Содержание

Как пользоваться этой книгой

С помощью примеров вы узнаете, как использовать встроенные функции JavaScript, поддерживаемые современными веб-браузерами, а снимки экрана проиллюстрируют результаты, полученные с помощью примеров кода. Необходимые фрагменты кода выделены цветом.

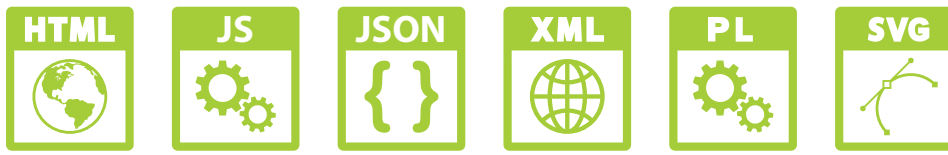
Синим цветом выделен JavaScript-код; красным — имена, присвоенные программистом; черным — текст; зеленым — комментарии к коду:

```
let sum = ( 9 + 12 ) / 3 // Эквивалентно 21 / 3.
document.getElementById( 'info' ).innerHTML += 'Grouped sum: ' + sum
```

Синим цветом выделены HTML-теги; черным — текст; оранжевым — значения атрибутов элементов в HTML- и JavaScript-коде:

```
<p id="info">JavaScript in easy steps</p>
```

Кроме того, для идентификации каждого исходного файла, описанного в пошаговых инструкциях, на полях указаны значки и имена соответствующих файлов:



page.html data.json external.js data.xml echo.pl banner.svg

Чтобы избежать повторений, исходный код HTML-документов, приведенных в примерах, указан не полностью. Однако он представляет собой весь фрагмент документа, к которому применяется указанный JavaScript-код. Вы можете скачать ZIP-архив, содержащий все полные файлы исходных кодов, выполнив следующие простые шаги:

1. Перейдите на сайт http://addons.eksmo.ru/it/js.zip — загрузка начнется автоматически
2. Извлеките содержимое архива в любое удобное место на вашем компьютере.

Если вы не довольны полученным результатом, просто сравните свой код с кодом исходных файлов.

1 Введение в JavaScript

Добро пожаловать в удивительный и захватывающий мир программирования на JavaScript! В этой главе вы узнаете, как добавлять сценарии в HTML-документы, используя переменные и функции JavaScript.

Знакомство с JS

Язык программирования JavaScript («JS») — это объектно-ориентированный язык сценариев, объекты которого встроены в программное обеспечение веб-браузера, например, Google Chrome, Microsoft Edge, Firefox, Opera и Safari. Для обеспечения функциональности это позволяет при загрузке страницы в браузере интерпретировать содержащиеся на веб-странице сценарии. В целях безопасности язык JavaScript не может считывать или записывать файлы, за исключением файлов cookie, в которых хранится минимальный объем данных.

Самая первая реализация JavaScript была создана Бренданом Эйхом (Айком) (Brendan Eich) в компании Netscape. Этот язык программирования был впервые представлен в декабре 1995 года и первоначально назывался «LiveScript». Однако вскоре из коммерческих соображений LiveScript был переименован в JavaScript, получив соответствующую лицензию у Sun Microsystem.

Брендан Эйх, создатель языка программирования JavaScript, а также соучредитель проекта Mozilla, помог запустить веб-браузер Firefox.

До введения JavaScript браузер вызывал сценарии на стороне сервера, а в связи с долгим ответом снижалась производительность. Эта проблема решилась с помощью вызова сценариев на стороне клиента.

Популярность языка JavaScript быстро росла. Но между компаниями Netscape и Microsoft возникли разногласия по поводу его лицензирования, поэтому Microsoft представила собственную версию языка под названием «JScript». Несмотря на то что новая версия JScript очень похожа на JavaScript, она имеет некоторые расширенные функции. В июне 1997 года Ecma International предложила стандартизировать JavaScript, и в результате появился «ECMAScript». Это помогло стабилизировать основные функции. Однако название не прижилось среди юзеров, поэтому язык программирования все же получил название JavaScript.

В книге представлены:

- Основы языка — синтаксис, ключевые слова, операторы, структура и встроенные объекты.

- Функциональность веб-страницы показывает, как объектная модель документа (DOM) браузера обеспечивает взаимодействие с пользователем.

- Веб-приложения — адаптивные веб-приложения и текстовый формат представления данных в нотации объекта JavaScript (JSON).

Добавление JavaScript в HTML-документ

В теге <script> применим атрибут type="text/javascript". Однако это не требуется, поскольку JavaScript — язык сценариев для HTML по умолчанию.

Вы можете добавить JavaScript-код в HTML-документ, поместив его между тегами **<script>** и **</script>**, например:

```
<script>
document.getElementById( 'message' ).innerText = 'Hello World!'
</script>
```

HTML-документ может включать в себя несколько сценариев, которые помещаются в раздел заголовка или тела документа. Однако рекомендуется размещать сценарии в конце основного раздела (непосредственно перед закрывающим тегом **</body>**), чтобы браузер мог отобразить веб-страницу до выполнения сценария.

Также в HTML-документ можно добавить JavaScript код, расположенный во внешнем файле с расширением.**js**. Это позволяет нескольким различным веб-страницам вызывать один и тот же сценарий. Подключение внешнего сценария выполняется с помощью атрибута **src** тега **<script>** следующим образом:

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

Не включайте теги <script> и </script> во внешний файл JavaScript. В них необходимо добавлять только код сценария.

Этот код также можно поместить в раздел заголовка или тела документа, и браузер будет рассматривать сценарий так, как если бы он был написан непосредственно в этой позиции в HTML-документе.

Если атрибуту src тега **<script>** присваивается только имя файла внешнего сценария, необходимо, чтобы файл сценария находился в одной папке (каталоге)

Внешние файлы сценариев упрощают сопровождение кода. Почти все примеры, представленные в этой книге, автономные, поэтому код сценария можно вставлять между тегами в HTML-документе.

с HTML-документом. Если сценарий находится в другом месте, вы можете указать относительный путь к файлу, например:

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

Если сценарий расположен в другом месте сайта, можно указать абсолютный путь к файлу, например:

```
<script src="https://www.example.com/js/external_script.js">
</script>
```

Кроме того, можно указать содержимое, которое будет отображаться на веб-странице только в том случае, если пользователь в своем веб-браузере отключил JavaScript, включив в HTML-документ элемент **<noscript>**, например:

```
<noscript>JavaScript is Not Enabled!</noscript>
```

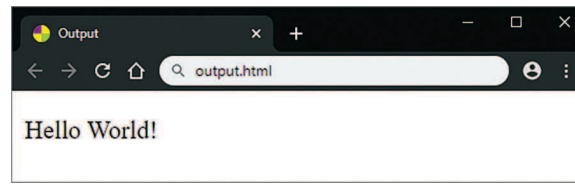
Вывод JavaScript

В JavaScript существует несколько вариантов отображения данных. Например:

```
document.getElementById( 'message' ).innerText = 'Hello World!'
```

Обратите внимание на использование оператора. (точка) в описании свойств или методов объекта с использованием «точечной нотации».

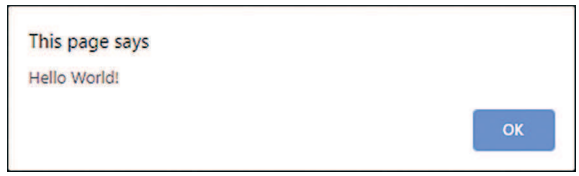
Атрибут **id** определяет уникальный идентификатор HTML-элемента. Свойство **innerText** определяет содержимое HTML-документа.



Также для отображения данных можно использовать всплывающее окно сообщений.

```
window.alert( 'Hello World!' )
```

Метод **alert( )** выводит на экран диалоговое окно с сообщением, указанным в коде в круглых скобках ().



В процессе изучения языка JavaScript предпочтительнее выводить данные в консоль браузера, например:

```
console.log( 'Hello World!' )
```

Метод **log( )** объекта **console** выводит в окно консоли содержимое, указанное в круглых скобках ( ). Фактически в любом браузере есть специальная панель веб-разработчика, доступ к которой осуществляется нажатием клавиши F12. Поскольку Google Chrome — самый популярный браузер на момент написания книги, я использовал его для демонстрации JavaScript, а консоль этого браузера используется для отображения данных.

1. Создайте HTML-документ, содержащий пустой абзац и программный код для отображения данных тремя способами.

```
<p id="message"></p>
<script>
document.getElementById( 'message' ).innerText =
                                        'Hello World!'
window.alert( 'Hello World!' )
console.log( 'Hello World!' )
</script>
```

2. Сохраните HTML-документ, затем откройте его в браузере, чтобы проанализировать полученный результат, как показано на рисунке ниже.

3. Нажмите клавишу **F12** или воспользуйтесь меню браузера, чтобы открыть **Developers Tools** (Инструменты разработчика).

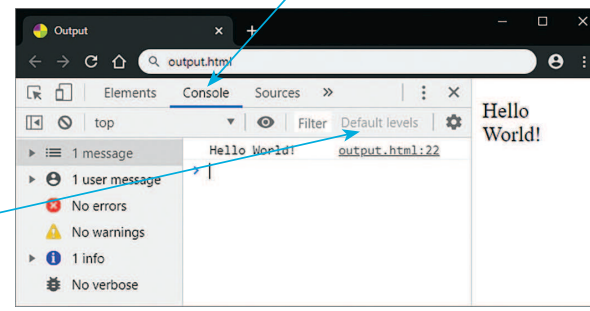
Существует также метод **document.write( )**, который перезаписывает весь заголовок и тело веб-страницы. Стоит отметить, что его использование считается плохой практикой.

При возникновении ошибок в коде консоль предоставляет полезные сообщения, что очень удобно для отладки программы.

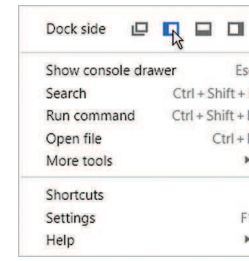
output.html

Убедитесь, что в консоли отображается содержимое, а также имя HTML-документа и номер строки, где есть соответствующий JavaScript-код.

4 Выберите вкладку **Console** (Консоль), чтобы увидеть содержимое, записанное в консоли.



5 Нажмите кнопку **Show/Hide** (Показать/Скрыть), чтобы скрыть или показать боковую панель. Нажмите кнопку **Customize** (Настроить), чтобы выбрать способ закрепления консоли в окне браузера. Затем нажмите кнопку **Clear** (Очистить), чтобы очистить все содержимое консоли.



Структура кода

Более подробную информацию о ключевых словах вы можете найти на стр. 17–18, а об операторах, значениях и выражениях вы узнаете позже.

JavaScript-код состоит из ряда инструкций, называемых «операторами». Обычно все инструкции на странице выполняются сверху вниз.

Каждая инструкция может содержать любой из следующих элементов:

- Ключевые слова — слова, имеющие особое значение в языке JavaScript.
- Операторы — специальные символы, выполняющие операции с одним или несколькими операндами.
- Значения — текстовые строки, числа, логическое значение **true** или **false**, значения **undefined** и **null**.

- Выражения — любая часть исходного кода программы, которая вычисляет значение.

Ранее в коде JavaScript каждый оператор должен был заканчиваться символом ; (точка с запятой), а каждое предложение — символом . (точка) в английской раскладке. Теперь это необязательно, поэтому, если вы не хотите писать несколько операторов в одной строке, то это правило можно опустить. В таком случае операторы необходимо разделять точкой с запятой, например:

Результат вычисления выражения — это значение, тогда как оператор выполняет действие.

```
оператор; оператор; оператор
```

Некоторые разработчики JavaScript по-прежнему предпочитают заканчивать каждый оператор символом ; (точка с запятой). В приведенных в книге примерах он опускается, но выбор остается за вами.

Интерпретатор JavaScript игнорирует отступы и пробелы. Поэтому, чтобы улучшить читабельность кода, необходимо использовать пробелы. Например, при сложении двух чисел:

```
total = 100 + 200 вместо total=100+200
```

Для удобства и читабельности кода создавайте отступы с помощью клавиши Пробел, так как при отладке кода в текстовых редакторах отступы, созданные клавишей Tab, могут обрабатываться по-разному.

Операторы JavaScript обычно сгруппированы с помощью фигурных скобок { }, разделяющих код на функциональные блоки, которые можно при необходимости многократно вызывать. Для удобства и читабельности кода рекомендуется делать отступы двумя пробелами, например:

```
{
  оператор
  оператор
  оператор
}
```

Синтаксис JavaScript — это набор правил, по которым строится программа. В JavaScript существуют два типа значений: фиксированные и переменные. Фиксированные числовые и текстовые строковые значения называются литералами:

Для использования строковых литералов рекомендуется выбрать один вид кавычек и придерживаться его. В наших примерах используются одинарные кавычки.

- Числовые литералы — целые числа, например 100, или числа с плавающей запятой, например 3,142.
- Строковые литералы — строка символов, заключенная в двойные кавычки, например **"JavaScript Fun"**, или одинарные кавычки, например **'JavaScript Fun'**.

Переменные — это некий контейнер, внутри которого можно хранить значения для дальнейшего использования в программе. В JavaScript предусмотрен способ объявления переменных с помощью ключевого слова **let**. Например, с помощью кода **let total** создается переменная с именем «total». Переменной можно присвоить значение, используя оператор присваивания **=**, например:

```
let total = 300
```

Прочие операторы JavaScript используются для создания выражений, которые будут вычислять одно значение. Как правило, выражение заключается в круглые скобки **( )**. Например, приведенное ниже выражение содержит числа и оператор сложения **+** и вычисляет одно значение 100:

```
( 80 + 20 )
```

Также выражения могут содержать значения переменных. Например, для вычисления одного значения 100 выражения могут содержать предыдущее значение переменной, оператор вычитания и число:

```
( total – 200 )
```

Иногда возникает необходимость закомментировать строки кода, чтобы предотвратить их выполнение при отладке.

JavaScript чувствителен к регистру символов, поэтому переменные с именами **total** и **Total** — это две совершенно разные переменные.

Хорошей практикой считается добавление к коду пояснительных комментариев. Они делают код более понятным для других пользователей, а также для вас при его повторном просмотре. Все, что находится в одной строке после символа **//** или между символов **/*** и ***/** в одной или нескольких строках, будет проигнорировано.

```
let total = 100 // Этот код БУДЕТ выполнен.
/* let total = 100
   Этот код НЕ БУДЕТ выполнен. */
```

Избегайте ключевых слов

В программе для переменных и функций вы можете указывать собственные имена. Лучше всего давать переменным осмысленные имена, которые будут отражать суть переменной или функции. Имя переменной может содержать буквы, цифры и символы подчеркивания, но никогда не начинается с цифры. Также в нем не используются пробелы. Запрещается использовать в качестве имен переменных приведенные ниже в таблице ключевые слова, которые имеют особое значение:

Ключевые (зарезервированные) слова JavaScript			
abstract	arguments	await	boolean
break	byte	case	catch
char	class	const	continue
debugger	default	delete	do
double	else	enum	eval
export	extends	false	final
finally	float	for	function
goto	if	implements	import
in	instanceof	int	interface
let	long	native	new
null	package	private	protected
public	return	short	static
super	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

Объекты, свойства и методы JavaScript			
Array	Date	eval	function
hasOwnProperty	Infinity	isFinite	isNaN
isPrototypeOf	length	Math	NaN
name	Number	Object	prototype
String	toString	undefined	valueOf

HTML-имена, объекты окна и свойства			
alert	all	anchor	anchors
area	assign	blur	button
checkbox	clearInterval	clearTimeout	clientInformation
close	closed	confirm	constructor
crypto	decodeURI	decodeURIcomponent	defaultStatus
document	element	elements	embed
embeds	encodeURI	encodeURIcomponent	escape
event	fileUpload	focus	form
forms	frame	innerHeight	innerWidth
layer	layers	link	location
mimeTypes	navigate	navigator	frames
frameRate	hidden	history	image
images	offscreenBuffering	open	opener
option	outerHeight	outerWidth	packages
pageXOffset	pageYOffset	parent	parseFloat
parseInt	password	pkcs11	plugin
prompt	propertyIsEnum	radio	reset
screenX	screenY	scroll	secure
select	self	setInterval	setTimeout
status	submit	taint	text
textarea	top	unescape	untaint
window			

Атрибуты событий в HTML Например:			
onclick	ondblclick	onfocus	onfocusout
onkeydown	onkeypress	onkeyup	onload
onmousedown	onmouseup	onmouseover	onmouseout
onmousemove	onchange	onreset	onsubmit