

Владимир Дронов

JavaScript

20 уроков для начинающих

Санкт-Петербург
«БХВ-Петербург»
2021

УДК 004.438JavaScript
ББК 32.973.26-018.1
Д75

Дронов В. А.

Д75 JavaScript: 20 уроков для начинающих. — СПб.: БХВ-Петербург, 2021. — 352 с.: ил. — (Для начинающих)

ISBN 978-5-9775-6589-9

В книге 20 иллюстрированных уроков, 40 практических упражнений на тему программирования веб-сценариев и более 70 заданий для самостоятельной работы. Изложены основы JavaScript: данные и операторы, выражения и управляющие конструкции, функции, классы, объекты и массивы, средства отладки. Раскрыты механизмы управления веб-страницами: события и их обработка, управление элементами, графика и мультимедиа, веб-формы и элементы управления, регулярные выражения, навигация и управление окнами. Рассмотрена работа с HTML API и компонентное программирование: асинхронное программирование, работа с внешними данными, программная графика, объявление своих классов, создание компонентов. Освещены технологии взаимодействия с сервером: AJAX, PHP, разработка фронтендов и бэкендов, серверные сообщения.

Электронный архив на сайте издательства содержит коды всех примеров и результаты выполнения упражнений.

Для начинающих веб-разработчиков

УДК 004.438JavaScript
ББК 32.973.26-018.1

Группа подготовки издания:

Руководитель проекта	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Екатерина Савитина</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Дизайн обложки	<i>Карины Соловьевой</i>

Подписано в печать 07.07.20.

Формат 70×100^{3/16}. Печать офсетная. Усл. печ. л. 28,38.

Тираж 1000 экз. Заказ №

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

Отпечатано с готового оригинал-макета

ООО "Принт-М", 142300, М.О., г. Чехов, ул. Полиграфистов, д. 1

Оглавление

Введение	11
Почему JavaScript?.....	11
Что вы найдете в этой книге?	12
Что вам понадобится?	12
Типографские соглашения.....	13
ЧАСТЬ I. НАЧАЛА JAVASCRIPT.....	15
Урок 1. Типы данных, операторы и переменные.....	17
1.1. Упражнение. Используем консоль веб-обозревателя.....	17
1.2. Типы данных и операторы	20
1.3. Числа, арифметические операторы и приоритет операторов	20
1.4. Строки. Конкатенация строк	22
1.5. Условия и операторы сравнения	23
1.6. Логические операторы	24
1.7. Переменные.....	24
1.7.1. Три оператора объявления переменных.....	27
1.7.2. Еще об объявлении переменных	27
1.8. Арифметические операторы, применимые лишь к переменным	28
1.9. Условный оператор	28
1.10. Получение типа данных.....	29
1.11. Преобразование типов.....	30
1.12. Самостоятельные упражнения	32
Урок 2. Выражения и управляющие конструкции.....	35
2.1. Упражнение. Пишем первый веб-сценарий	35
2.2. Упражнение. Применяем условные выражения	38
2.2.1. Сокращенная форма условного выражения.....	39
2.2.2. Множественное условное выражение	39
2.3. Упражнение. Используем блоки	40
2.3.1. Переменные в блоках	41
2.4. Выражение выбора	41
2.5. Упражнение. Используем цикл со счетчиком	43
2.5.1. Объявление счетчика непосредственно в цикле.....	46
2.5.2. В приращении можно записать несколько выражений.....	46
2.6. Циклы с предусловием и постусловием	46
2.7. Операторы <i>break</i> и <i>continue</i>	47
2.8. Комментарии.....	48
2.9. Строгий режим.....	48
2.10. Как набирать JavaScript-код?.....	49
2.11. Самостоятельные упражнения	49

Урок 3. Функции и массивы	51
3.1. Упражнение. Используем функции	51
3.2. Локальные переменные и вложенные функции.....	54
3.3. Функции с необязательными параметрами	55
3.3.1. Функции с необязательными параметрами в «старом» стиле.....	56
3.4. Функции с произвольным количеством параметров.....	56
3.5. Упражнение. Используем рекурсию.....	57
3.6. Анонимные функции и функции-стрелки	58
3.7. Функциональный тип данных	60
3.8. Упражнение. Создаем внешний веб-сценарий	60
3.9. Встроенные функции JavaScript.....	61
3.10. Массивы. Объектный тип данных.....	61
3.11. Самостоятельные упражнения	63
Урок 4. Классы и объекты	65
4.1. Объекты. Свойства и методы	65
4.1.1. Доступ к свойствам объекта посредством оператора <code>[]</code> (квадратные скобки)	66
4.2. Классы	66
4.3. Класс <i>Number</i>	67
4.4. Класс <i>String</i>	69
4.5. Класс <i>Array</i>	71
4.5.1. Сортировка массивов	76
4.6. Класс <i>Date</i>	78
4.7. Класс <i>Math</i>	80
4.8. Класс <i>Arguments</i> . Коллекции	82
4.9. Класс <i>Object</i> . Служебные объекты. Объектная нотация.....	82
4.10. Объектный тип. Значение <i>null</i>	83
4.11. Хранение объектов в переменных. Значение и ссылочные типы	84
4.12. Добавленные свойства	85
4.13. Дополнительные средства для работы с объектами.....	86
4.14. Самостоятельные упражнения	87
Урок 5. Средства отладки	89
5.1. Вывод сообщений в консоли	89
5.1.1. Вывод в консоли произвольных сообщений.....	89
5.2. Упражнение. Работаем с отладчиком веб-обозревателя.....	90
5.2.1. Удаление точек останова	95
5.3. Инспектор DOM	95
5.4. Исключения и их обработка	96
5.4.1. Генерирование исключений	98
5.5. Самостоятельные упражнения	99
ЧАСТЬ II. УПРАВЛЕНИЕ ВЕБ-СТРАНИЦЕЙ И ВЕБ-ОБОЗРЕВАТЕЛЕМ	101
Урок 6. События и их обработка	103
6.1. Упражнение. Обрабатываем события.....	103
6.1.1. Еще немного об обработчиках событий.....	107
6.1.2. Удаление привязки обработчика к событию	108
6.1.3. Альтернативный способ привязки обработчиков к событиям	108

6.2. События, поддерживаемые элементами страницы	109
6.2.1. Особенности события <i>beforeunload</i>	110
6.3. Упражнение. Получаем сведения о событии	111
6.3.1. Классы событий и их свойства	113
6.4. Упражнение. Изучаем фазы событий	116
6.4.1. Свойства и методы класса <i>Event</i> , имеющие отношение к «прохождению» событий	118
6.5. Упражнение. Отменяем обработку событий по умолчанию	118
6.5.1. Свойства класса <i>Event</i> , касающиеся обработки события по умолчанию	121
6.6. Самостоятельные упражнения	121
Урок 7. Управление элементами веб-страниц	123
7.1. Получение доступа к элементам страницы	123
7.1.1. Доступ к элементам определенного типа	123
7.1.2. Доступ к любому элементу страницы	124
7.1.3. Доступ к родителю, соседям и потомкам	126
7.1.4. Контекст исполнения веб-сценариев	127
7.2. Управление элементами веб-страницы	127
7.3. Упражнение. Делаем стильную полосу прокрутки	131
7.4. Упражнение. Управляем стилями	134
7.4.1. Средства для управления стилевыми классами	136
7.4.2. Средства для управления встроенными стилями	137
7.5. Изменение содержимого элементов	137
7.5.1. Методы <i>write</i> и <i>writeln</i>	139
7.6. Упражнение. Создаем новые элементы веб-страниц путем конструирования	139
7.6.1. Методы, конструирующие элементы веб-страниц	141
7.7. Получение сведений о веб-странице	144
7.8. Самостоятельные упражнения	144
Урок 8. Графика и мультимедиа	147
8.1. Получение сведений о графических изображениях	147
8.2. Управление мультимедийными элементами	148
8.2.1. Свойства	148
8.2.2. Методы	149
8.2.3. События	150
8.3. Упражнение. Реализуем свой видеопроигрыватель	151
8.4. Самостоятельные упражнения	154
Урок 9. Веб-формы и элементы управления	155
9.1. Взаимодействие с элементами управления	155
9.1.1. Получение и обработка введенного в элемент значения	158
9.1.2. Программное создание и удаление пунктов списка	159
9.2. Упражнение. Пишем первое клиентское веб-приложение	160
9.3. Упражнение. Работаем с веб-формой	162
9.3.1. Свойства, методы и события веб-формы	164
9.4. Упражнение. Реализуем валидацию данных в веб-форме	164
9.4.1. Вывод сообщений об ошибках в произвольном месте веб-страницы	166
9.4.2. Вывод произвольных сообщений об ошибках	167
9.5. Самостоятельные упражнения	168

Урок 10. Регулярные выражения.....	171
10.1. Введение в регулярные выражения	171
10.1.1. Создание регулярных выражений	171
10.1.2. Использование регулярных выражений: простые случаи.....	173
10.2. Литералы регулярных выражений	173
10.2.1. Метасимволы	173
10.2.2. Поднаборы.....	174
10.2.3. Вариант	175
10.2.4. Квантификаторы	175
10.2.5. Подквантификатор.....	176
10.2.6. Группы и обратные ссылки.....	177
10.2.7. Обычные символы	178
10.3. Поиск и обработка фрагментов, совпадающих с регулярными выражениями.....	178
10.3.1. Обычный режим поиска.....	178
10.3.2. Глобальный поиск	179
10.3.3. Многострочный поиск.....	180
10.3.4. Замена совпавших фрагментов.....	180
10.3.5. Прочие полезные инструменты	181
10.4. Упражнение. Выполняем программную валидацию с помощью регулярного выражения.....	181
10.5. HTML-валидация с применением регулярных выражений	182
10.6. Самостоятельные упражнения	182
Урок 11. Взаимодействие с веб-обозревателем	185
11.1. Работа с окном веб-обозревателя	185
11.1.1. Еще один способ записи веб-сценариев, манипулирующих элементами страницы.....	187
11.2. Упражнение. Навигация по якорям с подсветкой активной гиперссылки.....	188
11.3. Работа с текущим интернет-адресом	190
11.4. Получение сведений о клиентском компьютере.....	191
11.5. Вывод стандартных диалоговых окон	192
11.6. Самостоятельное упражнение	193
ЧАСТЬ III. HTML API И КОМПОНЕНТНОЕ ПРОГРАММИРОВАНИЕ.....	195
Урок 12. Таймеры и фоновые потоки.....	197
12.1. Упражнение. Используем периодические таймеры	197
12.2. Однократный таймер.....	201
12.3. Упражнение. Применяем фоновые потоки	202
12.4. Самостоятельные упражнения	205
Урок 13. Работа с файлами, хранение данных и перетаскивание.....	207
13.1. Работа с локальными файлами	207
13.1.1. Чтение сведений о файлах	207
13.1.2. Считывание текстовых файлов. Класс <i>FileReader</i>	208
13.1.3. Вывод индикатора процесса при считывании файла	210
13.1.4. Считывание графических файлов	210
13.2. Упражнение. Реализуем предварительный просмотр выбранного графического файла	210
13.3. Хранение данных на стороне клиента	212

13.4. Перетаскивание.....	213
13.4.1. Превращение элемента-источника в перетаскиваемый.....	213
13.4.2. Задание перемещаемых данных в источнике.....	213
13.4.3. Указание допустимых операций.....	214
13.4.4. Подготовка элемента-приемника.....	215
13.4.5. Завершение перетаскивания.....	216
13.5. Упражнение. Практикуемся в реализации перетаскивания.....	216
13.6. Перетаскивание файлов в поле выбора файлов.....	219
13.7. Самостоятельные упражнения.....	219

Урок 14. Программная графика..... 221

14.1. Холст HTML.....	221
14.2. Рисование прямоугольников.....	222
14.3. Указание основных параметров графики.....	222
14.3.1. Цвета линий и заливок.....	223
14.3.2. Параметры тени.....	223
14.4. Вывод текста.....	224
14.5. Упражнение. Рисуем диаграмму.....	225
14.6. Рисование сложных фигур.....	227
14.6.1. Начало и завершение рисования.....	227
14.6.2. Перемещение пера.....	227
Рисование прямых линий.....	228
Замыкание контура.....	228
Указание параметров линий.....	229
Рисование дуг.....	230
Рисование кривых Безье.....	231
Рисование прямоугольников.....	232
14.7. Градиенты и графические закраски.....	233
14.7.1. Линейные градиенты.....	233
14.7.2. Радиальные градиенты.....	235
14.7.3. Графическая закраска.....	236
14.8. Преобразования.....	237
14.9. Вывод сторонних изображений.....	238
14.10. Управление композицией.....	239
14.11. Создание масок.....	240
14.12. Сохранение и восстановление состояния холста.....	240
14.13. Упражнение. Рисуем цветок.....	241
14.14. Самостоятельные упражнения.....	242

Урок 15. Объявление своих классов..... 245

15.1. Объявление нового класса.....	245
15.1.1. Конструктор. Объявление свойств и методов в конструкторе.....	245
15.1.2. Прототип. Объявление свойств и методов в прототипе.....	246
15.2. Упражнение. Объявляем класс слайдера.....	247
15.3. Упражнение. Создаем динамическое свойство.....	251
15.3.1. Динамические свойства, доступные только для чтения и только для записи.....	252
15.3.2. Объявление обычных свойств методом <i>defineProperty</i>	252
15.4. Упражнение. Реализуем наследование классов.....	253
15.4.1. Переопределение методов.....	255
15.4.2. Использование метода <i>call</i> для вызова методов базового класса.....	256

15.5. Объявление статических свойств и методов	256
15.6. Упражнение. Расширяем функциональность встроенных классов	257
15.7. Самостоятельные упражнения	258
Урок 16. Компоненты	261
16.1. Понятие компонента	261
16.2. Упражнение. Создаем компонент	261
16.3. Замыкания	265
16.4. Упражнение. Изолируем компонент в замыкании	267
16.5. Самостоятельные упражнения	268
ЧАСТЬ IV. ВЗАИМОДЕЙСТВИЕ С СЕРВЕРОМ	269
Урок 17. Технология AJAX	271
17.1. Реализация AJAX	271
17.1.1. Подготовка объекта AJAX	272
17.1.2. Указание интернет-адреса загружаемого файла	272
17.1.3. Получение загруженного файла	272
17.1.4. Отправка веб-серверу запроса на получение файла	273
17.2. Упражнение. Пишем сверхдинамический веб-сайт	274
17.3. Синхронная загрузка файлов по технологии AJAX	276
17.4. Формат JSON	276
17.5. Упражнение. Загрузка и вывод JSON-данных	277
17.6. Упражнение. Пишем компонент <i>AJAXLoader</i>	280
17.7. Самостоятельные упражнения	282
Урок 18. Серверные веб-приложения. Платформа PHP	285
18.1. Серверные веб-приложения. Платформа PHP	285
18.2. Упражнение. Изучаем основы PHP	286
18.3. Упражнение. Пишем простейшую фотогалерею на PHP	289
18.4. Упражнение. Передаем данные методом GET	293
18.5. Упражнение. Передаем данные методом POST	296
18.6. Самостоятельные упражнения	300
Урок 19. Разработка фронтендов и бэкендов	301
19.1. Веб-разработка: старый и новый подходы. Фронтенды и бэкенды. Веб-службы	301
19.2. Упражнение. Новая фотогалерея. Вывод списка миниатюр	302
19.3. Упражнение. Новая фотогалерея. Показ изображений	305
19.4. Отправка веб-службе данных методом POST	309
19.4.1. Отправка веб-формы целиком	309
19.4.2. Отправка произвольных данных	310
19.4.3. Отправка данных в виде строки POST-параметров	310
19.5. Упражнение. Новая фотогалерея. Загрузка изображений в галерею	311
19.6. Самостоятельные упражнения	316
Урок 20. Серверные сообщения	317
20.1. Использование серверных сообщений	317
20.1.1. Что представляет собой серверное сообщение?	317
20.1.2. Отправка серверных сообщений	318
20.1.3. Прием серверных сообщений	318

20.1.4. Поддержание соединения и возобновление приема.....	319
20.1.5. Создание своих событий.....	320
20.1.6. Разрыв соединения	320
20.2. Упражнение. Новая фотогалерея. Более отзывчивый список миниатюр	321
Заключение.....	325
Приложение 1. Приоритет операторов.....	327
Приложение 2. Пакет хостинга XAMPP	329
П2.1. Установка пакета хостинга XAMPP.....	329
П2.2. Панель управления XAMPP. Запуск и остановка веб-сервера	334
П2.2.1. Указание языка при первом запуске	334
П2.2.2. Окно панели управления XAMPP	334
П2.2.3. Запуск веб-сервера	335
П2.2.4. Проверка работоспособности веб-сервера	335
П2.2.5. Остановка веб-сервера.....	336
П2.3. Использование веб-сервера	336
П2.3.1. Тестирование веб-сайта с применением XAMPP	336
П2.3.2. Просмотр журналов работы веб-сервера и PHP.....	336
П2.4. Решение проблем	337
П2.4.1. Увеличение максимального размера выгружаемых файлов	337
П2.4.2. Отключение кеширования файлов веб-обозревателем.....	338
Приложение 3. Описание электронного архива	341
Предметный указатель.....	343

Введение

Почему JavaScript?

Что вы найдете в этой книге?

Что вам понадобится в процессе чтения?

Типографские соглашения

Здравствуйтесь, уважаемый читатель!

Если вы держите в руках эту книгу, значит, желаете научиться языку программирования JavaScript. Это очень правильное желание. А это очень правильная книга!

JavaScript — один из популярнейших в настоящее время языков для написания программ, а в деле веб-программирования у него вообще нет конкурентов. Знание JavaScript требуется от каждого, кто намеревается серьезно заняться написанием сайтов. Любой работодатель, узнавший, что претендент на должность веб-работчика не владеет этим языком, тут же даст ему от ворот поворот.

Почему JavaScript?

Прежде всего, он — один из трех «китов», на которых зиждется величественный дворец веб-разработки. Первый «кит» — язык HTML — описывает саму веб-страницу, ее содержание, второй — CSS — ее оформление, а JavaScript — поведение.

JavaScript может «оживить» страницу, заставляя ее реагировать на действия пользователя. Он может превратить набор обычных картинок в красивую фотогалерею. Он может загрузить с сервера какие-либо данные и вывести их прямо на странице в виде абзаца, списка, таблицы или графика. Он даже может превратить страницу в настоящую программу, обрабатывающую занесенные пользователем данные и выводящую результаты их обработки.

JavaScript — единственный язык программирования, позволяющий «влезть» внутрь веб-страницы, исправить или дополнить ее содержание. JavaScript — единственный язык программирования, непосредственно поддерживаемый веб-обозревателями.

Ранее говорилось, что у этого языка в своей области нет конкурентов. И в ближайшем будущем они не появятся.

Что вы найдете в этой книге?

Философия программирования — это часть программирования, со всех сторон окруженная водой.

- ◆ Двадцать коротких, емких, наглядных, иллюстрированных уроков, дающих необходимые теоретические знания.
- ◆ Более сорока практических упражнений, выполняемых под руководством автора. Выполняя их, вы закрепите полученные знания и приобретете программистский опыт, что безусловно оценит ваш будущий работодатель.
- ◆ Восемнадцать упражнений, рассчитанных на самостоятельное выполнение. Чтобы справиться с ними, необходимо лишь внимательно изучать приведенный в уроках теоретический материал.
- ◆ Описание самых востребованных в настоящее время приемов программирования: объявления своих классов, написания компонентов, программирования фронтендов и бэкендов, использования фоновых процессов и серверных сообщений.
- ◆ Вводный курс популярной программной платформы PHP, знание которой очень пригодится веб-программисту.
- ◆ Полное отсутствие описаний устаревших и малополезных инструментов, отвлеченных рассуждений, философствований на тему программирования и прочей «воды».

|| *Чтобы по этой книге овладеть JavaScript-программированием, вам хватит и месяца.*

Книгу сопровождает электронный архив (см. приложение 3), содержащий результаты выполнения всех упражнений, равно как и необходимые исходные файлы. Архив выложен на FTP-сервер издательства «БХВ-Петербург» по адресу <ftp://ftp.bhv.ru/9785977565899.zip>. Ссылка доступна и со страницы книги на сайте издательства www.bhv.ru.

Что вам понадобится?

Для изучения этой книги вам, уважаемый читатель, нужны следующие программы:

- ◆ *веб-обозреватель*, разумеется. Автор тестировал подготовленные им примеры в Google Chrome и Mozilla Firefox (остальные веб-обозреватели либо аналогичны Chrome, либо малопопулярны);
- ◆ *текстовый редактор* — можно использовать Блокнот, но автор рекомендует какой-либо из чисто «программистских» редакторов: Visual Studio Code (<https://code.visualstudio.com/>), Atom (<https://atom.io/>), Notepad++ (<https://notepad-plus-plus.org/>), Sublime Text (<https://www.sublimetext.com/>), Brackets (<http://brackets.io/>) и т. п.;

- ◆ пакет веб-хостинга *XAMPP* (<https://www.apachefriends.org/ru/index.html>). Автор применял версию 7.3.5, включающую веб-сервер Apache HTTP Server 2.4.39 и PHP 7.3.5. Описание пакета веб-хостинга XAMPP и инструкция по его установке на компьютер приведены в *приложении 2*.

Еще от вас требуется:

- ◆ владеть языками HTML и CSS, на которых пишутся сами веб-страницы и оформление для них;
- ◆ иметь минимальные навыки веб-верстки;
- ◆ знать в общих чертах как работает веб-сервер (это понадобится нам в *части IV* книги).

Типографские соглашения

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

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

```
<script type="text/javascript">
    const ins = window.prompt('Величина в дюймах', 0);
    const cents = ins * 2.54;
</script>
```

- ◆ в угловые скобки <> заключаются наименования различных значений, которые дополнительно выделяются *курсивом*. В реальный код, разумеется, должны быть подставлены реальные значения. Например:

```
return <возвращаемое значение>
```

Здесь вместо подстроки *возвращаемое значение* должно быть подставлено реальное возвращаемое значение;

- ◆ в квадратные скобки [] заключаются необязательные фрагменты кода. Например:

```
toString([<основание>])
```

Здесь *основание* может указываться, а может и не указываться;

- ◆ слишком длинные, не помещающиеся на одной строке книги фрагменты, автор разрывал на несколько строк и в местах разрывов ставил знаки ¶. Например:

```
active = document.querySelector¶
('nav a.active');
```

Приведенный код здесь разбит на две строки, но должен быть набран в одну. Символ ¶ при этом нужно удалить;

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

```
function circleLength(d) {  
    d = d || 2;  
    . . .  
}
```

Здесь весь код между второй и последней строками пропущен.

Обычно такое можно встретить в исправленных впоследствии фрагментах кода — приведены лишь собственно исправленные строки, а оставшиеся неизменными пропущены. Также троеточие используется, чтобы показать, в какое место должен быть вставлен вновь написанный код: в начало исходного фрагмента, в его конец или в середину, между уже присутствующими в нем строками.

ВНИМАНИЕ!

Все приведенные здесь типографские соглашения имеют смысл только в примерах написания языковых конструкций JavaScript.

ЧАСТЬ I

Начала JavaScript

- ⇒ Данные, которыми манипулирует JavaScript.
- ⇒ Переменные.
- ⇒ Операторы.
- ⇒ Выражения.
- ⇒ Управляющие конструкции.
- ⇒ Функции.
- ⇒ Объекты и классы.
- ⇒ Веб-сценарии: внутренние и внешние.
- ⇒ Средства отладки.
- ⇒ Исключения и обработка ошибок.

Урок 1

Типы данных, операторы и переменные

Консоль веб-обозревателя
Типы данных
Числа
Строки
Логические значения
Операторы
Переменные
Преобразование типов

1.1. Упражнение.

Используем консоль веб-обозревателя

Изучение основ языка JavaScript удобно начать, набирая код в консоли.

Консоль — интерактивная исполняющая среда, встроенная в веб-обозреватель. В консоли можно набрать какое-либо выражение и сразу увидеть результат его выполнения.

Вызовем консоль у установленного у нас веб-обозревателя и выполним в ней несколько выражений.

1. Запустим веб-обозреватель (автор книги использовал Google Chrome).
2. Нажмем клавишу <F12>, чтобы вызвать встроенные в веб-обозреватель отладочные инструменты, в число которых входит и консоль. По умолчанию панель с отладочными инструментами появляется у нижней стороны окна веб-обозревателя (рис. 1.1).
3. В верхней части панели находятся корешки вкладок, на которых располагаются различные отладочные инструменты. Переключимся на вкладку **Console**, щелкнув на ней мышью (рис. 1.2).

На этой вкладке и находится консоль веб-обозревателя (рис. 1.3).

В белой области редактирования, занимающей большую часть вкладки, виден мигающий текстовый курсор. Здесь вводятся выражения JavaScript, которые нужно выполнить.

- Для начала сложим два числа, для чего наберем в области редактирования следующий простой код:

27 + 14

и нажмем клавишу <Enter>.

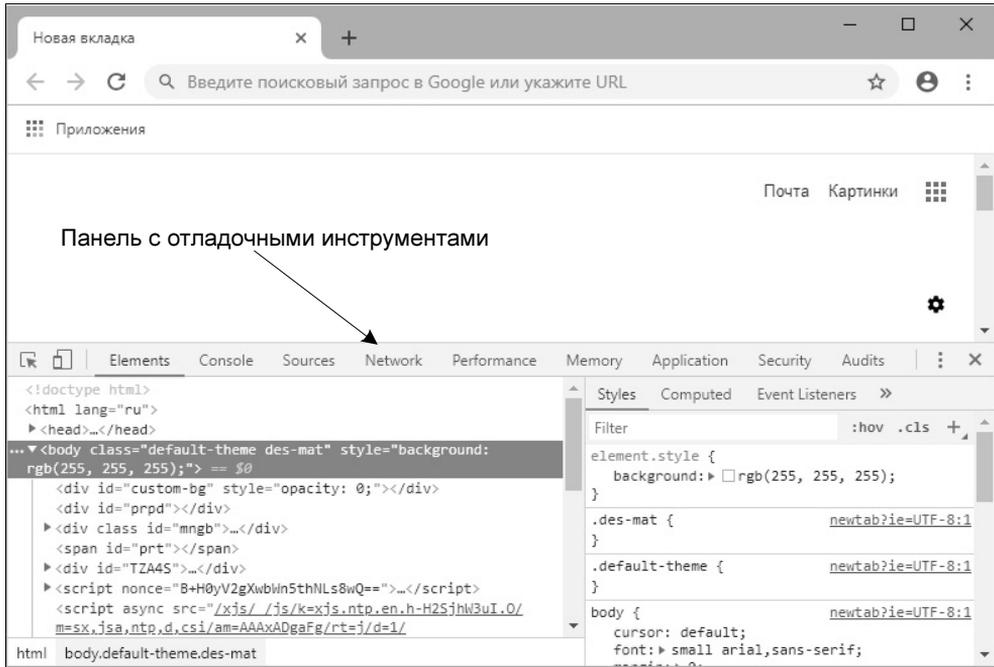


Рис. 1.1. Панель с отладочными инструментами веб-обозревателя

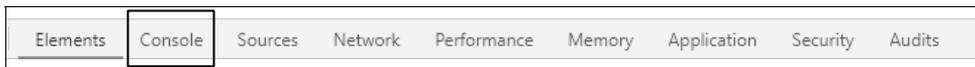


Рис. 1.2. Корешок вкладки Console на панели с отладочными инструментами веб-обозревателя

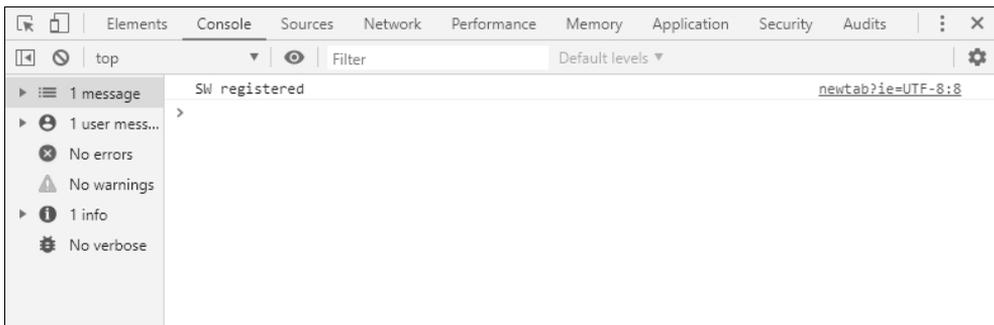


Рис. 1.3. Консоль веб-обозревателя

Ниже набранного кода мы увидим результат сложения чисел 27 и 14 (рис. 1.4).

Символом «больше» `>`, расположенным в консоли слева, помечается строка, в которую в настоящий момент будет вводиться код JavaScript.

```
SW registered
> 27 + 14
< 41
> |
```

Рис. 1.4. Результат сложения чисел 27 и 14 в консоли веб-обозревателя

Символом «меньше» `<` помечается строка, в которой выводится результат исполнения набранного кода после нажатия клавиши `<Enter>`.

- Мы можем обрабатывать и строки: давайте объединим строки `Java` и `Script` в одну:

```
> 'Java' + 'Script'
< "JavaScript"
```

Здесь символ `+` обозначает операцию объединения строк.

- Еще мы можем сравнивать значения: проверим, действительно ли число 10 меньше 20:

```
> 10 < 20
< true
```

Результатом будет логическое значение `true` — «истина». Так веб-обозреватель сообщает нам, что записанное сравнение выполняется (оно истинно).

- Напоследок для эксперимента выполним заведомо не выполняющееся, ложное сравнение:

```
> 15 > 25
< false
```

Результатом станет логическое значение `false` — «ложь».

- Очистим консоль, щелкнув на кнопке **Clear console**, что находится в окне консоли слева, под набором корешков вкладок (рис. 1.5). Для очистки консоли можно также нажать комбинацию клавиш `<Ctrl>+<L>`.

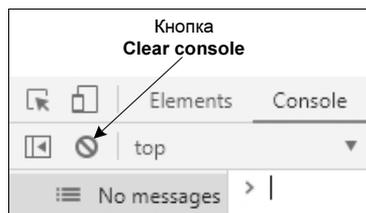


Рис. 1.5. Кнопка очистки консоли **Clear console**

1.2. Типы данных и операторы

Любое значение, обрабатываемое JavaScript, относится к определенному типу данных.

|| *Тип данных* — определяет характер значения и набор выполняемых над ним операций.

В разд. 1.1 мы познакомились с тремя типами данных (табл. 1.1).

Таблица 1.1. Три типа данных JavaScript

Числовой	Строковый	Логический
Обычные числа	Строки из любых символов	Логические значения
14, 27, 10000	'Java', 'Script', 'HTML и CSS'	true — «истина» false — «ложь»

Там же, в разд. 1.1, для обработки значений мы использовали операторы.

|| *Оператор* — выполняет элементарное действие над заданными значениями (*операндами*): сложение, объединение строк, сравнение и пр. — и производит выдачу (*возврат*) результата.

Примеры:

```
> 27 + 14      Оператор сложения + складывает операнды 27 и 14
< 41          После чего возвращает результат — сумму 41

> 10 < 20     Оператор «меньше» < сравнивает операнды 10 и 20
< true       И возвращает результат — логическое значение true
```

Некоторые операторы JavaScript не принимают операндов и (или) не возвращают результат.

1.3. Числа, арифметические операторы и приоритет операторов

Числа в языке JavaScript записываются по обычным (школьным) правилам. Но в дробных числах (их еще называют *вещественными* и *числами с плавающей точкой*) в качестве десятичного разделителя применяется точка (не запятая!).

Примеры:

Целые числа: 10, 20, 1234567

Вещественные числа: 1.5, 23.67, 0.053

У отрицательных чисел в начале ставится знак «минус»: -. Для записи чисел в экспоненциальной форме: $\langle \text{мантисса} \rangle \times 10^{\langle \text{порядок} \rangle}$ — применяется формат:

$\langle \text{мантисса} \rangle \text{e} \langle \text{порядок} \rangle$

Примеры:

Отрицательные числа:
-10, -20, -0.32, -0.053

Числа в экспоненциальной форме:
1e6 (1000000), 1.3e-5 (0,000013), -1e3 (-1000)

Помимо десятичных, можно записывать восьмеричные и шестнадцатеричные числа, предварив их, соответственно, нулем или символами 0x.

Примеры:

Восьмеричные числа:
010 (8), 0123 (83), -045 (-37)

Шестнадцатеричные числа:
0x10 (16), 0x53 (83), -0x321 (-801)

Для выполнения арифметических действий применяются *арифметические операторы*.

|| *Бинарные* (принимающие два операнда) операторы записываются в формате:
<операнд 1> <оператор> <операнд 2>

ПОЯСНЕНИЕ

Операторы бывают *унарные* (принимающие один операнд), *бинарные* (принимающие два операнда) и *тернарные* (принимающие три операнда). Обо всех их подробно рассказывается по мере их упоминания.

Поддерживаемые JavaScript арифметические операторы приведены в табл. 1.2.

Таблица 1.2. Поддерживаемые JavaScript арифметические операторы

Оператор	Описание	Пример	Результат
+ (плюс)	Сложение	> 27 + 14	< 41
- (минус)	Вычитание	> 40 - 100	< -60
* (звездочка)	Умножение	> 12 * 32	< 384
/ (слеш)	Деление	> 3 / 4	< 0.75
% (процент)	Остаток от деления	> 4 % 3	< 1
**	Возведение в степень	> 2 ** 10	< 1024

|| Очередность выполнения операторов задается их *приоритетом*: операторы с бóльшим приоритетом выполняются раньше операторов с меньшим приоритетом.

Из рассмотренных ранее операторов наибольший приоритет имеет возведение в степень: **, операторы умножения: *, деления: / и остатка от деления: % — меньший, а сложения: + и вычитания: - — еще меньший (таблицу с приоритетами разных операторов можно найти в *приложении 1*).

◆ В следующем примере сначала будет выполнено деление 82 на 2, а потом полученное частное сложится с 37:

```
> 37 + 82 / 2
< 78
```

- ◆ Операторы с одинаковым приоритетом выполняются в порядке слева направо, поэтому в следующем примере сначала будет выполнено сложение 10 и 34, а потом из суммы вычтется 26:

```
> 10 + 34 - 26,  
< 18
```

- ◆ Порядок выполнения операторов можно менять с помощью обычных круглых скобок. Оператор в круглых скобках будет выполнен раньше, независимо от его приоритета. Поэтому здесь сначала будет выполнено сложение 37 и 82, а потом полученная сумма разделится на 2:

```
> (37 + 82) / 2  
< 59.5
```

К числам также относятся три специальных значения:

- ◆ NaN (Not a Number, не число) — значение, получаемое при попытке, например, умножить число на строку (о строках будет рассказано позже):

```
> 20 * 'JavaScript'  
< NaN
```

- ◆ Infinity — математическая «бесконечность»: ∞ . Это значение можно получить делением положительного числа на 0:

```
> 10 / 0  
< Infinity
```

- ◆ -Infinity — математическая «минус бесконечность»: $-\infty$. Может быть получена делением отрицательного числа на 0.

1.4. Строки. Конкатенация строк

|| *Строки* заключаются в одинарные или двойные прямые кавычки (автор книги предпочитает одинарные).

Примеры строк: 'Java', "Script", 'Языки HTML и CSS применяются в веб-разработке'

Если строка заключена в одинарные кавычки, в ней не допускаются символы одинарной кавычки. Если строка заключена в двойные кавычки, в ней не допускаются символы двойной кавычки. В любом случае в строках не допускаются символы обратного слеша.

Примеры:

Правильно:
"Шеймус О'Брайен"
'Суши-бар "Йокогама"'

Неправильно:
'Шеймус О'Брайен'
"Суши-бар "Йокогама"'"

Указанные запреты можно обойти, применив *строковые литералы* (специальные последовательности символов): \', \" — для кавычек и \\ — для обратного слеша.

Примеры использования литералов:

```
'Шеймус О\Брайен'
"Суши-бар \"Йокогама"
```

Последовательность литералов `\r\n` вставляет в строковое значение разрыв строки.

Оператор `+`, будучи примененным к операндам-строкам, выполняет объединение (*конкатенацию*) строк в одну.

Объединяем строки `'Java'` и `'Script'`:

```
> 'Java' + 'Script'
< "JavaScript"
```

1.5. Условия и операторы сравнения

Для того чтобы сравнить какие-либо значения, применяется условие.

|| *Условие* — языковая конструкция, сравнивающая заданные значения. Записывается с помощью оператора сравнения.

|| *Оператор сравнения* — сравнивает два операнда и возвращает результат в виде логического значения `true` («истина») или `false` («ложь»).

Операторы сравнения являются *бинарными* и записываются в формате:

```
<операнд 1> <оператор> <операнд 2>
```

Все поддерживаемые JavaScript операторы сравнения приведены в табл. 1.3.

Таблица 1.3. Поддерживаемые JavaScript операторы сравнения

Оператор	Описание	Пример	Результат
<code>==</code>	Равно ¹	<code>> 1 == 1.000</code>	<code>< true</code>
<code>!=</code>	Не равно	<code>> 1 != 1.001</code>	<code>< true</code>
<code><</code>	Меньше	<code>> 1000 < 100</code>	<code>< false</code>
<code><=</code>	Меньше или равно	<code>> 100 <= 1000</code>	<code>< true</code>
<code>></code>	Больше	<code>> 10 > 9</code>	<code>< true</code>
<code>>=</code>	Больше или равно	<code>> 1e6 >= 1e9</code>	<code>< false</code>
<code>===</code>	Строго равно		
<code>!==</code>	Строго не равно		

Два последних оператора мы рассмотрим в конце урока, когда будем разбираться с преобразованием типов.

¹ Два знака «равно». Один знак «равно» — это совсем другой оператор, который будет рассмотрен позже.

1.6. Логические операторы

|| *Логический оператор* — принимает логические операнды и возвращает результат также в виде логического значения.

В JavaScript имеются три логических оператора:

- ◆ **!** (*логическое отрицание*, или *логическое НЕ*) — унарный, возвращает `false`, если *операнд* имеет значение `true`, и `true` в противном случае. Записывается этот оператор в формате: `!<операнд>` без пробела. Пример:

```
> !(10 == 100)
< true
```

В выражениях с оператором `!` операторы сравнения всегда следует брать в круглые скобки. Приоритет оператора `!` выше, чем у операторов сравнения, следовательно, он будет выполнен первым, и результат без скобок окажется неверным;

- ◆ **&&** (*логическое умножение*, или *логическое И*) — возвращает `true`, если оба *операнда* равны `true`, и `false` в противном случае;
- ◆ **||** (*логическое сложение*, или *логическое ИЛИ*) — возвращает `true`, если хотя бы один *операнд* равен `true`, и `false` в противном случае.

Два последних оператора записываются в формате:

```
<операнд 1> <оператор && или ||> <операнд 2>
```

Примеры:

<code>> 10 <= 100 && 0 != 1</code>	<code>> 10 <= 100 0 != 1</code>
<code>< true</code>	<code>< true</code>
<code>> 10 >= 100 && 0 != 1</code>	<code>> 10 >= 100 0 != 1</code>
<code>< false</code>	<code>< true</code>
<code>> 10 <= 100 && 0 == 1</code>	<code>> 10 <= 100 0 == 1</code>
<code>< false</code>	<code>< true</code>
<code>> 10 >= 100 && 0 == 1</code>	<code>> 10 >= 100 0 == 1</code>
<code>< false</code>	<code>< false</code>

1.7. Переменные

|| *Переменная* — ячейка памяти, предназначенная для временного хранения какого-либо значения. Должна иметь уникальное имя, по которому выполняется обращение к этой переменной.

|| *Имя переменной* может содержать буквы латиницы, цифры и символы подчеркивания `_`, причем оно **не** должно начинаться с цифры. Пробелы в имени переменной не допускаются.

Примеры:

Правильные имена:
a, b, someValue, _private

Неправильные имена:
число, some value, 234value

Перед первым использованием переменной ее следует объявить.

|| *Объявление переменной* — указание создать переменную с заданным именем. Должно предшествовать первому использованию переменной.

Сразу при объявлении переменной она получает значение `undefined`, обозначающее отсутствие какой-либо значащей величины.

Оператор объявления переменных `let` объявляет переменные с указанными именами:

```
let <список имен объявляемых переменных через запятую>
```

В качестве результата оператор `let` всегда возвращает значение `undefined`.

Для объявления переменных можно использовать еще два оператора, которые мы рассмотрим в конце этого раздела.

Объявляем переменную с именем `sum` (и получим результат `undefined`):

```
> let sum  
< undefined
```

ПРИМЕЧАНИЕ

В дальнейшем результат `undefined`, возвращаемый оператором `let`, ради компактности указываться не будет.

Объявляем сразу три переменные:

```
> let x, y, x
```

Объявив переменную, мы можем присвоить ей сохраняемое значение.

|| *Присваивание* — занесение значения в переменную. Выполняется *оператором присваивания* `=`¹, имеющим формат:
|| `<переменная> = <значение>`

Присваиваем только что объявленной переменной `sum` значение `10`:

```
> sum = 10  
< 10
```

В качестве результата оператор `=` возвращает присвоенное переменной значение.

Можно совместить объявление переменной с присваиванием ей значения — так мы сократим код. Для этого в операторе `let` вместо имени переменной нужно указать оператор присваивания.

Объявляем переменную `a` и сразу же присваиваем ей число `1`:

```
> let a = 1
```

Сохраненное в переменной значение можно использовать в вычислениях, выполнив обращение к этой переменной.

¹ Оператор присваивания обозначается одним знаком «равно» (`=`), а оператор сравнения «равно» — двумя (`==`). Не путайте эти два оператора!

Обращение к переменной — указание извлечь хранящееся в переменной значение и использовать его в качестве операнда какого-либо оператора или иным образом (например, для вывода на экран).

Обращение к переменной выполняется простым указанием имени нужной переменной в качестве операнда.

Присваиваем переменной `a` произведение значения переменной `sum` (в текущий момент: 10) и числа 20:

```
> a = sum * 20
< 200
```

ВНИМАНИЕ!

При присваивании переменной нового значения старое значение теряется.

Прибавляем к значению переменной `sum` (в текущий момент: 10) число 20 и заносим получившуюся сумму в ту же переменную:

```
> sum = sum + 20
< 30
```

Оператор комбинированного присваивания (табл. 1.4) извлекает из указанной переменной значение, выполняет над ним и заданным операндом какое-либо действие и присваивает результат той же переменной. Формат этого оператора:

```
<переменная> <оператор> <операнд>
```

Таблица 1.4. Операторы комбинированного присваивания

Оператор комбинированного присваивания	Его аналог
<code>a += b</code>	<code>a = a + b</code>
<code>a -= b</code>	<code>a = a - b</code>
<code>a *= b</code>	<code>a = a * b</code>
<code>a /= b</code>	<code>a = a / b</code>
<code>a %= b</code>	<code>a = a % b</code>

Прибавим к переменной `sum` (сейчас там 30) число 70:

```
> sum += 70
< 100
```

Операторы комбинированного присваивания выполняются быстрее аналогичных комбинаций арифметического оператора и обычного оператора присваивания.

1.7.1. Три оператора объявления переменных

Язык JavaScript поддерживает три *оператора объявления переменных*: уже знакомый нам `let`, а также `const` и `var`. Они имеют один и тот же формат записи, но разные особенности:

- ◆ `let` — объявляет обычную переменную, которой можно присваивать значения сколько угодно раз. Повторное объявление переменной этим оператором не допускается — оно вызовет ошибку:

```
> let x
> let x
< Uncaught SyntaxError: Identifier 'x' has already
    been declared at <anonymous>:1:1
```

Здесь в последних двух строках выведено сообщение об ошибке;

- ◆ `const` — объявляет *константу* — переменную, значение которой можно присвоить лишь один раз и лишь непосредственно при ее объявлении. Попытка присвоить константе новое значение вызовет ошибку. Пример:

```
> const y = 2
```

Если значение переменной не меняется в дальнейшем, рекомендуется объявлять ее как константу, поскольку константы обрабатываются веб-обозревателем быстрее;

- ◆ `var` — полностью аналогичен оператору `let`, однако позволяет объявить переменную повторно (в этом случае он ничего не делает). Есть еще один нюанс, который мы рассмотрим в *уроке 2*.

Оператор `var` остался от старых версий JavaScript. В настоящее время его применение не рекомендовано, однако он часто встречается в старом JavaScript-коде.

1.7.2. Еще об объявлении переменных

Объявлять переменные перед их использованием необязательно. Можно просто присвоить переменной какое-либо значение — и веб-обозреватель сам неявно объявит эту переменную:

```
sum = 10
```

Однако все же лучше объявлять переменные явно. Во-первых, это хороший стиль программирования, а во-вторых, при работе в строгом режиме (о котором будет рассказано в *уроке 2*) веб-обозреватель будет требовать явного объявления переменных.