3.8.3. Преобразование объектов в простые значения

Книга: Флэнаган - Javascript

Преобразование объектов в логические значения

Преобразование объектов в логические значения выполняется очень просто: все объекты (включая массивы и функции) преобразуются в значение **true**. Это справедливо и для объектов-оберток: результатом вызова new Boolean(false) является объект, а не простое значение, поэтому он также преобразуется в значение true.

Преобразование объектов в строку

Преобразование объекта в строку и преобразование объекта в число выполняется вызовом соответствующего метода объекта. Все осложняется тем, что объекты в языке JavaScript имеют два разных метода для выполнения преобразований, а также наличием нескольких специальных случаев, описываемых ниже. Обратите внимание, что правила преобразования объектов в строки и числа, описываемые здесь, применяются только к объектам самого языка JavaScript. Объекты среды выполнения (например, определяемые веб-броузерами) могут предусматривать собственные алгоритмы преобразования в числа и строки.

Все объекты наследуют два метода преобразования. Первый из них называется **toString(),** он возвращает строковое представление объекта. По умолчанию метод toString() не возвращает ничего особенно интересного (хотя эта информация иногда может оказаться полезной, как будет показано в примере 6.4):

```
({x:1, y:2}).toString() // => "[object Object]"
```

Многие классы определяют более специализированные версии метода toString() .Например, метод toString() класса Array преобразует все элементы массива в строки и объединяет результаты в одну строку, вставляя запятые между ними.

Метод toString() класса Function возвращает строковое представление функции, зависящее от реализации. На практике обычно реализации преобразуют пользовательские функции в строки с исходным программным кодом на языке JavaScript.

Класс Date определяет метод toString(), возвращающий строку с датой и временем в удобочитаемом формате (который может быть разобран средствами JavaScript).

Класс RegExp определяет метод toString(), преобразующий объект RegExp в строку, которая выглядит как литерал регулярного выражения:

```
[1,2,3].toString() // => "1,2,3"
(function(x) { f(x); }).toString() // => "function(x) {\n f(x);\n}"
/\d+/g.toString() // => "/\\d+/g"
new Date(2010,0,1).toString() // => "Fri Jan 01 2010 00:00:00 GMT+0300"
```

Преобразование объектов в число

Другая функция преобразования объектов называется **valueOf().** Задача этого метода определена не так четко: предполагается, что он должен преобразовать объект в представляющее его простое значение, если такое значение существует. Объекты по своей природе являются составными значениями, и большинство объектов не могут быть представлены в виде единственного простого значения, поэтому **по умолчанию** метод valueOf() возвращает не простое значение, а сам объект.

Классы-обертки определяют методы valueOf(), возвращающие обернутые простые значения.

Массивы, функции и регулярные выражения наследуют метод по умолчанию. Вызов метода valueOf() экземпляров этих типов возвращает сам объект.

Класс Date определяет метод valueOf(), возвращающий дату во внутреннем представлении: количество миллисекунд, прошедших с 1 января 1970 года:

```
var d = new Date(2010, 0, 1); // 1 января 2010 года, (время Московское)
d.valueOf() // => 1262293200000
```

Особенности преобразования объектов в строки и в числа

Теперь, разобравшись с методами toString() и valueOf(), можно перейти к обсуждению особенностей преобразования объектов в строки и в числа. Учтите, что существует несколько специальных случаев, когда JavaScript выполняет преобразование объектов в простые значения несколько иначе. Эти особые случаи рассматриваются в конце данного раздела.

Преобразование объектов в строку интерпретатор JavaScript выполняет в два этапа:

- Если объект имеет метод **toString(),** интерпретатор вызывает его. Если он возвращает **простое значение**, интерпретатор преобразует значение в строку (если оно не является строкой) и возвращает результат преобразования. Обратите внимание, что правила преобразований простых значений в строку четко определены для всех типов и перечислены в табл. 3.2.
- Если объект не имеет метода toString() или этот метод не возвращает простое значение, то интерпретатор проверяет наличие метода **valueOf().** Если этот метод определен, интерпретатор вызывает его. Если он возвращает **простое значение**, интерпретатор преобразует это значение в строку (если оно не является строкой) и возвращает результат преобразования.
- **В противном случае** интерпретатор делает вывод, что ни toString(), ни valueOf() не позволяют получить простое значение и возбуждает исключение **TypeError**.

При преобразовании объекта в число интерпретатор выполняет те же действия, но первым пытается применить метод valueOf():

- Если объект имеет метод **valueOf(),** возвращающий простое значение, интерпретатор преобразует (при необходимости) это значение в число и возвращает результат.
- Иначе, если объект имеет метод **toString(),** возвращающий простое значение, интерпретатор выполняет преобразование и возвращает полученное значение.
- В противном случае возбуждается исключение **TypeError**.

Описанный алгоритм преобразования объекта в число объясняет, почему пустой массив преобразуется в число 0, а массив с единственным элементом может быть преобразован в обычное число. Массивы наследуют по умолчанию метод valueOf(), который возвращает сам объект, а не простое значение, поэтому при преобразовании массива в число интерпретатор опирается на метод toString(). Пустые массивы преобразуются в пустую строку. А пустая строка преобразуется в число 0. Массив с единственным элементом преобразуется в ту же строку, что и единственный элемент массива. Если массив содержит единственное число, это число преобразуется в строку, а затем опять в число.

Оператор + в языке JavaScript выполняет сложение чисел и конкатенацию строк. Если какой-либо из его операндов является объектом, JavaScript преобразует объект, используя специальное преобразование объекта в простое значение вместо преобразования объекта в число, используемого другими арифметическими операторами. То же относится и к оператору равенства ==. Если выполняется сравнение объекта с простым значением, оператор выполнит преобразование объекта с использованием правил преобразования в простое значение.

Преобразование объектов в простые значения, используемое операторами + и ==, предусматривает особый подход для объектов Date. Класс Date является единственным типом данных в базовом JavaScript, который определяет осмысленные преобразования и в строку, и в число.

Преобразование любого объекта, не являющегося датой, в **простое значение** основано на преобразовании в число (когда первым применяется метод **valueOf(**)), тогда как для объектов типа Date используется преобразование в строку (первым применяется метод toString()). Однако преобразование выполняется не совсем так, как было описано выше: простое значение, возвращаемое методом valueOf() или toString(), используется непосредственно, без дополнительного преобразования в число или в строку.

Оператор < и другие операторы отношений выполняют преобразование объектов в простые значения подобно оператору ==, но не выделяя объекты Date: для любого объекта сначала предпринимается попытка применить метод valueOf(), а затем метод toString(). Любое простое значение, полученное таким способом, используется непосредственно, без дальнейшего преобразования в число или в строку.

+, ==, != и операторы отношений являются единственными, выполняющими специальное преобразование строки в простое значение. Другие операторы выполняют более явные преобразования в заданный тип и не предусматривают специальной обработки объектов Date. Оператор -, например, преобразует свои операнды в числа.

Следующий фрагмент демонстрирует поведение операторов +, -, ==
и > при работе с объектами Date:

```
var now = new Date(); // Создать объект Date
typeof (now + 1) // => "строка": + преобразует дату в строку
typeof (now - 1) // => "число": - выполнит побразование объекта в число
now == now.toString() // => true: неявное и явное преобразование в строку
now > (now -1) // => true: > преобразует объект Date в число
```