JavaScript за 60 минут

Wikipedia

JavaScript— прототипно-ориентированный сценарный язык программирования. Является диалектом языка ECMAScript.

Современный JavaScript — это «безопасный» язык программирования общего назначения. Он не предоставляет низкоуровневых средств работы с памятью, процессором, так как изначально был ориентирован на браузеры, в которых это не требуется.

Wikipedia

Основные архитектурные черты:

01. динамическая типизация
02. слабая типизация
03. автоматическое управление памятью
04. прототипное программирование
05. функции как объекты первого класса

Типы данных

01. number
02. **string**
03. **boolean**
04. null
05. undefined
06. **object**

number

```
01. var n = 536;
02. n = 3.1415;
03. alert( 1 / 0 ); // Infinity
04. alert( "нечисло" * 2 ); // NaN, ошибка
```

boolean

```
01. var amIAlwaysRight = true;
02. var areYouAlwaysRight = false;
```

Только два значения - истина и ложь

undefined

```
01. var foo;
02. console.log(foo); // undefined
03. console.log(window.bar); // undefined
04. console.log(bar); // ???
```

В явном виде undefined обычно не присваивают, так как это противоречит его смыслу. Для записи в переменную «пустого значения» используется null.

null

```
var age = null;
```

В JS null не является «ссылкой на несуществующий объект» или «нулевым указателем», как в некоторых других языках. Это просто специальное значение, которое имеет смысл «ничего» или «значение неизвестно».

object

```
01. var foo = {bar: (new Date).getTime()};
02. console.log(foo.bar); // 1424115898995
```

Особняком стоит шестой тип: «объекты». К нему относятся, например, даты, он используется для коллекций данных и для многого другого.

Преобразование примитивов

01. Строковое преобразование

02. Числовое преобразование

03. Преобразование к логическому значению

Строковое

01. Нужна строка (alert(123))
02. Явный вызов **String**(123)
03. Оператор "+" с одним из аргументов - **String**

Числовое

01. Математические функции и выражения
02. Нестрогое сравнение (”==”, ”!=”)
03. Оператор ”+” перед выражением или явный вызов **Number**

Логическое

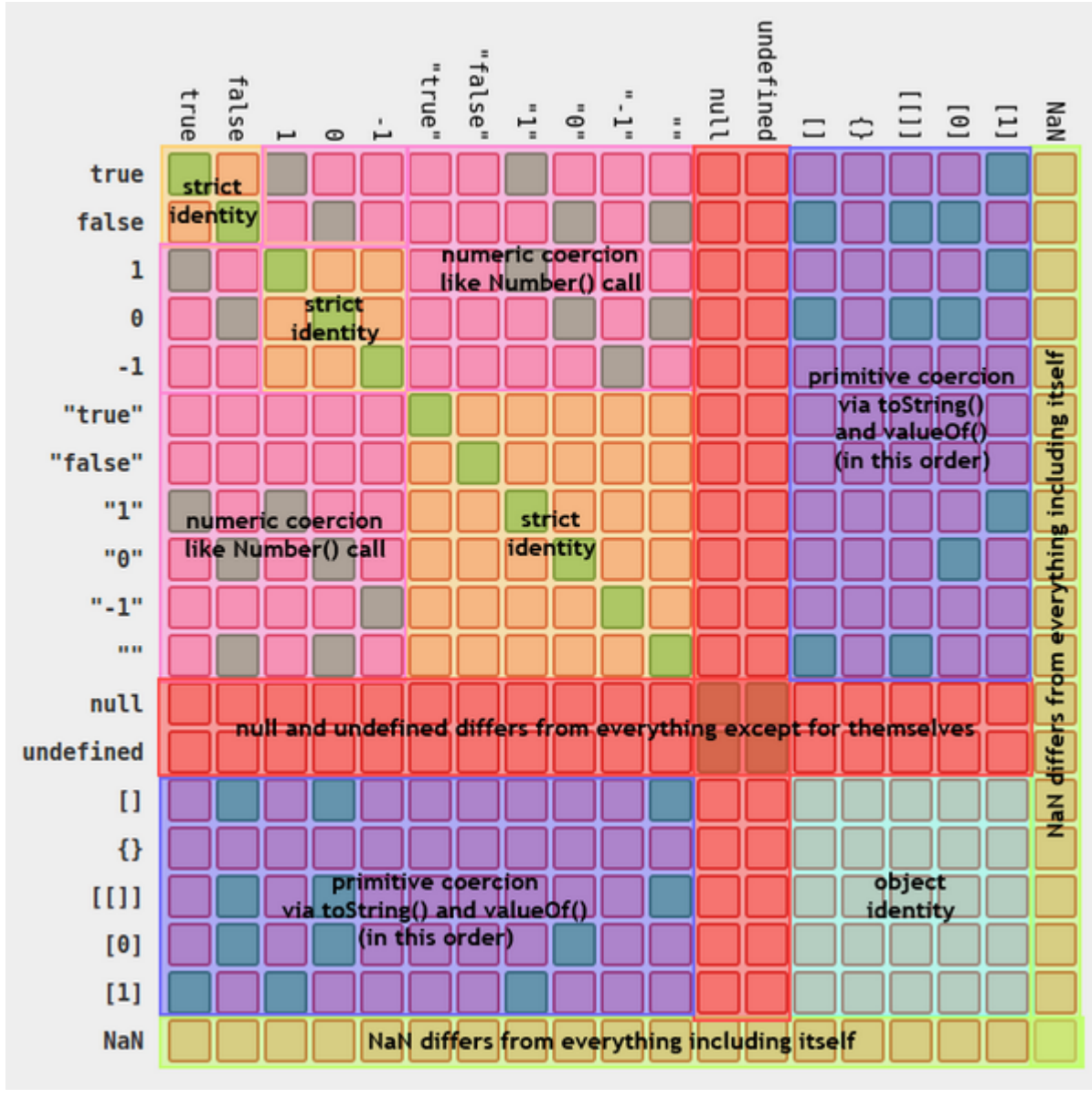
Преобразование к true/false происходит в логическом контексте, таком как if(obj), while(obj) и при применении логических операторов.

Значение	**Преобразуется в...**
Числа	Все true, кроме 0, NaN — false
Строки	Все true, кроме пустой строки "" — false
Объекты	Всегда true

WAAAT???

Среда выполнения ECMAScript выполняет автоматическое преобразование типа по необходимости. Чтобы разъяснить семантику, полезно определить набор операторов преобразования. Эти операторы не являются частью языка. Они определены здесь для удобства определения семантики языка. Операторы сравнения полиморфны, то есть они могут принимать значения любых стандартных типов, кроме типов Reference, List или Completion (т.е. внутренних типов).

Вся полнота проблемы


true
false
1
0
-1
"true"
"false"
"1"
"0"
"-1"
""
null
undefined
[]
{}
[[]]
[0]
[1]
NaN
strict identity
numeric coercion like Number() call
strict identity
primitive coercion via toString() and valueOf() (in this order)
numeric coercion like Number() call
strict identity
primitive coercion via toString() and valueOf() (in this order)
null and undefined differs from everything except for themselves
primitive coercion via toString() and valueOf() (in this order)
object identity
NaN differs from everything including itself
NaN differs from everything including itself

Битовые операции

Функции в JavaScript

01. Объявление

02. Области видимости переменных

03. Замыкания

Объявление функций

```
01. function foo () {
02.   console.log('foo');
03. }

04. var bar = function () {
05.   console.log('bar');
06. }
```

Области видимости переменных

```
01. var foo = 'baz';
02. var bar = function () {
03.   var foo = 'inner';
04.   console.log(foo); // 'inner'
05. }
06.  console.log(foo); // 'baz'
```

Замыкания

```
01. function wrapValue(n) {
02.   var localVariable = n;
03.   return function() { return localVariable; };
04. }
05. var wrap1 = wrapValue(1);
06. var wrap2 = wrapValue(2);
07. console.log(wrap1()); // 1
08. console.log(wrap2()); // 2
```

this

01. Режим конструктора
02. Метод объекта
03. Apply/**Call**
04. Простейший вызов

ООП (Наследование)

01. Прототипная модель наследования

02. Классическая модель наследования

Прекрасные черты JS

01. Функции как объекты первого класса

02. Динамические объекты с прототипным наследованием

03. Литералы объектов и массивов

Ужасные черты JS

01. Глобальные переменные
02. Области видимости
03. Зарезервированные слова
04. **typeof**
05. **parseInt**
06. **+**
07. 0.1 + 0.2
08. falsy

JS плохой?

Не так важно!

Плохие вещи в JS

01. ==
02. with
03. **eval**
04. **continue**
05. **switch**
06. разные **function**
07. void
08. битовые операторы(?)

Что почитать, посмотреть?

Почитать:

1. http://learn.javascript.ru/