ECMAScript 5, если функция определяется в строгом режиме, при вызове она получает в ключевом слове this не глобальный объект, а значение undefined. Подробнее о строгом режиме рассказывается в разделе 5.7.3.

4.6. Выражения создания объектов

Выражение *создания объекта* создает новый объект и вызывает функцию (называемую конструктором) для инициализации свойств этого объекта. Выражения создания объектов похожи на выражения вызова, за исключением того, что им предшествует ключевое слово new:

```
new Object()
new Point(2,3)
```

Если в выражении создания объекта функции-конструктору не передается ни одного аргумента, пустую пару круглых скобок можно опустить:

```
new Object
new Date
```

При вычислении выражения создания объекта интерпретатор JavaScript сначала создает новый пустой объект, как если бы для создания использовался пустой инициализатор объекта {}, а затем вызывает указанную функцию с указанными аргументами, передавая ей новый объект в качестве значения ключевого слова this. Функция может использовать его для инициализации свойств только что созданного объекта. Функции, которые создаются специально, чтобы играть роль конструктора, не должны возвращать значение, а значением выражения создания объекта становится созданный и инициализированный объект. Если конструктор возвращает какой-либо объект, этот объект становится значением всего выражения создания объекта, а вновь созданный объект уничтожается.

Более подробно конструкторы описываются в главе 9.

4.7. Обзор операторов

В языке JavaScript операторы используются в арифметических выражениях, выражениях сравнения, логических выражениях, выражениях присваивания и т. д. Перечень операторов приводится в табл. 4.1, которую можно использовать как справочник.

Таблица 4.1. Операторы JavaScript

Оператор	Операция	A	N	Типы значений
++	Префиксный и постфиксный инкремент	R	1	левостороннее выражение → число
--	Префиксный и постфиксный декремент	R	1	левостороннее выражение → число
-	Унарный минус	R	1	число → число
+	Преобразование в число	R	1	число → число
~	Поразрядная инверсия	R	1	целое → целое

Оператор	Операция	A	N	Типы значений
`!`	Логическая инверсия	R	1	логическое → логическое
`delete`	Удаление свойства	R	1	левостороннее выражение → логическое
`typeof`	Определение типа операнда	R	1	любое → строка
`void`	Возврат неопределенного значения	R	1	любое → `undefined`
`*`, `/`, `%`	Умножение, деление, деление по модулю	L	2	число, число → число
`+`, `-`	Сложение, вычитание	L	2	число, число → число
`+`	Конкатенация строк	L	2	строка, строка → строка
`<<`	Сдвиг влево	L	2	целое, целое → целое
`>>`	Сдвиг вправо с сохранением знака	L	2	целое, целое → целое
`>>>`	Сдвиг вправо с заполнением нулями	L	2	целое, целое → целое
`<`, `<=`, `>`, `>=`	Сравнение числовых значений	L	2	число, число → логическое
`<`, `<=`, `>`, `>=`	Сравнение строк	L	2	строка, строка → логическое
`instanceof`	Проверка на принадлежность классу	L	2	объект, функция → логическое
`in`	Проверка наличия свойства	L	2	строка, объект → логическое
`==`	Проверка равенства	L	2	любое, любое → логическое
`!=`	Проверка неравенства	L	2	любое, любое → логическое
`===`	Проверка идентичности	L	2	любое, любое → логическое
`!==`	Проверка неидентичности	L	2	любое, любое → логическое
`&`	Поразрядное И	L	2	целое, целое → целое
`^`	Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ	L	2	целое, целое → целое
`|`	Поразрядное ИЛИ	L	2	целое, целое → целое
`&&`	Логическое И	L	2	любое, любое → любое
`||`	Логическое ИЛИ	L	2	любое, любое → любое
`?:`	Выбор второго или третьего операнда	R	3	логическое, любое, любое → любое
`=`	Присваивание переменной или свойству	R	2	левостороннее выражение, любое → любое
`*=`, `/=`, `%=`, `+=`, `-=`, `&=`, `^=`, `|=`, `<<=`, `>>=`, `>>>=`	Операция с присваиванием	R	2	левостороннее выражение, любое → любое
`,`	Отбросить первый операнд, вернуть второй	L	2	любое, любое → любое

Обратите внимание, что большинство операторов обозначаются символами пунктуации, такими как `+` и `=`, а некоторые — ключевыми словами, например `delete` и `instanceof`. И ключевые слова, и знаки пунктуации обозначают обычные операторы, просто первые имеют менее лаконичный синтаксис.

Операторы в табл. 4.1 перечислены в порядке их приоритетов. Операторы, перечисленные первыми, имеют более высокий приоритет. Операторы, отделенные