Глава 2

Моя первая программа на JavaScript

В этой главе...

- Организация среды разработки
- Работа с JavaScript-кодом
- Пример простой программы на JavaScript
- О важности комментирования кода

"Секрет неуклонного движения вперед в том, чтобы сделать первый шаг".
Марк Твен

Научиться писать на JavaScript простые программы совсем не сложно. В этой главе мы проведем вас через весь процесс настройки компьютера для написания JavaScript-кода. Кроме того, вы напишете свою первую программу на JavaScript и ознакомитесь с базовым синтаксисом, лежащим в основе всего, что вы будете делать с помощью JavaScript на протяжении своей будущей карьеры программиста.

Настройка среды разработки

Прежде чем приступить к написанию первой программы на JavaScript, очень важно заранее позаботиться о том, чтобы все необходимые для этого инструменты были настроены и находились на своих местах. Мы проведем вас через весь процесс загрузки и установки наших излюбленных средств разработки на JavaScript, которые, естественно, используются и в этой книге. Если вы уже располагаете аналогичными инструментами, которые находите более удобными для себя, то можете смело использовать их. Однако в любом случае мы рекомендуем вам прочитать этот раздел, чтобы узнать о тех соображениях, которыми мы руководствовались при выборе инструментов, и уже после этого принимать решение относительно того, стоит или не стоит их использовать.

Закончив с установкой инструментальных средств, мы поделимся с вами некоторыми советами и подсказками относительно наиболее эффективных способов их использования.

Загрузка и установка браузера Chrome

Для работы с JavaScript мы предпочитаем использовать браузер Google Chrome. Если в повседневной работе вы привыкли пользоваться другим браузером, то, конечно же, ничего предосудительного в этом нет. Во всех браузерах JavaScript работает быстро и корректно. Однако некоторые из описанных в данной книге инструкций специфичны для браузера Google Chrome. Поэтому рекомендуется, чтобы вы в любом случае установили его на своем компьютере, следуя приведенному ниже описанию. Мы используем в этой книге Google Chrome по той причине, что в нем предлагаются великолепные инструменты, значительно облегчающие жизнь программистам, а также потому, что в настоящее время именно этот браузер является лидером по использованию в Интернете. (Это действительно так — он даже более популярен, чем Internet Explorer.)

Если на вашем компьютере браузер Chrome еще не установлен, то выполните процедуру его установки, следуя приведенному ниже описанию.

1. **Выполните переход по адресу `www.google.com/chrome`.**

   Вы будете автоматически перенаправлены на страницу, которая представлена на рис. 2.1.

2. **Щелкните на кнопке Скачать Chrome и следуйте экранным инструкциям.**
3. **Откройте загруженный файл и следуйте дальнейшим инструкциям по установке Chrome.**

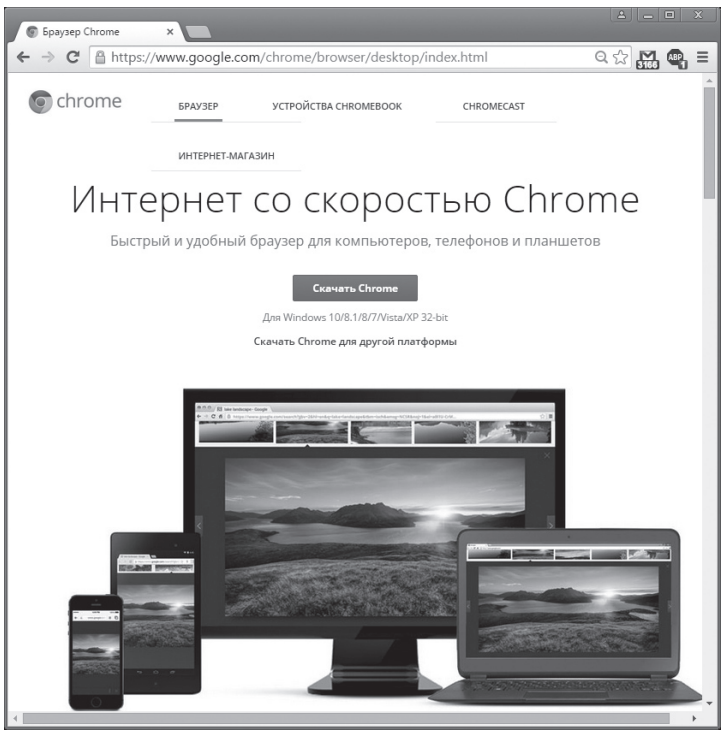


Рис. 2.1. Установка браузера Chrome не составляет большого труда

Теперь у вас имеется движок JavaScript с турбонаддувом!

В Google Chrome парсинг (синтаксический анализ), компиляция и выполнение JavaScript-кода осуществляются с помощью разработанного компанией Google JavaScript-движка V8. В зависимости от того, чьим эталонным тестам доверять больше, можно считать, что Chrome является либо браузером-лидером, либо одним из браузеров, обеспечивающих самое высокое быстродействие при выполнении JavaScript-кода. Основные производители браузеров постоянно соревнуются между собой, стремясь превзойти соперников. Фактически не имеет особого значения, чей браузер в какой-то определенный момент времени оказывается самым быстрым. Важно лишь то, что в результате конкуренции производительность движков JavaScript во всех браузерах стремительно увеличивается на протяжении всех последних лет.

Если вас интересуют фактические данные сравнительного тестирования движков JavaScript в различных браузерах, то посетите сайт `http://arewefastyet.com` (данные тестов там представлены в виде графиков), который поддерживается компанией Mozilla, создавшей браузер Firefox. Этот сайт автоматически проверяет и отображает в графическом виде результаты тестирования производительности движков JavaScript в наиболее популярных моделях браузеров, обновляя данные несколько раз на протяжении дня.

Загрузка и установка редактора исходного кода

Редактор исходного кода, или просто *редактор кода*, — это текстовый редактор с дополнительной функциональностью, упрощающей написание и редактирование программного кода. В качестве такового мы будем использовать редактор Sublime Text.

Учитывая, что спектр выбора редакторов кода очень широк, не исключено, что вы уже пользуетесь одним из них и успели к нему привыкнуть. Вам ничто не мешает и далее продолжать использовать его. Выбор редактора кода определяется исключительно личными предпочтениями, и, наверное, многие из вас уже привыкли работать с каким-то определенным редактором. Если по каким-либо причинам редактор Sublime Text вам не подходит, то просмотрите другие возможные варианты выбора, представленные в табл. 2.1.

Таблица 2.1. Редакторы исходного кода

Название	Адрес для загрузки	Совместимость
Coda	`http://panic.com/coda`	Mac
Aptana	`www.aptana.com`	Mac, Windows
Komodo Edit	`www.activestate.com/komodo-edit/downloads`	Mac, Windows
Dreamweaver	`http://adobe.com/products/dreamweaver.html`	Mac, Windows
Eclipse	`www.eclipse.org`	Mac, Windows
Notepad++	`http://notepad-plus-plus.org`	Windows
TextMate	`http://macromates.com`	Mac
BBEdit	`www.barebones.com/products/bbedit`	Mac
EMacs	`www.gnu.org/software/emacs`	Mac, Windows
TextPad	`www.textpad.com`	Windows
vim	`www.vim.org`	Mac, Windows
Netbeans	`https://netbeans.org`	Mac, Windows

Для этой книги мы выбрали редактор Sublime Text (рис. 2.2), поскольку он популярен среди программистов на JavaScript и предоставляет простой пользовательский интерфейс наряду с большим количеством плагинов, которыми пользователи после приобретения определенных навыков могут воспользоваться для решения более сложных задач программирования.

Рис. 2.2. С виду обычный текстовый редактор, Sublime Text располагает мощными функциональными возможностями

Если вы хотите установить Sublime Text, то следуйте приведенным ниже инструкциям.

1. **Посетите сайт `http://sublimetext.com` и выберите версию, совместимую с установленной на вашем компьютере операционной системой.**
2. **Откройте загруженный файл и следуйте дальнейшим инструкциям по установке Sublime Text.**

Начало работы с Sublime Text

Когда вы впервые откроете приложение Sublime Text, на экране отобразится пустая страница с мерцающим курсором (рис. 2.3).

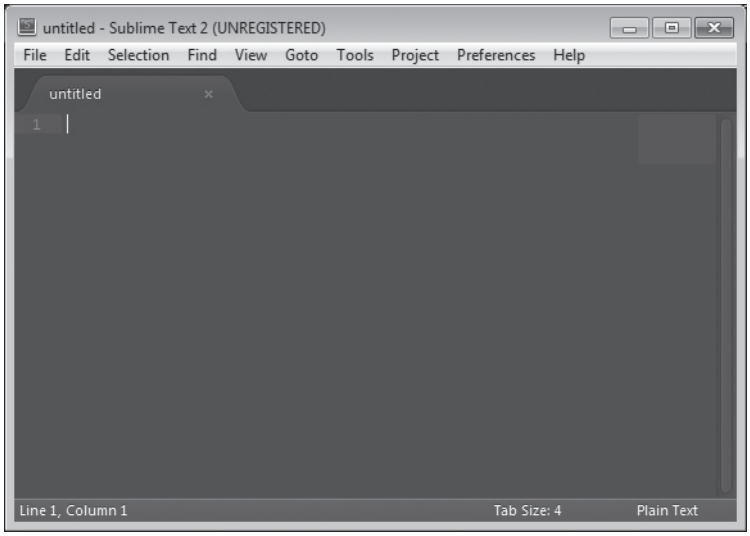


Рис. 2.3. Начальный интерфейс редактора Sublime Text. Как вам эта простота?

Если вы уже использовали Sublime Text, то слева может появиться боковая панель (рис. 2.4), на которой отображаются открытые файлы, а также файлы проекта, если таковой был создан ранее. Боковая панель упрощает работу, и мы рекомендуем держать ее открытой.

Чтобы открыть боковую панель, выберите пункты меню View⇨Sidebar⇨Show Sidebar (Вид⇨Боковая панель⇨Показать).

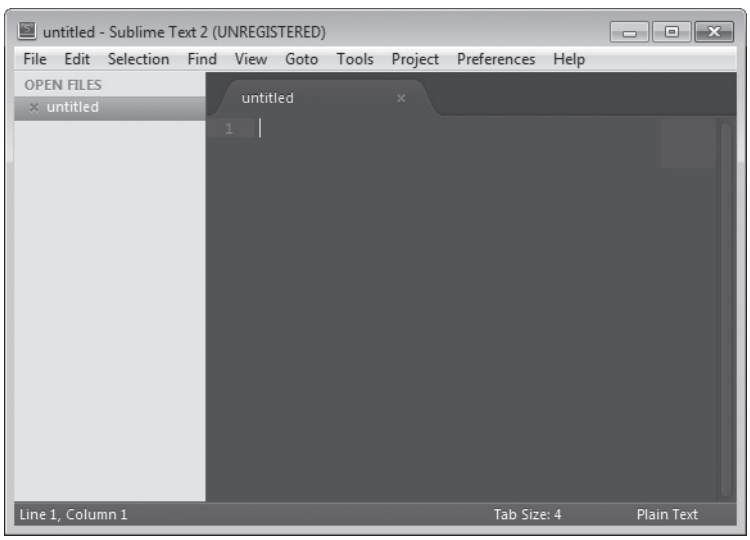


Рис. 2.4. Окно Sublime Text с открытой боковой панелью

Чтобы приступить к работе с вашим первым проектом Sublime Text, следуйте приведенным ниже инструкциям.

1. **Выберите пункты меню File⇨Save As (Файл⇨Сохранить как).**

   В открывшемся диалоговом окне отобразится заданная по умолчанию папка для сохранения файлов. Если предложенный вариант вас удовлетворяет (скорее всего, это будет папка `Documents` (Документы) в случае OS X или папка `My Documents` (Мои документы) в случае Windows), перейдите к выполнению п. 2. В противном случае перейдите в другое расположение на жестком диске компьютера, в котором хотите хранить свои файлы с исходным кодом.

2. **Создайте новую папку и присвойте ей имя.**

3. **Введите в поле Имя файла нужное имя файла и щелкните на кнопке Сохранить.**

   Имя нового файла отобразится в боковой панели, а также на открытой вкладке вместо ее прежнего имени.

4. **Выберите пункты меню Project⇨Save Project As (Проект⇨Сохранить как) и сохраните файл проекта Sublime Text в папке, которую создали перед этим.**

5. **Выберите пункты меню Project⇨Add Folder to Project (Проект⇨Добавить папку в проект), выберите папку, созданную в п. 1, и щелкните на кнопке Open (Открыть).**

   На боковой панели появится новый сворачиваемый список Folders (Папки), в котором отобразится ваша папка вместе с ее содержимым (включая файл проекта и файл `MyFirstProgram.html`), как показано на рис. 2.5.

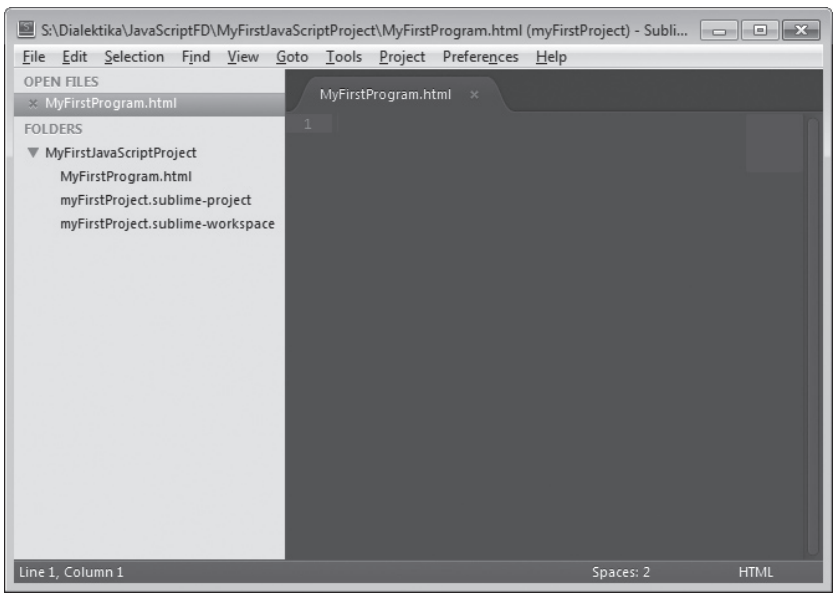


Рис. 2.5. Ваш первый проект Sublime Text подготовлен к работе

Чтобы ваши файлы и папки хранились в организованном виде, рекомендуем придерживаться определенной системы при присвоении им имен. Например, папку из п. 2 можно назвать `MyFirstJavaScriptProject`, файл из п.3 — `MyFirstProgram`, а проект из п. 4 — `myFirstProject`.

Выбор цветовой схемы для подсветки синтаксиса

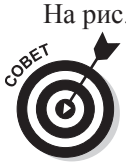
В Sublime Text цветовая подсветка синтаксиса основана на типе кода, который вы пишете, и расширении имени файла. Чтобы увидеть цветовую схему, заданную по умолчанию, введите в только что созданный вами файл код на основе HTML и JavaScript, представленный в листинге 2.1.

JavaScript требует педантичного отношения к себе, в чем вы вскоре сами сможете убедиться. Текст необходимо вводить в точности так, как показано в листинге, строго соблюдая регистр букв, иначе ваш сценарий вообще не будет работать, а если и будет, то неправильно.

Листинг 2.1. Пример HTML-файла, содержащего сценарий JavaScript

```
<!DOCTYPE html>
<html>
<head>
  <title>Привет, HTML!</title>
  <script>
    function countToTen(){
      var count = 0;
      while (count < 10) {
        count++;
        document.getElementById("theCount").innerHTML +=
          count + "<br>";
      }
    }
  </script>
</head>
<body onload="countToTen();">
  <h1>Посчитаем до 10 вместе с JavaScript!</h1>
  <p id="theCount"></p>
</body>
</html>
```

На рис. 2.6 показано, как этот файл отображается в окне Sublime Text.



Если текущая цветовая схема вас не устраивает, то ее можно изменить, выбрав пункты меню Preferences⇨Color Scheme (Установки⇨Цветовая схема) и задав другую схему.

Просмотрите несколько цветовых схем и выберите ту, которая вам больше всего по душе. В этой книге используется цветовая схема Monokai Bright.

Если хотите проверить работу программы, которую только что ввели, то выполните следующие действия.

1. **Сохраните файл, выбрав пункты меню File⇨Save (Файл⇨Сохранить).**
2. **Откройте браузер Chrome и нажмите комбинацию клавиш <Ctrl+O>.**
   Откроется окно `Открыть`.
3. **Перейдите к файлу на своем компьютере и выберите его.**
4. **Щелкните на кнопке Open (Открыть).**
   Содержимое файла отобразится в окне браузера.

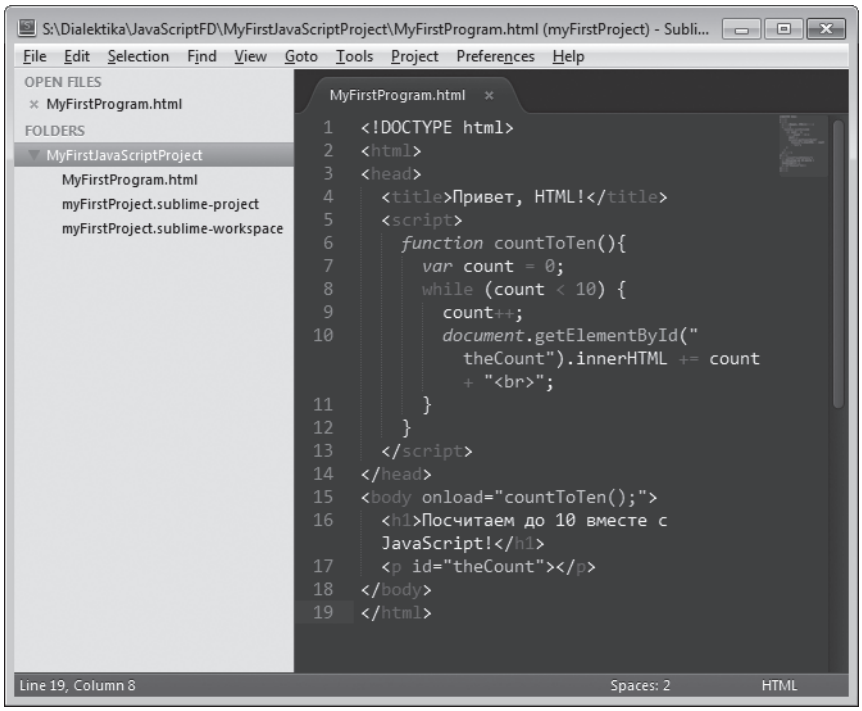


Рис. 2.6. Применение цветовой подсветки кода в окне редактора Sublime Text

Окно вашего браузера должно выглядеть так, как показано на рис. 2.7. Если оно будет иметь другой вид, тщательно проверьте свой код — возможно, вы где-то допустили опечатку. После внесения необходимых изменений не забудьте сохранить файл!

Для сохранения файлов можно также использовать комбинации клавиш <Command+S> (Mac) и <Ctrl+S> (Windows). Когда вы привыкнете с ними работать, они сэкономят вам массу времени.

Полезные комбинации клавиш в Sublime Text

Редактор кода Sublime Text на первый взгляд кажется простым текстовым редактором, но это обманчивое впечатление. Истинным показателем профессионализма программиста является его умение эффективно использовать комбинации клавиш, обеспечивающие значительную экономию времени при редактировании исходного кода.

В редакторе Sublime Text предусмотрено большое количество комбинаций клавиш (так называемых "горячих клавиш"), частичный перечень которых приведен в табл. 2.2. Освойте их, и вскоре вы поразите друзей и коллег тем, как быстро вы работаете.

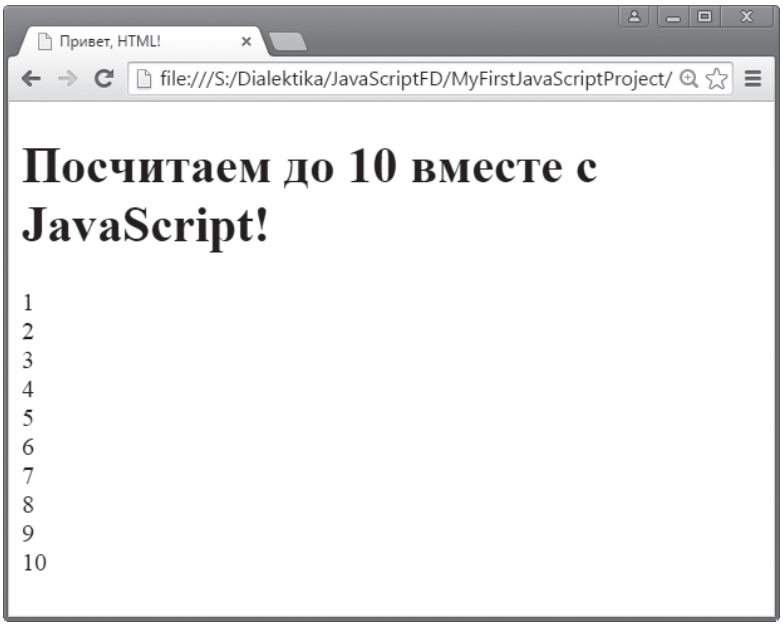


Рис. 2.7. Выполнение простой программы подсчета в Chrome

Таблица 2.2. Часто используемые комбинации клавиш для работы с текстом в редакторе Sublime Text

Mac	Windows	Описание
<Command+X>	<Ctrl+X>	Удалить строку
<Command+Return>	<Ctrl+Enter>	Вставить строку снизу
<Command+]>	<Ctrl+Shift+Enter>	Вставить строку сверху
<Command+Control+↑>	<Ctrl+Shift+↑>	Переместить строку/выделенный объект вверх
<Command+Control+↓>	<Ctrl+Shift+↓>	Переместить строку/выделенные строки вниз
<Command+L>	<Ctrl+L>	Выделить строку; повторить для выделения следующих строк
<Command+D>	<Ctrl+D>	Выделить слово; повторить для выделения других вхождений этого же слова
<Control+M>	<Ctrl+M>	Перейти к закрывающей скобке; повторить для перехода к открывающей скобке
<Control+Shift+M>	<Ctrl+Shift+M>	Выделить все содержимое в текущих скобках
<Command+K+Command+K>	<Ctrl+K+K> <Ctrl+Shift+Delete>	Удалить текст от текущей позиции курсора до конца строки
<Command+K+Delete>	<Ctrl+K+Backspace> <Ctrl+Shift+Backspace>	Удалить текст от текущей позиции курсора до начала строки

Окончание табл. 2.2

Mac	Windows	Описание
<Command+]>	<Ctrl+]>	Увеличить отступ текущей строки (выделенных строк)
<Command+[>	<Ctrl+[>	Уменьшить отступ текущей строки (выделенных строк)
<Command+Shift+D>	<Ctrl+Shift+D>	Дублировать строку
<Command+J>	<Ctrl+J>	Присоединить следующую строку к концу текущей строки
<Command+/>	<Ctrl+/>	Закомментировать/раскомментировать текущую строку (выделенные строки)
<Command+Option+/>	<Ctrl+Shift+/>	Закомментировать/раскомментировать блок выделенного текста
<Command+Y>	<Ctrl+Y>	Повторить последнее действие
<Command+Shift+V>	<Ctrl+Shift+V>	Вставить с отступом
<Control+пробел>	<Ctrl+пробел>	Выбор варианта автозаполнения
<Control+U>	<Ctrl+U>	"Мягкая" отмена; переход к месту внесения последнего изменения и отмена изменения после повторного применения этой команды
<Control+Shift+↑>	<Ctrl+Alt+↑>	Добавить курсор на предыдущей строке
<Control+Shift+↓>	<Ctrl+Alt+↓>	Добавить курсор на следующей строке
<Control+Shift+W>	<Alt+Shift+W>	Заключить выделенный текст в тег HTML

Оформление JavaScript-кода

Прежде чем приступить к написанию программ, вам надо узнать некоторые правила оформления кода на JavaScript.

- ✓ **JavaScript чувствителен к регистру символов.** Это напоминание будет неоднократно встречаться вам на протяжении всей книги, поскольку многие новички часто забывают об этом. В JavaScript слова "Baby" и "baby" — совершенно разные.
- ✓ **В JavaScript количество пробельных символов в коде не играет особой роли.** К числу пробельных символов относятся пробелы, а также символы табуляции и разрыва строки, т.е. любой символ, не имеющий визуального представления. При написании кода JavaScript не имеет значения, используете ли вы один пробел, два пробела или даже (в большинстве случаев) символ разрыва строки там, где требуется пробел. JavaScript игнорирует пробельные символы. Единственным исключением являются случаи, когда вы записываете текст, который должен быть выведен на экран. В подобных случаях пробелы сказываются на конечном результате. Наилучший подход заключается в том, чтобы использовать пробелы для повышения удобочитаемости кода, придерживаясь при этом определенной системы.

- ✓ **Избегайте использования зарезервированных слов.** В JavaScript есть список слов, имеющих специальное значение в этом языке. Список этих слов приведен в главе 3. А на данном этапе вам надо просто знать, что такие, например, слова, как `function`, `while`, `break` и `with`, имеют специальное значение.

  В JavaScript повсеместно используется символ **точка с запятой** (`;`). Код JavaScript состоит из отдельных предложений, или инструкций. Инструкции — фундаментальные строительные блоки программ на JavaScript, которые напоминают предложения в обычном языке, являющиеся строительными блоками абзацев. В JavaScript инструкции заканчиваются точкой с запятой.

  Если вы не закончите инструкцию точкой с запятой, то JavaScript сделает это за вас. Однако это может приводить к неожиданным результатам, поэтому обозначение конца инструкций точкой с запятой считается наилучшей практикой.

Выполнение JavaScript в окне браузера

Несмотря на то что JavaScript может встретиться вам в самых различных средах, чаще всего программы на JavaScript выполняются в браузерах. Управление вводом и выводом, манипулирование веб-страницами, обработка стандартных событий браузера, таких как щелчки и прокрутка, управление различными возможностями браузеров — вот для чего был создан JavaScript.

Существуют три способа выполнения JavaScript в браузере, каждый из которых рассматривается в последующих разделах:

- ✓ поместить код непосредственно в атрибут события HTML-элемента;
- ✓ поместить код между открывающим и закрывающим тегами `script`;
- ✓ поместить код в отдельный документ, который включается в документ HTML.

Вы будете не раз использовать различные комбинации всех этих трех подходов на одной и той же веб-странице. Однако знание того, когда именно следует применять тот или иной подход, имеет огромное значение, и это знание приходит лишь с практикой.

Использование JavaScript в атрибутах событий HTML-элементов

В HTML предусмотрено несколько специальных атрибутов, предназначенных для запуска JavaScript при наступлении определенных событий в браузере или при выполнении пользователем определенных действий. Вот пример HTML-элемента `button` с атрибутом события, обеспечивающим реакцию на щелчки мышью:

```
<button id="bigButton" onclick="alert('Привет,
          мир!');">Щелкните здесь</button>
```

В данном случае, когда пользователь щелкает на кнопке, созданной этим HTML-элементом, на экране появляется всплывающее окно, в котором отображается текст “Привет, мир!”.

В HTML насчитывается около 70 различных атрибутов событий. Наиболее часто используемые из них приведены в табл. 2.3.

Таблица 2.3. Часто используемые атрибуты событий элементов HTML

Атрибут	Условие запуска сценария
`onload`	Окончание загрузки страниц
`onfocus`	Получение элементом фокуса ввода (например, при активизации текстового поля)
`onblur`	Потеря элементом фокуса ввода (например, в результате выполнения пользователем щелчка в другом текстовом поле)
`onchange`	Изменение значения элемента
`onselect`	Выделение текста
`onsubmit`	Отправка формы
`onkeydown`	Нажатие клавиши
`onkeypress`	Нажатие и последующее отпускание клавиши
`onkeyup`	Отпускание клавиши
`onclick`	Щелчок мышью на элементе
`ondrag`	Перетаскивание элемента
`ondrop`	Вставка перетаскиваемого элемента
`onmouseover`	Перемещение указателя мыши над элементом

Вообще говоря, несмотря на простоту описанного способа, многие JavaScript-программисты считают этот подход не лучшим решением. Мы демонстрируем его применение в книге, поскольку атрибуты событий широко применяются и просты в изучении. Но уже сейчас вы должны знать, что для написания JavaScript-кода, реагирующего на события, существуют лучшие способы, чем использование атрибутов событий. Более подробно этот вопрос рассматривается в главе 11.

Использование JavaScript в элементе `script`

HTML-элемент `script` позволяет внедрить код JavaScript в HTML-документ. Элементы `script` часто помещают в элемент `head`, и ранее этот способ считался чуть ли не обязательным. Однако в наши дни элементы `script` используются как в элементе `head`, так и в теле веб-страниц.

Элемент `script` имеет очень простой формат.

```
<script>
    Сюда вставляется код JavaScript
</script>
```

С примером такого способа внедрения сценариев вы уже встречались в листинге 2.1. В листинге 2.2 представлен другой пример HTML-документа с тегом `script`, содержащим JavaScript-код. Однако в данном случае элемент `script` находится в конце тела документа (элемента `body`).

Листинг 2.2. Внедрение JavaScript-кода в элемент `script`

```
<!DOCTYPE html>
<html>
<head>
  <title>Привет, HTML!</title>
</head>
<body>
  <h1>Посчитаем до 10 вместе с JavaScript!</h1>
  <p id="theCount"></p>
  <script>
    var count = 0;
    while (count < 10) {
      count++;
      document.getElementById("theCount").innerHTML +=
            count + "<br>";
    }
  </script>
</body>
</html>
```

Если вы создадите новый файл в Sublime Text, введете в него содержимое листинга 2.2, а затем откроете его в браузере, то увидите, что этот листинг делает то же самое, что и листинг 2.1.

Зависимость момента запуска сценария JavaScript от расположения элемента `script`

Как правило, выполнение сценариев браузерами происходит по мере их загрузки. Браузер всегда читает веб-страницу сверху вниз, как это делаете и вы, читая обычную текстовую страницу. Но иногда желательно, чтобы выполнение сценария начиналось лишь после того, как браузер загрузит полностью всю страницу. В листинге 2.1 это достигалось за счет использования атрибута события `onload` в элементе `body`. Другой общепринятый способ, позволяющий задержать выполнение сценария, состоит в том, чтобы поместить подлежащий выполнению код в самом конце документа, как это сделано в листинге 2.2.

Ограничения JavaScript-кода, находящегося в элементах `script`

Несмотря на то что внедрение сценариев JavaScript в элементы `script` более предпочтительно по сравнению со встраиванием в атрибуты событий, у такого подхода имеются серьезные ограничения.

Наибольшим из них является то, что сценарии, внедренные таким способом, может использовать лишь та веб-страница, в которую они помещены. Иными словами, если вы помещаете свой сценарий в элемент `script`, то должны скопировать этот элемент

и вставить его во все веб-страницы, в которых он используется. Нетрудно понять, что сопровождение сайтов, содержащих сотни таких страниц, превращается в сплошной кошмар.

Когда целесообразно располагать JavaScript-код в элементах `script`

Этот метод внедрения JavaScript-кода имеет свои области применения. В случае небольших сценариев, которые просто вызывают другие сценарии JavaScript и изменяются лишь изредка, данный метод вполне приемлем и даже может привести к сокращению времени загрузки и отображения веб-страниц, поскольку количество запросов к серверу при этом уменьшается.

Одностраничные приложения, которые (как следует из самого их названия) состоят из единственной HTML-страницы, также великолепно подходят для использования данного типа внедрения сценариев, поскольку вносить изменения, если это потребуется, надо будет только в одном месте.

Однако, как правило, следует всегда использовать любую возможность минимизировать объем JavaScript-кода, внедряемого непосредственно в HTML-документ. Это упрощает сопровождение сайта и позволяет лучше структурировать код.

Включение внешних JavaScript-файлов

Третий и наиболее популярный способ включения кода JavaScript в HTML-документы основан на использовании атрибута `src` элемента `script`.

Элемент `script` с атрибутом `src` работает точно так же, как и элемент `script`, в котором JavaScript-код располагается между тегами, за исключением того, что в случае использования атрибута `src` код JavaScript загружается в HTML-документ из отдельного файла. Вот пример элемента `script` с атрибутом `src`:

```
<script src="myScript.js"></script>
```

В данном случае должен существовать отдельный файл `myScript.js`, располагающийся в той же папке, что и HTML-документ. Использование внешних файлов JavaScript предоставляет следующие преимущества:

- ✓ улучшается удобочитаемость HTML-файлов;
- ✓ упрощается сопровождение кода, поскольку вносить изменения или исправлять ошибки приходится только в одном месте.

Создание `.js`-файла

Создание внешнего JavaScript-файла напоминает создание HTML-файла и вообще файла любого другого типа. Чтобы заменить внедренный JavaScript-код в листинге 2.1 внешним JavaScript-файлом, выполните следующие действия.

1. **Выберите в редакторе Sublime Text пункты меню File⇨New File (Файл⇨Создать).**
2. **Скопируйте все, что находится между тегами `<script>` и `</script>` в файле `MyFirstProgram.html`, и вставьте его во вновь созданный `.js`-файл.**

Обратите внимание на то, что во внешние файлы копируется только JavaScript-код без элементов <script>.

3. **Сохраните новый файл с именем `countToTen.js` в той папке, в которой уже находится файл `MyFirstProgram.html`.**
4. **Измените элемент `<script>` в файле `MyFirstProgram.html`, добавив в него атрибут `src`, как показано ниже.**

```
<script src="countToTen.js"></script
```

Теперь файл `MyFirstProgram.html` должен содержать следующий текст.

```
<!DOCTYPE html>
<html>
<head>
  <title>Привет, HTML!</title>
  <script src="countToTen.js"></script>
</head>
<body onload="countToTen();">
  <h1>Посчитаем до 10 вместе с JavaScript!</h1>
  <p id="theCount"></p>
</body>
</html>
```

Содержимое нового файла `countToTen.js` должно быть таким.

```
function countToTen(){
  var count = 0;
  while (count < 10) {
    count++;
    document.getElementById("theCount").innerHTML +=
            count + "<br>";
  }
}
```

После того как вы сохраните оба файла, они должны появиться в папке вашего проекта, отображаемой в боковой панели редактора Sublime Text (рис. 2.8).

Организация `.js`-файлов проекта

Некоторые JavaScript-файлы могут достигать очень больших размеров. Во многих случаях целесообразно разбивать их на меньшие файлы, организованные по типу содержащихся в них функций. Например, один файл может содержать сценарии, связанные со входом пользователя в вашу программу, а второй — сценарии, связанные с ведением блога.

Однако для небольших программ обычно достаточно иметь всего лишь один файл, и, как правило, такому файлу присваивают какое-либо типовое имя, например `app.js`, `main.js` или `scripts.js`.

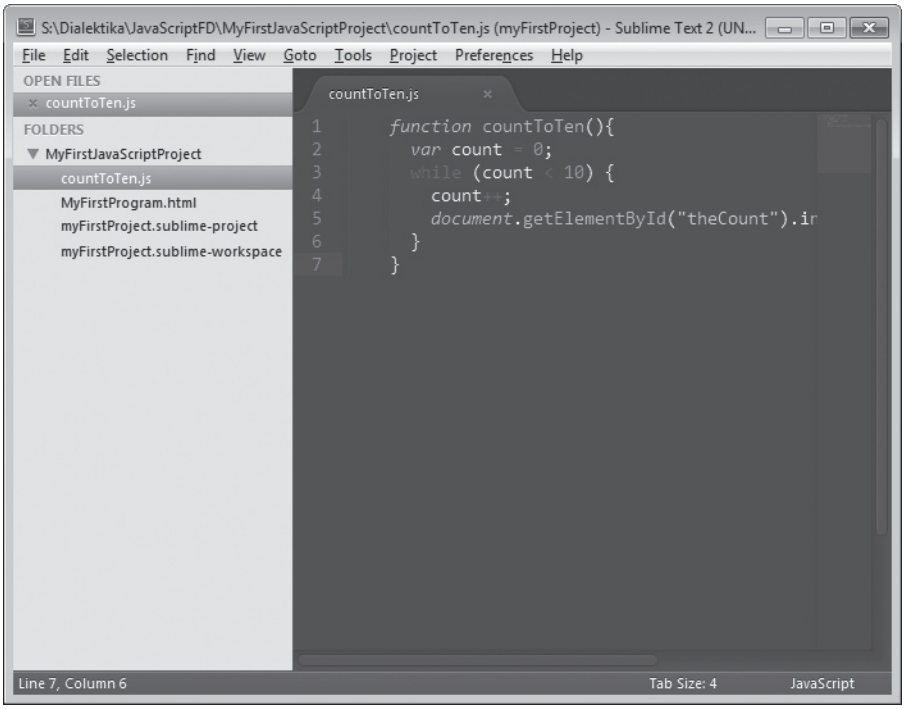


Рис. 2.8. Просмотр файлов, находящихся в папке проекта, в редакторе Sublime Text

JavaScript-файлы не обязательно должны находиться в той же папке, что и HTML-файл, который их включает. Более того, мы рекомендуем вам создавать новую папку, предназначенную специально для хранения внешних JavaScript-файлов. Большинство людей, которых мы знаем, присваивают таким папкам имена наподобие `js`.

Чтобы создать папку `js` в вашем проекте Sublime Text и переместить в нее `.js`-файл, выполните следующие действия.

1. **Щелкните правой кнопкой мыши на имени проекта в боковой панели Sublime Text.**

   Откроется подменю.

2. **Выберите в подменю пункт New Folder (Создать папку).**

   В нижней части окна Sublime Text появится поле `Folder Name` (Имя папки).

3. **Введите имя папки `js` в текстовом поле и нажмите клавишу <Enter>.**

   В боковой панели отобразится новая папка `js`.

4. **Откройте файл `countToTen.js`, выберите пункты меню File⇨Save As и сохраните файл в новой папке `js`.**

5. **Щелкните правой кнопкой мыши на версии файла `countToTen.js`, хранящейся вне папки `js`, и выберите в открывшемся подменю пункт Delete File (Удалить файл).**

6. **Откройте файл `MyFirstProgram.html` и измените элемент `<script>` таким образом, чтобы он отражал новое расположение `.js`-файла.**

```
<script src="js/countToTen.js"></script
```

Когда вы откроете файл `MyFirstProgram.html` в браузере (или просто обновите страницу), страница должна выглядеть точно так же, как и до перемещения JavaScript-файла в его собственную папку.

Использование консоли разработчика JavaScript

Иногда полезно иметь возможность запускать команды JavaScript без создания веб-страницы и последующего включения в нее кода сценариев или блоков `<script>`. В подобных ситуациях можно воспользоваться консолью JavaScript браузера Chrome (рис. 2.9).

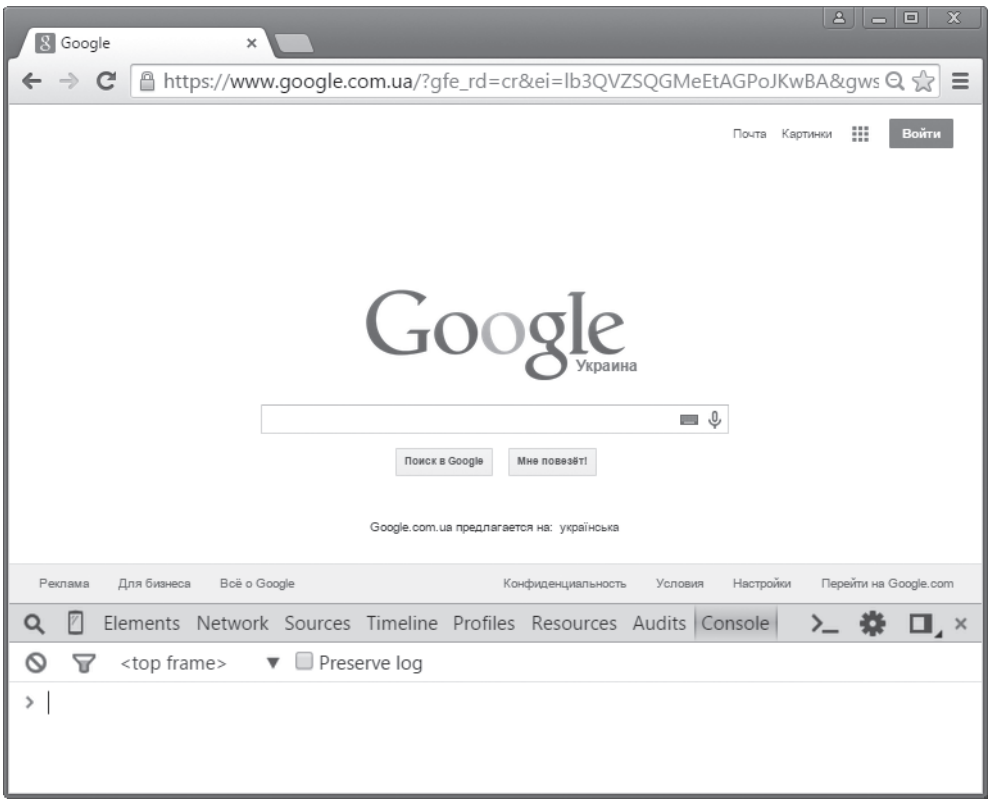


Рис. 2.9. Консоль JavaScript браузера Chrome

Чтобы получить доступ к консоли JavaScript, откройте меню Chrome, расположенное в правом верхнем углу окна браузера. Ему соответствует пиктограмма в виде трех

горизонтальных линий. Щелкните на ней и выберите в раскрывшемся меню пункт Дополнительные инструменты, а в следующем меню — пункт Консоль JavaScript.

Ах да! Для открытия окна консоли JavaScript существует гораздо более быстрый способ, требующий простого нажатия комбинации клавиш <Alt+Command+J> (Mac) или <Ctrl+Shift+J> (Windows).

Консоль JavaScript является, пожалуй, наилучшим другом разработчика на JavaScript. Она не только позволяет быстро и просто тестировать и выполнять JavaScript-код, но и сообщает, в каких местах кода содержатся ошибки, и предлагает средства, облегчающие обнаружение и разрешение проблем, связанных с работой кода.

Сразу же после открытия консоли можно начать ввод команд, которые будут выполняться после нажатия клавиши <Enter>. Чтобы попрактиковаться в этом, откройте консоль JavaScript и последовательно вводите приведенные ниже команды, нажимая клавишу <Enter> после ввода каждой из них.

```
1080/33
40 + 2
40 * 34
100%3
34++
34--
```

Комментирование кода

Когда вы изучите большее количество команд JavaScript и приступите к написанию более крупных программ, вы поймете, насколько полезными могут быть небольшие памятные записки, в которых вы фиксируете свои мысли относительно тех или иных аспектов программы или кратко описываете назначение отдельных фрагментов кода. На языке программистов эти записки, предназначенные для вас самих (а также для других людей, которые могут работать с вашим кодом), называются *комментариями*, а сам процесс их написания — *комментированием кода*.

Движок JavaScript полностью игнорирует комментарии, поскольку они предназначены исключительно для чтения людьми. Вы можете использовать их для записи необходимых пояснений или уточнений, изложения логики своих рассуждений, а также для фиксации того, что вы собираетесь сделать в дальнейшем для улучшения кода.

Комментарии в коде никогда не будут лишними. Даже если во время написания кода вы считаете, что он и так достаточно понятен, можно со стопроцентной уверенностью утверждать, что спустя несколько месяцев, когда вам понадобится внести изменения в код, восстановить в памяти логику его работы вам будет не так-то легко.

В JavaScript существуют два типа комментариев:

- ✓ однострочные;
- ✓ многострочные.

Однострочные комментарии

Однострочные комментарии начинаются двумя символами косой черты (//). Все, что находится после них до конца строки, игнорируется *парсером* (синтаксическим анализатором) JavaScript.

Однострочные комментарии не обязательно должны располагаться в начале строки. Они довольно часто встречаются в той же строке, что и комментируемый код, поясняя его назначение. Например:

```
pizzas = pizza + 1; // добавить еще одну пиццу
```

Многострочные комментарии

Многострочные комментарии начинаются символами /* и сообщают парсеру JavaScript о том, что весь последующий текст вплоть до пары символов */ следует игнорировать. Многострочные комментарии полезны для более полного документирования кода.

```
/* Функция countToTen выполняет следующие действия:
  * Инициализирует переменную count нулевым значением
  * Запускает цикл с проверкой того, что текущее значение
         count меньше 10
  * Увеличивает значение count на 1
  * Добавляет текущее значение count вместе со следующим за ним
         символом разрыва строки в абзац с id='theCount'
  * Запускает следующую итерацию цикла
*/
```

Использование комментариев для предотвращения выполнения кода

Комментарии полезны не только для документирования программ, но и для изоляции фрагментов кода в процессе отладки с целью локализации ошибок. Допустим, нам захотелось узнать, что произойдет, если удалить из функции countToTen строку, обеспечивающую приращение значения переменной count. Для этого достаточно ввести в начале строки символы однострочного комментария или, как говорят, "закомментировать" эту строку.

```
function countToTen(){
var count = 0;
while (count < 10) {
// count++;
document.getElementById("theCount").innerHTML +=
count + "<br>";
}
}
```

Если вы запустите эту программу, то строка count++ уже не будет выполняться и программа будет бесконечно (или пока вы не закроете окно браузера) выводить нули.

Конструкция, которую мы только что использовали, называется *бесконечным циклом*. Выполнение видоизмененной версии программы не причинит вреда вашему компьютеру, но, вероятно, заставит процессор вашего компьютера работать на бешеной скорости до тех пор, пока вы не закроете окно браузера, в котором выполняется программа.