершения установки

Появится зеленый индикатор выполнения
Установка займет 1-3 минуты
Можно откинуться на спинку стула и расслабиться
В это время компьютер может немного «задуматься» — это нормально!

Шаг 7: Завершаем установку

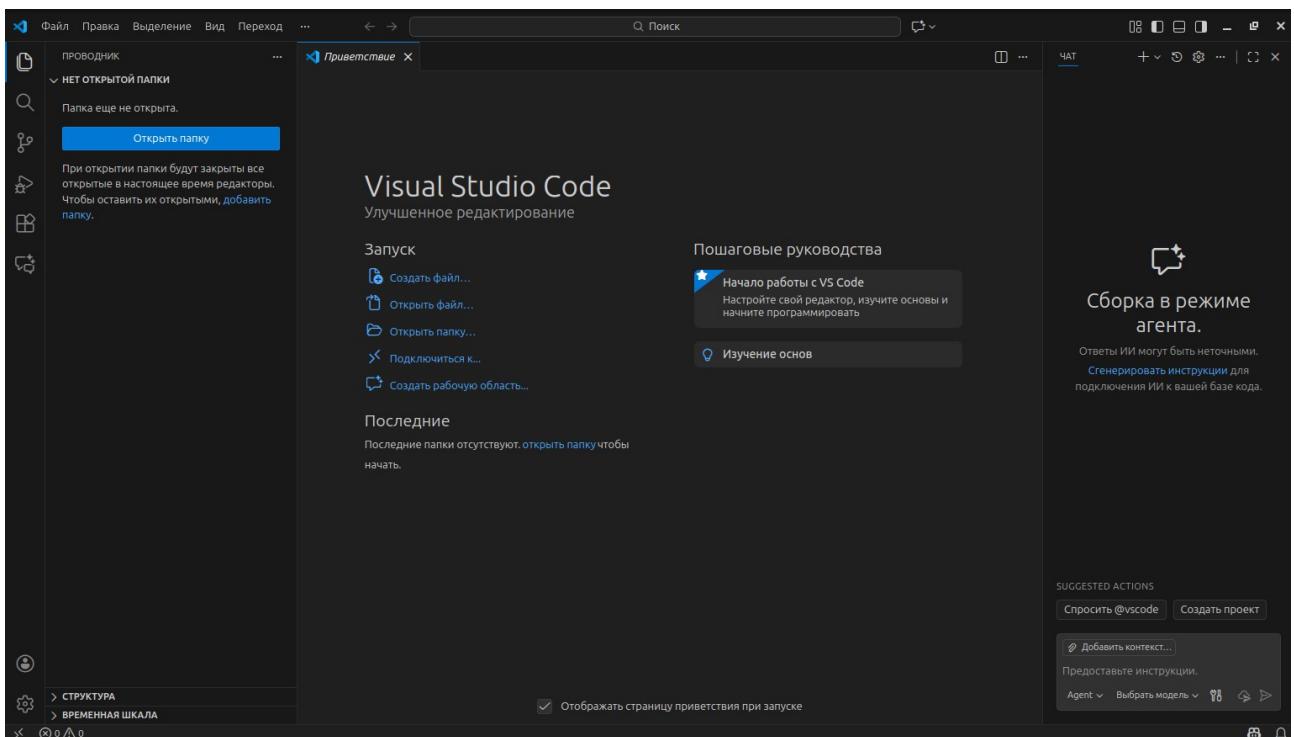
Когда установка завершится, вы увидите:
Кнопку «**Finish**» (Готово)
Галочку «**Launch Visual Studio Code**» (Запустить Visual Studio Code)

Рекомендую:

Оставьте галочку «Launch Visual Studio Code» (чтобы программа сразу запустилась)
Нажмите «**Finish**»

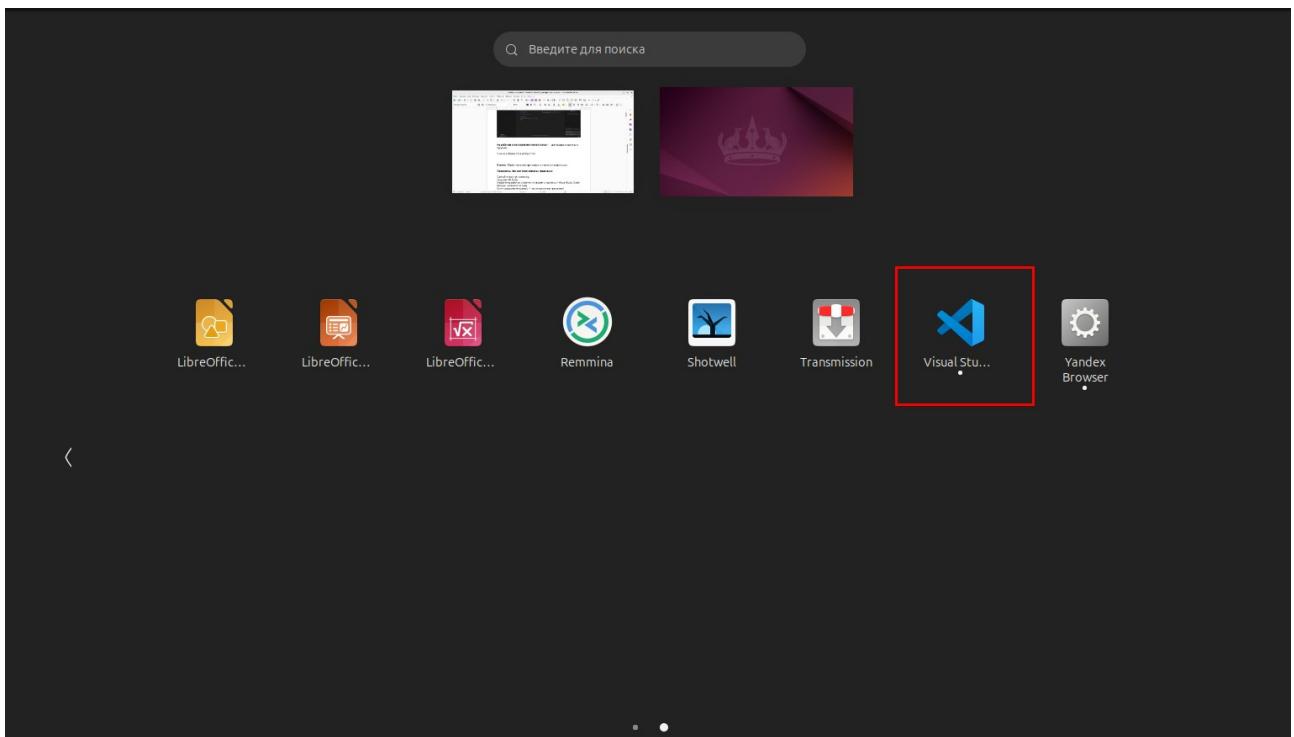
Что произойдет после установки

Сразу запустится VS Code — вы увидите стартовую страницу (сразу оговорюсь, что устанавливал свою версию через Центр приложений установленной у меня операционной системы Ubuntu; расширение с адаптацией к русскому языку уже установил — поэтому всё уже русифицировано😊)



На рабочем столе появится синий значок — для быстрого запуска в будущем

У меня в Ubuntu это выглядит так



В меню «Пуск» появится программа в списке установленных

Проверяем, что все установилось правильно

Сделайте простую проверку:

Закройте VS Code

Найдите на рабочем столе синий значок с надписью «Visual Studio Code»

Дважды щелкните по нему

Если программа открылась — все установлено правильно!

Если что-то пошло не так

Программа не запускается:

Перезагрузите компьютер и попробуйте снова

Попробуйте найти программу через меню «Пуск»

Значок не появился на рабочем столе:

Откройте меню «Пуск»

Найдите «Visual Studio Code»

Нажмите правой кнопкой мыши → «Дополнительно» → «Закрепить на начальном экране»

Возникает ошибка:

Запишите текст ошибки

Попробуйте запустить установочный файл еще раз

Важные моменты

-  **Не прерывайте установку** — дождитесь полного завершения
-  **Не отключайте компьютер** во время установки
-  **Не запускайте другие программы** пока идет установка
-  **Если антивирус спросит разрешение** — разрешите установку

Поздравляю! Вы только что установили профессиональный инструмент для веб-разработки. Это большой шаг в освоении новой профессии!

Теперь можно переходить к настройке программы под свои нужды. Но это мы уже рассмотрим в следующем разделе.

Запомните: процесс установки практически одинаков для всех программ в Windows. Освоив его один раз, вы сможете устанавливать любые нужные программы!

Первый запуск: что мы видим на экране?

Дорогой друг!

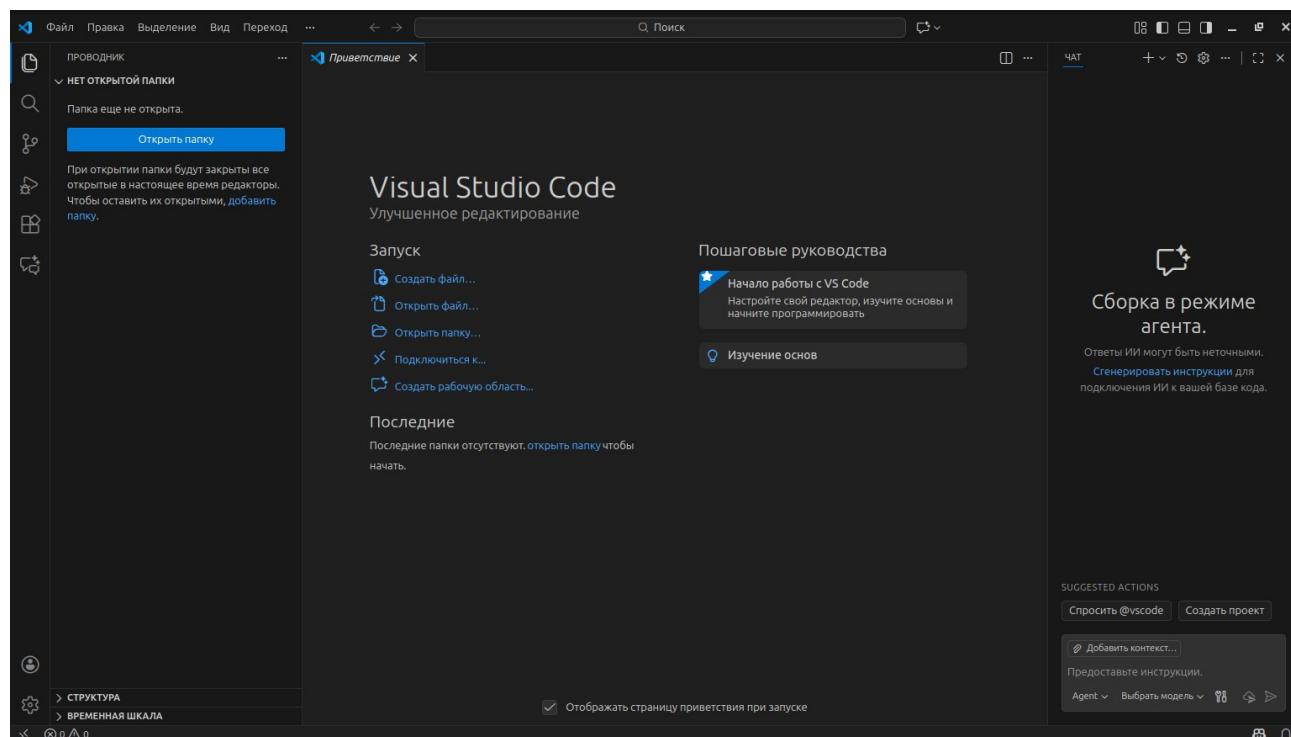
Вы только что запустили Visual Studio Code в первый раз. Давайте вместе разберемся, что перед нами появилось. Не пугайтесь обилию элементов — мы постепенно разберем каждый из них, и скоро этот интерфейс станет для вас родным!

Общий вид интерфейса

Когда программа откроется, вы увидите примерно такую картину:

text

[ЛЕВАЯ ПАНЕЛЬ] [ЦЕНТРАЛЬНАЯ ЧАСТЬ] [МОЖЕТ БЫТЬ ПРАВАЯ ПАНЕЛЬ]



Давайте пройдемся по каждому элементу по порядку.

1. Левая панель (Панель активности)

Это ваша **главная панель управления**. Смотрите на вертикальный ряд значков слева:

① Проводник (Explorer) — иконка с двумя документами

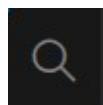


Самый важный элемент!

Здесь вы будете видеть все ваши файлы и папки

Пока он пустой — потому что мы еще не открыли папку с проектом

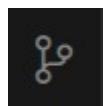
② Поиск (Search) — иконка с лупой



Для поиска текста во всех файлах проекта

Пока не понадобится

③ Система контроля версий (Source Control) — иконка с разветвлением



Для продвинутых пользователей

Пока можем игнорировать

④ Запуск и отладка (Run and Debug) — иконка с треугольником



Для тестирования программ

Пока не нужно

⑤ Расширения (Extensions) — иконка с квадратиками



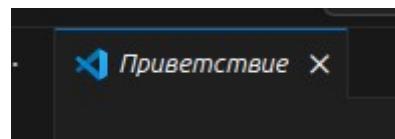
Магазин дополнений для программы

Очень полезно, но разберем позже

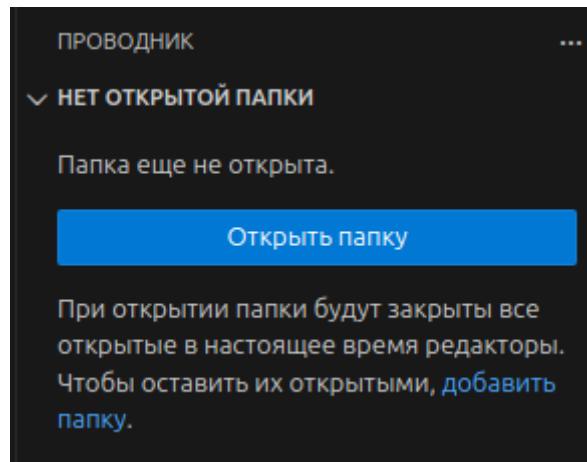
2. Центральная область — ваше рабочее пространство

Сейчас здесь вы видите **стартовую страницу**. Она состоит из нескольких разделов:

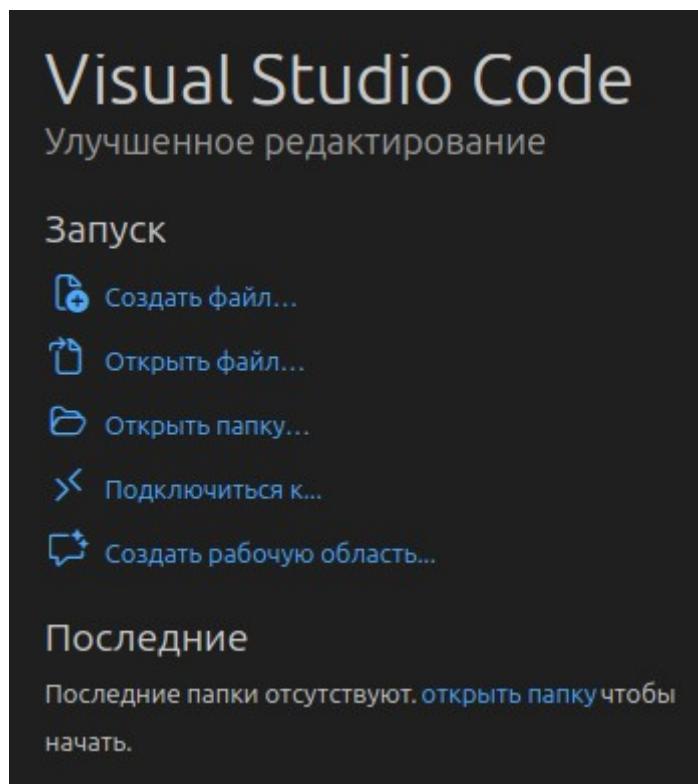
Вкладка «Welcome» (Добро пожаловать)



Крупные кнопки: «New file» (Новый файл), «Open folder» (Открыть папку)
Список недавно открытых файлов (пока пустой)

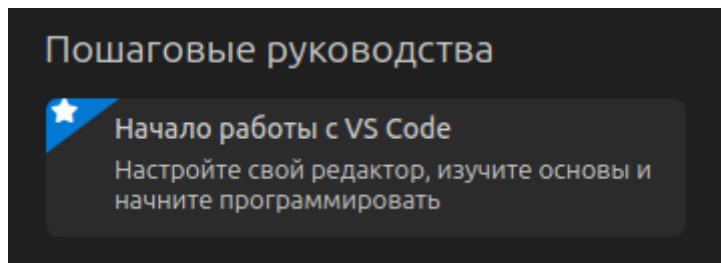


Раздел «Start» (Начать)



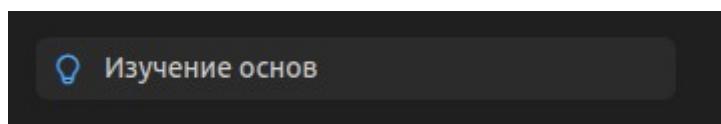
Полезные ссылки для обучения
Можно пока пропустить

Раздел «Customize» (Настройка)



Предлагает настроить программу под себя
Мы этим займемся в следующей главе

Раздел «Learn» (Обучение)



Ссылки на официальную документацию
Пока не обязательно

3. Панель состояния (в самом низу окна)

Это **информационная строка** — как приборная панель в автомобиле:

text
[СЛЕВА] [ЦЕНТР] [СПРАВА]

Слева:

Синий значок — уведомления (если есть)
Номер ветки — пока не важно

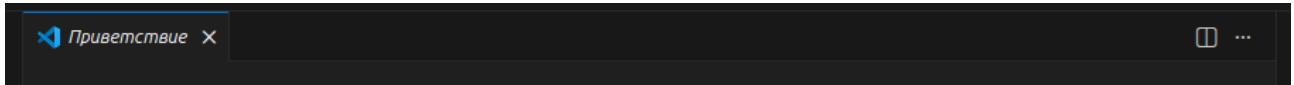
В центре:

Пока ничего нет — будет показывать ошибки при их наличии

Справа:

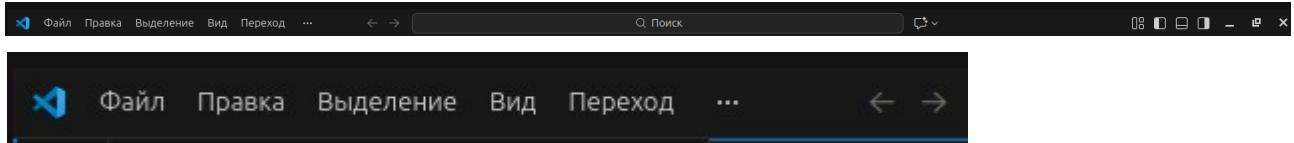
Язык — «Plain Text» (простой текст)
Формат конца строк — LF/CRLF (можно не обращать внимания)
Кодировка — UTF-8 (стандартная)
Позиция курсора — пока 1:1 (строка 1, символ 1)

4. Панель вкладок (сверху)



Сейчас здесь только одна вкладка — «**Welcome**». Когда вы откроете файлы, здесь будут появляться новые вкладки — точно так же, как в браузере!

5. Стока меню (самый верх)



Традиционное меню как во многих программах:

File (Файл) — создание, открытие, сохранение

Edit (Правка) — копирование, вставка, отмена

Selection (Выделение) — работа с выделенным текстом

View (Вид) — настройка внешнего вида

Go (Переход) — навигация

Run (Запуск) — запуск программ

Terminal (Терминал) — продвинутая функция

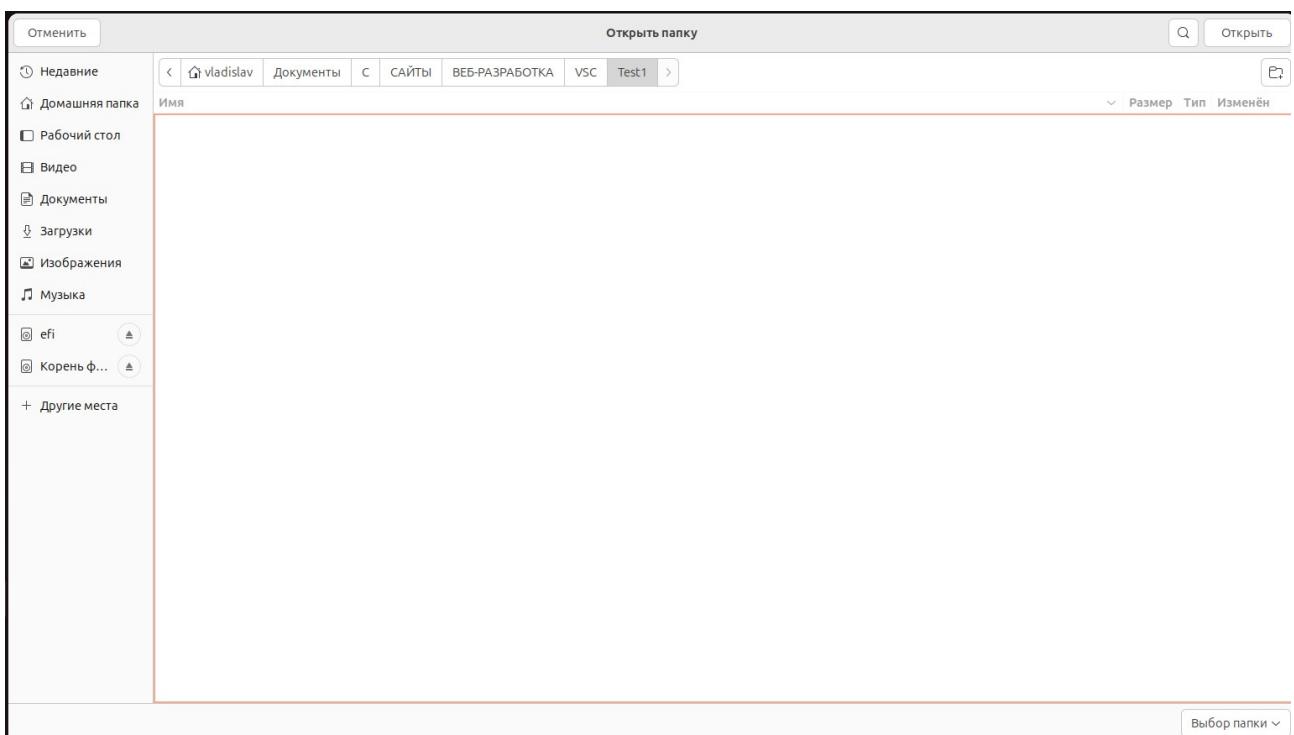
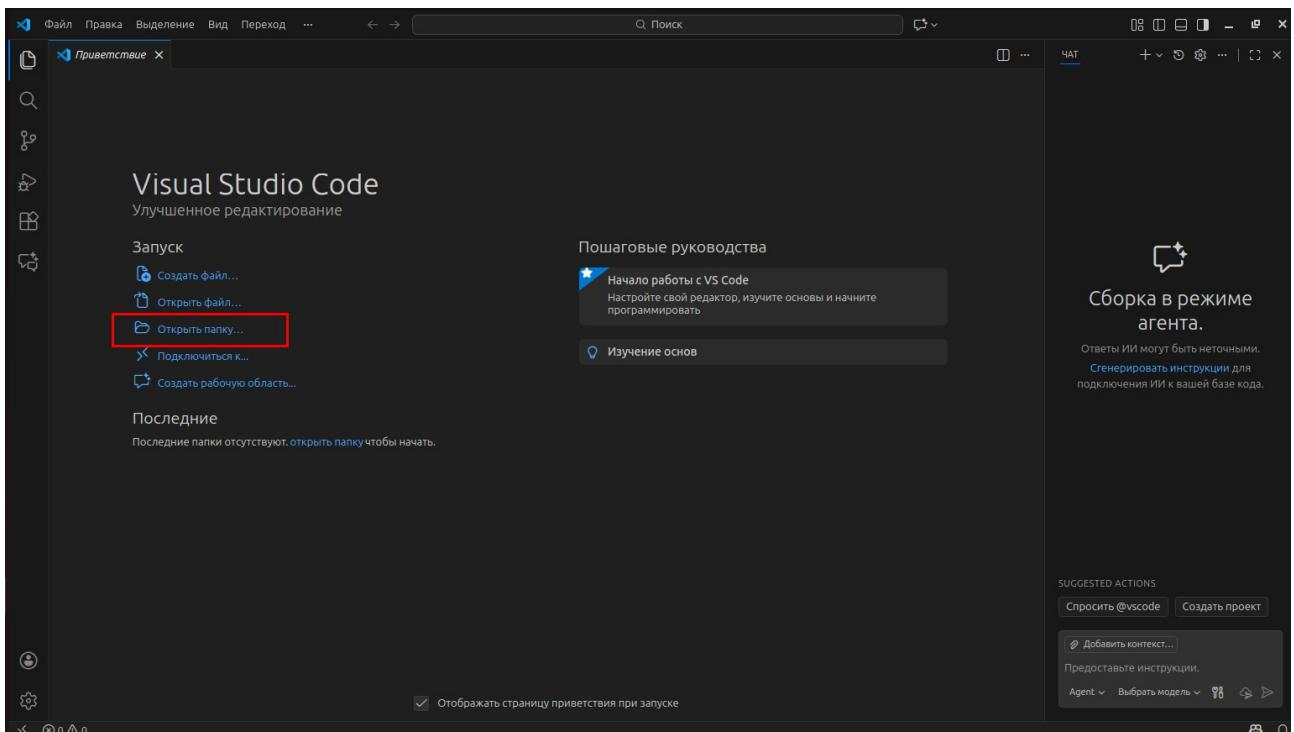
Help (Помощь) — справка

Что нам нужно сделать прямо сейчас?

Нажмите на иконку «Проводник» (два документа) в левой панели

Нажмите кнопку «Open Folder» в центральной области

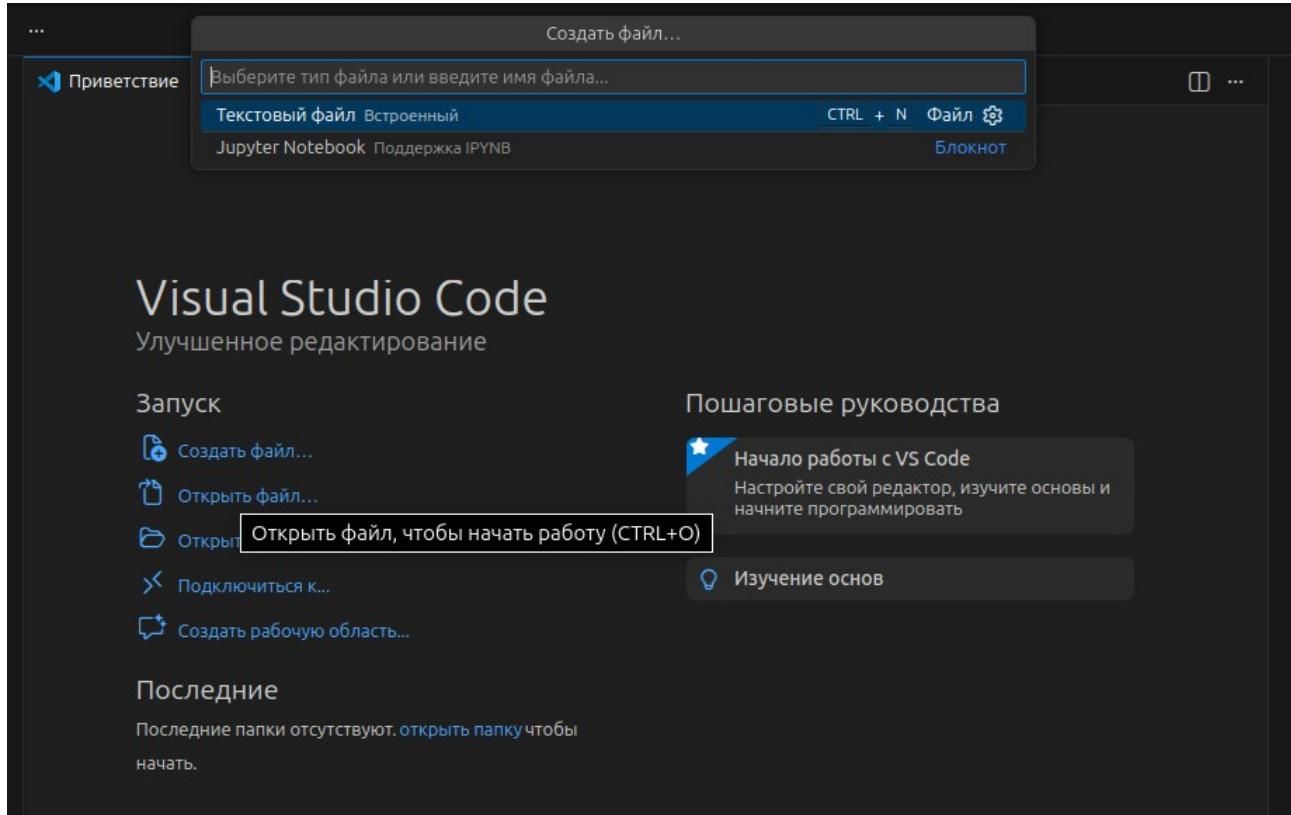
Выберите или создайте папку для вашего первого проекта



Чего НЕ нужно делать пока:

- ✗ Не нажимайте все кнопки подряд из любопытства
- ✗ Не меняйте настройки, которые не понимаете
- ✗ Не устанавливайте расширения пока
- ✗ Не пугайтесь английского интерфейса — скоро сделаем русский

Простой эксперимент для знакомства



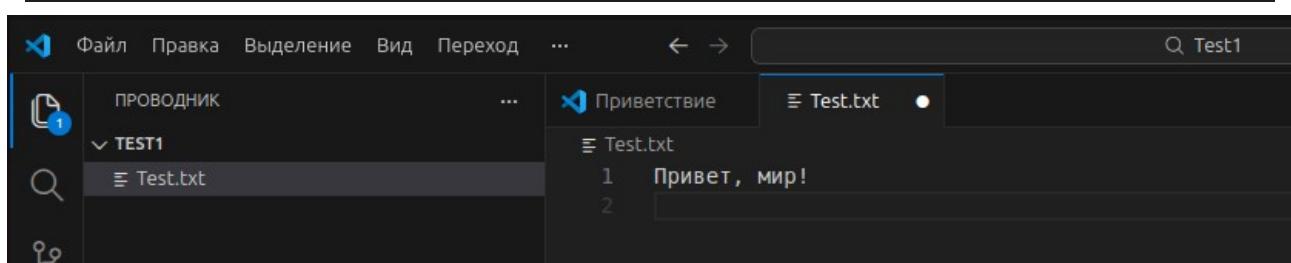
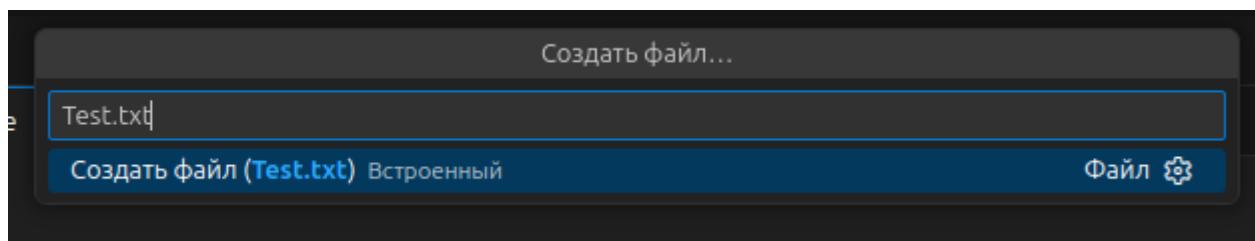
Давайте попробуем создать первый файл:

Нажмите **File** → **New File** (Файл → Новый файл)

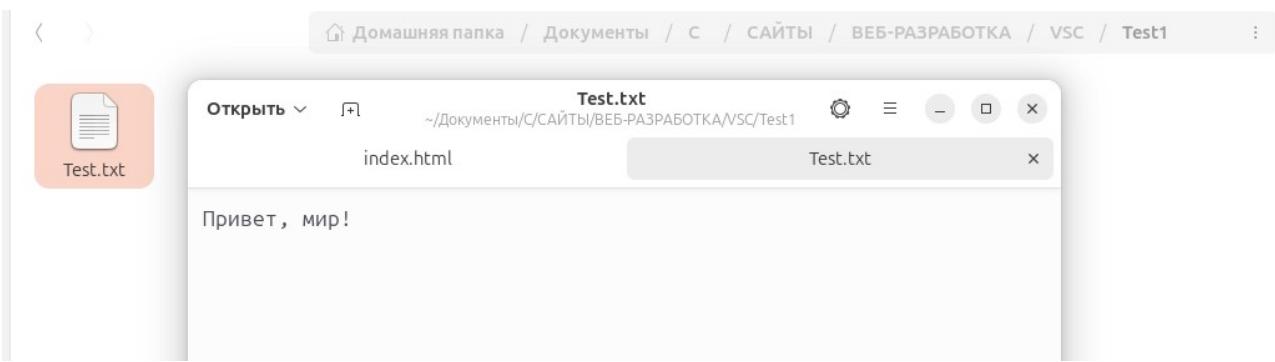
Ведите любой текст, например: «Привет, мир!»

Нажмите **File** → **Save** (Файл → Сохранить)

Сохраните файл как **test.txt**



Вы только что создали свой первый файл в VS Code!



Так выглядит этот документ у меня в проводнике и в открытом виде ↑

Если что-то выглядит иначе

Не переживайте, если у вас:

Другая цветовая схема — это нормально, мы настроим позже

Нет каких-то элементов — возможно, другая версия программы

Английский интерфейс — мы установим русский язык

Ваша задача на первое время

Пока просто запомните:

Слева — управление файлами

В центре — работа с текстом

Снизу — информация о состоянии

Сверху — меню и вкладки

Не стремитесь запомнить всё сразу! Сейчас главное — понять общую структуру. В процессе работы каждый элемент станет вам знаком и понятен.

Поздравляю с первым запуском! Вы только что сделали важный шаг в освоении современного инструмента веб-разработки!

Глава 2: Настраиваем VS Code "под себя"

Увеличиваем шрифт редактора и интерфейса

Дорогой друг!

Давайте сделаем программу удобной для ваших глаз. Это первое, что нужно настроить — чтобы работать было комфортно и не приходилось щуриться. Сейчас мы увеличим шрифт в двух местах: в **области редактирования кода** и в **общем интерфейсе** программы.

Способ 1: Через меню настроек (самый простой)

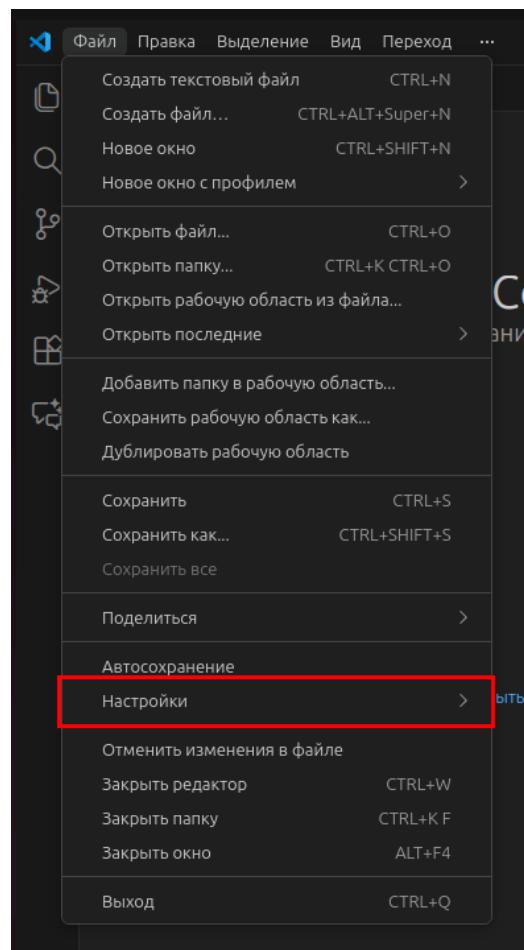
Шаг 1: Открываем настройки

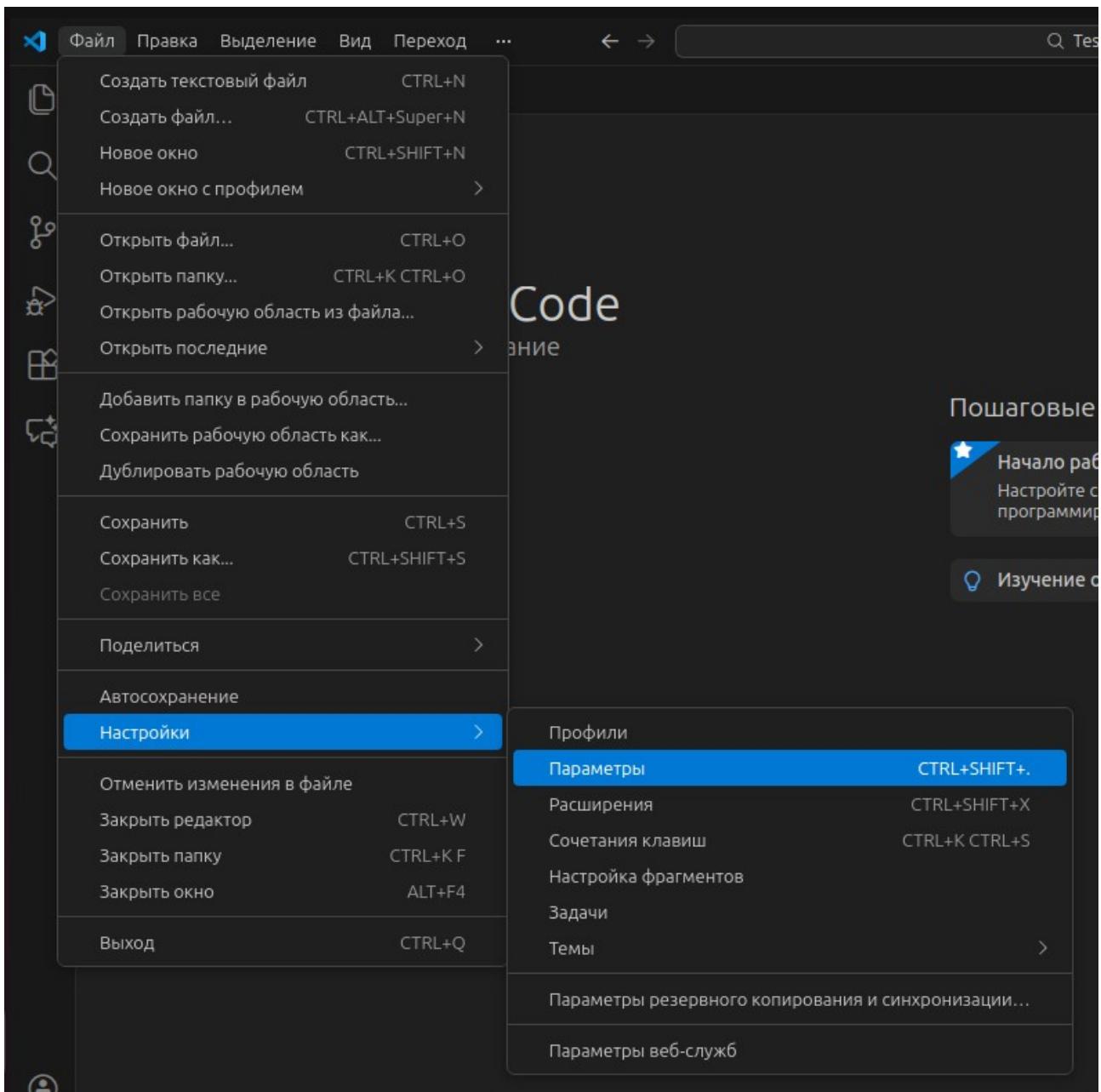
Нажмите на меню **File** (Файл) в левом верхнем углу

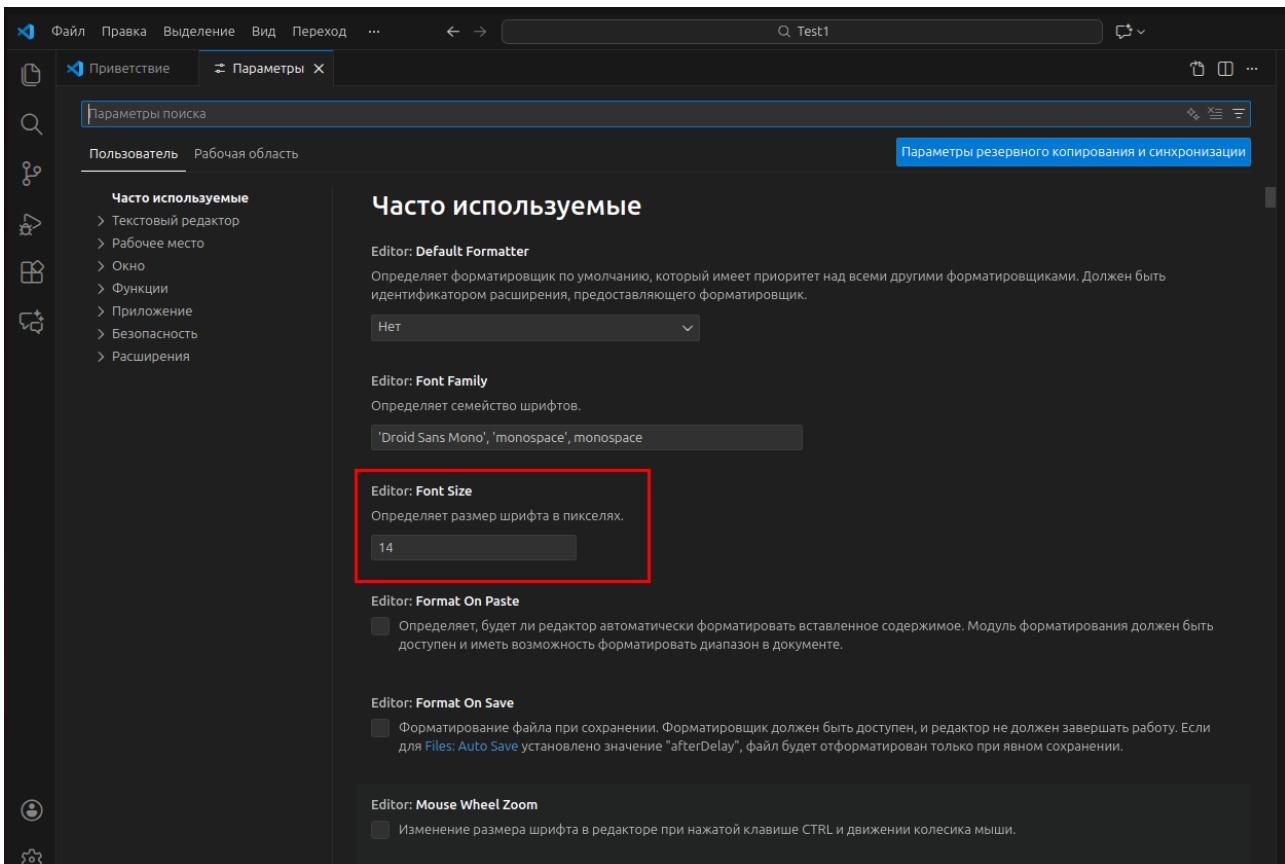
Нажмите **Settings** (Настройки)

Выберите пункт **Preferences** (Параметры)

Или просто нажмите сочетание клавиш: **Ctrl + ,** (зажмите Ctrl и нажмите запятую)







Шаг 2: Находим настройки шрифта

В открывшемся окне настроек вы увидите строку поиска вверху

Ведите в поиск: font size (размер шрифта)

Появятся две основные настройки:

text

Editor: Font Size - размер шрифта в области кода

Window: Zoom Level - масштаб всего интерфейса

Шаг 3: Настраиваем шрифт редактора

Найдите **Editor: Font Size**

Цифра по умолчанию обычно 14 или 15

Измените на 18 или 20 (можно экспериментировать!)

Просто кликните на цифру и введите новое значение

Шаг 4: Настраиваем масштаб интерфейса

Найдите **Window: Zoom Level**

По умолчанию стоит 0 (обычный размер)

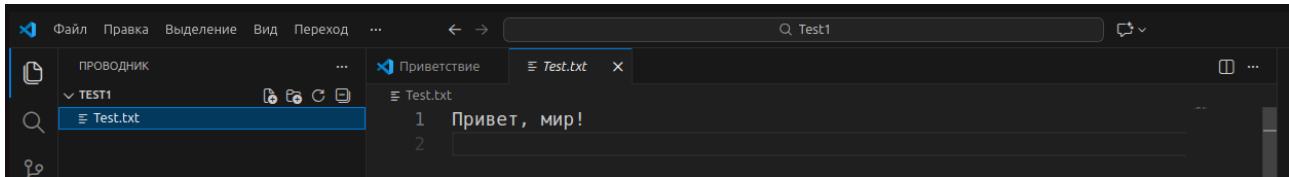
Измените на 1 или 2 — это увеличит все элементы интерфейса

Шаг 5: Проверяем результат

Закройте окно настроек (просто нажмите крестик)

Посмотрите, комфортно ли теперь читать текст

Если нужно — вернитесь в настройки и подберите идеальный размер



Способ 2: Через файл настроек (для продвинутых)

Если вы уже немного освоились, можно редактировать настройки вручную:

Нажмите **Ctrl + Shift + P** (откроется панель команд)

Ведите: **Preferences: Open Settings (JSON)**

Нажмите **Enter**

Добавьте в открывшийся файл строки:

```
json
{
  "editor.fontSize": 18,
  "window.zoomLevel": 1,
  "editor.lineHeight": 1.5
}
```

Рекомендуемые настройки для комфортной работы

Для хорошего зрения:

```
json
{
  "editor.fontSize": 16,
  "window.zoomLevel": 0,
  "editor.lineHeight": 1.3
}
```

Для слабого зрения:

```
json
{
  "editor.fontSize": 20,
  "window.zoomLevel": 1,
```

```
"editor.lineHeight": 1.6  
}
```

Для очень слабого зрения:

```
json  
{  
"editor.fontSize": 24,  
"window.zoomLevel": 2,  
"editor.lineHeight": 1.8  
}
```

Дополнительные настройки для удобства чтения

Межстрочный интервал:

Найдите настройку **Editor: Line Height**

По умолчанию 0 — значит автоматический

Установите 1.5 или 1.8 — между строками будет больше пространства

Межбуквенный интервал:

Найдите **Editor: Letter Spacing**

Установите 0.5 или 1 — буквы будут немного разряжены

Высота строки:

Найдите **Editor: Line Height**

Установите 30 или 35 — каждая строка будет выше

Что изменится после настройки

После увеличения **Editor: Font Size**:

Станет крупнее текст в центральной области редактирования

Код будет лучше читаться

Не нужно будет приближаться к монитору

После увеличения **Window: Zoom Level (уровень масштабирования)**:

Увеличатся все элементы интерфейса

Кнопки станут крупнее

Текст в меню будет больше

Панели и вкладки увеличиваются

Проверка результатов

Создайте тестовый файл чтобы оценить настройки:

Нажмите `Ctrl + N` (новый файл)

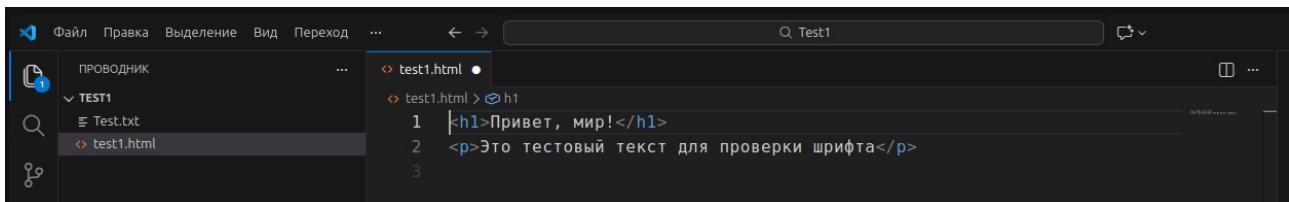
Ведите любой код, например:

html

`<h1>Привет, мир!</h1>`

`<p>Это тестовый текст для проверки шрифта</p>`

Посмотрите, комфортно ли читать



Если что-то пошло не так

Шрифт стал слишком большим:

Вернитесь в настройки

Уменьшите значения `editor.fontSize` и `window.zoomLevel`

Начните с 16 и 0 соответственно

Интерфейс съехал:

Иногда при сильном увеличении элементы могут накладываться друг на друга

Уменьшите `window.zoomLevel`

Увеличьте `editor.fontSize` отдельно

Не сохраняются настройки:

Закройте и откройте VS Code заново

Проверьте, нет ли сообщений об ошибках в правом нижнем углу

Полезные советы

Экспериментируйте! Подберите значения, подходящие именно вам

Учитывайте размер монитора — на маленьком мониторе лучше увеличивать только шрифт редактора

Сделайте несколько попыток — иногда нужно 2-3 раза подкорректировать настройки

Запомните комбинацию `Ctrl + ,` — чтобы быстро попадать в настройки

Не забудьте: После всех настроек перезапустите VS Code чтобы убедиться, что изменения применились ко всем элементам интерфейса.

Теперь ваша среда разработки стала гораздо удобнее и комфортнее для глаз! Можно приступать к изучению верстки без лишнего напряжения.

Включаем тему "Для слабого зрения"

Дорогой друг!

Visual Studio Code имеет специальные темы, созданные для людей с ослабленным зрением. Давайте настроим программу так, чтобы работать было максимально комфортно и без напряжения для глаз.

Что такое темы и зачем они нужны?

Тема — это набор цветов для разных элементов программы. Представьте, что это как выбрать обои для комнаты: одни люди предпочитают светлые тона, другие — темные, а для слабого зрения нужны специальные высококонтрастные варианты.

Способ 1: Быстрый выбор темы через команду

Шаг 1: Открываем палитру команд

Нажмите сочетание клавиш **Ctrl + Shift + P**

Появится строка ввода в верхней части экрана

Шаг 2: Ищем тему для слабого зрения

Ведите в строку: **Preferences: Color Theme**

Нажмите **Enter**

Откроется список всех доступных тем

Шаг 3: Выбираем подходящую тему

Прокрутите список и найдите темы с пометкой **High Contrast** (Высокая контрастность):

text

- [] Dark (Visual Studio)
- [] Dark+ (default dark)
- [] Light (Visual Studio)
- [] Light+ (default light)
- [] High Contrast (Высокая контрастность) ★
- [] High Contrast Light (Светлая высокая контрастность) ★

Рекомендуемые варианты:

High Contrast — темная тема с максимальным контрастом

High Contrast Light — светлая тема с максимальным контрастом

Выберите нужную тему с помощью стрелок клавиатуры и нажмите Enter.

Способ 2: Поиск дополнительных тем

Если стандартных тем недостаточно:

Нажмите **Ctrl + Shift + P**

Ведите: **Preferences: Color Theme**

В самом верху списка найдите **Install Additional Color Themes**

Нажмите **Enter** — откроется магазин расширений

В поиске введите: **high contrast** или **low vision**

Найдите и установите подходящие темы

Популярные темы для слабого зрения

Встроенные в VS Code:

High Contrast — черный фон, яркие цвета

High Contrast Light — белый фон, темные цвета

Из магазина расширений:

High Contrast++ — улучшенная версия

Large Type — с увеличенными шрифтами

Color Blind — для людей с дальтонизмом

Дополнительные настройки для комфорта

После выбора темы настройте дополнительные параметры:

1. Увеличиваем контрастность выделения:

Откройте настройки (**Ctrl + ,**)

Найдите: **workbench.colorCustomizations**

Нажмите **Edit in settings.json**

Добавьте код:

```
json
{
  "workbench.colorCustomizations": {
    "editor.selectionBackground": "#ff0000",
```

```
"editor.selectionHighlightBackground": "#ffff00"  
}  
}  
}
```

2. Увеличиваем контрастность текущей строки:

```
json  
{  
"workbench.colorCustomizations": {  
"editor.lineHighlightBackground": "#aaffaa",  
"editor.lineHighlightBorder": "#00ff00"  
}  
}
```

3. Делаем скобки более заметными:

```
json  
{  
"workbench.colorCustomizations": {  
"editorBracketMatch.background": "#ff0000",  
"editorBracketMatch.border": "#ffff00"  
}  
}
```

Настройки для конкретных элементов

Чтобы сделать более контрастными отдельные элементы:

Откройте настройки (Ctrl + ,)
Найдите workbench.colorCustomizations
Нажмите Edit in settings.json
Используйте этот шаблон:

```
json  
{  
"workbench.colorTheme": "High Contrast",  
"workbench.colorCustomizations": {  
"editor.background": "#000000",  
"editor.foreground": "#FFFFFF",  
"editorCursor.foreground": "#FFFF00",  
"editorLineNumber.foreground": "#00FF00"  
}  
}
```

Проверка результата

После применения темы создайте тестовый файл:

```
html
<!DOCTYPE html>
<html>
  <head>
    <title>Тест контрастности</title>
  </head>
  <body>
    <h1>Заголовок первого уровня</h1>
    <p>Обычный текст для проверки читаемости</p>
    <div class="container">
      <span>Текст внутри контейнера</span>
    </div>
  </body>
</html>
```

Проверьте:

- Хорошо ли видны все элементы кода?
- Достаточно ли контрастны разные типы текста?
- Не режут ли глаза цвета?
- Удобно ли читать длительное время?

Решение возможных проблем

Цвета слишком яркие:

Попробуйте тему **High Contrast Light**

Уменьшите яркость монитора

Добавьте в настройки менее насыщенные цвета

Не все элементы хорошо видны:

Вернитесь в настройки цветов

Найдите проблемный элемент через поиск

Измените его цвет на более контрастный

Тема не применяется:

Перезапустите VS Code
Проверьте, нет ли ошибок в файле настроек
Попробуйте другую тему

Дополнительные рекомендации

Для темной темы High Contrast:

Установите "editor.fontSize": 18
Установите "editor.lineHeight": 1.8
Включите "editor.bracketPairColorization": true

Для светлой темы High Contrast Light:

Установите "editor.fontSize": 16
Установите "editor.lineHeight": 1.6
Добавьте "workbench.colorCustomizations" как выше

Полезные комбинации настроек

Максимальная читаемость:

```
json
{
  "workbench.colorTheme": "High Contrast",
  "editor.fontSize": 20,
  "editor.lineHeight": 2.0,
  "editor.fontWeight": "bold",
  "window.zoomLevel": 1
}
```

Комфортная работа:

```
json
{
  "workbench.colorTheme": "High Contrast Light",
  "editor.fontSize": 18,
  "editor.lineHeight": 1.8,
  "editor.cursorBlinking": "solid"
}
```

Не забудьте сохранить настройки и перезапустить VS Code для применения всех изменений!

Теперь ваша среда разработки стала намного удобнее и безопаснее для глаз. Вы можете работать над версткой сайтов без напряжения и усталости!

Настраиваем масштаб интерфейса

Дорогой друг!

Теперь давайте настроим общий масштаб интерфейса VS Code. Это нужно, чтобы все элементы программы — меню, панели, кнопки — были удобного размера и вам не приходилось щуриться или приближаться к монитору.

Что такое масштаб интерфейса и зачем он нужен?

Масштаб интерфейса — это общее увеличение всех элементов программы. Если сравнить с книгой: увеличение шрифта — это как увеличить только текст, а масштаб интерфейса — как увеличить всю книгу вместе с обложкой, полями и иллюстрациями.

Способ 1: Быстрое изменение масштаба (самый простой способ)

Используем сочетания клавиш:

Увеличить масштаб: `Ctrl + =` (зажмите `Ctrl` и нажмите знак равенства)

Уменьшить масштаб: `Ctrl + -` (зажмите `Ctrl` и нажмите знак минуса)

Сбросить масштаб: `Ctrl + 0` (зажмите `Ctrl` и нажмите ноль)

Нажмите комбинации несколько раз, пока не достигнете комфортного размера.

Способ 2: Точная настройка через параметры

Шаг 1: Открываем настройки

Нажмите `Ctrl + ,` (зажмите `Ctrl` и нажмите запятую)

Или через меню: **File → Preferences → Settings**

Шаг 2: Находим настройку масштаба

В строке поиска вверху введите: `zoom level`

Найдите параметр `Window: Zoom Level`

Шаг 3: Настраиваем масштаб

По умолчанию значение 0 (100%)

Измените значение на:

1 = 125% увеличения

2 = 150% увеличения

3 = 175% увеличения

-1 = 75% уменьшения

Начните с значения 1 или 2 и проверьте, комфортно ли вам

Способ 3: Настройка через файл настроек

Для точной кастомизации:

Нажмите Ctrl + Shift + P

Ведите: Preferences: Open Settings (JSON)

Нажмите Enter

Добавьте строку:

```
json
{
  "window.zoomLevel": 1,
  "editor.fontSize": 18,
  "editor.lineHeight": 1.5
}
```

Рекомендуемые значения масштаба

Для мониторов 21-24 дюйма:

```
json
{
  "window.zoomLevel": 1,
  "editor.fontSize": 16
}
```

Для мониторов 15-20 дюймов:

```
json
{
  "window.zoomLevel": 2,
  "editor.fontSize": 18
}
```

Для слабого зрения:

```
json
{
  "window.zoomLevel": 2,
  "editor.fontSize": 20,
  "editor.lineHeight": 1.8
}
```

Что именно увеличивается при масштабировании

При изменении `Window: Zoom Level` увеличиваются:

- Все панели инструментов
- Меню и подменю
- Текст в боковых панелях
- Иконки и кнопки
- Вкладки файлов
- Статусная строка
- Панель активности слева

Дополнительные настройки для комфорта

Увеличиваем только интерфейс (без кода):

```
json
{
  "window.zoomLevel": 2,
  "editor.fontSize": 14
}
```

Увеличиваем всё вместе:

```
json
{
  "window.zoomLevel": 1,
  "editor.fontSize": 18,
  "editor.lineHeight": 1.6
}
```

Для очень слабого зрения:

```
json
{
  "window.zoomLevel": 3,
```

```
"editor.fontSize": 22,  
"editor.lineHeight": 2.0,  
"editor.fontWeight": "bold"  
}
```

Пошаговая инструкция настройки

Откройте VS Code

Нажмите `Ctrl +`, для открытия настроек

В поиске введите `zoom`

Найдите `Window: Zoom Level`

Установите значение `1`

Закройте настройки

Оцените результат

Если нужно — повторите с значением `2`

Проверка результата

После настройки проверьте:

Левая панель — иконки и текст стали крупнее?

Верхнее меню — пункты меню читаются легко?

Вкладки файлов — названия файлов хорошо видны?

Статусная строка — текст внизу окна читается без напряжения?

Решение возможных проблем

Элементы накладываются друг на друга:

Уменьшите `window.zoomLevel` на `1`

Увеличьте `editor.fontSize` отдельно

Текст слишком крупный:

Установите `window.zoomLevel: 0`

Настройте только `editor.fontSize`

Не применяются изменения:

Перезапустите VS Code

Проверьте правильность написания параметров

Полезные советы

Экспериментируйте с разными значениями

Учитывайте разрешение монитора — на 4К мониторах нужно большее увеличение

Сделайте перерыв после настройки — вернитесь через 15 минут и оцените, комфортно ли глазам

Используйте горячие клавиши для быстрой регулировки в процессе работы

Итоговые рекомендации

Для начала установите:

```
json
{
  "window.zoomLevel": 1,
  "editor.fontSize": 16,
  "editor.lineHeight": 1.5
}
```

После тестирования можно настроить точнее:

Если мелко — увеличьте `zoomLevel` до 2

Если размыто — уменьшите `zoomLevel` и увеличьте `editor.fontSize`

Если устают глаза — увеличьте `lineHeight`

Теперь ваш VS Code стал визуально комфортным! Все элементы интерфейса имеют оптимальный размер, и вы можете сосредоточиться на изучении верстки без лишнего напряжения глаз.

Сохраняем настройки

Дорогой друг!

Теперь, когда мы настроили VS Code под ваши потребности, важно правильно сохранить все изменения. Давайте разберем, как сохранить настройки, чтобы они не сбились и работали при каждом запуске программы.

Как работают настройки в VS Code

Настройки сохраняются автоматически в специальном файле. Есть два способа их редактирования:

Через графический интерфейс (простой способ)

Через JSON-файл (продвинутый способ)

Способ 1: Сохранение через графический интерфейс

Шаг 1: Открываем настройки

Нажмите **Ctrl + ,** (зажмите Ctrl и нажмите запятую)

Или через меню: **File → Preferences → Settings**

Шаг 2: Вносим изменения

Все изменения сохраняются **автоматически**

После изменения параметра просто закройте окно настроек

Не нужно нажимать кнопку "Save" — её нет!

Шаг 3: Проверяем сохранение

Измените любой параметр (например, размер шрифта)

Закройте окно настроек

Перезапустите VS Code

Проверьте, сохранились ли изменения

Способ 2: Ручное редактирование через JSON-файл

Шаг 1: Открываем файл настроек

Нажмите **Ctrl + Shift + P**

Ведите: Preferences: Open Settings (JSON)
Нажмите Enter

Шаг 2: Редактируем настройки

Откроется файл settings.json с таким содержимым:

```
json
{
    "window.zoomLevel": 1,
    "editor.fontSize": 18,
    "editor.lineHeight": 1.5,
    "workbench.colorTheme": "High Contrast"
}
```

Шаг 3: Сохраняем изменения

Нажмите Ctrl + S (Сохранить)
Или через меню: File → Save

Где физически хранятся настройки

Ваши настройки сохраняются в файле:

```
text
C:\Users\ВашеИмя\AppData\Roaming\Code\User\settings.json
```

Чтобы найти этот файл:

Нажмите Ctrl + Shift + P
Ведите: Preferences: Open Settings (JSON)
Нажмите правой кнопкой на вкладке файла
Выберите Copy Path (Скопировать путь)

Рекомендуемые настройки для сохранения

Базовый набор для комфортной работы:

```
json
{
    // Масштаб интерфейса
    "window.zoomLevel": 1,
```

```
// Шрифт редактора
"editor.fontSize": 18,
"editor.lineHeight": 1.6,
"editor.fontFamily": "Consolas, 'Courier New', monospace",

// Тема для слабого зрения
"workbench.colorTheme": "High Contrast",

// Автосохранение
"files.autoSave": "afterDelay",
"files.autoSaveDelay": 1000,

// Перенос строк
"editor.wordWrap": "on",

// Увеличение курсора
"editor.cursorBlinking": "solid",
"editor.cursorWidth": 2
}
```

Создание резервной копии настроек

Чтобы не потерять настройки:

Откройте файл настроек (Ctrl + Shift + P → Preferences: Open Settings (JSON))

Нажмите Ctrl + A (Выделить всё)

Нажмите Ctrl + C (Скопировать)

Откройте Блокнот

Нажмите Ctrl + V (Вставить)

Сохраните файл как VS_Code_Backup.txt на Рабочем столе

Восстановление настроек из резервной копии

Если настройки сбились:

Откройте файл VS_Code_Backup.txt

Нажмите Ctrl + A → Ctrl + C

В VS Code откройте настройки JSON

Нажмите Ctrl + A → Ctrl + V

Сохраните (Ctrl + S)

Автоматическое сохранение файлов

Настройте автосохранение чтобы не потерять работу:

```
json
{
  "files.autoSave": "afterDelay",
  "files.autoSaveDelay": 1000,
  "files.autoSaveWhenNoErrors": true
}
```

Проверка сохраненных настроек

Создайте тестовый файл для проверки:

Нажмите **Ctrl + N** (новый файл)

Ведите код:

```
html
<!DOCTYPE html>
<html>
<head>
  <title>Проверка настроек</title>
</head>
<body>
  <h1>Тест размера шрифта</h1>
  <p>Проверяем читаемость текста</p>
</body>
</html>
```

Проверьте:

- Размер шрифта комфортный?
- Цветовая схема не режет глаза?
- Между строками достаточное расстояние?
- Курсор хорошо виден?

Решение проблем с сохранением

Настройки не сохраняются:

Проверьте, нет ли ошибок в JSON-файле

Убедитесь, что вы сохранили файл (Ctrl + S)
Перезапустите VS Code

Сброс настроек к defaults:

Нажмите Ctrl + Shift + P
Ведите: Preferences: Open Settings (JSON)
Удалите всё содержимое файла
Сохраните и перезапустите VS Code

Ошибки в файле настроек:

Проверьте запятые (должны быть после каждого элемента кроме последнего)
Проверьте кавычки (должны быть двойные)
Убедитесь, что нет лишних символов

Полезные команды для работы с настройками

Preferences: Open Settings (UI) — графический интерфейс
Preferences: Open Settings (JSON) — редактирование кода
Preferences: Open Default Settings (JSON) — настройки по умолчанию
Preferences: Open Workspace Settings (JSON) — настройки проекта

Итоговый чек-лист

После настройки проверьте:
Размер шрифта редактора
Масштаб интерфейса
Цветовая тема
Межстрочный интервал
Автосохранение включено
Сделана резервная копия
Все настройки работают после перезапуска

Поздравляю! Теперь ваш VS Code полностью настроен под ваши потребности и сохранит все изменения. Вы можете спокойно работать над версткой сайтов, не беспокоясь о настройках программы.

Глава 3: Знакомство с интерфейсом

Панель активности (слева)

Дорогой друг!

Давайте подробно разберем самую важную панель в VS Code — **Панель активности**. Это ваша главная панель управления, которая находится слева. Здесь собраны все основные инструменты для работы над сайтом.

Что такое Панель активности?

Это вертикальная панель с иконками слева от основного окна.

Представьте, что это ваш **главный пульт управления** — как панель приборов в автомобиле, где все важные кнопки под рукой.

Основные иконки Панели активности

Сверху вниз расположены 5 основных иконок:

text

[1] Проводник (Explorer)



[2] Поиск (Search)

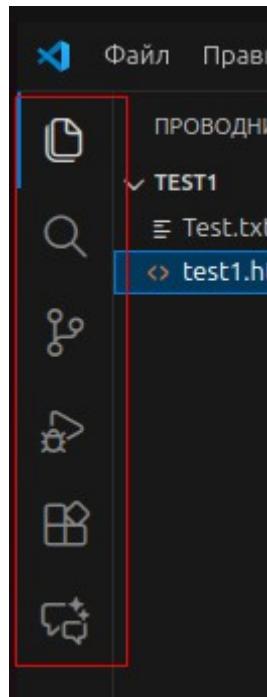


[3] Система контроля версий (Git) ⚙

[4] Запуск и отладка (Run) ▶



[5] Расширения (Extensions)



Давайте разберем каждую иконку подробно.

1. Проводник (Explorer) — иконка с двумя листами бумаги

Это ваш главный инструмент! Здесь вы видите все файлы и папки вашего проекта.

Что вы видите в Проводнике:

text
OPEN EDITORS (Открытые редакторы)
- index.html ●
- style.css ●

МОЙ ПЕРВЫЙ САЙТ (название вашей папки)
images
index.html
style.css

Как работать с Проводником:

Открыть файл: Кликните дважды по названию файла
Создать файл: Нажмите на иконку "Новый файл" рядом с названием папки
Создать папку: Нажмите на иконку "Новая папка"

Переименовать: Правой кнопкой мыши → "Rename"

Удалить: Правой кнопкой мыши → "Delete"

Практическое задание:

Нажмите на иконку "Новый файл"

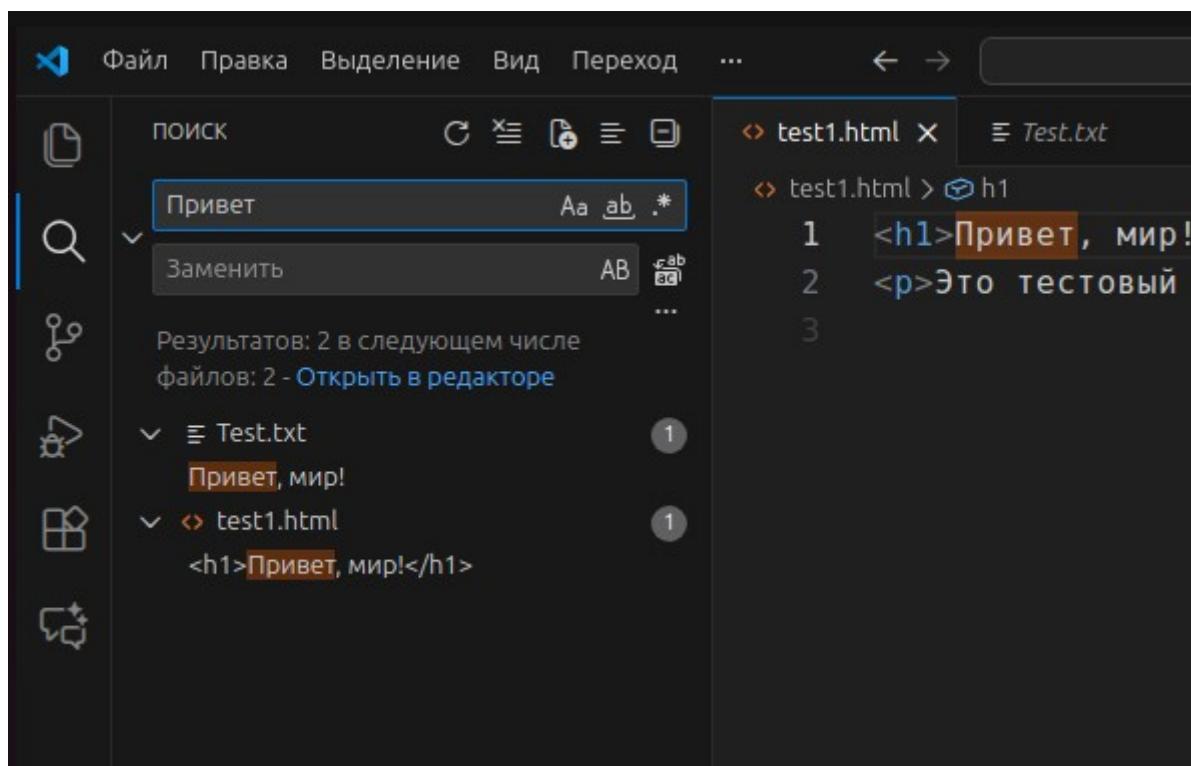
Назовите файл `test.html`

Нажмите Enter

Ура! Вы создали свой первый файл в VS Code

2. Поиск (Search) — иконка с лупой 🔎

Это ваш мощный инструмент для поиска. С его помощью можно найти любое слово или фразу во всех файлах проекта.



Как использовать поиск:

`text`

Поиск: [поле для ввода слова]

Заменить: [поле для замены]

Файлы для включения: [где искать]

Файлы для исключения: [где не искать]

Пример использования:

Хотите найти все упоминания слова "header" в проекте?
Ведите "header" в поле поиска → нажмите Enter
VS Code покажет все файлы, где встречается это слово

Практическое задание:

Откройте файл `test.html`
Напишите: `<h1>Привет мир</h1>`
Сохраните файл (Ctrl + S)
Откройте поиск (иконка с лупой)
Ведите "Привет"
Увидите, что файл `test.html` найден!

3. Система контроля версий (Git) — иконка с разветвлениями ◆

Это инструмент для продвинутых пользователей. Пока мы его использовать не будем, но знать о его существовании полезно.

Для чего нужен Git:

Сохранение истории изменений
Возможность отката к предыдущим версиям
Совместная работа над проектом
Пока просто знайте, что эта иконка здесь есть. Вернемся к ней позже!

4. Запуск и отладка (Run) — иконка с треугольником ▷

Инструмент для тестирования и отладки кода.

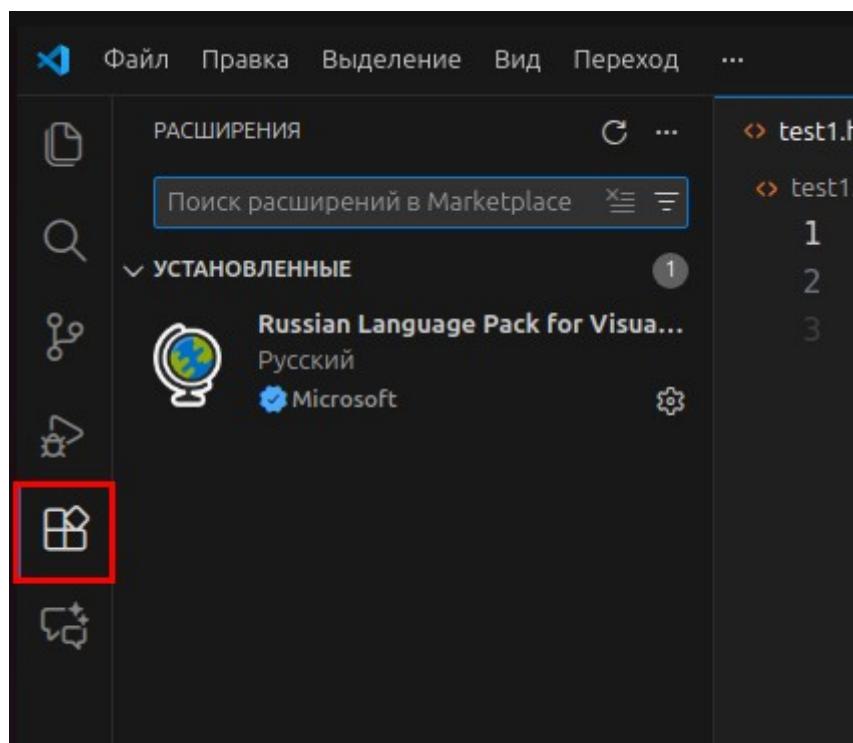
Что можно делать:

Запускать программы
Находить ошибки в коде
Тестируировать работу сайта

Практическое использование:

Когда мы установим расширение "Live Server", именно здесь будет кнопка для запуска сайта в браузере.

5. Расширения (Extensions) — иконка с квадратиками



Это магазин дополнений для VS Code. Самая полезная иконка после Проводника!

Что можно найти в Расширениях:

Русский язык интерфейса
Инструменты для HTML и CSS
Темы оформления
И многое другое

Как пользоваться:

Нажмите на иконку Расширений
В строке поиска введите нужное расширение
Нажмите "Install" (Установить)

Как скрыть/показать Панель активности

Иногда нужно больше места для кода. Вы можете временно скрыть панель:

Скрыть панель: Ctrl + B

Показать панель: снова Ctrl + B

Практическое задание для закрепления

Давайте создадим структуру настоящего проекта:

Откройте Проводник (первая иконка)

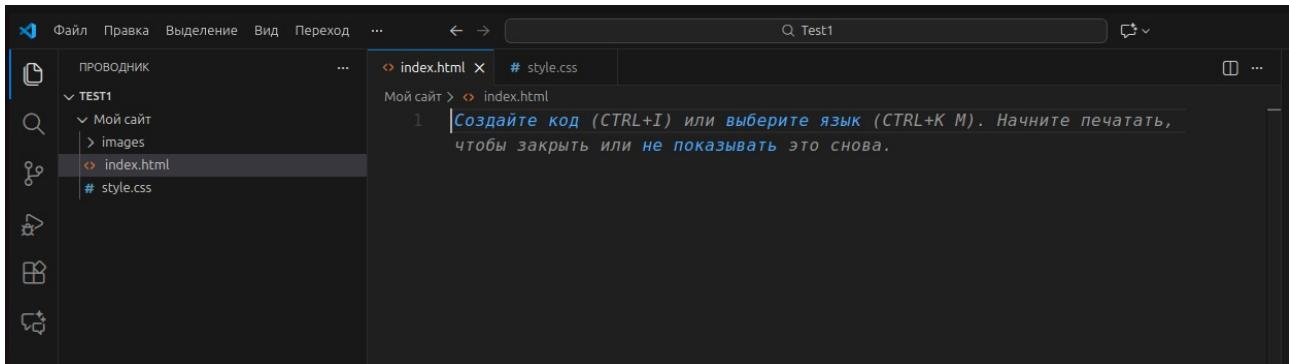
Создайте новую папку "Мой сайт"

Внутри папки создайте:

Файл `index.html`

Файл `style.css`

Папку `images`



Откройте файл `index.html` двойным кликом

Начните писать код!

Полезные советы

Запомните сочетание клавиш `Ctrl + B` для быстрого скрытия/показа панели

Используйте Проводник для организации файлов — это избавит от хаоса

Не бойтесь ошибаться — любую папку или файл можно переименовать или удалить

Чаще используйте поиск — это сэкономит много времени

Что делать, если что-то пошло не так

Панель активности пропала:

Нажмите `Ctrl + B` — она вернется

Или через меню: View → Appearance → Show Activity Bar

Не вижу нужной иконки:

Возможно, вы установили другую тему оформления
Все основные иконки всегда на своем месте

Не получается создать файл:

Убедитесь, что вы выделили папку, а не файл
Попробуйте правой кнопкой мыши → New File

**Теперь вы знакомы с главной панелью управления VS Code!
Постепенно вы будете использовать все эти инструменты все
более уверенно. Главное — не бойтесь экспериментировать и
нажимать на кнопки!**

Проводник файлов

Дорогой друг!

Давайте подробно изучим **Проводник файлов** — самый важный инструмент в VS Code. Это ваша главная панель для управления всеми файлами и папками будущего сайта.

Что такое Проводник файлов?

Это область в левой части экрана, где отображается **структура вашего проекта**. Представьте, что это ваш цифровой письменный стол, где все файлы разложены по папкам и всегда под рукой.

Как открыть Проводник файлов

Способ 1: Нажмите на первую иконку в Панели активности (два листа бумаги) 

Способ 2: Используйте горячие клавиши **Ctrl + Shift + E**

Способ 3: Через меню: **View → Explorer**

Из чего состоит Проводник

Когда вы откроете папку с проектом, вы увидите:

text

OPEN EDITORS (Открытые редакторы)

- index.html ●
- style.css ●

МОЙ ПЕРВЫЙ САЙТ (название вашей папки)

images

index.html

style.css

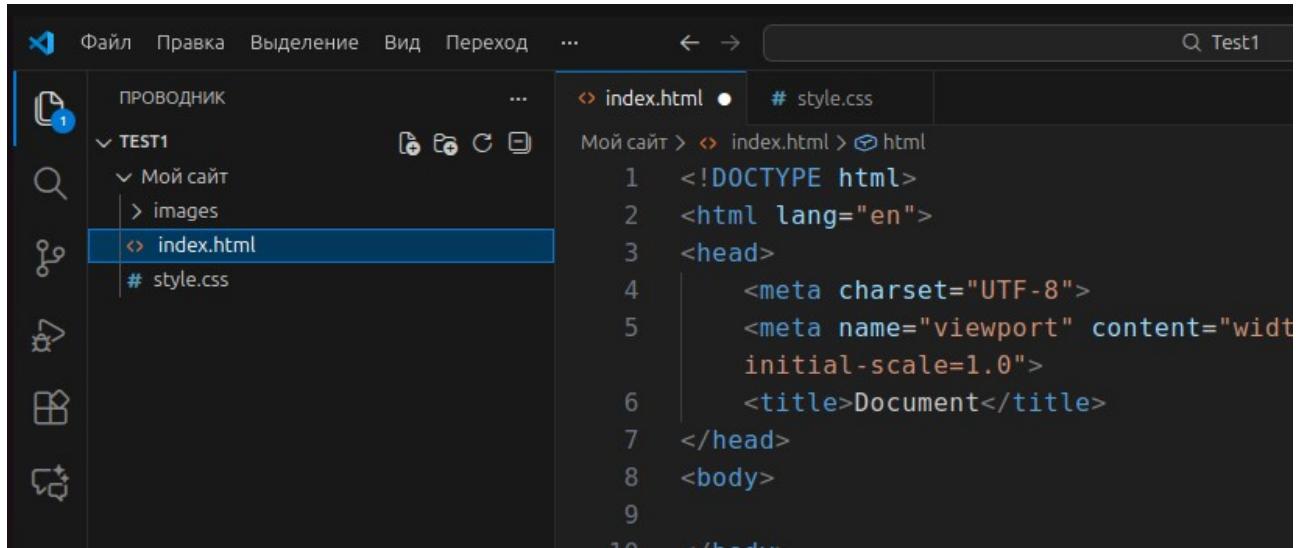
css

js

Давайте разберем каждую секцию:

1. OPEN EDITORS (Открытые редакторы)

Здесь отображаются все файлы, которые у вас открыты в данный момент.



The screenshot shows the 'OPEN EDITORS' panel of a code editor. On the left is a sidebar with icons for File Explorer, Search, and other tools. Below it is a tree view of a project named 'TEST1'. Under 'TEST1', there is a folder 'Мой сайт' containing 'images' and two files: 'index.html' and '# style.css'. The file 'index.html' is currently selected and highlighted with a blue bar at the top of its preview area. The preview area shows the HTML code for 'index.html':

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9
10 </body>
```

Точка (●) означает, что в файле есть несохраненные изменения

Крестик (✗) закрывает файл

Файлы расположены в порядке их открытия

Практическое использование:

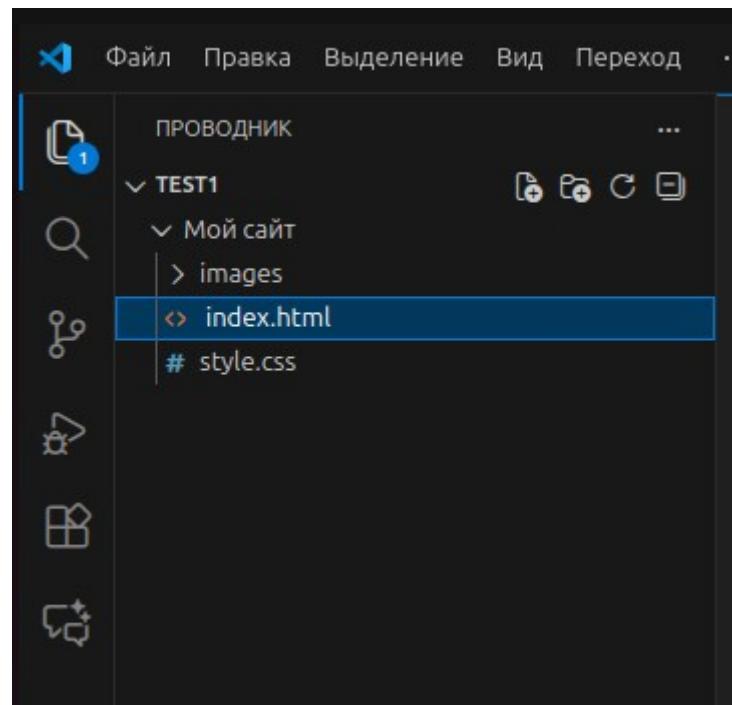
Быстро переключайтесь между открытymi файлами

Видите, какие файлы требуют сохранения

Закрывайте ненужные файлы одним кликом

2. Структура папки проекта

Это основная область, где вы видите все файлы и папки.



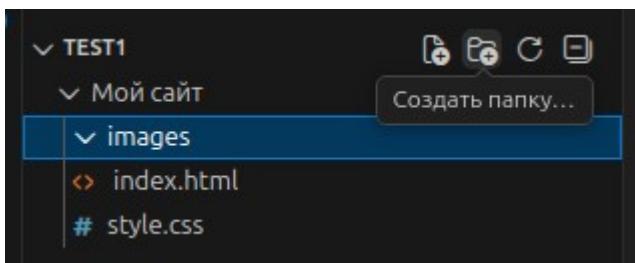
Обозначения:

- папка
- файл
- свернутая папка (можно развернуть)
- развернутая папка

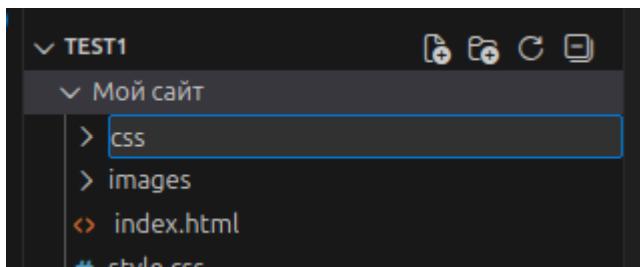
Как создать новую папку

Способ 1: Через иконку

Наведите курсор на название вашего проекта
Нажмите на иконку "**Новая папка**" (папка с плюсом)



Введите название папки (например, `images`)

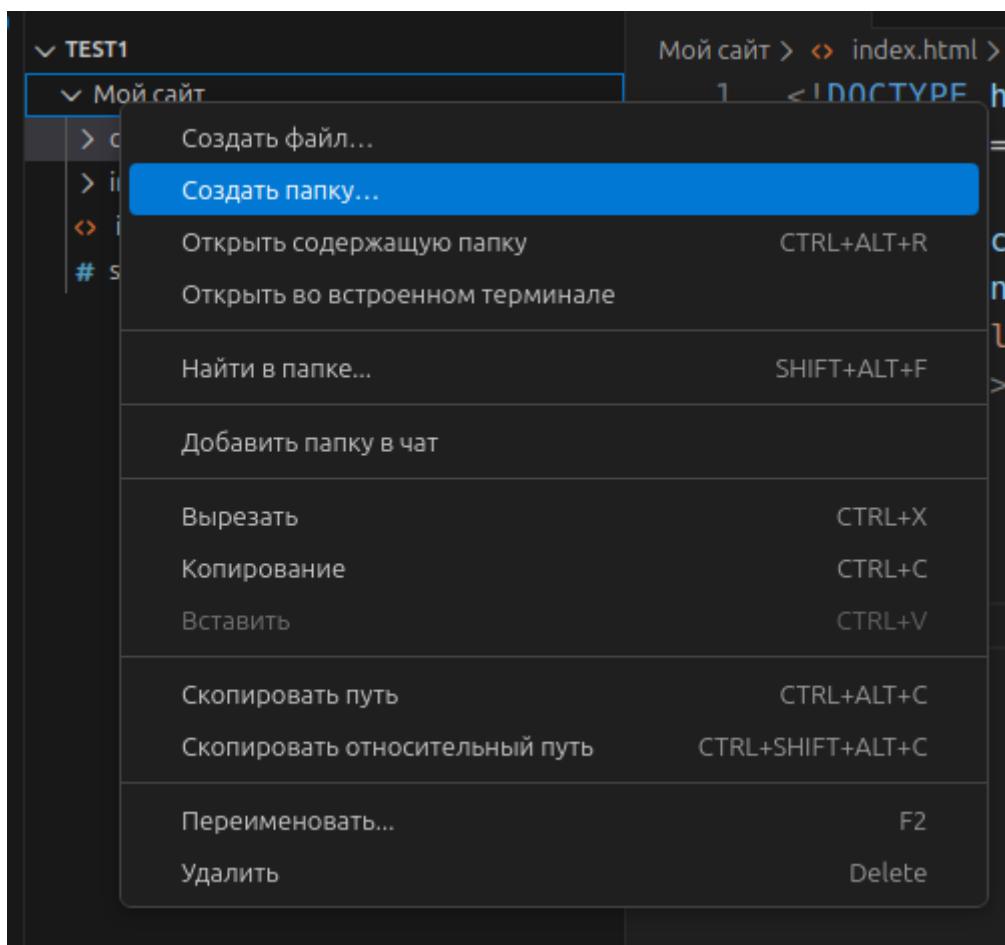


Нажмите Enter

Способ 2: Через правую кнопку мыши

Правой кнопкой мыши на названии проекта

Выберите **New Folder**



Введите название

Нажмите Enter

Как создать новый файл

Способ 1: Через иконку

Наведите курсор на папку, где хотите создать файл
Нажмите на иконку "**Новый файл**" (лист бумаги с плюсом)
Ведите название файла с расширением (например, index.html)
Нажмите Enter

Способ 2: Через правую кнопку мыши

Правой кнопкой мыши на папке
Выберите **New File**
Ведите название
Нажмите Enter

Основные операции с файлами и папками

Открыть файл: Двойной клик левой кнопкой мыши

Переименовать:

Правой кнопкой мыши на файле/папке
Выберите **Rename**
Ведите новое название
Нажмите Enter

Удалить:

Правой кнопкой мыши на файле/папке
Выберите **Delete**
Подтвердите удаление

Копировать путь:

Правой кнопкой мыши на файле
Выберите **Copy Path**
Теперь путь к файлу в буфере обмена

Организация структуры проекта

Правильная структура для веб-проекта:

text
МОЙ САЙТ/
|——  images/ # Папка для изображений

```
├─── └─── css/      # Папка для стилей  
├─── └─── js/       # Папка для скриптов  
└─── └─── index.html # Главная страница  
└─── └─── about.html # Страница "О нас"  
└─── └─── style.css  # Основные стили
```

Как создать такую структуру:

Создайте основную папку проекта

Добавьте папку images **для картинок**

Создайте папку css **для файлов стилей**

Создайте папку js **для JavaScript-файлов**

Создайте файл index.html **в корневой папке**

Практическое задание

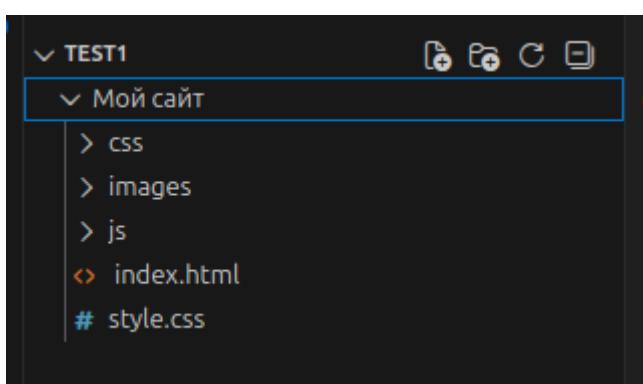
Давайте создадим реальную структуру проекта:

Откройте Проводник (Ctrl + Shift + E)

Нажмите "Open Folder" и создайте папку "Мой первый сайт"

Создайте структуру:

```
└─── css  
└─── images  
└─── js  
└─── index.html  
└─── style.css
```

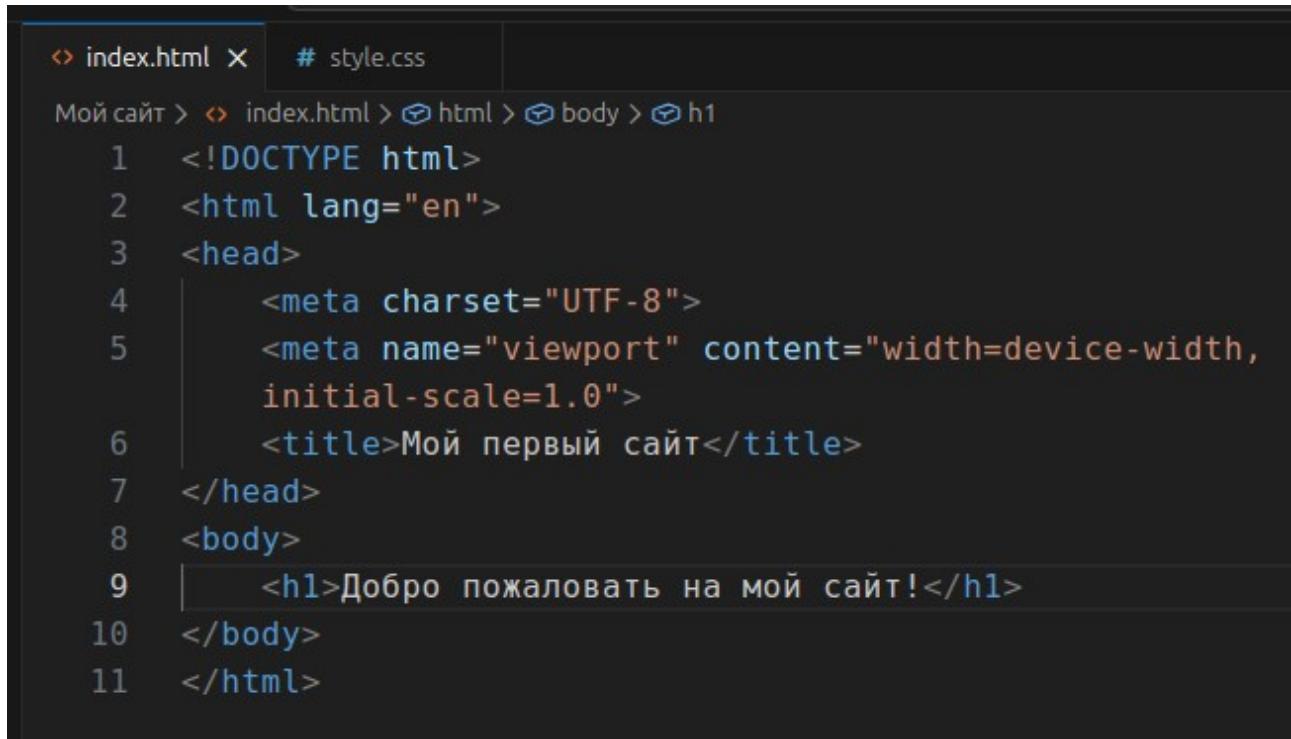


Откройте index.html двойным кликом

Напишите базовый HTML-код:

```
html  
<!DOCTYPE html>
```

```
<html>
<head>
<title>Мой первый сайт</title>
</head>
<body>
<h1>Добро пожаловать на мой сайт!</h1>
</body>
</html>
```



The screenshot shows a code editor window with two tabs: 'index.html' and '# style.css'. The '# style.css' tab is currently inactive. Below the tabs, a breadcrumb navigation bar shows the file path: 'Мой сайт > index.html > html > body > h1'. The main area displays the HTML code with line numbers from 1 to 11. The code includes doctype, meta tags for charset and viewport, a title, and a single h1 element.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width,
6          initial-scale=1.0">
6      <title>Мой первый сайт</title>
7  </head>
8  <body>
9      <h1>Добро пожаловать на мой сайт!</h1>
10 </body>
11 </html>
```

Полезные функции Проводника

Поиск в проводнике:

Нажмите **Ctrl + F** находясь в проводнике
Ведите название файла
Быстро находите нужные файлы

Сортировка файлов:

Правой кнопкой в пустой области проводника
Sort by Name — по имени
Sort by Type — по типу
Sort by Modified — по дате изменения

Скрытие файлов:

Создайте файл .vscode/settings.json

Добавьте код:

```
json
{
  "files.exclude": {
    "**/*.log": true,
    "**/node_modules": true
  }
}
```

Горячие клавиши для работы с Проводником

Ctrl + Shift + E	— открыть проводник
Ctrl + P	— быстрый поиск файла по имени
Ctrl + B	— скрыть/показать всю левую панель
F2	— переименовать выделенный файл
Delete	— удалить выделенный файл

Решение частых проблем

Файл не сохраняется:

Нажмите Ctrl + S после изменений
Или включите автосохранение в настройках

Не вижу созданный файл:

Обновите проводник (Ctrl + R)
Проверьте, в той ли папке вы создали файл

Заблудились в папках:

Используйте хлебные крошки вверху проводника
Или нажмите на доменную иконку  чтобы подняться в корень

Не могу найти нужный файл:

Используйте поиск (Ctrl + F в проводнике)
Или глобальный поиск (Ctrl + Shift + F)

Советы для эффективной работы

Создавайте логичную структуру папок с самого начала

Давайте файлам понятные имена на английском языке

Используйте папки для группировки похожих файлов

Регулярно сохраняйте файлы (Ctrl + S)

Используйте горячие клавиши для скорости

Теперь вы знакомы с Проводником файлов! Это ваш главный инструмент для организации работы над сайтом. Практикуйтесь создавать папки и файлы — скоро это станет для вас естественным процессом!

Основная область редактирования

Дорогой друг!

Теперь давайте изучим самую важную часть VS Code — **основную область редактирования**. Это то место, где вы будете проводить большую часть времени, создавая и редактируя код вашего сайта.

Что такое основная область редактирования?

Это центральная большая часть окна VS Code, где вы непосредственно работаете с кодом. Представьте, что это ваш **цифровой рабочий стол**, где вы пишете, редактируете и просматриваете код вашего сайта.

Из чего состоит область редактирования

Когда вы откроете файл, вы увидите:

text

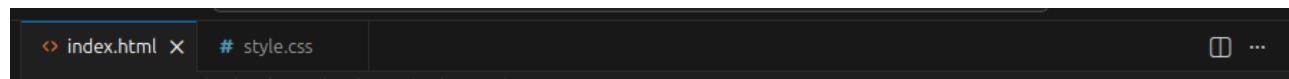
[ВКЛАДКИ] [ПАНЕЛЬ РЕДАКТИРОВАНИЯ] [ВЕРТИКАЛЬНАЯ ПАНЕЛЬ]

```
index.html # style.css
Мой сайт > index.html > html > body > h1
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width,
6          initial-scale=1.0">
6      <title>Мой первый сайт</title>
7  </head>
8  <body>
9      <h1>Добро пожаловать на мой сайт!</h1>
10 </body>
11 </html>
```

Строка 9, столбец 43

Давайте разберем каждый элемент подробно.

1. Панель вкладок (сверху)



Это горизонтальная панель вверху, где отображаются все открытые файлы.

Как выглядит:

text
[index.html] [style.css] [script.js] ×

Особенности вкладок:

Активная вкладка выделена цветом

Точка (•) означает несохраненные изменения

Крестик (x) закрывает вкладку

Можно перетаскивать вкладки мышью для изменения порядка

Как работать с вкладками:

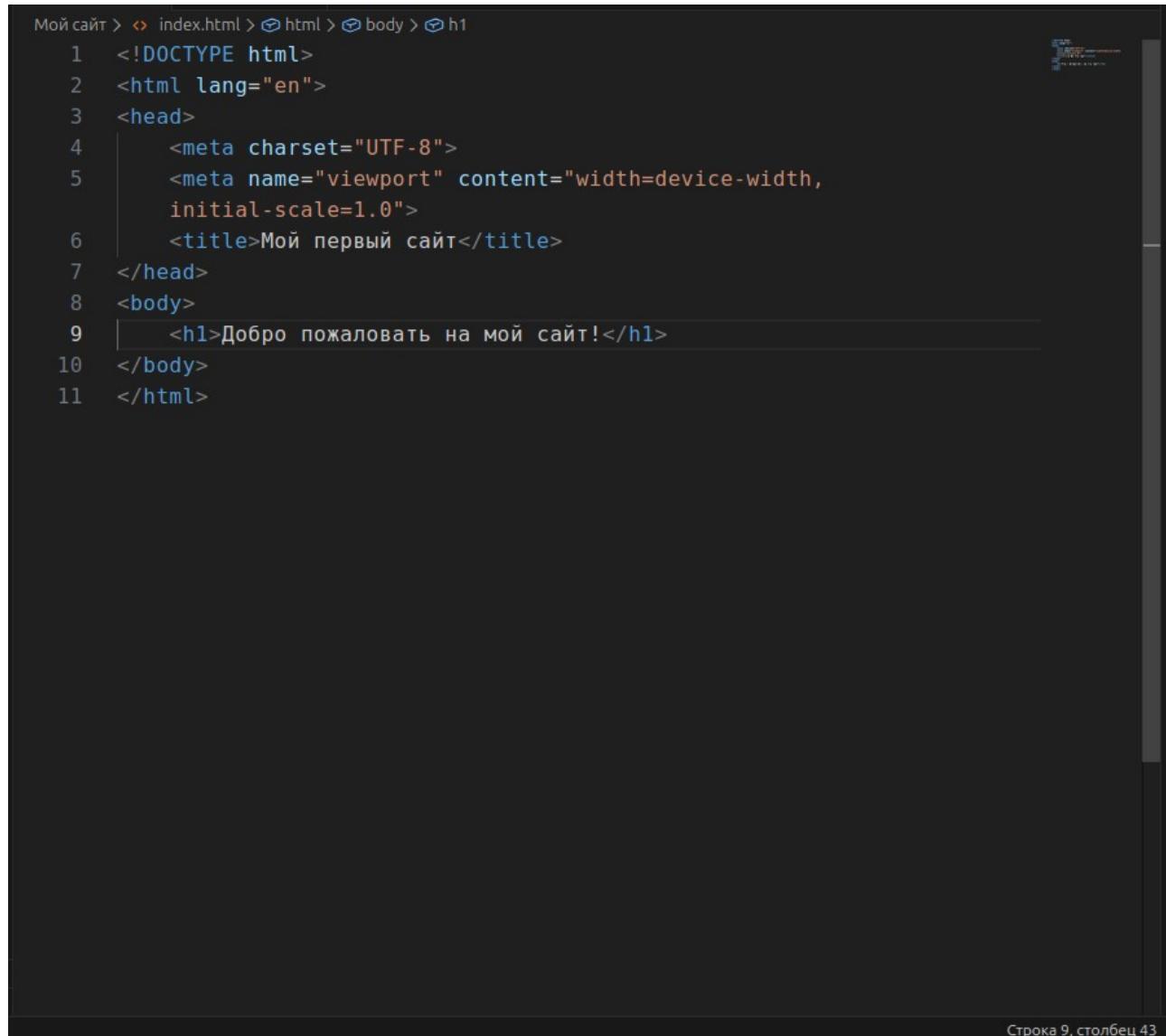
Переключение между вкладками: Клик левой кнопкой мыши

Закрытие вкладки: Клик на крестик или **Ctrl + W**

Закрытие всех вкладок: Правой кнопкой → Close All

Закрытие других вкладок: Правой кнопкой → Close Others

2. Область написания кода



Мой сайт > index.html > html > body > h1

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width,
6          initial-scale=1.0">
6      <title>Мой первый сайт</title>
7  </head>
8  <body>
9      <h1>Добро пожаловать на мой сайт!</h1>
10 </body>
11 </html>
```

Строка 9, столбец 43

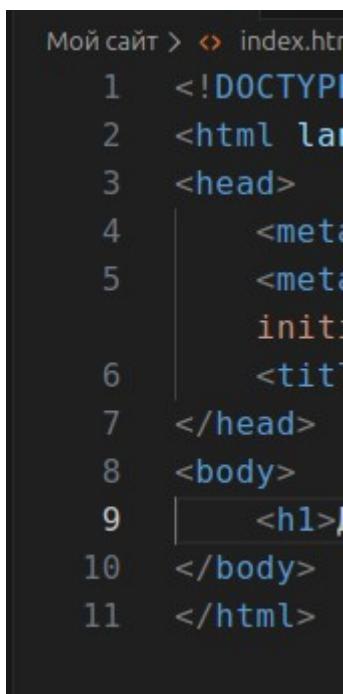
Это основное пространство, где вы видите и редактируете код.

Элементы области редактирования:

Нумерация строк:

html

```
1 | <!DOCTYPE html>
2 | <html>
3 | <head>
4 |   <title>Мой сайт</title>
5 | </head>
```



A screenshot of a code editor window titled "Мой сайт > index.htm". The code is numbered from 1 to 11. The syntax highlighting includes blue for HTML tags like <!DOCTYPE>, <html>, <head>, <title>, <meta>, <body>, <h1>; red for attributes like lang and title; and white for text content.

```
1 | <!DOCTYPE html>
2 | <html lang="ru">
3 | <head>
4 |   <meta charset="UTF-8">
5 |   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 |   <title>Мой сайт</title>
7 | </head>
8 | <body>
9 |   <h1>Мой сайт</h1>
10| </body>
11| </html>
```

Подсветка синтаксиса:

HTML-теги выделяются синим цветом

Атрибуты — голубым

Значения — красным

Текст — белым/черным в зависимости от темы

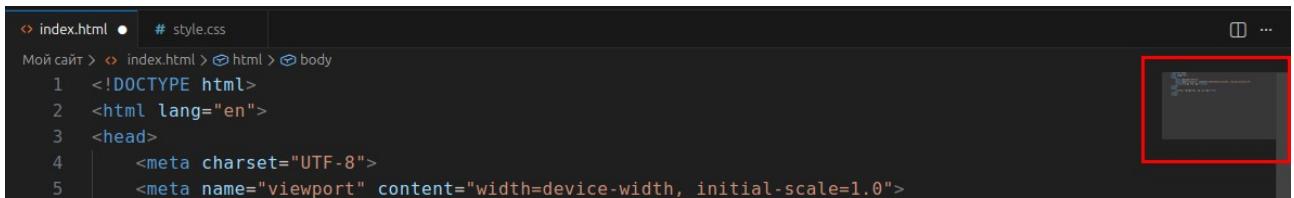
Вертикальная полоса:

Показывает рекомендуемую ширину кода (обычно 80-120 символов)

Помогает сохранять код аккуратным

3. Мини-карта (справа)

Это уменьшенная копия вашего файла, которая отображается справа.



index.html # style.css

Мой сайт > index.html > html > body

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Для чего нужна мини-карта:

Быстрая навигация по большим файлам

Понимание общей структуры кода

Быстрый переход к нужному участку

Как использовать мини-карту:

Кликните на любую область мини-карты — курсор перейдет в это место

Выделенная область показывает видимую часть кода

Можно отключить через настройки если мешает

Режимы отображения

Один файл:

Одна вкладка занимает всю область

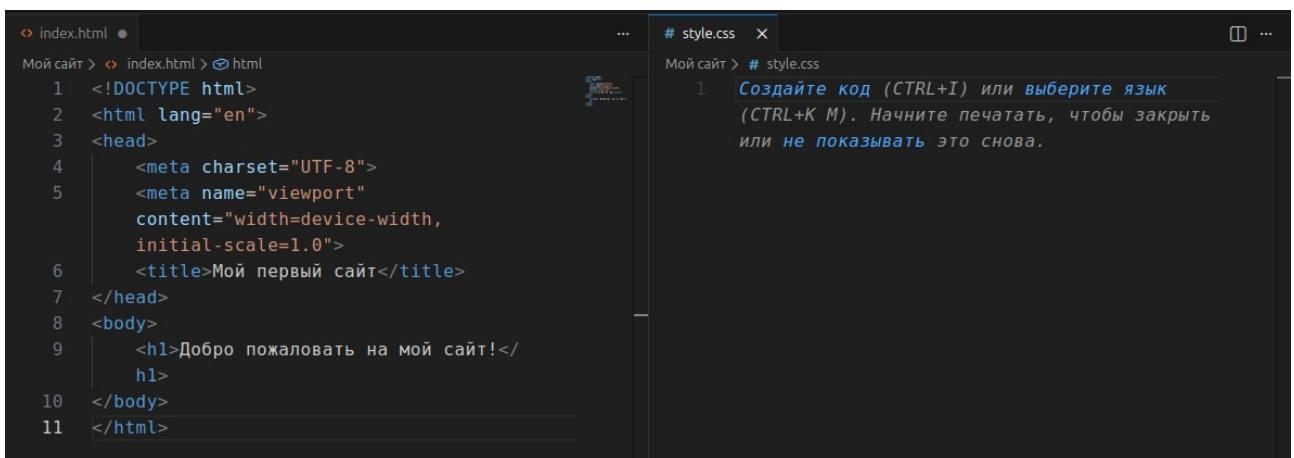
Идеально для концентрации

Разделение экрана:

text

[index.html] [style.css]

[HTML код] [CSS код]



index.html # style.css

Мой сайт > index.html > html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport"
       content="width=device-width,
               initial-scale=1.0">
6   <title>Мой первый сайт</title>
7 </head>
8 <body>
9   <h1>Добро пожаловать на мой сайт!</h1>
10 </body>
11 </html>
```

Мой сайт > # style.css

```
1 Создайте код (CTRL+I) или выберите язык
(CTRL+K M). Начните печатать, чтобы закрыть
или не показывать это снова.
```

Создать разделение: Ctrl + \

Перемещение между панелями: Ctrl + 1, Ctrl + 2

Закрытие панели: Ctrl + W

Интеллектуальные возможности редактора

Автодополнение кода:

Когда вы начинаете писать код, VS Code предлагает варианты завершения:

html

```
<!-- Начните писать "di" -->  
di → div (нажмите Tab или Enter)
```

Подсказки параметров:

При наведении на HTML-тег или CSS-свойство появляется всплывающая подсказка с описанием.

Выделение парных скобок и тегов:

Когда мышкой кликаем на теге, парный тег подсвечивается:

html

```
<div> ← подсвечивается, когда мышкой кликаем здесь
```

Текст

```
</div> ← и здесь
```

Горячие клавиши для редактирования

Базовые команды:

Ctrl + S — сохранить файл

Ctrl + Z — отменить действие

Ctrl + Y — повторить действие

Ctrl + C — копировать

Ctrl + X — вырезать

Ctrl + V — вставить

Навигация по тексту:

Home	— в начало строки
End	— в конец строки
Ctrl + Home	— в начало файла
Ctrl + End	— в конец файла
Ctrl + →	— на слово вперед
Ctrl + ←	— на слово назад

Работа с выделением:

Ctrl + A	— выделить все
Ctrl + D	— выделить следующее вхождение слова
Alt + клик	— добавить курсор в другое место
Ctrl + Shift + L	— выделить все вхождения слова

Практическое задание

Давайте создадим и отредактируем настоящий HTML-файл:

Создайте новый файл index.html

Ведите базовую структуру HTML:

```
html
<!DOCTYPE html>
<html>
<head>
<title>Мой первый сайт</title>
</head>
<body>
<h1>Привет, мир!</h1>
<p>Это мой первый сайт</p>
</body>
</html>
```

Поэкспериментируйте с функциями:

Попробуйте автодополнение (начните писать <h2>)

Используйте подсветку ошибок (удалите закрывающий тег)

Сохраните файл (Ctrl + S)

Разделите экран (Ctrl + \) и откройте тот же файл

Полезные настройки для области редактирования

Добавьте в настройки (Ctrl + ,):

```
json
{
// Перенос строк
"editor.wordWrap": "on",

// Нумерация строк
"editor.lineNumbers": "on",

// Мини-карта
"editor.minimap.enabled": true,

// Подсветка текущей строки
"editor.renderLineHighlight": "all",

// Автоформатирование при сохранении
"editor.formatOnSave": true
}
```

Решение частых проблем

Текст слишком мелкий:

Увеличьте масштаб: Ctrl + =

Или настройте шрифт в настройках

Не вижу нумерации строк:

Проверьте настройку "editor.lineNumbers": "on"

Или правой кнопкой в области кода → Toggle Line Numbers

Пропала мини-карта:

Нажмите Ctrl + Shift + P

Ведите: View: Toggle Minimap

Файл не сохраняется:

Проверьте, нет ли точки (•) на вкладке — это означает несохраненные изменения

Нажмите **Ctrl + S**

Советы для эффективной работы

Используйте горячие клавиши — это ускорит работу в разы

Доверяйте автодополнению — оно редко ошибается

Следите за подсветкой ошибок — красный цвет означает проблему

Регулярно сохраняйтесь (**Ctrl + S**)

Используйте разделение экрана для одновременной работы с HTML и CSS

Теперь вы знакомы с основной областью редактирования! Это ваше главное рабочее пространство, где будут рождаться все ваши будущие сайты. Практикуйтесь каждый день, и скоро работа с кодом станет для вас такой же естественной, как письмо на бумаге!

Панель состояния (внизу)

Дорогой друг!

Давайте изучим **Панель состояния** — это информационная строка в самом низу окна VS Code. Она работает как приборная панель в автомобиле, показывая важную информацию о вашем проекте и текущем файле.

Что такое Панель состояния?



Это серая или синяя полоса в нижней части окна VS Code, которая отображает:

Информацию о текущем файле
Состояние редактора
Уведомления о ошибках и предупреждениях
Быстрый доступ к настройкам

Из чего состоит Панель состояния

Вот как выглядит типичная панель состояния:

text [] [main] [0 2] [UTF-8] [LF] [Spaces: 4] [HTML] [] []

Давайте разберем каждый элемент слева направо.

1. Ветка Git (синий индикатор)

Что показывает: Текущую ветку системы контроля версий Git

Как выглядит: [main] или [master]

Что означает:

Синий круг — статус Git (синий = всё нормально)

main/master — название текущей ветки

Для начинающих: Пока можете не обращать внимания, но знать о её существовании полезно.

2. Индикатор ошибок и предупреждений

Что показывает: Количество ошибок и предупреждений в проекте

Как выглядит: [2 5]



Что означает:

🔴 Красный кружок с цифрой — количество ошибок

⚠ Желтый треугольник с цифрой — количество предупреждений

Практическое использование:

Кликните на индикатор ошибок — откроется панель с списком проблем

Исправляйте ошибки по мере их появления

Предупреждения часто можно игнорировать, но лучше проверять

3. Кодировка файла

UTF-8

Что показывает: Кодировку текущего файла

Как выглядит: [UTF-8]

Что означает:

UTF-8 — стандартная кодировка для веб-проектов

Другие варианты: ASCII, Windows-1251, ISO-8859-1

Как изменить:

Кликните на надпись "UTF-8"

Выберите нужную кодировку из списка

Для веб-разработки всегда используйте UTF-8

4. Формат конца строк

Что показывает: Формат переноса строк

Как выглядит: [LF] или [CRLF]

Что означает:

LF (Line Feed) — стандарт для Linux/Mac

CRLF (Carriage Return + Line Feed) — стандарт для Windows

Практическое значение:

Обычно VS Code автоматически выбирает правильный формат

Может влиять на отображение файлов в разных операционных системах

5. Отступы (Indentation)

Пробелов: 4

Что показывает: Настройки отступов в файле

Как выглядит: [Spaces: 4] или [Tab Size: 2]

Что означает:

Spaces: 4 — используются пробелы, размер отступа 4 символа

Tab Size: 2 — используется указанное количество нажатий на кнопку Tab, размер отступа 2 символа

Как изменить:

Кликните на надпись

Выберите "Indent Using Spaces" или "Indent Using Tabs"

Выберите размер отступа (2, 4, 8)

Рекомендация для веб-разработки: Используйте "Spaces: 2"

6. Язык файла

Что показывает: Язык программирования текущего файла

Как выглядит: [HTML], [CSS], [JavaScript]

Что означает:

Определяет подсветку синтаксиса

Влияет на автодополнение кода

Устанавливается автоматически по расширению файла

Как изменить:

Кликните на название языка

Выберите нужный язык из списка

Или создайте файл с правильным расширением (.html, .css, .js)

7. Уведомления (колокольчик)



Что показывает: Наличие системных уведомлений

Как выглядит: [🔔] (иногда с цифрой)

Что означает:

Обычный колокольчик — уведомлений нет

Колокольчик с точкой — есть непрочитанные уведомления

Колокольчик с цифрой — количество уведомлений

Как использовать:

Кликните на колокольчик чтобы просмотреть уведомления

Здесь появляются сообщения об обновлениях, ошибках плагинов и т.д.

8. Аккаунт (профиль)

Что показывает: Статус входа в аккаунт Microsoft

Как выглядит: [👤]

Что означает:

Просто иконка — не авторизован

Иконка с галочкой — авторизован

Для начинающих: Авторизация не обязательна для базовой работы.

Практическое использование Панели состояния

Пример реальной работы:

Откройте файл с ошибкой:

text

[main] [1 3] [UTF-8] [LF] [Spaces: 2] [HTML] [] []

Кликните на 1 — откроется панель проблем

Исправьте ошибку в коде

Наблюдайте как индикатор изменится: [0 3]

Полезные функции Панели состояния

Быстрый доступ к настройкам:

Кликните на любой элемент для быстрого изменения

Не нужно искать в меню

Мониторинг состояния проекта:

Сразу видно, есть ли ошибки

Можно быстро оценить качество кода

Информация о файле:

Всегда знаете в какой кодировке работаете

Контролируете форматирование кода

Настройка Панели состояния

Вы можете настроить отображаемые элементы через настройки:

Нажмите **Ctrl + ,**

Ведите в поиск: **status bar**

Настройте нужные параметры

Или добавьте в **settings.json**:

```
json
{
  "workbench.statusBar.visible": true,
  "git.countBadge": "all",
  "problems.autoReveal": true
}
```

Решение частых проблем

Панель состояния пропала:

Нажмите **Ctrl + Shift + P**

Ведите: **View: Toggle Status Bar Visibility**

Или через меню: **View → Appearance → Show Status Bar**

Не вижу нужный индикатор:

Возможно, он скрыт настройками

Проверьте настройки статусной панели

Постоянно меняется кодировка:

Установите кодировку по умолчанию в настройках:

```
json
{
  "files.encoding": "utf8"
}
```

Советы для эффективной работы

Регулярно смотрите на индикатор ошибок — исправляйте проблемы сразу

Используйте кликабельные элементы для быстрого доступа к настройкам

Следите за кодировкой чтобы избежать проблем с русским текстом

Настройте отступы под себя для удобства чтения кода

Обращайте внимание на уведомления — там может быть важная информация

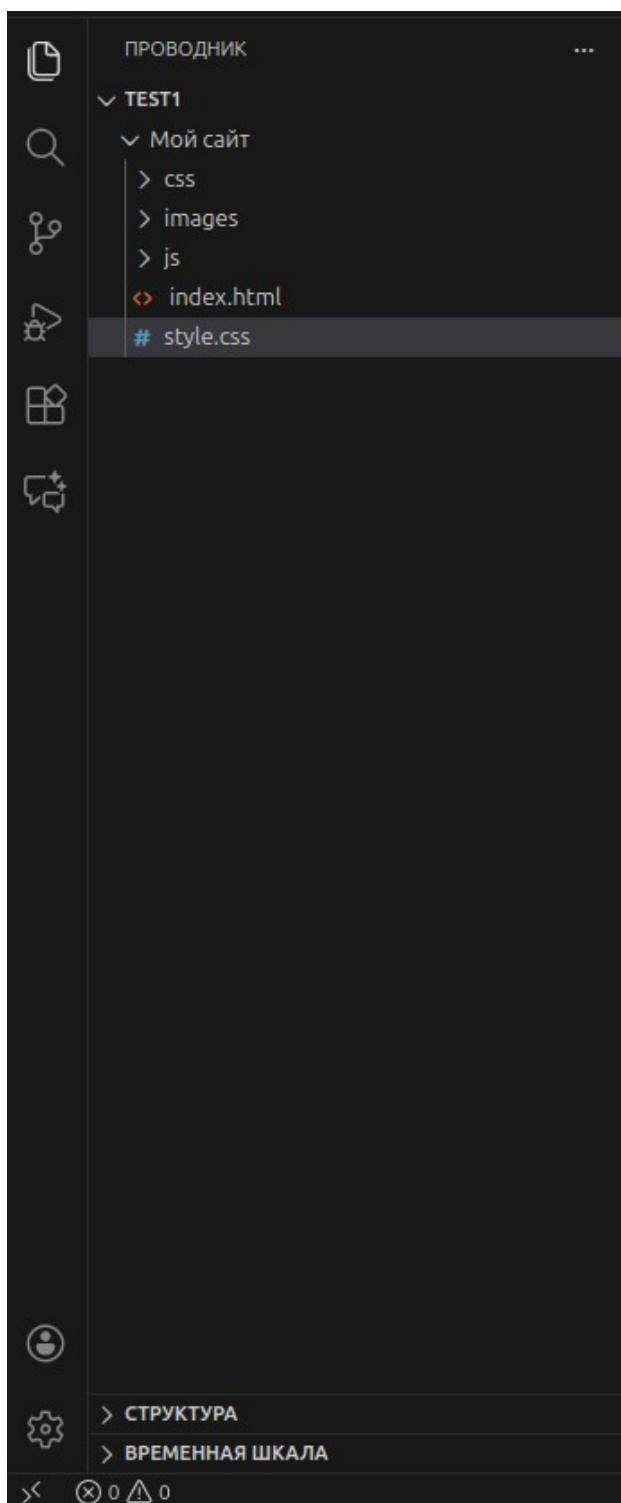
Теперь вы знакомы с Панелью состояния! Это ваш надежный помощник, который всегда подскажет, что происходит с вашим проектом. Со временем вы научитесь использовать её возможности на 100%!

Боковая панель

Дорогой друг!

Давайте подробно изучим **Боковую панель** в VS Code. Это многофункциональная область, которая может находиться слева или справа и содержит различные панели для удобной работы над проектом.

Что такое Боковая панель?



Это вертикальная область, которая обычно находится слева от основного редактора и содержит **Панель активности** с иконками. Но при необходимости её можно переместить вправо или даже скрыть.

Расположение Боковой панели

По умолчанию: Слева

text
[] [Основная область редактирования]

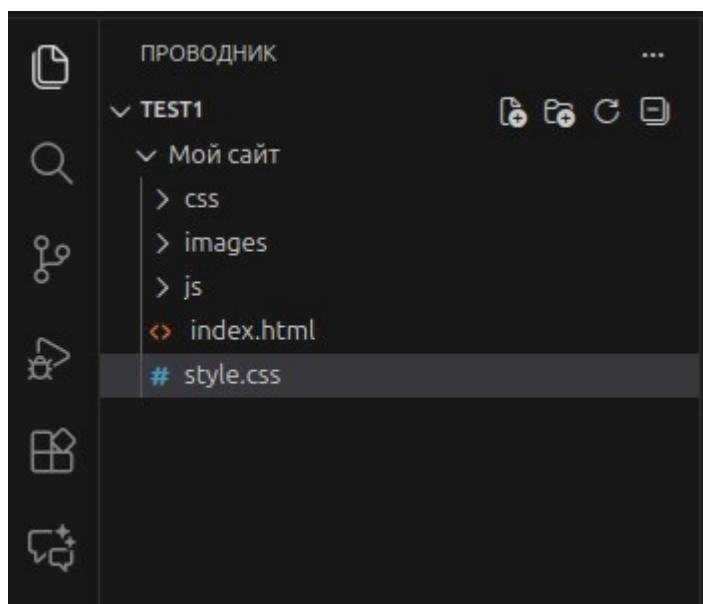
Можно переместить вправо:

text
[Основная область редактирования] []

Основные компоненты Боковой панели

Боковая панель состоит из двух основных частей:

1. Панель активности (вертикальный ряд иконок)



text
 - Проводник
 - Поиск
- Система контроля версий
▷ - Запуск и отладка
 - Расширения

2. Контентная область

Отображает содержимое выбранной иконки
Меняется в зависимости от выбранного инструмента

Подробный разбор каждой панели

1. ПРОВОДНИК (Explorer) -

Что содержит:

Структуру файлов и папок проекта
Открытые редакторы
Кнопки для создания файлов и папок

Практическое использование:

html

МОЙ ПРОЕКТ/

```
├── └── images/
├── └── css/
│   └── └── style.css
└── └── js/
└── └── index.html
└── └── about.html
```

2. ПОИСК (Search) -

Что содержит:

Поле для ввода поискового запроса
Поле для замены текста
Настройки поиска
Результаты поиска

Пример использования:

text

Найти: "header"

Заменить: "main-header"

3. СИСТЕМА КОНТРОЛЯ ВЕРСИЙ (Source Control) -

Что содержит:

Измененные файлы

Инструменты для коммитов

Историю изменений

Для начинающих: Пока можно не использовать, но знать о её существовании полезно.

4. ЗАПУСК И ОТЛАДКА (Run and Debug) - ▶

Что содержит:

Конфигурации запуска

Кнопки для отладки

Переменные и точки останова

Практическое значение: Понадобится когда будем устанавливать Live Server.

5. РАСШИРЕНИЯ (Extensions) - ☒

Что содержит:

Поиск расширений

Установленные расширения

Рекомендации

Обновления

Управление Боковой панелью

Переключение между панелями

С помощью мыши:

Кликните на нужную иконку в Панели активности

С помощью горячих клавиш:

Ctrl+Shift+E - Проводник

Ctrl+Shift+F - Поиск

Ctrl+Shift+G - Git

Ctrl+Shift+D - Запуск и отладка

Ctrl+Shift+X - Расширения

Изменение размера Боковой панели

С помощью мыши:

Наведите курсор на границу панели

Когда появится двусторонняя стрелка ←→
Перетащите границу влево или вправо

С помощью клавиатуры:

Ctrl+B - показать/скрыть боковую панель
Ctrl+Shift+P → "View: Toggle Side Bar Visibility"

Перемещение Боковой панели

В правую часть экрана:

Нажмите Ctrl+Shift+P
Введите: View: Move Side Bar Right
Нажмите Enter

Обратно в левую часть:

Ctrl+Shift+P
View: Move Side Bar Left
Enter

Дополнительные панели

При установке расширений могут появиться новые иконки в Боковой панели:

-  **Live Server** - для запуска локального сервера
-  **GitLens** - расширенные возможности Git
-  **Color Picker** - работа с цветами
-  **Settings Sync** - синхронизация настроек

Практическое задание

Давайте освоим работу с Боковой панелью:

- Откройте Проводник (Ctrl+Shift+E)**
- Создайте новую папку "practice"**
- Создайте файл test.html в этой папке**
- Перейдите в Поиск (Ctrl+Shift+F)**
- Введите в поиск "html"**
- Вернитесь в Проводник**
- Попробуйте изменить размер панели перетаскиванием**
- Скройте панель (Ctrl+B) и снова откройте**

Настройка Боковой панели

Добавьте в настройки (Ctrl+,):

```
json
{
    // Положение Боковой панели
    "workbench.sideBar.location": "left",

    // Размер иконок в Панели активности
    "workbench.iconTheme": "vs-seti",

    // Автоматически скрывать панель
    "workbench.activityBar.visible": true,

    // Показывать только нужные иконки
    "workbench.activityBar.iconClickBehavior": "toggle"
}
```

Полезные советы

Используйте горячие клавиши для быстрого переключения

Настройте размер панели под свои предпочтения

Скрывайте панель когда нужно больше места для кода

Изучите все панели чтобы понимать их возможности

Добавляйте полезные расширения которые добавят новые панели

Решение частых проблем

Боковая панель пропала:

Нажмите Ctrl+B

Или View → Appearance → Show Side Bar

Не вижу нужную иконку:

Возможно, она скрыта настройками

Проверьте через View → Appearance → Show Activity Bar

Панель слишком узкая/широкая:

Перетащите границу панели мышью

Или сбросьте настройки через `View → Reset View Locations`

Иконки слишком мелкие:

Увеличьте масштаб интерфейса: `Ctrl+=`

Или настройте в параметрах доступности

Горячие клавиши для работы с Боковой панелью

`Ctrl+B` - показать/скрыть Боковую панель

`Ctrl+Shift+E` - перейти в Проводник

`Ctrl+Shift+F` - перейти в Поиск

`Ctrl+Shift+X` - перейти в Расширения

`Ctrl+0` - перейти фокус на Боковую панель

Теперь вы полностью освоили Боковую панель! Это ваш главный навигационный центр в VS Code, который сделает работу над проектами удобной и организованной. Практикуйтесь использовать разные панели — скоро это станет вашей второй натурой!

Часть 2: Организация работы над сайтом

Глава 4: Первый проект - создаем папку для сайта

Как создать папку для проекта на Рабочем столе

Дорогой друг!

Давайте научимся создавать папку для проекта прямо на Рабочем столе. Это важный первый шаг — чтобы все файлы вашего будущего сайта были в одном месте и не потерялись.

Способ 1: Создание папки прямо на Рабочем столе

Шаг 1: Откройте Рабочий стол

Закройте или сверните все окна

Нажмите на клавишу **Win** (клавиша с эмблемой Windows)

Или нажмите на кнопку «Свернуть все окна» в правом нижнем углу

Шаг 2: Создайте новую папку

Наведите курсор на пустое место Рабочего стола

Нажмите правую кнопку мыши

В появившемся меню выберите «**Создать**»

В дополнительном меню выберите «**Папку**»

Шаг 3: Дайте папке название

Вы увидите новую папку с подсвеченным названием «Новая папка»

Сразу начинайте печатать нужное название

Например: «**Мой первый сайт**»

Нажмите **Enter** для сохранения

Способ 2: Использование горячих клавиш

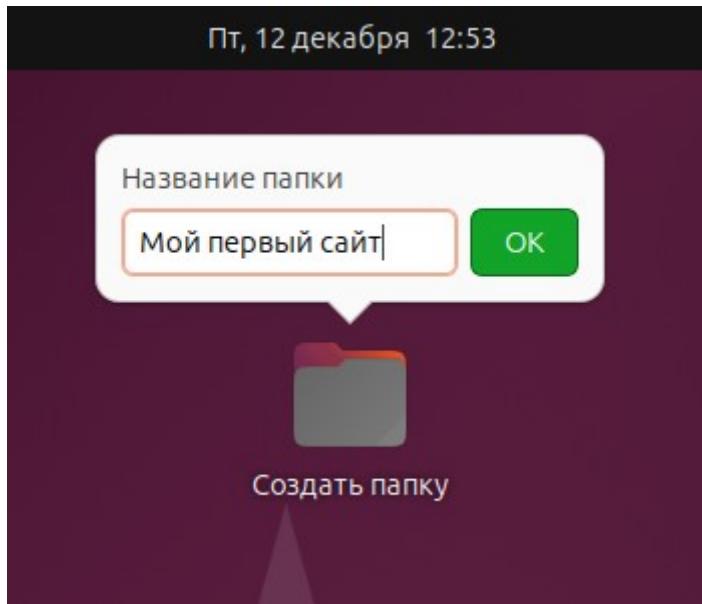
Более быстрый способ:

Откройте Рабочий стол

Нажмите **Ctrl + Shift + N** одновременно

Сразу введите название папки

Нажмите **Enter**



Способ 3: Через Проводник Windows

Шаг 1: Откройте Проводник

Нажмите **Win + E**

Или кликните на иконку Проводника в панели задач

Шаг 2: Перейдите на Рабочий стол

В левом меню найдите и кликните «**Рабочий стол**»

Или введите в адресной строке: **C:\Users\ВашеИмя\Desktop**

Шаг 3: Создайте папку

Нажмите кнопку «**Новая папка**» в верхней панели

Или правой кнопкой → «Создать» → «Папку»

Рекомендации по названию папки

Правильно:

Мой первый сайт
Сайт-визитка
Проект HTML
my_first_website

Неправильно:

новая папка (непонятно что внутри)
проект 1 (может быть много таких)
сайт (слишком общее название)

Практическое задание

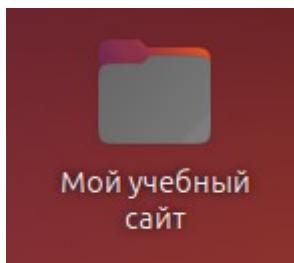
Давайте создадим папку для реального проекта:

Откройте Рабочий стол

Создайте папку под названием «Мой учебный сайт»

Откройте папку двойным кликом мыши

Убедитесь, что папка пустая

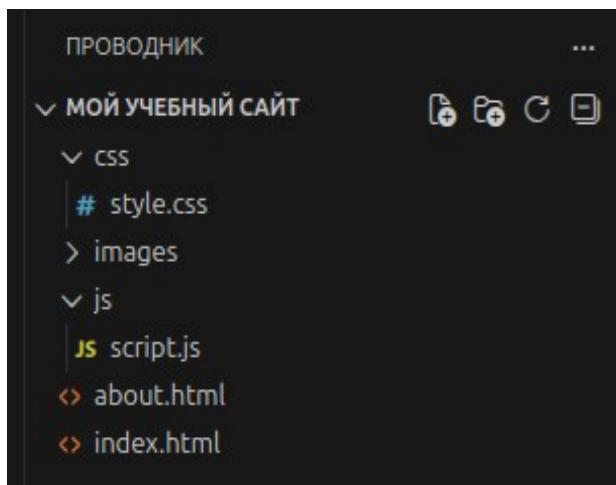


Структура папки проекта

Правильная организация с самого начала:

text

```
Мой учебный сайт/
├── images/    # для картинок
├── css/       # для стилей
├── js/        # для скриптов
└── index.html # главная страница
   └── about.html # страница "Обо мне"
```



Как создать такую структуру

Вариант 1: Сразу создать все папки

На Рабочем столе: правая кнопка → «Создать» → «Папка»

Назовите: «Мой учебный сайт»

Откройте папку двойным кликом

Внутри создайте еще три папки: `images`, `css`, `js`

Вариант 2: Создавать по мере необходимости

Сначала только основную папку

Остальные папки создавать когда они понадобятся

Проверка создания папки

Убедитесь, что папка создана правильно:

- Папка находится на Рабочем столе
- Название понятное и без ошибок
- Папка открывается двойным кликом
- Внутри пусто (или созданные подпапки)

Частые ошибки и их решение

Ошибка 1: «Папка с таким именем уже существует»

Решение: Выберите другое название

Например: «Мой сайт 2» или «Мой проект HTML»

Ошибка 2: Не могу найти созданную папку

Решение: Используйте поиск на Рабочем столе

Или отсортируйте файлы по дате создания

Ошибка 3: Случайно переименовал другую папку

Решение: Нажмите `Ctrl + Z` для отмены

Или верните прежнее название вручную

Дополнительные возможности

Изменение значка папки:

Правая кнопка на папке → «Свойства»

Вкладка «Настройка»

Кнопка «Сменить значок»

Выберите подходящий значок

Создание ярлыка папки:

Правая кнопка на папке → «Создать ярлык»

Ярлык можно переместить в удобное место

Полезные советы

Создавайте папку ДО начала работы в VS Code

Используйте понятные названия — через месяц будете помнить что внутри

Храните все файлы проекта в одной папке — так ничего не потеряется

Делайте резервные копии важных проектов

Используйте единый стиль названий для всех проектов

Подготовка к работе в VS Code

После создания папки:

Запомните её расположение (на Рабочем столе)

Запомните точное название (например, «Мой учебный сайт»)

Будьте готовы открыть её в VS Code в следующем шаге

Практическое упражнение

Потренируйтесь создавать папки:

Создайте папку «Тестовая папка»

Переименуйте её в «Учебные материалы»

Удалите её (правая кнопка → «Удалить»)

Восстановите из Корзины (если нужно)

Теперь вы умеете создавать папки для проектов! Это базовый, но очень важный навык для любого веб-разработчика. Все профессиональные разработчики начинают каждый новый проект именно с этого шага!

Как открыть папку в VS Code

Дорогой друг!

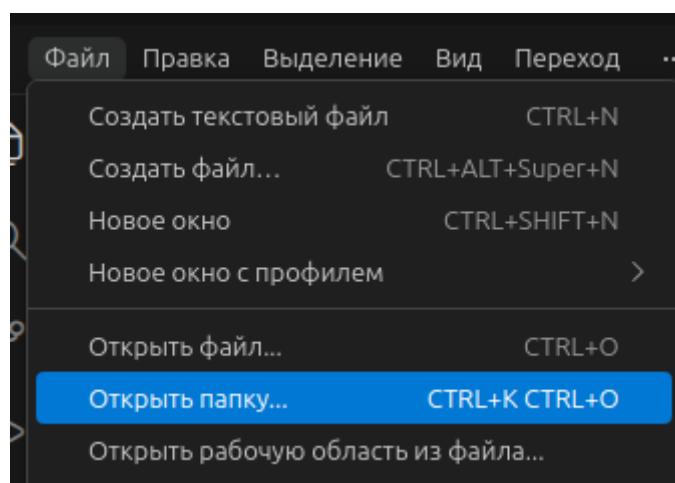
Теперь, когда мы создали папку для проекта на Рабочем столе, давайте научимся открывать её в VS Code. Это важный шаг — ведь именно так вы начинаете работу над любым проектом.

Способ 1: Через меню File (самый простой способ)

Шаг 1: Запустите VS Code

Найдите синий значок на Рабочем столе
Дважды щелкните по нему левой кнопкой мыши
Дождитесь полной загрузки программы

Шаг 2: Откройте меню File



В левом верхнем углу нажмите «File»
В выпадающем меню выберите «Open Folder...»
Или просто нажмите:
Ctrl + K затем Ctrl + O (удерживая Ctrl, нажмите K, затем O)
Или Ctrl + O и выберите «Open Folder»

Шаг 3: Выберите вашу папку

Откроется окно Проводника Windows
Найдите **Рабочий стол** в левом меню
Найдите папку «**Мой учебный сайт**»
Выделите её одни кликом мыши
Нажмите кнопку «**Выбор папки**»

Способ 2: Перетаскивание папки в VS Code

Очень наглядный способ:

Откройте VS Code

Откройте Рабочий стол (сверните или закройте другие окна)

Захватите папку «Мой учебный сайт» левой кнопкой мыши

Перетащите её прямо в область VS Code

Отпустите кнопку мыши

Результат: Папка автоматически откроется в VS Code!

Способ 3: Через контекстное меню папки

Самый быстрый способ (после настройки):

Найдите папку «Мой учебный сайт» на Рабочем столе

Нажмите правую кнопку мыши на папке

Выберите «Open with Code»

Если этого пункта нет — сначала установите его:

Откройте VS Code

Нажмите Ctrl + Shift + P

Ведите: Shell Command: Install 'code' command in PATH

Нажмите Enter

Перезапустите VS Code

Способ 4: Использование командной строки

Для продвинутых пользователей:

Нажмите Win + R

Ведите: cmd → Enter

В командной строке введите:

text

cd Desktop

code "Мой учебный сайт"

Что должно произойти после открытия папки

Признаки успешного открытия:

В Панели активности слева (Проводник) вы увидите:

text

МОЙ УЧЕБНЫЙ САЙТ

└──  (пусто)

В заголовке окна VS Code появится название папки:

text

Мой учебный сайт - Visual Studio Code

В нижнем левом углу (статус бар) отображается название папки

Практическое задание

Давайте откроем нашу папку в VS Code:

Создайте папку «Мой учебный сайт» на Рабочем столе (если ещё не создали)

Запустите VS Code

Используйте Способ 1 (через меню File)

Найдите и откройте вашу папку

Убедитесь, что папка открылась правильно

Если папка не открывается

Проблема 1: Не вижу папку в списке

Решение: Убедитесь, что вы находитесь в правильной директории (Рабочий стол)

Проверьте название папки — возможно, есть опечатка

Проблема 2: VS Code показывает ошибку

Решение: Закройте и перезапустите VS Code

Попробуйте другой способ открытия

Проблема 3: Открылась не та папка

Решение: Закройте текущую папку: File → Close Folder

Откройте нужную папку заново

Проверка правильности открытия

После открытия папки сделайте тест:

Создайте тестовый файл в VS Code:

В Проводнике наведите на название папки

Нажмите иконку «Новый файл»

Назовите файл `test.txt`

Нажмите Enter

Проверьте на Рабочем столе:

Откройте папку «Мой учебный сайт»

Убедитесь, что файл `test.txt` появился внутри

Удалите тестовый файл:

В VS Code: правой кнопкой на файле → Delete

Или на Рабочем столе: просто удалите файл

Дополнительные возможности

Открытие нескольких папок:

VS Code позволяет работать с несколькими проектами одновременно

`File → Add Folder to Workspace...`

Создание рабочего пространства:

Сохраните текущий набор папок как рабочее пространство

`File → Save Workspace As...`

Полезные советы

Всегда начинайте проект с открытия папки в VS Code

Используйте способ с перетаскиванием — он самый наглядный

Закрывайте папку после окончания работы: `File → Close Folder`

Запомните горячие клавиши `Ctrl + K` `Ctrl + O` для быстрого открытия

Следите за названием в заголовке — чтобы не перепутать проекты

Что дальше?

После успешного открытия папки вы можете:

Создавать файлы и папки прямо в VS Code

Редактировать код с подсветкой синтаксиса

Организовывать структуру проекта

Работать над своим сайтом!

Поздравляю! Теперь вы умеете открывать папки с проектами в VS Code. Это фундаментальный навык, который вы будете использовать каждый раз при работе над новым сайтом!

Создаем первую структуру файлов:

index.html

style.css

images (папка для картинок)

Дорогой друг!

Теперь давайте создадим правильную структуру файлов для вашего первого сайта. Это как построить прочный фундамент для дома — от этого зависит успех всего проекта!

Что мы будем создавать

text

МОЙ УЧЕБНЫЙ САЙТ/

```
└── index.html    # Главная страница
└── style.css     # Стили для сайта
└── images/        # Папка для изображений
```

Шаг 1: Создаем папку для изображений

В Проводнике VS Code:

Наведите курсор на название вашей папки «МОЙ УЧЕБНЫЙ САЙТ»

Нажмите на иконку «Новая папка» (папка с плюсом)

Введите название: images

Нажмите Enter

Результат:

text

МОЙ УЧЕБНЫЙ САЙТ/

```
└── images
```

Шаг 2: Создаем файл index.html

Способ 1: Через иконку нового файла

Наведите курсор на «МОЙ УЧЕБНЫЙ САЙТ» (не на папку images!)

Нажмите на иконку «Новый файл» (лист бумаги с плюсом)

Введите название: index.html

Нажмите Enter

Способ 2: Через правую кнопку мыши

Правой кнопкой мыши на «МОЙ УЧЕБНЫЙ САЙТ»

Выберите «New File»

Ведите: index.html

Нажмите Enter

Результат:

text
МОЙ УЧЕБНЫЙ САЙТ/
└──  images
└──  index.html

Шаг 3: Создаем файл style.css

Повторите те же действия:

Наведите курсор на «МОЙ УЧЕБНЫЙ САЙТ»

Нажмите иконку «Новый файл»

Ведите: style.css

Нажмите Enter

Финальная структура:

text
МОЙ УЧЕБНЫЙ САЙТ/
└──  images
└──  index.html
└──  style.css

Практическое задание

Давайте создадим структуру вместе:

Убедитесь, что папка «МОЙ УЧЕБНЫЙ САЙТ» открыта в VS Code

Создайте папку images

Создайте файл index.html

Создайте файл style.css

Проверьте, что структура выглядит правильно

Проверка создания файлов

Убедитесь, что все создано правильно:

В Проводнике VS Code вы видите все три элемента

Иконки файлов выглядят по-разному (HTML и CSS)

Названия файлов написаны без ошибок:

index.html (не index.htm или index.html.txt)

style.css (не style.csss или style.cass)

Что такое эти файлы и для чего они нужны

index.html

Это главная страница вашего сайта

Содержит структуру и содержание сайта

Открывается первой при заходе на сайт

Как книга — содержит текст и разметку

style.css

Это файл стилей вашего сайта

Определяет как сайт выглядит (цвета, шрифты, размеры)

Как дизайнер — делает сайт красивым и удобным

images

Это папка для всех изображений сайта

Хранит фотографии, иконки, картинки

Как фотоальбом — содержит все визуальные материалы

Частые ошибки и их решение

Ошибка 1: Файл создался не в той папке

Ситуация: Файл оказался внутри папки `images`

Решение: Удалите файл и создайте заново, убедившись что курсор на основной папке

Ошибка 2: Неправильное расширение файла

Ситуация: Создался файл `index.html.txt`

Решение: Переименуйте файл, убедившись что в названии только `index.html`

Ошибка 3: Опечатка в названии

Ситуация: `idx.html` вместо `index.html`

Решение: Правой кнопкой на файле → Rename → исправьте название

Правильные названия файлов

Рекомендуется:

Использовать **латинские буквы**

Использовать **нижний регистр** (строчные буквы)

Использовать **подчеркивание** вместо пробелов

Соблюдать **правильные расширения**

Примеры правильных названий:

```
index.html  
about_me.html  
main_style.css  
background.jpg
```

Дополнительные возможности

Создание нескольких HTML-страниц:

`about.html` — страница «Обо мне»

`contacts.html` — страница контактов

`gallery.html` — галерея фотографий

Создание дополнительных папок:

`css/` — для нескольких файлов стилей

`js/` — для JavaScript файлов

`fonts/` — для шрифтов

Проверка на реальной файловой системе

Убедитесь, что файлы создались на самом деле:

Откройте Проводник Windows

Перейдите на Рабочий стол

Откройте папку «Мой учебный сайт»

Проверьте, что внутри есть:

Папка `images`
Файл `index.html`
Файл `style.css`

Полезные советы

Соблюдайте порядок с самого начала
Используйте понятные названия файлов
Храните изображения в папке `images` — не разбрасывайте по разным местам
Создавайте структуру ДО начала написания кода
Регулярно сохраняйте файлы (`Ctrl + S`)

Что дальше?

После создания структуры мы:
Наполним `index.html` содержимым
Добавим стили в `style.css`
Разместим изображения в папке `images`
Свяжем все файлы вместе

Поздравляю! Вы создали свою первую профессиональную структуру проекта. Теперь у вас есть прочный фундамент для создания настоящего веб-сайта!

Глава 5: Проводник файлов - ваш главный помощник

Как создавать новые файлы и папки прямо в VS Code

Дорогой друг!

Давайте подробно изучим все способы создания файлов и папок прямо в VS Code. Это основной навык, который вы будете использовать постоянно при работе над любым проектом.

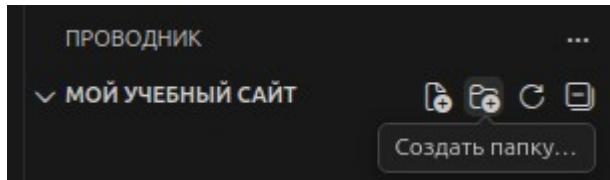
Основные способы создания

Способ 1: Через иконки в Проводнике (самый простой)

Для создания ПАПКИ:

Откройте Проводник (Ctrl + Shift + E)

Наведите курсор на папку, где хотите создать новую папку



Нажмите на иконку «Новая папка» (📁 с плюсом)

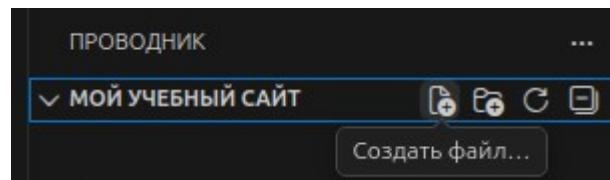
Ведите название папки

Нажмите Enter

Для создания ФАЙЛА:

Откройте Проводник (Ctrl + Shift + E)

Наведите курсор на папку, где хотите создать файл



Нажмите на иконку «Новый файл» (📄 с плюсом)

Ведите название файла с расширением (например, index.html)

Нажмите Enter

Способ 2: Через контекстное меню (правой кнопкой мыши)

Создание папки:

Правой кнопкой мыши на папке в Проводнике

Выберите «New Folder»

Введите название

Нажмите Enter

Создание файла:

Правой кнопкой мыши на папке в Проводнике

Выберите «New File»

Введите название с расширением

Нажмите Enter

Способ 3: Использование горячих клавиш

Быстрое создание файла:

Выделите папку в Проводнике

Нажмите Ctrl + N (новый файл)

Сохраните файл в нужную папку:

Ctrl + S

Выберите папку

Введите название

Нажмите «Сохранить»

Способ 4: Через палитру команд

Нажмите Ctrl + Shift + P

Ведите: File: New File

Нажмите Enter

Сохраните файл в нужное место

Практическое задание: Создаем полную структуру проекта

Давайте создадим профессиональную структуру папок:

Создайте основную папку «Мой сайт»

Внутри создайте папки:

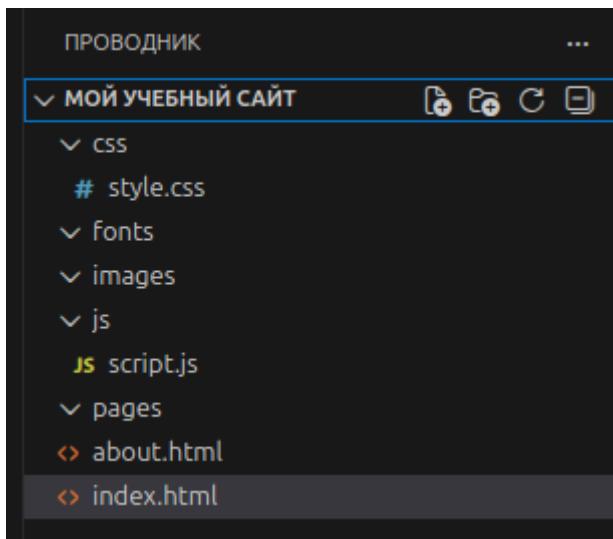
css — для файлов стилей

js — для JavaScript файлов

images — для изображений

fonts — для шрифтов

pages — для дополнительных страниц



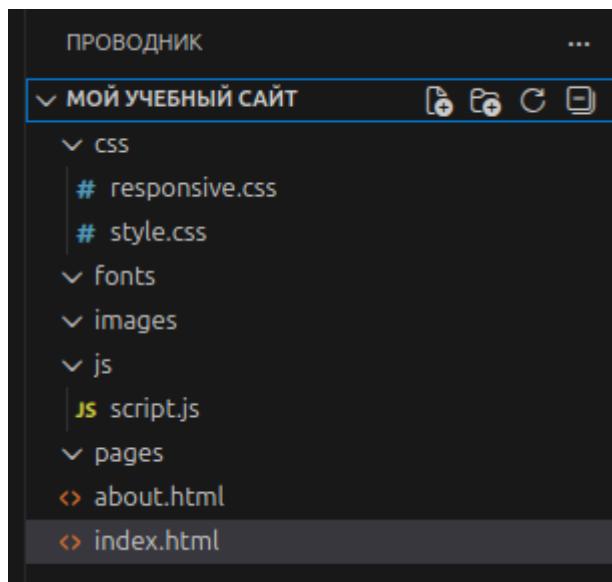
Создайте файлы:

В корне: index.html

В папке css: style.css

В папке css: responsive.css

В папке js: script.js



Результат должен выглядеть так:

text
МОЙ САЙТ/
| └── CSS/
| └── responsive.css
| └── style.css
| └── fonts/
| └── images/



Особенности создания разных типов файлов

HTML файлы:

Название: index.html, about.html, contact.html

Расширение: .html

Автоматически получают подсветку синтаксиса HTML

CSS файлы:

Название: style.css, main.css, layout.css

Расширение: .css

Автоматически получают подсветку синтаксиса CSS

JavaScript файлы:

Название: script.js, main.js, app.js

Расширение: .js

Автоматически получают подсветку синтаксиса JavaScript

Правила именования файлов и папок

Рекомендуется:

Использовать латинские буквы

Использовать нижний регистр (строчные буквы)

Использовать подчеркивание вместо пробелов: my_file.html

Или дефисы: my-file.html

Делать названия понятными и краткими

Не рекомендуется:

Использовать русские буквы

Использовать пробелы

Использовать специальные символы: @, #, \$, %

Делать названия слишком длинными

Полезные хитрости

Быстрое создание нескольких файлов:

Создайте первый файл

Нажмите F2 для переименования
Измените название
Повторите для остальных файлов

Создание файла с готовой структурой:

Для HTML: введите ! и нажмите Tab (работает с расширением Emmet)
Для CSS: сразу начинайте писать стили

Групповое создание файлов:

Создайте папку pages
Сразу создайте в ней все нужные файлы:
about.html
contact.html
gallery.html
services.html

Решение частых проблем

Проблема 1: Не вижу иконки создания файла/папки

Решение: Наведите курсор точно на название папки
Иконки появляются только при наведении

Проблема 2: Файл создается не в той папке

Решение: Внимательно следите, какая папка выделена синим цветом
Именно в ней создастся новый файл/папка

Проблема 3: Неправильное расширение файла

Решение: Всегда указывайте расширение вручную
.html для HTML файлов
.css для CSS файлов
.js для JavaScript файлов

Проблема 4: Файл не сохраняется

Решение: Всегда нажмите Ctrl + S после создания
Или включите автосохранение в настройках

Дополнительные возможности

Создание файлов через терминал:

Откройте терминал в VS Code (`Ctrl + ``)

Введите команды:

```
bash
touch newfile.html
mkdir newfolder
```

Шаблоны файлов:

Создайте файлы-шаблоны с базовой структурой
Копируйте их когда нужно создать новый файл

Практические упражнения

Упражнение 1: Базовая структура

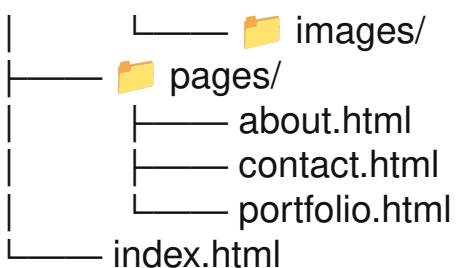
```
text
ПРОЕКТ/
├── css/
├── js/
├── img/
└── index.html
```

Упражнение 2: Расширенная структура

```
text
ИНТЕРНЕТ-МАГАЗИН/
├── css/
├── js/
├── images/
├── fonts/
├── pages/
│   ├── catalog.html
│   ├── product.html
│   └── cart.html
└── index.html
```

Упражнение 3: Многостраничный сайт

```
text
САЙТ-ВИЗИТКА/
├── assets/
│   ├── css/
│   └── js/
```



Проверка знаний

После изучения этого раздела вы должны уметь:

- Создавать папки через иконку в Проводнике
- Создавать файлы через иконку в Проводнике
- Использовать правую кнопку мыши для создания
- Правильно называть файлы и папки
- Создавать сложные структуры проектов

**Теперь вы полностью освоили создание файлов и папок в VS Code!
Этот навык станет основой для всех ваших будущих веб-проектов.
Практикуйтесь регулярно — скоро это будет получаться у вас
автоматически!**

Как переименовывать и удалять файлы

Дорогой друг!

Теперь давайте научимся переименовывать и удалять файлы в VS Code. Это важные навыки, которые помогут вам поддерживать порядок в проекте и исправлять ошибки в названиях файлов.

Переименование файлов и папок

Способ 1: Через медленное двойное нажатие (самый простой)

Для переименования:

Найдите файл или папку в Проводнике

Медленно нажмите два раза левой кнопкой мыши на название

Название выделится синим цветом

Ведите новое название

Нажмите Enter для сохранения

Важно: Нажмайте медленно, с паузой между кликами!

Способ 2: Через правую кнопку мыши

Пошаговая инструкция:

Правой кнопкой мыши на файле или папке

The screenshot shows a file explorer window with a sidebar on the left containing a tree view of files and folders. The main area displays the content of 'index.html'. A context menu is open over the file 'gallery.html' in the list, with several options highlighted in blue:

- Открыть сбоку (CTRL+Enter)
- Открыть с помощью...
- Открыть содержащую папку (CTRL+ALT+R)
- Открыть во встроенным терминале
- Выбрать для сравнения
- Открыть временную шкалу
- Добавить файл в чат
- Вырезать (CTRL+X)
- Копирование (CTRL+C)
- Скопировать путь (CTRL+ALT+C)
- Скопировать относительный путь (CTRL+SHIFT+ALT+C)
- Переименовать... (F2) — this option is highlighted with a blue background
- Удалить (Delete)

Выберите «Rename» из контекстного меню

Ведите новое название

Нажмите Enter

Способ 3: Использование горячих клавиш

Быстрый способ:

Выделите файл или папку в Проводнике

Нажмите клавишу F2

**Введите новое название
Нажмите Enter**

Способ 4: Через палитру команд

Выделите файл в Проводнике

Нажмите Ctrl + Shift + P

Ведите: File: Rename

Нажмите Enter

Введите новое название

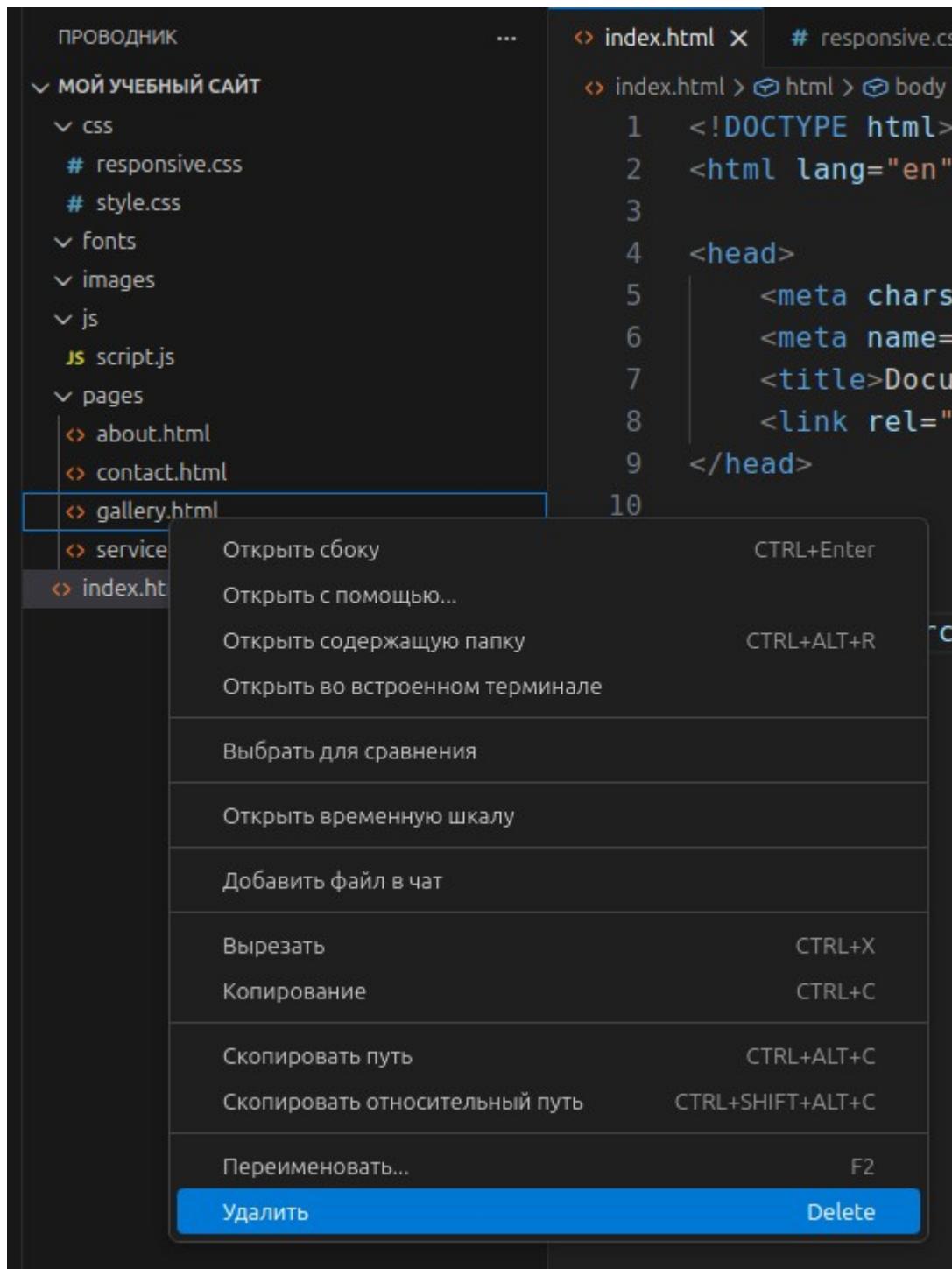
Нажмите Enter

Удаление файлов и папок

Способ 1: Через правую кнопку мыши (рекомендуемый)

Безопасное удаление:

Правой кнопкой мыши на файле или папке



**Выберите «Delete» из меню
Подтвердите удаление в диалоговом окне**

Способ 2: Использование горячих клавиш

**Быстрое удаление:
Выделите файл или папку
Нажмите клавишу Delete
Подтвердите удаление**

Способ 3: Через палитру команд

Выделите файл в Проводнике

Нажмите Ctrl + Shift + P

Ведите: File: Delete

Нажмите Enter

Подтвердите удаление

Практическое задание: Тренировка переименования

Давайте потренируемся на примерах:

Создайте тестовую папку «test_folder»

Создайте тестовый файл «old_name.html»

Переименуйте папку в «practice_folder»

Переименуйте файл в «new_name.html»

Удалите созданные файлы и папки

Особенности переименования

Сохраняем расширения файлов:

При переименовании .html файла сохраняйте расширение

Пример: index.html → main.html (правильно)

Пример: index.html → main (неправильно!)

Переименование с изменением расширения:

Если нужно изменить тип файла, меняйте и расширение

style.css → style.scss (если переходите на SASS)

Безопасное удаление - важные правила

Перед удалением проверяйте:

- Нет ли в файле важного кода
- Не используется ли файл в проекте
- Есть ли резервная копия

Создавайте резервные копии:

Перед удалением скопируйте важный код

Или создайте копию файла с другим названием

Работа с несколькими файлами

Переименование группы файлов:

Выделите несколько файлов (Ctrl + клик)

Нажмите F2

Ведите шаблон имени

Файлы переименуются с добавлением номеров

Удаление группы файлов:

Выделите несколько файлов (Ctrl + клик)

Нажмите Delete

Подтвердите удаление

Практические сценарии

Сценарий 1: Исправление опечатки

text

Было: index.html (с русской "х")

Стало: index.html (с английской "x")

Сценарий 2: Изменение структуры проекта

text

Было: styles.css

Стало: css/main.css (перемещение в папку)

Сценарий 3: Реорганизация изображений

text

Было: photo1.jpg, picture2.png

Стало: images/hero.jpg, images/logo.png

Решение частых проблем

Проблема 1: Не получается переименовать

Причина: Файл открыт в редакторе

Решение: Закройте файл и попробуйте снова

Проблема 2: Файл используется другой программой

Причина: Файл открыт в браузере или другой программе

Решение: Закройте все программы, использующие файл

Проблема 3: Ошибка "Permission denied"

Причина: Нет прав на изменение файла

Решение: Запустите VS Code от имени администратора

Проблема 4: Случайно удалили нужный файл

Решение: Нажмите **Ctrl + Z** сразу после удаления

Или восстановите из корзины Windows

Полезные хитрости

Отмена переименования:

Перед сохранением нажмите **Esc** для отмены

Или **Ctrl + Z** после сохранения

Быстрое перемещение между файлами:

После переименования нажмите **Tab** для перехода к следующему файлу

Массовое переименование:

Используйте расширение "Batch Rename"

Или функцию поиска и замены в коде

Горячие клавиши для работы с файлами

F2 — переименовать выделенный файл/папку

Delete — удалить выделенный файл/папку

Ctrl + Z — отменить последнее действие

Ctrl + Shift + P → "Rename" — переименовать через палитру

Ctrl + Shift + P → "Delete" — удалить через палитру

Практические упражнения

Упражнение 1: Исправление структуры

text

Было:

project/

 └── stylles.css

 └── index.html

 └── imegers/

Стало:

project/

```
├── style.css  
├── index.html  
└── images/
```

Упражнение 2: Реорганизация проекта

text

Было:

site/

```
├── main.css  
├── script.js  
├── logo.jpg  
└── background.png
```

Стало:

site/

```
├── css/  
│   └── style.css  
├── js/  
│   └── main.js  
└── images/  
    ├── logo.jpg  
    └── bg.png
```

Безопасность при удалении

Перед удалением всегда задавайте вопросы:

Этот файл точно не нужен?

Есть ли ссылки на этот файл в других местах?

Не нарушит ли удаление работу сайта?

Есть ли резервная копия?

Создавайте папку "archive":

Вместо удаления перемещайте файлы в папку "archive"

Так вы сможете восстановить их если понадобятся

Проверка знаний

После изучения этого раздела вы должны уметь:

- Переименовывать файлы через двойной клик
- Переименовывать файлы через F2
- Удалять файлы через правую кнопку мыши
- Удалять файлы через Delete
- Исправлять опечатки в названиях файлов
- Безопасно удалять ненужные файлы

Теперь вы освоили переименование и удаление файлов! Эти навыки помогут вам поддерживать порядок в проектах и быстро исправлять ошибки. Помните: аккуратная структура — залог успешного проекта!

Быстрая навигация между файлами проекта

Дорогой друг!

Давайте научимся быстро перемещаться между файлами в вашем проекте. Когда файлов становится много, умение быстро находить и переключаться между ними экономит массу времени и сил!

Основные способы навигации

Способ 1: Использование вкладок (самый простой)

Переключение между открытыми файлами:

Кликните левой кнопкой на нужной вкладке

Используйте горячие клавиши: Ctrl + Tab

Циклическое переключение: Ctrl + Tab (вперед), Ctrl + Shift + Tab (назад)

Порядок вкладок:

text

[index.html] [style.css] [script.js]

Активная вкладка выделена цветом

Точка (•) означает несохраненные изменения

Способ 2: Быстрое открытие файла по имени

Используем быстрый открыватель:

Нажмите Ctrl + P (Command + P на Mac)

Начните вводить название файла

Выберите файл из списка стрелками

Нажмите Enter

Пример:

text

> index.html

> style.css

> script.js

Фильтрация по типу файла:

Введите @ для группировки по символам
Введите # для поиска по CSS-классам
Введите : для перехода к определенной строке

Способ 3: Навигация через Проводник

Использование мыши:

Двойной клик на файле в Проводнике
Одиночный клик для предпросмотра (файл открывается во временной вкладке)

Использование клавиатуры:

Откройте Проводник (Ctrl + Shift + E)
Используйте стрелки для навигации по файлам
Нажмите Enter для открытия файла

Способ 4: Использование хлебных крошек

Навигация по пути файла:

Включите хлебные крошки: View → Show Breadcrumbs
Кликайте на элементах пути для быстрой навигации

Пример:

text
project > src > components > Header > index.html

Горячие клавиши для навигации

Основные комбинации:

Ctrl + P — быстрый поиск файла
Ctrl + G — переход к определенной строке
Ctrl + Tab — переключение между вкладками
Ctrl + 1, 2, 3... — переход к конкретной вкладке
Ctrl + K Ctrl + P — показать активный файл в Проводнике

Навигация между вкладками:

Ctrl + PageUp — предыдущая вкладка
Ctrl + PageDown — следующая вкладка

Ctrl + W — закрыть текущую вкладку
Ctrl + K Ctrl + W — закрыть все вкладки

Продвинутые техники навигации

Работа с группами вкладок

Разделение экрана:

**Ctrl + ** — разделить редактор
Ctrl + 1 — перейти в левую панель
Ctrl + 2 — перейти в правую панель
Ctrl + Shift + P → "View: Split Editor"

Пример разделения:

text
[index.html] [style.css]

[HTML код] [CSS код]

Использование закладок

Установка закладок:

Ctrl + F2 — установить/снять закладку
F2 — перейти к следующей закладке
Shift + F2 — перейти к предыдущей закладке

Просмотр всех закладок:

Ctrl + Shift + P → "View: Show Bookmarks"

Навигация по истории файлов

Возврат к предыдущему файлу:

Ctrl + - (минус) — вернуться назад
Ctrl + Shift + - — перейти вперед
Alt + ← / Alt + → — навигация по истории

Практическое задание

Давайте создадим проект и потренируем навигацию:

Создайте структуру проекта:

```
text
my-project/
    └── index.html
    └── about.html
    └── contact.html
    └── css/
        └── style.css
        └── responsive.css
    └── js/
        └── script.js
```

Потренируйте навигацию:

Откройте все файлы по очереди

Используйте **Ctrl + P** для быстрого переключения

Разделите экран и работайте с HTML и CSS одновременно

Поставьте закладки в важных местах

Использование мини-карты

Быстрая навигация в больших файлах:

Мини-карта отображается справа

Кликните на любую область для быстрого перехода

Используйте для ориентирования в больших файлах

Настройки для удобной навигации

Добавьте в `settings.json`:

```
json
{
    // Показывать хлебные крошки
    "breadcrumbs.enabled": true,
    // Включаем мини-карту
    "editor.minimap.enabled": true,
```

```
// Показывать номер вкладки
"workbench.editor.showTabs": true,  
  
// Ограничить количество открытых вкладок
"workbench.editor.limit.enabled": true,  
"workbench.editor.limit.value": 10,  
  
// Предпросмотр файлов при наведении
"workbench.editor.enablePreview": true  
}
```

Полезные расширения для навигации

File Utils:

Дополнительные команды для работы с файлами
Установка: Ctrl + Shift + X → поиск "File Utils"

Project Manager:

Быстрое переключение между проектами
Сохранение часто используемых проектов

Auto Rename Tag:

Автоматическое переименование парных HTML-тегов
Упрощает навигацию по HTML-файлам

Решение частых проблем

Проблема 1: Слишком много открытых вкладок

Решение: Используйте Ctrl + P вместо поиска глазами
Закрывайте неиспользуемые вкладки Ctrl + W

Проблема 2: Не могу найти нужный файл

Решение: Используйте поиск в Проводнике (Ctrl + F в проводнике)
Или глобальный поиск Ctrl + Shift + F

Проблема 3: Потерялся в большом файле

Решение: Используйте мини-карту справа
Или перейдите к определенной строке Ctrl + G

Проблема 4: Нужно работать с двумя файлами одновременно

Решение: Разделите экран `Ctrl + \`

Или откройте второй экземпляр VS Code

Практические сценарии

Сценарий 1: Быстрое редактирование HTML и CSS

Откройте `index.html` и `style.css`

Разделите экран `Ctrl + \`

Переключайтесь между панелями `Ctrl + 1, Ctrl + 2`

Сценарий 2: Поиск конкретного файла в большом проекте

Нажмите `Ctrl + P`

Ведите первые буквы названия файла

Выберите из списка

Сценарий 3: Навигация в большом HTML-файле

Используйте мини-карту для общего обзора

Переходите к конкретным разделам с помощью закладок

Используйте `Ctrl + F` для поиска внутри файла

Эффективные рабочие привычки

Держите открытыми только нужные вкладки

Используйте горячие клавиши вместо мыши

Организуйте файлы в логичные папки

Используйте понятные названия файлов

Регулярно закрывайте неиспользуемые вкладки

Упражнения для закрепления

Упражнение 1: Слепая навигация

Закройте все вкладки

Используя только `Ctrl + P`, откройте 5 разных файлов

Повторите 3 раза, стараясь ускориться

Упражнение 2: Разделение экрана

Откройте HTML и CSS файл

Разделите экран
Практикуйте одновременную работу

Упражнение 3: Работа с закладками

В большом файле расставьте 3 закладки
Попрактикуйте быстрый переход между ними

Теперь вы освоили быструю навигацию между файлами! Этот навык значительно ускорит вашу работу и сделает процесс разработки более комфортным. Практикуйтесь ежедневно — скоро эти действия станут автоматическими!

Часть 3: Основные возможности для верстки

Глава 6: Интеллектуальное редактирование

Подсветка синтаксиса - как в Notepad++

Дорогой друг!

Давайте разберем, как работает подсветка синтаксиса в VS Code и как её настроить для максимального удобства. Это одна из самых полезных функций, которая превращает обычный текст в цветной, понятный код!

Что такое подсветка синтаксиса?

Подсветка синтаксиса — это раскрашивание разных элементов кода в различные цвета. Как в детской раскраске — каждый тип элемента получает свой цвет, что делает код намного читабельнее.

Автоматическая подсветка синтаксиса

VS Code автоматически определяет тип файла:

Файлы .html → подсветка HTML

Файлы .css → подсветка CSS

Файлы .js → подсветка JavaScript

Файлы .php → подсветка PHP

Файлы .json → подсветка JSON

Как проверить, что подсветка работает

Создайте тестовый HTML-файл:

```
html
<!DOCTYPE html>
<html>
<head>
<title>Мой сайт</title>
<style>
body { color: blue; }
</style>
</head>
<body>
```

```
<h1>Привет, мир!</h1>
<p>Это тестовый текст</p>
</body>
</html>
```

Вы должны увидеть разные цвета:

Теги `<html>`, `<head>`, `<body>` — **синие**
Названия тегов `DOCTYPE`, `title`, `h1` — **синие**
Текст внутри тегов — **светло-синий и пурпурный**
CSS-свойства `color` — **красные**
Значения `blue` — **оранжевые**

Если подсветка не работает

Проверьте следующие моменты:

Правильное расширение файла:

`index.html` (правильно)
`index.htm` (может не работать)
`index.html.txt` (неправильно!)

Язык файла вручную:

Нажмите на язык в правом нижнем углу
Или `Ctrl + K M`
Выберите нужный язык из списка

Перезапустите VS Code

Настройка цветов подсветки

Способ 1: Выбор цветовой темы

Смена темы оформления:

Нажмите `Ctrl + K Ctrl + T`
Выберите тему из списка
Популярные темы:
`Dark+` (`default dark`) — тёмная
`Light+` (`default light`) — светлая
`High Contrast` — для слабого зрения

Monokai — как в Notepad++

Способ 2: Ручная настройка цветов

Через настройки JSON:

Нажмите Ctrl + Shift + P

Ведите: Preferences: Open Settings (JSON)

Добавьте настройки:

```
json
{
    // Цвета для HTML
    "editor.tokenColorCustomizations": {
        "textMateRules": [
            {
                "scope": "entity.name.tag",
                "settings": { "foreground": "#569CD6" }
            },
            {
                "scope": "string.quoted",
                "settings": { "foreground": "#CE9178" }
            }
        ]
    }
}
```

Цвета как в Notepad++

Чтобы сделать подсветку как в Notepad++:

```
json
{
    "workbench.colorTheme": "Dark+",
    "editor.tokenColorCustomizations": {
        "comments": "#57A64A",      // Зеленый для комментариев
        "strings": "#D69D85",      // Коричневый для строк
        "keywords": "#569CD6",      // Синий для ключевых слов
        "variables": "#9CDCFE",      // Голубой для переменных
        "functions": "#DCDCAA",      // Бежевый для функций
        "numbers": "#B5CEA8",      // Светло-зеленый для чисел
        "types": "#4EC9B0"          // Бирюзовый для типов
    }
}
```

```
}
```

Настройка для конкретных языков

HTML подсветка:

```
json
{
  "editor.tokenColorCustomizations": {
    "textMateRules": [
      {
        "scope": "text.html.basic entity.name.tag",
        "settings": { "foreground": "#E06C75" } // Теги
      },
      {
        "scope": "text.html.basic entity.other.attribute-name",
        "settings": { "foreground": "#D19A66" } // Атрибуты
      }
    ]
  }
}
```

CSS подсветка:

```
json
{
  "editor.tokenColorCustomizations": {
    "textMateRules": [
      {
        "scope": "source.css entity.other.attribute-name.class",
        "settings": { "foreground": "#98C379" } // Классы
      },
      {
        "scope": "source.css support.type.property-name",
        "settings": { "foreground": "#61ADEF" } // Свойства
      }
    ]
  }
}
```

Популярные цветовые схемы

Темная тема (рекомендуется):

```
json
{
  "workbench.colorTheme": "Dark+",
  "editor.background": "#1E1E1E",
  "editor.foreground": "#D4D4D4"
}
```

Светлая тема:

```
json
{
  "workbench.colorTheme": "Light+",
  "editor.background": "#FFFFFF",
  "editor.foreground": "#333333"
}
```

Высококонтрастная:

```
json
{
  "workbench.colorTheme": "High Contrast",
  "editor.background": "#000000",
  "editor.foreground": "#FFFFFF"
}
```

Практическое задание

Давайте настроим подсветку под себя:

Создайте файл test.html

Добавьте код:

```
html
<!DOCTYPE html>
<html>
<head>
<title>Тест подсветки</title>
```

```
<style>
.my-class {
color: #ff0000;
font-size: 16px;
}
</style>
</head>
<body>
<div class="my-class">
<p>Текст для теста</p>
</div>
</body>
</html>
```

Поменяйте темы (Ctrl + K Ctrl + T)
Найдите самую удобную для ваших глаз

Дополнительные настройки для удобства

Увеличение контрастности:

```
json
{
  "workbench.colorCustomizations": {
    "editor.selectionBackground": "#264F78",
    "editor.lineHighlightBackground": "#2A2D2E",
    "editorBracketMatch.background": "#515A6B",
    "editorBracketMatch.border": "#FFFF00"
  }
}
```

Для слабого зрения:

```
json
{
  "editor.tokenColorCustomizations": {
    "[High Contrast)": {
      "comments": "#00FF00",
      "strings": "#FFFF00",
      "keywords": "#00FFFF",
      "variables": "#FFFFFF"
    }
  }
}
```

}

Полезные расширения для подсветки

Установка расширений:

Ctrl + Shift + X

Поиск тем:

One Dark Pro — популярная тёмная тема
Material Theme — стиль Material Design
Winter is Coming — мягкая тёмная тема
GitHub Theme — как на GitHub

Решение проблем

Проблема 1: Нет подсветки в файле

Решение: Проверьте расширение файла
Установите язык вручную через панель статуса

Проблема 2: Неправильные цвета

Решение: Сбросьте настройки темы
Или переустановите расширение темы

Проблема 3: Подсветка работает intermittently

Решение: Обновите VS Code
Проверьте конфликтующие расширения

Сравнение с Notepad++

Преимущества VS Code:

Более умная подсветка
Поддержка современных языков
Настраиваемые темы
Интеграция с расширениями

Чтобы сделать как в Notepad++:

Используйте тему Dark+
Настройте яркие контрастные цвета
Увеличьте размер шрифта

Финальные настройки для комфорта

```
json
{
  "workbench.colorTheme": "Dark+",
  "editor.fontSize": 18,
  "editor.lineHeight": 1.5,
  "editor.fontFamily": "Consolas, 'Courier New', monospace",
  "editor.tokenColorCustomizations": {
    "comments": "#6A9955",
    "strings": "#CE9178",
    "keywords": "#569CD6",
    "variables": "#9CDCFE",
    "functions": "#DCDCAA"
  }
}
```

Теперь у вас есть красивая и удобная подсветка синтаксиса!
Поэкспериментируйте с разными настройками и найдите то, что
подходит именно вам. Хорошая подсветка — залог удовольствия от
программирования!

Автодополнение тегов - VS Code подсказывает продолжение

Дорогой друг!

Теперь давайте изучим одну из самых волшебных возможностей VS Code — **автодополнение тегов**. Это как умный помощник, который всегда подскажет, что писать дальше, и поможет избежать ошибок!

Что такое автодополнение?

Автодополнение — это когда VS Code анализирует ваш код и предлагает возможные варианты продолжения. Это работает для:

HTML-тегов

CSS-свойств

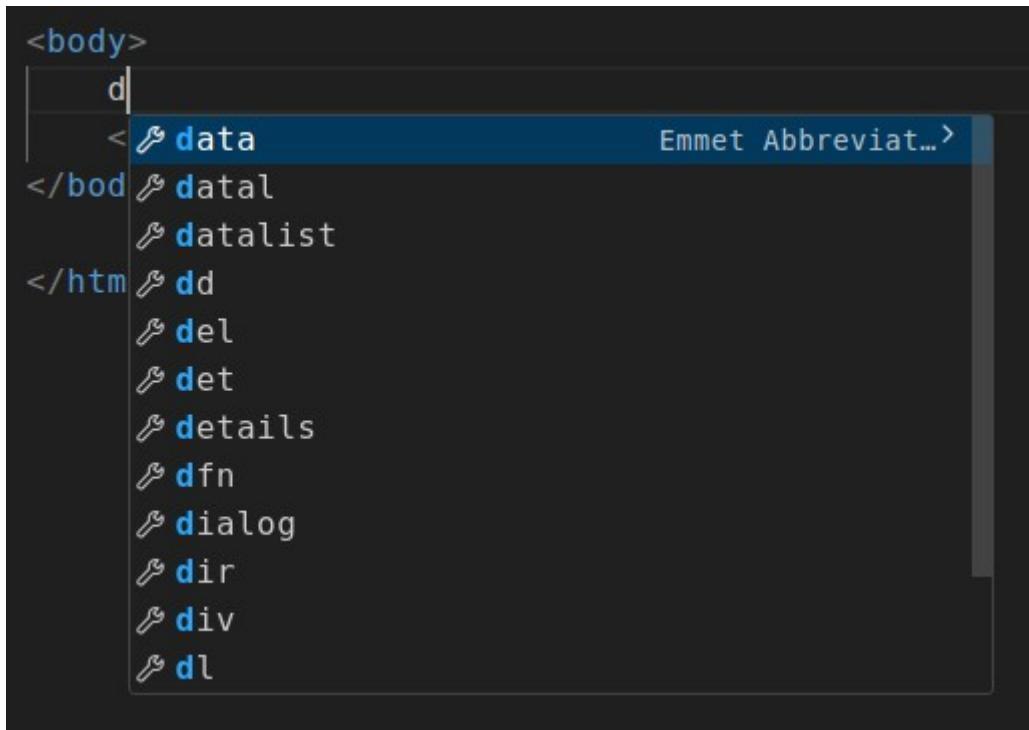
JavaScript-функций

И много другого!

Как работает автодополнение

Когда вы начинаете печатать, появляется окно подсказок:

```
text
<div> ← вы печатаете "d"
▼ div
▼ dl
▼ dt
▼ details
```



Чтобы принять подсказку:

Нажмите Tab или Enter
Или кликните на нужный вариант мышью

Базовые примеры автодополнения

HTML автодополнение:

Пример 1: Создание тега

```
html
<!-- Начните печатать "h1" -->
h1 → <h1></h1> (нажмите Tab)
```

Пример 2: Создание структуры HTML

```
html
<!-- Начните печатать "html" -->
html → <html></html> (нажмите Tab)
```

Пример 3: Автозакрытие тегов

```
html
```

```
<!-- Напечатайте "<div>" -->
<div> → </div> добавляется автоматически!
```

Практическое задание 1

Давайте попрактикуемся с HTML:

Создайте новый файл autocomplete.html

Попробуйте ввести следующие теги:

```
html → нажмите Tab
head → нажмите Tab
title → нажмите Tab
body → нажмите Tab
div → нажмите Tab
```

Результат должен быть таким:

```
html
<html>
<head>
<title></title>
</head>
<body>
<div></div>
</body>
</html>
```

CSS автодополнение

Пример 1: CSS-свойства

```
CSS
/* Начните печатать "col" */
col → color: |; (курсор будет на месте |)
```

Пример 2: Значения свойств

```
CSS
/* После "color: " */
color: → появляются варианты: red, blue, #000 и т.д.
```

Пример 3: Полные правила

css

```
/* Напечатайте "margin" */  
margin: → margin: 10px; (предлагает значения)
```

Практическое задание 2

Потренируем CSS автодополнение:

Создайте файл `style.css`

Попробуйте ввести:

```
color → нажмите Tab  
font-size → нажмите Tab  
margin → нажмите Tab  
background → нажмите Tab
```

Добавьте значения:

```
color: → выберите red из списка  
font-size: → введите 16px  
margin: → выберите 10px
```

Emmet - супер-автодополнение

Emmet — это мощное расширение, встроенное в VS Code, которое позволяет писать код в разы быстрее.

Базовые команды Emmet:

Создание структуры HTML:

```
html  
! → нажмите Tab
```

Результат:

```
html  
<!DOCTYPE html>  
<html lang="en">
```

```
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<body>

</body>
</html>
```

Emmet для быстрой верстки

Создание тегов с классами:

html
div.container → <div class="container"></div>

Создание вложенных структур:

html
ul>li*3 →

Создание сложных структур:

html
div.header+div.main+div.footer →
<div class="header"></div>
<div class="main"></div>
<div class="footer"></div>

Практическое задание 3

Освоим Emmet:

В HTML-файле введите: ! → нажмите Tab
В теле body введите: div.container>h1+p*2 → Tab
Результат:

```
html
<div class="container">
<h1></h1>
<p></p>
<p></p>
</div>
```

Настройка автодополнения

Включение/выключение автодополнения:

Через настройки:

Ctrl + .

Поиск: editor.suggest

Настройте параметры:

```
json
{
    // Показывать подсказки автоматически
    "editor.quickSuggestions": {
        "other": true,
        "comments": false,
        "strings": false
    },
    // Задержка перед показом подсказок (мс)
    "editor.quickSuggestionsDelay": 10,
    // Принимать подсказки на Tab
    "editor.acceptSuggestionOnEnter": "on",
    // Показывать сниппеты в подсказках
    "editor.snippetSuggestions": "inline"
}
```

Полезные горячие клавиши

Ctrl + Space — принудительно показать подсказки

Tab — принять текущую подсказку

Enter — принять подсказку и перейти на новую строку

Ctrl + . — быстрые исправления

F12 — переход к определению

Интеллектуальное автодополнение

VS Code запоминает ваш стиль кода:

Пример:

```
html
<div class="container">
<!-- Если вы часто используете "container" --&gt;
<!-- При вводе "con" будет предлагать "container" первым --&gt;
&lt;/div&gt;</pre>
```

Контекстное автодополнение:

```
CSS
/* В HTML-файле внутри style тега */
<style>
di → предлагает CSS-свойства, а не HTML-теги
</style>
```

Решение проблем с автодополнением

Проблема 1: Не работают подсказки

Решение: Нажмите `Ctrl + Space`

Проверьте настройки `editor.suggest`

Проблема 2: Неправильные подсказки

Решение: Убедитесь, что правильный язык выбран

Проверьте расширение файла

Проблема 3: Emmet не работает

Решение: Убедитесь, что файл имеет правильное расширение

Проверьте настройки Emmet

Расширенные возможности

Сниппеты (заготовки кода)

VS Code включает готовые сниппеты:

HTML сниппеты:

```
ul>li* — список с элементами  
table>tr*3>td*4 — таблица  
form:post — форма с method="post"
```

CSS сниппеты:

```
bg → background: |;  
m10 → margin: 10px;  
df → display: flex;
```

Создание своих сниппетов

Ctrl + Shift + P

Ведите: Preferences: Configure User Snippets

Выберите язык (html, css)

Добавьте свой сниппет:

```
json  
{  
  "My Header": {  
    "prefix": "myheader",  
    "body": [  
      "<header>",  
      "  <h1>$1</h1>",  
      "  <nav>$2</nav>",  
      "</header>"  
    ],  
    "description": "Мой шаблон заголовка"  
  }  
}
```

Практическое задание 4

Создадим свой сниппет:

Откройте настройки сниппетов для HTML

Добавьте сниппет для карточки товара:

```
json  
{  
  "Product Card": {  
    "prefix": "product",
```

```
"body": [
  "<div class=\"product-card\">",
  "  <img src=\"$1\" alt=\"$2\">",
  "  <h3>$3</h3>",
  "  <p>$4</p>",
  "  <span class=\"price\">$5</span>",
  "</div>"
]
}
}
```

Протестируйте: введите product → Tab

Полезные советы

Доверяйте подсказкам — они редко ошибаются

Используйте Tab для принятия — это быстрее

Изучите Emmet — это ускорит верстку в 10 раз

Создавайте свои снippetsы для часто используемого кода

Не бойтесь экспериментировать — автодополнение поможет

Чек-лист освоения автодополнения

Умею принимать подсказки Tab/Enter

Знаю основные Emmet-аббревиатуры

Умею создавать структуру HTML командой !

Понимаю, как работают CSS-подсказки

Могу создавать свои снippetsы

Умею вызывать подсказки принудительно (Ctrl + Space)

Теперь вы владеете мощным инструментом автодополнения! Это значительно ускорит вашу работу и сделает процесс программирования более приятным. Практикуйтесь ежедневно, и скоро вы будете удивляться, как раньше жили без этих подсказок!

Автозакрытие тегов - программа сама добавляет закрывающие теги

Дорогой друг!

Теперь давайте изучим одну из самых удобных функций VS Code — **автозакрытие тегов**. Это волшебная возможность, которая избавляет вас от необходимости вручную печатать закрывающие теги и следить за их правильностью!

Что такое автозакрытие тегов?

Автозакрытие тегов — это когда VS Code автоматически добавляет закрывающий тег сразу после того, как вы напечатали открывающий тег.

Это работает для:

HTML-тегов

XML-тегов

JSX (для React)

И других языков разметки

Как работает автозакрытие

Базовый пример:

html

<!-- Вы печатаете: -->
<div>

<!-- VS Code автоматически добавляет: -->
<div></div>

<!-- Курсор оказывается между тегами: -->
<div>|</div>

Практические примеры автозакрытия

Пример 1: Простые теги

html

<!-- Вы вводите: -->
<p> → становится <p>|</p>

<h1> → становится <h1>|</h1>

 → становится |

Пример 2: Вложенные теги

html

<!-- Вы вводите: -->

<div>

<p>

<!-- Становится: -->

<div>

<p>|</p>

</div>

Пример 3: Теги с атрибутами

html

<!-- Вы вводите: -->

<div class="container">

<!-- Становится: -->

<div class="container">|</div>

<!-- Или с переносом строк: -->

<div class="container">

|

</div>

Настройка автозакрытия тегов

Включение/выключение функции:

Через настройки JSON:

json

{

// Автоматическое закрытие HTML-тегов

"html.autoClosingTags": true,

```
// Автоматическое закрытие скобок
"editor.autoClosingBrackets": "always",

// Автоматическое закрытие кавычек
"editor.autoClosingQuotes": "always",

// Автоматическое закрытие в стиле XML
"editor.autoClosingOvertype": "auto"
}
```

Через графический интерфейс:

Нажмите **Ctrl +**,
В поиске введите **auto closing**
Настройте нужные параметры

Практическое задание 1

Давайте попрактикуем автозакрытие:

Создайте файл autoclose.html

Попробуйте ввести следующие теги:

```
<html>
<head>
<title>
<body>
<div>
<p>
<span>
```

Обратите внимание:

Закрывающий тег добавляется автоматически
Курсор ставится между тегами
Вы сразу можете начинать писать контент

Особые случаи автозакрытия

Самозакрывающиеся теги:

html

```
<!-- Эти теги закрываются автоматически: -->  
<br> → <br> (не становится <br></br>)  
<hr> → <hr>  
<img> → <img>  
<meta> → <meta>  
<link> → <link>
```

Теги с атрибутами:

html

```
<!-- Автозакрытие работает с атрибутами: -->  
<a href="#"> → <a href="#">|</a>
```

```
 → 
```

```
<input type="text"> → <input type="text">
```

Работа с курсором после автозакрытия

После автозакрытия тега:

Курсор автоматически помещается между тегами
Вы можете сразу начинать ввод контента
При нажатии Enter создается перенос строк

Пример:

```
html  
<div>  
| ← курсор здесь  
</div>
```

Практическое задание 2

Создадим структуру страницы с автозакрытием:

Ведите: `<html>`

Нажмите Enter — создастся структура:

```
html  
<html>  
|  
</html>
```

Ведите: `<head>` внутри html
Ведите: `<title>` внутри head
Добавьте: `<body>` после head

Результат:

```
html
<html>
<head>
<title>|</title>
</head>
<body>
|
</body>
</html>
```

Автозакрытие для разных языков

HTML:

```
html
<div> → <div>|</div>
<p> → <p>|</p>
<span> → <span>|</span>
```

CSS (для фигурных скобок):

```
css
.container { → .container { | }
```

JavaScript (для скобок и кавычек):

```
javascript
function test() { → function test() { | }
```

```
let name = "" → let name = "|"
```

Продвинутые возможности

Умное автозакрытие:

VS Code понимает контекст:

html

```
<!-- Если вы начинаете новый тег внутри существующего: -->
<div>
<p> →
<div>
<p>|</p>
</div>
```

Автозакрытие при редактировании:

Если вы редактируете открывающий тег:

html

```
<!-- Было: -->
<div>content</div>

<!-- Вы меняете <div> на <section>: -->
<section>content</div>

<!-- VS Code может предложить изменить закрывающий тег -->
```

Настройка поведения автозакрытия

Тонкая настройка в `settings.json`:

```
json
{
  // Когда включать автозакрытие тегов
  "html.autoClosingTags": true,

  // Стиль форматирования при автозакрытии
  "html.format.wrapLineLength": 120,
  "html.format.unformatted": "span,em,strong",

  // Автозакрытие для других языков
  "javascript.autoClosingTags": true,
  "typescript.autoClosingTags": true,
  "xml.autoClosingTags": true
}
```

Решение проблем

Проблема 1: Автозакрытие не работает

Решение:

Проверьте настройку "html.autoClosingTags": true

Убедитесь, что файл имеет расширение .html

Перезапустите VS Code

Проблема 2: Лишние закрывающие теги

Решение:

Это может произойти если вы быстро печатаете

Удалите лишний тег и продолжайте работу

Используйте Ctrl + Z для отмены

Проблема 3: Неправильное автозакрытие

Решение:

Проверьте синтаксис тега

Убедитесь, что нет незакрытых кавычек

Закройте и снова откройте файл

Практическое задание 3

Создадим карточку товара с автозакрытием:

Ведите: <div class="product-card">

Добавьте внутри:

```
<img src="" alt="">
<h3></h3>
<p></p>
<span class="price"></span>
```

Результат:

```
html
<div class="product-card">
<img src="" alt="">
<h3></h3>
<p></p>
<span class="price"></span>
</div>
```

Полезные советы

Доверяйте автозакрытию — это избавляет от многих ошибок
Следите за курсором — он показывает где вы сейчас редактируете
Используйте Tab для перехода между тегами
Не бойтесь удалять лишнее — VS Code поможет восстановить
Практикуйтесь — скоро это станет естественным

Горячие клавиши для работы с тегами

Ctrl + Shift + A — выделить тег и его содержимое
Alt + Shift + W — обернуть выделение в тег
Ctrl + Shift + . — перейти к следующему тегу
Ctrl + Shift + , — перейти к предыдущему тегу

Чек-лист освоения автозакрытия

Понимаю, как работает автозакрытие тегов
Умею создавать базовую HTML-структуру
Знаю, какие теги самозакрываются
Умею работать с тегами с атрибутами
Могу настроить параметры автозакрытия
Знаю, как решать common проблемы

Поздравляю! Теперь вы освоили автозакрытие тегов — одну из самых удобных функций VS Code. Это значительно ускорит вашу верстку и избавит от многих синтаксических ошибок. Продолжайте практиковаться, и скоро вы будете удивляться, как раньше обходились без этой функции!

Глава 7: Работа с вкладками и разделение экрана

Как открыть несколько файлов одновременно

Дорогой друг!

Давайте научимся работать с несколькими файлами одновременно в VS Code. Это важный навык, который позволит вам эффективно работать над сложными проектами и легко переключаться между разными частями кода.

Способы открытия нескольких файлов

Способ 1: Открытие через Проводник (самый простой)

Пошаговая инструкция:

Откройте Проводник (Ctrl + Shift + E)

Найдите нужные файлы в структуре проекта

Кликните на файлы с зажатой клавишей Ctrl

Нажмите Enter или двойной клик

Пример:

```
text
МОЙ ПРОЕКТ/
├── images
├── index.html [Ctrl + клик]
├── style.css [Ctrl + клик]
└── script.js [Ctrl + клик]
```

Способ 2: Открытие через быстрый выбор

Использование Ctrl+P:

Нажмите Ctrl + P

Вводите названия файлов по очереди

Выбирайте файлы из списка

Пример:

```
text
> index.html [Enter]
```

```
> style.css [Enter]  
> script.js [Enter]
```

Способ 3: Открытие всех файлов из папки

Массовое открытие:

В Проводнике выделите папку
Правой кнопкой → "Open in Editor"
Или используйте расширения для массового открытия

Управление открытыми файлами

Панель вкладок

Внешний вид панели вкладок:

text
[index.html] [style.css] [script.js] [x]

Активная вкладка — выделена цветом
Неактивные вкладки — серые или прозрачные
Точка (•) — есть несохраненные изменения
Крестик (x) — закрыть вкладку

Горячие клавиши для переключения

Ctrl + Tab — переключение вперед
Ctrl + Shift + Tab — переключение назад
Ctrl + 1, 2, 3... — переход к конкретной вкладке
Ctrl + K Ctrl + → — следующая группа вкладок
Ctrl + K Ctrl + ← — предыдущая группа вкладок

Практическое задание 1

Давайте откроем несколько файлов:

Создайте проект с файлами:

text
my-project/
|—— index.html
|—— style.css

```
├── script.js  
└── about.html
```

Откройте все файлы одновременно:

Через Проводник с зажатым Ctrl
Или через Ctrl+P по очереди

Попрактикуйте переключение:

Ctrl + Tab — вперед
Ctrl + Shift + Tab — назад
Ctrl + 1 — первая вкладка

Организация работы с несколькими файлами

Группировка вкладок

Перетаскивание вкладок:

Захватите вкладку мышью
Перетащите в нужное место
Создайте группы вкладок

Пример группировки:

```
text  
[HTML] [CSS]  
index.html style.css  
about.html layout.css  
contact.html
```

Закрепление вкладок

Закрепите важные файлы:

Правой кнопкой на вкладке
Выберите "Pin"
Закрепленная вкладка останется слева

Пример:

```
text
```

[index.html] [style.css] [script.js] [about.html]

↑ закреплен ↑ закреплен

Разделение экрана

Вертикальное разделение:

Правой кнопкой на вкладке

Выберите "Split Right"

Или нажмите **Ctrl + **

Результат:

text

[index.html] [style.css]

[HTML] [CSS]

[код] [код]

Горизонтальное разделение:

**Ctrl + K Ctrl + **

Или через меню: View → Editor Layout

Практическое задание 2

Потренируем разделение экрана:

Откройте index.html и style.css

Разделите экран **Ctrl + **

Расположите:

Слева — HTML

Справа — CSS

Переключайтесь между панелями: **Ctrl + 1, Ctrl + 2**

Работа с группами вкладок

Создание нескольких групп:

Откройте несколько файлов

Перетащите вкладки чтобы создать группы

Используйте **Ctrl + K Ctrl + →** для навигации

Пример сложной организации:

```
text
[Группа 1] [Группа 2] [Группа 3]
index.html style.css script.js
about.html layout.css utils.js
```

Настройки для работы с несколькими файлами

Добавьте в settings.json:

```
json
{
    // Максимальное количество открытых вкладок
    "workbench.editor.limit.enabled": true,
    "workbench.editor.limit.value": 10,

    // Закрывать вкладки при достижении лимита
    "workbench.editor.limit.perEditorGroup": false,

    // Показывать предпросмотр файлов
    "workbench.editor.enablePreview": true,

    // Закрывать предпросмотр при открытии нового файла
    "workbench.editor.enablePreviewFromQuickOpen": true,

    // Расположение новых вкладок
    "workbench.editor.openPositioning": "right"
}
```

Полезные расширения

File Utils:

Массовое открытие файлов
Расширенные операции с файлами

Установка:

Ctrl + Shift + X

Поиск: "File Utils"

Установите расширение

Project Manager:

Быстрое переключение между проектами
Сохранение наборов файлов

Решение проблем

Проблема 1: Слишком много открытых вкладок

Решение:

Используйте `Ctrl + W` для закрытия ненужных

Настройте лимит вкладок в настройках

Используйте `Ctrl + P` вместо хранения всех вкладок открытыми

Проблема 2: Потерялся в большом количестве вкладок

Решение:

Используйте `Ctrl + Shift + E` для перехода к файлу в проводнике

Закрепляйте важные вкладки

Группируйте связанные файлы

Проблема 3: Файлы открываются в неудобных местах

Решение:

Настройте "workbench.editor.openPositioning"

Используйте перетаскивание для организации

Создавайте шаблоны рабочих пространств

Практическое задание 3

Создадим организованное рабочее пространство:

Откройте файлы:

`index.html, about.html, contact.html`

`style.css, layout.css`

`script.js`

Организуйте вкладки:

HTML-файлы — в одну группу

CSS-файлы — в другую группу

JavaScript — в третью группу

Разделите экран для одновременной работы

Продвинутые техники

Рабочие пространства (Workspaces):

Сохранение текущего набора файлов:

`File → Save Workspace As...`

**Дайте имя рабочему пространству
При следующем запуске — откроются те же файлы**

Сессии (Sessions):

VS Code запоминает открытые файлы и автоматически восстанавливает их при следующем запуске.

Горячие клавиши для продвинутых пользователей

Ctrl + K S — сохранить все файлы

Ctrl + K W — закрыть все файлы

Ctrl + K Ctrl + W — закрыть все группы

Ctrl + Shift + T — открыть последний закрытый файл

Ctrl + K E — показать explorer

Чек-лист эффективной работы

Умею открывать несколько файлов через Проводник

Использую Ctrl+P для быстрого доступа

Организую вкладки в логичные группы

Использую разделение экрана для одновременной работы

Закрепляю важные вкладки

Закрываю ненужные вкладки для чистоты workspace

Практический сценарий: Верстка страницы

Эффективный workflow:

Откройте index.html и style.css

Разделите экран вертикально

Закрепите обе вкладки

Работайте одновременно с HTML и CSS

Используйте Ctrl + Tab для быстрого переключения

Теперь вы мастер работы с несколькими файлами! Этот навык сделает вашу работу над проектами намного эффективнее и организованнее. Практикуйтесь регулярно, и скоро вы будете интуитивно управлять десятками открытых файлов без всяких усилий!

Разделение экрана для работы с HTML и CSS одновременно

Дорогой друг!

Давайте научимся разделять экран в VS Code для одновременной работы с HTML и CSS. Это одна из самых полезных возможностей для веб-разработчиков, которая значительно ускоряет процесс верстки!

Зачем нужно разделение экрана?

Преимущества одновременной работы:

Видите изменения в CSS сразу при редактировании HTML
Можете копировать классы и идентификаторы между файлами
Не нужно постоянно переключаться между вкладками
Повышает продуктивность на 50-70%

Способы разделения экрана

Способ 1: Горячие клавиши (самый быстрый)

Вертикальное разделение:

`Ctrl + \` — разделить редактор вертикально

Горизонтальное разделение:

`Ctrl + K Ctrl + \` — разделить редактор горизонтально

Способ 2: Через меню

Откройте меню "View"

Выберите "Editor Layout"

Выберите вариант разделения:

Split Up (разделить вверх)

Split Down (разделить вниз)

Split Left (разделить влево)

Split Right (разделить вправо)

Способ 3: Перетаскиванием вкладок

Захватите вкладку файла мышью

Перетащите в нужную область экрана

Появятся синие области — отпустите в нужной

Практическое задание 1: Базовое разделение

Давайте создадим базовое разделение экрана:

Откройте файлы: index.html и style.css

Нажмите Ctrl + \

Результат:

text

[index.html] [style.css]

[HTML код] [CSS код]

[] []

Настройка разделенного workspace

Оптимальная настройка для верстки:

json

{

// Размер панелей при разделении

"workbench.editor.splitSizing": "distribute",

// Расположение новых редакторов

"workbench.editor.openPositioning": "right",

// Закрывать панели при закрытии последней вкладки

"workbench.editor.closeEmptyGroups": true,

// Максимальное количество панелей

"workbench.editor.limit.enabled": false

}

Практическое задание 2: Создаем рабочее пространство

Оптимальная настройка для HTML/CSS:

Откройте index.html

Нажмите Ctrl + \ **для разделения**

В правой панели откройте style.css

Настройте размеры перетаскиванием границы

Итоговый вид:

text

[index.html] [style.css]

```
[ <!DOCTYPE> ][ body { } ]
[ <html> ][ margin: ]
[ <head> ][ padding: ]
[ <title> ][ {} ]
[ </head> ][ .container ]
[ <body> ][ { } ]
[ <div> ][ width: ]
[ </body> ][ {} ]
[ </html> ][ ]
```

Навигация между панелями

Горячие клавиши для навигации:

Ctrl + 1 — перейти в левую панель
Ctrl + 2 — перейти в правую панель
Ctrl + 3 — перейти в третью панель
Ctrl + K Ctrl + ← — предыдущая панель
Ctrl + K Ctrl + → — следующая панель

Мышь:

Просто кликните в нужную панель
Курсор перемещается туда, где вы кликаете

Практическое задание 3: Эффективная навигация

Потренируем переключение между панелями:

Создайте разделение с HTML и CSS

Переключайтесь: Ctrl + 1 → Ctrl + 2

Попрактикуйте быстрый набор:

Наберите в HTML <div class="container">

Ctrl + 2 — перейдите в CSS

Наберите .container { }

Ctrl + 1 — вернитесь в HTML

Продвинутые техники работы

Копирование между панелями:

Эффективный workflow:

В HTML создайте класс: class="header"

Выделите "header" → Ctrl + C

Перейдите в CSS: Ctrl + 2

Вставьте: Ctrl + V

Добавьте стили: .header { color: blue; }

Синхронная прокрутка:

Включите синхронную прокрутку:

Правой кнопкой на номер строки

Выберите "Scroll Now"

Или используйте расширения для синхронизации

Группировка связанных файлов:

Создайте логические группы:

text

[Группа HTML] [Группа CSS]

index.html style.css

about.html layout.css

contact.html components.css

Практическое задание 4: Реальная верстка

Создадим карточку товара вместе:

В HTML панели создайте структуру:

html

```
<div class="product-card">

<h3 class="product-title">Название товара</h3>
<p class="product-description">Описание товара</p>
<span class="product-price">1000 руб.</span>
</div>
```

В CSS панели добавьте стили:

```
css
.product-card {
border: 1px solid #ddd;
padding: 20px;
margin: 10px;
}
```

```
.product-title {
font-size: 18px;
color: #333;
}
```

```
.product-price {
font-weight: bold;
color: #e44d26;
}
```

Настройки для комфортной работы

Оптимальные настройки для разделенного экрана:

```
json
{
// Размер шрифта
"editor.fontSize": 16,

// Межстрочный интервал
"editor.lineHeight": 1.5,

// Включить перенос строк
"editor.wordWrap": "on",

// Показывать мини-карту
"editor.minimap.enabled": true,

// Цветовая тема
"workbench.colorTheme": "Default Dark+",

// Размер панелей
"workbench.panel.defaultLocation": "right"
}
```

Решения частых проблем

Проблема 1: Не хватает места на экране

Решение:

Используйте **Ctrl + B** чтобы скрыть боковую панель

Уменьшите размер шрифта: **Ctrl + -**

Используйте горизонтальное разделение вместо вертикального

Проблема 2: Слишком много панелей

Решение:

Закройте лишние панели: **Ctrl + W**

Объедините связанные файлы в группы

Используйте **Ctrl + P** для быстрого переключения

Проблема 3: Теряю фокус между панелями

Решение:

Используйте **Ctrl + 1**, **Ctrl + 2** для точного переключения

Настройте цвет активной панели в темах

Практикуйте слепой метод переключения

Полезные расширения

Auto Rename Tag:

Автоматически переименовывает парные HTML-теги

Очень полезно при работе с двумя панелями

Color Highlight:

Подсвечивает цвета в CSS

Видно сразу в обеих панелях

Live Server:

Запускает локальный сервер

Автоматическое обновление страницы при изменениях

Практическое задание 5: Полный workflow

Давайте создадим полноценную страницу:

Настройте разделение экрана

В левой панели - HTML:

```
html
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="style.css">
</head>
<body>
<header class="header">
<nav>Навигация</nav>
</header>
<main class="main">
<section class="hero">
<h1>Заголовок</h1>
</section>
</main>
</body>
</html>
```

В правой панели - CSS:

css

```
.header {
background: #333;
color: white;
padding: 1rem;
}
```

```
.hero {
text-align: center;
padding: 2rem;
}
```

Переключайтесь между панелями и добавляйте контент

Горячие клавиши для максимальной эффективности

Ctrl + \ — разделить экран

Ctrl + 1 / Ctrl + 2 — переключение панелей

Ctrl + S — сохранить файл в активной панели

Ctrl + Tab — переключение между файлами в панели

Ctrl + B — скрыть/показать боковую панель

Чек-лист освоения разделения экрана

Умею создавать вертикальное разделение (Ctrl + \)

Могу создавать горизонтальное разделение (Ctrl + K Ctrl + \)

Умею переключаться между панелями (Ctrl + 1, Ctrl + 2)

Могу эффективно копировать код между панелями

Настроил оптимальное расположение для верстки

Использую разделение экрана в ежедневной работе

Поздравляю! Теперь вы освоили разделение экрана — мощный инструмент для эффективной верстки. Эта техника сделает вашу работу намного быстрее и приятнее. Практикуйтесь ежедневно, и скоро вы будете удивляться, как раньше работали без этого!

Горячие клавиши для переключения между вкладками

Дорогой друг!

Давайте подробно изучим все горячие клавиши для переключения между вкладками в VS Code. Это один из самых важных навыков для эффективной работы - он сэкономит вам массу времени и сделает процесс программирования более плавным!

Базовые горячие клавиши переключения

Основные комбинации:

Ctrl + Tab — переключение вперед (следующая вкладка)

Ctrl + Shift + Tab — переключение назад (предыдущая вкладка)

Ctrl + PageUp — предыдущая вкладка (альтернатива)

Ctrl + PageDown — следующая вкладка (альтернатива)

Как работает переключение:

Пример с тремя вкладками:

text
[HTML] [CSS] [JavaScript]
↑ ↑ ↑
| | |
Tab Tab Tab
| | |
Ctrl+Tab движется в этом направлении

Прямой переход к конкретной вкладке

Переход по номерам:

Ctrl + 1 — перейти к первой вкладке

Ctrl + 2 — перейти ко второй вкладке

Ctrl + 3 — перейти к третьей вкладке

Ctrl + 4 — перейти к четвертой вкладке

Ctrl + 5 — перейти к пятой вкладке

Ctrl + 6 — перейти к шестой вкладке

Ctrl + 7 — перейти к седьмой вкладке

Ctrl + 8 — перейти к восьмой вкладке

Ctrl + 9 — перейти к последней вкладке

Практическое задание 1

Давайте потренируем базовое переключение:

Откройте 3 файла: index.html, style.css, script.js

Попрактикуйте:

Ctrl + Tab — вперед по кругу

Ctrl + Shift + Tab — назад по кругу

Ctrl + 1 → Ctrl + 2 → Ctrl + 3 — прямой переход

Переключение между группами вкладок

Когда у вас разделенный экран:

text

[Группа 1] [Группа 2]

index.html style.css

about.html layout.css

Ctrl + K Ctrl + ← — перейти к предыдущей группе

Ctrl + K Ctrl + → — перейти к следующей группе

Быстрое переключение между двумя файлами

Метод "Последний файл":

Ctrl + Tab (быстрое нажатие) — переключаться между текущим и

предыдущим файлом

Практический пример:

Вы работаете с index.html и style.css

Ctrl + Tab быстро переключает между ними

Не нужно проходить через все вкладки

Практическое задание 2

Освоим быстрое переключение между двумя файлами:

Откройте только index.html и style.css

Попрактикуйте быстрое нажатие Ctrl + Tab

Обратите внимание — переключается только между этими двумя файлами

Навигация с помощью мыши

Альтернативные способы:

Клик левой кнопкой по вкладке — прямой переход

Колесико мыши над панелью вкладок — прокрутка вкладок

Ctrl + K Ctrl + E — показать проводник и выделить текущий файл

Горячие клавиши для работы с вкладками

Управление вкладками:

Ctrl + W — закрыть текущую вкладку

Ctrl + K W — закрыть все вкладки в группе

Ctrl + K Ctrl + W — закрыть все вкладки во всех группах

Ctrl + Shift + T — открыть последнюю закрытую вкладку

Ctrl + K P — копировать путь к файлу

Практическое задание 3

Потренируем управление вкладками:

Откройте 4-5 файлов

Попрактикуйте:

Ctrl + W — закрыть текущую вкладку

Ctrl + Shift + T — вернуть закрытую вкладку

Ctrl + K W — закрыть все вкладки в группе

Продвинутые техники навигации

Использование Ctrl+P для навигации:

Ctrl + P — быстрый поиск файла по имени

Ведите название файла — мгновенный переход

Не нужно запоминать номера вкладок!

Пример:

text

> index.html [Enter] — перейти к index.html

> style.css [Enter] — перейти к style.css

Настройки для удобной навигации

Оптимальные настройки:

```
json
{
    // Показывать вкладки
    "workbench.editor.showTabs": true,
    // Максимальное количество вкладок
    "workbench.editor.limit.enabled": true,
    "workbench.editor.limit.value": 8,
    // Закрывать вкладки при достижении лимита
    "workbench.editor.limit.perEditorGroup": true,
    // Показывать предпросмотр файлов
    "workbench.editor.enablePreview": true,
    // Расположение новых вкладок
    "workbench.editor.openPositioning": "right"
}
```

Практическое задание 4

Создадим эффективную систему навигации:

Откройте 6 файлов:

index.html, about.html, contact.html

style.css, layout.css, script.js

Настройте лимит вкладок на 6

Попрактикуйте разные методы навигации

Найдите самый удобный для себя способ

Решения проблем с навигацией

Проблема 1: Слишком много вкладок

Решение:

Используйте Ctrl + P вместо переключения между вкладками

Настройте лимит вкладок

Закрывайте неиспользуемые вкладки Ctrl + W

Проблема 2: Не могу найти нужную вкладку

Решение:

Используйте **Ctrl + P** для поиска по имени

Группируйте связанные файлы

Используйте закрепление важных вкладок

Проблема 3: Потерялся в группах вкладок

Решение:

Используйте **Ctrl + 1, Ctrl + 2** для прямого перехода

Ctrl + K Ctrl + ←/→ для навигации между группами

Упростите структуру групп

Полезные хитрости

Закрепление важных вкладок:

Правой кнопкой на вкладке

Выберите "Pin"

Закрепленная вкладка остается слева и не закрывается

Быстрое меню вкладок:

Ctrl + Shift + Tab (удерживать) — показать меню всех вкладок

Выберите вкладку стрелками и Enter

Горячие клавиши для разных операционных систем

Windows:

Ctrl + Tab — следующая вкладка

Ctrl + Shift + Tab — предыдущая вкладка

Ctrl + W — закрыть вкладку

Mac:

Cmd + Tab — следующая вкладка

Cmd + Shift + Tab — предыдущая вкладка

Cmd + W — закрыть вкладку

Linux:

Ctrl + Tab — следующая вкладка

Ctrl + Shift + Tab — предыдущая вкладка

Ctrl + W — закрыть вкладку

Практическое задание 5

Реальный сценарий работы:

Откройте: index.html, style.css, script.js

Разделите экран: Ctrl + \

В левой панели: HTML файлы

В правой панели: CSS и JS файлы

Практикуйте навигацию:

Ctrl + 1 / Ctrl + 2 — между панелями

Ctrl + Tab — внутри панели

Ctrl + P — быстрый поиск любого файла

Чек-лист освоения горячих клавиш

Ctrl + Tab — следующая вкладка

Ctrl + Shift + Tab — предыдущая вкладка

Ctrl + 1-9 — переход к конкретной вкладке

Ctrl + W — закрыть вкладку

Ctrl + Shift + T — вернуть закрытую вкладку

Ctrl + P — быстрый поиск файла

Ctrl + K Ctrl + ←/→ — навигация между группами

Поздравляю! Теперь вы владеете всеми необходимыми горячими клавишами для эффективной навигации между вкладками.

Практикуйтесь ежедневно, и скоро эти действия станут автоматическими - вы будете переключаться между файлами быстрее, чем думать об этом!

Глава 8: Поиск и замена

Простой поиск в одном файле

Дорогой друг!

Давайте подробно изучим простой поиск в одном файле — это одна из самых часто используемых функций, которая поможет вам быстро находить нужные участки кода в больших файлах.

Базовый поиск в файле

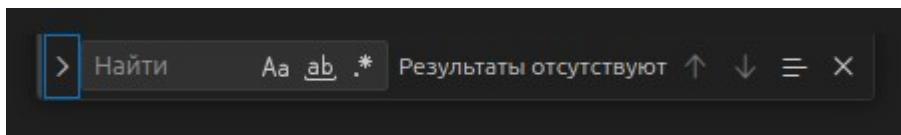
Основные горячие клавиши:

Ctrl + F — открыть панель поиска в текущем файле

F3 — найти следующее совпадение

Shift + F3 — найти предыдущее совпадение

Как выглядит панель поиска:



Практическое задание 1

Давайте попрактикуем базовый поиск:

Откройте файл index.html с таким содержимым:

```
html
<!DOCTYPE html>
<html>
<head>
<title>Мой сайт</title>
</head>
<body>
<div class="container">
<h1>Заголовок страницы</h1>
<p>Текст параграфа</p>
<div class="content">
<p>Еще один параграф</p>
</div>
</div>
```

```
</body>  
</html>
```

Нажмите **Ctrl + F**

Ведите **div**

Наблюдайте — все **div** подсветятся в файле

Навигация по результатам поиска

После выполнения поиска:

Enter — перейти к следующему совпадению

Shift + Enter — перейти к предыдущему совпадению

F3 — следующее совпадение

Shift + F3 — предыдущее совпадение

Esc — закрыть панель поиска

Индикатор результатов:

text

[2 / 8] — означает "2-е совпадение из 8"

Практическое задание 2

Потренируем навигацию по результатам:

В том же файле найдите **p**

Используйте:

Enter — перейти к следующему **<p>**

Shift + Enter — вернуться к предыдущему

F3 — следующее совпадение

Обратите внимание на индикатор [1/3], [2/3], [3/3]

Опции поиска (флажки)

Три основные опции:

Match Case — учитывать регистр (Aa)

Whole Word — целое слово

Regex — регулярные выражения

Match Case (учет регистра):

Без учета регистра:

Поиск div найдет: div, Div, DIV

С учетом регистра:

Поиск div найдет только: div

Whole Word (целое слово):

Без Whole Word:

Поиск div найдет: div, divid, individual

С Whole Word:

Поиск div найдет только: div

Практическое задание 3

Поэкспериментируем с опциями:

Найдите class в файле

Попробуйте с разными опциями:

Без опций — найдет все class

C Match Case — только class (не Class, CLASS)

C Whole Word — только отдельное слово class

Быстрый поиск без панели

Инкрементальный поиск:

Ctrl + F → начинайте печатать — поиск происходит сразу

Ctrl + G — перейти к определенной строке

Выделение всех совпадений:

Alt + Enter — выделить все найденные совпадения

Можно редактировать все выделенные области одновременно!

Практическое задание 4

Освоим выделение всех совпадений:

Найдите div в файле

Нажмите Alt + Enter

Все div будут выделены

Попробуйте напечатать section — все выделенные div заменятся на section

Поиск и выделение

Выделение совпадений при поиске:

Когда вы используете поиск, все совпадения автоматически подсвечиваются в файле. Вы можете:

Видеть все совпадения сразу

Быстро переключаться между ними

Редактировать найденные участки

Визуальное выделение:

html

```
<div class="container"> ← подсвачено  
<h1>Заголовок</h1>  
<div class="content"> ← подсвачено  
<p>Текст</p>  
</div>  
</div>
```

Практическое задание 5

Создадим реальный сценарий поиска:

Представьте, что у вас большой CSS-файл:

CSS

```
.header { background: blue; }  
.content { margin: 10px; }  
.footer { border: 1px solid black; }  
  
.container .header { color: white; }  
.main .content { padding: 20px; }
```

Найдите все использование класса header

Убедитесь, что нашли оба: .header и container .header

Горячие клавиши для поиска

Основные комбинации:

Ctrl + F — открыть поиск

F3 — следующее совпадение

Shift + F3 — предыдущее совпадение
Alt + Enter — выделить все совпадения
Ctrl + D — добавить следующее совпадение к выделению
Ctrl + U — отменить последнее выделение

Дополнительные возможности

Поиск с клавиатуры:

Начните печатать при открытом файле

Нажмите Ctrl + F — введенный текст автоматически попадет в поиск

Быстрое закрытие поиска:

Esc — закрыть панель поиска

Ctrl + F — повторное нажатие закрывает поиск

Настройки поиска

Пользовательские настройки:

```
json
{
    // Исследовать при вводе (инкрементальный поиск)
    "editor.find.seedSearchStringFromSelection": true,

    // Автоматически включать опции
    "editor.find.autoFindInSelection": false,

    // Цвет выделения найденных элементов
    "editor.findMatchBackground": "#ffd700",
    "editor.findMatchHighlightBackground": "#ffd70033",

    // Цвет текущего найденного элемента
    "editor.findRangeHighlightBackground": "#ffd70022"
}
```

Практическое задание 6

Решим реальную проблему с поиском:

У вас есть HTML с ошибками:

```
html
<div calss="container">
```

```
<p clas="text">Текст</p>
<div calss="content">
<span clas="highlight">Выделение</span>
</div>
</div>
```

Найдите все опечатки `calss` и `clas`
Исправьте их на `class`

Решение частых проблем

Проблема 1: Не могу найти текст, который вижу

Решение:

Проверьте опцию `Match Case` — может быть включен учет регистра

Проверьте `Whole Word` — может искааться только целое слово

Убедитесь, что нет лишних пробелов

Проблема 2: Поиск находит слишком много результатов

Решение:

Используйте `Whole Word` для поиска целых слов

Уточните поисковый запрос

Используйте регулярные выражения для точного поиска

Проблема 3: Не вижу все совпадения

Решение:

Прокрутите файл — совпадения могут быть за пределами видимой области

Используйте навигационные клавиши `F3/Shift + F3`

Полезные советы

Используйте `Ctrl + F` часто — не тратьте время на визуальный поиск

Alt + Enter — мощный инструмент для массового редактирования

Запоминайте частые поиски — создавайте шаблоны для часто искомых элементов

Сочетайте с навигацией — используйте `F3` для быстрого перемещения

Не забывайте про `Esc` — чтобы вернуться к обычному редактированию

Чек-лист освоения поиска

Умею открывать поиск (`Ctrl + F`)

Могу находить текст в файле

Умею навигировать (перемещаться) по результатам (`F3, Shift + F3`)

Понимаю опции Match Case и Whole Word

Использую Alt + Enter для выделения всех совпадений

Могу быстро закрыть поиск (Esc)

Использую поиск в ежедневной работе

Теперь вы мастер поиска в файлах! Эта простая, но мощная функция сэкономит вам часы времени при работе с большими проектами. Практикуйтесь регулярно, и скоро поиск станет вашей второй натурой!

Поиск и замена во всем проекте

Дорогой друг!

Теперь давайте изучим поиск и замену во всем проекте — это невероятно мощный инструмент, который позволяет находить и изменять текст сразу во всех файлах вашего проекта. Это настоящая магия для рефакторинга и массовых изменений!

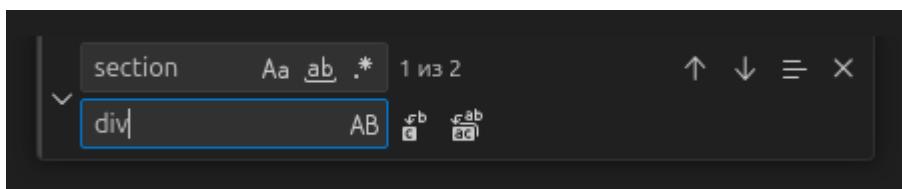
Открытие поиска по проекту

Основные горячие клавиши:

Ctrl + Shift + F — открыть поиск во всем проекте

Ctrl + Shift + H — открыть замену во всем проекте

Как выглядит панель поиска по проекту:



Практическое задание 1

Давайте создадим тестовый проект для практики:

Создайте папку "поиск-проект" со структурой:

text

поиск-проект/
|—— index.html
|—— about.html
|—— style.css
|—— script.js

Заполните файлы одинаковым текстом:

html

```
<!-- в index.html и about.html -->  
<div class="old-class">  
<p>Старый текст</p>  
</div>
```

```
css
/* в style.css */
.old-class { color: red; }
.new-class { color: blue; }
```

```
javascript
// в script.js
var oldClass = "old-class";
var newClass = "new-class";
```

Базовый поиск по проекту

Поиск текста во всех файлах:

Нажмите Ctrl + Shift + F

Ведите old-class

Нажмите Enter

Результат:

text
СОВПАДЕНИЯ (4)
index.html
1: <div class="old-class">

about.html
1: <div class="old-class">

style.css
1: .old-class { color: red; }

script.js
1: var oldClass = "old-class";

Практическое задание 2

Потренируем поиск по проекту:

Откройте поиск Ctrl + Shift + F

Найдите все вхождения:

old-class

div

color

Посмотрите как результаты группируются по файлам

Замена текста во всем проекте

Массовая замена:

Нажмите Ctrl + Shift + H

В поле поиска введите old-class

В поле замены введите updated-class

Нажмите кнопку "Replace All"

Безопасная замена с подтверждением:

Введите поиск и замену

Нажмите кнопку "Replace" рядом с каждым файлом

Или нажмите  для предпросмотра каждого изменения

Практическое задание 3

Сделаем безопасную замену:

Откройте замену Ctrl + Shift + H

Замените old-class на updated-class

Используйте кнопку "Replace" для каждого файла по отдельности

Проверьте, что изменения применились корректно

Фильтры поиска

Включение файлов (include):

*.html — искать только в HTML файлах

*.css, *.js — искать в CSS и JavaScript файлах

src/** — искать в папке src и всех подпапках

Исключение файлов (exclude):

node_modules — не искать в node_modules

*.min.js — исключить минифицированные файлы

dist/, build/ — исключить папки сборки

Практическое задание 4

Поэкспериментируем с фильтрами:

Найдите class только в HTML файлах:

В поле include введите: *.html

Найдите color только в CSS файлах:

В поле include введите: *.css

Исключите определенный файл:
В поле exclude введите: `about.html`

Опции поиска

Четыре основные опции:

Match Case (Aa) — учет регистра
Whole Word — целое слово
Regex (.*) — регулярные выражения
Replace — показывать поле замены

Match Case примеры:

`class` найдет: `class`, `Class`, `CLASS` (без учета регистра)
`class` найдет только: `class` (с учетом регистра)

Whole Word примеры:

`class` найдет: `class`, `my-class`, `classname` (без Whole Word)
`class` найдет только: `class` (с Whole Word)

Практическое задание 5

Протестируем опции поиска:

Найдите `var` с опцией Whole Word

Должны найтиться только отдельные слова `var`

Найдите `Class` с опцией Match Case

Должны найтиться только `Class` (не `class`, `CLASS`)

Регулярные выражения (Regex)

Базовые regex-шаблоны:

`\.class$` — найти `.class` в конце строки
`^div` — найти `div` в начале строки
`colou?r` — найти `color` или `colour`
`\d+` — найти числа
`[a-zA-Z]+` — найти слова

Практическое задание 6

Попробуем регулярные выражения:

Включите опцию Regex

Найдите все CSS-классы:

Ведите: `\.\w+`

Найдите все HTML-теги:

Ведите: </?w+

Предпросмотр замены

Безопасный workflow:

Ведите поиск и замену

Нажмите  рядом с "Replace All"

Просмотрите все изменения перед применением

Подтвердите замену для каждого файла

Пример предпросмотра:

text

 index.html

- <div class="old-class">

+ <div class="updated-class">

style.css

- .old-class { color: red; }

+ .updated-class { color: red; }

Практическое задание 7

Создадим сложный сценарий замены:

У нас есть проект с устаревшими классами:

html

<!-- В нескольких файлах -->

<div class="old-header">

<div class="old-content">

<div class="old-footer">

Замените все old- на new- используя поиск по проекту

Используйте предпросмотр чтобы убедиться в правильности

Горячие клавиши для поиска по проекту

Основные комбинации:

Ctrl + Shift + F — поиск по проекту

Ctrl + Shift + H — замена по проекту

Alt + R — включить/выключить регулярные выражения
Alt + C — включить/выключить учет регистра
Alt + W — включить/выключить целое слово
Ctrl + Alt + Enter — заменить все

Расширенные возможности

Сохраняемые поиски:

Нажмите ... в правом верхнем углу поиска

Выберите "Save Search"

Сохраните частые поисковые запросы

История поиска:

VS Code запоминает ваши последние поиски

Используйте стрелки в поле поиска для навигации по истории

Настройки поиска

Пользовательские настройки:

```
json
{
  // Исключаемые папки по умолчанию
  "search.exclude": {
    "**/node_modules": true,
    "**/dist": true,
    "**/build": true,
    "**/.git": true
  },
  // Включать скрытые файлы
  "search.useIgnoreFiles": true,
  // Следить за изменениями файлов
  "search.usePCRE2": false,
  // Максимальное количество результатов
  "search.maxResults": 20000,
  // Показывать результаты по мере ввода
  "search.searchOnType": true
}
```

Практическое задание 8

Настроим поиск под себя:

Добавьте в настройки исключение папки `images`

Установите максимальное количество результатов 1000

Протестируйте как изменилось поведение поиска

Решение реальных задач

Задача 1: Рефакторинг CSS-классов

`text`

Поиск: `\.old-style`

Замена: `new-design`

Задача 2: Изменение цветовой схемы

`text`

Поиск: `#ff0000`

Замена: `#e74c3c`

Задача 3: Обновление API-эндпоинтов

`text`

Поиск: `api/v1/`

Замена: `api/v2/`

Практическое задание 9

Решим реальную проблему:

Создайте файлы с устаревшим кодом:

`html`

```
<div class="container old-theme">
<button onclick="oldFunction()">Click</button>
</div>
```

Найдите и замените все `old-` префиксы на `new-`

Убедитесь, что изменения применились во всех файлах

Полезные советы

Всегда используйте предпросмотр перед массовой заменой
Сохраняйте частые поисковые запросы
Используйте фильтры для точного поиска
Регулярные выражения — ваш лучший друг для сложных замен
Создавайте бэкап перед массовыми изменениями

Чек-лист освоения

Умею открывать поиск по проекту (Ctrl + Shift + F)
Могу выполнять замену по проекту (Ctrl + Shift + H)
Использую фильтры include/exclude
Понимаю опции Match Case, Whole Word, Regex
Использую предпросмотр перед заменой
Сохраняю частые поисковые запросы
Настроил исключения для поиска

Поздравляю! Теперь вы владеете мощнейшим инструментом поиска и замены по проекту. Этот навык сэкономит вам дни работы при рефакторинге и обновлении больших проектов. Используйте его с умом и всегда проверяйте изменения перед применением!

Как найти и исправить ошибку во всех файлах сразу

Дорогой друг!

Давайте научимся находить и исправлять ошибки во всех файлах проекта одновременно. Это супер-способность, которая сэкономит вам огромное количество времени при работе над большими проектами!

Подготовка к массовому исправлению

Создадим тестовый проект с ошибками:

text

```
проект-с-ошибками/
    ├── index.html
    ├── about.html
    ├── style.css
    └── script.js
```

Наполним файлы с типичными ошибками:

html

```
<!-- index.html -->
<!DOCTYPE html>
<html>
<head>
<title>Мой сайт</title>
<style>
.conteiner { color: red; } <!-- Опечатка! -->
</style>
</head>
<body>
<div class="conteiner"> <!-- Опечатка! -->
<h1>Добро пожаловать</h1>
<p clas="text">Текст страницы</p> <!-- Опечатка! -->
</div>
</body>
</html>
```

html

```
<!-- about.html -->
<div class="conteiner"> <!-- Опечатка! -->
```

```
<h2>О нас</h2>
<p clas="description">Информация о компании</p> <!-- Опечатка! -->
</div>
```

css

```
/* style.css */
.conteiner { /* Опечатка! */
margin: 0 отто; /* Опечатка! */
padding: 20px;
}

.buton { /* Опечатка! */
background-colour: blue; /* Опечатка! */
}
```

javascript

```
// script.js
var utilis = { // Опечатка!
initialize: function() { // Опечатка!
console.log("Скрипт загружен");
}
};

// Неиспользуемая переменная
var unusedVar = "Это нигде не используется";
```

Шаг 1: Поиск ошибок по проекту

Используем поиск с заменой:

Нажмите `Ctrl + Shift + H`

Мы будем искать следующие ошибки:

conteiner → container
clas → class
otto → auto
buton → button
colour → color
utilis → utils
initalize → initialize

Практическое задание 1

Найдем первую ошибку - "conteiner":

Откройте замену **Ctrl + Shift + H**

В поиск введите: **conteiner**

В замену введите: **container**

Посмотрите результаты — должно найтись 3 вхождения

Шаг 2: Безопасное исправление с предпросмотром

Метод предпросмотра перед заменой:

Не нажмайте "Replace All" сразу!

Нажмите  **рядом с "Replace All"**

Просмотрите все изменения:

text

 index.html (2 изменения)

```
- <div class="conteiner">  
+ <div class="container">  
- .conteiner { color: red; }  
+ .container { color: red; }
```

about.html (1 изменение)

```
- <div class="conteiner">  
+ <div class="container">
```

style.css (1 изменение)

```
- .conteiner {  
+ .container {
```

Убедитесь, что все изменения корректны

Нажмите "Replace All" в окне предпросмотра

Практическое задание 2

Исправим "conteiner" с предпросмотром:

Выполните поиск **conteiner** → **container**

Используйте предпросмотр чтобы увидеть все изменения

Примените замену только если уверены в результате

Шаг 3: Исправление нескольких ошибок последовательно

План исправления всех ошибок:

conteiner → **container**

clas → class
otto → auto
buton → button
colour → color
utilis → utils
initalize → initialize

Практическое задание 3

Исправим все опечатки последовательно:
Исправьте каждую ошибку из списка по очереди
Всегда используйте предпросмотр
Проверяйте результаты после каждой замены

Шаг 4: Поиск сложных ошибок с регулярными выражениями

Использование Regex для сложных случаев:
Найдем неиспользуемые переменные в JavaScript:

text
Поиск: `var \w+ = [^;]+;\n(?!.*\1)`
Опция: Regex

Найдем незакрытые HTML-теги:

text
Поиск: `<(\w+)[^>]*>(?!.*</\1>)`
Опция: Regex

Практическое задание 4

Найдем проблемные места с Regex:
Включите опцию Regex
Найдите все CSS-свойства с неправильным синтаксисом:
Поиск: [a-z-]+:[^;]+\$ (находит свойства без точки с запятой)

Шаг 5: Фильтрация файлов для точного поиска

Исправление ошибок только в определенных файлах:

Только в HTML файлах:
Include: *.html
Поиск: clas

Замена: class

Только в CSS файлах:

Include: *.css

Поиск: colour

Замена: color

Исключение определенных файлов:

Exclude: *.min.js, *.min.css

Практическое задание 5

Исправим ошибки с фильтрацией:

Исправьте clas → class **только** в HTML файлах

Исправьте CSS-ошибки только в CSS файлах

Убедитесь, что замены применились только к нужным файлам

Шаг 6: Массовое исправление с учетом контекста

Умные замены с Whole Word:

Проблема: Замена class может затронуть my-class, classname

Решение: Используем Whole Word

Whole Word — заменит только отдельные слова

Без Whole Word — может затронуть составные слова

Практическое задание 6

Исправим ошибки с учетом контекста:

Найдите buton с опцией Whole Word

Убедитесь, что не затронуты слова типа button-group, mybutton

Выполните замену

Шаг 7: Создание чек-листа исправлений

Организованный подход:

Создайте список для проверки:

markdown

Чек-лист исправлений

- [] conteiner → container

- [] clas → class

- [] otto → auto

- [] buton → button

- [] colour → color
- [] utilis → utils
- [] initialize → initialize
- [] Проверить неиспользуемые переменные
- [] Проверить незакрытые теги

Практическое задание 7

Создайте свой чек-лист исправлений:

Просканируйте проект и выпишите все найденные ошибки

Создайте чек-лист в отдельном файле

Выполните исправления по списку, отмечая выполненные

Шаг 8: Автоматизированное исправление с синтаксисом

Создание шаблонов для частых исправлений:

Добавьте в пользовательские синтаксисы:

```
json
{
  "Fix Common Typos": {
    "prefix": "fixTypos",
    "body": [
      "// Автоматическое исправление опечаток",
      "conteiner → container",
      "clas → class",
      "buton → button",
      "colour → color"
    ],
    "description": "Шаблон для исправления частых опечаток"
  }
}
```

Практическое задание 8

Создайте синтаксис для частых исправлений:

Откройте пользовательские синтаксисы

Добавьте шаблон с вашими частыми ошибками

Используйте его при следующем рефакторинге

Шаг 9: Проверка результатов

Финальная проверка после исправлений:

Запустите поиск по исправленным словам

Убедитесь, что ошибок больше нет

Проверьте работоспособность проекта

Создайте коммит с описанием исправлений

Практическое задание 9

Проверим результаты исправлений:

Выполните поиск по всем исправленным словам

Убедитесь, что больше нет вхождений типа `conteiner`, `clas` и т.д.

Откройте файлы и визуально проверьте корректность

Продвинутые техники

Использование расширений для поиска ошибок:

ESLint/TSLint — поиск ошибок в JavaScript/TypeScript

HTMLHint — проверка HTML

Stylelint — проверка CSS

Интеграция с линтерами:

Установите соответствующие расширения

Настройте правила проверки

Проблемы будут показываться в панели "Problems"

Горячие клавиши для эффективного исправления

Ctrl + Shift + H — замена по проекту

Alt + C — переключение учета регистра

Alt + W — переключение целого слова

Alt + R — переключение регулярных выражений

Ctrl + Alt + Enter — заменить все (осторожно!)

Чек-лист мастерства

Умею находить ошибки поиском по проекту

Использую предпросмотр перед массовой заменой

Применяю фильтры для точного поиска

Использую Regex для сложных случаев

Создаю чек-листы для организованного исправления
Проверяю результаты после исправлений
Использую линтеры для автоматического поиска ошибок

Поздравляю! Теперь вы владеете мощнейшей техникой массового исправления ошибок. Этот навык сделает вас настоящим супергероем рефакторинга — вы сможете за минуты исправить то, на что раньше уходили часы! Используйте эту силу с умом и всегда делайте бэкапы перед массовыми изменениями.

Часть 4: Полезные расширения для верстки

Глава 9: Магазин расширений

Как открыть магазин расширений

Дорогой друг!

Давайте научимся открывать и использовать магазин расширений в VS Code. Это волшебное место, где вы найдете тысячи бесплатных дополнений, которые сделают вашу работу удобнее и эффективнее!

Что такое магазин расширений?

Магазин расширений — это встроенный каталог дополнений для VS Code. Здесь вы можете найти:

Темы оформления

Инструменты для HTML/CSS

Языковые пакеты

Средства отладки

И многое другое!

Способ 1: Через панель активности (самый простой)

Пошаговая инструкция:

Найдите в левой панели иконку расширений:

text

[][][][][] ← эта иконка!

Нажмите на иконку с квадратиками (или блоками)

Откроется панель расширений:

text

ПОИСК РАСШИРЕНИЙ

———— РЕКОМЕНДУЕМЫЕ ————

- Russian Language Pack

- Live Server
- Auto Rename Tag

———— ПОПУЛЯРНЫЕ ————

- Prettier
- ESLint
- GitLens

Практическое задание 1

Давайте откроем магазин расширений:

Посмотрите на левую панель VS Code

Найдите иконку с квадратиками или блоками

Нажмите на нее — откроется магазин расширений

Способ 2: Через горячие клавиши (самый быстрый)

Быстрое открытие:

Ctrl + Shift + X — мгновенное открытие магазина расширений

Просто запомните: **X** как eXtensions!

Практическое задание 2

Потренируем быстрый доступ:

Нажмите Ctrl + Shift + X

Магазин расширений откроется мгновенно

Нажмите еще раз Ctrl + Shift + X — панель закроется

Способ 3: Через команду меню

Традиционный способ:

Нажмите на меню "View"

Выберите "Extensions"

Или "Расширения" (если русский интерфейс)

Альтернативный путь:

Ctrl + Shift + P — открыть палитру команд

Ведите: Extensions: Show Extensions
Нажмите Enter

Практическое задание 3

Откроем разными способами:

Попробуйте все три способа:

Через иконку в панели

Через Ctrl + Shift + X

Через меню View → Extensions

Определите, какой способ вам удобнее

Навигация в магазине расширений

Основные разделы панели:

Верхняя часть:

text
🔍 [поле поиска] : (три точки - меню)

Левая часть:

text
— ИССЛЕДОВАТЬ —
— УСТАНОВЛЕННЫЕ —
— РЕКОМЕНДУЕМЫЕ —

Центральная часть:

Список расширений
Описания
Рейтинги
Кнопки установки

Практическое задание 4

Изучим интерфейс магазина:

Откройте магазин расширений
Найдите следующие элементы:
Поле поиска

Список рекомендуемых расширений
Кнопку "Установить"
Кнопку "Включить/Выключить"

Поиск расширений

Как искать нужные расширения:

По названию:

В поле поиска введите: `Russian` — найдет русский язык
Введите: `HTML` — найдет инструменты для HTML

По категориям:

`@category:"themes"` — темы оформления
`@category:"debuggers"` — отладчики
`@category:"snippets"` — сниппеты

По популярности:

Сортировка по количеству установок
Сортировка по рейтингу

Практическое задание 5

Поищем полезные расширения:

Найдите расширения для:

Русского языка: `Russian`
Верстки HTML: `HTML`
Работы с CSS: `CSS`
Красивых тем: `theme`

Установка расширений

Процесс установки:

Найдите нужное расширение
Нажмите кнопку "Install" (Установить)
Дождитесь завершения установки
Перезапустите VS Code если потребуется

Пример установки русского языка:

Поиск: Russian Language Pack
Нажмите "Install"
Нажмите "Restart Now" когда предложат
VS Code перезапустится на русском языке

Практическое задание 6

Установим первое расширение:

Найдите Russian Language Pack
Установите его
Перезапустите VS Code
Убедитесь, что интерфейс стал русским

Управление установленными расширениями

Вкладка "УСТАНОВЛЕННЫЕ":

text
— УСТАНОВЛЕННЫЕ —
• Russian Language Pack [ВКЛ] 
• Live Server [ВКЛ] 
• Auto Rename Tag [ВЫКЛ] 

Действия с расширениями:

Включить/Выключить — кнопка переключатель
 (шестеренка) — настройки расширения
Обновить — если доступно обновление
Удалить — корзина 

Практическое задание 7

Управляем установленными расширениями:

Перейдите на вкладку "УСТАНОВЛЕННЫЕ"
Найдите установленные расширения
Попробуйте отключить и включить расширение

Рекомендуемые расширения для начала

Топ-5 расширений для новичка:

Russian Language Pack — русский интерфейс
Live Server — запуск сайта в браузере
Auto Rename Tag — автопереименование тегов
Auto Close Tag — авто-закрытие тегов
HTML/CSS Support — подсказки для HTML/CSS

Как установить несколько расширений:

Откройте магазин расширений
Поиск: Live Server → Install
Поиск: Auto Rename Tag → Install
Поиск: Auto Close Tag → Install

Практическое задание 8

Установим полезные расширения:

Установите Live Server
Установите Auto Rename Tag
Установите Auto Close Tag
Перезапустите VS Code

Решение проблем

Проблема 1: Магазин не открывается

Решение:
Проверьте интернет-соединение
Попробуйте Ctrl + Shift + X
Перезапустите VS Code

Проблема 2: Не могу найти расширение

Решение:
Проверьте правильность написания
Попробуйте английское название
Используйте категории поиска

Проблема 3: Расширение не устанавливается

Решение:
Проверьте доступ к интернету
Попробуйте установить другое расширение
Перезапустите VS Code

Проблема 4: Не вижу кнопку установки

Решение:

Возможно, расширение уже установлено
Проверьте вкладку "УСТАНОВЛЕННЫЕ"

Полезные советы

Не устанавливайте слишком много расширений — это может замедлить VS Code

Регулярно обновляйте расширения — исправления ошибок и новые функции

Отключайте неиспользуемые расширения — для увеличения производительности

Читайте описания и отзывы перед установкой

Создавайте резервные копии настроек расширений

Безопасность при установке

Проверяйте расширения:

Количество установок (чем больше, тем надежнее)

Рейтинг и отзывы

Дата последнего обновления

Автор расширения (официальные лучше)

Подозрительные признаки:

Мало установок

Нет обновлений больше года

Плохие отзывы

Требует слишком много разрешений

Горячие клавиши для работы с расширениями

Ctrl + Shift + X — открыть/закрыть магазин расширений

Ctrl + Shift + P → "Extensions: Show Extensions" — через палитру команд

Ctrl + , → "Extensions" — настройки расширений

Чек-лист освоения магазина расширений

Умею открывать магазин через иконку

Знаю горячие клавиши **Ctrl + Shift + X**

Могу искать расширения по названию

Умею устанавливать расширения

Могу управлять установленными расширениями

Знаю полезные расширения для начала работы

Проверяю безопасность перед установкой

Поздравляю! Теперь вы знаете, как открывать и использовать магазин расширений. Это ваш портал в мир бесконечных возможностей VS Code. Устанавливайте полезные расширения, экспериментируйте и находите те инструменты, которые сделают вашу работу максимально удобной и эффективной!

Поиск и установка нужных расширений

Дорогой друг!

Теперь давайте научимся искать и устанавливать именно те расширения, которые вам нужны. Это как ходить в библиотеку — нужно знать, как найти нужную книгу среди тысяч других!

Как открыть поиск расширений

Быстрый старт:

Нажмите **Ctrl + Shift + X**

Или кликните на иконку расширений в левой панели

Вы увидите поле поиска вверху:

text

 [Ведите название расширения...]

Практическое задание 1

Откроем поиск расширений:

Нажмите **Ctrl + Shift + X**

Убедитесь, что видите поле поиска

Кликните в поле поиска — курсор должен замигать

Эффективный поиск расширений

Метод 1: Поиск по точному названию

Если знаете название расширения:

Введите: Russian Language Pack

Введите: Live Server

Введите: Auto Rename Tag

Метод 2: Поиск по функционалу

Если не знаете название, но знаете что нужно:

Для русского языка: `russian`
Для работы с HTML: `html`
Для подсветки кода: `syntax`
Для темы оформления: `theme`

Метод 3: Поиск по категориям

Используйте специальные префиксы:

`@category:"themes"` — все темы оформления
`@category:"debuggers"` — отладчики
`@category:"snippets"` — сниппеты кода
`@category:"formatters"` — форматирование кода

Практическое задание 2

Потренируем разные методы поиска:

Найдите расширение для русского языка:
Ведите: `russian`

Найдите все темы оформления:
Ведите: `@category:"themes"`

Найдите инструменты для HTML:
Ведите: `html`

Как выбрать правильное расширение

На что обращать внимание:

Информационная карточка расширения:

`text`
Название: Live Server
Автор: Ritwick Dey
Рейтинг: ★★★★★ (4.7/5)
Установок: 10 млн+
Обновлено: 2 недели назад

Описание: Запускает локальный сервер...

Критерии выбора:

Количество установок — чем больше, тем надежнее

Рейтинг — звездочки от 1 до 5

Дата обновления — свежие обновления важны

Описание — четко ли описаны возможности

Скриншоты — показывают как выглядит расширение

Практическое задание 3

Проанализируем расширение:

Найдите Live Server

Изучите информацию:

Сколько установок?

Какой рейтинг?

Когда последнее обновление?

Читабельное ли описание?

Процесс установки шаг за шагом

Полный процесс установки:

Найдите нужное расширение

Нажмите кнопку "Install" (Установить)

Дождитесь загрузки и установки

При необходимости перезапустите VS Code

Визуальные подсказки при установке:

Синяя кнопка "Install" → еще не установлено

Серая кнопка с галочкой ✓ → уже установлено

Кружок загрузки  → идет установка

Практическое задание 4

Установим первое расширение:

Найдите Russian Language Pack

Нажмите синюю кнопку "Install"

Наблюдайте процесс установки

Перезапустите VS Code когда предложат

Установка нескольких расширений

Пакетная установка полезных расширений:

Для веб-разработки установите:

Live Server — запуск сайта в браузере

Auto Rename Tag — автопереименование HTML-тегов

Auto Close Tag — авто-закрытие тегов

HTML CSS Support — подсказки для HTML/CSS

Color Highlight — подсветка цветов в коде

Порядок установки:

Установите первое расширение

Дождитесь завершения

Установите следующее

Не устанавливайте все сразу — тестируйте по одному

Практическое задание 5

Установим набор для верстки:

Установите Live Server

Установите Auto Rename Tag

Установите Auto Close Tag

Проверьте, что все установилось корректно

Поиск по популярности

Как найти самые популярные расширения:

Оставьте поле поиска пустым

Посмотрите раздел "РЕКОМЕНДУЕМЫЕ"

Или "ПОПУЛЯРНЫЕ"

Или отсортируйте по количеству установок

Топ-10 самых популярных расширений:

Prettier — форматирование кода

ESLint — проверка JavaScript

GitLens — улучшенная работа с Git

Live Server — локальный сервер
Material Theme — темы оформления
Bracket Pair Colorizer — подсветка скобок
Path Intellisense — автодополнение путей
npm Intellisense — автодополнение npm-пакетов
Code Runner — запуск кода
Settings Sync — синхронизация настроек

Практическое задание 6

Изучим популярные расширения:

Очистите поле поиска
Посмотрите раздел "ПОПУЛЯРНЫЕ"
Изучите 3-5 самых популярных расширений
Прочтайте их описания

Поиск по тегам и метаданным

Расширенный поиск:

Поиск по автору:
@id:ritwickdey — все расширения автора Ritwick Dey

Поиск по тегам:
tag:html — расширения с тегом html
tag:css — расширения с тегом css
tag:javascript — расширения с тегом javascript

Комбинированный поиск:
html tag:formatter — HTML-форматтеры
theme tag:dark — темные темы

Практическое задание 7

Используем расширенный поиск:

Найдите все расширения для форматирования HTML:

Ведите: html tag:formatter

Найдите темные темы оформления:

Ведите: theme tag:dark

Проверка совместимости

Важные моменты перед установкой:

Требования VS Code — какая версия нужна

Зависимости — требуются ли другие расширения

Размер — большие расширения могут замедлить работу

Поддержка ОС — работает ли на вашей операционной системе

Где посмотреть информацию:

В карточке расширения есть раздел "Details"

Там указаны требования и зависимости

Практическое задание 8

Проверим совместимость:

Выберите любое расширение

Найдите раздел "Details" или "Подробности"

Проверьте требования к версии VS Code

Посмотрите список зависимостей

Управление установленными расширениями

После установки:

Вкладка "УСТАНОВЛЕННЫЕ" показывает:

text

Russian Language Pack [ВКЛЮЧЕНО] 

Live Server [ВКЛЮЧЕНО] 

Auto Rename Tag [ВЫКЛЮЧЕНО] 

Доступные действия:

Включить/выключить — переключатель

Настройки  — конфигурация расширения

Обновить  — если есть обновление

Удалить  — удалить расширение

Практическое задание 9

Управляем установленными расширениями:

Перейдите на вкладку "УСТАНОВЛЕННЫЕ"

Найдите установленное расширение

Попробуйте его отключить

Снова включите

Решение проблем при установке

Проблема 1: "Install" кнопка неактивна

Решение:

Убедитесь, что есть интернет-соединение

Проверьте, не установлено ли уже это расширение

Попробуйте обновить VS Code

Проблема 2: Установка зависла

Решение:

Закройте и откройте панель расширений

Перезапустите VS Code

Попробуйте установить позже

Проблема 3: Расширение не работает после установки

Решение:

Перезапустите VS Code

Проверьте, включено ли расширение

Посмотрите настройки расширения

Безопасность при установке

Красные флаги (чего опасаться):

Меньше 1000 установок

Последнее обновление больше года назад

Нет описания или скриншотов

Много негативных отзывов

Требует подозрительные разрешения

Зеленые флаги (что хорошо):

Более 100,000 установок

Регулярные обновления

Хорошее описание со скриншотами

Высокий рейтинг (4+ звезды)

Официальный автор или известная компания

Полезные советы

Устанавливайте по одному — так легче понять, какое расширение вызывает проблемы

Тестируйте после установки — проверьте работу расширения

Читайте документацию — у многих расширений есть инструкции по использованию

Отключайте ненужные расширения — для увеличения скорости работы

Регулярно обновляйте — для получения исправлений и новых функций

Чек-лист поиска и установки

Умею открывать поиск расширений

Могу искать по названию и функционалу

Знаю как оценить качество расширения

Умею устанавливать расширения

Могу управлять установленными расширениями

Проверяю совместимость перед установкой

Следую правилам безопасности

Поздравляю! Теперь вы эксперт по поиску и установке расширений. Вы можете находить именно те инструменты, которые нужны для вашей работы, и безопасно их устанавливать. Помните — хорошие расширения делают хорошего разработчика еще лучше!

Безопасность: устанавливаем только проверенные расширения

Дорогой друг!

Давайте поговорим о безопасности при установке расширений. Это очень важная тема, потому что расширения имеют доступ к вашему коду и компьютеру. Будем учиться выбирать только безопасные и проверенные расширения!

Почему безопасность важна?

Расширения могут:

- Читать и изменять ваш код
- Получать доступ к файловой системе
- Отправлять данные в интернет
- Выполнять произвольный код на вашем компьютере

Красные флаги (опасные признаки)

1. Подозрительное количество установок

Что смотреть:

text

 Опасные:

- 12 установок
- 100 установок
- 500 установок

 Безопасные:

- 1,200,000 установок
- 500,000 установок
- 10,000,000 установок

Правило: Избегайте расширений с менее чем **10,000 установок**, если только это не от известного разработчика.

2. Старые обновления

Дата последнего обновления:

text

 Опасные:

- 3 года назад

 Безопасные:

- 2 недели назад

- 1.5 года назад
- 8 месяцев назад
- 1 месяц назад
- 3 месяца назад

Правило: Не устанавливайте расширения, которые не обновлялись **больше года**.

Практическое задание 1

Проверим расширение на безопасность:

Найдите расширение Live Server

Проверьте:

Сколько установок?

Когда последнее обновление?

Сделайте вывод — безопасно ли это расширение?

3. Слишком много разрешений

Опасные разрешения:

В описании расширения ищите предупреждения:

text

 Этот модуль требует:

- Доступ ко всем файлам на вашем компьютере
- Возможность запуска произвольного кода
- Доступ к интернету без ограничений

Безопасные разрешения:

text

 Это расширение:

- Читает только файлы текущего проекта
- Не имеет доступа к интернету
- Работает только в контексте редактора

Практическое задание 2

Проанализируем разрешения:

Откройте страницу любого расширения

Найдите раздел "Contributions" или "Разрешения"

Посмотрите, какие права запрашивает расширение

4. Плохие отзывы и рейтинг

Как оценивать отзывы:

Читаем отзывы внимательно:

text

 Плохие отзывы:

- "Ломает VS Code"
- "Содержит вирус"
- "Крадет данные"
- "Не обновляется 2 года"

 Хорошие отзывы:

- "Отлично работает"
- "Автор быстро отвечает"
- "Регулярные обновления"
- "Помог решить проблему"

Смотрим рейтинг:

★★★★★ (4.5-5.0) — отлично

★★★★☆ (4.0-4.5) — хорошо

★★★☆☆ (3.0-4.0) — осторожно

★★☆☆☆ (2.0-3.0) — избегать

★☆☆☆☆ (0.0-2.0) — никогда не устанавливать

Практическое задание 3

Проанализируем отзывы:

Выберите популярное расширение

Прочтайте 5-10 отзывов

Обратите внимание на повторяющиеся проблемы

5. Неизвестные разработчики

Проверяем автора:

Проверенные авторы:

text

 Microsoft — разработчики VS Code

 Ritwick Dey — автор Live Server

 Christian Kohler — автор Path Intellisense

 Shan Khan — автор Auto Rename Tag

Проверяем профиль автора:

Сколько расширений создал?
Как давно на Marketplace?
Есть ли сайт или документация?

Практическое задание 4

Исследуем автора расширения:

Нажмите на имя автора в карточке расширения
Посмотрите другие его расширения
Проверьте его профиль на GitHub (если есть ссылка)

6. Отсутствие документации

Что должно быть у безопасного расширения:

Обязательные элементы:

text

-  README файл с описанием
-  Инструкция по установке
-  Примеры использования
-  Скриншоты работы
-  Ссылка на исходный код (GitHub)
-  Информация о лицензии

Тревожные признаки:

text

-  Нет описания
-  Нет скриншотов
-  Скрытый исходный код
-  Нет контактов автора

Практическое задание 5

Проверим документацию:

Откройте страницу расширения
Прокрутите вниз до раздела "README"

Оцените качество документации

Безопасный процесс установки

Пошаговый план безопасности:

Исследование (5-10 минут)

Проверка (2-3 минуты)

Установка (1 минута)

Тестирование (5 минут)

Практическое задание 6

Создадим чек-лист безопасности:

Возьмите лист бумаги или откройте текстовый файл

Напишите ваш чек-лист проверки расширения

Используйте его при следующей установке

Проверенные расширения для начала

Безопасный стартовый набор:

Язык и интерфейс:

1. **Russian Language Pack** (Microsoft) — ★★★★★ 10M+

2. **Material Icon Theme** (Philipp Kief) — ★★★★★ 12M+

Веб-разработка:

3. **Live Server** (Ritwick Dey) — ★★★★☆ 18M+

4. **Auto Rename Tag** (Jun Han) — ★★★★★ 8M+

5. **Auto Close Tag** (Jun Han) — ★★★★★ 6M+

Качество кода:

6. **Prettier** (Prettier team) — ★★★★★ 22M+

7. **ESLint** (Microsoft) — ★★★★★ 20M+

Практическое задание 7

Установим безопасный набор:

Установите 2-3 расширения из безопасного списка

Следуйте вашему чек-листву безопасности

Записывайте свои наблюдения

Что делать, если расширение вызывает проблемы

Симптомы проблем:

VS Code стал медленным

Появились странные ошибки

Расширение не работает как ожидалось

Подозрительная активность

Действия при проблемах:

Отключите расширение

Проверьте не исчезла ли проблема

Удалите проблемное расширение

Сообщите о проблеме автору

Напишите честный отзыв

Практическое задание 8

Потренируем отключение проблемного расширения:

Найдите вкладку "УСТАНОВЛЕННЫЕ"

Выберите любое расширение

Нажмите "Отключить"

Убедитесь, что оно отключилось

Регулярное обслуживание

Ежемесячная проверка:

Просмотрите все установленные расширения

Удалите неиспользуемые

Обновите остальные

Проверьте не появились ли плохие отзывы

Как проверить обновления:

Откройте панель расширений

Найдите вкладку "УСТАНОВЛЕННЫЕ"

Посмотрите есть ли кнопка "Обновить"

Практическое задание 9

Проведем аудит установленных расширений:

Сделайте список всех установленных расширений

Отметьте те, которые не используете

Решите какие оставить, а какие удалить

Особые случаи

Расширения для профессиональных задач:

Если вам нужно специализированное расширение:

Ищите от известных компаний (Microsoft, Google, GitHub)

Проверяйте наличие enterprise-версии

Ищите расширения с открытым исходным кодом

Предпочтительнее платные расширения от известных разработчиков

Расширения для обучения:

Безопасные образовательные расширения:

Code Runner (Jun Han) — для запуска кода

GitLens (GitKraken) — для работы с Git

Markdown All in One (Yu Zhang) — для Markdown

Горячие клавиши для безопасности

Ctrl + Shift + X — открыть расширения

Ctrl + , → "Extensions" — настройки расширений

F1 → "Extensions: Show Installed Extensions" — список установленных

Чек-лист безопасности

Перед установкой проверьте:

Более 10,000 установок

Обновлялось менее года назад

Рейтинг 4.0+ звезд

Хорошие отзывы (прочтите несколько)

Известный или проверенный автор

Есть документация и скриншоты
Запрашивает разумные разрешения
Есть ссылка на исходный код

После установки:

Расширение работает корректно
Не замедляет VS Code
Не вызывает ошибок
Автор отвечает на вопросы (если есть)

Полезные ресурсы для проверки

Официальный Marketplace — всегда устанавливайте отсюда
GitHub репозитории — проверяйте исходный код
Stack Overflow — ищите отзывы разработчиков
Официальная документация VS Code — списки рекомендуемых расширений

Поздравляю! Теперь вы знаете, как безопасно выбирать и устанавливать расширения. Помните: лучше установить меньше расширений, но качественных и проверенных, чем много сомнительных. Ваша безопасность и производительность стоят того, чтобы потратить 10 минут на проверку перед установкой!

Глава 10: Самые нужные расширения для начинающих

Live Server - просмотр сайта с автоматическим обновлением

Дорогой друг!

Давайте подробно изучим **Live Server** — самое популярное расширение для веб-разработчиков. Это настоящая магия, которая превращает вашу рутинную работу в удовольствие!

Что такое Live Server?

Live Server — это расширение, которое:

Запускает локальный веб-сервер на вашем компьютере

Автоматически обновляет страницу в браузере при изменении кода

Показывает ваш сайт так, как его увидят пользователи

Почему Live Server лучше простого открытия файла?

Без Live Server:

Изменили код → Сохранили файл (Ctrl + S)

Переключились в браузер

Нажали F5 (обновить)

Увидели изменения

Вернулись в VS Code

С Live Server:

Изменили код → Сохранили файл (Ctrl + S)

Браузер автоматически обновился! 🎉

Установка Live Server

Безопасная установка:

Откройте магазин расширений (Ctrl + Shift + X)

В поиске введите: Live Server

Найдите расширение от Ritwick Dey

Проверьте:

Установок: 18+ миллионов ✓

Рейтинг: 4.7/5 ★★★★★ ✓

Последнее обновление: недавно ✓

Нажмите "Install"

Практическое задание 1

Установим Live Server:

Найдите и установите Live Server

Дождитесь завершения установки

Перезапустите VS Code если потребуется

Первый запуск Live Server

Создадим тестовый проект:

Создайте папку "live-server-test"

Создайте файл index.html с содержимым:

```
html
<!DOCTYPE html>
<html>
<head>
<title>Тест Live Server</title>
</head>
<body>
<h1>Привет, Live Server!</h1>
<p>Этот текст мы будем менять</p>
</body>
</html>
```

Запуск сервера:

Способ 1: Через правую кнопку мыши (рекомендуется)

Правой кнопкой мыши на файле index.html

Выберите "Open with Live Server"

Способ 2: Через панель состояния

Найдите внизу кнопку "Go Live"

Нажмите на нее

Способ 3: Горячие клавиши

Alt + L затем **Alt + O** — открыть Live Server

Или **Alt + L** затем **Alt + C** — закрыть

Практическое задание 2

Запустим Live Server:

Создайте тестовый проект

Запустите Live Server любым способом

Убедитесь, что в браузере открылась ваша страница

Что происходит после запуска

Индикаторы работы:

В VS Code:

В панели состояния (внизу) появляется: Port: 5500 или другой номер
Кнопка "Go Live" меняется на "Port: 5500"

В браузере:

Открывается новая вкладка с адресом: http://127.0.0.1:5500/
Или: http://localhost:5500/

Важно: 127.0.0.1 и localhost — это ваш собственный компьютер!

Практическое задание 3

Проверим работу:

Посмотрите адрес в браузере

Запомните номер порта (обычно 5500)

Найдите индикаторы в VS Code

Автоматическое обновление — магия Live Server

Потрясающий эксперимент:

В файле index.html **измените** текст:

html

<p>Этот текст мы будем менять</p>

на:

html

```
<p>Смотрите! Текст изменился сам!</p>
```

Сохраните файл (Ctrl + S)

Посмотрите в браузер — страница обновилась автоматически!

Практическое задание 4

Поэкспериментируем с автообновлением:

Изменяйте HTML-код

Сохраняйте файл (Ctrl + S)

Наблюдайте как браузер обновляется сам

Попробуйте изменить:

Текст внутри тегов

Добавить новые теги

Изменить атрибуты

Работа с CSS файлами

Подключим CSS к проекту:

Создайте файл style.css в той же папке:

CSS

```
body {  
background-color: lightblue;  
}
```

```
h1 {  
color: darkblue;  
text-align: center;  
}
```

Подключите CSS в HTML:

html

```
<head>  
<title>Тест Live Server</title>  
<link rel="stylesheet" href="style.css">  
</head>
```

Практическое задание 5

Протестируем автообновление CSS:

Измените цвет в style.css:

CSS

```
body {  
background-color: lightgreen; /* было lightblue */  
}
```

Сохраните (Ctrl + S)

Убедитесь, что цвет фона в браузере изменился

Множественные файлы

Работа с несколькими HTML файлами:

Создайте about.html

Запустите Live Server из index.html

В браузере перейдите: <http://localhost:5500/about.html>

Важно: Live Server работает для **всех файлов** в папке проекта!

Практическое задание 6

Создадим многостраничный сайт:

Создайте 3 HTML файла:

```
index.html  
about.html  
contact.html
```

Создайте навигацию:

```
html  
<nav>  
<a href="index.html">Главная</a>  
<a href="about.html">О нас</a>  
<a href="contact.html">Контакты</a>  
</nav>
```

Протестируйте переходы между страницами
Настройки Live Server

Открываем настройки:

Нажмите **Ctrl + ,**

В поиске введите: **live server**

Или откройте настройки расширения:

Панель расширений → Live Server →  (шестеренка)

Полезные настройки:

```
json
{
    // Изменить порт по умолчанию
    "liveServer.settings.port": 3000,
    // Автоматически открывать браузер
    "liveServer.settings.CustomBrowser": "chrome",
    // Не показывать "работает на Live Server"
    "liveServer.settings.AdvanceCustomBrowserCmdLine": "",
    // Задержка перед обновлением (мс)
    "liveServer.settings.fullReload": true
}
```

Практическое задание 7

Настроим Live Server под себя:

Измените порт на 3000

Перезапустите Live Server

Убедитесь, что работает на новом порту

Горячие клавиши Live Server

Основные комбинации:

Alt + L Alt + O — открыть/запустить Live Server

Alt + L Alt + C — закрыть/остановить Live Server

Alt + L Alt + P — изменить порт

Быстрый доступ:

Откройте палитру команд: Ctrl + Shift + P

Ведите:

Live Server: Open with Live Server — запустить

Live Server: Stop Live Server — остановить

Live Server: Change Port — изменить порт

Практическое задание 8

Освоим горячие клавиши:

Запустите Live Server через Alt + L Alt + O

Остановите через Alt + L Alt + C

Попробуйте через палитру команд

Отладка с Live Server

Инструменты разработчика в браузере:

Откройте сайт через Live Server

Нажмите F12 в браузере

Используйте:

Elements/Элементы — просмотр и изменение HTML/CSS

Console/Консоль — отладка JavaScript

Network/Сеть — проверка загрузки файлов

Особенность: Изменения в инструментах разработчика не сохраняются — только в VS Code!

Практическое задание 9

Поотлаживаем с Live Server:

Откройте инструменты разработчика (F12)

Измените CSS в браузере (для теста)

Измените CSS в VS Code и сохраните

Убедитесь, что изменения из VS Code перезаписывают изменения в браузере

Решение проблем

Проблема 1: "Cannot GET /" ошибка

Решение:

Убедитесь, что файл называется index.html

Запускайте из папки, где есть index.html

Попробуйте перезапустить Live Server

Проблема 2: Порт уже используется

Решение:

Измените порт в настройках

Или закройте другую программу, использующую порт 5500

Проблема 3: Не обновляется автоматически

Решение:

Убедитесь, что сохраняете файл (Ctrl + S)

Проверьте, не отключен ли авто-релоад в настройках

Попробуйте перезапустить Live Server

Проблема 4: Не открывается в нужном браузере

Решение:

Настройте браузер по умолчанию в настройках Live Server

Или запускайте правой кнопкой → "Open with Live Server"

Продвинутые возможности

Работа с JavaScript:

Создайте script.js:

```
javascript
document.addEventListener('DOMContentLoaded', function() {
  console.log('Страница загружена!');
  document.querySelector('h1').textContent = 'Привет из JavaScript!';
});
```

Подключите к HTML:

html

```
<script src="script.js"></script>
```

Изображения и другие ресурсы:

Live Server обслуживает **все файлы** в папке проекта
Картинки, шрифты, видео — всё доступно
Пути указываются относительно папки проекта

Пример с изображением:

html

```

```

Файл должен быть в папке `images/` внутри вашего проекта.

Чек-лист освоения Live Server

Установил Live Server безопасно
Могу запустить сервер разными способами
Понимаю, что такое localhost и порт
Вижу автоматическое обновление при сохранении
Работаю с CSS и вижу изменения
Использую несколько HTML страниц
Настроил под себя (порт, браузер)
Использую горячие клавиши
Умею решать частые проблемы
Работаю с JavaScript и изображениями

Профессиональный рабочий процесс

Идеальный workflow с Live Server:

Открываю VS Code с проектом
Запускаю Live Server (Alt + L Alt + O)
Работаю над HTML/CSS/JS в VS Code
Сохраняю (Ctrl + S) каждые 30-60 секунд
Смотрю изменения в браузере (автообновление!)
Использую инструменты разработчика для отладки
Останавливаю Live Server в конце работы

Поздравляю! Теперь вы владеете Live Server — одним из самых мощных инструментов для веб-разработки. С ним ваша работа станет в разы быстрее и приятнее. Помните: хороший разработчик не нажимает F5 — у него работает Live Server!

Russian Language Pack - русский интерфейс

Дорогой друг!

Давайте подробно разберем **Russian Language Pack** — расширение, которое переводит интерфейс VS Code на русский язык. Это сделает программу гораздо понятнее и комфортнее для вас!

Что такое Russian Language Pack?

Russian Language Pack — это официальное расширение от Microsoft, которое:

Переводит меню, кнопки и сообщения на русский
Не затрагивает ваш код (он остается на английском)
Сделано профессиональными переводчиками
Регулярно обновляется

Зачем нужен русский интерфейс?

Преимущества для начинающих:

Не нужно запоминать английские названия команд
Быстрее находите нужные функции
Понимаете сообщения об ошибках
Чувствуете себя увереннее

Важно: Код (HTML, CSS, JavaScript) остается на английском — это мировой стандарт!

Установка Russian Language Pack

Проверенная безопасная установка:

Откройте магазин расширений (Ctrl + Shift + X)
В поиске введите: Russian Language Pack

Найдите расширение от Microsoft:

text

Russian Language Pack for VS Code

Author: Microsoft

★★★★★ 4.8/5 (10+ миллионов установок)

Проверьте безопасность:

- Автор: Microsoft (самые разработчики VS Code)
- Установок: 10+ миллионов
- Рейтинг: 4.8/5
- Последнее обновление: недавно

Нажмите "Установить" (Install)

Практическое задание 1

Установим Russian Language Pack:

Найдите расширение в магазине

Убедитесь, что это от Microsoft

Установите его

Дождитесь завершения установки

Активация русского языка

После установки:

Способ 1: Автоматическое предложение

VS Code предложит переключиться на русский

Нажмите "Restart Now" (Перезапустить сейчас)

Способ 2: Вручную через команду

Нажмите Ctrl + Shift + P

Введите: Configure Display Language

Выберите "ru" (Russian)

Перезапустите VS Code

Способ 3: Через настройки

Нажмите Ctrl + ,

В поиске введите: locale

Найдите: Locale: Configure Display Language

Выберите "ru"

Перезапустите

Практическое задание 2

Активируем русский язык:

Дождитесь предложения перезапуска
Или активируйте вручную через **Ctrl + Shift + P**
Перезапустите VS Code
Убедитесь, что интерфейс стал русским

Как выглядит русский интерфейс

Основные изменения:

Меню вверху:

text

Было:	Стало:
File	Файл
Edit	Правка
View	Вид
Go	Переход
Run	Запуск
Terminal	Терминал
Help	Справка

Панель активности слева:

text

Было:	Стало:
EXPLORER	ПРОВОДНИК
SEARCH	ПОИСК
SOURCE CONTROL	СИСТЕМА КОНТРОЛЯ ВЕРСИЙ
RUN AND DEBUG	ЗАПУСК И ОТЛАДКА
EXTENSIONS	РАСШИРЕНИЯ

Статусная строка внизу:

text

Было:	Стало:
UTF-8	UTF-8
LF	LF
Spaces: 4	Пробелов: 4
HTML	HTML

Практическое задание 3

Изучим русский интерфейс:

Посмотрите на верхнее меню

Изучите панель активности слева

Проверьте статусную строку

Найдите знакомые элементы в новом виде

Палитра команд на русском

Как искать команды:

Раньше (на английском):

Ctrl + Shift + P → вводите: new file

Теперь (на русском):

Ctrl + Shift + P → вводите: новый файл

Или можно продолжать на английском:

VS Code понимает оба языка!

Примеры команд:

Text

Английский: Русский:

new file новый файл

open folder открыть папку

settings настройки

extensions расширения

Практическое задание 4

Попрактикуем палитру команд:

Нажмите Ctrl + Shift + P

Ведите на русском: новый файл

Ведите на английском: new file

Убедитесь, что оба варианта работают

Контекстные меню на русском

Правая кнопка мыши:

В проводнике:

text

Было: Стало:

New File Новый файл

New Folder Новая папка

Rename Переименовать

Delete Удалить

В редакторе кода:

text

Было: Стало:

Cut Вырезать

Copy Копировать

Paste Вставить

Format Document Форматировать документ

Практическое задание 5

Изучим контекстные меню:

Правой кнопкой в проводнике

Правой кнопкой в области кода

Посмотрите какие команды стали понятнее

Сообщения об ошибках

Теперь ошибки на русском:

Раньше:

text

Error: Cannot read property 'length' of undefined

Теперь:

text

Ошибка: Не удается прочитать свойство 'length' значения undefined

Диалоговые окна:

Сохранение файла:

text

Было:

Do you want to save changes?

Save

Don't Save

Cancel

Стало:

Хотите сохранить изменения в файле?

Сохранить

Не сохранять

Отмена

Практическое задание 6

Протестируем сообщения:

Создайте несохраненные изменения в файле

Попробуйте закрыть файл

Посмотрите на диалоговое окно на русском

Настройки на русском

Окно настроек:

Открываем: Ctrl + ,

Разделы настроек:

text

Было:	Стало:
Commonly Used	Часто используемые
Text Editor	Текстовый редактор
Workbench	Рабочая область
Window	Окно
Features	Возможности
Application	Приложение
Extensions	Расширения

Поиск в настройках:

Можете искать на русском: размер шрифта
Или на английском: font size
Оба варианта работают!

Практическое задание 7

Поищем настройки на русском:
Откройте настройки (Ctrl + ,)

В поиске введите:

размер шрифта

тема

автосохранение

Убедитесь, что поиск работает

Возврат к английскому интерфейсу

Если захотите вернуться:

Ctrl + Shift + P

Ведите: настроить язык интерфейса

Выберите "en" (English)

Перезапустите VS Code

Или через настройки:

json
{

```
"locale": "en"
```

```
}
```

Практическое задание 8

Попробуем переключиться обратно:

Переключитесь на английский интерфейс

Посмотрите, как изменился VS Code

Вернитесь к русскому интерфейсу

Особенности работы

Что переводится:

- Меню и подменю
- Кнопки и надписи
- Сообщения и ошибки
- Названия настроек
- Диалоговые окна

Что НЕ переводится:

- Ваш код (HTML, CSS, JavaScript)
- Названия файлов и папок
- Команды терминала
- Некоторые технические термины

Практическое задание 9

Проверим что переведено, а что нет:

Посмотрите на свой код — он на английском?

Проверьте названия файлов в проводнике

Убедитесь, что технические термины могут остаться на английском

Решение проблем

Проблема 1: Не предлагает перезапуск после установки

Решение:

Ctrl + Shift + P

настроить язык интерфейса

Выберите "ru"

Перезапустите вручную

Проблема 2: Часть интерфейса осталась на английском

Решение:

Это нормально для некоторых технических терминов

Перезапустите VS Code еще раз

Проверьте обновления расширения

Проблема 3: Хочу видеть оригинальные английские термины

Решение:

Наводите курсор на элемент — часто показывается оригинальное название

Или временно переключитесь на английский интерфейс

Проблема 4: Перевод кажется неточным

Решение:

Это официальный перевод от Microsoft

Технические термины часто не имеют идеального перевода

Можно предложить улучшение через GitHub Microsoft

Полезные советы

Используйте оба языка в поиске — что-то легче найти на русском, что-то на английском

Запоминайте английские термины — они пригодятся при чтении документации

Не бойтесь переключаться между языками

Сообщайте о проблемах с переводом — помогайте улучшать

Горячие клавиши

Важно: Горячие клавиши **не меняются**:

Ctrl + S — все равно Сохранить

Ctrl + C — все равно Копировать

Ctrl + V — все равно Вставить

Ctrl + Z — все равно Отменить

Работа с английской документацией

Как быть с англоязычными ресурсами:

Если читаете урок на английском:

Автор говорит: "Go to File menu"

Вы ищете: меню "Файл"

Находите нужную команду

Или запоминаете соответствия:

File = Файл

Edit = Правка

View = Вид

и т.д.

Чек-лист освоения

Установил Russian Language Pack от Microsoft

Активировал русский интерфейс

Научился использовать палитру команд на русском

Понимаю русские сообщения об ошибках

Могу искать настройки на русском

Знаю, как вернуться к английскому интерфейсу

Понимаю, что код остается на английском

Знаю, как работать с англоязычными уроками

Профессиональный подход

Рекомендуемый путь:

Начинайте с русского интерфейса — чтобы быстро освоиться

Постепенно запоминайте английские термины

Через 2-3 месяца попробуйте переключиться на английский

Или оставайтесь на русском — как вам удобнее

Для командной работы:

Договоритесь с коллегами об языке интерфейса

Или используйте английский для совместимости

Главное — код всегда на английском!

Поздравляю! Теперь у вас русский интерфейс VS Code. Это сделает обучение веб-разработке гораздо комфортнее. Помните: язык интерфейса — это всего лишь оболочка, а самое важное — код, который вы создаете. И он, как и у всех профессиональных разработчиков в мире, будет на английском!

Auto Rename Tag - автоматическое переименование парных тегов

Дорогой друг!

Давайте изучим **Auto Rename Tag** — одно из самых полезных расширений для верстки. Это волшебный инструмент, который избавит вас от рутины и сделает работу с HTML в разы быстрее!

Что такое Auto Rename Tag?

Auto Rename Tag — это расширение, которое:

Автоматически переименовывает закрывающий тег при изменении открывающего

Синхронизирует имена парных тегов

Работает с HTML, XML, JSX и другими языками разметки

Экономит массу времени и предотвращает ошибки

Проблема, которую решает расширение

Без Auto Rename Tag:

```
html
<div class="container">
<h1>Заголовок</h1>
<p>Текст</p>
</div> <!-- Если переименовать <div> на <section>, -->
<!-- нужно ВРУЧНУЮ менять и </div> на </section> -->
```

С Auto Rename Tag:

```
html
<section class="container"> <!-- Меняете <div> на <section> -->
<h1>Заголовок</h1>
<p>Текст</p>
</section> <!-- </div> АВТОМАТИЧЕСКИ меняется на </section>! -->
```

Установка Auto Rename Tag

Безопасная установка:

Откройте магазин расширений (Ctrl + Shift + X)

В поиске введите: Auto Rename Tag

Найдите расширение от Jun Han:

text

Auto Rename Tag

Author: Jun Han

★★★★★ 4.9/5 (8+ миллионов установок)

Проверьте безопасность:

- Автор: Jun Han (известный разработчик)
- Установок: 8+ миллионов
- Рейтинг: 4.9/5 (почти идеальный!)
- Последнее обновление: активно обновляется

Нажмите "Установить"

Практическое задание 1

Установим Auto Rename Tag:

Найдите и установите расширение

Не требуется перезапуск VS Code

Расширение готово к работе сразу!

Как работает Auto Rename Tag

Базовый пример:

Исходный код:

html

<div>Текст</div>

Действия:

**Изменяете открывающий тег: <div> → **

**Закрывающий тег автоматически меняется: </div> → **

Результат:

html

Текст <!-- Оба тега изменились! -->

Практическое задание 2

Проверим базовую работу:

Создайте файл test.html

Напишите: <div>Тест</div>

Изменяйте тег <div> на разные варианты:

<p>

<section>

Убедитесь, что закрывающий тег меняется автоматически

Работа со сложными тегами

Теги с атрибутами:

html

<!-- Было: -->

<div class="container" id="main">Содержимое</div>

<!-- Меняем <div> на <section> -->

<section class="container" id="main">Содержимое</section>

<!-- Закрывающий тег тоже изменился! -->

Вложенные теги:

html

<!-- Было: -->

<div>

<p>Текст в параграфе</p>

</div>

<!-- Меняем <div> на <article> -->

<article>

<p>Текст в параграфе</p>

</article>

<!-- Все закрывающие теги корректно синхронизированы -->

Практическое задание 3

Попрактикуем со сложными структурами:

Создайте сложный HTML:

```
html
<div class="header">
<nav id="main-nav">
<ul class="menu">
<li><a href="#">Главная</a></li>
</ul>
</nav>
</div>
```

Поэкспериментируйте:

Измените `<div class="header">` на `<header class="header">`

Измените `<nav id="main-nav">` на `<div id="main-nav">`

Посмотрите, как меняются все парные теги

Работа с атрибутами

Важная особенность:

Расширение **не трогает атрибуты**, меняет только имя тега:

```
html
<!-- Было: -->
<div class="old-class">Текст</div>

<!-- Меняем <div> на <section> -->
<section class="old-class">Текст</section>
<!-- Класс остался прежним! -->
```

Практическое задание 4

Проверим сохранение атрибутов:

Создайте тег с несколькими атрибутами:

```
html
<div class="container" id="main" data-test="value">Текст</div>
```

Измените тег на другой

Убедитесь, что все атрибуты сохранились

Поддерживаемые языки

Auto Rename Tag работает с:

HTML — основной язык

XML — языки разметки

JSX — React компоненты

Vue — компоненты Vue.js

PHP — встроенный HTML

JavaScript/TypeScript — шаблонные строки

Пример с JSX (React):

```
jsx
// Было:
const Component = () => {
  return <div className="container">Content</div>;
};

// Меняем <div> на <section>:
const Component = () => {
  return <section className="container">Content</section>;
};
```

Практическое задание 5

Попробуем с другими языками:

Создайте XML файл (test.xml):

```
xml
<book>
  <title>Название</title>
  <author>Автор</author>
</book>
```

Измените теги и убедитесь, что работает

Настройки Auto Rename Tag

Доступ к настройкам:

Нажмите **Ctrl + ,**

В поиске введите: auto rename tag
Или откройте настройки расширения через панель расширений

Основные настройки:

```
json
{
  // Включить/выключить расширение
  "auto-rename-tag.enable": true,

  // Пары тегов, которые не переименовывать
  "auto-rename-tag.excludedTags": [
    "area", "base", "br", "col", "command",
    "embed", "hr", "img", "input", "keygen",
    "link", "meta", "param", "source", "track", "wbr"
  ],

  // Задержка перед переименованием (мс)
  "auto-rename-tag.activationOnEnter": true
}
```

Практическое задание 6

Настроим под себя:

Добавьте тег `textarea` в исключения
Проверьте, что он теперь не переименовывается
Верните настройки обратно

Горячие клавиши

Хотя чаще всего работает автоматически:

Принудительное переименование:

Нет специальных горячих клавиш — работает автоматически
Но можно использовать стандартные:
`F2` — переименовать (может работать с тегами)
`Ctrl + D` — выделить следующее вхождение

Практическое задание 7

Используем стандартные горячие клавиши:

Выделите открывающий тег

Нажмите F2

Ведите новое название

Нажмите Enter

Убедитесь, что оба тега изменились

Особые случаи

Самозакрывающиеся теги:

html

<!-- Эти теги не имеют пары, поэтому не переименовываются: -->

 → если изменить, ничего не произойдет

<input type="text">

Непарные теги:

html

<!-- Если тег не имеет закрывающего: -->

<div>Текст <!-- забыли закрыть -->

<!-- При изменении <div> расширение ничего не сделает -->

Практическое задание 8

Проверим особые случаи:

Создайте самозакрывающиеся теги

Попробуйте их изменить

Убедитесь, что расширение не реагирует

Работа с большими файлами

Производительность:

Расширение очень легкое

Не замедляет работу даже с большими файлами

Мгновенный отклик

Пример с большим файлом:

```
html
<!-- Даже в таком случае: -->
<div>
<!-- 1000 строк кода внутри -->
<div>
<!-- Еще 1000 строк -->
<div>Текст</div> <!-- Изменяете этот тег -->
</div>
</div>
<!-- Все парные теги на всех уровнях синхронизируются! -->
```

Решение проблем

Проблема 1: Не переименовывает теги

Решение:

Проверьте, что расширение установлено и включено

Убедитесь, что файл имеет правильное расширение (.html, .xml)

Перезапустите VS Code

Проблема 2: Переименовывает неправильно

Решение:

Проверьте, нет ли синтаксических ошибок

Убедитесь, что теги правильно закрыты

Попробуйте в простом файле

Проблема 3: Замедляет работу

Решение:

Такое бывает крайне редко

Проверьте другие расширения

Попробуйте отключить и снова включить

Проблема 4: Не работает с определенным языком

Решение:

Проверьте, поддерживается ли язык

Установите расширение для этого языка

Или сообщите разработчику

Полезные комбинации

C Auto Close Tag:

Эти два расширения отлично работают вместе:

Auto Close Tag — автоматически закрывает теги

Auto Rename Tag — автоматически переименовывает парные теги

Установите оба:

json

В магазине расширений:

1. Auto Close Tag
2. Auto Rename Tag

Практическое задание 9

Установим и протестируем вместе:

Установите Auto Close Tag

Создайте новый тег: `<div>`

Auto Close Tag закроет его: `<div></div>`

Измените тег: `<div>` → `<section>`

Auto Rename Tag изменит закрывающий: `</div>` → `</section>`

Профессиональные сценарии использования

Рефакторинг HTML:

Задача: Изменить все `<div class="card">` на `<article class="card">`

Без расширения: Менять каждый тег вручную (опасно ошибиться)

С расширением:

Найдите первый `<div class="card">`

Меняете на `<article>`

Закрывающий меняется автоматически

Переходите к следующему

Миграция верстки:

Старая верстка:

html

```
<div class="header">...</div>
<div class="main">...</div>
<div class="footer">...</div>
```

Новая семантическая верстка:

```
html
<header>...</header>
<main>...</main>
<footer>...</footer>
```

Чек-лист освоения

Установил Auto Rename Tag

Понимаю, как работает автоматическое переименование

Могу работать с простыми тегами

Работаю со сложными вложенными структурами

Знаю, что атрибуты сохраняются

Работаю с разными языками (HTML, XML, JSX)

Настроил под себя

Знаю, как решать проблемы

Использую в комбинации с Auto Close Tag

Применяю в профессиональных сценариях

Итоговое упражнение

Реальный пример рефакторинга:

Создайте HTML-структуру блога:

```
html
<div class="blog">
<div class="post">
<div class="post-title">Заголовок</div>
<div class="post-content">Текст статьи</div>
</div>
</div>
```

Рефакторите на семантические теги:

```
<div class="blog"> → <main>
<div class="post"> → <article>
<div class="post-title"> → <h2>
<div class="post-content"> → <div> (оставить как есть)
```

Используйте Auto Rename Tag для каждого изменения

Поздравляю! Теперь вы владеете Auto Rename Tag — незаменимым инструментом для современного веб-разработчика. Это

расширение сэкономит вам часы работы и избавит от множества ошибок. Помните: хорошие инструменты делают хорошего разработчика еще лучше!

Глава 11: Emmet - верстка на скорости

Что такое Emmet и как он ускоряет работу

Дорогой друг!

Давайте познакомимся с **Emmet** — настоящим волшебством для веб-разработчиков! Это инструмент, который позволяет писать HTML и CSS с невероятной скоростью. Представьте, что вы можете написать сложную HTML-структуру всего несколькими символами!

Что такое Emmet?

Emmet — это плагин (встроенный в VS Code!), который:

Превращает короткие аббревиатуры в полноценный HTML/CSS код

Ускоряет верстку в **10-20 раз**

Понимает сложные селекторы и вложенности

Работает "из коробки" — ничего устанавливать не нужно!

Как выглядит магия Emmet

Без Emmet:

Вы пишете вручную:

```
html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<body>

</body>
</html>
```

С Emmet:

Вы пишете: !
Нажимаете: Tab
И получаете всю структуру мгновенно! ✨

Практическое задание 1

Попробуем первую магию Emmet:

Создайте новый файл emmet-test.html
Ведите: !
Нажмите: Tab (или Enter)
Наблюдайте чудо — появилась полная HTML5 структура!

Базовые аббревиатуры Emmet

Создание тегов:

Одиночные теги:

text
div + Tab = <div></div>
p + Tab = <p></p>
span + Tab =

Теги с ID:

text
div#container + Tab = <div id="container"></div>
p#main-text + Tab = <p id="main-text"></p>

Теги с классом:

text
div.header + Tab = <div class="header"></div>
p.text-center + Tab = <p class="text-center"></p>

Практическое задание 2

Попрактикуем базовые аббревиатуры:

Создайте несколько тегов с ID
Создайте несколько тегов с классами
Экспериментируйте с разными тегами

Комбинирование классов и ID

Несколько классов:

text

```
div.header.main.container + Tab =  
<div class="header main container"></div>
```

Классы и ID вместе:

text

```
div#main.container.wrapper + Tab =  
<div id="main" class="container wrapper"></div>
```

Сложные комбинации:

text

```
ul#nav.menu.horizontal + Tab =  
<ul id="nav" class="menu horizontal"></ul>
```

Практическое задание 3

Создадим навигационное меню:

Ведите: nav#main-nav.menu.horizontal + Tab

Внутри добавьте: ul>li*5>a + Tab

Наблюдайте как создается полноценное меню

Вложенность элементов

Символы вложенности:

> — дочерний элемент

+ — соседний элемент

^ — подняться на уровень выше

Примеры:

text

```
div>p + Tab =  
<div>  
<p></p>
```

```
</div>

text
div>p>span + Tab =
<div>
<p>
<span></span>
</p>
</div>
```

Практическое задание 4

Создадим сложную структуру:

Ведите: `div.container>header+main+footer + Tab`
Посмотрите как создается семантическая структура
Добавьте вложенность: `header>nav>ul>li*3`

Умножение элементов

Символ умножения *:

Создание нескольких одинаковых элементов:

```
text
li*5 + Tab =
<li></li>
<li></li>
<li></li>
<li></li>
<li></li>
```

С нумерацией:

```
text
li.item$*3 + Tab =
<li class="item1"></li>
<li class="item2"></li>
<li class="item3"></li>
```

Практическое задание 5

Создадим список товаров:

Ведите: `ul.products>li.product-item$*5>h3{Товар $}+p{Описание товара $}` + Tab

Наблюдайте как создается каталог товаров с нумерацией

Измените количество товаров на 10

Добавление текста

Фигурные скобки {}:

Добавляем текст:

`text`

`p{Привет, мир!} + Tab = <p>Привет, мир!</p>`

Текст с умножением:

`text`

`p{Параграф $}*3 + Tab =`

`<p>Параграф 1</p>`

`<p>Параграф 2</p>`

`<p>Параграф 3</p>`

Практическое задание 6

Создадим статью с заголовками:

Ведите: `article>h1{Заголовок}+p{Введение}+h2{Подзаголовок}`

`+p{Основной текст} + Tab`

Добавьте несколько параграфов с разным текстом

Атрибуты в квадратных скобках

Добавление любых атрибутов:

Ссылки:

`text`

`a[href="#"]{Перейти} + Tab = Перейти`

Изображения:

`text`

`img[src="photo.jpg" alt="Описание"] + Tab =`

```

```

Формы:

text

```
input[type="text" placeholder="Введите имя"] + Tab =  
<input type="text" placeholder="Введите имя">
```

Практическое задание 7

Создадим форму регистрации:

Ведите: form>input[type="text" placeholder="Имя"]+input[type="email" placeholder="Email"]+input[type="password" placeholder="Пароль"] +button[type="submit"]{Отправить} + Tab

Убедитесь, что все атрибуты на месте

CSS с Emmet

Emmet работает и с CSS!

Свойства CSS:

text

```
m10 + Tab = margin: 10px;  
p20 + Tab = padding: 20px;  
bgc#f00 + Tab = background-color: #f00;
```

Полные правила:

text

```
.w100+h100+m10+p20 + Tab =  
width: 100px;  
height: 100px;  
margin: 10px;  
padding: 20px;
```

Практическое задание 8

Напишем CSS с Emmet:

Создайте файл style.css

Ведите: .container{m0+p20+bgc#333+c#fff} + Tab

Посмотрите как разворачивается CSS:

CSS

```
.container {  
margin: 0;  
padding: 20px;  
background-color: #333;  
color: #fff;  
}
```

Популярные шаблоны

Готовые структуры:

Карточка товара:

text

```
.card>img[src="product.jpg"]+h3{Название}+p{Описание}  
+span.price{1000₽} + Tab
```

Навигационное меню:

text

```
nav>ul>li*5>a[href="#" "{Пункт $} + Tab
```

Сетка Bootstrap-стиль:

text

```
.container>.row>.col-md-4*3>p{Колонка $} + Tab
```

Практическое задание 9

Создадим блочную структуру сайта:

Ведите: header#header>nav>ul>li*5>a{Ссылка
\$}^^main#main>.container>.row>.col*3>p{Контент
\$}^^footer#footer{p{Футер}} + Tab

Анализируйте сложную структуру, созданную одной строкой

Настройка Emmet в VS Code

Emmet уже настроен, но можно улучшить:

```
json
{
  // Включить Emmet для всех языков
  "emmet.includeLanguages": {
    "javascript": "html",
    "vue": "html",
    "php": "html"
  },
  // Настройки триггеров
  "emmet.triggerExpansionOnTab": true,
  // Предпросмотр аббревиатур
  "emmet.showAbbreviationSuggestions": true,
  // Синтаксис Emmet в CSS
  "emmet.syntaxProfiles": {
    "css": "scss"
  }
}
```

Горячие клавиши Emmet

Основные команды:

Tab — развернуть аббревиатуру
Ctrl + Space — показать подсказки Emmet
Ctrl + U — обновить теги (после переименования)
Ctrl + D — перейти к следующему редактируемому полю

Дополнительные команды:

Wrap with Abbreviation — обернуть выделенный текст в тег
Balance outward/inward — выделить тег/выйти из тега

Продвинутые возможности

Группировка скобками ():

Сложные структуры:

text
(header>nav)+(main>section*3)+(footer>p) + Tab

Это создаст:

html
<header>
<nav></nav>
</header>
<main>
<section></section>
<section></section>
<section></section>
</main>
<footer>
<p></p>
</footer>

Счетчик с шагом:

Нумерация с шагом 2:

text
li.item\$@2*3 + Tab =
<li class="item2">
<li class="item4">
<li class="item6">

Обратная нумерация:

text
li.item\$@-*3 + Tab =
<li class="item3">
<li class="item2">
<li class="item1">

Реальные примеры экономии времени

Пример 1: Создание таблицы

Без Emmet: 2-3 минуты ручного набора

С Emmet: 10 секунд

text

```
table>thead>tr>th*3{Заголовок $}^^tbody>tr*5>td*3{Ячейка $} + Tab
```

Пример 2: Форма обратной связи

Без Emmet: 1-2 минуты

С Emmet: 15 секунд

text

```
form.contact-form>label{Имя}+input[type="text"]+label{Email}  
+input[type="email"]+label{Сообщение}+textarea+button[type="submit"]  
{Отправить} + Tab
```

Чек-лист освоения Emmet

Понимаю, что такое Emmet и зачем он нужен

Умею создавать базовые теги (div, p, span)

Добавляю классы и ID (.class, #id)

Создаю вложенные структуры (>)

Использую умножение элементов (*)

Добавляю текст в фигурных скобках ({})

Задаю атрибуты в квадратных скобках ([])

Пишу CSS с помощью Emmet

Использую готовые шаблоны

Знаю горячие клавиши

Использую продвинутые возможности (группировка, счетчики)

Практический экзамен

Создайте страницу интернет-магазина за 2 минуты:

Шапка с логотипом и навигацией

Главный баннер с заголовком

Сетка товаров (3x3)

Форма подписки на рассылку

Футер с контактами
Используйте Emmet для каждого блока!

Подсказка: Разбейте на части:

```
text
header>div.logo{Лого}+nav>ul>li*5>a{Меню $}
section.hero>h1{Магазин}+p{Описание}
div.products>.product*9>img+h3+span
form.subscribe>input+button
footer>p{Контакты}
```

Профессиональные советы

Начинайте с простого — освойте базовые абреквиатуры
Создавайте свои сниппеты для часто используемых структур
Используйте предпросмотр (**Ctrl + Space**) чтобы увидеть что получится
Комбинируйте Emmet с другими расширениями (Auto Rename Tag)
Практикуйтесь ежедневно — скорость придет с опытом

Что делать если не работает

Проверьте:

Файл имеет правильное расширение (.html для HTML, .css для CSS)
Язык файла установлен правильно (в правом нижнем углу)
Emmet включен в настройках
Вы используете правильный синтаксис

Поздравляю! Теперь вы владеете Emmet — супер-оружием веб-разработчика. С ним вы будете верстать быстрее, точнее и с меньшим количеством ошибок. Помните: мастерство Emmet приходит с практикой — чем больше используете, тем лучше получается!

Основные команды Emmet для HTML

Дорогой друг!

Давайте подробно изучим основные команды Emmet для HTML. Это ваш «волшебный словарь», который превращает несколько символов в полноценные HTML-структуры! Запомните эти команды, и вы сможете верстать с невероятной скоростью.

Базовый синтаксис Emmet

Структура команд:

text

элемент#ид.класс[атрибут=значение]{текст}*количество

1. Создание элементов (тегов)

Простейшие теги:

emmet

div → <div></div>

p → <p></p>

span →

a →

img →

Как использовать:

Вводите абривиатуру

Нажимаете Tab

Получаете готовый тег!

Практическое задание 1

Создайте базовые теги:

div + Tab

p + Tab

span + Tab

a + Tab

img + Tab

2. Классы (.) и ID (#)

Добавление классов:

emmet

```
.container → <div class="container"></div>  
p.text → <p class="text"></p>  
.header.main → <div class="header main"></div>
```

Добавление ID:

emmet

```
#main → <div id="main"></div>  
section#hero → <section id="hero"></section>
```

Комбинации классов и ID:

emmet

```
div#header.container → <div id="header" class="container"></div>  
form#contact.contact-form → <form id="contact"  
class="contact-form"></form>
```

Практическое задание 2

Создайте элементы с классами и ID:

```
.header → Tab  
#sidebar → Tab  
nav#main.menu → Tab  
button.btn.btn-primary → Tab
```

3. Вложенность элементов (>)

Дочерние элементы:

emmet

```
div>p →  
<div>  
<p></p>  
</div>
```

```
ul>li →
```

```
<ul>  
<li></li>  
</ul>
```

Многоуровневая вложенность:

```
emmet  
div>ul>li →  
<div>  
<ul>  
<li></li>  
</ul>  
</div>
```

Сложные структуры:

```
emmet  
nav>ul.menu>li.item*5 →  
<nav>  
<ul class="menu">  
<li class="item"></li>  
<li class="item"></li>  
<li class="item"></li>  
<li class="item"></li>  
<li class="item"></li>  
</ul>  
</nav>
```

Практическое задание 3

Создайте вложенные структуры:

```
div>span → Tab  
section>article>p → Tab  
header>nav>ul>li*3 → Tab  
form>label+input+button → Tab
```

4. Соседние элементы (+)

Элементы на одном уровне:

```
emmet  
div+p+span →  
<div></div>
```

```
<p></p>  
<span></span>
```

Комбинация вложенности и соседства:

```
emmet  
div>h1+p →  
<div>  
<h1></h1>  
<p></p>  
</div>
```

Сложные комбинации:

```
emmet  
header+main+footer →  
<header></header>  
<main></main>  
<footer></footer>
```

Практическое задание 4

Создайте соседние элементы:

```
h1+p+div → Tab  
div.container>h1+p → Tab  
header+main.content+footer → Tab
```

5. Умножение элементов (*)

Создание нескольких одинаковых элементов:

```
emmet  
li*3 → Tab  
<li></li>  
<li></li>  
<li></li>
```

С классом:

```
emmet  
div.item*4 → Tab  
<div class="item"></div>
```

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

Практическое задание 5

Умножьте элементы:

```
p*5 → Tab
div.box*3 → Tab
ul>li.item*4 → Tab
```

6. Нумерация элементов (\$)

Автоматическая нумерация:

```
emmet
.item$*3 →
<div class="item1"></div>
<div class="item2"></div>
<div class="item3"></div>
```

Нумерация в тексте:

```
emmet
p{Пункт $}*3 →
<p>Пункт 1</p>
<p>Пункт 2</p>
<p>Пункт 3</p>
```

Двойные цифры:

```
emmet
.item$$*3 →
<div class="item01"></div>
<div class="item02"></div>
<div class="item03"></div>
```

Обратная нумерация:

```
emmet
.item$@-*3 →
<div class="item3"></div>
<div class="item2"></div>
```

```
<div class="item1"></div>
```

Практическое задание 6

Поэкспериментируйте с нумерацией:

```
li.item$*5 → Tab  
h3{Заголовок $}*4 → Tab  
div.section$$*3 → Tab  
span.number$@-*5 → Tab
```

7. Текст внутри элементов {}

Добавление текста:

emmet

```
p{Привет, мир!} → <p>Привет, мир!</p>
```

Текст с классами:

emmet

```
.btn{Нажми меня} → <button class="btn">Нажми меня</button>
```

Текст с нумерацией:

emmet

```
li{Пункт $}*4 →  
<li>Пункт 1</li>  
<li>Пункт 2</li>  
<li>Пункт 3</li>  
<li>Пункт 4</li>
```

Практическое задание 7

Добавьте текст элементам:

```
h1{Главный заголовок} → Tab  
a{Перейти на сайт} → Tab  
p{Абзац $}*3 → Tab
```

8. Атрибуты элементов []

Атрибуты ссылок:

emmet

a[href="#" →

a[href="about.html" target="_blank"] → <a href="about.html"

target="_blank">

Атрибуты изображений:

emmet

img[src="photo.jpg" alt="Описание"] →

Атрибуты форм:

emmet

input[type="text" placeholder="Имя"] → <input type="text"
placeholder="Имя">

Практическое задание 8

Добавьте атрибуты:

img[src="logo.png" alt="Логотип"] → Tab

a[href="contact.html" title="Контакты"]{Связь} → Tab

input[type="email" required placeholder="Email"] → Tab

9. Поднятие на уровень выше (^)

Выход из вложенности:

emmet

div>p>span^h2 →

<div>

<p></p>

<h2></h2>

</div>

Несколько уровней:

```
emmet
div>p>span^^h1 →
<div>
<p><span></span></p>
</div>
<h1></h1>
```

Практическое задание 9

Поэкспериментируйте с ^:

```
div>p>span^h2 → Tab
section>article>h1+p^^footer → Tab
```

10. Группировка элементов ()

Создание групп:

```
emmet
(header>nav)+(main>section)+(footer>p) →
<header>
<nav></nav>
</header>
<main>
<section></section>
</main>
<footer>
<p></p>
</footer>
```

Сложные группировки:

```
emmet
div>(header>h1)+(section>p*3)+footer →
<div>
<header>
<h1></h1>
</header>
<section>
<p></p>
<p></p>
<p></p>
</section>
```

```
<footer></footer>  
</div>
```

Практическое задание 10

Создайте группы элементов:

```
(div>p)+(div>span) → Tab  
section>(header>h1)+(main>article)+(footer>p) → Tab
```

Популярные шаблоны (готовые рецепты)

HTML5 структура:

emmet
! → Полная HTML5 страница

Навигационное меню:

```
emmet  
nav>ul>li*5>a[href="#"]{Пункт $} →  
<nav>  
<ul>  
<li><a href="#">Пункт 1</a></li>  
<li><a href="#">Пункт 2</a></li>  
<li><a href="#">Пункт 3</a></li>  
<li><a href="#">Пункт 4</a></li>  
<li><a href="#">Пункт 5</a></li>  
</ul>  
</nav>
```

Форма входа:

```
emmet  
form.login>input[type="text" placeholder="Логин"]+input[type="password"  
placeholder="Пароль"]+button[type="submit"]{Войти} →  
<form class="login">  
<input type="text" placeholder="Логин">  
<input type="password" placeholder="Пароль">  
<button type="submit">Войти</button>  
</form>
```

Карточка товара:

еммет

```
.card>img[src="product.jpg"]+h3{Название}+p{Описание}  
+span.price{1000₽}+button{Купить} →  
<div class="card">  
  
<h3>Название</h3>  
<p>Описание</p>  
<span class="price">1000₽</span>  
<button>Купить</button>  
</div>
```

Таблица:

еммет

```
table>thead>tr>th*3{Заголовок $}^^tbody>tr*3>td*3{Данные $} →  
<table>  
<thead>  
<tr>  
<th>Заголовок 1</th>  
<th>Заголовок 2</th>  
<th>Заголовок 3</th>  
</tr>  
</thead>  
<tbody>  
<tr>  
<td>Данные 1</td>  
<td>Данные 2</td>  
<td>Данные 3</td>  
</tr>  
<tr>  
<td>Данные 1</td>  
<td>Данные 2</td>  
<td>Данные 3</td>  
</tr>  
<tr>  
<td>Данные 1</td>  
<td>Данные 2</td>  
<td>Данные 3</td>  
</tr>  
</tbody>
```

</table>

Комбинирование всех возможностей

Полная страница за 30 секунд:

emmet

```
!+(header>nav>ul>li*5>a[href="#"]{Меню $})+  
(main.container>(section*3>h2{Раздел $}+p*2))+footer{© 2024}
```

Практическое задание 11

Создайте страницу блога:

emmet

```
!+header#header>h1{Мой блог}+nav>a[href="#"]{Главная}+a[href="#"]{Обо  
мне}+a[href="#"]{Контакты}^^main>article*3>h2{Статья $}+p{Текст статьи  
$}+a[href="#"]{Читать далее}^^footer>p{© 2024 Мой блог}
```

Шпаргалка-памятка

Символы Emmet:

.	— класс
#	— id
>	— дочерний элемент
+	— соседний элемент
*	— умножение
\$	— нумерация
{}	— текст
[]	— атрибуты
^	— подняться на уровень
()	— группировка

Частые ошибки новичков:

Пробелы — Emmet не любит пробелы в аббревиатурах

Порядок символов — сначала тег, потом классы/ID

Не нажимать Tab — забывают нажать Tab для разворачивания

Неправильный язык файла — проверьте в правом нижнем углу

Профессиональные советы:

Начните с простого — освойте базовые команды
Создайте свою шпаргалку — распечатайте и держите рядом
Практикуйте ежедневно — скорость придет с опытом
Используйте готовые шаблоны — не изобретайте велосипед
Комбинируйте команды — создавайте сложные структуры

Чек-лист освоения Emmet

Умею создавать простые теги
Добавляю классы и ID
Создаю вложенные структуры (>)
Добавляю соседние элементы (+)
Умножаю элементы (*)
Использую нумерацию (\$)
Добавляю текст в {}
Задаю атрибуты в []
Выхожу из вложенности (^)
Группирую элементы ()
Использую готовые шаблоны
Могу создать полную страницу

Итоговое упражнение

Создайте страницу портфолио за 1 минуту:

Шапка с именем и профессией
Навигация по разделам
Секция «Обо мне»
Портфолио (3 работы)
Контакты (форма и соцсети)
Футер
Используйте Emmet для каждого блока!

Готовый код для проверки:

```
emmet
!+header#hero>h1{Имя Фамилия}+p{Frontend разработчик}
+nav>a[href="#about">{Обо мне}+a[href="#portfolio">{Портфолио}
+a[href="#contacts">{Контакты}^^section#about>h2{Обо мне}+p{Текст обо
мне}^^section#portfolio>h2{Портфолио}
+div.works>article.work*3>img[src="work$.jpg"]+h3{Проект $}
+p{Описание}^^section#contacts>h2{Контакты}+form>input[type="text"
placeholder="Имя"]+input[type="email" placeholder="Email"]
```

```
+textarea[placeholder="Сообщение"]+button{Отправить}  
+div.social>a[href="#"]{Twitter}+a[href="#"]{GitHub}+a[href="#"]  
{LinkedIn}^^footer>p{© 2024 Все права защищены}
```

Поздравляю! Теперь вы владеете основными командами Emmet для HTML. Это мощный инструмент, который сделает вашу верстку быстрой и точной. Практикуйтесь каждый день, и скоро эти команды станут вашей второй натурой!

Практические примеры быстрой верстки

Дорогой друг!

Давайте перейдем от теории к практике! Я покажу вам конкретные примеры, как использовать Emmet для быстрой верстки реальных элементов. Вы убедитесь, что сложные структуры можно создавать за секунды!

Пример 1: Карточка товара (интернет-магазин)

Что нам нужно:

text
[Картинка]
Название товара
Описание
Цена
Кнопка "Купить"

Без Emmet (ручная верстка):

```
html
<div class="card">

<h3 class="title">Название товара</h3>
<p class="description">Описание товара</p>
<span class="price">1000 руб.</span>
<button class="buy-btn">Купить</button>
</div>
```

Время: 1-2 минуты

С Emmet:

```
emmet
.card>img[src="product.jpg" alt="Название товара"]+h3.title{Название
товара}+p.description{Описание товара}+span.price{1000 руб.}+button.buy-
btn{Купить}
```

Время: 10 секунд!

Практическое задание 1

Создайте карточку товара:

Откройте новый HTML файл

Ведите Emmet код:

emmet

```
.card>img[src="product.jpg"]+h3.title{Смартфон}+p.description{Мощный  
смартфон с отличной камерой}+span.price{25 999 ₽}+button.buy-btn{В  
корзину}
```

Нажмите Tab

Посмотрите на готовую карточку!

Пример 2: Навигационное меню сайта

Без Emmet:

html

```
<nav class="main-nav">  
<ul class="menu">  
<li class="menu-item"><a href="index.html">Главная</a></li>  
<li class="menu-item"><a href="about.html">О нас</a></li>  
<li class="menu-item"><a href="services.html">Услуги</a></li>  
<li class="menu-item"><a href="portfolio.html">Портфолио</a></li>  
<li class="menu-item"><a href="contact.html">Контакты</a></li>  
</ul>  
</nav>
```

С Emmet:

emmet

```
nav.main-nav>ul.menu>li.menu-item*5>a[href="#"]{$}
```

И добавим текст:

emmet

```
nav.main-nav>ul.menu>li.menu-item*5>a[href="#"]{${Главная, О  
нас, Услуги, Портфолио, Контакты}}
```

Практическое задание 2

Создайте меню за 5 секунд:

Ведите: nav.main-nav>ul.menu>li.menu-item*5>a[href="#"]{\$}

Нажмите Tab

Заполните текст:

text

Главная
О компании
Услуги
Портфолио
Контакты

Пример 3: Форма регистрации

Полная форма регистрации:

emmet

```
form.registration-form>h2{Регистрация}+label[for="name"]{Имя}  
+input[type="text" id="name" placeholder="Введите имя"]+label[for="email"]  
{Email}+input[type="email" id="email" placeholder="Введите email"]  
+label[for="password"]{Пароль}+input[type="password" id="password"]  
placeholder="Введите пароль"]+label[for="confirm"]{Подтвердите пароль}  
+input[type="password" id="confirm" placeholder="Повторите пароль"]  
+button[type="submit"]{Зарегистрироваться}  
+p.agreement>input[type="checkbox" id="agree"]+label[for="agree"]  
{Согласен с условиями}
```

Упрощенная версия:

emmet

```
form.reg>h2{Регистрация}+(label{$}+input[type="$"  
placeholder="$"])*4{Имя,text,Введите имя,Email,email,Введите  
email,Пароль,password,Введите  
пароль,Подтверждение,password,Повторите пароль}+button{Отправить}
```

Практическое задание 3

Создайте форму входа:

emmet

```
form.login>h2{Вход}+input[type="text" placeholder="Логин"]  
+input[type="password" placeholder="Пароль"]+button{Войти}+a[href="#"]  
{Забыли пароль?}+p{Нет аккаунта? }>a[href="#"]{Зарегистрироваться}
```

Пример 4: Сетка карточек (Bootstrap-style)

3 колонки на больших экранах:

emmet

```
.container>.row>.col-md-4*3>.card>img[src="img$.jpg"]+h3{Заголовок $}  
+p{Текст карточки $}+a[href="#"]{Подробнее}
```

Адаптивная сетка:

emmet

```
.container>.row>(.col-sm-6.col-md-4>.card>img[src="img$.jpg"]+h3{Товар $}  
+p{Описание $}+span{999₽}+button{Купить})*6
```

Практическое задание 4

Создайте галерею товаров:

emmet

```
.gallery>.product*9>img[src="product$.jpg"]+h4{Товар $}+p{Описание  
товара $}+span.price{Цена: 1000₽}+button{В корзину}
```

Пример 5: Таблица с данными

Таблица пользователей:

emmet

```
table.user-table>thead>tr>th*4{$ID,Имя,Email,Дата  
регистрации}^^tbody>tr*5>td*4{$$,Пользователь $,user$@mail.com,2024-  
01-$}
```

Таблица заказов:

emmet

```
table.orders>thead>tr>th*5{$  
{№,Товар,Количество,Цена,Сумма}}^^tbody>(tr>td{@-}+td{Товар $}+td{$  
$}+td{1000₽}+td{ $$00₽})*10
```

Практическое задание 5

Создайте таблицу расписания:

emmet

```
table.schedule>thead>tr>th*6{$
{Понедельник,Вторник,Среда,Четверг,Пятница,Суббота}}^^tbody>tr*8>td*
6${$8:00-
9:30,Математика,Физика,Химия,Литература,История,Физкультура}}
```

Пример 6: Комментарии к статье

Древовидные комментарии:

emmet

```
.comments>(.comment>img[src="avatar$.jpg"]+.content>h4{Пользователь $}
+p{Текст комментария $}+.meta>span.time{2 часа назад}
+button.reply{Ответить}+(.reply*2>img[src="avatar$@5.jpg"]
+.content>h4{Автор ответа $}+p{Ответ на комментарий $}
+.meta>span.time{1 час назад}))*3
```

Практическое задание 6

Создайте блок комментариев:

emmet

```
.comments-section>h3{Комментарии (5)}+
(.comment*5>img.avatar[src="user$.jpg"]+div>h5{Имя пользователя $}
+p.comment-text{Текст комментария номер $}+.comment-
meta>span.date{Сегодня, 14:$0}+button.reply{Ответить})
```

Пример 7: Подвал сайта (footer)

Сложный футер с разделами:

emmet

```
footer.site-footer>.container>.row>.col-md-3*4>h5{$О
компании,Услуги,Контакты,Подписка}}+ul>li*3>a[href="#"]{$}+^.col-md-
12.text-center>p{© 2024 Компания. Все права защищены.}
+.social>a[href="#"[title="Facebook"]{FB}+a[href="#"[title="Twitter"]{TW}
+a[href="#"[title="Instagram"]{IG}
```

Практическое задание 7

Создайте подвал сайта:

emmet

```
footer>.footer-content>.column*3>h4{$(Навигация,Контакты,Соцсети)}  
+ul>li*3>a[href="#"]{$}+^^.copyright>p{© 2024 Мой сайт | }>a[href="#"  
{Политика конфиденциальности}
```

Пример 8: Блог-пост

Полная статья блога:

emmet

```
article.blog-post>header>h1{Заголовок статьи}+.meta>span.author{Автор:  
Иван Иванов}+span.date{Опубликовано: 15 января 2024}+span.read-  
time{Время чтения: 5 мин}+^^section.content>p*3{Абзац текста статьи.  
Здесь содержится основной контент. $}+blockquote{Важная цитата из  
статьи}+p{Продолжение текста после цитаты}+ul>li*4{Пункт списка $}  
+^^footer.tags>span.tag*5{$(JavaScript,HTML,CSS,Web,Разработка)}  
+share>button{Поделиться}
```

Практическое задание 8

Создайте шаблон статьи:

emmet

```
article>.post-header>h1{Моя первая статья}+.post-  
meta>span.date{Сегодня}+span.category{HTML}+^^.post-  
content>p{Введение в статью.}+h2{Первый раздел}+p{Текст раздела.}  
+code{console.log('Hello World')}+h2{Второй раздел}+p{Продолжение  
статьи.}+ul>li*3{Пункт $}+^^.post-footer>.tags>a.tag*3[href="#"{$}  
{HTML,CSS,JavaScript}}+.actions>button.like{Нравится}  
+button.share{Поделиться}
```

Пример 9: Панель управления (dashboard)

Элементы дашборда:

emmet

```
.dashboard>.sidebar>nav>ul>li*6>a[href="#">i.icon{★}+span{$  
{Главная,Статистика,Пользователи,Настройки,Помощь,Выход}}^^.main-
```

```
content>header>h1{Панель управления}+.stats>.stat-card*4>h3{$  
{Пользователи,Заказы,Доход,Конверсия}}+p.value{$$$}+p.change{+$  
%}^^.recent-orders>h2{Последние заказы}+table>thead>tr>th*4{$  
{ID,Клиент,Сумма,Статус}}^^tbody>tr*5>td{@100}+td{Клиент $}+td{$$$$}  
+td{${Обработан,В процессе,Доставляется,Выполнен,Отменен}}
```

Практическое задание 9

Создайте простой дашборд:

emmet

```
.dashboard>header>h1{Панель администратора}+nav>a[href="#"]  
{Статистика}+a[href="#"]{Пользователи}+a[href="#"]  
{Настройки}^^main>.widgets>.widget*3>h3{${Посетители,Продажи,Доход}}  
+p{$$$}+^^.table-container>table>thead>tr>th*4{$  
{Дата,Показатель,Значение,Изменение}}^^tbody>tr*5>td{2024-01-$}  
+td{Показатель $}+td{$$$}+td{+$%}
```

Пример 10: Модальное окно

Диалоговое окно:

emmet

```
.modal#myModal>.modal-dialog>.modal-content>.modal-header>h4.modal-  
title{Заголовок окна}+button.close{×}^^.modal-body>p{Содержимое  
модального окна. Здесь может быть текст, форма или другие элементы.}  
+form>input[type="text" placeholder="Введите данные"]  
+button[type="submit"]{Отправить}^^.modal-footer>button.btnCancel{Отмена}+button.btnSave{Сохранить}
```

Практическое задание 10

Создайте окно подтверждения:

emmet

```
.modal.confirm>.modal-content>.modal-header>h3{Подтверждение  
действия}^^.modal-body>p{Вы уверены, что хотите выполнить это  
действие?}^^.modal-footer>button.cancel{Отмена}  
+button.confirm{Подтвердить}
```

Секретный прием: Генерация lorem ipsum

Автоматический текст:

emmet

p>lorem10 → 10 слов

p>lorem50 → 50 слов

div.container>p*3>lorem20 → 3 параграфа по 20 слов

С заголовками:

emmet

article>h2{Раздел \$}+p>lorem30+h3{Подраздел}+p>lorem20

Практическое задание 11

Создайте страницу с демо-контентом:

emmet

!+main>.container>article*3>h2{Статья \$}

+p>lorem50+ul>li*4>lorem5+button{Читать далее}

Профессиональный рабочий процесс

Как верстать целую страницу за 5 минут:

Структура: ! → Tab

Header: header>nav>ul>li*5>a[href="#"]{\$}

Hero секция: section.hero>h1{Заголовок}+p{Подзаголовок}
+button{Кнопка}

Особенности: section.features>.feature*3>h3{Фича \$}+p{Описание}

Контент: section.content>article*2>h2{Статья \$}+p>lorem40

Футер: footer>p{© 2024}+.links>a[href="#"*3{\$}{Главная, О нас, Контакты}}

Чек-лист скорости верстки

Научитесь создавать за 30 секунд:

Навигационное меню

Форму входа/регистрации

Карточку товара

Таблицу данных

Сетку карточек
Блог-пост
Подвал сайта
Модальное окно
Панель статистики
Полную HTML5 страницу

Типичные ошибки и как их избежать

Забыли Tab — Emmet не сработает
Пробелы в аббревиатуре — `div .class` не работает, нужно `div.class`
Неправильный порядок — сначала тег, потом классы: `div.class`, а не `.class div`
Файл не HTML — проверьте расширение файла

Полезные советы для максимальной скорости

Создайте шпаргалку — распечатайте основные команды
Тренируйтесь ежедневно — скорость приходит с практикой
Начните с простого — сначала базовые команды, потом сложные
Используйте автодополнение — VS Code подсказывает Emmet-аббревиатуры
Комбинируйте с другими инструментами — Live Server для просмотра

Итоговый вызов

Создайте лендинг за 3 минуты:

```
emmet
!+header.site-header>nav>ul.menu>li*5>a[href="#"]{$
{Главная,Услуги,Портфолио,Отзывы,Контакты}}^^section.hero>h1{Креатив
ные решения}+p{Мы создаем уникальные веб-сайты}
+button.cta{Заказать}^^section.features>.container>.row>.col-md-
4*3>.feature>h3${Дизайн,Разработка,Поддержка}}+p{Качественное
исполнение $}^^section.portfolio>.project*6>img[src="project$.jpg"]
+h4{Проект $}+p{Описание
проекта}^^section.testimonials>.testimonial*3>p.review{Отличная работа!}
+span.author{Клиент $}^^section.contact>form>input[type="text"
placeholder="Имя"]+input[type="email" placeholder="Email"]
+textarea[placeholder="Сообщение"]+button{Отправить}^^footer.site-
footer>.container>.row>.col-md-4*3>h5${Компания,Услуги,Контакты}}
+ul>li*3>a[href="#"]{$}+^^.col-md-12>p{© 2024 Все права защищены}
```

Поздравляю! Теперь вы настоящий мастер быстрой верстки с Emmet. Эти примеры показывают, как можно создавать сложные структуры за секунды. Практикуйтесь, экспериментируйте и скоро вы будете удивлять коллег скоростью своей работы!

Часть 5: Отладка и проверка кода

Глава 12: Подсказки и обнаружение ошибок

Как VS Code подсвечивает ошибки

Дорогой друг!

Давайте подробно разберем, как VS Code помогает находить и исправлять ошибки. Это как иметь личного учителя, который всегда рядом и подсказывает, где вы ошиблись!

Типы подсветки ошибок в VS Code

VS Code использует **три уровня индикации ошибок**:

-  **Красная волнистая линия** — Критическая ошибка
-  **Желтая волнистая линия** — Предупреждение
-  **Лампочка на полях** — Быстрое исправление

Примеры ошибок в реальном времени

1. Ошибки в HTML

Пример 1: Незакрытый тег

```
html
<div>
<p>Текст
</div>
Под <p> появится красная волнистая линия
Сообщение: 'p' is not closed
```

Пример 2: Неправильный вложенный тег

```
html
<p>
<div>Текст</div>
</p>
Под <div> появится желтая волнистая линия
Сообщение: 'div' tag cannot be inside 'p' tag
```

Практическое задание 1

Создадим HTML с ошибками:

Создайте файл errors.html

Ведите код с ошибкой:

```
html
<!DOCTYPE html>
<html>
<body>
<div>
<p>Текст без закрытия
</div>
</body>
</html>
```

Найдите красную волнистую линию

Наведите курсор на ошибку — увидите подсказку

2. Ошибки в CSS

Пример 1: Несуществующее свойство

```
CSS
.element {
  colr: red; /* Опечатка! Должно быть color */
}
```

colr подчеркнется **красным**

Сообщение: Unknown property: 'colr'

Пример 2: Пропущенная точка с запятой

```
CSS
.element {
  color: red /* Нет точки с запятой! */
  margin: 10px;
}
```

После red появится **красная линия**

Сообщение: Expected ','

Практическое задание 2

Найдем CSS ошибки:

Создайте файл style.css

Ведите с ошибками:

```
css
.box {
widht: 100px; /* Опечатка */
background-colour: blue; /* Британский английский */
margin: 10px /* Нет ; */
padding: 20px;
}
```

Найдите все три ошибки

Исправьте их

3. Ошибки в JavaScript

Пример 1: Неопределенная переменная

```
javascript
console.log(myVar); // myVar не объявлена
myVar подчеркнется красным
Сообщение: 'myVar' is not defined
```

Пример 2: Синтаксическая ошибка

```
javascript
function test() {
return "Hello"
// Нет точки с запятой или закрывающей фигурной скобки
```

Появится красная линия

Сообщение: Expected ';' или Expected '}'

Практическое задание 3

Найдем JavaScript ошибки:

Создайте файл script.js

Ведите:

```
javascript
function sayHello() {
  console.log("Hello, " + name);

var x = 10
var y = 20
console.log(x + y)
```

Найдите 3 ошибки

Панель "Проблемы" (Problems Panel)

Централизованный просмотр всех ошибок:

Как открыть:

Ctrl + Shift + M (M = Mistakes = Ошибки)

Или в меню: View → Problems

Или кликнуть на индикатор ошибок в статус-баре

Как выглядит панель:

```
text
📁 ПРОБЛЕМЫ (3)
  ▼ errors.html (2)
    ✗ Отсутствует закрывающий тег (строка 5)
    ⚠ Неправильное использование тега (строка 8)
  ▼ style.css (1)
    ✗ Неизвестное CSS свойство (строка 3)
```

Практическое задание 4

Используем панель Проблемы:

Создайте файлы с ошибками

Откройте панель Ctrl + Shift + M

Посмотрите список всех ошибок

Кликните на ошибку — перейдете к строке

Быстрые исправления (Quick Fixes)

Лампочка  — ваш лучший помощник!

Как это работает:

VS Code видит ошибку

Рядом с номером строки появляется **лампочка** 

Кликаете на лампочку

Выбираете вариант исправления

Примеры быстрых исправлений:

Для HTML:

html

 <!-- Нет alt атрибута -->

Лампочка предложит: Add missing attribute 'alt'

Результат:

Для CSS:

CSS

.colr: red; /* Опечатка */

Лампочка предложит: Change spelling to 'color'

Результат: color: red;

Практическое задание 5

Используем быстрые исправления:

Создайте CSS с ошибкой:

CSS

```
div {  
bakground: blue;  
}
```

Найдите лампочку  рядом с номером строки

Кликните и выберите исправление

Убедитесь, что ошибка исправлена

Интеллектуальные подсказки

VS Code предупреждает об потенциальных проблемах:

Пример 1: Неиспользуемая переменная

```
javascript
function test() {
    var unused = 10; // Нигде не используется
    return "Hello";
}
unused подчеркнется серым пунктиром
Сообщение: 'unused' is declared but its value is never read
```

Пример 2: Устаревший метод

```
javascript
document.write("Hello"); // Устаревший метод
write подчеркнется желтым пунктиром
Сообщение: 'document.write' is deprecated
```

Настройки подсветки ошибок

Персонализация под себя:

```
json
{
    // Цвета подсветки ошибок
    "editorError.foreground": "#ff0000",
    "editorWarning.foreground": "#ff9900",
    "editorInfo.foreground": "#007acc",

    // Показывать ошибки при наборе (не только при сохранении)
    "editor.liveErrors": true,

    // Задержка перед проверкой (мс)
    "editor.liveErrors.delay": 500,

    // Подчеркивание ошибок
}
```

```
"editor.occurrencesHighlight": true,  
  
// Показывать панель проблем автоматически  
"problems.autoReveal": true  
}
```

Практическое задание 6

Настроим подсветку:

Откройте настройки **Ctrl + ,**
Найдите **error** в поиске
Измените цвет ошибок на более заметный
Проверьте, как изменилось отображение

Расширения для улучшения проверки

Дополнительные инструменты:

ESLint — проверка JavaScript/TypeScript
HTMLHint — проверка HTML
Stylelint — проверка CSS
Prettier — автоматическое форматирование

Установка ESLint:

Ctrl + Shift + X
Поиск: **ESLint**
Установите от Microsoft
Перезапустите VS Code

Практическое задание 7

Установим HTMLHint:
Установите расширение HTMLHint
Создайте HTML с ошибками
Посмотрите, как расширение добавляет новые проверки

Работа с несколькими языками

В одном файле могут быть ошибки разных типов:

HTML со встроенным CSS и JavaScript:

```
html
<!DOCTYPE html>
<html>
<head>
<style>
.box {
color: red; /* CSS ошибка */
}
</style>
</head>
<body>
<div> <!-- HTML ошибка: нет закрывающего тега -->
<script>
console.log(undefinedVar); // JS ошибка
</script>
</div>
</html>
```

CSS ошибка в `<style>`

HTML ошибка в `<div>`

JavaScript ошибка в `<script>`

Практическое задание 8

Найдите все ошибки в смешанном файле:

```
html
<!DOCTYPE html>
<html>
<head>
<style>
.container {
width: 100px
/* Нет ;
}
</style>
</head>
<body>
<div class="container">
<p>Текст
<!-- Незакрытый тег -->
```

```
<script>
var x = 10
console.log(y); // y не определена
</script>
</div>
</body>
</html>
```

Иерархия серьезности ошибок

VS Code ранжирует ошибки:

-  **Ошибка (Error)** — Код не будет работать
-  **Предупреждение (Warning)** — Работает, но могут быть проблемы
-  **Совет (Hint)** — Можно улучшить
-  **Информация (Info)** — Дополнительная информация

Примеры каждой категории:

```
javascript
// ✗ ОШИБКА: Синтаксическая ошибка
function test() {

// ⚠ ПРЕДУПРЕЖДЕНИЕ: Устаревший метод
element.innerHTML = "<div>test</div>";

//💡 СОВЕТ: Можно упростить
if (x === true) { ... }

//ℹ ИНФОРМАЦИЯ: Тип переменной
var name = "Иван"; // string
```

Отладка сложных ошибок

Метод "разделяй и властвуй":

Когда много ошибок:

Закомментируйте часть кода (Ctrl + /)
Посмотрите, какие ошибки исчезли
Постепенно раскомментируйте, находя источник

Пример:

```
html
<!-- Было: -->
<div>
<p>Текст <!-- Ошибка 1 -->
<span>Еще текст</span>
</div>
 <!-- Ошибка 2 -->

<!-- Закомментировали вторую часть: -->
<div>
<p>Текст <!-- Ошибка 1 -->
<span>Еще текст</span>
</div>
<!--  -->
```

Практическое задание 9

Отладьте сложный код:

```
html
<!DOCTYPE html>
<html>
<body>
<div>
<p>Параграф 1
<p>Параграф 2
</div>

<script>
function test() {
var a = 10
var b = 20
console.log(a + b

test()
</script>
```

```
<style>
div {
color: red
```

```
margin: 10px  
}  
</style>  
</body>  
</html>
```

Автоматическое исправление при сохранении

Настройте авто-фикссы:

```
json  
{  
    // Исправлять ошибки при сохранении  
    "editor.codeActionsOnSave": {  
        "source.fixAll": true,  
        "source.organizeImports": true  
    },  
  
    // Форматировать при сохранении  
    "editor.formatOnSave": true,  
  
    // Форматировать при вставке  
    "editor.formatOnPaste": true  
}
```

Чек-лист освоения подсветки ошибок

Умею находить красные волнистые линии
Различаю ошибки и предупреждения
Использую панель Проблемы (Ctrl + Shift + M)
Применяю быстрые исправления (лампочки )
Понимаю интеллектуальные подсказки
Настроил цвета подсветки под себя
Установил дополнительные линтеры
Работаю со смешанным кодом (HTML+CSS+JS)
Понимаю иерархию серьезности ошибок
Использую метод отладки "разделяй и властвуй"
Настроил авто-исправление при сохранении

Профессиональные советы

Не игнорируйте предупреждения — они часто указывают на будущие проблемы

Используйте панель Проблемы для общего обзора

Настраивайте под себя — сделайте ошибки максимально заметными

Устанавливайте линтеры для своего стека технологий

Привыкайте к быстрым исправлениям — это экономит время

Итоговое упражнение

Проанализируйте и исправьте все ошибки:

```
html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Тест ошибок</title>
<style>
body {
font-family: Arial
margin: 0
padding: 20px
}

.container {
width: 100%
max-widht: 1200px
margin: 0 auto
}
</style>
</head>
<body>
<div class="container">
<h1>Тестовая страница
<p>Это параграф с <strong>выделенным текстом</p></strong>

<script>
function calculate() {
var x = 10
var y = 20
var result = x + y
console.log("Результат: " + result)
}
```

```
calculate()  
console.log(unusedVariable)  
</script>
```

```
  
</div>  
</body>  
</html>
```

Поздравляю! Теперь вы мастер обнаружения ошибок в VS Code. Вы знаете, как находить, понимать и исправлять ошибки любого типа. Помните: чем раньше найдете ошибку, тем легче ее исправить. VS Code — ваш верный помощник в этом деле!

Всплывающие подсказки по HTML и CSS

Дорогой друг!

Давайте подробно изучим всплывающие подсказки в VS Code — это как иметь персонального помощника, который всегда готов объяснить любой элемент HTML или CSS!

Что такое всплывающие подсказки?

Всплывающие подсказки (Hover Tooltips) — это информационные окна, которые появляются при наведении курсора на:

HTML-теги
CSS-свойства
HTML-атрибуты
CSS-значения
И многое другое!

Как вызвать подсказку?

Простой способ:

Наведите курсор на интересующий элемент

Задержите на 1-2 секунды

Появится подсказка с информацией!

Быстрый способ:

Наведите и нажмите `Ctrl` (удерживая)

Подсказка появится мгновенно!

Практическое задание 1

Попробуем вызвать подсказку:

Создайте файл `hints.html`

Ведите: `<div>`

Наведите курсор на слово `div`

Подождите 2 секунды

Увидите подсказку про тег `<div>`

Подсказки для HTML

1. Подсказки для тегов

Для тега <div>:

text

<div>- Элемент Content Division

HTML-элемент `<div>` является универсальным контейнером для содержимого flow. Он не влияет на содержимое или макет, пока не оформлен каким-либо образом с помощью CSS.

Веб-документы MDN: [https://developer.mozilla.org/...](https://developer.mozilla.org/)

Для тега <a>:

text

<a> - Анкерный элемент

Создает гиперссылку на веб-страницы, файлы, адреса электронной почты, местоположения на той же странице или на что-либо еще, к чему может обращаться URL-адрес.

Атрибуты: ссылка, цель, загрузка, rel, тип

Практическое задание 2

Изучим подсказки для тегов:

Наведите на разные теги:

<p>

<input>

Прочтайте описания

Обратите внимание на ссылки на документацию

2. Подсказки для атрибутов

Для атрибута href:

html

```
<a href="">ссылка</a>
```

text

href: Указывает URL-адрес страницы, на которую ведет ссылка

Возможные значения:

- Абсолютный URL-адрес: <https://www.example.com/>
- Относительный URL-адрес: about.html
- Email: mailto:someone@example.com
- Phone: tel:+1234567890

Для атрибута src:

html

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

text

src: Указывает путь к изображению

Обязательный атрибут для элементов .

Может быть относительным или абсолютным URL.

Практическое задание 3

Исследуем атрибуты:

Создайте элементы с атрибутами:

html

```
<img src="" alt="" width="" height="">  
<input type="text" placeholder="" required>
```

Наведите на каждый атрибут

Узнайте их назначение

3. Подсказки для значений атрибутов

Для type у <input>:

html

```
<input type="">
```

При наведении на type появится список:

text

text, password, email, number, tel, url,
search, date, time, checkbox, radio,
file, submit, button, reset, hidden

Для атрибута rel у <link>:

html

```
<link rel="">
```

Появится список значений:

text

stylesheet, icon, preload, prefetch,
alternate, author, help, license, next,
nofollow, noopener, noreferrer, prev, search

Практическое задание 4

Изучим варианты значений:

Создайте: <input type="">

Наведите на type

Посмотрите все возможные значения

Попробуйте с другими атрибутами

Подсказки для CSS

1. Подсказки для свойств

Для свойства display:

CSS

```
div {  
display: ;  
}
```

text

`display`: определяет, будет ли элемент обрабатываться как блочный или встроенный элемент, а также макет, используемый для его дочерних элементов.

Values: `block`, `inline`, `inline-block`, `flex`, `grid`, `none`, `contents`, `table`, etc.

Default: `inline`

Для свойства `position`:

CSS

```
.element {  
  position: ;  
}
```

text

`position`: Задает расположение элемента в документе.

Values: `static`, `relative`, `absolute`, `fixed`, `sticky`

Default: `static`

Практическое задание 5

Изучим CSS свойства:

Создайте CSS:

CSS

```
.box {  
  display: ;  
  position: ;  
  float: ;  
}
```

Наведите на каждое свойство
Прочтайте описания

2. Подсказки для значений свойств

При наведении на значение flex:

CSS

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

text

flex: Элемент ведет себя как блочный элемент и отображает свое содержимое в соответствии с моделью flexbox.

Особенности:

- Одномерный макет
- Адаптивный и гибкий
- Управление выравниванием

Для значения #ff0000:

CSS

```
.color {  
color: #ff0000;  
}
```

text

#ff0000 - Red

RGB: rgb(255, 0, 0)

HSL: hsl(0, 100%, 50%)

Preview:  (красный квадратик)

Практическое задание 6

Изучим значения CSS:

Создайте:

CSS

```
.element {  
display: flex;  
color: #00ff00;  
background: rgba(0, 0, 255, 0.5);  
}
```

Наведите на значения

Посмотрите предпросмотр цветов

3. Подсказки для функций CSS

Для calc():

CSS

```
.width {  
width: calc(100% - 20px);  
}
```

text

calc(): Позволяет выполнять вычисления при указании значений свойств CSS.

Синтаксис: calc(выражение)

Пример: calc(100% - 20px)

Поддерживает: +, -, *, /

Для rgba():

CSS

```
.bg {  
background: rgba(0, 0, 0, 0.5);  
}
```

text

rgba(): Определяет цвета, используя модель Red-Green-Blue-Alpha.

Параметры:

- red: 0-255
- green: 0-255
- blue: 0-255
- alpha: 0.0-1.0 (прозрачность)

Практическое задание 7

Изучим CSS функции:

Создайте:

CSS

```
.box {
```

```
width: calc(50% - 10px);  
background: linear-gradient(red, blue);  
transform: rotate(45deg);  
}
```

Наведите на функции
Прочтайте синтаксис

Продвинутые подсказки

1. Подсказки для псевдоклассов

Для :hover:

css

```
a:hover {  
color: red;  
}
```

text

:hover: Совпадает, когда пользователь взаимодействует с элементом с помощью указательного устройства, но не обязательно активирует его.

Обычное использование:

- Изменение цвета при наведении курсора
- Отображение всплывающих подсказок
- Анимация элементов

Для :nth-child():

css

```
li:nth-child(2n) {  
background: gray;  
}
```

text

:nth-child(): Сопоставляет элементы в зависимости от их положения среди родственных элементов.

Примеры:

- :nth-child(2) - второй ребенок
- :nth-child(odd) - странные элементы
- :nth-child(3n) - каждый третий элемент

- :nth-child(3n+1) - 1-й, 4-й, 7-й...

Практическое задание 8

Изучим псевдоклассы:

Создайте:

css

```
/* Изучите эти псевдоклассы */
a:link {}
a:visited {}
input:focus {}
li:first-child {}
li:last-child {}
```

**Наведите на каждый
Узнайте их назначение**

2. Подсказки для медиа-запросов

Для @media:

css

```
@media (max-width: 768px) {
  .container { width: 100%; }
}
```

text

@media: Используется для применения стилей на основе результата одного или нескольких медиазапросов.

Общие точки останова:

- Mobile: < 768px
- Tablet: 768px - 1024px
- Desktop: > 1024px

Особенности: width, height, orientation, prefers-color-scheme

Практическое задание 9

Изучим медиа-запросы:

Создайте:

CSS

```
@media screen and (max-width: 600px) {}  
@media print {}  
@media (prefers-color-scheme: dark) {}
```

Наведите на @media
Узнайте о возможностях

3. Подсказки для кастомных свойств (CSS переменных)

Для --variable:

CSS

```
:root {  
  --main-color: #3498db;  
}  
  
.element {  
  color: var(--main-color);  
}
```

text
--main-color: Пользовательское свойство (переменная CSS)

Использование:

- Определить: --name: value;
- Использовать: var(--name);

Выгоды:

- Повторно используемые значения
- Простое переключение тем
- Ограниченнное или глобальное

Интерактивные подсказки

1. Быстрые действия (Quick Actions)

Когда появляется лампочка  :

html

```

<!-- Наведите - появится: -->

text
[💡] Добавить отсутствующий атрибут 'alt'
Click to add alt=""
```

Или для CSS:

```
CSS
.colr: red;
/* Наведите - появится: */
```

```
text
[💡] Измените написание на "цвет"
Нажмите, чтобы исправить опечатку
```

2. Переход к определению

Удерживайте Ctrl и кликните:

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

Удерживая Ctrl и кликая на "style.css",
перейдете к этому файлу!

```
CSS
.element {
color: var(--main-color);
}
```

Удерживая Ctrl и кликая на --main-color,
перейдете к определению переменной!

Практическое задание 10

Используем интерактивные возможности:

Создайте два файла:

```
index.html с <link href="style.css">
style.css с CSS-кодом
```

Удерживая Ctrl, кликните на ссылку
Перейдите между файлами
Попробуйте с CSS переменными

Настройка подсказок

Персонализируем под себя:

```
json
{
    // Время перед показом подсказки (мс)
    "editor.hover.delay": 500,

    // Показывать подсказки
    "editor.hover.enabled": true,

    // Показывать подсказку при выделении
    "editor.hover.showAtCursor": true,

    // Максимальная ширина подсказки
    "editor.hover.maxWidth": 500,

    // Показывать документацию в подсказках
    "editor.hover.documentation": true,

    // Подсказки для HTML
    "html.hover.documentation": true,
    "html.hover.references": true,

    // Подсказки для CSS
    "css.hover.documentation": true,
    "css.hover.references": true
}
```

Практическое задание 11

Настроим подсказки:

Откройте настройки (Ctrl + ,)
Найдите hover в поиске
Измените задержку на 1000 мс

Проверьте, как изменилось поведение

Полезные сочетания клавиш

Для работы с подсказками:

Ctrl + клик — переход к определению

Ctrl + наведение — мгновенная подсказка

F12 — переход к определению (альтернатива)

Ctrl + F12 — переход к реализации

Shift + F12 — показать все ссылки

Для быстрых действий:

Ctrl + . — показать доступные действия

Alt + Enter — применить быстрое исправление

Расширения для улучшения подсказок

Дополнительные инструменты:

IntelliSense for CSS — улучшенные подсказки для CSS

HTML CSS Support — подсказки для классов и ID

Color Highlight — подсветка цветов в коде

Bracket Pair Colorizer — подсветка парных скобок

Установка улучшений:

bash

1. **Ctrl + Shift + X**
2. Поиск: "**CSS IntelliSense**"
3. Установить
4. Перезапустить VS Code

Работа с иностранными языками

Если подсказки на английском:

Для русских подсказок:

Установите **Russian Language Pack**

Перезапустите VS Code

Подсказки будут на русском!

Пример русской подсказки:

text

<div> - Элемент разделения контента

Элемент <div> является универсальным контейнером для потокового контента. Не влияет на контент или макет до тех пор, пока не будет стилизован с помощью CSS.

Чек-лист освоения подсказок

Умею вызывать подсказки наведением

Знаю про Ctrl+клик для перехода

Понимаю подсказки для HTML тегов

Разбираюсь в подсказках для атрибутов

Использую подсказки для CSS свойств

Читаю подсказки для значений

Знаю про подсказки для функций

Использую быстрые действия (лампочки)

Настроил подсказки под себя

Установил дополнительные расширения

Работаю с русскими подсказками

Профессиональные советы

Не игнорируйте подсказки — они содержат ценную информацию

Используйте Ctrl+клик для навигации по проекту

Читайте документацию в подсказках — там часто есть ответы

Настройте задержку под свою скорость работы

Устанавливайте расширения для вашего стека технологий

Итоговое упражнение

Проанализируйте код с помощью подсказок:

html

<!DOCTYPE html>

<html>

<head>

<style>

:root {

```
--primary-color: #3498db;
--spacing: 20px;
}

.container {
display: grid;
grid-template-columns: repeat(3, 1fr);
gap: var(--spacing);
padding: calc(var(--spacing) * 2);
}

.card {
background: var(--primary-color);
border-radius: 8px;
transition: transform 0.3s ease;
}

.card:hover {
transform: translateY(-5px);
}

@media (max-width: 768px) {
.container {
grid-template-columns: 1fr;
}
}
</style>
</head>
<body>
<div class="container">
<div class="card" data-id="1">
<h2>Заголовок</h2>
<p>Текст карточки</p>
<a href="#" target="_blank">Ссылка</a>
</div>
</div>

<script>
// JavaScript код для практики
const cards = document.querySelectorAll('.card');
cards.forEach(card => {
card.addEventListener('click', () => {
const id = card.getAttribute('data-id');

```

```
console.log(`Clicked card ${id}`);
});
});
</script>
</body>
</html>
```

Задания:

Наведите на все HTML теги

Изучите все CSS свойства

Проверьте подсказки для атрибутов

Используйте Ctrl+клик для навигации

Найдите все быстрые действия

Поздравляю! Теперь вы мастер всплывающих подсказок в VS Code.
Вы знаете, как быстро получать информацию о любом элементе
кода, переходить к определениям и применять быстрые
исправления. Эти навыки значительно ускорят вашу работу и
сделают обучение более эффективным!

Проверка валидности кода

Дорогой друг!

Давайте научимся проверять валидность (правильность) кода в VS Code. Это как иметь личного корректора, который проверяет ваш код на соответствие стандартам!

Что такое валидность кода?

Валидный код — это код, который:

Соответствует стандартам W3C (World Wide Web Consortium)
Не содержит синтаксических ошибок
Правильно использует теги и атрибуты
Работает во всех современных браузерах

Зачем проверять валидность?

Совместимость — код работает везде

Доступность — сайт доступен для людей с ограничениями

SEO — поисковики лучше понимают правильный код

Поддержка — меньше ошибок в будущем

Встроенная проверка валидности в VS Code

1. Базовые проверки HTML

Автоматическое обнаружение ошибок:

Пример 1: Незакрытый тег

```
html
<div>
<p>Текст без закрытия
</div>
```

Красная волнистая линия под <p>

Сообщение: 'p' is not closed

Пример 2: Неправильная вложенность

```
html
<p>
<div>Ошибка!</div>
</p>
```

Желтая волнистая линия под `<div>`
Сообщение: 'div' cannot be inside 'p'

Практическое задание 1

Создадим HTML с ошибками:
Создайте файл valid.html

Ведите проблемный код:

```
html
<!DOCTYPE html>
<html>
<body>
<div>
<p>Текст 1
<p>Текст 2
</div>

<ul>
<li>Пункт 1</li>
<li>Пункт 2</li>
</ul>
</body>
</html>
```

Найдите все ошибки (их 3)

2. Проверка атрибутов

Обязательные атрибуты:

Для ``:

```
html
 <!-- Нет alt! -->
```

Желтая волнистая линия

Сообщение: Required attribute 'alt' not present

Для <a>:

html

<a>Ссылка без href

Желтая волнистая линия

Сообщение: An 'a' element must have an 'href' attribute

Практическое задание 2

Проверим обязательные атрибуты:

Создайте элементы без обязательных атрибутов:

html

```

<a name="anchor">Якорь</a>
<area shape="rect" coords="0,0,100,100">
```

Посмотрите какие предупреждения появляются

Исправьте добавлением нужных атрибутов

3. Проверка CSS валидности

Синтаксические ошибки CSS:

Несуществующее свойство:

CSS

```
.box {
  colr: red; /* Опечатка! */
}
```

Красная волнистая линия

Сообщение: Unknown property: 'colr'

Пропущенная точка с запятой:

CSS

```
.container {  
width: 100px /* Нет ; */  
height: 200px;  
}  
Красная волнистая линия
```

Сообщение: Expected ;'

Практическое задание 3

Найдем CSS ошибки:

Создайте CSS файл:

```
CSS  
/* Проверка валидности CSS */  
.element {  
display: flex  
flex-direction: column  
  
/* Неправильные свойства */  
border-radius: 10px;  
text-align: center;  
  
/* Неправильное значение */  
position: absolut;  
  
/* Лишние скобки */  
color: red;)  
}
```

Найдите и исправьте 6 ошибок

Панель "Проблемы" для валидации

Централизованный просмотр всех проблем:

Открываем панель:

Ctrl + Shift + M (Problems)
Или: View → Problems
Или клик на иконку ошибок в статус-баре

Что видим в панели:

text

- 📁 PROBLEMS (5)
 - ▼ index.html (3)
 - ✗ 'img' must have 'alt' attribute (строка 10)
 - ⚠ 'div' cannot be inside 'p' (строка 5)
 - ✗ Expected closing tag for 'span' (строка 7)
 - ▼ style.css (2)
 - ✗ Unknown property 'widht' (строка 3)
 - ✗ Expected ';' (строка 6)

Практическое задание 4

Используем панель Проблемы:

Создайте файлы с ошибками

Откройте панель Проблемы

Кликните на каждую ошибку

Переходите к проблемным строкам

Исправьте все ошибки

Расширения для валидации кода

1. **HTMLHint** — продвинутая проверка HTML

Установка:

Ctrl + Shift + X

Поиск: HTMLHint

Установите расширение

Перезапустите VS Code

Что проверяет **HTMLHint**:

html

<!-- 1. Теги в нижнем регистре -->

<DIV>Текст</DIV> <!-- Ошибка: тег в верхнем регистре -->

<!-- 2. Дублирующиеся атрибуты -->

<div class="one" class="two"></div> <!-- Ошибка -->

```
<!-- 3. Пустые ссылки -->
<a href="#">Пустая ссылка</a> <!-- Предупреждение -->
```

```
<!-- 4. Специфические атрибуты -->
 <!-- alt не может быть пустым -->
```

Практическое задание 5

Настройка HTMLHint:

Создайте файл `.htmlhintrc` в корне проекта:

```
json
{
  "tagname-lowercase": true,
  "attr-lowercase": true,
  "attr-value-double-quotes": true,
  "doctype-first": true,
  "tag-pair": true,
  "spec-char-escape": true,
  "id-unique": true,
  "src-not-empty": true,
  "alt-require": true,
  "title-require": true
}
```

Проверьте как работает расширение

2. Stylelint — проверка CSS

Установка:

`Ctrl + Shift + X`

Поиск: `stylelint`

Установите

Создайте `.stylelintrc.json`

Что проверяет Stylelint:

`CSS`

```
/* 1. Свойства в алфавитном порядке */
.element {
  color: red; /* Должно быть после display */
  display: block;
}

/* 2. Ненулевые значения без единиц */
.element {
  margin: 0px; /* Должно быть 0 */
}

/* 3. Избыточные селекторы */
div.container {} /* div избыточен */

/* 4. Магические числа */
.anim {
  transition: 0.3s; /* Должна быть переменная */
}
```

Практическое задание 6

Настройте Stylelint:

Установите расширение

Создайте конфиг:

```
json
{
  "extends": "stylelint-config-standard",
  "rules": {
    "color-hex-case": "lower",
    "color-named": "never",
    "declaration-block-no-duplicate-properties": true,
    "declaration-block-single-line-max-declarations": 1,
    "max-nesting-depth": 3,
    "selector-max-id": 0,
    "selector-no-qualifying-type": true
  }
}
```

3. W3C Validation — онлайн проверка

Интеграция с VS Code:

Хотя есть расширения, лучше использовать онлайн:

Откройте validator.w3.org

Выберите вкладку "Validate by Direct Input"

Вставьте код из VS Code (Ctrl + A, Ctrl + C)

Нажмите Check

Получите детальный отчет

Пример отчета W3C:

text

Validation Output: 3 Errors

1. Line 5, Column 10:

Element "div" not allowed as child of element "p"

2. Line 7, Column 15:

Required attribute "alt" not specified

3. Line 10, Column 5:

End tag "span" missing

Практическое задание 7

Проверьте код через W3C Validator:

Создайте HTML с ошибками

Скопируйте весь код

Перейдите на validator.w3.org

Вставьте и проверьте

Исправьте ошибки по отчету

Автоматическая валидация при сохранении

Настройка авто-проверки:

```
json
{
    // Проверять HTML при сохранении
    "html.validate.scripts": true,
    "html.validate.styles": true,

    // Проверять CSS при сохранении
    "css.validate": true,

    // Показывать ошибки в реальном времени
    "editor.liveErrors": true,

    // Автоматическое форматирование при сохранении
    "editor.formatOnSave": true,
    "html.format.enable": true,
    "css.format.enable": true,

    // Автоматические исправления
    "editor.codeActionsOnSave": {
        "source.fixAll": true
    }
}
```

Практическое задание 8

Настройте авто-валидацию:

Откройте настройки (Ctrl + ,)
Настройте как показано выше
Создайте файл с ошибками
Сохраните (Ctrl + S)
Убедитесь, что ошибки подсвечиваются

Проверка доступности

Важность доступности:

Проблемный код:

```
html
<!-- 1. Нет текстовой альтернативы -->
<button>
```

```
  
</button>  
  
<!-- 2. Нет меток для полей формы -->  
<input type="text">  
<input type="checkbox">  
  
<!-- 3. Низкий контраст -->  
<p style="color: #aaa; background: #fff;">Текст</p>
```

Расширение для доступности:

Установите: axe Accessibility Linter

Что проверяет:

Цветовой контраст
Альтернативный текст
Структура заголовков
Доступность клавиатуры
ARIA атрибуты

Практическое задание 9

Проверьте доступность:

Установите axe Accessibility Linter

Создайте проблемный код:

```
html  
<div onclick="submit()">Отправить</div>  
  
<input type="text" id="name">  
<label for="email">Email</label>  
<input type="email">
```

Посмотрите какие ошибки доступности найдены

Семантическая валидность

Проверка семантики:

Неправильно (несемантично):

```
html
<div class="header">
<div class="nav">
<div>Главная</div>
<div>О нас</div>
</div>
</div>

<div class="main-content">
<div class="article">
<div class="big-text">Заголовок</div>
</div>
</div>
```

Правильно (семантично):

```
html
<header>
<nav>
<ul>
<li><a href="#">Главная</a></li>
<li><a href="#">О нас</a></li>
</ul>
</nav>
</header>

<main>
<article>
<h1>Заголовок</h1>
</article>
</main>
```

Практическое задание 10

Улучшите семантику:

Возьмите несемантичный код:

```
html
<div id="page">
<div class="top">
```

```
<span class="logo">Лого</span>
<div class="menu">...</div>
</div>

<div class="middle">
<div class="left">...</div>
<div class="right">...</div>
</div>

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

Перепишите с семантическими тегами
Проверьте валидность

Валидация JavaScript

Проверка JS с ESLint:

Установка:

```
bash
# В терминале VS Code:
npm init -y
npm install eslint --save-dev
npx eslint --init
```

Конфигурация .eslintrc.json:

```
json
{
  "env": {
    "browser": true,
    "es2021": true
  },
  "extends": "eslint:recommended",
  "rules": {
    "no-unused-vars": "error",
    "no-console": "warn",
    "eqeqeq": "error",
    "curly": "error",
    "quotes": ["error", "single"]
  }
}
```

```
}
```

Практическое задание 11

Настройте ESLint:

Установите ESLint через терминал

Настройте конфигурацию

Создайте JS с ошибками:

```
javascript
```

```
var x = 10
var y = 20
console.log(x + y)
```

```
if (x = 10) {
console.log("равно")
}
```

```
function test() {
var unused = "не используется"
return "результат"
}
```

Проверьте какие ошибки найдет ESLint

Интеграция с Git (pre-commit hooks)

Автоматическая проверка перед коммитом:

Установка Husky + lint-staged:

```
bash
```

```
npm install husky lint-staged --save-dev
```

Настройка package.json:

```
json
```

```
{
  "husky": {
    "hooks": {
      "pre-commit": "lint-staged"
    }
  },
  "lint-staged": {
```

```
"*.html": ["htmlhint", "prettier --write"],  
"*.css": ["stylelint", "prettier --write"],  
"*.js": ["eslint", "prettier --write"]  
}  
}
```

Чек-лист валидации

Для каждого проекта проверяйте:

HTML:

- Доктайп указан
- Язык страницы задан
- Кодировка UTF-8
- Все теги закрыты
- Правильная вложенность
- Атрибут `alt` у изображений
- Семантические теги
- Валидные атрибуты

CSS:

- Корректный синтаксис
- Известные браузеру свойства
- Правильные единицы измерения
- Нет дублирующихся свойств
- Порядок свойств (опционально)
- Достаточный цветовой контраст

JavaScript:

- Синтаксические ошибки
- Неиспользуемые переменные
- Строгое сравнение (`====`)
- Обработка ошибок
- Современный синтаксис

Профессиональный рабочий процесс

Идеальный пайплайн валидации:

Пишете код в VS Code

Встроенные проверки работают в реальном времени
Сохраняете (Ctrl + S) → автоформатирование
Проверяете в панели Проблемы (Ctrl + Shift + M)
Запускаете линтеры через терминал
Проверяете онлайн через W3C Validator
Делаете коммит → pre-commit hooks проверяют
Тестируете в разных браузерах

Распространенные ошибки валидации

Топ-10 ошибок новичков:

Пропущенный alt у изображений
Незакрытые теги (особенно <p>,)
Неправильная вложенность (<div> в <p>)
Устаревшие атрибуты (border, align)
Инлайн-стили вместо CSS классов
Таблицы для верстки вместо CSS Grid/Flexbox
Пустые ссылки ()
Отсутствие меток у полей формы
Низкий контраст текста
Несемантичные теги (все через <div>)

Итоговый тест валидации

Проанализируйте и исправьте код:

```
html
<!DOCTYPE html>
<HTML>
<HEAD>
<TITLE>Тест валидности</TITLE>
<style>
.container {
width: 100%;
border: 1px solid black
padding: 0px;
}

p {
color: #777;
font-size: 14px;
}
```

```
.button {  
background-colour: blue;  
color: white;  
}  
</style>  
</HEAD>  
<BODY>  
<div class="container">  
<p>Текст абзаца  
<br><br>  
  
  
  
<table border="1" cellpadding="10">  
<tr>  
<td>Ячейка 1</td>  
<td>Ячейка 2</td>  
</tr>  
</table>  
  
<form>  
<input type="text" name="username">  
<input type="submit" value="Отправить">  
</form>  
  
<div onclick="alert('Click')">Кнопка</div>  
  
<font size="4" color="red">Текст</font>  
</div>  
  
<script>  
var x = 10  
if (x = 20) {  
console.log("x равно 20")  
}  
  
var unused = "не используется"  
</script>  
</BODY>  
</HTML>
```

Найдите и исправьте минимум 15 ошибок!

Поздравляю! Теперь вы эксперт по валидации кода. Вы знаете, как проверять HTML, CSS и JavaScript на соответствие стандартам, находить и исправлять ошибки. Помните: валидный код — это профессиональный код, который работает везде и для всех!

Глава 13: Встроенный терминал

Что такое терминал и зачем он нужен

Дорогой друг!

Давайте разберемся с терминалом — это одна из самых мощных возможностей VS Code, которая может сначала напугать, но на самом деле очень проста и полезна!

Что такое терминал?

Терминал — это текстовый интерфейс для взаимодействия с компьютером. Представьте, что это **командная строка**, где вы можете давать компьютеру команды, набирая их на клавиатуре.

Простая аналогия:

Графический интерфейс (кнопки, меню) — как поездка на такси (водитель все делает за вас)

Терминал — как вождение собственного автомобиля (вы контролируете все сами)

Терминал в VS Code выглядит так:

```
PS C:\Users\ВашеИмя\project>
```

```
(курсор мигает здесь)
```

Text

Зачем веб-разработчику терминал?

1. Установка программ и инструментов

bash

```
# Установить Node.js пакет  
npm install package-name
```

```
# Установить расширение для VS Code  
code --install-extension extension-name
```

2. Запуск локального сервера

```
bash
# Запуск Live Server (альтернатива кнопке)
live-server
```

3. Работа с системой контроля версий Git

```
bash
# Сохранить изменения в проекте
git add .
git commit -m "Добавил новую страницу"
git push
```

Практическое задание 1

Откроем терминал в VS Code:

Нажмите `Ctrl + `` (клавиша над Tab, под Esc)
Внизу окна откроется панель терминала
Увидите приглашение командной строки
Напечатайте: echo "Привет, терминал!"
Нажмите Enter
Увидите ответ: Привет, терминал!

Базовые команды терминала

Навигация по папкам:

Windows (PowerShell):

```
powershell
# Посмотреть где я
pwd # или Get-Location

# Посмотреть что в папке
dir # или ls, или Get-ChildItem

# Перейти в папку
cd имя_папки
```

```
# Выйти на уровень выше  
cd ..  
  
# Перейти в другую папку  
cd C:\Users\ВашеИмя\Desktop
```

Mac/Linux:

```
bash  
# Посмотреть где я  
pwd  
  
# Посмотреть что в папке  
ls
```

```
# Перейти в папку  
cd имя_папки  
  
# Выйти на уровень выше  
cd ..  
  
# Перейти в домашнюю папку  
cd ~
```

Практическое задание 2

Попрактикуем навигацию:

Узнайте где вы: `pwd`
Посмотрите файлы: `dir` или `ls`
Создайте папку: `mkdir test_folder`
Зайдите в нее: `cd test_folder`
Вернитесь назад: `cd ..`

Работа с файлами

Создание файлов:

```
bash  
# Создать пустой файл  
touch index.html # на Mac/Linux  
New-Item index.html # на Windows
```

```
# Создать файл с содержимым  
echo "<h1>Привет</h1>" > index.html
```

Просмотр файлов:

```
bash  
# Показать содержимое файла  
cat index.html # на Mac/Linux  
Get-Content index.html # на Windows
```

```
# Показать начало файла  
head -n 10 style.css
```

```
# Показать конец файла  
tail -n 5 script.js
```

Практическое задание 3

Поработаем с файлами:

Создайте HTML файл: echo

```
"<html><body><h1>Тест</h1></body></html>" > test.html
```

Посмотрите содержимое: cat test.html или Get-Content test.html

Создайте CSS файл: echo "body { color: red; }" > style.css

Проверьте оба файла

Полезные команды для веб-разработки

1. Node.js и npm команды

Проверка установки:

```
bash  
# Проверить версию Node.js  
node --version
```

```
# Проверить версию npm  
npm --version
```

```
# Создать новый проект
```

```
npm init -y
```

Установка пакетов:

bash

```
# Установить пакет  
npm install package-name
```

```
# Установить пакет для разработки  
npm install --save-dev package-name
```

```
# Установить глобально (для всех проектов)  
npm install -g package-name
```

Практическое задание 4

Проверим Node.js:

Напишите: node --version

Если установлено — увидите номер версии

Если нет — не страшно, мы можем работать и без него

Попробуйте: npm --version

2. Git команды

Базовые операции:

bash

```
# Инициализировать Git в папке  
git init
```

```
# Проверить статус  
git status
```

```
# Добавить файлы  
git add index.html  
git add . # все файлы
```

```
# Сохранить изменения  
git commit -m "Описание изменений"
```

```
# Отправить на GitHub
```

`git push`

Практическое задание 5

Попробуем Git:

Создайте папку: `mkdir git-test`

Зайдите в нее: `cd git-test`

Инициализируйте Git: `git init`

Создайте файл: `echo "# Мой проект" > README.md`

Проверьте статус: `git status`

Терминал в VS Code — особенности

Преимущества терминала VS Code:

Интеграция с редактором — можно открывать файлы прямо из терминала

Несколько терминалов — можно открыть несколько вкладок

Быстрый доступ к папке проекта

Копирование/вставка как в обычном тексте

Как работать с терминалом в VS Code:

Открыть/закрыть:

``Ctrl + ``` — открыть/закрыть терминал

``Ctrl + Shift + ``` — новый терминал

`Ctrl + Shift + 5` — разделить терминал

Навигация:

`Ctrl + PageUp / Ctrl + PageDown` — переключение между терминалами

`Ctrl + L` — очистить терминал

Практическое задание 6

Освоим управление терминалом:

Откройте терминал (`Ctrl + ```)

Создайте новый терминал (`Ctrl + Shift + ```)

Посмотрите что у вас два терминала

Переключайтесь между ними (`Ctrl + PageUp/Down`)

Закройте один из терминалов

Полезные сочетания клавиш

Для работы в терминале:

Tab — автодополнение команд и путей
Стрелки вверх/вниз — история команд
Ctrl + C — остановить текущую команду
Ctrl + D — закрыть терминал (на Mac/Linux)
Ctrl + A — в начало строки
Ctrl + E — в конец строки
Ctrl + U — очистить строку

Практическое задание 7

Потренируем автодополнение:

Начните писать: cd Des (не нажимайте Enter)

Нажмите Tab

Терминал должен дописать Desktop если такая папка есть

Попробуйте с другими командами

Создание своих команд (алиасы)

Упрощаем частые команды:

Windows PowerShell:

```
powershell
# Добавить в профиль PowerShell
notepad $PROFILE
```

Добавить алиасы:

```
function proj { cd C:\Projects }
function home { cd ~ }
function list { Get-ChildItem }
```

Mac/Linux:

```
bash
# Редактировать .bashrc или .zshrc
```

```
nano ~/.bashrc
```

```
# Добавить алиасы:  
alias proj='cd ~/Projects'  
alias home='cd ~'  
alias ll='ls -la'
```

Практическое задание 8

Создадим простой алиас:

Для Windows:

```
powershell  
function sites { cd C:\Users\ВашеИмя\Sites }
```

Для Mac/Linux:

```
bash  
alias sites='cd ~/Sites'
```

Перезапустите терминал

Напишите: sites

Убедитесь, что перешли в нужную папку

Решение проблем в терминале

Частые проблемы новичков:

"Команда не найдена"

Причина: команды нет в системе

Решение: установите нужную программу или проверьте написание

"Отказано в доступе"

Причина: нет прав

Решение: запустите VS Code от имени администратора

Терминал не открывается

Решение: перезапустите VS Code

Или: View → Terminal

Зависшая команда

Решение: Ctrl + C для остановки

Практическое задание 9

Потренируем остановку команд:

Ведите команду, которая будет "думать":

```
bash
ping google.com # Будет пинговать постоянно
Подождите 5 секунд
Нажмите Ctrl + C
Команда остановится
```

Терминал для конкретных задач

1. Для веб-разработки:

Сборка проекта:

```
bash
# Собрать проект
npm run build

# Запустить в режиме разработки
npm run dev

# Проверить код на ошибки
npm run lint

# Запустить тесты
npm test
```

2. Для работы с базами данных:

```
bash
# Подключиться к базе данных
mysql -u username -p

# Экспорт базы данных
mysqldump -u username -p database > backup.sql
```

3. Для системного администрирования:

```
bash
# Посмотреть процессы
top # на Mac/Linux
Get-Process # на Windows

# Проверить дисковое пространство
df -h # на Mac/Linux
Get-PSDrive # на Windows
```

Практическое задание 10

Создадим простой веб-проект через терминал:

Создайте папку проекта:

```
bash
mkdir my-web-project
cd my-web-project
```

Создайте структуру:

```
bash
mkdir css js images
touch index.html
touch css/style.css
touch js/script.js
```

Проверьте структуру:

```
bash
dir # или ls -la
```

Откройте в VS Code:

```
bash
code . # точка означает текущую папку
```

Интеграция терминала с редактором

Полезные трюки:

Открыть файл из терминала:

bash

code index.html # откроет файл в VS Code

Открыть папку:

bash

code . # текущая папка

code .. # родительская папка

Открыть на определенной строке:

bash

code -g index.html:10 # откроет на 10 строке

Сравнить файлы:

bash

code -d file1.html file2.html

Практическое задание 11

Интегрируем терминал и редактор:

В терминале создайте файл: touch test.txt

Откройте его в VS Code: code test.txt

Напишите в файле "Текст из терминала"

Сохраните (Ctrl + S)

Вернитесь в терминал: Ctrl +

Проверьте содержимое: cat test.txt

Безопасность в терминале

Важные правила:

Не запускайте неизвестные команды из интернета

**Внимательно читайте что делает команда
Не давайте root/администраторские права без необходимости
Регулярно обновляйте установленные программы**

Опасные команды (не выполняйте их!):

```
bash
# УДАЛИТ ВСЕ ФАЙЛЫ!
rm -rf / # на Mac/Linux
rmdir /s /q C:\ # на Windows
```

```
# ФОРМАТИРУЕТ ДИСК!
format C: # на Windows
```

Чек-лист освоения терминала

Умею открывать/закрывать терминал (^Ctrl + ``)
Знаю базовые команды навигации (cd, ls/dir, pwd)
Могу создавать файлы и папки (touch/mkdir)
Работаю с Git через терминал
Использую Node.js/npm команды
Знаю горячие клавиши терминала
Умею останавливать команды (Ctrl + C)
Могу открывать файлы из терминала в VS Code (code .)
Создал полезные алиасы для себя
Понимаю основы безопасности

Профессиональный рабочий процесс

Как профессионалы используют терминал:

Быстрая навигация по проекту
Установка зависимостей (npm install)
Запуск серверов разработки
Работа с Git (коммиты, пуши, пулсы)
Автоматизация задач (скрипты сборки)
Отладка (логи, мониторинг)
Деплой (загрузка на сервер)

Пример дня веб-разработчика:

```
bash
```

```
# Утро: начать работу  
cd ~/projects/my-site  
git pull # получить свежие изменения  
npm install # обновить зависимости  
npm run dev # запустить сервер разработки
```

```
# День: разработка  
# ... работа в VS Code ...
```

```
# Вечер: сохранить изменения  
git add .  
git commit -m "Добавил новую функциональность"  
git push
```

Итоговое упражнение

Создайте проект с нуля через терминал:

Создайте структуру:

```
bash  
mkdir -p my-app/{css,js,images}  
cd my-app  
touch index.html css/style.css js/app.js
```

Инициализируйте Git:

```
bash  
git init  
git add .  
git commit -m "Initial commit"
```

Создайте package.json (если Node.js установлен):

```
bash  
npm init -y
```

Установите Live Server:

```
bash  
npm install --save-dev live-server
```

Добавьте скрипт в package.json:

```
json
"scripts": {
  "start": "live-server"
}
```

Запустите сервер:

```
bash
npm start
```

Откройте проект в VS Code:

```
bash
code .
```

Поздравляю! Теперь вы понимаете, что такое терминал и как его использовать. Не бойтесь командной строки — это мощный инструмент, который сделает вашу работу эффективнее. Начинайте с простых команд, постепенно осваивайте более сложные, и скоро терминал станет вашим лучшим помощником!

Как открыть терминал в VS Code

Дорогой друг!

Давайте научимся открывать и использовать терминал в VS Code. Это проще, чем кажется! Я покажу вам все способы, и вы выберете самый удобный для себя.

Способ 1: Горячие клавиши (самый быстрый)

Основная горячая клавиша:

Ctrl + ` — открыть/закрыть терминал

Где находится эта клавиша:

text

На клавиатуре ищите клавишу:



Она слева от цифры 1, под клавишей Esc

Важно: Это **обратный апостроф**, не путать с одинарной кавычкой!

Как это работает:

Нажмите и удерживайте Ctrl

Нажмите клавишу ` (обратный апостроф)

Отпустите обе клавиши

Терминал откроется в нижней части окна

Практическое задание 1

Попробуем открыть терминал горячими клавишами:

Откройте VS Code

Найдите клавишу ` на клавиатуре

Нажмите Ctrl +

Увидите как снизу появляется панель терминала

Нажмите снова ` Ctrl + `` — терминал закроется

Способ 2: Через меню (самый наглядный)

Пошаговая инструкция:

Нажмите на меню "View" (Вид) в верхней части окна
Выберите пункт "Terminal" (Терминал)
Или "Terminal: New Terminal" (Терминал: Новый терминал)

На русском интерфейсе:

text

Вид → Терминал → Новый терминал

Практическое задание 2

Откроем терминал через меню:

Посмотрите на верхнюю строку VS Code
Найдите меню "View" (или "Вид")
Наведите курсор, откроется выпадающее меню
Найдите "Terminal" (Терминал)
Нажмите — откроется терминал

Способ 3: Через палитру команд (самый гибкий)

Использование палитры команд:

Нажмите `Ctrl + Shift + P` (откроется палитра команд)
Начните печатать: `terminal`
Выберите нужную команду:
`View: Toggle Terminal` — показать/скрыть терминал
`Terminal: Create New Terminal` — создать новый терминал
`Terminal: Split Terminal` — разделить терминал

Практическое задание 3

Используем палитру команд:

Нажмите `Ctrl + Shift + P`

Начните печатать `terminal`

Выберите "View: Toggle Terminal"

Терминал откроется

Повторите чтобы закрыть

Способ 4: Через кнопку в меню (самый простой для начинающих)

Если не получается с горячими клавишами:

Посмотрите на верхнее меню VS Code

Найдите пункт "Terminal" (Терминал)

Нажмите на него

Выберите "New Terminal"

Альтернативный путь:

Правой кнопкой мыши в области файлов

Выберите "Open in Integrated Terminal"

Практическое задание 4

Найдем кнопку терминала:

Внимательно посмотрите на интерфейс VS Code

Найдите вверху: Terminal → New Terminal

Или найдите иконку терминала (если есть)

Кликните и откройте терминал

Как выглядит открытый терминал

Когда терминал откроется, вы увидите:

text

```
Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights...
```

```
PS C:\Users\ВашеИмя\project>
```

Или на Mac/Linux:

text

```
Last login: Mon Dec 1 10:00:00 on consol  
username@MacBook-Pro ~ %
```

Основные элементы терминала:

Приглашение командной строки (PS C:\...> или username@... %)

Мигающий курсор (—) — здесь вы вводите команды

Вкладки — если открыто несколько терминалов

Кнопки управления (свернуть, закрыть, новый)

Практическое задание 5

Изучим интерфейс терминала:

Откройте терминал любым способом

Найдите мигающий курсор

Посмотрите какая командная строка указана (PowerShell, bash, etc.)

Ведите echo "Привет" и нажмите Enter

Увидите ответ: Привет

Разные типы терминалов

VS Code поддерживает несколько терминалов:

PowerShell (Windows по умолчанию)

Command Prompt (cmd) (старый Windows терминал)

Git Bash (если установлен Git)

bash/zsh (Mac/Linux)

WSL (Windows Subsystem for Linux)

Как выбрать тип терминала:

При первом открытии:

Нажмите на стрелку рядом с + в правом верхнем углу терминала

Выберите нужный тип терминала

Или через палитру команд:

Ctrl + Shift + P

Ведите: Terminal: Select Default Profile

Выберите нужный терминал

Практическое задание 6

Сменим тип терминала:

Откройте терминал

Найдите маленькую стрелку рядом с + вверху терминала

Нажмите на стрелку

Посмотрите какие типы терминалов доступны

Выберите другой тип (если есть)

Настройки терминала

Чтобы сделать терминал удобнее:

Откройте настройки терминала:

Нажмите Ctrl + ,

В поиске введите: terminal

Или откройте раздел: Terminal → Integrated

Полезные настройки:

```
json
{
    // Размер шрифта терминала
    "terminal.integrated.fontSize": 16,
    // Количество строк в истории
    "terminal.integrated.scrollback": 10000,
    // Курсор в терминале
    "terminal.integrated.cursorStyle": "line",
    "terminal.integrated.cursorBlinking": true,
    // Автозакрытие терминала при выходе
    "terminal.integrated.confirmOnExit": false,
    // Папка по умолчанию
    "terminal.integrated.cwd": "${workspaceFolder}"
}
```

Практическое задание 7

Настроим терминал под себя:

Откройте настройки (**Ctrl + ,**)

В поиске введите: terminal font

Увеличьте шрифт до 18

Вернитесь в терминал — шрифт должен стать больше

Попробуйте другие настройки

Управление несколькими терминалами

Как работать с несколькими терминалами:

Создать новый терминал:

`**Ctrl + Shift + ``** — новый терминал

Или нажать + в правом верхнем углу терминала

Переключение между терминалами:

Ctrl + PageUp — предыдущий терминал

Ctrl + PageDown — следующий терминал

Или кликнуть на вкладку терминала

Разделить терминал:

Ctrl + Shift + 5 — разделить терминал

Или правой кнопкой → Split Terminal

Практическое задание 8

Поэкспериментируем с несколькими терминалами:

Откройте терминал

Создайте второй терминал (`**Ctrl + Shift + ``**)

Переключайтесь между ними (**Ctrl + PageUp/Down**)

В каждом введите разные команды:

В первом: echo "Терминал 1"

Во втором: echo "Терминал 2"

Посмотрите как работают оба

Быстрые действия в терминале

Полезные трюки:

Копировать/вставить в терминале:

Копировать: Выделить текст → **Ctrl + C**

Вставить: **Ctrl + V** или правой кнопкой → Paste

Очистить терминал:

Ctrl + L — очистить экран

Или ввести **clear** (Mac/Linux) или **cls** (Windows)

Поиск в терминале:

Ctrl + F — найти текст в терминале

Практическое задание 9

Потренируем быстрые действия:

Ведите команду: echo "Это тестовый текст для поиска"

Нажмите **Ctrl + F**

Ведите "поиска"

Найдите текст в терминале

Очистите терминал: **cls** или **clear**

Решение проблем с открытием терминала

Если терминал не открывается:

Проблема 1: "Горячие клавиши не работают"

Решение:

Проверьте, что нажимаете **Ctrl + `**, а не **Ctrl + 1**

Попробуйте через меню: View → Terminal

Перезапустите VS Code

Проблема 2: "Терминал открывается пустым"

Решение:

Подождите несколько секунд

Нажмите Enter

Попробуйте другой тип терминала

Проблема 3: "Нет пункта Terminal в меню"

Решение:

Возможно, устаревшая версия VS Code

Обновите VS Code

Или используйте палитру команд (Ctrl + Shift + P)

Проблема 4: "Ошибка при открытии терминала"

Решение:

Проверьте настройки терминала

Попробуйте сбросить настройки

Переустановите VS Code

Практическое задание 10

Диагностируем проблему:

Если терминал не открывается горячими клавишами

Попробуйте открыть через меню

Если не получается — через палитру команд

Если все плохо — перезапустите компьютер

В 99% случаев это помогает!

Интеграция терминала с редактором

Крутые возможности:

Открыть файл из терминала:

```
bash
```

Открыть файл в VS Code

```
code index.html
```

Открыть папку

```
code .
```

Открыть на определенной строке

```
code -g style.css:10
```

Запустить команду и открыть результат:

```
bash
```

```
# Создать файл и открыть его  
echo "<h1>Тест</h1>" > test.html && code test.html
```

Практическое задание 11

Интегрируем терминал и редактор:

В терминале создайте файл:

```
bash  
echo "body { color: red; }" > test.css
```

Откройте его в VS Code:

```
bash  
code test.css
```

Файл откроется в новой вкладке редактора

Измените что-нибудь и сохраните

Вернитесь в терминал и проверьте:

```
bash  
cat test.css
```

Чек-лист освоения

Умею открывать терминал горячими клавишами (`Ctrl + ```)

Могу открыть через меню (View → Terminal)

Использую палитру команд для открытия

Знаю как выглядит открытый терминал

Могу выбрать тип терминала

Настроил терминал под себя

Работаю с несколькими терминалами

Использую быстрые действия (копирование, поиск)

Знаю как решать проблемы с открытием

Интегрирую терминал с редактором

Открываю файлы из терминала в VS Code

Профессиональные советы

Запомните горячие клавиши — это самый быстрый способ

Настройте терминал под себя — сделайте комфортным для глаз

Используйте несколько терминалов для разных задач

Не бойтесь экспериментировать — терминал прощает ошибки
Начинайте с простых команд — echo, cd, ls/dir

Итоговое упражнение

Полный цикл работы с терминалом:

Откройте терминал (`Ctrl + ``)

Создайте папку проекта:

```
bash
mkdir my-terminal-project
cd my-terminal-project
```

Создайте файлы:

```
bash
touch index.html style.css script.js
```

Откройте папку в VS Code:

```
bash
code .
```

Вернитесь в терминал (Ctrl + `)

Проверьте файлы:

```
bash
dir # или ls
```

Создайте второй терминал (Ctrl + Shift + `)
В первом следите за файлами
Во втором запустите Live Server (если установлен)
Переключайтесь между терминалами

Поздравляю! Теперь вы мастер открытия терминала в VS Code. Вы знаете все способы, умеете настраивать терминал под себя и интегрировать его с редактором. Терминал — ваш мощный союзник в веб-разработке, используйте его с умом!

Простые команды для навигации по папкам

Дорогой друг!

Давайте освоим простые команды для навигации по папкам в терминале. Это как научиться ходить по своему компьютеру с закрытыми глазами — знаете куда идете, даже не глядя!

Самые важные команды навигации

1. Узнать где я нахожусь (Кто я? Где я?)

Windows (PowerShell):

```
powershell  
# Способ 1 (самый простой):  
pwd  
  
# Способ 2 (длинный):  
Get-Location
```

```
# Вы увидите что-то вроде:  
PS C:\Users\Иван\Documents>
```

Mac/Linux:

```
bash  
pwd  
# Вы увидите: /Users/ivan/Documents
```

Что значит вывод:

C:\ — диск C (Windows)
/Users/ — папка пользователей (Mac/Linux)
Иван или ivan — ваше имя пользователя
Documents — текущая папка

Практическое задание 1

Узнаем где мы находимся:

Откройте терминал (`Ctrl + ``)

Ведите: `pwd`

Посмотрите результат

Запомните где вы сейчас находитесь

2. Посмотреть что в текущей папке (Что здесь есть?)

Windows:

powershell

Способ 1 (краткий):

`dir`

Способ 2 (подробный):

`dir /A`

Способ 3 (PowerShell):

`Get-ChildItem`

Способ 4 (с цветами):

`ls`

Mac/Linux:

bash

Просто список:

`ls`

Подробный список:

`ls -la`

Только файлы:

`ls -p | grep -v /`

Что покажет команда:

Text

Directory: C:\Users\Иван\Documents

Mode	LastWrite	Time	Length	Name
d---	01.12.2024	10:00		Downloads
d---	01.12.2024	10:00		Pictures
d---	01.12.2024	10:00		Projects
-a---	01.12.2024	10:00	1024	notes.txt
-a---	01.12.2024	10:00	2048	todo.md

Расшифровка:

d — папка (directory)

- — файл

Имя — название папки или файла

Практическое задание 2

Посмотрим что в папке:

В терминале введите: dir (Windows) или ls (Mac/Linux)

Посмотрите список файлов и папок

Попробуйте подробный вариант: dir /A или ls -la

Найдите знакомые папки (Documents, Desktop и т.д.)

3. Перейти в другую папку (Пойдем туда!)

Базовый переход:

bash

Зайти в папку:

cd имя_папки

Пример:

cd Documents

cd Downloads

cd Pictures

Важно: Название папки должно существовать!

Практическое задание 3

Перейдем в другую папку:

Посмотрите какие папки есть (`dir` или `ls`)

Выберите папку (например, `Documents`)

Ведите: `cd Documents`

Проверьте где вы теперь: `pwd`

Посмотрите что в этой папке: `dir` или `ls`

4. Специальные переходы

Вернуться назад (на уровень выше):

```
bash  
cd ..  
# Точки означают "родительская папка"
```

Вернуться в корень диска:

```
bash  
# Windows:  
cd \  
  
# Mac/Linux:  
cd /
```

Вернуться домой (в свою папку пользователя):

```
bash  
# Windows:  
cd ~  
# или  
cd $HOME
```

```
# Mac/Linux:  
cd ~  
# или  
cd
```

Перейти на конкретный диск (только Windows):

```
powershell  
# Перейти на диск D:  
D:
```

```
# Перейти на диск C:  
C:
```

Практическое задание 4

Потренируем специальные переходы:

Перейдите в какую-нибудь папку: `cd Documents`

Вернитесь назад: `cd ..`

Проверьте: `pwd`

Вернитесь домой: `cd ~` или просто `cd`

Проверьте снова: `pwd`

5. Абсолютные и относительные пути

Относительный путь (от текущей папки):

```
bash  
# Если вы в C:\Users\Иван  
cd Documents/Projects  
# Перейдете в C:\Users\Иван\Documents\Projects
```

Абсолютный путь (полный адрес):

```
bash  
# Windows:  
cd C:\Users\Иван\Documents\Projects  
  
# Mac/Linux:  
cd /Users/ivan/Documents/Projects
```

Практическое задание 5

Сравним пути:

Узнайте где вы: `pwd`

Запомните полный путь

Перейдите куда-нибудь относительным путем: `cd Documents`

Вернитесь домой: `cd ~`

Перейдите туда же абсолютным путем (вставьте ваш путь)

Убедитесь, что пришли в ту же папку

6. Создание папок (Создадим новый дом)

Создать одну папку:

```
bash  
# Windows/Mac/Linux:  
mkdir новая_папка
```

```
# Пример:  
mkdir projects  
mkdir my_website  
mkdir test_folder
```

Создать несколько папок сразу:

```
bash  
# Windows:  
mkdir папка1, папка2, папка3
```

```
# Mac/Linux:  
mkdir папка1 папка2 папка3
```

Создать вложенные папки:

```
bash  
# Windows:  
mkdir проект\css\js\images
```

```
# Mac/Linux:  
mkdir -p проект/css/js/images
```

Практическое задание 6

Создадим структуру проекта:

Вернитесь домой: cd ~

Создайте папку проекта:

```
bash  
mkdir мой_сайт
```

Зайдите в нее:

```
bash  
cd мой_сайт
```

Создайте структуру:

```
bash  
# Windows:  
mkdir css, js, images
```

```
# Mac/Linux:  
mkdir css js images  
Проверьте: dir или ls
```

7. Удаление папок (Аккуратно!)

Удалить пустую папку:

```
bash  
# Windows:  
rmdir папка
```

```
# Mac/Linux:  
rmdir папка  
# или  
rm -d папка
```

Удалить папку с содержимым (ОСТОРОЖНО!):

```
bash  
# Windows (PowerShell):  
Remove-Item папка -Recurse -Force
```

```
# Windows (cmd):  
rmdir /s папка
```

```
# Mac/Linux:  
rm -rf папка
```

Внимание: Команда `rm -rf` или `rmdir /s` удаляет ВСЁ в папке без возможности восстановления!

Практическое задание 7

Создадим и удалим тестовую папку:

Создайте тестовую папку: `mkdir test_delete`

Проверьте, что она создалась: `dir` или `ls`

Удалите ее: `rmdir test_delete` (Windows) или `rmdir test_delete` (Mac/Linux)

Проверьте, что ее нет

8. Переименование папок

Изменить имя папки:

bash

Windows:

`Rename-Item старое_имя новое_имя`

или

`ren старое_имя новое_имя`

Mac/Linux:

`mv старое_имя новое_имя`

Практическое задание 8

Переименуем папку:

Создайте папку: `mkdir old_name`

Переименуйте:

Windows: `Rename-Item old_name new_name`

Mac/Linux: `mv old_name new_name`

Проверьте, что папка теперь с новым именем

9. Копирование папок

Скопировать папку:

bash

Windows:

`Copy-Item исходная_папка целевая_папка -Recurse`

```
# Mac/Linux:  
cp -r исходная_папка целевая_папка
```

Практическое задание 9

Скопириуем папку:

Создайте папку с файлом:

```
bash  
mkdir original  
echo "Тест" > original/test.txt
```

Скопирийте папку:

Windows: Copy-Item original copy -Recurse

Mac/Linux: cp -r original copy

Проверьте что в копии: dir copy или ls copy

10. Автодополнение (самый полезный трюк!)

Использование Tab для автодополнения:

Как это работает:

Начните печатать название папки

Нажмите Tab

Система допишет название за вас

Пример:

```
bash  
# У вас есть папка "Документы"  
# Вы печатаете:  
cd До # Нажимаете Tab  
# Становится: cd Документы
```

Практическое задание 10

Попрактикуем автодополнение:

Создайте папку с длинным именем: mkdir
очень_длинное_название_папки
Начните печатать: cd оче

Нажмите Tab

Команда допишется автоматически

Нажмите Enter

Полезные команды для удобства

Посмотреть историю команд:

bash

Windows (PowerShell):

[history](#)

Mac/Linux:

[history](#)

или используйте стрелки вверх/вниз

Очистить экран терминала:

bash

Windows:

[cls](#)

Mac/Linux:

[clear](#)

или [Ctrl + L](#)

Показать справку по команде:

bash

Windows:

[help cd](#)

или

[Get-Help cd](#)

Mac/Linux:

[man cd](#)

или

[cd --help](#)

Практическое задание 11

Используем вспомогательные команды:

Введите несколько команд: pwd, ls, cd ~

Посмотрите историю: history

Найдите свои команды в истории

Очистите экран: cls или clear

Посмотрите справку: help cd или man cd

Шпаргалка команд навигации

Windows (PowerShell):

text

pwd	- где я?
dir / ls	- что здесь?
cd папка	- зайти в папку
cd ..	- назад
cd ~	- домой
mkdir папка	- создать папку
rmdir папка	- удалить пустую папку
Remove-Item папка	- удалить с содержимым
Rename-Item	- переименовать
Copy-Item	- скопировать
cls	- очистить экран
help команда	- справка

Mac/Linux:

text

pwd	- где я?
ls	- что здесь?
cd папка	- зайти в папку
cd ..	- назад
cd ~ / cd	- домой
mkdir папка	- создать папку
rmdir папка	- удалить пустую папку
rm -rf папка	- удалить с содержимым (осторожно!)
mv старое новое	- переименовать/переместить
cp -r исход целевой	- скопировать
clear / Ctrl+L	- очистить экран
man команда	- справка

Чек-лист освоения

Умею узнать где нахожусь (pwd)
Могу посмотреть что в папке (dir/ls)
Умею заходить в папки (cd)
Знаю как вернуться назад (cd ..)
Могу вернуться домой (cd ~)
Умею создавать папки (mkdir)
Знаю как удалять папки (аккуратно!)
Могу переименовывать папки
Умею копировать папки
Использую автодополнение (Tab)
Знаю как очистить экран
Умею смотреть справку по командам

Профессиональные советы

Всегда знайте где вы — перед действием проверяйте pwd
Используйте Tab — это экономит время и предотвращает ошибки
Начинайте с относительных путей — проще и короче
Создавайте структуру перед началом проекта
Будьте осторожны с удалением — проверяйте что удаляете
Изучайте одну команду за раз — не пытайтесь запомнить всё сразу
Практикуйтесь ежедневно — лучше 5 минут каждый день, чем час раз в неделю

Итоговое упражнение

Создайте структуру веб-проекта:

Начните с домашней папки: cd ~
Создайте папку проекта: mkdir my_web_project
Зайдите в нее: cd my_web_project

Создайте структуру:

```
bash
mkdir css js images pages
```

Создайте файлы:

```
bash
# Windows:
echo. > index.html
echo. > css/style.css
```

```
echo. > js/script.js
```

```
# Mac/Linux:  
touch index.html  
touch css/style.css  
touch js/script.js
```

Посмотрите структуру:

```
bash  
# Windows:  
dir /s
```

```
# Mac/Linux:  
find . -type f
```

Перейдите в папку css: cd css

Вернитесь в корень проекта: cd ../../ или cd ~/my_web_project

Откройте проект в VS Code: code .

Поздравьте себя — вы создали проект через терминал!

Поздравляю! Теперь вы мастер навигации по папкам в терминале.
Вы можете свободно перемещаться по файловой системе,
создавать структуры проектов и управлять папками. Помните:
практика делает мастера — используйте терминал каждый день, и
скоро эти команды станут для вас естественными как дыхание!

Часть 6: Советы и хитрости для ежедневной работы

Глава 14: Горячие клавиши для экономии времени

Сохранение файла

Дорогой друг!

Теперь, когда мы немного освоились с навигацией, давайте поговорим о самом важном действии в любой работе — **сохранении файла**. В веб-разработке это то же самое, что положить только что написанное письмо в конверт и отправить его. Если не сохранить — ваша работа может «потеряться в пути».

Сохранять файлы в VS Code очень просто, и есть несколько способов сделать это. Давайте разберем их все!

Самые важные способы сохранения

1. Быстрое сохранение одного файла (Классический способ)

Это самый частый и основной способ. Вы работаете с файлом и хотите зафиксировать все изменения.

Что делаем: Просто сохраняем текущий открытый файл.

Команда на клавиатуре:

Windows/Linux: Ctrl + S

Mac: Cmd + S (⌘ + S)

Как это работает:

Вы написали код в файле index.html.

Нажимаете Ctrl + S.

В **строке заголовка** (верхняя часть окна VS Code) рядом с именем файла пропадет маленькая белая точка •. **Эта точка — индикатор несохраненных изменений. Нет точки — файл сохранен!**

Файл записан на ваш жесткий диск. Всё готово!

Совет: Привыкайте нажимать **Ctrl + S** постоянно, после каждого небольшого логического куска кода. Это должно стать рефлексом, как моргнуть. Это спасет вас от потерь, если что-то пойдет не так.

2. Сохранение всех открытых файлов разом (Уборка в мастерской)

Представьте, что вы поработали над несколькими файлами (например, `index.html`, `style.css`, `script.js`) и в каждом есть несохраненные изменения (белые точки горят). Сохранять каждый по отдельности долго.

Что делаем: Сохраняем разом все файлы, в которых есть изменения.

Команда на клавиатуре:

Windows/Linux: **Ctrl + K**, затем **S** (Нужно нажать две клавиши последовательно: удерживая **Ctrl**, жмем **K**, отпускаем обе, затем жмем **S**).

Mac: **Cmd + K**, затем **S** (**⌘ + K**, потом **S**).

Альтернатива через меню: **Файл (File) → Сохранить все (Save All)**.

Когда это полезно: Перед запуском проекта в браузере, перед закрытием программы или просто для наведения порядка.

3. Сохранение с новым именем или в другую папку («Сохранить как...»)

Вы создали отличный шаблон кнопки в файле `button.html` и теперь хотите создать на его основе другую кнопку, не переписывая исходник.

Что делаем: Создаем копию текущего файла с другим именем или в другой папке. Оригинальный файл остается нетронутым.

Команда на клавиатуре:

Windows/Linux: **Ctrl + Shift + S**

Mac: **Cmd + Shift + S** (**⌘ + ⌘ + S**)

Альтернатива через меню: **Файл (File) → Сохранить как... (Save As...)**.

Что произойдет: Откроется стандартное окно проводника вашей операционной системы, где вы сможете выбрать папку, ввести новое имя файла (например, button_blue.html) и нажать «Сохранить».

4. Автоматическое сохранение (Помощник, который не забывает)

VS Code может сохранять файлы за вас автоматически! Это очень удобная опция.

Как включить:

Нажмите **Файл (File) → Настройки (Preferences) → Параметры (Settings)**.

В поиске настроек введите `autosave`.

Найдите пункт «**Files: Auto Save**».

Варианты настройки:

`off`: Автосохранение выключено. (По умолчанию). Сохраняете сами.

`afterDelay`: Сохранить через короткое время после остановки печати. **Самый популярный вариант!** Рекомендую выбрать его и установить задержку (**Files: Auto Save Delay**) в 1000 миллисекунд (1 секунда).

`onFocusChange`: Сохраняет файл, когда вы переключаетесь с него на другое окно или вкладку.

`onWindowChange`: Сохраняет все, когда вы переключаетесь на другое приложение (например, в браузер).

Важно! Даже с включенным автосохранением, привычка нажимать `Ctrl+S` перед важными действиями (запуск кода, тестирование) — это признак хорошего тона.

Практическое задание: Учимся сохранять

Цель: Создать и сохранить ваш первый HTML-файл.

Создайте файл: В боковой панели VS Code нажмите правой кнопкой мыши на пустом месте вашей папки проекта и выберите «**Новый файл**». Назовите его `моя_первая_страница.html`.

Напишите базовый код: Вставьте в него этот простой код:

```
html
<!DOCTYPE html>
<html>
<head>
    <title>Моя первая страница</title>
</head>
<body>
    <h1>Привет, мир! Я это сделал(а)!</h1>
    <p>Сегодня я научился(ась) сохранять файлы в VS Code.</p>
</body>
</html>
```

Посмотрите на индикатор: Убедитесь, что в заголовке вкладки есть белая точка •.

Сохранение 1 (Вручную): Нажмите `Ctrl + S` (или `Cmd + S`). Точка должна исчезнуть.

Сохранение 2 («Сохранить как»): Нажмите `Ctrl + Shift + S`. Сохраните копию файла с именем `моя_вторая_страница.html`. Теперь у вас в панели слева два файла!

Проверка: Откройте папку вашего проекта в Проводнике (Windows) или Finder (Mac). Убедитесь, что файлы действительно там лежат.

Откройте в браузере: Щелкните правой кнопкой мыши по файлу `моя_первая_страница.html` в боковой панели VS Code и выберите «**Copy Path**». Вставьте этот путь в адресную строку вашего браузера (Chrome, Edge и т.д.). Вы должны увидеть свою страницу с большим заголовком!

Ключевые выводы

`Ctrl + S` — ваш лучший друг. Нажимайте его часто.

Белая точка • рядом с именем файла — сигнал «есть несохраненные изменения».

Автосохранение (afterDelay) — отличный помощник, но не отменяет вашу внимательность.

«Сохранить как...» — инструмент для быстрого создания копий и вариаций.

Поздравляю! Вы только что освоили один из самых фундаментальных навыков в работе программиста. Теперь ваши творения в безопасности!

Копирование и вставка

Дорогой друг!

Давайте поговорим о двух самых магических действиях в цифровом мире — **копировании и вставке!** В веб-разработке это не просто удобно, это **суперспособность**, которая экономит часы работы.

Представьте, что вы нашли в интернете красивый пример кода или написали идеальную кнопку, которую хотите использовать в нескольких местах. Зачем писать всё заново, если можно просто «перенести» это?

В VS Code копировать и вставлять можно множеством способов — давайте разберем все, как опытные мастера!

Самые важные способы копирования и вставки

1. Классический способ (как в любом редакторе)

Это тот метод, который вы, скорее всего, уже знаете. Он работает везде: в текстовых редакторах, браузере, почте.

Копирование (взять в буфер обмена)

Windows/Linux: Ctrl + C

Mac: Cmd + C (⌘ + C)

Как это работает:

Выделите нужный текст мышкой (зажмите левую кнопку и проведите) или клавишами со стрелками, удерживая Shift.

Нажмите **Ctrl + C**. Текст **не исчезает** с экрана, а невидимо сохраняется в «буфере обмена» (временной памяти компьютера).

Вставка (положить из буфера)

Windows/Linux: **Ctrl + V**

Mac: **Cmd + V** (**⌘ + V**)

Как это работает:

Поставьте курсор (мигающую черточку) в то место, куда хотите вставить текст.

Нажмите **Ctrl + V**. Текст появится в новом месте!

Полезный брат-близнец: Вырезать (**Ctrl + X**)

Это копирование + удаление оригинала. Идеально для **переноса** текста из одного места в другое.

2. Копирование целых строк и блоков кода (Секретное оружие верстальщика)

В программировании часто нужно работать не с произвольным куском текста, а с **целыми строками**. VS Code для этого создан!

Копирование текущей строки (где стоит курсор)

Команда: **Ctrl + C** (без выделения текста!)

Что произойдет: Если у вас **не выделен никакой текст**, а вы просто поставили курсор на строку и нажали **Ctrl + C** — скопируется **вся строка целиком**. Магия!

Копирование или перемещение строк вверх/вниз

Это одна из самых полезных функций!

Переместить строку вниз: **Alt + ↓** (Win/Linux) / **Option + ↓** (Mac)

Переместить строку вверх: **Alt + ↑** (Win/Linux) / **Option + ↑** (Mac)

Скопировать строку вверх/вниз: **Shift + Alt + ↓** или **Shift + Alt + ↑**

Практический пример в HTML:

```
html
<ul>
<li>Яблоко</li>
<li>Апельсин</li> <!-- Курсор на этой строке --&gt;
&lt;li&gt;Банан&lt;/li&gt;
&lt;/ul&gt;</pre>
```

Нажмите Shift + Alt + ↓, и строка Апельсин мгновенно продублируется ниже. Теперь у вас два апельсина без ручного копирования!

3. Множественное копирование и вставка (Работа с несколькими курсорами — суперсила!)

Это продвинутая, но невероятно мощная функция. Иногда нужно изменить одно и то же слово в нескольких местах одновременно.

Способ 1: Выделить все вхождения слова

Кликните мышью на слове, которое нужно заменить (например, color).

Нажмите Ctrl + D (Win/Linux) / Cmd + D (Mac).

VS Code выделит **следующее** такое же слово. Нажимайте Ctrl + D дальше, пока не выделятся все нужные слова.

Теперь просто начните печатать — текст изменится во всех выделенных местах сразу!

Способ 2: Добавить курсоры кликом мыши

Удерживайте Alt и кликайте левой кнопкой мыши в разные места текста. Появятся несколько мигающих курсоров. Всё, что вы начнете печатать, появится во всех этих местах одновременно. Для отмены — нажмите Esc.

4. Копирование и вставка между файлами и программами

Буфер обмена один на весь компьютер! Это значит:

Вы можете скопировать (Ctrl+C) красивый пример кода из браузера (с сайта учебника).

Переключиться в VS Code на свой файл style.css.

Вставить (Ctrl+V) этот код прямо туда.

И наоборот: скопировать свою работу из VS Code и вставить в письмо или сообщение коллеге.

Практическое задание: Становимся мастером копирования

Цель: Создать список навигации для сайта, используя разные методы копирования.

Откройте файл index.html, который мы создали ранее.

Создайте основу списка: После тега `<h1>` добавьте:

```
html
<nav>
<ul>
<li>Главная</li>
</ul>
</nav>
```

Потренируем «Копирование строки»:

Поставьте курсор на строку `Главная`.

Нажмите `Shift + Alt + ↓` 3 раза. У вас появится 4 одинаковых пункта списка.

Используем «Множественное редактирование»:

Кликните на слове «**Главная**» во втором пункте списка.

Нажмите `Ctrl + D` два раза, чтобы выделить это слово в третьем и четвертом пункте.

Теперь, не снимая выделения, просто напечатайте: **О нас**. Все три выделенных слова изменятся одновременно!

Повторите для оставшихся пунктов, переименовав их в **Услуги** и **Контакты**.

Скопируем готовый блок:

Выделите мышью весь блок `<nav>...</nav>`.

Нажмите `Ctrl + C`.

Перейдите в самый низ файла, перед закрывающим `</body>`.

Нажмите `Ctrl + V`. Теперь у вас два одинаковых меню на странице (вверху и внизу) — это обычная практика для сайтов!

Скопируем ссылку из браузера:

Откройте браузер, найдите любой сайт (например, [yandex.ru](#)).

Скопируйте адрес из адресной строки браузера (**Ctrl+C**).

Вернитесь в VS Code.

В первом пункте меню замените `Главная` на `Главная`.

Выделите символ `#` внутри `href="#"`.

Нажмите **Ctrl + V**. Символ `#` заменится на реальный URL из буфера обмена!

Важные нюансы и советы

Буфер обмена хранит только последнюю скопированную вещь. Скопировали новое — старое забылось.

Расширение «Multiple Clipboards»: Если вам нужно хранить несколько фрагментов для вставки, установите это расширение из Marketplace. Оно даст вам историю буфера обмена.

Форматирование: При вставке кода из других источников VS Code часто предлагает его **отформатировать** (красиво расставить отступы). Примите это предложение или нажмите **Shift + Alt + F**.

Копирование пути к файлу: Щелкните правой кнопкой мыши по файлу в боковой панели Explorer и выберите «**Copy Path**» или «**Copy Relative Path**». Это очень полезно для указания путей к картинкам или CSS-файлам.

Ключевые выводы

Ctrl+C / Ctrl+V — основа основ. Работает везде.

Ctrl+X — чтобы перемещать, а не копировать.

Shift+Alt+↓/↑ — волшебная кнопка для дублирования строк. Запомните ее!

Ctrl+D — для одновременного изменения одинаковых слов.

Буфер обмена общий — свободно копируйте код между программами.

Теперь вы вооружены техниками, которые используют профессиональные разработчики каждый день. Копируйте с умом, и пусть ваш код множится и процветает! Удачи в практике, дорогой друг

Комментирование кода

Дорогой друг!

Сегодня мы поговорим об одном из самых важных навыков не только для начинающих, но и для опытных разработчиков — **комментировании кода**.

Комментарии — это заметки, которые вы оставляете себе и другим в коде. Они похожи на записки на полях книги или пояснения в рецепте. Компьютер их полностью игнорирует, но для человека они бесценны!

Зачем нужны комментарии?

-  **Объяснить сложный код** — чтобы через месяц вы сами поняли, что хотели сделать
-  **Помочь коллегам** — если над проектом работает несколько человек
-  **Временно отключить код** — для тестирования разных вариантов
-  **Разметить структуру** — создать оглавление для длинного файла
-  **Оставить напоминания** — что нужно доделать или исправить

Самые важные способы комментирования

1. Однострочные комментарии (Короткие заметки)

Идеально для кратких пояснений рядом с кодом.

HTML комментарии

```
html
<!-- Это комментарий в HTML -->
<div class="header">
<!-- Здесь будет логотип сайта -->


<!-- Навигационное меню - добавить позже -->
<!-- <nav>Меню в разработке</nav> -->
</div>
```

Как создать: <!-- ваш текст -->

Совет: В VS Code есть горячие клавиши! Выделите текст и нажмите **Ctrl + /** (Win/Linux) или **Cmd + /** (Mac)

CSS комментарии

```
css
/* Это комментарий в CSS */
.header {
background-color: #f0f0f0; /* Светло-серый фон */
padding: 20px; /* Отступы вокруг */

/* Временно убрано для теста
border: 1px solid #ccc;
box-shadow: 0 2px 5px rgba(0,0,0,0.1);
*/
}
```

Как создать: /* ваш текст */

Горячие клавиши: Тот же Ctrl + / или Cmd + /

JavaScript комментарии

```
javascript
// Это однострочный комментарий в JavaScript
let userName = "Иван"; // Здесь хранится имя пользователя

// Следующая строка выводит приветствие в консоль
console.log("Привет, " + userName + "!");

// TODO: Добавить проверку возраста пользователя
// FIXME: Исправить расчет скидки после 18:00
```

Как создать: // ваш текст

Горячие клавиши: Тот же Ctrl + / или Cmd + /

2. Многострочные комментарии (Длинные объяснения)

Когда нужно закомментировать большой блок кода или написать подробное описание.

HTML (также работает для CSS внутри style-тега)

```
html
<!--
БЛОК: ШАПКА САЙТА
Автор: Иван Петров
Дата создания: 15.05.2023
Описание: Этот блок содержит логотип,
основную навигацию и поисковую строку.
```

Структура:

1. Логотип (слева)
2. Навигационное меню (центр)
3. Кнопка корзины (справа)

-->

```
<!--
<div class="old-header">
Этот код устарел, но мы его пока не удаляем.
Он может пригодиться для восстановления старой версии.
</div>
-->
```

CSS и JavaScript

```
CSS
/*
ФАЙЛ: main.css
НАЗНАЧЕНИЕ: Основные стили сайта
СОДЕРЖАНИЕ:
1. Сброс стилей (reset)
2. Типографика (шрифты)
3. Основная сетка (grid)
4. Компоненты (кнопки, формы)
```

ИСТОРИЯ ИЗМЕНЕНИЙ:

2023-05-15: Добавлена темная тема
2023-04-10: Оптимизация для мобильных
*/

/* ===== СБРОС СТИЛЕЙ ===== */

```
* {  
margin: 0;  
padding: 0;  
box-sizing: border-box;  
}
```

3. Специальные комментарии (TODO, FIXME, NOTE)

Это особые виды комментариев, которые многие редакторы (включая VS Code) умеют находить и показывать в специальном списке.

javascript

```
// TODO: Добавить валидацию email перед отправкой формы  
// FIXME: На мобильных устройствах кнопка съезжает  
// NOTE: Этот алгоритм оптимизирован для больших данных  
// HACK: Временное решение, нужно переписать  
// BUG: При скролле иногда пропадает верхнее меню  
// OPTIMIZE: Можно ускорить загрузку изображений  
  
// В VS Code эти комментарии можно увидеть:  
// 1. Нажмите Ctrl+Shift+P (или Cmd+Shift+P на Mac)  
// 2. Введите "Show TODO"  
// 3. Выберите "Todo Tree: Show TODO Tree"  
// 4. Слева появится панель со всеми такими комментариями!
```

Практическое задание: Учимся правильно комментировать

Цель: Создать хорошо прокомментированную HTML-страницу с CSS-стилями.

Шаг 1: Создайте базовую структуру

```
html  
<!DOCTYPE html>  
<html lang="ru">  
<head>  
<!--  
МЕТА-ИНФОРМАЦИЯ  
Здесь настраивается кодировка, заголовок и подключение стилей
```

```
-->
<meta charset="UTF-8">
<title>Мой сайт - Главная</title>

<!-- Подключаем наш файл стилей -->
<link rel="stylesheet" href="style.css">
</head>
<body>
<!-- TODO: Добавить шапку сайта -->

<!-- БЛОК: ОСНОВНОЕ СОДЕРЖИМОЕ -->
<main>
<h1>Добро пожаловать на мой сайт!</h1>

<!-- Параграф с приветствием -->
<p>Здесь я учусь веб-разработке.</p>

<!-- FIXME: Кнопка должна быть зеленой, а не синей -->
<button class="cta-button">Начать обучение</button>
</main>

<!-- ПОДВАЛ САЙТА -->
<footer>
<!-- NOTE: В подвале нужно добавить ссылки на соцсети -->
<p>&copy; 2023 Мой учебный сайт</p>
</footer>
</body>
</html>
```

Шаг 2: Создайте и прокомментируйте CSS

Создайте файл `style.css` и добавьте:

```
CSS
/*
ФАЙЛ: style.css
НАЗНАЧЕНИЕ: Основные стили для учебного сайта
АВТОР: Иван Петров
*/
/* ===== БАЗОВЫЕ СТИЛИ ===== */
```

```
body {
font-family: Arial, sans-serif; /* Основной шрифт */
line-height: 1.6; /* Межстрочный интервал */
color: #333; /* Цвет текста */
margin: 0;
padding: 0;

/* Временно для наглядности */
/* border: 1px dashed red; */
}

/* ===== ОСНОВНОЕ СОДЕРЖИМОЕ ===== */
main {
max-width: 800px; /* Ограничиваем ширину */
margin: 0 auto; /* Центрируем блок */
padding: 20px;

/* TODO: Добавить фоновое изображение */
background-color: #f9f9f9; /* Светлый фон */
}

/* ===== КНОПКИ ===== */
.cta-button {
/* FIXME: Изменить цвет на зеленый (#4CAF50) */
background-color: #007bff; /* Синий цвет */
color: white; /* Белый текст */
padding: 12px 24px; /* Отступы внутри */
border: none; /* Убираем рамку */
border-radius: 4px; /* Закругляем углы */
cursor: pointer; /* Меняем курсор при наведении */
font-size: 16px;

/* Эффект при наведении */
transition: background-color 0.3s ease;
}

.cta-button:hover {
background-color: #0056b3; /* Темнее при наведении */
}
```

```
/* ===== ПОДВАЛ ===== */
footer {
    /* NOTE: Стили для подвала */
    background-color: #333; /* Темный фон */
    color: white; /* Белый текст */
    text-align: center; /* Выравнивание по центру */
    padding: 20px;
    margin-top: 40px; /* Отступ сверху */
}

/* Медиа-запрос для мобильных устройств */
@media (max-width: 600px) {
    /* На маленьких экранах уменьшаем отступы */
    main {
        padding: 10px;
    }
}

.cta-button {
    width: 100%; /* Кнопка на всю ширину */
}
```

Шаг 3: Поработайте с горячими клавишами

Откройте HTML-файл в VS Code

Выделите любую строку с кодом

Нажмите `Ctrl + /` (Win/Linux) или `Cmd + /` (Mac)

Наблюдайте, как строка закомментируется/раскомментируется

Попробуйте выделить несколько строк и нажать эту же комбинацию

Что такое "закомментировать код" и зачем это нужно?

Закомментировать код — значит временно отключить его выполнение, не удаляя из файла. Это как поставить блюдо в холодильник, а не выбросить.

Зачем это нужно?

Тестирование: Проверить, как работает программа без определенного блока

Отладка: Поиск ошибок, поочередно отключая части кода

Хранение старого кода: На всякий случай, если новая версия не сработает

Обучение: Показать разные варианты решения одной задачи

Пример отладки с помощью комментариев:

javascript

```
// Исходный код с ошибкой
function calculateTotal(price, quantity) {
// let total = price * quantity; // Старая формула
let total = price + quantity; // ОШИБКА: должно быть умножение!

// console.log("Цена:", price); // Для отладки
// console.log("Количество:", quantity);
console.log("Итого:", total);

return total;
}
```

// Раскомментируйте строки 4-6, чтобы увидеть значения переменных

Лучшие практики комментирования

✓ ХОРОШО:

javascript

```
// Рассчитываем площадь круга: π * r2
function circleArea(radius) {
return Math.PI * radius * radius;
}
```

✗ ПЛОХО:

javascript

```
// Функция circleArea принимает радиус и возвращает площадь
function circleArea(radius) {
// Умножаем π на радиус в квадрате
return Math.PI * radius * radius; // Возвращаем результат
}
```

Правила хорошего тона:

Пишите "ПОЧЕМУ", а не "ЧТО" — код и так показывает, что происходит

Комментируйте сложные алгоритмы, но не очевидные вещи
Обновляйте комментарии вместе с кодом
Используйте TODO/FIXME для пометок
Не комментируйте каждую строку — это делает код нечитаемым

Полезные расширения VS Code для работы с комментариями

Todo Tree — находит все TODO, FIXME, NOTE в проекте
Better Comments — раскрашивает разные типы комментариев
Document This — автоматически создает комментарии для функций

Ключевые выводы

Ctrl + / или **Cmd + /** — волшебная кнопка для быстрого комментирования
<!-- --> — для HTML
/* */ — для CSS и многострочных комментариев
// — для JavaScript и однострочных комментариев
TODO/FIXME — специальные метки, которые VS Code умеет искать
Комментарии — это документация внутри кода
Комментируйте с умом — слишком много комментариев так же плохо, как и их отсутствие

Помните: Хорошо прокомментированный код — это подарок себе в будущем. Через месяц вы точно вспомните, что хотели сделать и почему выбрали именно такое решение!

Удачи в освоении этого важного навыка, дорогой друг! Ваш будущий "я" будет вам благодарен.

Быстрое форматирование текста

Дорогой друг!

Сегодня мы освоим настоящее волшебство — **быстрое форматирование текста** в VS Code. Это как научиться готовить по рецепту, где все ингредиенты нарезаются и смешиваются одним движением руки!

Зачем нужно быстрое форматирование?

- ⚡ **Экономия времени** — минуты вместо часов ручной работы
 - ✨ **Создание структуры** — заголовки, списки, таблицы за секунды
 - 🎨 **Единый стиль** — весь код выглядит аккуратно и профессионально
 - 🔄 **Легкое редактирование** — быстро менять тип элементов
-

Самые важные инструменты быстрого форматирования

1. Emmet — суперсила для верстальщиков!

Emmet — это встроенный в VS Code инструмент, который превращает короткие аbbревиатуры в полноценный HTML/CSS код. Это главный секрет профессионалов!

Базовые конструкции HTML

Просто наберите и нажмите Tab:

html

<!-- Создаст тег div -->

div → <div></div>

<!-- Создаст заголовок h1 -->

h1 → <h1></h1>

<!-- Создаст параграф с текстом -->

p{Привет мир!} → <p>Привет мир!</p>

<!-- Создаст ссылку -->

a{Кликни меня} → Кликни меня

<!-- Создаст изображение -->

img →

Создание структуры с вложенностью

html

<!-- Создать div с классом container -->
.container → <div class="container"></div>

<!-- Создать div с id header -->
#header → <div id="header"></div>

<!-- Создать навигацию с списком -->
nav>ul>li*3 →
<nav>

</nav>

<!-- Создать карточку товара -->
.card>img+h3{Название}+p{Описание}+button{Купить} →
<div class="card">

<h3>Название</h3>
<p>Описание</p>
<button>Купить</button>
</div>

Создание целых страниц одной строкой!

html

<!-- Создать базовую структуру HTML5 страницы -->
! → Нажмите Tab и получите:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
```

```
</head>
<body>

</body>
</html>
```

Как это работает:

> — создает вложенность (родитель > ребенок)
+ — создает соседние элементы
* — умножает элемент (li*5 создаст 5 пунктов списка)
. — добавляет класс
— добавляет id
{ } — добавляет текст внутрь тега

2. Быстрое форматирование CSS

Emmet работает и для CSS! Наберите сокращение и нажмите Tab:

css

```
/* Сокращения для свойств */
m10 → margin: 10px;
p20-30 → padding: 20px 30px;
bd1-s-#000 → border: 1px solid #000;
bg#f00 → background: #ff0000;
c#fff → color: #fff;
w100p → width: 100%;
h50px → height: 50px;
dib → display: inline-block;
df → display: flex;
jcc → justify-content: center;
aic → align-items: center;
posa → position: absolute;
t0 → top: 0;
r0 → right: 0;
b0 → bottom: 0;
l0 → left: 0;
```

```
/* Значения с префиксами */
trf → transform: ;
bxz → box-sizing: border-box;
```

3. Множественное редактирование (Multi-cursor)

Мы уже касались этой темы, но для форматирования это особенно полезно!

Способ 1: Добавить курсоры к одинаковым словам

html

```
<!-- Исходный код -->
<h2>Заголовок 1</h2>
<p>Текст 1</p>
<h2>Заголовок 2</h2>
<p>Текст 2</p>
<h2>Заголовок 3</h2>
<p>Текст 3</p>
```

<!-- Хотим все h2 сделать h3 -->

1. Кликните на первом "h2"
2. Нажмите Ctrl+D два раза (выделятся все "h2")
3. Напечатайте "h3" — все заголовки изменятся!

Способ 2: Добавить курсоры стрелками

html

```
<!-- Нужно добавить класс ко всем li -->
<ul>
<li>Яблоко</li>
<li>Груша</li>
<li>Слива</li>
</ul>
```

1. Поставьте курсор после первого
 2. Нажмите Alt+Shift+↓ (добавится курсор на следующей строке)
 3. Нажмите еще раз (добавится третий курсор)
 4. Напечатайте: class="fruit-item"
 5. Получится:
- ```
<li class="fruit-item">Яблоко
<li class="fruit-item">Груша
<li class="fruit-item">Слива
```

## Способ 3: Добавить курсоры мышкой

Удерживайте Alt и кликайте левой кнопкой мыши в нужных местах. Или выделите мышкой вертикальный блок, удерживая Alt+Shift.

---

## 4. Обернуть текст в теги (Wrap with Abbreviation)

Это золотая функция для быстрого форматирования!

html

<!-- Исходный текст без тегов: -->

Главная

О нас

Контакты

<!-- Чтобы обернуть каждую строку в тег <li>: -->

1. Выделите все три строки
2. Нажмите Ctrl+Shift+P (или Cmd+Shift+P на Mac)
3. Введите "Wrap with Abbreviation"
4. Введите "li\*" (это значит: создать li для каждой строки)
5. Нажмите Enter

<!-- Результат: -->

<li>Главная</li>

<li>О нас</li>

<li>Контакты</li>

<!-- Можно сразу создать сложную структуру: -->

Выделите текст → Wrap with Abbreviation → "ul>li\*"

Получится:

<ul>

<li>Главная</li>

<li>О нас</li>

<li>Контакты</li>

</ul>

**Бонус:** Есть горячая клавиша! Ctrl+Shift+A (Win) или Cmd+Shift+A (Mac).

---

## 5. Автоформатирование всего документа

Когда код стал "рваным" и неаккуратным, VS Code может привести его в порядок за секунду!

html

```
<!-- ДО форматирования (всё в одну строку): -->
<div><h1>Привет</h1><p>Текст внутри</p></div>
```

<!-- ПОСЛЕ форматирования: -->

1. Нажмите Shift+Alt+F (Win) или Shift+Option+F (Mac)
2. Или: правой кнопкой мыши → "Format Document"

<!-- Результат: -->

```
<div>
<h1>Привет</h1>
<p>Текст внутри</p>
</div>
```

**Для HTML:** VS Code использует встроенный форматтер.

**Для CSS/JS:** Можно установить расширения типа Prettier для более тонкой настройки.

---

## Практическое задание: Создаем страницу за 5 минут

**Цель:** Используя Emmet и быстрые команды, создать структуру интернет-магазина.

### Шаг 1: Создайте базовую страницу

Откройте новый файл index.html

Напечатайте ! и нажмите Tab

Измените lang="en" на lang="ru"

В <title> напишите "Интернет-магазин"

### Шаг 2: Создайте шапку сайта

Внутри <body> напишите:

html

```
header.container>nav>ul>li*4>a
```

Нажмите Tab и вы получите:

```
html
<header class="container">
<nav>

</nav>
</header>
```

Теперь:

Выделите все четыре </a> (поставьте курсор на первый, нажмите Ctrl+D три раза)

Напечатайте {\$} — это специальный символ Emmet для нумерации

Получится:

```
html
1
2
3
4
```

Вручную поменяйте цифры на: "Главная", "Каталог", "О нас", "Контакты"

### Шаг 3: Создайте секцию с товарами

Под header напишите:

```
html
section#products>.product*3>img+h3{Товар $}+p{Описание товара $}
+span.price{1000₽}+button{Купить}
```

Нажмите Tab — получите 3 одинаковые карточки товара с нумерацией!

### Шаг 4: Быстро добавить классы кнопкам

Выделите все три button (клик на первом, Ctrl+D два раза)

Напечатайте: class="btn btn-primary"

Результат:

```
html
<button class="btn btn-primary">Купить</button>
<button class="btn btn-primary">Купить</button>
<button class="btn btn-primary">Купить</button>
```

## Шаг 5: Создайте подвал

В конце body напишите:

```
html
footer>.container>p{© 2023 Магазин}>ul.social>li*3>a{Соцсеть $}
```

## Шаг 6: Отформатируйте весь документ

Нажмите Shift+Alt+F — и ваш код станет красиво отформатированным с правильными отступами!

---

## Полезные сочетания клавиш для форматирования

Действие	Windows/ Linux	Mac
Форматировать документ	Shift+Alt+F	Shift+Option+F
Форматировать выделенное	Ctrl+K Ctrl+F	Cmd+K Cmd+F
Обернуть в тег	Ctrl+Shift+A	Cmd+Shift+A
Дублировать строку	Shift+Alt+↓/↑	Shift+Option+↓/↑
Добавить курсор выше/ниже	Ctrl+Alt+↑/↓	Ctrl+Option+↑/↓
Выделить все вхождения слова	Ctrl+D	Cmd+D
Добавить комментарий	Ctrl+/	Cmd+/
Переместить строку	Alt+↑/↓	Option+↑/↓

---

## Расширения для еще более быстрого форматирования

**Auto Rename Tag** — меняет открывающий и закрывающий тег одновременно

**HTML CSS Support** — автодополнение классов из CSS файлов

**Prettier** — мощный форматтер для HTML/CSS/JS

**Bracket Pair Colorizer** — подсвечивает парные скобки разными цветами

**Live Server** — мгновенный просмотр изменений в браузере

---

## Типичные ошибки и как их избежать

### ✗ Ошибка: Emmet не работает

**Решение:** Убедитесь, что:

Файл сохранен с правильным расширением (.html, .css)

В нижнем правом углу VS Code указан правильный язык (HTML, CSS)

Emmet включен: File → Preferences → Settings → поиск "Emmet"

### ✗ Ошибка: Неправильные отступы после форматирования

**Решение:** Настройте табуляцию:

Нажмите на пробелы/табы в нижней синей строке

Выберите "Indent Using Spaces" или "Indent Using Tabs"

Обычно используют 2 или 4 пробела для HTML/CSS

### ✗ Ошибка: Форматирование ломает код

**Решение:** Всегда сохраняйте перед форматированием (Ctrl+S). Если что-то пошло не так — нажмите Ctrl+Z для отмены.

---

## Советы от профессионалов

**Изучите 10-15 самых частых Emmet-сокращений** — этого хватит для 80% работы

**Настройте свои сниппеты** (заготовки кода) через File → Preferences → User Snippets

**Используйте множественные курсоры для однотипных правок**

**Форматируйте документ перед сохранением** — так код всегда будет аккуратным

**Создавайте свои шаблоны** для часто используемых блоков (шапка, подвал, карточки)

---

## Ключевые выводы

**Emmet** — ваш главный помощник в верстке. Tab — волшебная кнопка!

**! + Tab** — создает базовую структуру HTML-страницы

**Shift+Alt+F** — мгновенно приводит код в порядок

**Множественные курсоры** (**Ctrl+D**, **Alt+клик**) — для массовых правок

**Обертывание тегами** (**Ctrl+Shift+A**) — превращает текст в HTML-структуру

**Сохраняйте часто** (**Ctrl+S**) — особенно перед сложными операциями

**Помните:** Чем больше вы практикуете эти приемы, тем быстрее они станут вашей второй натурой. Скоро вы будете создавать целые страницы быстрее, чем другие набирают один абзац!

Успехов в освоении этого волшебства, дорогой друг!

## Глава 15: Работа с цветами

### Встроенный выбор цветов для CSS

#### Дорогой друг!

Сегодня мы будем творить чудеса с цветами прямо в VS Code! Представьте, что у вас есть волшебная палитра художника,строенная прямо в редактор кода. Не нужно угадывать шестнадцатеричные коды или искать цвета в интернете — всё под рукой!

#### Зачем нужен встроенный выбор цветов?

-  **Визуальный выбор** — видите цвет сразу, не запоминая коды
-  **Быстрое редактирование** — меняете цвета за секунды
-  **Разные форматы** — HEX, RGB, HSL, названия цветов
-  **Точные значения** — никаких ошибок в кодах
-  **Прозрачность** — легко настраивать полупрозрачные цвета

---

#### Самые важные способы работы с цветами

##### 1. Цветной подсказчик — ваш первый помощник

VS Code автоматически подсвечивает цветовые значения в CSS!

CSS

```
/* Просто начните писать цвет - увидите магию! */
.selector {
 color: #ff0000; /* Подсветится красным квадратиком */
 background: rgb(0, 0, 255); /* Синий квадратик */
 border-color: green; /* Зеленый квадратик */
}
```

#### Как это работает:

В любом файле CSS (или в <style> в HTML)

Напишите свойство, которое принимает цвет

Начните вводить цветовое значение

VS Code покажет маленький цветной квадратик слева от значения

## 2. Открываем палитру цветов — основная магия!

Это самая полезная функция. Не нужно помнить коды!

CSS

```
.element {
/* Подведите курсор к любому цветному квадратику */
color: #3498db; /* Наведите курсор на этот код */
}
```

**Что произойдет:**

Наведите курсор мыши на цветной квадратик или на сам код цвета  
**Появится всплывающее окно с информацией о цвете**

В этом окне будет:

Большой образец цвета

Текущее значение в разных форматах

Кнопка для открытия полноценной палитры

---

## 3. Полноценная палитра цветов — выбираем как художники!

Чтобы открыть полноценный выборщик цветов:

**Способ 1: Клик по квадратику**

Найдите цветное значение в CSS

Нажмите **ЛЕВОЙ кнопкой мыши** на цветной квадратик

Откроется палитра цветов!

**Способ 2: Горячие клавиши**

Установите курсор на цветовое значение

Нажмите **Ctrl + Shift + C** (Windows/Linux) или **Cmd + Shift + C** (Mac)

**Способ 3: Контекстное меню**

Кликните правой кнопкой мыши по цветовому значению

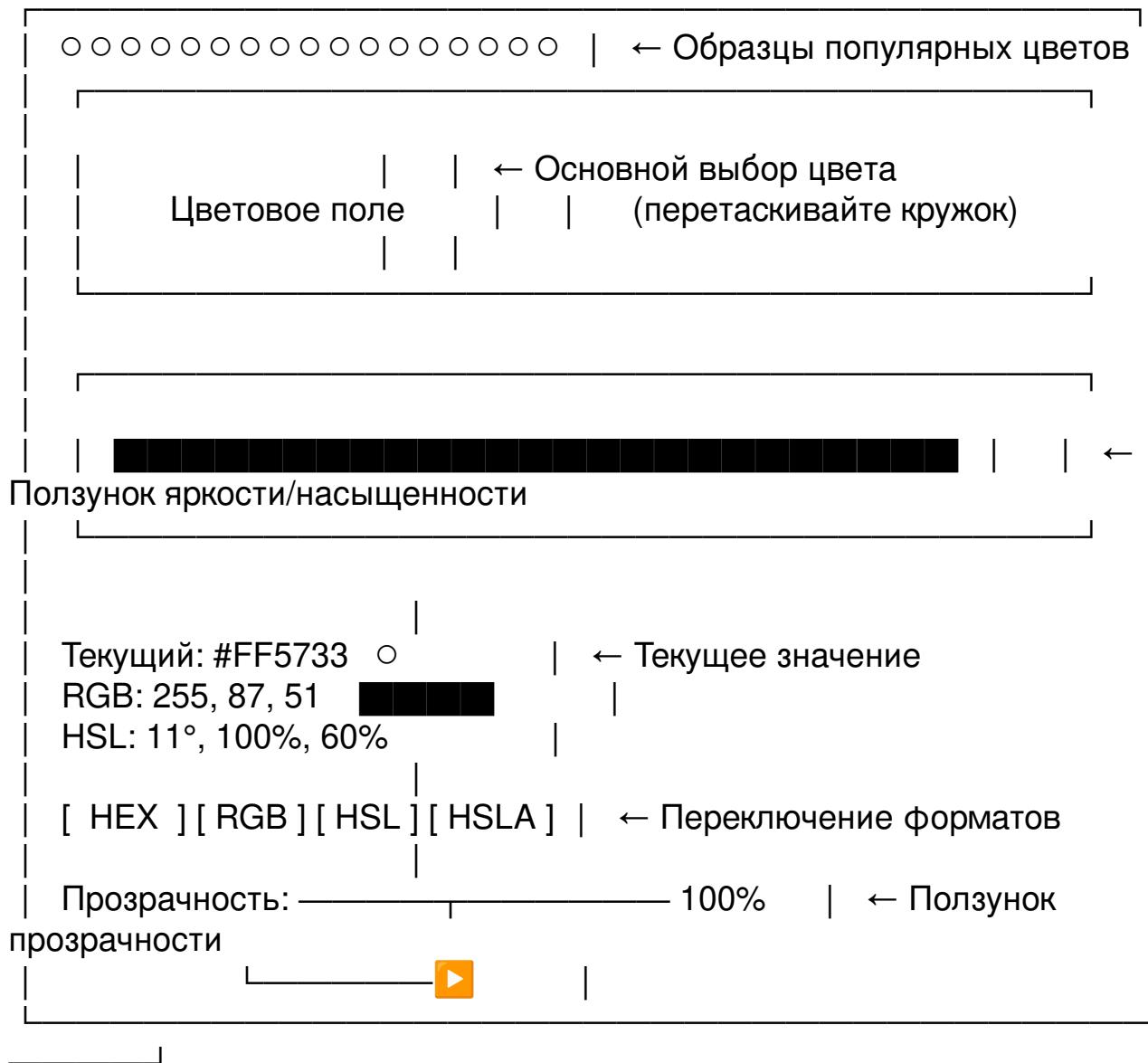
Выберите "Change Color..."

---

## Как работает палитра цветов в VS Code

Когда вы откроете палитру, вы увидите:

text



### Что можно делать в палитре:

**Выбирать цвет мышкой** — просто кликайте в цветовом поле

**Перетаскивать кружок** для тонкой настройки

**Менять яркость/насыщенность** — двигайте вертикальный ползунок

**Копировать значение** — кликните на текущее значение

**Менять формат** — HEX, RGB, HSL, названия цветов

**Настраивать прозрачность** — от 0% (полностью прозрачный) до 100% (непрозрачный)

---

## Разные форматы цветов в CSS

VS Code понимает и поддерживает ВСЕ форматы цветов!

### 1. HEX (шестнадцатеричный) — самый популярный

CSS

```
/* Формат: #RRGGBB */
color: #ff0000; /* Красный */
color: #00ff00; /* Зеленый */
color: #0000ff; /* Синий */
color: #ff5733; /* Оранжевый */

/* Сокращенный формат: #RGB (для пар одинаковых символов) */
color: #f00; /* То же, что #ff0000 */
color: #0f0; /* То же, что #00ff00 */
color: #00f; /* То же, что #0000ff */

/* С прозрачностью: #RRGGBBAA */
color: #ff000080; /* Красный на 50% прозрачности */
```

### 2. RGB и RGBA (Red, Green, Blue)

CSS

```
/* Формат: rgb(red, green, blue) */
color: rgb(255, 0, 0); /* Красный */
color: rgb(0, 255, 0); /* Зеленый */
color: rgb(0, 0, 255); /* Синий */

/* С прозрачностью: rgba(red, green, blue, alpha) */
color: rgba(255, 0, 0, 0.5); /* Красный на 50% прозрачности */
color: rgba(0, 255, 0, 0.3); /* Зеленый на 30% прозрачности */

/* Значения от 0 до 255 для цветов, от 0 до 1 для прозрачности */
```

### 3. HSL и HSLA (Hue, Saturation, Lightness) — очень удобно!

CSS

```
/* Формат: hsl(hue, saturation, lightness) */
color: hsl(0, 100%, 50%); /* Красный */
color: hsl(120, 100%, 50%); /* Зеленый */
color: hsl(240, 100%, 50%); /* Синий */

/* С прозрачностью: hsla(hue, saturation, lightness, alpha) */
color: hsla(0, 100%, 50%, 0.5); /* Полупрозрачный красный */

/* Как запомнить:
Hue (оттенок) - 0-360° (цветовой круг)
Saturation (насыщенность) - 0-100% (0% - серый, 100% - яркий)
Lightness (светлота) - 0-100% (0% - черный, 50% - нормальный, 100% - белый)
*/
```

## 4. Именованные цвета — для простоты

CSS

```
/* Базовые цвета */
color: red;
color: blue;
color: green;
color: yellow;

/* Современные браузеры поддерживают 140+ названий */
color: cornflowerblue; /* Васильковый */
color: lavender; /* Лавандовый */
color: peachpuff; /* Персиковый */
color: goldenrod; /* Золотистый */

/* Системные цвета */
color: currentColor; /* Текущий цвет */
background: transparent; /* Прозрачный */
```

---

### Практическое задание: Создаем цветовую тему для сайта

**Цель:** Используя встроенную палитру, создать гармоничную цветовую схему.

## Шаг 1: Создайте базовый CSS файл

Создайте файл colors.css:

CSS

```
/* ЦВЕТОВАЯ ТЕМА САЙТА */

:root {
 /* Основные цвета - мы их подберем через палитру */
 --primary-color: #3498db; /* Основной цвет */
 --secondary-color: #2ecc71; /* Вторичный цвет */
 --accent-color: #e74c3c; /* Акцентный цвет */

 /* Нейтральные цвета */
 --light-color: #ffffff; /* Светлый */
 --dark-color: #2c3e50; /* Тёмный */
 --gray-color: #95a5a6; /* Серый */

 /* Полупрозрачные цвета */
 --overlay-color: rgba(0, 0, 0, 0.5); /* Затемнение */
 --shadow-color: rgba(0, 0, 0, 0.1); /* Тень */
}

/* ШАПКА САЙТА */
.header {
 background-color: var(--primary-color);
 color: var(--light-color);
 padding: 20px;
}

/* КНОПКИ */
.btn {
 padding: 10px 20px;
 border: none;
 border-radius: 5px;
 cursor: pointer;
 font-size: 16px;
 transition: all 0.3s ease;
}

.btn-primary {
```

```
background-color: var(--primary-color);
color: white;
}

.btn-secondary {
background-color: var(--secondary-color);
color: white;
}

/* КАРТОЧКИ */
.card {
background-color: var(--light-color);
border: 1px solid var(--gray-color);
border-radius: 8px;
padding: 20px;
box-shadow: 0 4px 6px var(--shadow-color);
}

/* ФУТЕР */
.footer {
background-color: var(--dark-color);
color: var(--light-color);
padding: 30px 0;
}
```

## Шаг 2: Используем палитру для выбора цветов

**Откройте палитру для --primary-color:**

Наведите курсор на #3498db

Нажмите на цветной квадратик

В палитре выберите новый цвет (например, сине-фиолетовый)

Обратите внимание, как значение меняется автоматически!

**Попробуйте разные форматы:**

После выбора цвета, нажмите на кнопки [HEX], [RGB], [HSL]

Посмотрите, как одно и то же цвет представлется по-разному

Вернитесь в формат HEX

**Добавьте прозрачность для --shadow-color:**

Кликните на rgba(0, 0, 0, 0.1)

В палитре найдите ползунок "Прозрачность"  
Измените на 0.2 (20%)  
Посмотрите, как изменился код

### Шаг 3: Создайте градиент с помощью подсказок

CSS

```
.gradient-box {
 /* Начните вводить gradient - увидите подсказки! */
 background: linear-gradient(to right, #3498db, #2ecc71);
 /* ↑ нажмите Ctrl+Space для подсказок */
}
```

#### Как использовать подсказки:

Начните вводить `linear-gradient`  
Нажмите `Ctrl + Space` (Windows/Linux) или `Cmd + Space` (Mac)  
Выберите вариант из списка  
VS Code поможет вам с синтаксисом!

### Шаг 4: Быстрое копирование цветов

Наведите курсор на любой цвет  
Появится всплывающее окно  
Кликните на значение цвета (например, `#3498db`)  
Оно скопируется в буфер обмена  
Вставьте в другое место (`Ctrl+V`)

---

## Продвинутые приемы работы с цветами

### 1. Создание цветовых вариантов с помощью HSL

CSS

```
:root {
 --base-hue: 200; /* Синий оттенок */

 --primary: hsl(var(--base-hue), 100%, 50%);
 --primary-light: hsl(var(--base-hue), 100%, 70%);
 --primary-dark: hsl(var(--base-hue), 100%, 30%);

 --secondary: hsl(calc(var(--base-hue) + 60), 100%, 50%);
```

```
/* ↑ добавляем 60° на цветовом круге */
}
```

## 2. Использование CSS переменных для согласованной палитры

css

```
/* Определяем палитру один раз */
:root {
--color-blue: #3498db;
--color-blue-light: #5dade2;
--color-blue-dark: #2e86c1;

--color-green: #2ecc71;
--color-red: #e74c3c;
}

/* Используем везде */
.header { background: var(--color-blue); }
.button { background: var(--color-green); }
.alert { background: var(--color-red); }
```

## 3. Темная тема одним переключением

css

```
/* Светлая тема (по умолчанию) */
:root {
--bg-color: #ffffff;
--text-color: #333333;
--primary-color: #3498db;
}

/* Тёмная тема */
@media (prefers-color-scheme: dark) {
:root {
--bg-color: #1a1a1a;
--text-color: #f0f0f0;
--primary-color: #5dade2;
}
}

body {
background-color: var(--bg-color);
```

```
color: var(--text-color);
}
```

---

## Настройка VS Code для лучшей работы с цветами

### 1. Включение/отключение цветных подсказок

Нажмите `Ctrl + ,` (Windows/Linux) или `Cmd + ,` (Mac)

В поиске введите "Color Decorators"

Найдите `Editor > Color Decorators`

Убедитесь, что галочка включена

### 2. Изменение формата цвета по умолчанию

Откройте настройки (`Ctrl + ,`)

Найдите "Color Formatter"

Выберите предпочтительный формат: `hex`, `rgb`, `hsl`

### 3. Полезные расширения для работы с цветами

**Color Highlight** — подсвечивает цвета прямо в коде

**Color Picker** — добавляет расширенную палитру

**CSS Peek** — показывает цвет при наведении

**GitHub Theme** — красивые цветовые схемы для редактора

---

## Типичные ошибки и их решение

### ✖ Проблема: Цвета не подсвечиваются

#### Решение:

Убедитесь, что файл сохранен с расширением `.css`

Проверьте, что внизу окна VS Code указан язык "CSS"

Перезагрузите VS Code (`Ctrl+Shift+P → "Reload Window"`)

### ✖ Проблема: Палитра не открывается

#### Решение:

Убедитесь, что курсор стоит НА цветовом значении

Попробуйте кликнуть именно на цветной квадратик, а не на текст

Проверьте горячие клавиши в настройках

### ✖ Проблема: Не могу выбрать прозрачность в HEX

**Решение:** HEX с прозрачностью (#RRGGBBAA) поддерживается не во всех браузерах. Используйте RGBA или HSLA:

CSS

```
/* Вместо: */
color: #ff000080; /* Может не работать в старых браузерах */

/* Используйте: */
color: rgba(255, 0, 0, 0.5); /* Работает везде */
```

---

## Советы от профессионалов

**Используйте CSS переменные** для цветов — легко менять всю тему  
**HSL** часто удобнее **HEX** — легче создавать светлые/темные варианты  
**Проверяйте контрастность** — для доступности сайта  
**Создавайте палитру в начале проекта** — избегайте хаоса в цветах  
**Используйте инструменты проверки контраста** (например, WebAIM)  
**Тестируйте цвета на реальных устройствах** — мониторы разные

---

## Ключевые выводы

**Цветные квадратики** — ваш визуальный ориентир в коде  
**Клик по квадратику** или **Ctrl+Shift+C** — открывает палитру  
**HEX (#RRGGBB)** — самый популярный формат  
**RGB/A (rgb(255,0,0))** — понятные числовые значения  
**HSL/A (hsl(0,100%,50%))** — удобен для создания палитр  
**Именованные цвета (red, blue)** — для простых случаев  
**CSS переменные (--primary-color)** — для поддержания порядка  
**Всегда проверяйте контраст** — для доступности вашего сайта

**Помните:** Хорошо подобранная цветовая схема — это половина успеха вашего сайта. С встроенной палитрой VS Code вы становитесь настоящим художником веба!  
Удачи в создании красивых и гармоничных сайтов, дорогой друг! Пусть ваши цвета всегда будут сочными, а контрасты — четкими.



## Пипетка для копирования цветов

### Дорогой друг!

Сегодня мы освоим настоящую магию — **пипетку для копирования цветов** прямо в VS Code! Это как волшебная палочка, которая позволяет "украсть" любой цвет с экрана и использовать его в своём коде. Больше не нужно угадывать коды или искать цвета в интернете — просто берете их из мира вокруг!

### Что такое пипетка в VS Code?

Это инструмент, который позволяет выбрать цвет с **ЛЮБОЙ** части экрана и автоматически вставить его шестнадцатеричный код (HEX) в ваш CSS-файл. Это похоже на пипетку в Photoshop, но прямо в редакторе кода!

---

### Как работает пипетка в VS Code

#### Когда она доступна:

Когда вы редактируете **CSS, SCSS, LESS** файлы  
Когда вы находитесь **внутри тега <style>** в HTML  
Когда курсор находится **на месте для ввода цвета**

#### Как её вызвать:

Есть несколько способов открыть пипетку:

#### Способ 1: При вводе цвета (самый простой!)

```
CSS
.element {
background-color: #| /* Поставьте курсор после решетки # */
}
```

#### Что делать:

Начните вводить цвет в формате HEX

После знака **#** появится **маленькая иконка пипетки**

Кликните по этой иконке — курсор превратится в увеличительное стекло

## Способ 2: Горячие клавиши

Когда курсор находится на месте для цвета:

**Windows:** Ctrl + Shift + C

**Mac:** Cmd + Shift + C

**Linux:** Ctrl + Shift + C

## Способ 3: Через контекстное меню

Кликните правой кнопкой мыши там, где хотите вставить цвет

Выберите "Выбрать цвет" или "Pick Color"

Или найдите "Pick Color" в меню Edit → Color Picker

---

## Пошаговая инструкция: Ваша первая пипетка

Давайте вместе пройдем весь процесс от начала до конца:

### Шаг 1: Подготовьте рабочее пространство

Создайте новый CSS-файл: colors.css

Напишите простой CSS-код:

CSS

```
.my-element {
background-color: #|
}
```

Установите курсор после # (как показано выше)

### Шаг 2: Откройте пипетку

**Либо** нажмите на появившуюся иконку пипетки

**Либо** используйте горячие клавиши Ctrl+Shift+C

### Шаг 3: Выберите цвет с экрана

Курсор превратится в **увеличительное стекло** (🔍)

Под увеличительным стеклом появится **увеличенная область** экрана

В центре будет **перекрестье** для точного выбора

Рядом отображается **выбранный цвет** и его **HEX-код**

## Шаг 4: Захватите цвет

**Наведите** увеличительное стекло на любой цвет на экране:

Другой веб-сайт в браузере

Изображение на рабочем столе

Элемент интерфейса Windows/Mac

Даже цвет из другого окна VS Code!

**Кликните левой кнопкой мыши** — цвет будет захвачен

## Шаг 5: Цвет вставлен!

После клика:

CSS

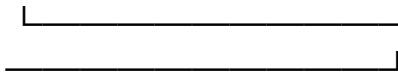
```
.my-element {
background-color: #3a86ff; /* Ваш захваченный цвет появится здесь! */
}
```

## Что видит пипетка: Окно выбора цвета в деталях

Когда пипетка активна, вы видите:

text





## Ключевые элементы:

**Увеличенная область** — увеличение 8x или 16x для точности  
**Перекрестие (x)** — показывает пиксель, который будет захвачен  
**HEX-код** — отображается в реальном времени при движении мышкой  
**Образец цвета** — показывает, как выглядит выбранный цвет  
**Инструкции** — напоминает, как подтвердить или отменить выбор

---

## Где можно "красть" цвета: Практические примеры

### Пример 1: С веб-сайта

Хотите повторить цветовую схему известного сайта?  
Откройте браузер с сайтом, цвета которого вам нравятся  
В VS Code откройте пипетку в CSS-файле  
Перетащите увеличительное стекло в окно браузера  
Выберите понравившийся цвет с сайта  
Цвет автоматически скопируется в ваш код!

### Пример 2: С изображения

У вас есть дизайн-макет или картинка с нужными цветами?  
Откройте изображение в любом просмотрщике  
Используйте пипетку, чтобы захватить цвета с картинки

Создайте палитру для своего сайта:

```
CSS
:root {
--main-color: #ff6b6b; /* С кожи персонажа */
--accent-color: #4ecdc4; /* С фона */
--dark-color: #556270; /* С теней */
}
```

### Пример 3: С рабочего стола

Хотите, чтобы сайт гармонировал с вашей системой?  
Откройте пипетку

Выберите цвет с панели задач Windows/Mac  
Или с любой иконки на рабочем столе  
Или с обоев рабочего стола

### Пример 4: С другого окна VS Code

Удобно для согласования цветов в проекте:  
Откройте два окна VS Code рядом  
В одном — дизайн-макет или пример кода  
В другом — ваш CSS-файл с активной пипеткой  
Захватывайте цвета прямо из соседнего окна!

---

## Практическое задание: Создаем палитру из реального сайта

**Цель:** "Украсть" цветовую схему с известного сайта и применить в своём проекте.

### Шаг 1: Выберите сайт-источник

Откройте в браузере любой понравившийся сайт, например:

[Google.com](#)

[Apple.com](#)

[Behance.net](#)

Или ваш любимый блог

### Шаг 2: Создайте структуру CSS

Создайте файл `stolen-palette.css`:

```
css
/* ПАЛИТРА, "ПОЗАИМСТВОВАННАЯ" У [название сайта] */
/* Дата создания: [сегодняшняя дата] */
```

```
:root {
 /* Основные цвета сайта */
 --primary-color: #| /* Основной брендовый цвет */
 --secondary-color: #| /* Вторичный акцентный цвет */
 --accent-color: #| /* Яркий цвет для привлечения внимания */

 /* Текст */
```

```
--text-primary: #| /* Основной цвет текста */
--text-secondary: #| /* Второстепенный текст */

/* Фоны */
--bg-light: #| /* Светлый фон */
--bg-dark: #| /* Темный фон */
--bg-gray: #| /* Нейтральный серый фон */
}

/* Для примера использования */
.header {
background-color: var(--primary-color);
color: white;
}

.button {
background-color: var(--accent-color);
color: white;
border: none;
padding: 10px 20px;
}
```

### Шаг 3: Начинаем "кражу" цветов!

#### Основной брендовый цвет (--primary-color):

Установите курсор после # у --primary-color  
Откройте пипетку (клик на иконку или Ctrl+Shift+C)  
Перетащите увеличительное стекло в браузер  
Найдите логотип или основной элемент дизайна сайта  
Кликните по нему — цвет скопируется!

#### Цвет текста (--text-primary):

Повторите процесс для обычного текста на сайте  
Обычно это темно-серый или почти черный цвет

#### Акцентный цвет (--accent-color):

Найдите кнопки, ссылки или выделенные элементы  
Часто это более яркий или контрастный цвет

## Фоновые цвета (--bg-light, --bg-dark):

Выберите цвета с разных областей сайта  
Светлые области для --bg-light  
Темные области (подвал, боковые панели) для --bg-dark

## Шаг 4: Проанализируйте результат

После заполнения всех цветов, у вас получится что-то вроде:

css

```
:root {
 /* ПАЛИТРА С GOOGLE.COM */
 --primary-color: #4285f4; /* Синий из логотипа Google */
 --secondary-color: #34a853; /* Зеленый из логотипа */
 --accent-color: #ea4335; /* Красный из логотипа */

 --text-primary: #3c4043; /* Текст на главной странице */
 --text-secondary: #5f6368; /* Второстепенный текст */

 --bg-light: #ffffff; /* Основной фон */
 --bg-dark: #f8f9fa; /* Фон поисковой строки */
 --bg-gray: #dadce0; /* Цвет рамок */
}
```

## Шаг 5: Создайте тестовую страницу

Создайте HTML-файл и примените вашу "украденную" палитру:

```
html
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="stolen-palette.css">
<style>
body {
 font-family: Arial, sans-serif;
 background-color: var(--bg-light);
 color: var(--text-primary);
 margin: 0;
 padding: 20px;
}
```

```
.header {
background-color: var(--primary-color);
color: white;
padding: 20px;
border-radius: 8px;
margin-bottom: 20px;
}

.button {
background-color: var(--accent-color);
color: white;
border: none;
padding: 10px 20px;
border-radius: 4px;
cursor: pointer;
font-size: 16px;
}

.card {
background-color: var(--bg-light);
border: 1px solid var(--bg-gray);
border-radius: 8px;
padding: 20px;
margin-bottom: 20px;
}
</style>
</head>
<body>
<div class="header">
<h1>Мой сайт с "украденной" палитрой</h1>
</div>

<div class="card">
<h2>Пример текста</h2>
<p>Это основной текст с цветом --text-primary</p>
<p style="color: var(--text-secondary)">А это второстепенный текст</p>
<button class="button">Кнопка с акцентным цветом</button>
</div>
</body>
</html>
```

---

## Расширенные возможности и настройки

### 1. Изменение формата цвета

По умолчанию пипетка вставляет цвета в формате HEX (#RRGGBB). Но можно изменить:

Откройте настройки VS Code: Ctrl + , (или Cmd + , на Mac)

Найдите "Color Picker Format"

Выберите предпочтительный формат:

hex — #RRGGBB (по умолчанию)

rgb — rgb(R, G, B)

hsl — hsl(H, S, L)

auto — автоматический выбор

### 2. Использование с другими форматами

Пипетка работает даже если вы не начинаете с #:

css

```
.element {
color: rgb(); /* Поставьте курсор внутри скобок */
border-color: hsl(); /* И здесь тоже сработает! */
}
```

### 3. Полезные расширения для работы с цветами

Установите через Marketplace (Ctrl+Shift+X):

**Color Picker** — расширенная функциональность пипетки

**Color Highlight** — подсвечивает цвета прямо в коде

**Color Manager** — управление палитрами и цветовыми схемами

**Image Preview** — показывает цвета из изображений

### 4. Горячие клавиши для быстрого доступа

Настройте свои сочетания клавиш:

Ctrl+Shift+P → "Preferences: Open Keyboard Shortcuts"

Введите "Pick Color"

Нажмите "+" для добавления новой комбинации

Например, назначьте Alt+C для быстрого вызова пипетки

---

## Типичные проблемы и их решения

## **Проблема 1: Пипетка не появляется**

**Причины и решения:**

**Не тот тип файла** — пипетка работает только в CSS, SCSS, LESS и внутри `<style>` в HTML

**Не то место** — курсор должен быть на месте для ввода цвета

**Расширение отключено** — проверьте, что CSS-поддержка активна

**Перезагрузите VS Code** — иногда помогает простой перезапуск

## **Проблема 2: Не могу выбрать цвет вне VS Code**

**Решение:**

Убедитесь, что окно VS Code не развернуто на весь экран

Оставьте небольшой участок экрана видимым

Или сверните VS Code, но оставьте курсор на месте цвета

## **Проблема 3: Цвета выглядят по-разному**

**Причина:** Разные мониторы, настройки цветопередачи, освещение

**Решение:**

Используйте пипетку на том же мониторе, где будет смотреться сайт

Проверяйте цвета на нескольких устройствах

Используйте инструменты проверки контраста (WebAIM)

## **Проблема 4: Пипетка захватывает не тот пиксель**

**Решение:**

Используйте увеличение (обычно 16x)

Двигайте мышку медленнее

Убедитесь, что перекрестие точно на нужном пикселе

---

## **Профессиональные советы по работе с пипеткой**

### **Совет 1: Создавайте библиотеки цветов**

CSS

```
/* colors-library.css */
:root {
/* "Украдено" с Apple.com */
--apple-blue: #007aff;
--apple-gray: #8e8e93;
--apple-green: #34c759;
```

```
/* "Украдено" с Google.com */
```

```
--google-blue: #4285f4;
```

```
--google-red: #ea4335;
```

```
--google-yellow: #fb9bc05;
```

```
/* "Украдено" с Twitter.com */
```

```
--twitter-blue: #1da1f2;
```

```
--twitter-dark: #14171a;
```

```
}
```

## Совет 2: Захватывайте градиенты

Захватите начальный цвет градиента

Захватите конечный цвет градиента

Создайте градиент в CSS:

css

```
.background {
```

```
background: linear-gradient(to right, #ff6b6b, #4ecdc4);
```

```
}
```

## Совет 3: Используйте для цветовой гармонии

Захватите основной цвет с изображения

Используйте цветовые круги (адоб) для подбора гармоничных цветов

Или используйте сайты типа [Coolors.co](#) для генерации палитр

## Совет 4: Документируйте источники

css

```
/*
```

*ЦВЕТОВАЯ СХЕМА: Весенний сад*

*Источник: фотография сада (сайт Unsplash.com)*

*Дата захвата: 15.05.2023*

*Цвета:*

1. #8ac926 - Зелень листьев (основной)

2. #1982c4 - Голубое небо (акцент)

3. #ffca3a - Солнечный свет (подсветка)

```
*/
```

## Совет 5: Проверяйте доступность

После захвата цвета проверьте:

**Контрастность** — для читаемости текста

**Цветовую слепоту** — как видят люди с дальтонизмом

**Сочетаемость** — не режут ли глаза комбинации цветов

---

## Альтернативные инструменты для захвата цветов

Если пипетка VS Code не работает или нужны дополнительные функции:

### 1. Пипетка в браузере

**Chrome/Firefox:** Инструменты разработчика (F12) → пипетка в панели цветов

**Edge:** Встроенная пипетка в меню "..." → "Захват цвета"

### 2. Сторонние программы

**ShareX** (бесплатно) — мощный инструмент для захвата всего

**ColorCop** (Windows) — простая пипетка с историей цветов

**Digital Color Meter** (Mac) — встроенная утилита в macOS

**Gpick** (Linux) — расширенный выборщик цветов

### 3. Онлайн-инструменты

**ColorZilla** — расширение для браузера

**Colors** — генератор и захват цветов

**Adobe Color** — профессиональный инструмент для палитр

---

## Ключевые выводы

**Пипетка в VS Code** — встроенный инструмент для захвата цветов с экрана

**Вызывается:** иконкой при вводе цвета или `Ctrl+Shift+C`

**Работает:** в CSS, SCSS, LESS и внутри `<style>` в HTML

**Форматы:** HEX (по умолчанию), можно настроить RGB/HSL

**Источники:** любые окна, браузеры, изображения на экране

**Практическое применение:** создание палитр, копирование дизайнов, цветовая гармония

**Важно:** проверять контрастность и доступность цветов

**Помните:** Пипетка — это мощный инструмент для вдохновения и обучения. "Задумайтесь" цвета у лучших, анализируйте, почему они работают, и создавайте свои уникальные сочетания. Но всегда уважайте авторские права и используйте цвета этично!

Теперь вы настоящий художник-колорист в мире веб-разработки! Удачи в создании красивых и гармоничных сайтов, дорогой друг!

## Палитра цветов с предпросмотром

### Дорогой друг!

Сегодня мы погрузимся в мир цветов с профессиональной **палитрой с предпросмотром** в VS Code. Это как иметь художественную мастерскую прямо в редакторе кода — вы не только выбираете цвета, но и сразу видите, как они будут выглядеть в вашем дизайне!

### Что такое палитра с предпросмотром?

Это интеллектуальный инструмент, который показывает вам:

-  **Увеличенный просмотр** — цвета под лупой
-  **Готовые палитры** — гармоничные сочетания цветов
-  **Разные форматы** — HEX, RGB, HSL, названия
-  **Предпросмотр в реальном времени** — как цвет будет смотреться
-  **Историю цветов** — ваши недавние выборы

---

### Как открыть и использовать палитру с предпросмотром

#### Базовые способы доступа

##### Способ 1: Через контекстное меню (самый наглядный)

CSS

```
.button {
background-color: #3498db; /* Кликните ПРАВОЙ кнопкой мыши на цвет */
}
```

#### Что делать:

Найдите любой цвет в вашем CSS-коде

**Кликните правой кнопкой мыши** по цветному значению

Выберите "Выбрать цвет" или "Preview Color"

Откроется полноценная палитра с предпросмотром!

##### Способ 2: Через цветной индикатор

CSS

```
.header {
```

```
color: #e74c3c; /* Наведите курсор на цветной квадратик */
}
```

Найдите цветной квадратик слева от цветового значения

**Наведите курсор** на этот квадратик

Появится всплывающее окно с предпросмотром

**Кликните левой кнопкой** для открытия полной палитры

### Способ 3: Горячие клавиши

Когда курсор находится на цветовом значении:

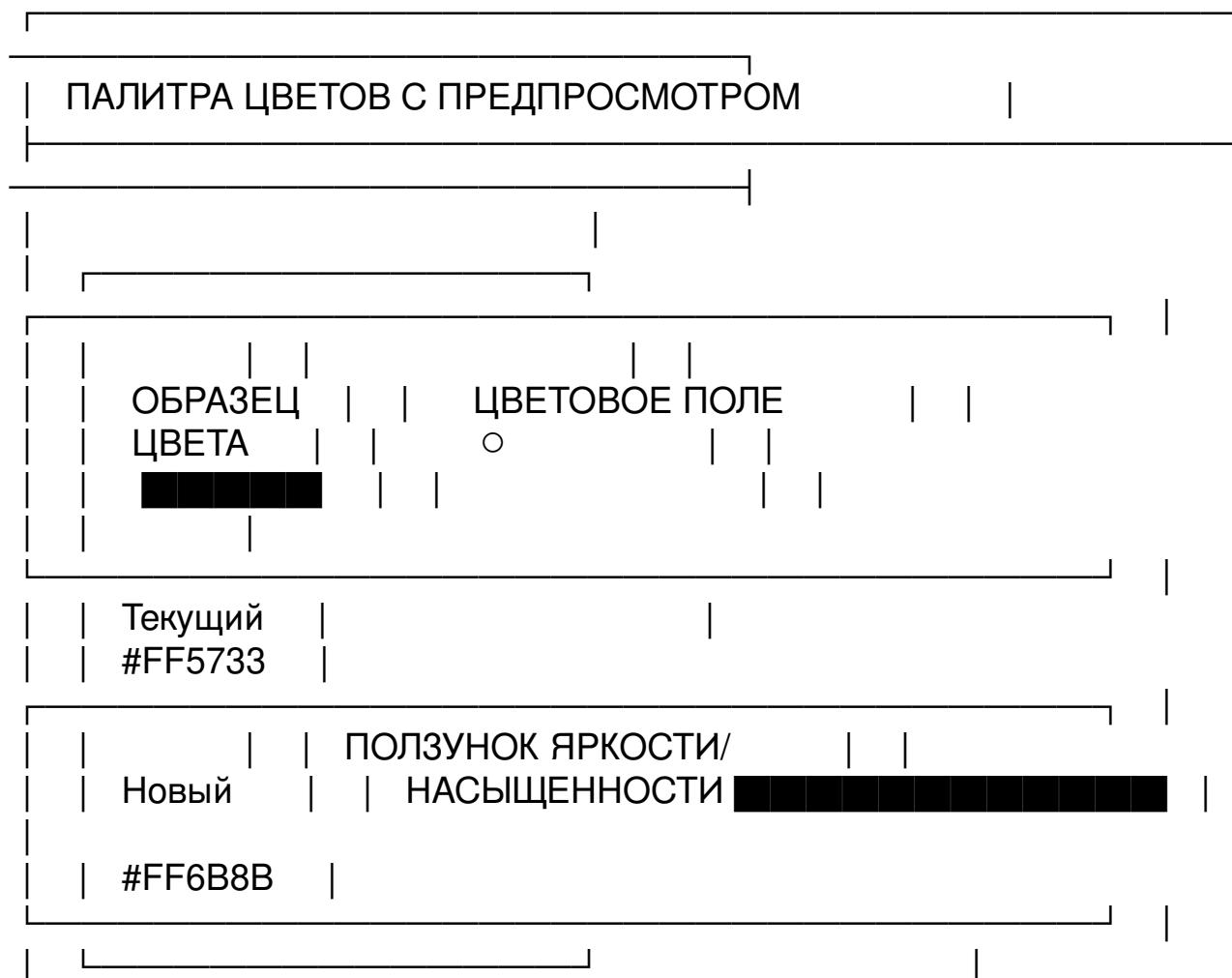
**Windows/Linux:** Ctrl + Shift + C

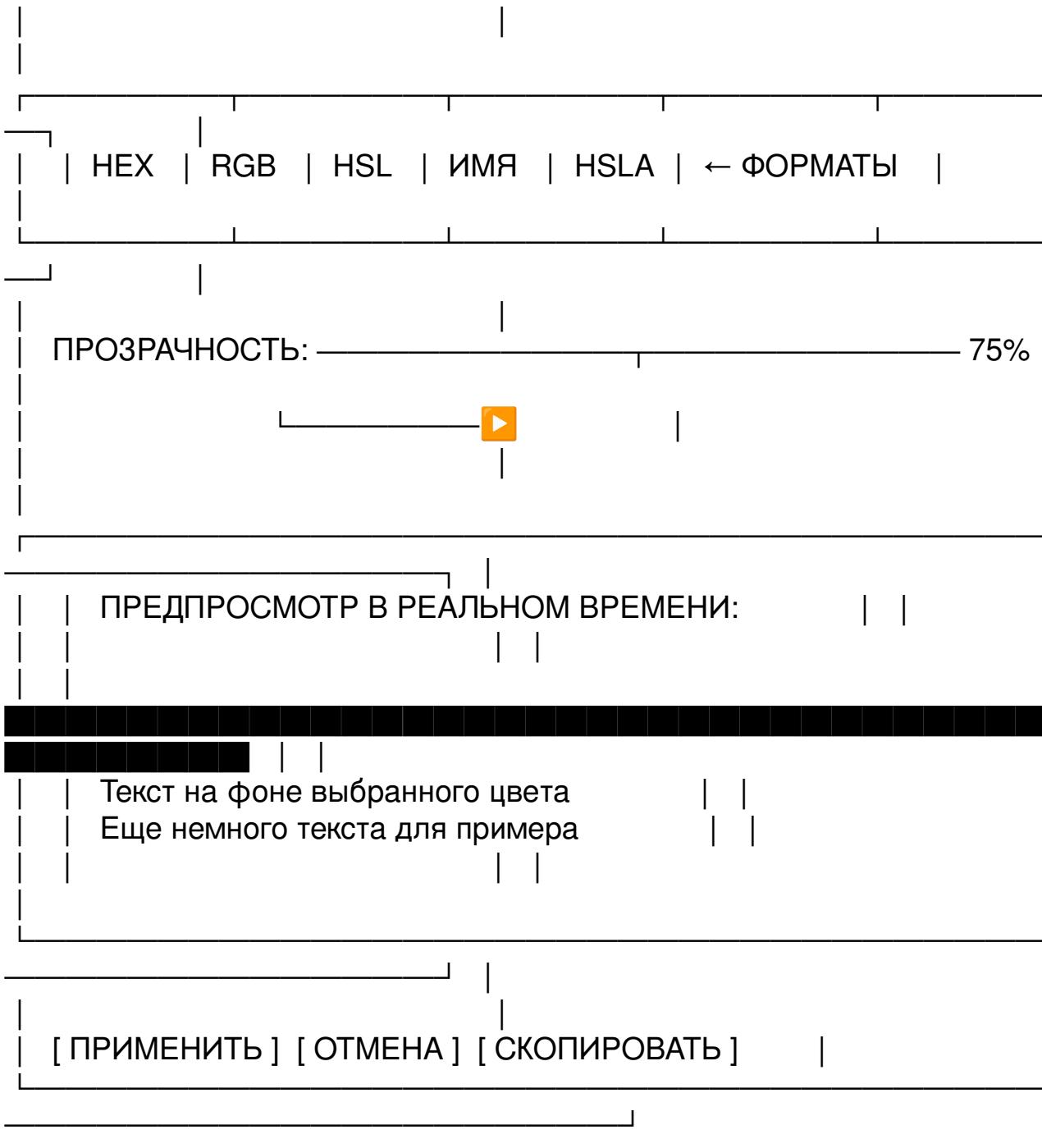
**Mac:** Cmd + Shift + C

### Детальный обзор интерфейса палитры

Когда вы откроете палитру, увидите **полноценный цветовой редактор**:

text





## Ключевые функции палитры с предпросмотром

### 1. Цветовое поле с лупой (главный инструмент)

**Как работает:**

**Большое поле** — спектр всех возможных цветов

**Белый кружок** — перетаскивайте его для выбора цвета

**Увеличение при наведении** — под кружком появляется лупа 8-16x

**Точный выбор** — видите отдельные пиксели

## Практический пример:

CSS

```
/* Хотите подобрать идеальный синий для кнопки */
.btn-primary {
background-color: #| /* Откройте палитру здесь */
}
```

Откройте палитру

Перетащите кружок в синюю область

Используйте вертикальный ползунок для регулировки яркости

Смотрите предпросмотр в реальном времени

## 2. Образец цвета с сравнением

**Что видите:**

**Текущий цвет (сверху)** — который был в коде

**Новый цвет (снизу)** — который выбираете сейчас

**Визуальное сравнение** — сразу видно разницу

**Цветовые коды** — оба значения отображаются

CSS

```
/* Было: темно-синий */
.header { background: #2c3e50; }

/* В палитре выбираем светлее: */
/* Текущий: #2c3e50 */
/* Новый: #3498db ← видим разницу сразу! */
```

## 3. Предпросмотр в реальном времени (самая полезная функция!)

**Что это дает:**

Видите, как цвет будет смотреться **с текстом**

Оцениваете **читаемость и контрастность**

Тестируете **сочетания** цветов

**Пример использования:**

CSS

```
/* Выбираем цвет фона для текстового блока */
.article {
```

```
background-color: #| /* Открываем палитру */
color: #333; /* Цвет текста остается темным */
}
```

В палитре вы увидите:

text

ПРЕДПРОСМОТР:



Это текст на выбранном фоне. Проверьте, насколько хорошо он читается!



## 4. Переключение форматов цвета

Доступные форматы:

**HEX** — #FF5733 (шестнадцатеричный, самый популярный)

**RGB** — rgb(255, 87, 51) (красный, зеленый, синий)

**HSL** — hsl(11, 100%, 60%) (оттенок, насыщенность, светлота)

**Имя цвета** — tomato (если у цвета есть название)

**HSLA** — hsla(11, 100%, 60%, 0.8) (HSL с прозрачностью)

Почему это важно:

CSS

/\* HEX - компактно \*/

color: #ff5733;

/\* RGB - понятные числа \*/

color: rgb(255, 87, 51);

/\* HSL - легко создавать вариации \*/

color: hsl(11, 100%, 60%);

/\* Для более светлого варианта: \*/

color: hsl(11, 100%, 70%);

/\* Для более темного: \*/

color: hsl(11, 100%, 40%);

## 5. Регулятор прозрачности (альфа-канал)

### Как использовать:

**Ползунок** — от 0% (полная прозрачность) до 100% (непрозрачность)

**Предпросмотр** — видите эффект прозрачности сразу

**Автоматическое переключение** — при изменении прозрачности формат меняется на RGBA/HSLA

css

```
/* Выбираем полупрозрачный оверлей */
.overlay {
background-color: rgba(0, 0, 0, 0.5); /* 50% прозрачности */
}
```

### В палитре:

Выберите черный цвет (#000000)

Перетащите ползунок прозрачности на 50%

VS Code автоматически изменит формат на `rgba(0, 0, 0, 0.5)`

---

## Практическое задание: Создаем профессиональную цветовую схему

**Цель:** Используя палитру с предпросмотром, создать полную цветовую схему для блога.

### Шаг 1: Подготовительная работа

Создайте файл `blog-colors.css`:

css

```
/* ЦВЕТОВАЯ СХЕМА БЛОГА "МИР ПУТЕШЕСТВИЙ" */

:root {
/* 1. Основной брендовый цвет */
--brand-primary: #3498db;

/* 2. Цвета текста */
--text-primary: #2c3e50;
--text-secondary: #7f8c8d;
```

```
--text-inverse: #ffffff;

/* 3. Фоновые цвета */
--bg-light: #ffffff;
--bg-dark: #f8f9fa;
--bg-brand: #| /* Выберем через палитру */

/* 4. Акцентные цвета */
--accent-success: #27ae60;
--accent-warning: #f39c12;
--accent-error: #e74c3c;

/* 5. Границы и разделители */
--border-light: #ecf0f1;
--border-dark: #bdc3c7;
}
```

## Шаг 2: Выбираем основной цвет через палитру

**Найдите** --brand-primary: #3498db;  
**Кликните правой кнопкой** на #3498db  
**Выберите "Выбрать цвет"**  
**В палитре сделайте:**  
Посмотрите на **текущий цвет** (синий #3498db)  
Перетащите кружок в цветовом поле  
Попробуйте разные оттенки синего  
**Следите за предпросмотром** — как текст выглядит на этом фоне  
Когда найдете подходящий — нажмите "**Применить**"

## Шаг 3: Создаем гармоничную палитру

**Техника: используем HSL формат для создания вариаций**  
**Основной цвет уже выбрали**  
**Для текста** нужен контрастный цвет:

css

```
--text-primary: #| /* Откройте палитру здесь */
В палитре:
Выберите очень темный цвет (близкий к черному)
В предпросмотре убедитесь, что текст читается
Попробуйте не чисто черный (#000), а темно-серый (#2c3e50)
Для фона нужен светлый, но не белый:
```

CSS

`-bg-light: #| /* Откройте палитру */`

### В палитре:

Выберите почти белый цвет

Используйте очень светлый оттенок вашего основного цвета

Например, если основной синий, сделайте очень светлый синий

## Шаг 4: Тестируем сочетания цветов

Создайте блок для тестирования прямо в CSS:

CSS

`/* ТЕСТОВЫЙ БЛОК ДЛЯ ПРЕДПРОСМОТРА */`

```
.color-test {
padding: 20px;
border-radius: 8px;
margin: 20px 0;
}
```

`/* Тест основного цвета */`

```
.test-primary {
background-color: var(--brand-primary);
color: var(--text-inverse);
}
```

`/* Тест фона с текстом */`

```
.test-bg-light {
background-color: var(--bg-light);
color: var(--text-primary);
border: 1px solid var(--border-light);
}
```

`/* Тест акцентного цвета */`

```
.test-accent {
background-color: var(--accent-success);
color: var(--text-inverse);
}
```

## Шаг 5: Используем историю цветов

**Важная фича:** VS Code запоминает ваши последние выборы!

После работы с палитрой закройте ее

Снова откройте палитру в любом месте  
**Наведите курсор** на историю (обычно внизу палитры)  
Вы увидите **последние 5-10 цветов**, которые выбирали  
Кликните на любой, чтобы быстро его применить

CSS

```
/* Быстрый доступ к недавним цветам */
.recent-color-1 { color: /* из истории */; }
.recent-color-2 { background: /* из истории */; }
```

---

## Расширенные возможности и настройки

### 1. Настройка формата по умолчанию

**Чтобы всегда получать нужный формат:**

Откройте настройки: **Ctrl + ,** (Win/Linux) или **Cmd + ,** (Mac)  
Найдите "Color Picker Format"

Выберите:

hex — #RRGGBB  
rgb — rgb()  
hsl — hsl()  
auto — автоматический выбор

### 2. Палитра с цветовым кругом (расширение)

**Установите расширение "Color Picker":**

**Ctrl + Shift + X → Marketplace**  
Найдите "Color Picker"  
Установите его  
Теперь в палитре появится **цветовой круг** (hue circle)

**Преимущества цветового круга:**

Видно **гармоничные сочетания** (комплементарные, триады)  
Легче подбирать цвета по правилам колористики  
Профессиональный интерфейс

### 3. Сохранение пользовательских палитр

## Создайте файл с часто используемыми цветами:

```
CSS
/* my-palettes.css */
:root {
 /* Природная палитра */
 --nature-green: #27ae60;
 --nature-blue: #3498db;
 --nature-brown: #8b4513;
 --nature-sand: #f1c40f;

 /* Техно палитра */
 --tech-dark: #2c3e50;
 --tech-blue: #2980b9;
 --tech-purple: #8e44ad;
 --tech-green: #16a085;

 /* Пастельная палитра */
 --pastel-pink: #ffb6c1;
 --pastel-blue: #add8e6;
 --pastel-green: #98fb98;
 --pastel-yellow: #ffffacd;
}
```

## Импортируйте в проекты:

```
CSS
@import url('my-palettes.css');

.element {
 background-color: var(--nature-green);
}
```

## 4. Интеграция с расширением "Color Highlight"

### Что дает:

Устанавливаете "Color Highlight"  
Теперь цвета в коде подсвечиваются фоном  
**Двойной клик** по цвету открывает палитру  
Еще лучше видно, как цвета выглядят

CSS

```
/* После установки Color Highlight */
.colors {
 color: #FF5733; /* Подсвечивается оранжевым */
 background: #3498db; /* Подсвечивается синим */
 border: 2px solid #2ECC71; /* Подсвечивается зеленым */
}
```

## 5. Горячие клавиши для профессионалов

### Настройте быстрые клавиши:

Ctrl+Shift+P → "Preferences: Open Keyboard Shortcuts"

Найдите "colorPicker"

Добавьте свои сочетания:

```
json
{
 "key": "ctrl+alt+c",
 "command": "editor.action.colorPicker"
}
```

---

## Профессиональные техники работы с палитрой

### Техника 1: Создание градиентов с предпросмотром

CSS

```
.gradient-box {
 background: linear-gradient(
 to right,
 #|, /* Откройте палитру для первого цвета */
 #| /* И для второго цвета */
);
}
```

### Как делать:

Выберите первый цвет через палитру

Запомните его HSL значения

Для второго цвета измените только Lightness (светлоту)

Или Hue (оттенок) на 30-60 градусов для гармонии

## Техника 2: Проверка контрастности WCAG

### Ручная проверка в палитре:

Выберите цвет фона

В предпросмотре посмотрите на черный текст

Затем выберите цвет текста

В предпросмотре посмотрите на белый фон

**Эмпирическое правило:** если плохо видно — нужен больше контраст

### Формула для проверки (примерно):

Хороший контраст: разница яркости > 4.5:1

В палитре: темный текст на светлом фоне всегда читается лучше

## Техника 3: Создание монохроматической палитры

### Используйте HSL для создания оттенков одного цвета:

CSS

```
:root {
 --base-hue: 200; /* Синий тон */

 /* Основной цвет */
 --color-primary: hsl(var(--base-hue), 100%, 50%);

 /* Более светлые варианты */
 --color-light-1: hsl(var(--base-hue), 100%, 70%);
 --color-light-2: hsl(var(--base-hue), 100%, 85%);
 --color-light-3: hsl(var(--base-hue), 100%, 95%);

 /* Более темные варианты */
 --color-dark-1: hsl(var(--base-hue), 100%, 30%);
 --color-dark-2: hsl(var(--base-hue), 100%, 20%);
 --color-dark-3: hsl(var(--base-hue), 100%, 10%);
}
```

## Техника 4: Адаптация цветов для темной темы

CSS

```
/* Светлая тема (по умолчанию) */
:root {
 --bg-primary: #ffffff;
 --text-primary: #000000;
```

```
}

/* Темная тема */
@media (prefers-color-scheme: dark) {
:root {
--bg-primary: #121212; /* Не чисто черный! */
--text-primary: #e0e0e0; /* Не чисто белый! */
}
}
```

/\* Используйте палитру для подбора:  
- В светлой теме: dark text on light background  
- В темной теме: light text on dark background  
\*/

---

## Типичные проблемы и решения

### ✗ Проблема: Цвета в палитре и в браузере разные

#### Причины:

Разные цветовые профили мониторов

Разные настройки яркости/контраста

Разное освещение

#### Решение:

Калибруйте монитор (хотя бы программно)

Проверяйте цвета на нескольких устройствах

Используйте стандартные, "безопасные" цвета

### ✗ Проблема: Предпросмотр не показывает реальный вид

#### Решение:

Используйте расширение **Live Server**

Открывайте страницу в браузере в реальном времени

Делайте изменения и смотрите результат сразу

### ✗ Проблема: Не могу подобрать гармоничные цвета

#### Решение:

Используйте **правило 60-30-10**:

60% основного цвета

30% дополнительного

10% акцентного

Или установите расширение "**Color Manager**" с готовыми палитрами

## ✖ Проблема: Забываю, какие цвета использовал

**Решение:**

Ведите **цветовую документацию** в комментариях

Используйте **осмыслиенные имена переменных**

Создайте файл colors-guide.md с примерами

---

## Полезные расширения для работы с цветами

1. **Color Highlight** — подсветка цветов в коде
  2. **Color Picker** — расширенная палитра с цветовым кругом
  3. **Color Manager** — управление палитрами и схемами
  4. **CSS Peek** — быстрый просмотр цветов
  5. **Auto Rename Tag** — помогает при работе с CSS переменными
- 

## Ключевые выводы

**Палитра с предпросмотром** — ваш главный инструмент для работы с цветами

**Открывается:** правой кнопкой на цвете или **Ctrl+Shift+C**

**Основные функции:** цветовое поле, сравнение, предпросмотр, форматы, прозрачность

**Предпросмотр в реальном времени** — самая ценная функция, тестируйте читаемость!

**HSL формат** — лучший для создания гармоничных палитр

**История цветов** — быстрый доступ к недавним выборам

**Всегда проверяйте контрастность** — для доступности вашего сайта

**Золотое правило:** Хорошая цветовая схема — это не просто красивые цвета, а **функциональные** цвета, которые решают задачи дизайна: направляют внимание, организуют информацию, создают настроение. Теперь вы вооружены профессиональным инструментом для создания потрясающих цветовых решений! Удачи в творчестве, дорогой друг!

## Приложения

### Приложение А: Шпаргалка по горячим клавишам

#### Дорогой друг!

Эта шпаргалка станет вашим надежным спутником в работе с VS Code. Сохраните её, распечатайте или держите всегда под рукой. Постепенно эти сочетания клавиш станут вашей второй натурой и ускорят работу в разы!

#### Содержание

[Базовые операции](#)

[Навигация по коду](#)

[Редактирование текста](#)

[Работа с файлами](#)

[Поиск и замена](#)

[Мультикурсор и выделение](#)

[Форматирование кода](#)

[Работа с окнами](#)

[Терминал](#)

[Отладка](#)

#### [Базовые операции](#)

Действие	Windows/ Linux	Mac	Когда использовать
Команда палитры	Ctrl + Shift + P	Cmd + Shift + P	Главное меню, любая команда
Быстрый поиск файла	Ctrl + P	Cmd + P	Переход к любому файлу в проекте
Поиск в проекте	Ctrl + Shift + F	Cmd + Shift + F	Поиск по всем файлам
Настройки	Ctrl + ,	Cmd + ,	Открыть настройки

Действие	Windows/ Linux	Mac	Когда использовать
<b>Показать/скрыть боковую панель</b>	Ctrl + B	Cmd + B	Больше места для кода
<b>Показать/скрыть терминал</b>	Ctrl + J	Ctrl + J	Работа с командной строкой
<b>Показать/скрыть панель проблем</b>	Ctrl + Shift + M	Cmd + Shift + M	Ошибки и предупреждения
<b>Переключить комментарий</b>	Ctrl + /	Cmd + /	Закомментировать/раскомментировать строку
<b>Повторить действие</b>	Ctrl + Y	Cmd + Y или Shift + Cmd + Z	Повторить отмененное
<b>Копировать строку</b>	Shift + Alt + ↓	Shift + Option + ↓	Дублировать строку вниз
	Shift + Alt + ↑	Shift + Option + ↑	Дублировать строку вверх
<b>Удалить строку</b>	Ctrl + Shift + K	Cmd + Shift + K	Удалить текущую строку
<b>Переместить строку</b>	Alt + ↓	Option + ↓	Переместить строку вниз
	Alt + ↑	Option + ↑	Переместить строку вверх

## 🧭 Навигация по коду

Действие	Windows/ Linux	Mac	Когда использовать
<b>Перейти к строке</b>	Ctrl + G	Ctrl + G	Переход к конкретной строке
<b>Перейти к определению</b>	F12	F12	Перейти к определению функции/класса
<b>Показать определение</b>	Alt + F12	Option + F12	Показать определение во всплывающем окне

Действие	Windows/ Linux	Mac	Когда использовать
<b>Вернуться назад</b>	Alt + ←	Ctrl + -	Назад по истории навигации
<b>Перейти вперед</b>	Alt + →	Ctrl + Shift + -	Вперед по истории навигации
<b>Перейти к скобке</b>	Ctrl + Shift + \	Cmd + Shift + \	Перейти к парной скобке
<b>Свернуть/развернуть блок</b>	Ctrl + Shift + [	Cmd + Option + [	Свернуть текущий блок
	Ctrl + Shift + ]	Cmd + Option + ]	Развернуть текущий блок
<b>Свернуть все регионы</b>	Ctrl + K, Ctrl + 0	Cmd + K, Cmd + 0	Свернуть весь код
<b>Развернуть все регионы</b>	Ctrl + K, Ctrl + J	Cmd + K, Cmd + J	Развернуть весь код
<b>Перейти к символу в файле</b>	Ctrl + Shift + O	Cmd + Shift + O	Поиск функций/классов в файле
<b>Перейти к символу в проекте</b>	Ctrl + T	Cmd + T	Поиск по всем файлам проекта

## Редактирование текста

Действие	Windows/ Linux	Mac	Когда использовать
<b>Выделить все вхождения</b>	Ctrl + D	Cmd + D	Выделить следующее вхождение слова
<b>Выделить все вхождения сразу</b>	Ctrl + Shift + L	Cmd + Shift + L	Выделить все вхождения слова
<b>Добавить курсор выше</b>	Ctrl + Alt + ↑	Cmd + Option + ↑	Несколько курсоров
<b>Добавить курсор ниже</b>	Ctrl + Alt + ↓	Cmd + Option + ↓	Несколько курсоров
<b>Добавить курсор в конец строк</b>	Shift + Alt + I	Shift + Option + I	Курсор в конце каждой выделенной строки

Действие	Windows/ Linux	Mac	Когда использовать
Обернуть выделенное	Ctrl + Shift + A	Cmd + Shift + A	Обернуть выделенный текст в тег
Форматировать выделенное	Ctrl + K, Ctrl + F	Cmd + K, Cmd + F	Форматировать выделенный фрагмент
Форматировать документ	Shift + Alt + F	Shift + Option + F	Форматировать весь файл
Переключить регистр	Ctrl + Shift + U	Cmd + Shift + U	Поменять регистр выделенного текста
Выделить текущую строку	Ctrl + L	Cmd + L	Выделить всю строку
Выделить текущее слово	Ctrl + D (один раз)	Cmd + D (один раз)	Выделить слово под курсором
Увеличить/ уменьшить отступ	Tab / Shift + Tab	Tab / Shift + Tab	Отступы в коде

## 📁 Работа с файлами

Действие	Windows/Linux	Mac	Когда использовать
Новый файл	Ctrl + N	Cmd + N	Создать пустой файл
Открыть файл	Ctrl + O	Cmd + O	Открыть существующий файл
Сохранить файл	Ctrl + S	Cmd + S	Сохранить текущий файл
Сохранить все файлы	Ctrl + K, S	Cmd + K, S	Сохранить все открытые файлы
Сохранить как...	Ctrl + Shift + S	Cmd + Shift + S	Сохранить под новым именем
Закрыть редактор	Ctrl + W	Cmd + W	Закрыть текущую вкладку
Закрыть все	Ctrl + K, Ctrl + W	Cmd + K, Cmd + W	Закрыть все

Действие	Windows/Linux	Mac	Когда использовать
<b>редакторы</b>		W	вкладки
<b>Перейти к следующему редактору</b>	Ctrl + PgDown	Cmd + Option + →	Следующая вкладка
<b>Перейти к предыдущему редактору</b>	Ctrl + PgUp	Cmd + Option + ←	Предыдущая вкладка
<b>Разделить редактор</b>	Ctrl + \	Cmd + \	Разделить окно редактора
<b>Новая папка</b>	В Explorer: Ctrl + Shift + E, затем N	В Explorer: Cmd + Shift + E, затем N	Создать папку в проекте

## 🔍 Поиск и замена

Действие	Windows/Linux	Mac	Когда использовать
<b>Поиск в файле</b>	Ctrl + F	Cmd + F	Поиск в текущем файле
<b>Замена в файле</b>	Ctrl + H	Cmd + Option + F	Замена текста в файле
<b>Поиск следующего</b>	F3	Cmd + G	Следующее совпадение
<b>Поиск предыдущего</b>	Shift + F3	Cmd + Shift + G	Предыдущее совпадение
<b>Быстрый поиск по символам</b>	Ctrl + ;	Cmd + ;	Поиск при наборе текста
<b>Перейти к файлу по названию</b>	Ctrl + P, затем ввести >	Cmd + P, затем ввести >	Быстрый переход

## 🎯 Мультикурсор и выделение

Действие	Windows/ Linux	Mac	Когда использовать
<b>Добавить следующий курсор</b>	Ctrl + Alt + ↓	Cmd + Option + ↓	Курсор на строке ниже
<b>Добавить курсор к выделениям</b>	Ctrl + Shift + L	Cmd + Shift + L	Курсор ко всем совпадениям
<b>Добавить курсор мышкой</b>	Alt + ЛКМ	Option + ЛКМ	Кликать в нужные места
<b>Прямоугольное выделение</b>	Shift + Alt + ЛКМ	Shift + Option + ЛКМ	Выделить прямоугольник
<b>Переместить курсор в начало строк</b>	Home	Cmd + ←	Начало строки
<b>Переместить курсор в конец строк</b>	End	Cmd + →	Конец строки
<b>Переместить курсор в начало файла</b>	Ctrl + Home	Cmd + ↑	Начало файла
<b>Переместить курсор в конец файла</b>	Ctrl + End	Cmd + ↓	Конец файла

## ✨ Форматирование кода

Действие	Windows/ Linux	Mac	Когда использовать
<b>Форматировать документ</b>	Shift + Alt + F	Shift + Option + F	Форматировать весь файл
<b>Форматировать выделенное</b>	Ctrl + K, Ctrl + F	Cmd + K, Cmd + F	Форматировать выделение
<b>Emmet: развернуть аббревиатуру</b>	Tab	Tab	После ввода Emmet-сокращения
<b>Emmet: обернуть выделенное</b>	Ctrl + Shift + A	Cmd + Shift + A	Обернуть текст в тег
<b>Переместить строку/блок</b>	Alt + ↑/↓	Option + ↑/↓	Переместить строку или выделение
<b>Копировать строку/блок</b>	Shift + Alt + ↑/↓	Shift + Option + ↑/↓	Копировать строку или выделение

Действие	Windows/ Linux	Mac	Когда использовать
<b>Выровнять отступы</b>	Ctrl + ] / Ctrl + [	Cmd + ] / Cmd + [	Увеличить/уменьшить отступ

---

## Работа с окнами

Действие	Windows/ Linux	Mac	Когда использовать
<b>Разделить редактор</b>	Ctrl + \	Cmd + \	Два окна кода рядом
<b>Переключиться между окнами</b>	Ctrl + 1, Ctrl + 2...	Cmd + 1, Cmd + 2...	Переход между областями
<b>Закрыть область</b>	Ctrl + W	Cmd + W	Закрыть текущую область
<b>Переместить редактор</b>	Ctrl + Alt + ←/→	Cmd + Option + ←/→	Переместить вкладку в другую область
<b>Максимализировать редактор</b>	Ctrl + Shift + M	Cmd + Shift + M	Во весь экран
<b>Закрыть все окна</b>	Ctrl + K, W	Cmd + K, W	Закрыть все вкладки

---

## Терминал

Действие	Windows/ Linux	Mac	Когда использовать
<b>Открыть/закрыть терминал</b>	Ctrl + J	Ctrl + J	Панель терминала
<b>Создать новый терминал</b>	`Ctrl + Shift + ``	`Ctrl + Shift + ``	Новое окно терминала
<b>Очистить терминал</b>	Ctrl + L	Cmd + K	Очистить вывод
<b>Копировать из терминала</b>	Ctrl + C	Cmd + C	Копировать выделенное
<b>Вставить в терминал</b>	Ctrl + V	Cmd + V	Вставить текст
<b>Переключить фокус на</b>	`Ctrl + ``	`Ctrl + ``	Фокус на терминал

Действие	Windows/ Linux	Mac	Когда использовать
<b>терминал</b>			
<b>Убить процесс</b>	Ctrl + C	Ctrl + C	Прервать выполнение

---

## Отладка

Действие	Windows/ Linux	Mac	Когда использовать
<b>Запустить отладку</b>	F5	F5	Начать отладку
<b>Остановить отладку</b>	Shift + F5	Shift + F5	Завершить отладку
<b>Шаг с заходом</b>	F11	F11	Войти в функцию
<b>Шаг с обходом</b>	F10	F10	Пройти мимо функции
<b>Шаг с выходом</b>	Shift + F11	Shift + F11	Выйти из функции
<b>Добавить/удалить точку останова</b>	F9	F9	Точка останова на строке
<b>Показать все точки останова</b>	Ctrl + Shift + F8	Cmd + Shift + F8	Список точек останова

---

## Специальные для верстки

Действие	Windows/Linux	Mac	Когда использовать
<b>Открыть палитру цветов</b>	Ctrl + Shift + C	Cmd + Shift + C	Выбор цвета
<b>Перейти к парному тегу</b>	Ctrl + Shift + .	Cmd + Shift + .	Найти закрывающий тег
<b>Переключить wrap тегов</b>	Alt + W, A	Option + W, A	Обернуть тегом
<b>Показать все Emmet-команды</b>	Ctrl + Shift + P, ввести "Emmet"	Cmd + Shift + P, ввести "Emmet"	Список всех Emmet-команд

Действие	Windows/Linux	Mac	Когда использовать
Обновить Live Server	Сохранить файл (Ctrl+S)	Сохранить файл (Cmd+S)	При установленном Live Server

---

## 20 самых важных сочетаний для начала

Если всё запомнить сложно, начните с этих **20 ключевых сочетаний**:

Ctrl + S / Cmd + S — **Сохранить** (делайте это часто!)

Ctrl + Z / Cmd + Z — **Отменить**

Ctrl + Shift + Z / Cmd + Shift + Z — **Повторить**

Ctrl + C / Cmd + C — **Копировать**

Ctrl + V / Cmd + V — **Вставить**

Ctrl + X / Cmd + X — **Вырезать**

Ctrl + F / Cmd + F — **Поиск в файле**

Ctrl + H / Cmd + Option + F — **Замена в файле**

Ctrl + // Cmd + / — **Закомментировать строку**

Tab — **Emmet-развертывание** и отступы

Shift + Alt + F / Shift + Option + F — **Форматировать код**

Ctrl + D / Cmd + D — **Выделить следующее вхождение**

Alt + ↑/↓ / Option + ↑/↓ — **Переместить строку**

Shift + Alt + ↑/↓ / Shift + Option + ↑/↓ — **Копировать строку**

Ctrl + Shift + \ / Cmd + Shift + \ — **К парной скобке**

Ctrl + B / Cmd + B — **Скрыть/показать боковую панель**

Ctrl + J / Ctrl + J — **Показать/скрыть терминал**

Ctrl + P / Cmd + P — **Быстрый переход к файлу**

Ctrl + Shift + P / Cmd + Shift + P — **Палитра команд**

Ctrl + , / Cmd + , — **Настройки**



## Как запоминать горячие клавиши

### Метод 1: Постепенное изучение

**Первая неделя:** Освойте 5 основных сочетаний (сохранение, копирование, вставка, отмена, поиск)

**Вторая неделя:** Добавьте 5 для редактирования (комментарии, дублирование, перемещение строк)

**Третья неделя:** 5 для навигации (переход к файлу, поиск в проекте)

**Четвертая неделя:** Оставшиеся 5 для специализированных задач

## **Метод 2: Стикеры на монитор**

Распечатайте эту шпаргалку или сделайте свои стикеры с часто используемыми командами и прикрепите рядом с монитором.

## **Метод 3: Практика с таймером**

Выделите 10 минут в день для целенаправленной практики:

2 минуты: базовые операции  
3 минуты: редактирование текста  
3 минуты: навигация  
2 минуты: новые сочетания

## **Метод 4: Игра "Без мышки"**

Попробуйте выполнить небольшую задачу (создать HTML-страничку) вообще не используя мышь. Сначала будет медленно, но эффективно!

---

### **Настройка своих горячих клавиш**

Если какое-то сочетание вам неудобно или вы хотите добавить своё:

**Откройте настройки горячих клавиш:**

Ctrl + Shift + P → "Preferences: Open Keyboard Shortcuts"

Или Ctrl + K, Ctrl + S

**Найдите команду**, для которой хотите изменить сочетание

**Нажмите на карандашик** и введите новое сочетание

**Пример настройки для верстки:**

```
json
[
{
 "key": "ctrl+shift+e",
 "command": "editor.action.emmet.expandAbbreviation",
 "when": "editorTextFocus && !editorReadonly"
},
{
 "key": "alt+c",
 "command": "editor.action.colorPicker",
 "when": "editorTextFocus && !editorReadonly"
}
```

}

]

---

## Памятка для печати (краткая версия)

Text

VS CODE: 15 ГЛАВНЫХ КЛАВИШ		
Сохранить	Ctrl+S	Cmd+S
Поиск в файле	Ctrl+F	Cmd+F
Форматировать	Shift+Alt+F	Shift+Option+F
Комментарий	Ctrl+/	Cmd+/
Дублировать строку	Shift+Alt+↓	Shift+Option+↓
К парной скобке	Ctrl+Shift+\\	Cmd+Shift+\\
Палитра команд	Ctrl+Shift+P	Cmd+Shift+P
Перейти к файлу	Ctrl+P	Cmd+P
Терминал	Ctrl+J	Ctrl+J
Боковая панель	Ctrl+B	Cmd+B
Настройки	Ctrl+,	Cmd+,
Выделить вхожд.	Ctrl+D	Cmd+D
Переместить строку	Alt+↑/↓	Option+↑/↓
Палитра цветов	Ctrl+Shift+C	Cmd+Shift+C
Emmet (развер.)	Tab	Tab

---

## Заключительные советы

**Не пытайтесь выучить всё сразу** — начинайте с 2-3 сочетаний в день  
**Используйте стикеры** — пока сочетание не станет автоматическим  
**Практикуйтесь ежедневно** — даже по 5 минут в день дадут результат  
**Настройте под себя** — если сочетание неудобное, поменяйте его!  
**Помните:** Мышь — для дизайна, клавиатура — для кода!

**Дорогой друг!** Помните, что все профессионалы когда-то начинали с нуля. Каждое новое сочетание клавиш — это шаг к тому, чтобы стать

эффективнее. Скоро вы будете работать с кодом так быстро, что даже не заметите, как используете эти сочетания автоматически!  
Удачи в освоении! Ваш прогресс уже впечатляет! 

## Приложение Б: Список полезных расширений

### Дорогой друг!

Сегодня мы поговорим о "волшебных" дополнениях для VS Code — **расширениях**. Это как превратить ваш обычный инструмент в супермощную мастерскую веб-разработчика! Каждое расширение добавляет новые возможности, ускоряет работу и делает процесс верстки более приятным.

**Важное правило:** Не устанавливайте все расширения сразу! Начните с самых необходимых, постепенно добавляя новые по мере необходимости.

---

### Содержание

[Как устанавливать расширения](#)

[Обязательные расширения](#)

[Для HTML/CSS верстки](#)

[Для JavaScript](#)

[Для продуктивности](#)

[Внешний вид и темы](#)

[Инструменты разработчика](#)

[Расширения для начинающих](#)

[Как управлять расширениями](#)

[Чек-лист установки](#)

---

### ⌚ Как устанавливать расширения

#### Способ 1: Через панель расширений

Нажмите на иконку расширений в левой панели (или `Ctrl+Shift+X`)

Введите название расширения в поиск

Нажмите "Install" (Установить)

#### Способ 2: Через Marketplace в браузере

[Откройте Marketplace VS Code](#)

Найдите нужное расширение  
Нажмите "Install"  
Откроется VS Code с предложением установить

### Способ 3: Из .vsix файла

Скачайте файл .vsix  
В VS Code: Ctrl+Shift+P → "Extensions: Install from VSIX..."  
Выберите файл

---

## ОБЯЗАТЕЛЬНЫЕ РАСШИРЕНИЯ

Эти расширения нужны практически каждому веб-разработчику:

### 1. Live Server

**Что делает:** Запускает локальный сервер с автоматической перезагрузкой страницы при сохранении файлов.

**Почему обязательно:**

Видите изменения в браузере мгновенно

Не нужно вручную обновлять страницу

Поддерживает несколько браузеров одновременно

**Как использовать:**

Установите расширение

Откройте HTML-файл

Нажмите правой кнопкой → "Open with Live Server"

Или нажмите "Go Live" в правом нижнем углу VS Code

**Настройки (рекомендуемые):**

```
json
{
 "liveServer.settings.port": 5500,
 "liveServer.settings.root": "/",
 "liveServer.settings.CustomBrowser": "chrome"
}
```

### 2. Prettier - Code formatter

**Что делает:** Автоматически форматирует ваш код по единым правилам.

**Почему обязательно:**

Код всегда выглядит аккуратно

Единый стиль во всем проекте  
Экономит время на ручное форматирование

### Настройки для верстки:

```
json
{
 "editor.formatOnSave": true,
 "prettier.htmlWhitespaceSensitivity": "css",
 "prettier.printWidth": 100,
 "prettier.tabWidth": 2,
 "prettier.useTabs": false
}
```

### 3. Auto Rename Tag

**Что делает:** Автоматически переименовывает парный HTML/XML тег.

#### Пример:

```
html
<!-- Меняете открывающий тег: -->
<div>Текст</div>
<!-- Закрывающий автоматически станет <section> -->
```

### 4. Bracket Pair Colorizer

**Что делает:** Раскрашивает парные скобки, фигурные и квадратные скобки в разные цвета.

#### Почему полезно:

Легко находить парные скобки  
Видеть вложенность кода  
Найти ошибки в структуре

---

## 🎨 ДЛЯ HTML/CSS ВЕРСТКИ

### 1. HTML CSS Support

**Что делает:** Автодополнение для классов и ID из CSS-файлов.

#### Пример работы:

```
html
<!-- В HTML файле: -->
<div class="|" <!-- Начните писать class="", появится список классов из
CSS -->
```

## 2. CSS Peek

**Что делает:** Позволяет заглянуть в определение CSS класса прямо из HTML.

**Как использовать:**

В HTML файле наведите курсор на class

Нажмите **Ctrl** (или **Cmd** на Mac) и кликните

Откроется всплывающее окно с CSS-правилами

## 3. Color Highlight

**Что делает:** Подсвечивает цветовые значения в коде.

**Пример:**

css

```
/* Цвета будут подсвечены соответствующим фоном */
.color {
 color: #ff0000; /* Красный фон */
 background: #00ff00; /* Зеленый фон */
 border-color: blue; /* Синий фон */
}
```

## 4. Image preview

**Что делает:** Показывает превью изображений прямо в редакторе.

**Почему полезно:**

Видите картинки не открывая их

Проверяете пути к изображениям

Видите размеры и формат

## 5. SVG Viewer

**Что делает:** Просмотр SVG файлов прямо в VS Code.

## 6. Emmet Live

**Что делает:** Улучшенная поддержка Emmet с предпросмотром.

---

## ДЛЯ JAVASCRIPT

### 1. JavaScript (ES6) code snippets

**Что делает:** Коллекция синтаксических сокращений (шорткатов) для JavaScript.

**Примеры сокращений:**

imp → import moduleName from 'module'

exp → export default moduleName

clg → console.log()

### 2. ES7+ React/Redux/React-Native snippets

**Что делает:** Синтаксические сокращения для современного JavaScript, даже если не используете React.

### 3. npm Intellisense

**Что делает:** Автодополнение для промисов в import/require.

### 4. Import Cost

**Что делает:** Показывает размер импортируемых библиотек прямо в коде.

---

## ДЛЯ ПРОДУКТИВНОСТИ

### 1. Todo Tree

**Что делает:** Собирает все комментарии TODO, FIXME, NOTE в проекте в единое дерево.

**Пример использования:**

```
javascript
// TODO: Добавить валидацию формы
// FIXME: Исправить баг на мобильных
// NOTE: Важная информация о коде
```

После установки слева появится панель со всеми такими комментариями.

## 2. Project Manager

**Что делает:** Управление несколькими проектами, быстрое переключение между ними.

**Почему полезно:**

Сохраняет список ваших проектов

Быстрый доступ к часто используемым

Группировка проектов по категориям

## 3. Bookmarks

**Что делает:** Закладки для строк кода.

**Как использовать:**

Ctrl+Alt+K — добавить/удалить закладку

Ctrl+Alt+J — следующая закладка

Ctrl+Alt+L — перейти к закладке

## 4. GitLens

**Что делает:** Расширенные возможности для работы с Git.

**Основные функции:**

Показывает, кто и когда менял каждую строку кода

История изменений файла

Встроенные команды Git

## 5. Error Lens

**Что делает:** Показывает ошибки и предупреждения прямо в строке кода.

**Пример:**

css

```
/* Ошибка появится прямо в коде: */
.selector {
color: red;
/* [css] Unknown property: 'colr' */
}
```

## ВНЕШНИЙ ВИД И ТЕМЫ

### 1. Material Theme / Material Icon Theme

**Что делает:** Красивые темы и иконки для VS Code.

**Почему полезно:**

Приятнее работать визуально

Иконки помогают быстро ориентироваться в файлах

Разные темы для разных времени суток

### 2. vscode-pets

**Что делает:** Добавляет питомцев в ваш редактор кода!

**Почему весело:**

Маленькое животное в углу экрана

Разные виды: кошки, собаки, динозавры

Помогает не скучать во время работы

### 3. Bracket Pair Colorizer 2

**Что делает:** Улучшенная версия раскраски скобок.

### 4. indent-rainbow

**Что делает:** Раскрашивает отступы в разные цвета.

**Почему полезно:**

Видите уровень вложенности

Легко находите ошибки в отступах

Красиво выглядит

---

## ИНСТРУМЕНТЫ РАЗРАБОТЧИКА

### 1. REST Client

**Что делает:** Отправка HTTP-запросов прямо из VS Code.

**Пример файла .http:**

http

GET https://api.example.com/users

Content-Type: application/json

```

POST https://api.example.com/users
Content-Type: application/json
```

```
{
 "name": "Иван",
 "email": "ivan@example.com"
}
```

## 2. Thunder Client

**Что делает:** Альтернатива Postman прямо в VS Code.

## 3. Docker

**Что делает:** Работа с Docker контейнерами.

## 4. Database Client

**Что делает:** Работа с базами данных (MySQL, PostgreSQL, SQLite).

---

## 💡 РАСШИРЕНИЯ ДЛЯ НАЧИНАЮЩИХ

### 1. Code Spell Checker

**Что делает:** Проверка орфографии в коде.

#### Почему полезно для начинающих:

Ловит опечатки в именах переменных

Проверяет комментарии

Можно добавить свои словари

### 2. Polacode

**Что делает:** Создание красивых скриншотов кода.

#### Как использовать:

Выделите код

Ctrl+Shift+P → "Polacode"

Настройте фон, тени

Сохраните изображение

### **3. Quokka.js**

**Что делает:** Интерактивная песочница для JavaScript.

**Почему полезно для обучения:**

Видите результат выполнения кода сразу

Можно экспериментировать

Отладка в реальном времени

### **4. Markdown Preview Enhanced**

**Что делает:** Просмотр Markdown файлов с расширенными возможностями.

**Почему полезно:**

Пишете документацию в Markdown

Видите результат сразу

Экспорт в PDF, HTML

### **5. Code Runner**

**Что делает:** Запуск кода на разных языках прямо из VS Code.

**Поддерживает:**

JavaScript, TypeScript

Python, PHP, Go

Java, C, C++

И многие другие

---

## **ДЛЯ ВЕБ-РАЗРАБОТКИ (СПЕЦИАЛИЗИРОВАННЫЕ)**

### **1. CSS Navigation**

**Что делает:** Навигация по CSS-правилам как в браузере.

### **2. HTMLHint**

**Что делает:** Линтер (проверка качества) для HTML..

**Находит:**

Незакрытые теги

Неправильные атрибуты

Проблемы с доступностью

### **3. Stylelint**

**Что делает:** Линтер для CSS/SCSS/Less.

### **4. Path Intellisense**

**Что делает:** Автодополнение путей к файлам.

**Пример:**

html

 *<!-- Начните писать, появится список файлов в папке -->*

## **5. Bootstrap 4, Font awesome 4, Font Awesome 5 Free & Pro snippets**

**Что делает:** Сниппеты для Bootstrap и Font Awesome.

---

## **КАК УПРАВЛЯТЬ РАСШИРЕНИЯМИ**

### **1. Отключение расширений**

Если расширение мешает или замедляет работу:

Ctrl+Shift+X → панель расширений

Найдите расширение

Нажмите "Disable" (Отключить) или "Uninstall" (Удалить)

### **2. Синхронизация настроек**

**Settings Sync** расширение:

Установите Settings Sync

Войдите через GitHub

Ваши настройки и расширения сохраняются в облаке

На другом компьютере просто войдите и всё восстановится

### **3. Группировка расширений по профилям**

Создавайте разные профили для разных проектов:

json

{

"workbench.experimental.settings.profiles": [

```
{
 "name": "Веб-разработка",
 "extensions": [
 "ritwickdey.liveserver",
 "esbenp.prettier-vscode"
],
},
{
 "name": "Документация",
 "extensions": [
 "shd101wy.markdown-preview-enhanced"
]
}
]
```

#### 4. Полезные команды для работы с расширениями

Ctrl+Shift+P → "Extensions: Show Installed Extensions"

Ctrl+Shift+P → "Extensions: Show Recommended Extensions"

Ctrl+Shift+P → "Extensions: Disable All Installed Extensions"

### СРАВНЕНИЕ ПОПУЛЯРНЫХ РАСШИРЕНИЙ

#### Форматтеры кода:

Расширение	Плюсы	Минусы	Для кого
Prettier	Самый популярный, много настроек	Может быть медленным на больших файлах	Для всех
Beautify	Быстрый, простой	Меньше настроек	Для начинающих
HTML-CSS-JS Prettify	Специализированный для веба	Только для веб-языков	Для верстальщиков

#### Live-серверы:

Расширение	Плюсы	Минусы
Live Server	Самый популярный,	Базовые функции

Расширение	Плюсы	Минусы
	простой	
Five Server	Более быстрый, больше функций	Менее популярный
Live Preview	Встроенный в VS Code	Ограниченные возможности

## CSS помощники:

Расширение	Что делает
CSS Peek	Заглянуть в определение класса
IntelliSense for CSS	Умное автодополнение
CSS Navigation	Навигация как в DevTools

## ЧТО НЕЛЬЗЯ УСТАНАВЛИВАТЬ (ОПАСНЫЕ РАСШИРЕНИЯ)

### Признаки опасных расширений:

Мало установок (менее 10,000)

Нет описания или описание на непонятном языке

Нет обновлений больше года

Подозрительные разрешения (доступ ко всем файлам)

Нет отзывов или отрицательные отзывы

### Проверяйте перед установкой:

Автора расширения

Дата последнего обновления

Количество установок

Отзывы и рейтинг

Необходимые разрешения

## ЧЕК-ЛИСТ УСТАНОВКИ ДЛЯ НАЧИНАЮЩЕГО ВЕРСТАЛЬЩИКА

### Фаза 1: Самые необходимые (День 1)

Live Server — для просмотра страниц

Auto Rename Tag — для удобной работы с HTML

**Prettier** — для красивого кода

### Фаза 2: Улучшаем верстку (Неделя 1)

**HTML CSS Support** — автодополнение классов

**CSS Peek** — быстрый просмотр стилей

**Color Highlight** — визуализация цветов

**Image preview** — просмотр изображений

### Фаза 3: Продуктивность (Месяц 1)

**Todo Tree** — управление задачами

**Project Manager** — управление проектами

**Bracket Pair Colorizer** — навигация по коду

**Error Lens** — ошибки на виду

### Фаза 4: Внешний вид (Когда освоитесь)

**Material Icon Theme** — красивые иконки

**Нравится тема** — выберите тему по вкусу

**indent-rainbow** — раскраска отступов

### Фаза 5: Специализация (По необходимости)

**Для JavaScript** — если пишете скрипты

**Для фреймворков** — если используете React/Vue

**Для бэкенда** — если работаете с сервером

---

## ⌚ НАСТРОЙКИ ПОСЛЕ УСТАНОВКИ РАСШИРЕНИЙ

После установки ключевых расширений добавьте в настройки:

```
json
{
 // Автосохранение
 "files.autoSave": "afterDelay",
 "files.autoSaveDelay": 1000,
 // Форматирование при сохранении
 "editor.formatOnSave": true,
```

```
// Live Server
"liveServer.settings.donotVerifyTags": true,

// Prettier
"prettier.singleQuote": true,
"prettier.trailingComma": "es5",

// Цвета скобок
"bracketPairColorizer.consecutivePairColors": [
 "()",
 "[]",
 "{}",
 ["#FFD700", "#FFA500", "#FF8C00"]
]
}
```

---



## СОВЕТЫ ПО ИСПОЛЬЗОВАНИЮ РАСШИРЕНИЙ

### Совет 1: Не перегружайте VS Code

Каждое расширение потребляет память.  
Если VS Code начинает тормозить:

Отключите неиспользуемые расширения  
Используйте профили для разных типов проектов  
Регулярно проверяйте и удаляйте ненужное

### Совет 2: Изучайте возможности расширений

После установки:

Прочтайте документацию  
Посмотрите настройки  
Изучите горячие клавиши  
Проверьте, какие команды добавляет расширение

### Совет 3: Создайте свой набор

Со временем вы поймете, какие расширения нужны именно вам.  
Сохраните их список:  
markdown

# Мой набор расширений VS Code

## Обязательные:

1. Live Server
2. Prettier
3. Auto Rename Tag

## Для верстки:

1. CSS Peek
2. Color Highlight
3. HTML CSS Support

## Для продуктивности:

1. Todo Tree
2. Bookmarks
3. Project Manager

Дата обновления: 2024

## Совет 4: Обновляйте регулярно

Разработчики выпускают обновления:

Включайте автообновления

Раз в месяц проверяйте обновления вручную

Читайте changelog важных расширений

## Совет 5: Делайте бэкап настроек

Используйте **Settings Sync** или экспортируйте настройки:

Ctrl+Shift+P → "Preferences: Export Settings"

Сохраните файл в надежном месте

При пере установке импортируйте

---

## ❓ ЧАСТЫЕ ВОПРОСЫ

**Вопрос: VS Code стал медленно работать после установки расширений**

**Ответ:**

Откройте панель разработчика: Ctrl+Shift+P → "Developer: Show Running Extensions"

Посмотрите, какие расширения потребляют много ресурсов  
Отключите тяжелые расширения или найдите альтернативы

### **Вопрос: Расширение конфликтует с другим**

**Ответ:**

Попробуйте отключить одно из расширений

Проверьте порядок загрузки

Посмотрите issues на GitHub расширения

### **Вопрос: Как поделиться списком расширений с коллегой?**

**Ответ:**

bash

```
Создать список установленных расширений
code --list-extensions > extensions.txt
```

```
Установить все расширения из списка
```

```
cat extensions.txt | xargs -L 1 code --install-extension
```

### **Вопрос: Где хранятся установленные расширения?**

**Ответ:**

**Windows:** %USERPROFILE%\.vscode\extensions

**Mac:** ~/.vscode/extensions

**Linux:** ~/.vscode/extensions

---

## МОЙ ЛИЧНЫЙ ТОП-10 РАСШИРЕНИЙ

**Live Server** — без него уже не могу

**Prettier** — код всегда аккуратный

**Todo Tree** — все задачи перед глазами

**Auto Rename Tag** — экономит кучу времени

**Material Icon Theme** — красиво и удобно

**CSS Peek** — не переключаясь между файлами

**Bracket Pair Colorizer** — вижу структуру кода

**Error Lens** — ошибки сразу заметны

**Image preview** — вижу что подключаю

**vscode-pets** — для хорошего настроения

---

## РЕСУРСЫ ДЛЯ ПОИСКА РАСШИРЕНИЙ

### Официальные источники:

[VS Code Marketplace](#)

[Официальная документация](#)

### Подборки расширений:

[Awesome VS Code](#)

[VS Code Can Do That?](#)

[VS Code Tips & Tricks](#)

### YouTube каналы с обзорами:

"VS Code расширения для веб-разработки"

"Топ-20 расширений для верстальщика"

"Как ускорить работу в VS Code"

---

## ЗАКЛЮЧЕНИЕ

**Дорогой друг!** Расширения — это как суперспособности для вашего редактора кода. Начните с самого необходимого, постепенно добавляя новые инструменты в свой арсенал.

### Помните:

**Качество важнее количества** — лучше 10 полезных расширений, чем 50 бесполезных

**Изучайте перед использованием** — каждая минута, потраченная на изучение, сэкономит час работы

**Настраивайте под себя** — нет идеальных настроек, есть удобные лично для вас

**Делитесь находками** — если нашли крутое расширение, расскажите коллегам

### Ваш путь:

**Неделя 1:** Установите Live Server и Prettier

**Неделя 2:** Добавьте Auto Rename Tag и CSS Peek

**Месяц 1:** Поэкспериментируйте с 2-3 расширениями из списка

**Месяц 3:** У вас уже будет свой настроенный идеально VS Code

**Удачи в освоении новых инструментов!** Помните, что каждый профессионал когда-то начинал с установки своего первого расширения. Скоро вы будете работать так эффективно, что даже не заметите, как эти инструменты стали частью вашего рабочего процесса! 

## Приложение В: Частые проблемы и их решение

**Дорогой друг!**

В этой шпаргалке собраны самые частые проблемы, с которыми сталкиваются начинающие веб-мастера при работе с VS Code. Не переживайте — каждая из этих проблем решаема, и скоро вы будете справляться с ними на автомате!

---

### Содержание

[Проблемы с установкой и запуском](#)

[Проблемы с файлами и папками](#)

[Проблемы с HTML/CSS](#)

[Проблемы с расширениями](#)

[Проблемы с терминалом](#)

[Проблемы с Live Server](#)

[Проблемы с русским языком](#)

[Проблемы с производительностью](#)

[Проблемы с сохранением](#)

[Разное и полезное](#)

---



### ПРОБЛЕМЫ С УСТАНОВКОЙ И ЗАПУСКОМ

#### ✗ Проблема 1: VS Code не запускается после установки

**Симптомы:** Дважды кликаете по иконке, но ничего не происходит.

**Решение:**

**Проверьте системные требования:**

Windows 8+, macOS 10.11+, Linux

1 ГБ RAM минимум

Процессор не старше 5-7 лет

**Попробуйте запустить от администратора:**

Windows: правой кнопкой → "Запуск от имени администратора"

Mac: через терминал: `sudo code`

**Удалите старые настройки:**

`text`

Windows: C:\Users\ВАШЕ\_ИМЯ\.vscode

Mac: `~/.vscode`

Linux: `~/.config/Code`

Удалите папку и переустановите VS Code

**Проверьте антивирус:** иногда блокирует запуск

## ✖ Проблема 2: Не открываются файлы двойным кликом

**Симптомы:** Файлы `.html`, `.css` не открываются в VS Code.

**Решение:**

**Windows:**

```
powershell
Откройте PowerShell от администратора
Установите ассоциацию для HTML:
ftype VSCode.html="C:\Path\To\Code.exe" "%1"
Для CSS:
ftype VSCode.css="C:\Path\To\Code.exe" "%1"
```

**Или через интерфейс:**

Правой кнопкой на файле → "Открыть с помощью"

Выберите VS Code

Поставьте галочку "Всегда использовать это приложение"

**Mac:**

Правой кнопкой → "Get Info"

В "Open with:" выберите VS Code

Нажмите "Change All..."

## ✖ Проблема 3: VS Code открывается с пустым окном

**Симптомы:** Запускается, но нет ни меню, ни панелей.

**Решение:**

**Восстановите настройки:**

`text`

`Ctrl+Shift+P` → "Developer: Reload Window"

**Запустите с флагами:**

`bash`

# Windows:

`code --disable-extensions`

# Mac/Linux:

code --disable-extensions

**Сбросьте настройки:**

bash

# Windows:

code --user-data-dir="C:\temp\vscode"

---

## ПРОБЛЕМЫ С ФАЙЛАМИ И ПАПКАМИ

### Проблема 4: "Файл не найден" при открытии картинок/стилей

**Симптомы:** В браузере картинки не отображаются, стили не применяются.

**Решение:**

html

```
<!-- НЕПРАВИЛЬНО: -->

<!-- ПРАВИЛЬНО: -->

```

**Шаги решения:**

**Используйте относительные пути:**

text

```
проект/
 ├── index.html
 ├── style.css
 └── images/
 └── logo.png
```

В HTML: 

**Проверьте расширение файла:**

Откройте Проводник (Windows) или Finder (Mac)

Включите отображение расширений файлов

Убедитесь, что файл называется logo.png, а не logo.png.jpg

**Проверьте регистр букв:**

Logo.PNG ≠ logo.png (на Linux/Mac)

Всегда используйте маленькие буквы

## Проблема 5: Не могу сохранить файл

**Симптомы:** VS Code просит сохранить как другой файл или говорит "доступ запрещен".

**Решение:**

**Сохраните в другую папку:**

Например, C:\Users\Иван\Documents\MySite

Не сохраняйте в Program Files или на рабочий стол

**Проверьте права:**

Windows: правой кнопкой на папке → "Свойства" → "Безопасность"

Mac: Cmd+I на папке → "Общий доступ и права доступа"

**Используйте автосохранение:**

```
json
// В настройках VS Code:
{
 "files.autoSave": "afterDelay",
 "files.autoSaveDelay": 1000
}
```

## Проблема 6: Файлы в папке не отображаются

**Симптомы:** Открыли папку, но файлы не видны в боковой панели.

**Решение:**

**Проверьте фильтры:**

В поиске (вверху Explorer) может быть текст

Очистите поиск

**Исключения в settings.json:**

```
json
{
 "files.exclude": {
 "**/.git": true,
 "**/.DS_Store": true
 }
}
```

Убедитесь, что ваши файлы не попали в exclude

**Обновите папку:**

Нажмите F1 или Ctrl+Shift+P

Введите "File: Refresh Explorer"

## Проблема 7: Кодировка файлов (кракозябры)

**Симптомы:** Русский текст отображается как Ðóññêèé ðåêñò.

**Решение:**

**Сохраните в UTF-8:**

В правом нижнем углу VS Code нажмите на кодировку (например, "UTF-8")

Выберите "Сохранить с кодировкой"

Выберите "UTF-8"

**Установите кодировку по умолчанию:**

```
json
{
 "files.encoding": "utf8",
 "files.autoGuessEncoding": true
}
```

**В HTML укажите кодировку:**

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

---

## ПРОБЛЕМЫ С HTML/CSS

### Проблема 8: HTML-код не отображается в браузере

**Симптомы:** Виден исходный код вместо страницы.

**Решение:**

**Проверьте расширение файла:**

Должно быть .html, а не .txt или .htm

Включите отображение расширений в системе

**Проверьте начало файла:**

```
html
<!-- ДОЛЖНО БЫТЬ: -->
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">

<!-- А НЕ: -->
<?xml version="1.0" encoding="UTF-8"?>
```

**Проверьте сервер:**  
Открывайте через `http://`, а не `file://`

Используйте Live Server

## ✖ Проблема 9: CSS не применяется

**Симптомы:** Стили написаны, но не работают.

**Решение:**

html

```
<!-- 1. Проверьте подключение: -->
<link rel="stylesheet" href="style.css">
<!-- HE: -->
<link rel="stylesheet" href="style">
```

**Пошаговая диагностика:**

**F12 в браузере → вкладка Console**

Если видите ошибку 404 — файл не найден

Проверьте путь

**Проверьте синтаксис CSS:**

CSS

```
/* ПРАВИЛЬНО: */
.class { color: red; }

/* НЕПРАВИЛЬНО: */
.class { color: red } /* нет точки с запятой */
```

**Проверьте специфичность:**

CSS

```
/* ЭТО не перекроет ЭТО: */
div { color: blue; }
.my-class { color: red; }
```

```
<!-- Если в HTML: -->
<div class="my-class">Текст</div>
```

**Используйте !important (только для теста):**

CSS

```
.class { color: red !important; }
```

## ✖ Проблема 10: Картинки не отображаются

**Решение:**

**Проверьте путь (самая частая ошибка):**

html  
  <!-- Если структура: -->

  проект/  
    index.html  
    img/  
    photo.jpg

<!-- То путь должен быть: -->



**Проверьте имя файла:**

На Windows: photo.jpg ≠ Photo.JPG

На Mac/Linux: регистр важен!

**Проверьте формат:**

Должен быть .jpg, .png, .gif, .svg

Не .jreg (обычно)

**Проверьте размер:**

Откройте картинку в просмотрщике

Может быть повреждена

## ✖ Проблема 11: Шрифты не работают

**Решение:**

css

```
/* 1. Проверьте синтаксис: */
@font-face {
 font-family: 'MyFont';
 src: url('fonts/myfont.woff2') format('woff2');
}
```

```
body {
 font-family: 'MyFont', sans-serif; /* кавычки! */
}
```

**Шаги:**

**Проверьте форматы:**

Современные браузеры: .woff2, .woff

Старые: .ttf, .eot

**Проверьте пути:**

text

проект/

```
├── index.html
├── style.css
└── fonts/
 └── myfont.woff2
```

В CSS (если style.css в корне):

```
src: url('fonts/myfont.woff2')
```

**Проверьте CORS (если шрифт с другого сайта):**

CSS

```
@font-face {
 font-display: swap; /* чтобы текст был виден сразу */
}
```



## ПРОБЛЕМЫ С РАСШИРЕНИЯМИ

### ✗ Проблема 12: Расширения не устанавливаются

**Симптомы:** Кнопка "Install" неактивна или установка зависает.

**Решение:**

**Проверьте интернет:**

VS Code нужен интернет для установки

Проверьте прокси-настройки

**Очистите кэш:**

text

Windows: %USERPROFILE%\.vscode\extensions

Mac: ~/.vscode/extensions

Удалите папку и перезапустите VS Code

**Установите вручную:**

Скачайте .vsix файл с Marketplace

В VS Code: Ctrl+Shift+P → "Extensions: Install from VSIX..."

### ✗ Проблема 13: Расширение не работает

**Решение:**

**Перезагрузите VS Code:**

Ctrl+Shift+P → "Developer: Reload Window"

**Проверьте требования:**

Некоторые расширения требуют других расширений

Читайте документацию

**Проверьте настройки:**

```
json
{
 "liveServer.settings.port": 5500,
 "prettier.enable": true
}
```

**Запустите без расширений:**

bash  
code --disable-extensions

Если проблема исчезла — виновато расширение

## ✖ Проблема 14: VS Code тормозит из-за расширений

**Решение:**

**Найдите виновника:**

Ctrl+Shift+P → "Developer: Show Running Extensions"

Посмотрите, какие потребляют много CPU

**Отключите ненужные:**

Оставляйте только для текущего проекта

Используйте профили расширений

**Полезные команды:**

```
json
{
 "extensions.ignoreRecommendations": true,
 "extensions.autoUpdate": false
}
```

---

## 💻 ПРОБЛЕМЫ С ТЕРМИНАЛОМ

### ✖ Проблема 15: Терминал не открывается

**Симптомы:** Ctrl+J не работает или терминал пустой.

**Решение:**

**Сбросьте терминал:**

Ctrl+Shift+P → "Terminal: Relaunch Active Terminal"

**Смените оболочку:**

В правом нижнем углу терминала нажмите +

Выберите другую (PowerShell, cmd, bash)

**Проверьте настройки:**

```
json
```

```
{
 "terminal.integrated.shell.windows": "C:\\Windows\\System32\\cmd.exe",
 "terminal.integrated.defaultProfile.windows": "Command Prompt"
}
```

## ✖ Проблема 16: Команды не работают в терминале

**Симптомы:** npm, git и другие команды "не найдены".

**Решение:**

**Добавьте в PATH:**

Windows: Панель управления → Система → Переменные среды

Добавьте пути к программам

**Перезапустите VS Code:** после изменения PATH

**Используйте полный путь:**

bash

# Вместо:

npm install

# Попробуйте:

C:\\Program Files\\nodejs\\npm.cmd install

## ✖ Проблема 17: Терминал показывает неправильную кодировку

**Симптомы:** Русский текст как кракозябры.

**Решение:**

**Для Windows (PowerShell):**

powershell

# В профиле PowerShell:

[Console]::OutputEncoding = [System.Text.Encoding]::UTF8

**Настройки VS Code:**

```
json
```

```
{
```

```
 "terminal.integrated.defaultProfile.windows": "Command Prompt",
```

```
 "terminal.integrated.shellArgs.windows": ["-NoExit", "/K", "chcp 65001"]
```

```
}
```

**Используйте Windows Terminal** (новый терминал от Microsoft)

---

## ПРОБЛЕМЫ С LIVE SERVER

### Проблема 18: Live Server не запускается

**Симптомы:** Кнопка "Go Live" неактивна или сервер не стартует.

**Решение:**

**Проверьте порт:**

Live Server использует порт 5500

Если занят, смените:

```
json
{
 "liveServer.settings.port": 5501
}
```

**Запустите из HTML файла:**

Откройте HTML файл

Правой кнопкой → "Open with Live Server"

**Проверьте firewall:**

Windows может блокировать

Разрешите VS Code в firewall

### Проблема 19: Страница не обновляется автоматически

**Решение:**

**Проверьте автосохранение:**

```
json
{
 "files.autoSave": "afterDelay",
 "files.autoSaveDelay": 500
}
```

**Обновите вручную:** Ctrl+S

**Перезапустите сервер:**

Нажмите "Go Live" еще раз

Или перезапустите VS Code

### Проблема 20: Live Server открывает не тот файл

**Симптомы:** Открывается index.html, а у вас about.html.

**Решение:**

**Укажите стартовый файл:**

```
json
{
 "liveServer.settings.root": "/",
 "liveServer.settings.file": "about.html"
}
```

### **Запускайте из нужного файла:**

Откройте about.html

Запустите Live Server

---

## ПРОБЛЕМЫ С РУССКИМ ЯЗЫКОМ

### Проблема 21: Русские буквы в коде и комментариях

#### **Решение:**

#### **Всегда используйте UTF-8:**

В правом нижнем углу нажмите на кодировку

Выберите "Сохранить с кодировкой" → "UTF-8"

#### **Настройки по умолчанию:**

```
json
{
 "files.encoding": "utf8",
 "files.autoGuessEncoding": true
}
```

### **HTML:**

```
html
<!DOCTYPE html>
<html lang="ru"> <!-- Важно! -->
<head>
<meta charset="UTF-8"> <!-- Обязательно! -->
```

### Проблема 22: Русские имена файлов и папок

#### **Решение:**

#### **Лучше использовать английские имена:**

```
text
вместо: "проект/index.html"
```

лучше: "project/index.html"

### Если нужно по-русски:

Убедитесь в UTF-8

В путях используйте /, а не \

### Проблемы с GitHub:

Некоторые сервисы плохо работают с кириллицей

Используйте транслит: proekt вместо проект

## ✖ Проблема 23: Русский текст в консоли браузера

### Решение:

javascript

// Правильно:

```
console.log("Привет, мир!");
```

// В HTML:

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

// В JavaScript файле:

// Убедитесь, что файл сохранен в UTF-8 без BOM

---

## ⚠ ПРОБЛЕМЫ С ПРОИЗВОДИТЕЛЬНОСТЬЮ

## ✖ Проблема 24: VS Code работает медленно

### Решение:

Отключите ненужные расширения

Увеличьте память:

```
json
{
 "editor.renderWhitespace": "none",
 "editor.minimap.enabled": false,
 "workbench.enableExperiments": false
}
```

Проверьте аппаратное ускорение:

json

```
{
```

```
"disable-hardware-acceleration": false
}
```

Запустите: code --disable-hardware-acceleration

## ✖ Проблема 25: Автодополнение тормозит

**Решение:**

```
json
{
 "editor.quickSuggestions": {
 "other": true,
 "comments": false,
 "strings": false
 },
 "editor.suggestOnTriggerCharacters": true,
 "editor.acceptSuggestionOnEnter": "on"
}
```

## ✖ Проблема 26: Большие файлы открываются медленно

**Решение:**

**Используйте ограничения:**

```
json
{
 "files.maxMemoryForLargeFilesMB": 4096,
 "editor.largeFileOptimizations": true
}
```

**Разбейте большие файлы:**

CSS: на несколько файлов

HTML: используйте include или компоненты

---

## 💾 ПРОБЛЕМЫ С СОХРАНЕНИЕМ

### ✖ Проблема 27: "Файл изменен вне редактора"

**Симптомы:** VS Code показывает предупреждение при сохранении.

**Решение:**

**Это нормально, если:**

Вы открыли файл в двух окнах VS Code

Файл изменила другая программа

**Действия:**

Нажмите "Перезагрузить", если хотите загрузить внешние изменения

Нажмите "Сохранить", чтобы перезаписать

**Отключите предупреждение:**

```
json
{
 "files.hotExit": "onExit",
 "files.autoSave": "afterDelay"
}
```

 **Проблема 28: Файл не сохраняется с нужным расширением**

**Решение:**

**При сохранении как...:**

В диалоге сохранения введите полное имя: style.css

Не style или style.

**Проверьте настройки:**

```
json
{
 "files.defaultLanguage": "html", // для .html файлов
 "files.associations": {
 "*.css": "css"
 }
}
```

 **Проблема 29: Потеря несохраненных изменений**

**Решение:**

**Включите автосохранение:**

```
json
{
 "files.autoSave": "afterDelay",
 "files.autoSaveDelay": 1000
}
```

**Используйте временные файлы:**

VS Code сохраняет временные копии

При аварийном закрытии предложит восстановить

**Git:** начните использовать контроль версий

---

## РАЗНОЕ И ПОЛЕЗНОЕ

### Проблема 30: Не работает Emmet

**Решение:**

**Проверьте контекст:**

Emmet работает в HTML, CSS, JavaScript (JSX)

Убедитесь, что внизу окна правильный язык

**Используйте Tab:**

html

div.class<!-- напечатайте и нажмите Tab -->

**Настройки:**

```
json
{
 "emmet.triggerExpansionOnTab": true,
 "emmet.includeLanguages": {
 "javascript": "javascriptreact"
 }
}
```

### Проблема 31: Не вижу ошибок в коде

**Решение:**

**Включите валидацию:**

```
json
{
 "html.validate.scripts": true,
 "html.validate.styles": true,
 "css.validate": true
}
```

**Установите расширения:**

HTMLHint для HTML

Stylelint для CSS

**Используйте браузер:**

F12 → Console

Там видны все ошибки

## Проблема 32: Не работают горячие клавиши

**Решение:**

**Проверьте конфликты:**

Ctrl+Shift+P → "Preferences: Open Keyboard Shortcuts"

Ведите команду, посмотрите назначенные клавиши

**Сбросьте настройки:**

Удалите keybindings.json

Перезапустите VS Code

**Проверьте раскладку:**

Некоторые горячие клавиши не работают на русской раскладке

Переключитесь на английскую

## Проблема 33: Пропала боковая панель

**Решение:**

**Верните панель:**

Ctrl+Shift+E — Explorer (файлы)

Ctrl+Shift+X — Extensions (расширения)

Ctrl+Shift+G — Source Control (Git)

**Или через меню:**

View → Appearance → Show Sidebar

**Сбросьте layout:**

Ctrl+Shift+P → "View: Reset View Locations"

## Проблема 34: Съехала layout (расположение окон)

**Решение:**

**Перетащите панели:**

Зажмите заголовок панели и перетащите

**Сбросьте:**

json

Ctrl+Shift+P → "View: Reset View Locations"

**Сохраните layout:**

Используйте расширение "Settings Sync"

## Проблема 35: Не видно расширений файлов

**Решение:**

**В VS Code:**

Внизу окна нажмите на язык (например, "HTML")

Выберите нужный

**В системе:**

Windows: Проводник → Вид → Показать расширения

Mac: Finder → Настройки → Дополнительно → "Показывать расширения"

---

 **АВАРИЙНЫЕ СИТУАЦИИ**

 **СРОЧНО: VS Code не запускается, а нужно работать**

**Аварийное решение:**

**Используйте онлайн-редакторы:**

[CodePen](#) — для HTML/CSS/JS

[JSFiddle](#)

[StackBlitz](#)

**Блокнот:**

Сохраняйте как .html

Открывайте в браузере

**Установите другой редактор на время:**

Notepad++ (Windows)

Sublime Text

Atom

 **СРОЧНО: Удалил важный файл**

**Восстановление:**

**Корзина:** проверьте корзину

**Временные файлы VS Code:**

text

Windows: %APPDATA%\Code\Backups

Mac: ~/Library/Application Support/Code/Backups

**Автосохранение:** если было включено

 **СРОЧНО: Закрыл окно с несохраненным кодом**

**Решение:**

VS Code обычно сохраняет временные копии

При повторном открытии предложит восстановить

Проверьте папку Backups (см. выше)

---

## ИНСТРУМЕНТЫ ДЛЯ ДИАГНОСТИКИ

### Встроенные команды для диагностики:

text

Ctrl+Shift+P → "Developer: Show Running Extensions"

Ctrl+Shift+P → "Developer: Open Process Explorer"

Ctrl+Shift+P → "Developer: Toggle Developer Tools"

### Логи:

text

Windows: %APPDATA%\Code\logs

Mac: ~/Library/Application Support/Code/logs

Linux: ~/.config/Code/logs

### Полезные команды:

bash

# Запуск с диагностикой:

code --verbose

code --disable-extensions

code --user-data-dir="C:\temp\vscode"

# Создание дампа проблем:

code --status

---

## КОГДА ОБРАЩАТЬСЯ ЗА ПОМОЩЬЮ

### Что подготовить перед обращением:

**Точное описание проблемы:** что делали, что ожидали, что получили

**Версия VS Code:** Help → About

**ОС:** Windows/Mac/Linux, версия

**Расширения:** список установленных

**Код:** минимальный пример для воспроизведения

**Где искать помощь:**

**Официальная документация:** [code.visualstudio.com/docs](https://code.visualstudio.com/docs)

**Stack Overflow:** используйте теги [vscode], [html], [css]

**GitHub Issues:** для конкретных расширений

**Русскоязычные сообщества:** Хабр, Телеграм-чаты

## Как правильно задать вопрос:

markdown

**\*\*Проблема:\*\*** [краткое описание]

**\*\*Ожидаемое поведение:\*\*** [что должно было произойти]

**\*\*Фактическое поведение:\*\*** [что произошло на самом деле]

**\*\*Шаги для воспроизведения:\*\***

1. Открыть VS Code
2. Создать файл test.html
3. Вписать [код]
4. Сохранить
5. Запустить Live Server

**\*\*Код:\*\***

```
```html
<!-- минимальный пример -->
```

Скриншоты: [если нужно]

VS Code версия: 1.XX.X

ОС: Windows 10 Pro 64-bit

Расширения: Live Server, Prettier, etc.

text

🎯 ЧЕК-ЛИСТ "ПОИСК И РЕШЕНИЕ ПРОБЛЕМ"

Когда возникает проблема:

1. [] **Сохранили ли файл?** (`Ctrl+S`)
2. [] **Проверили в браузере?** (F5 или Ctrl+R)
3. [] **Посмотрели консоль браузера?** (F12 → Console)
4. [] **Перезагрузили VS Code?** (окно или `Ctrl+Shift+P` → Reload)
5. [] **Проверили пути к файлам?**
6. [] **Убедились в кодировке UTF-8?**
7. [] **Закрыли лишние вкладки?**
8. [] **Отключили расширения?** (`code --disable-extensions`)

Если не помогло:

1. [] **Поискали ошибку в Google** (точная формулировка)
2. [] **Проверили Stack Overflow**

3. [] **Спросили в сообществе**
 4. [] **Создали минимальный пример для воспроизведения**
 5. [] **Обратились к документации**
-

✨ ПРОФИЛАКТИКА ПРОБЛЕМ

- ### **Ежедневные привычки:**
1. **Сохраняйте часто** (`Ctrl+S` каждые 5 минут)
 2. **Используйте Git** для контроля версий
 3. **Делайте бэкапы** важных проектов
 4. **Обновляйте VS Code** раз в месяц
 5. **Чистите расширения** раз в квартал

Настройки для стабильности:

```
```json
{
 // Автосохранение
 "files.autoSave": "afterDelay",
 "files.autoSaveDelay": 1000,

 // Бэкапы
 "files.defaultLanguage": "html",

 // Производительность
 "editor.minimap.enabled": false,
 "workbench.enableExperiments": false,

 // Кодировка
 "files.encoding": "utf8",
 "files.autoGuessEncoding": true
}
```

---

## ❤️ ПОСЛЕДНИЙ СОВЕТ

**Дорогой друг!** Помните, что каждая решенная проблема делает вас сильнее как разработчика. Не расстраивайтесь, если что-то не получается — все проходили через это!

**Золотые правила:**

**Не паникуйте** — все проблемы решаемы

**Сохраняйтесь чаще** — Ctrl+S должен стать рефлексом

**Ищите причину** — не просто "не работает", а "почему не работает"

**Учитесь на ошибках** — каждая ошибка это урок

**Делитесь знаниями** — помогите другому, когда решите проблему

**И помните:** Даже senior-разработчики гуглят ошибки и сталкиваются с проблемами. Главное — знать, как их решать!

**Удачи в освоении, и пусть ваши баги будут мелкими, а решения — быстрыми!** 

## Приложение Г: Ресурсы для дальнейшего обучения

### Дорогой друг!

Поздравляю с освоением основ VS Code! Теперь у вас есть прочный фундамент. Но обучение — это бесконечный путь. В этом приложении я собрал лучшие ресурсы, которые помогут вам расти как веб-мастеру.

**Помните:** Не пытайтесь изучить всё сразу. Выбирайте по 1-2 ресурса в каждом разделе и двигайтесь постепенно.

### ДОРОЖНАЯ КАРТА ОБУЧЕНИЯ

#### Этап 1: Основы (1-3 месяца)

text

##### HTML5

- Семантическая верстка
- Формы и таблицы
- Мультимедиа

##### CSS3

- Селекторы и каскадность
- Box model
- Flexbox
- Grid
- Адаптивный дизайн
- Анимации

##### Инструменты

- VS Code
- Git и GitHub
- DevTools браузера
- Figma (базово)

#### Этап 2: Углубление (3-6 месяцев)

text

##### JavaScript основы

- Синтаксис
- DOM манипуляции
- События

## └── AJAX и Fetch API

- ✓ Препроцессоры и сборщики
  - └── SASS/SCSS
  - └── Gulp/Webpack (базово)
  - └── Babel

- ✓ Доступность (a11y)
  - └── ARIA атрибуты
  - └── Семантическая верстка
  - └── Тестирование скринридерами

## Этап 3: Современный стек (6-12 месяцев)

text

- ✓ Фреймворки (выберите один)
  - └── React
  - └── Vue.js
  - └── Angular

- ✓ State management
  - └── Redux (для React)
  - └── Vuex (для Vue)
  - └── RxJS

- ✓ TypeScript
  - └── Типизация
  - └── Интерфейсы
  - └── Generics

## Этап 4: Профессионализм (1-2 года)

text

- ✓ Оптимизация
  - └── Производительность
  - └── Безопасность
  - └── SEO

- ✓ Тестирование
  - └── Unit tests (Jest)
  - └── E2E tests (Cypress)
  - └── Интеграционные тесты

- ✓ DevOps основы
  - CI/CD
  - Docker (базово)
  - Основы серверов

## Ресурсы для дорожной карты:

- [Frontend Developer Roadmap](#) — интерактивная карта
  - [Developer Roadmap](#) — на GitHub
  - [Собственная карта](#) — создайте свою в Miro
- 

## ⌚ СОВЕТЫ ДЛЯ ЭФФЕКТИВНОГО ОБУЧЕНИЯ

### Метод 30 минут в день:

- 15 минут теории — статья, видео, документация
- 15 минут практики — повторите, напишите свой вариант
- Ежедневно — лучше понемногу каждый день, чем много раз в неделю

### Правило 80/20 в обучении:

- 80% времени — **практика**
  - 20% времени — **теория**
- Сразу применяйте новые знания в проектах

### Создайте портфолио:

- Лендинг — о себе или вымышленной компании
- Интернет-магазин — каталог товаров, корзина
- Соц. сеть/блог — авторизация, посты, комментарии
- Админ-панель — таблицы, фильтры, статистика

### Участвуйте в open-source:

- Ищите проекты с меткой `good-first-issue`
- Начните с документации или перевода
- Делайте маленькие PR (Pull Requests)
- Изучайте код других разработчиков

### Ведите блог обучения:

Пишите о том, что изучили  
Объясняйте сложное простыми словами  
Создавайте туториалы  
Это лучшее закрепление знаний

---

## ПЛАН НА ПЕРВЫЕ 90 ДНЕЙ

### Дни 1-30: Основы

text

Неделя 1-2:

- HTML: структура, теги, семантика
- CSS: селекторы, цвета, шрифты, отступы
- Проект: простой лендинг

Неделя 3-4:

- CSS: Flexbox, позиционирование
- Адаптивная верстка
- Проект: блог с адаптивным дизайном

### Дни 31-60: Практика

text

Неделя 5-6:

- CSS Grid, анимации
- Препроцессор SASS
- Проект: галерея с grid и анимациями

Неделя 7-8:

- JavaScript основы
- DOM манипуляции
- Проект: интерактивный список дел

### Дни 61-90: Реальные задачи

text

Неделя 9-10:

- Формы, валидация
- Работа с API
- Проект: погодное приложение

Неделя 11-12:

- Git и GitHub
  - Оптимизация
  - Финальный проект: полноценный сайт
- 

## МОТИВАЦИЯ И ПОДДЕРЖКА

**Когда опускаются руки:**

**Вспомните, зачем начали** — запишите свою цель на видном месте  
**Разбейте на маленькие шаги** — не "выучить JS", а "сегодня разобраться с функциями"  
**Отмечайте прогресс** — ведите дневник достижений  
**Ищите единомышленников** — учиться вместе веселее

**Истории успеха:**

Почитайте истории людей, которые начали учиться после 40-50 лет  
Посмотрите интервью с self-taught разработчиками  
Участвуйте в митапах и конференциях (много онлайн)

**Полезные привычки:**

**Утренний код** — 30 минут до работы  
**Фокус-сессии** — 25 минут работы, 5 минут перерыв (метод Помодоро)  
**Код-ревью** — попросите посмотреть ваш код  
**Преподавание** — объясните изученное другу или запишите видео

---

## БОНУС: МОИ ЛЮБИМЫЕ РЕСУРСЫ

**Для вдохновения:**

[Awwwards](#) — лучшие сайты  
[Dribbble](#) — дизайн интерфейсов  
[CodePen Explore](#) — креативные примеры кода

**Для решения проблем:**

[OverAPI](#) — шпаргалки по всем языкам  
[Devhints](#) — быстрые справочники

[CanInclude](#) — что может быть внутри тега

Для развлечения:

[CSS Battle](#) — игра на CSS

[Codepen Challenges](#) — ежемесячные челленджи

[Advent of Code](#) — новогодние задачки

---

## ЗАКЛЮЧИТЕЛЬНОЕ ПИСЬМО

Дорогой друг!

Вы проделали огромный путь — от первого знакомства с VS Code до создания полноценных веб-страниц. Гордитесь собой!

Помните:

**Обучение — это марафон, а не спринт** — не торопитесь, наслаждайтесь процессом

**Ошибаться — это нормально** — каждая ошибка делает вас лучше

**Сообщество — ваша сила** — не бойтесь спрашивать и помогать другим

**Практика — ключ к мастерству** — код, код и еще раз код

**Ваш путь только начинается.** С каждым новым проектом вы будете становиться увереннее, с каждой решенной проблемой — мудрее.

**Самый важный ресурс — это вы.** Ваше упорство, любознательность и готовность учиться.

**Держите эту шпаргалку под рукой, возвращайтесь к ней, когда нужно. И помните — я верю в вас!**

**Удачи в освоении веб-разработки! Ваше путешествие только начинается, и впереди столько интересного!** 

**С уважением и верой в ваш успех,**

Ваш гид в мире веб-разработки

---

**P.S.** Сохраните эту шпаргалку в закладки. Возвращайтесь к ней, когда будет нужно вдохновение или направление. И не забывайте — лучший способ отблагодарить меня — это помочь следующему начинающему разработчику, когда станете опытнее! 