Николай Прохоренок
Владимир Дронов

PRO
ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

JavaScript и Node.js

для веб-разработчиков

Итераторы и генераторы
Модули
Класс Promise
Работа с файловой системой
Программирование веб-серверов
Объектная модель документа
AJAX и Fetch API
Webpack
Visual Studio Code

Материалы на www.bhv.ru

bhv®

Николай Прохоренок
Владимир Дронов

JavaScript и Node.js

для веб-разработчиков

Санкт-Петербург
«БХВ-Петербург»
2022

УДК 004.438JavaScript
ББК 32.973.26-018.1
П84

Прохоренок, Н. А.

П84 JavaScript и Node.js для веб-разработчиков / Н. А. Прохоренок, В. А. Дронов. — СПб.: БХВ-Петербург, 2022. — 768 с.: ил. — (Профессиональное программирование)

ISBN 978-5-9775-6847-0

Книга рассказывает о языке программирования JavaScript, разработке на нем как программ общего назначения, выполняющихся в среде Node.js, так и скриптов для веб-страниц. Даны основы JavaScript: типы данных, операторы, работа с числами, строками, датой и временем, массивами, функции, классы (как старого, так и нового синтаксиса), итераторы, генераторы и класс Promise. Объяснена работа с отладчиком, встроенным в редактор Visual Studio Code. Рассказано о модулях, средствах для работы с файловой системой и программирования веб-серверов. Описана объектная модель документа. Рассмотрены средства для работы с элементами веб-страницы, самой страницей и браузером и технология AJAX (в том числе Fetch API), а также готовые программные пакеты для разработки веб-сайтов, в частности Webpack.

Электронный архив на сайте издательства содержит коды всех пронумерованных листингов.

Для веб-разработчиков

УДК 004.438JavaScript
ББК 32.973.26-018.1

Группа подготовки издания:

Руководитель проекта	*Евгений Рыбаков*
Зав. редакцией	*Людмила Гауль*
Компьютерная верстка	*Ольги Сергиенко*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Карины Соловьевой*

"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

ISBN 978-5-9775-6847-0

Оглавление

Введение

JavaScript (также известен как *ECMAScript*) — язык программирования, созданный для написания скриптов — программ, которые встраиваются в веб-страницы и выполняются непосредственно веб-браузером. Самый простой скрипт может в ответ на возникновение какого-либо события (например, наведения мыши на определенный элемент страницы) выводить на странице заданное сообщение (скажем, поясняющий текст). Более сложный скрипт способен прочитать значения, занесенные пользователем в форму, чтобы проверить их на корректность перед отправкой на сервер или даже обработать самостоятельно и тут же вывести результат. И наконец, высший пилотаж JavaScript-программирования — программная подгрузка данных с сервера для формирования на их основе содержимого целой страницы.

JavaScript оказался настолько удачным языком, что остальные программисты стали завидовать разработчикам веб-страниц. И вот в 2009 году появилась первая версия программной платформы Node.js, которая была создана на основе интерпретатора (исполняющей среды) языка JavaScript, встроенного в популярный браузер Google Chrome, и позволяла исполнять JavaScript-программы без веб-браузера. С появлением Node.js появилась возможность писать на JavaScript программы практически любого назначения: инструментальные и системные утилиты, серверы и даже приложения с графическим интерфейсом (правда, для этого понадобятся дополнительные библиотеки).

Предлагаемая вниманию читателей книга рассказывает о языке JavaScript, его применении для написания как программ, работающих под управлением Node.js, так и скриптов, выполняемых веб-браузерами. По сути, это всеобъемлющее руководство для JavaScript-программистов, желающих знать об этом языке все.

Структура книги

- В *главе 1* описывается редактор программного кода Visual Studio Code, который предназначен именно для программистов и которым мы будем пользоваться на протяжении всего обучения. Редактор корректно работает с кодировкой UTF-8, имеет подсветку синтаксиса языков HTML, CSS, JavaScript и др., выводит различные полезные подсказки и, вообще, всячески помогает разработчику.

- *Глава 2* является вводной. Мы установим платформу Node.js, создадим и запустим в ней первую программу — из командной строки и из редактора Visual Studio Code. Кроме того, вкратце разберемся со структурой программы, научимся выводить результаты ее работы и получать данные от пользователя.
- В *главе 3* мы познакомимся с переменными и типами данных, поддерживаемыми JavaScript.
- В *главе 4* мы рассмотрим операторы, выполняющие различные действия с данными, в частности изучим операторы ветвления и циклы, позволяющие изменять порядок выполнения инструкций в программе.
- *Глава 5* полностью посвящена работе с числами. Мы узнаем, какие числовые типы данных поддерживает JavaScript, научимся применять различные функции, генерировать случайные числа и др.
- *Глава 6* познакомит с массивами и множествами JavaScript. Мы научимся создавать массивы, одно- и многомерные, перебирать элементы массива, сортировать, выполнять поиск значений в массивах и др.
- *Глава 7* полностью посвящена работе со строками.
- В *главе 8* мы рассмотрим регулярные выражения, которые очень пригодятся при выполнении сложного поиска или замены в строках.
- *Глава 9* рассказывает о работе со значениями даты и времени (их еще называют временны́ми отметками).
- В *главе 10* мы научимся создавать пользовательские функции — фрагменты произвольного программного кода, которые можно вызывать на выполнение в любом месте программы.
- *Глава 11* познакомит нас с *объектно-ориентированным программированием* (*ООП*) — еще одним способом многократного использования кода. В отличие от функций, ООП позволяет описать предметы реального мира в виде объектов и организовать связи между этими объектами.
- В *главе 12* мы научимся размещать код в отдельных файлах — *модулях*, выполнять экспорт идентификаторов из одного модуля и импортировать эти идентификаторы для использования в других модулях.
- В *главе 13* мы рассмотрим способы поиска ошибок в программе и научимся отлаживать код в отладчике, встроенном в Visual Studio Code.
- *Глава 14* познакомит нас с асинхронным выполнением программного кода и событиями Node.js.
- В *главе 15* мы рассмотрим класс `Buffer`, который позволит работать с отдельными байтами и выполнять преобразование кодировок.
- *Главы 16*, *17* и *18* научат работать с файлами и каталогами, читать и записывать файлы в различных форматах.
- В *главе 19* мы напишем и запустим собственный веб-сервер.

- В *главе 20* мы оставим в покое Node.js и познакомимся с особенностями исполнения JavaScript-скриптов в веб-браузерах.
- *Глава 21* рассматривает объектную модель документа (DOM) веб-браузера, благодаря которой мы сможем получить из скрипта доступ к странице и ее элементам и управлять ими: менять содержимое элементов, добавлять, изменять и удалять элементы страницы, работать со стилями CSS и др.
- *Глава 22* познакомит нас с многочисленными событиями, возникающими в веб-страницах и их элементах, и обработкой этих событий.
- В *главе 23* мы научимся программно обрабатывать данные, занесенные пользователем в элементы формы.
- *Глава 24* познакомит нас с программным рисованием на холсте, реализуемым тегом `<canvas>`.
- В *главе 25* рассматривается технология AJAX, предназначенная для фоновой загрузки произвольных данных с сервера, и два реализующих ее программных инструмента.
- И наконец, в *главе 26* мы рассмотрим готовые программные пакеты для Node.js, которые могут помочь при разработке веб-сайтов. Например, пакет sass транслирует таблицы стилей, написанные на языках SCSS и SASS, в поддерживаемый браузерами CSS. А пакет Webpack может оказаться очень полезным при сборке сложных веб-проектов из разнородных исходных файлов.

Для полного понимания материала книги от читателя потребуются знания HTML и CSS в объеме первых двух глав книги "HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера".

Все листинги из этой книги вы найдете в файле Listings.doc, электронный архив с которым можно загрузить с FTP-сервера издательства "БХВ" по ссылке: **ftp://ftp.bhv.ru/9785977568470.zip** или со страницы книги на сайте **https://bhv.ru** (см. *приложение*).

Типографские соглашения

В книге будут часто приводиться различные языковые конструкции, применяемые в программировании. Для наглядности при их написании приняты следующие типографские соглашения (в реальном коде они недействительны):

- HTML-, CSS-, JavaScript- и PHP-код набран моноширинным шрифтом:

```
for (var i = 0; i < 5; i++) {
   console.log(i);
}
console.log('i =', i);
```

- в угловые скобки <> заключены наименования различных языковых конструкций. В код программы, разумеется, должны быть подставлены реальные конструкции. Например:

```
const <Имя константы> = <Значение константы>
```

Здесь вместо фраз `<Имя константы>` и `<Значение константы>` должны быть подставлены реальные имя и значение;

- в квадратные скобки `[]` заключены необязательные фрагменты кода:

```
process.exit([<Код завершения>]);
```

Здесь параметр `<Код завершения>` может указываться, а может и отсутствовать;

- вертикальной чертой `|` разделены доступные для выбора языковые конструкции, из которых в код можно подставить лишь одну:

```
var|let <Переменная>
```

Здесь можно подставить либо ключевое слово `var`, либо слово `let`;

- слишком длинные, не помещающиеся на одной строке книги фрагменты кода разделены на несколько строк, и в местах разрывов поставлены знаки ↳:

```
node-sass -o dist/css/ --include-path node_modules/ ↳
scss/main.scss
```

Приведенный здесь код хотя и разбит на две строки, но должен быть набран в одну. Символ ↳ при этом следует удалить;

- троеточием . . . помечены фрагменты кода, пропущенные ради сокращения объема текста книги:

```
class SomeClass {
    *#method1() { . . . }
    static *staticMethod2() { . . . }
    static *#staticMethod3() { . . . }
}
```

Здесь весь код внутри фигурных скобок пропущен.

> ***НЕ ЗАБЫВАЙТЕ!***
>
> Все приведенные здесь типографские соглашения имеют смысл только в примерах написания языковых конструкций. В реальном программном коде они приведут к возникновению ошибки.

Желаем приятного изучения и надеемся, что книга поможет вам написать сначала простые, а потом и более сложные веб-приложения.

ГЛАВА 1

Редактор Visual Studio Code

Для создания файлов с кодом из книги можно воспользоваться любым текстовым редактором, например Блокнотом. Однако мы будем работать с многобайтовой кодировкой UTF-8, а Блокнот при сохранении в этой кодировке добавляет *метку порядка байтов* (сокращенно *BOM*), которая может стать причиной программных ошибок. Чтобы избежать этого, в качестве редактора кода на протяжении всего обучения воспользуемся программой *Visual Studio Code* (далее *VS Code*). Она также позволяет корректно работать с однобайтовой кодировкой Windows-1251 и имеет подсветку синтаксиса языков HTML, CSS, JavaScript и др.

Для быстрого редактирования файлов советуем дополнительно установить редактор Notepad++. Скачать Notepad++ можно абсолютно бесплатно со страницы **https://notepad-plus-plus.org/**. Из двух вариантов (архив и инсталлятор) советуем выбрать именно инсталлятор, т. к. при установке можно будет указать язык интерфейса программы. Установка Notepad++ предельно проста и в пояснениях не нуждается.

1.1. Установка VS Code

Для установки VS Code переходим на сайт **https://code.visualstudio.com/**, прокручиваем страницу в самый низ, в раздел загрузок, и скачиваем установщик **User Installer 64 bit** (для 64-разрядной редакции Windows) или **User Installer 32 bit** (для ее 32-разрядной редакции). Файл установщика будет иметь имя VSCODEUSERSETUP-X64-1.55.2.EXE или VSCodeUserSetup-ia32-1.55.2.exe соответственно.

Запускаем программу установки. На первом шаге принимаем лицензионное соглашение и нажимаем кнопку **Далее** (рис. 1.1). На втором шаге при желании можно изменить каталог, в который будет установлена программа, или оставить каталог по умолчанию (рис. 1.2). Нажимаем кнопку **Далее**. На следующем шаге имеется возможность изменить имя группы, которая будет создана в меню **Пуск** после установки программы, также можно оставить имя группы по умолчанию (рис. 1.3). Нажимаем кнопку **Далее**. На следующем шаге устанавливаем все флажки, кроме **Зарегистрировать Code в качестве редактора для поддерживаемых типов файлов** (рис. 1.4). Для этой цели лучше использовать более простой редактор,

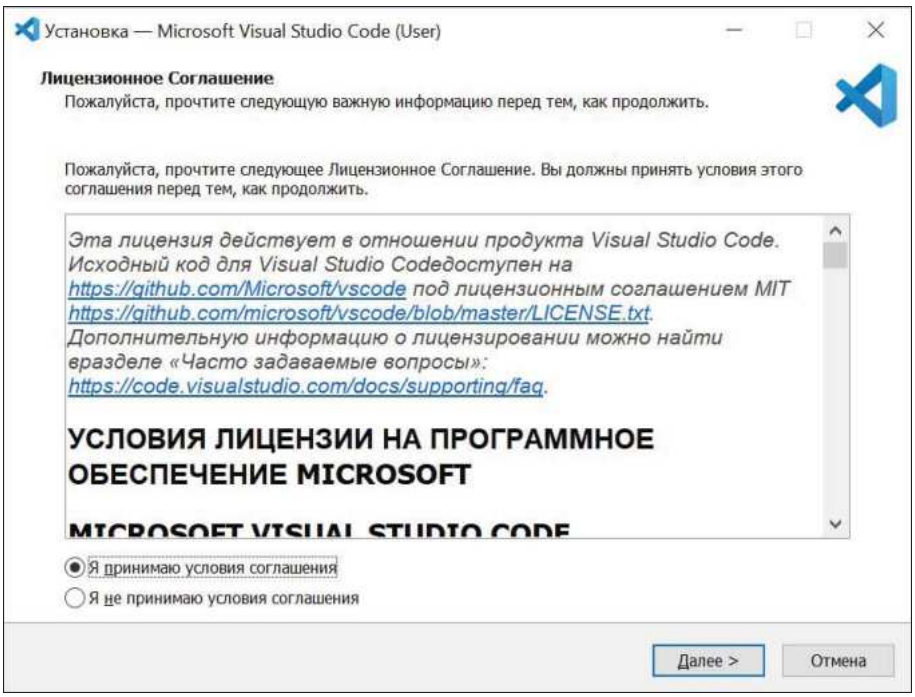

Рис. 1.1. Установка VS Code: шаг 1

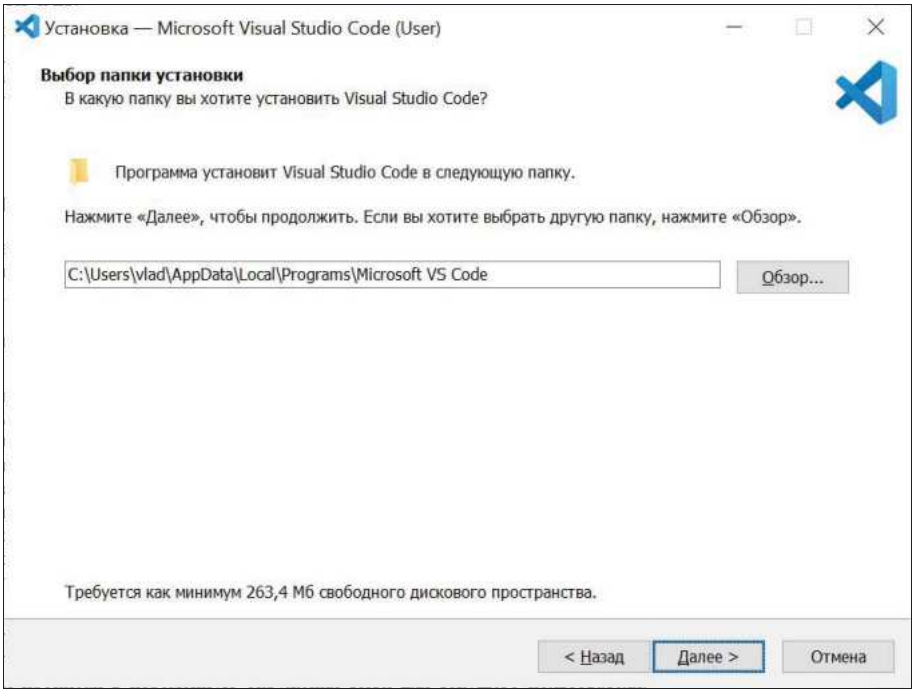

Рис. 1.2. Установка VS Code: шаг 2

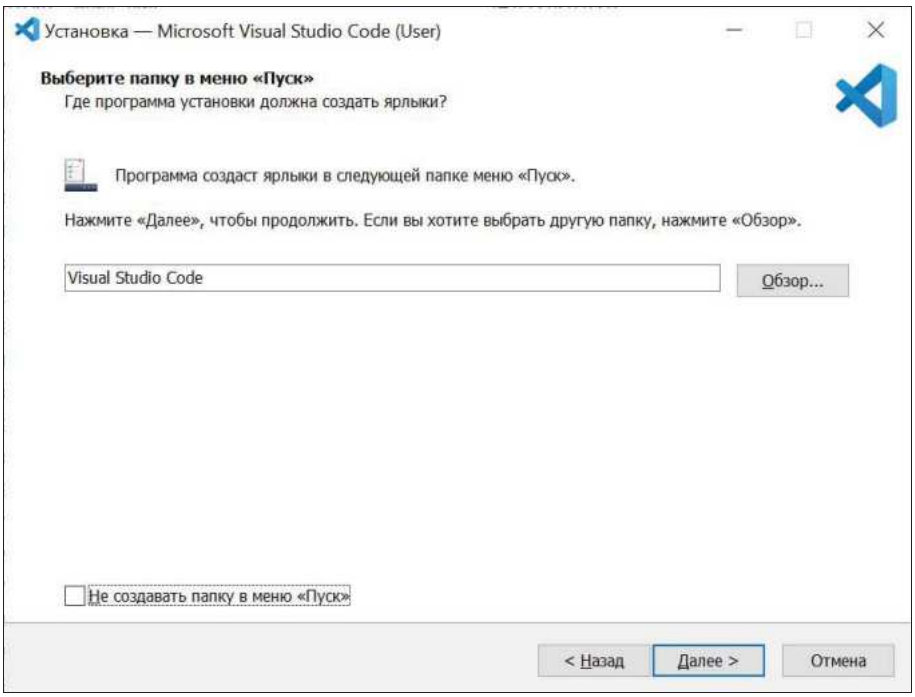

Рис. 1.3. Установка VS Code: шаг 3

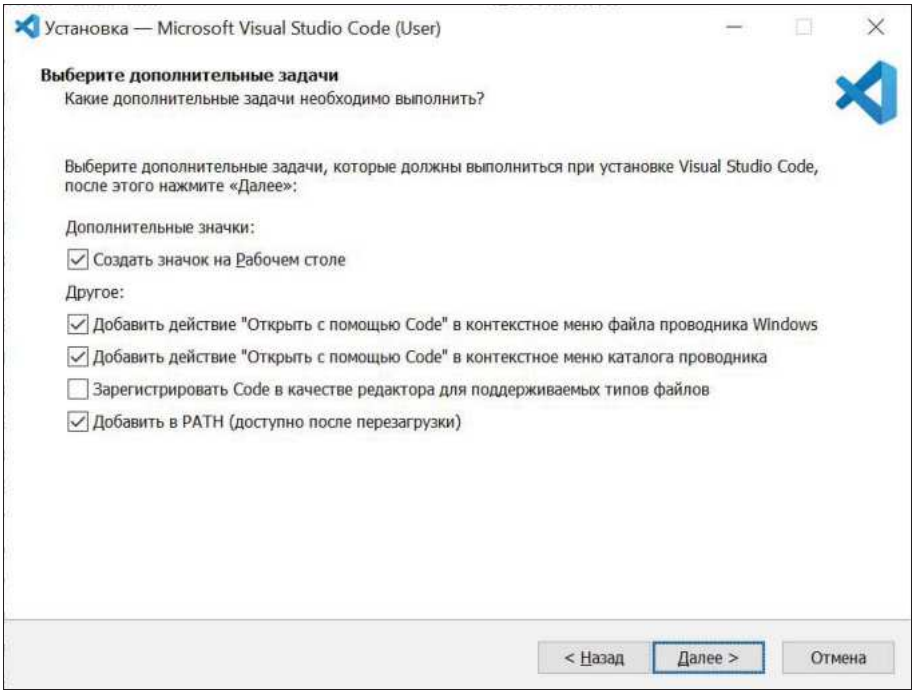

Рис. 1.4. Установка VS Code: шаг 4

наподобие Notepad++. Нажимаем кнопку **Далее**. На очередном шаге проверяем настройки и нажимаем кнопку **Установить** (рис. 1.5) для запуска процесса установки. На последнем шаге, по окончании установки, нажимаем кнопку **Завершить** (рис. 1.6).

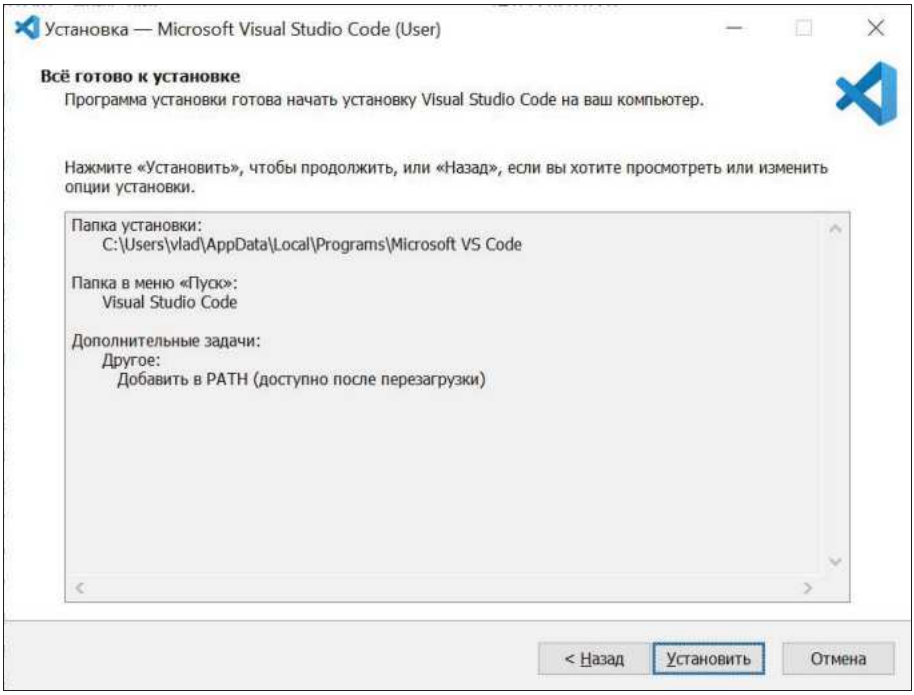

Рис. 1.5. Установка VS Code: шаг 5

По умолчанию, если на шаге 2 (см. рис. 1.2) не был выбран другой путь, редактор устанавливается в каталог C:\Users\<Имя пользователя>\AppData\Local\Programs\Microsoft VS Code. Путь до вложенного каталога bin записывается в переменную окружения `PATH` для текущего пользователя. Кроме того, на Рабочем столе создается ярлык для запуска редактора. В меню **Пуск** добавляется группа **Visual Studio Code** (имя по умолчанию, может быть изменено на шаге 3, см. рис. 1.3) с одноименным пунктом. В контекстное меню Проводника Windows добавляются пункты для открытия в редакторе файла или целого каталога. Чтобы открыть файл, щелкаем правой кнопкой мыши на значке файла и из контекстного меню выбираем пункт **Открыть с помощью Code**. Аналогичный пункт будет при щелчке правой кнопкой мыши на значке каталога или внутри свободной области каталога в Проводнике Windows.

Если на последнем шаге был установлен флажок **Запустить Visual Studio Code** (см. рис. 1.6), то редактор автоматически запустится. Если после запуска отобразилось черное окно, и долго ничего не меняется, нужно выполнить дополнительную настройку. Закрываем редактор. На Рабочем столе находим ярлык для запуска редактора и щелкаем на нем правой кнопкой мыши. Из контекстного меню выбираем

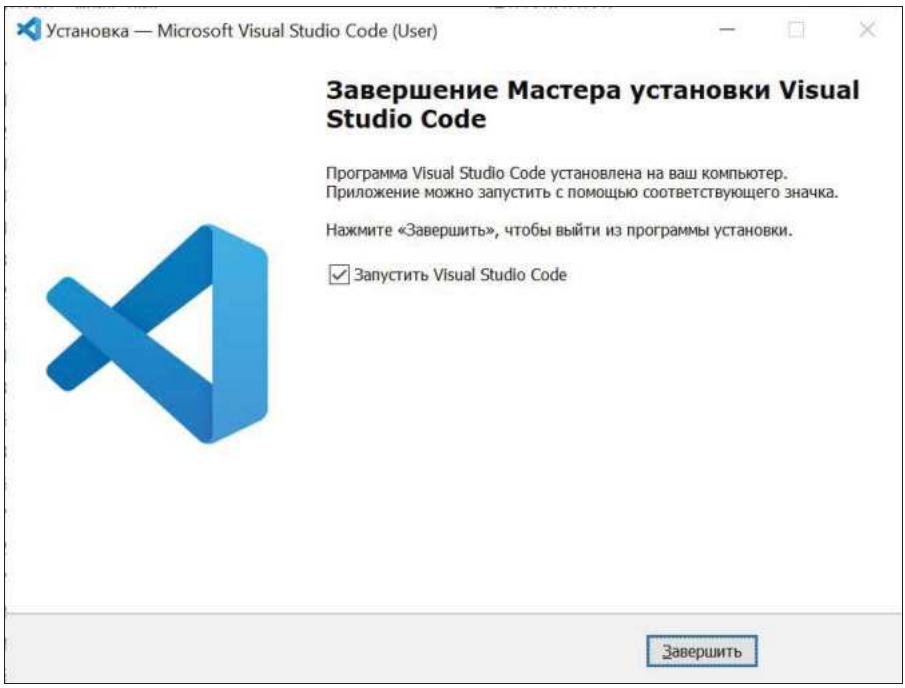

Рис. 1.6. Установка VS Code: шаг 6

пункт **Свойства**. В открывшемся окне на вкладке **Ярлык** в конец содержимого поля **Объект** добавляем пробел и флаг `--disable-gpu`:

```
"C:\Users\<Имя пользователя>\AppData\Local\Programs\Microsoft VS Code\
Code.exe" --disable-gpu
```

Вместо `<Имя пользователя>` должно быть указано имя пользователя вашего компьютера. Сохраняем изменения и запускаем редактор двойным щелчком на ярлыке. Если редактор успешно запустился, то мы можем продолжить изучение, но предварительно произведем его настройку.

1.2. Настройка редактора

По умолчанию редактор не поддерживает русский язык. Добавить поддержку русского языка можно, установив расширение Russian Language Pack for Visual Studio Code. Для этого в меню **View** выбираем пункт **Extensions** или нажимаем комбинацию клавиш <Shift>+<Ctrl>+<X>. В результате в левой части окна редактора отобразится панель **Extensions**. В строке поиска вводим имя искомого расширения: "Russian Language Pack for Visual Studio Code". Как только найденное расширение появится в списке, устанавливаем его, нажав небольшую кнопку **Install**, которая находится в правом нижнем углу пункта списка, представляющего это расширение. После установки расширения перезапускаем редактор.

Дополнительно устанавливаем расширение Auto Complete Tag, которое автоматически установит еще два расширения: Auto Rename Tag и Auto Close Tag. Эти рас-

ширения упростят работу с HTML-тегами (например, при переименовании открывающего тега будет автоматически изменяться содержимое закрывающего тега). Поскольку редактор теперь "говорит" по-русски, для вывода панели **Расширения** следует выбрать одноименный пункт меню **Вид**, а для установки расширения — щелкнуть кнопку **Установить** в пункте списка.

> ***Внимание!***
>
> В дальнейшем будет описываться редакция Visual Studio Code, русифицированная с помощью расширения Russian Language Pack for Visual Studio Code.

> ***На заметку***
>
> Расширения сохраняются в каталоге C:\Users\<Имя пользователя>\.vscode\extensions.

Теперь переходим к настройкам. В меню **Файл** выбираем пункт **Настройки | Параметры** или нажимаем комбинацию клавиш <Ctrl>+<запятая>. Выполняем настройку следующих параметров в разделе **Пользователь** (вводим их названия в поле для поиска):

- **Editor: Font Family** — названия используемых шрифтов через запятую. Рекомендуем значение "Consolas, 'Courier New', monospace".
- **Editor: Font Size** — размер шрифта. Можете подобрать удобное значение, например 16 пунктов.
- **Editor: Tab Size** — число пробелов в табуляции. Вводим значение 3.
- **Editor: Detect Indentation** — сбрасываем флажок, чтобы всегда использовать пробелы вместо символов табуляции.
- **Editor: Mouse Wheel Zoom** — если флажок установлен, изменять масштаб можно с помощью колесика мыши, удерживая нажатой клавишу <Ctrl>.
- **Editor: Word Wrap** — стиль переноса длинных строк. Из списка выбираем пункт **bounded**, включающий автоматический перенос длинных строк.
- **Editor: Word Wrap Column** — количество символов в строке. Должно быть указано значение 80 (задано по умолчанию).
- **Files: Auto Save** — управляет автоматическим сохранением файла. Рекомендуем из списка выбрать пункт **onFocusChange**, при котором файл будет автоматически сохраняться при потере фокуса ввода.
- **Editor: Links** — снимите флажок, если не хотите, чтобы внутри файла выделялись ссылки. С одной стороны, удобно открывать файл переходом по ссылке, с другой — подчеркнутые ссылки мешают.
- **Files: Encoding** — из списка выбираем пункт **utf8**, который обозначает кодировку UTF-8 без BOM (метки порядка байтов).
- **Files: Eol** — задает символ перевода строк. Из списка выбираем пункт **\n**. Чтобы сменить символ перевода строк для текущего файла, в строке состояния щелкаем на значке **CRLF** или **LF**, а затем из списка выбираем нужный пункт.

Точно таким же способом можно задать и другие настройки.

Все изменения настроек в разделе **Пользователь** сохраняются в файле settings.json, который расположен в каталоге C:\Users\<Имя пользователя>\AppData\Roaming\Code\User. Этот файл можно открыть в VS Code, щелкнув расположенную в правой части строки заголовков вкладок кнопку **Открыть параметры**, и изменить вручную.

Изменения настроек в разделе **Рабочая область** записываются в файле settings.json, созданном в каталоге .vscode, вложенном в каталог проекта. Таким образом, можно указывать какие-либо настройки на уровне отдельного проекта.

1.3. Смена цветовой темы, тем значков файлов и продукта

По умолчанию используется темная тема оформления. Для смены темы в меню **Файл** выбираем пункт **Настройки | Цветовая тема** или нажимаем комбинацию клавиш <Ctrl>+<K>, а затем комбинацию <Ctrl>+<T>. В итоге отобразится список установленных тем (рис. 1.7). Доступны темы светлые, темные и с высокой контрастностью. Существует также возможность установить тему как расширение или создать свою собственную тему. При выборе пункта из списка внешний вид редактора сразу изменится. Советуем попробовать выбрать темную тему Monokai.

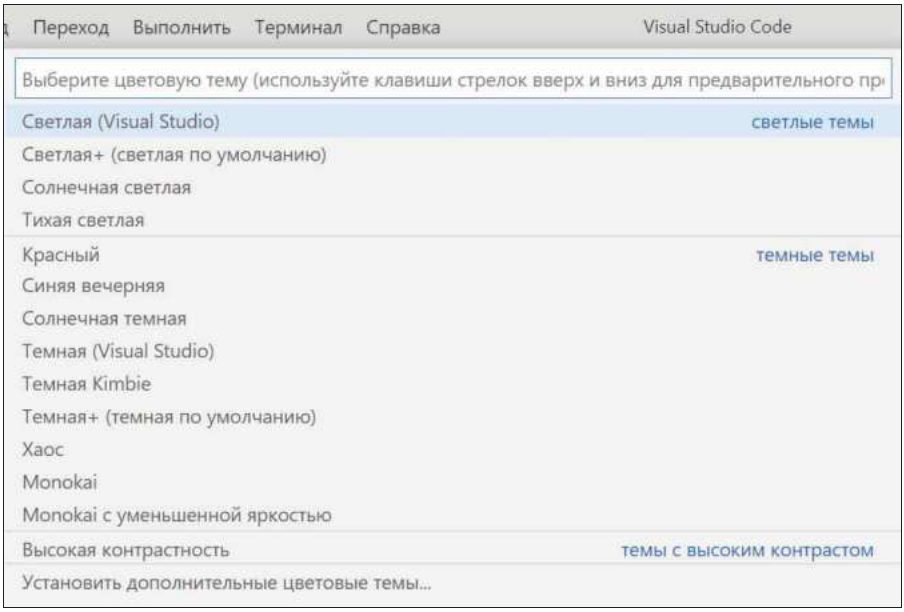

Рис. 1.7. Выбор цветовой темы

Все установленные темы доступны для выбора в настройках редактора. В строке поиска настроек введите **Workbench: Color Theme**, затем выберите тему из раскрывающегося списка (рис. 1.8). Так, при выборе темы "Светлая (Visual Studio)" в файл settings.json будет добавлена следующая строка:

```
"workbench.colorTheme": "Visual Studio Light"
```

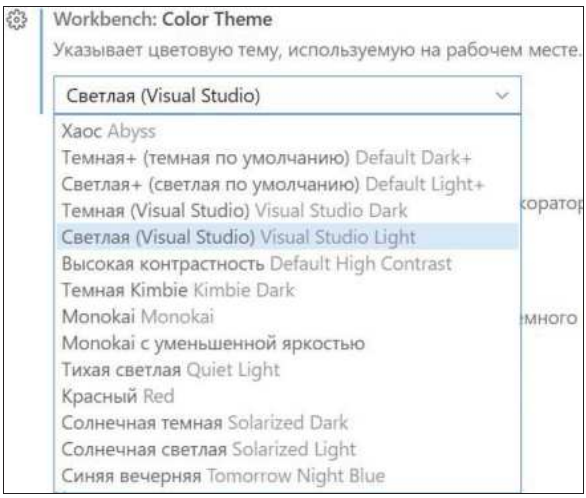

Рис. 1.8. Выбор цветовой темы в настройках редактора

Также имеется возможность изменить значки файлов, выводимые в панели **Проводник** (будет описана далее), задав другую тему значков. Для этого в меню **Файл** выбираем пункт **Настройки | Тема значков файлов**. На экране отобразится список доступных тем значков файлов, аналогичный списку цветовых тем (см. рис. 1.7), в котором можно выбрать нужную тему.

Тему значков файлов можно указать и в настройках редактора. В строке поиска настроек введите **Workbench: Icon Theme** и выберите тему из раскрывающегося списка, аналогичного списку цветовых тем (см. рис. 1.8). Так, при выборе темы "Минимальная (Visual Studio Code)" в файл settings.json будет добавлена следующая строка:

```
"workbench.iconTheme": "vs-minimal"
```

Наконец, можно изменить вид значков, выводящихся на панели действий редактора (будет описана далее), выбрав другую тему. В меню **Файл** выбираем пункт **Настройки | Тема значков продукта**. На экране появится список доступных тем значков продукта, аналогичный списку цветовых тем (см. рис. 1.7), в котором можно выбрать нужную тему. Правда, в этом списке изначально присутствует лишь одна тема, используемая по умолчанию, так что дополнительные темы придется устанавливать в панели **Расширения**.

Тему значков продукта можно задать и в настройках редактора, выполнив поиск по фразе **Workbench: Product Icon Theme** и выбрав тему из раскрывающегося списка. Так, при выборе темы "Fluent Icons" в файл settings.json будет добавлена следующая строка:

```
"workbench.productIconTheme": "fluent-icons"
```

> ***Внимание!***
>
> В книге будут приведены скриншоты, сделанные с использованием цветовой темы "Светлая (Visual Studio)", стандартной темы значков файлов "Seti (Visual Studio Code)" и стандартной темы значков продукта.

1.4. Структура окна редактора

При запуске редактора в центральной части окна отображается вкладка с приветствием (рис. 1.9). На ней расположены ссылки, с помощью которых можно создать файл, открыть каталог или последний проект, получить справочную информацию и т. д. Доступны также советы по установке расширений и настройке редактора. В нижней части вкладки расположен флажок **Отображать страницу приветствия при запуске**. Нам эта вкладка не понадобится, поэтому советуем сбросить флажок. В файл settings.json будет добавлена следующая строка:

```
"workbench.startupEditor": "newUntitledFile"
```

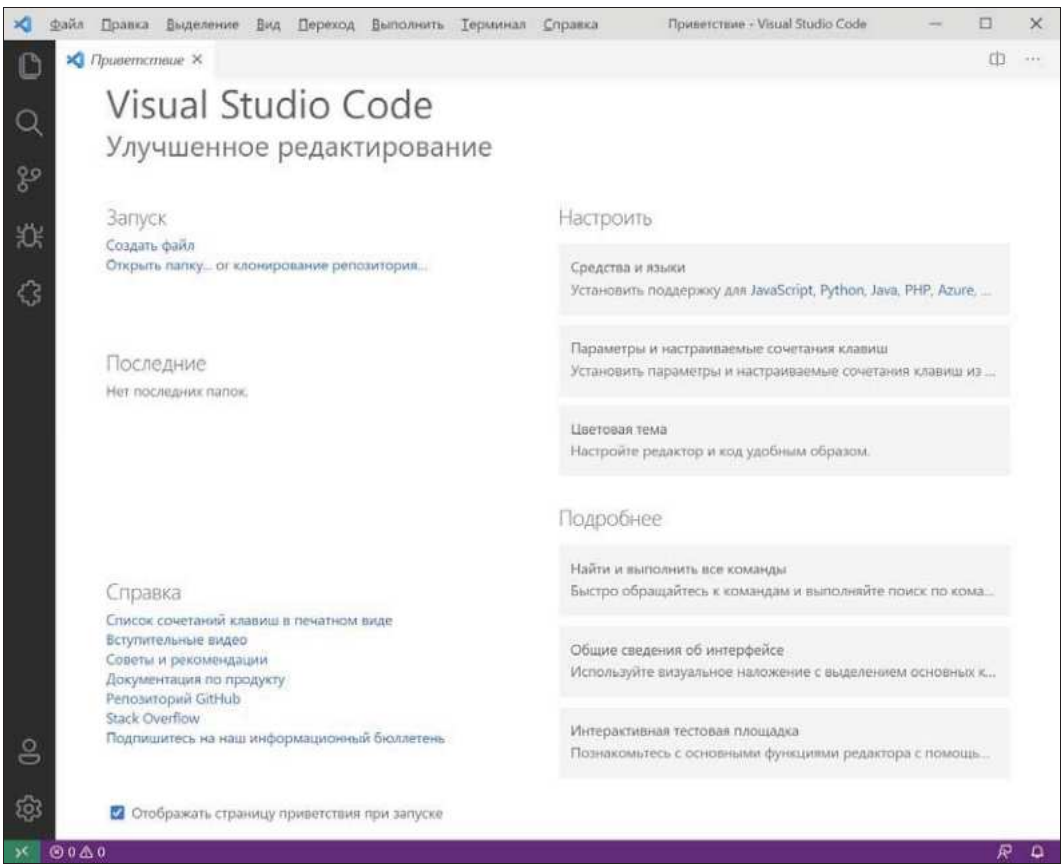

Рис. 1.9. Редактор Visual Studio Code

Значение `newUntitledFile` означает, что при запуске будет создаваться новый файл. Нас это также не устраивает. Переходим в настройки редактора и в строке поиска настроек вводим **Workbench: Startup Editor**. Из списка выбираем пункт **none**. В файл settings.json будет добавлена следующая строка вместо предыдущей:

```
"workbench.startupEditor": "none"
```

В результате при запуске редактор не будет содержать никаких вкладок. Чтобы вывести вкладку с приветствием, в меню **Справка** выбираем пункт **Приветствие** или удаляем из файла settings.json строку с упомянутой ранее настройкой.

1.4.1. Главное меню

В верхней части окна расположена строка меню, имя файла, открытого в активной вкладке, и название редактора, а также стандартные для окон кнопки **Свернуть**, **Развернуть** и **Закрыть**. Кратко рассмотрим структуру главного меню.

- Меню **Файл** — команды для работы с файлами, каталогами и проектами (рис. 1.10). При выборе пункта **Настройки** отобразится меню с командами, позволяющими выполнить настройку редактора (рис. 1.11).

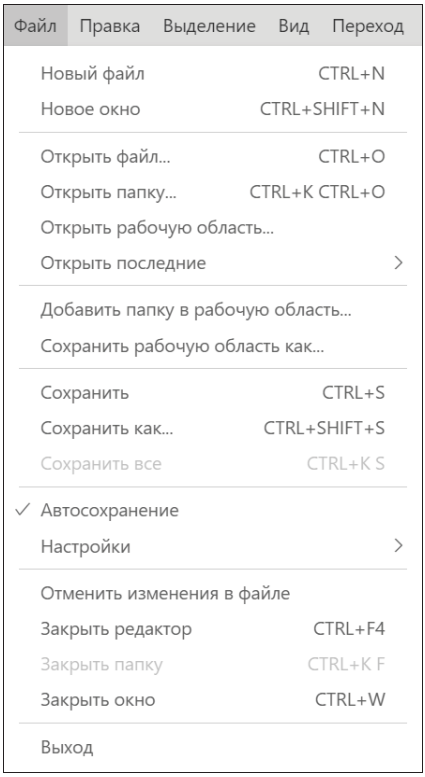

Рис. 1.10. Меню **Файл**

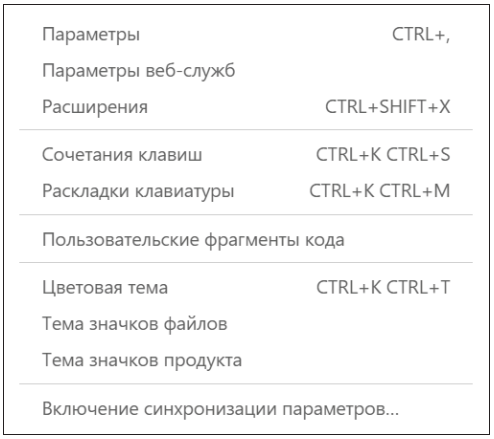

Рис. 1.11. Меню **Файл**, пункт **Настройки**

- Меню **Правка** — команды для редактирования кода (рис. 1.12).
- Меню **Выделение** — команды для работы с выделением и курсорами (рис. 1.13).
- Меню **Вид** — команды для настройки внешнего вида редактора (рис. 1.14). При выборе пункта **Внешний вид** появится меню с командами для управления панелями (рис. 1.15), а при выборе пункта **Макет редактора** — меню с командами, позволяющими разделить содержимое окна редактора на несколько областей (рис. 1.16).