



Javascript精度问题

TD (telecom digital)

```
1 | the calculable result of running an experiment.
2 |
3 | class Scientist::Result
4 |   # An Array of candidate Observations.
5 |   attr_reader :candidates
6 |   # The control Observation to which the rest are compared.
7 |   attr_reader :control
8 |   # An Experiment.
9 |   attr_reader :experiment
10 |   # An Array of observations which didn't match the control. See what happens.
11 |   attr_reader :ignored
12 |   # An Array of observations which didn't match the control.
13 |   attr_reader :mismatched
14 |   # An Array of Observations in execution order.
15 |   attr_reader :observations
16 |
17 |   # Internal: Create a new result.
18 |   #
19 |   # experiment - the Experiment this result is for
20 |   # observations - an Array of Observations, in execution order
21 |   # control: - the control Observation
22 |
23 |   def initialize(experiment, observations = [], control = nil)
24 |     @experiment = experiment
25 |     @observations = observations
26 |     @control = control
27 |     @candidates = observations - [control]
28 |     @ignored = []
29 |     @mismatched = []
30 |     freeze
31 |   end
32 |
33 |   # Public: the experiment's context
34 |   def context
35 |     experiment.context
36 |   end
37 |
38 |   # Public: the name of the experiment
39 |   def experiment_name
40 |     experiment.name
41 |   end
42 |
43 |   # Public: was the result a match between all candidates and the control?
44 |   def matched?
45 |     candidates.all? { |c| c == control }
46 |   end
47 |
48 |   lib/scientist/result.rb 1:1
```

数字的二进制

$$173.8125 = 10101101.1101$$

1. 整数部分173, 除2取余, 逆序排列

$$173 / 2 = 86 \dots 1$$

$$86 / 2 = 43 \dots 0$$

$$43 / 2 = 21 \dots 1$$

$$21 / 2 = 10 \dots 1$$

$$10 / 2 = 5 \dots 0$$

$$5 / 2 = 2 \dots 1$$

$$2 / 2 = 1 \dots 0$$

$$1 / 2 = 0 \dots 1$$



10101101

$$2^7 + 2^5 + 2^3 + 2^2 + 1$$

173.8125

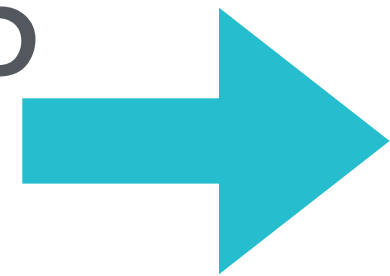
2. 小数部分0.8125, 乘2取整, 顺序排列

$$0.8125 * 2 = 1.625$$

$$0.625 * 2 = 1.25$$

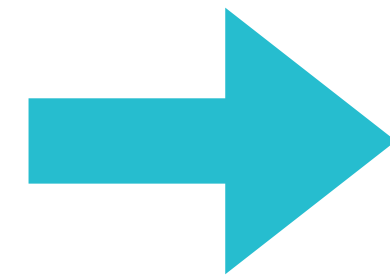
$$0.25 * 2 = 0.5$$

$$0.5 * 2 = 1$$



1101

$$2^{-1} + 2^{-2} + 2^{-4}$$



$$0.8125 = (1 + 0.625) / 2$$

$$= 1/2 + (0.625 * 2) / (2 * 2)$$

$$= 1/2 + (1 + 0.25) / 4$$

$$= 1/2 + 1/4 + (0.25 * 2) / (4 * 2)$$

$$= 1/2 + 1/4 + 0.5 / 8$$

$$= 1/2 + 1/4 + (0.5 * 2) / (8 * 2)$$

$$= 1/2 + 1/4 + 1/16$$

$$= 2^{-1} + 2^{-2} + 2^{-4}$$

使用以上知识，将10进制小数0.1转为二进制：

$$0.1 * 2 = 0.2$$

$$0.2 * 2 = 0.4$$

$$0.4 * 2 = 0.8$$

$$0.8 * 2 = 1.6$$

$$0.6 * 2 = 1.2$$

$$0.2 * 2 = 0.4$$

$$0.4 * 2 = 0.8$$

$$0.8 * 2 = 1.6$$

$$0.6 * 2 = 1.2$$

...



0.0001100110011... 1100无限重复

0.001100110011... \Rightarrow 0.2

为什么这么垃圾？0.1，0.2这么简单的

数字都搞不定，其实0.1 ~ 0.9 之间除了

0.5能搞定之外其他都凉凉~

这么垃圾的方法有别的方法没？

为什么小数部分要用类似于 $2^{-1} + 2^{-2} + 2^{-4}$ 这样的方式去表示？

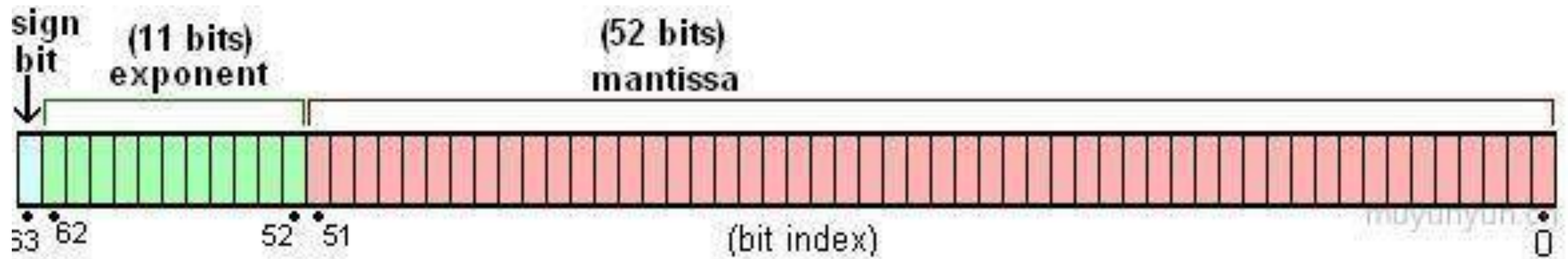
0.8125 取成整数 8125 然后二进制表示，很开心啊~

No, $1/3$ 0.3333333... 就傻了

还有一个原因，因为浮点数使用指数形式去表示的，需要将
10101101.1101 表示成为 $1.0101101101 * 2^7$ 所以必须这样

浮点数存储

javascript中所有数值都以IEEE-754标准的64 bit双精度浮点数进行存储



存储结构的优点：可以归一化处理整数和小数，节省空间

64位比特可分为以下三部分：

- ▶ 符号位S：第一位是正负数符号位(sign) 0是整数 1是负数
- ▶ 指数位E：中间的11位存储指数(exponent)，用来表示次方数
- ▶ 尾数为M：最后52位是尾数(mantissa)，超出部分自动进1舍0

实际由以下公式来计算：

$$V = (-1)^S \times 2^E \times M$$

S 为0或1

M为尾数一般为1.xxxx 所以1会被舍去

E是一个无符号整数长度11位取值范围0 ~ 2047($2^{11} - 1$)但是科学技术法中指数可以为负数，所以减去一个中间数1023，[0, 1022]表示为负，[1024, 2047]表示为正。

$$V = (-1)^S \times 2^{E-1023} \times (M + 1)$$

$0.1 + 0.2 \neq 0.3$

0.1 to 0.0001100110011001100(1100循环) to 1.100110011001100x2⁻⁴

对应成刚才的公式:

$$S = 0$$

$$E - 1023 = -4 \quad E = 1029 = 0111111011$$

M舍去首位的1, 得到10011001100... (52位)

$$0.1 + 0.2 = 0.300000000000000004$$

$$0.00011001100110011001100110011001100110011001100110011001100110011010 +$$

$$0.001100110011001100110011001100110011001100110011001100110011010 =$$

$$0.0100110011001100110011001100110011001100110011001100110011011$$

转成十进制就是 0.300000000000000004

`(2.55).toFixed(1)` 结果是2.5 而不是2.6

实际上0.55的二进制是0.100110011001100...(1100循环)

而从二进制换算过来就是2.54999999999999982

所以根本原因在于2.55的存储要比实际存储小一点，导致0.05的第一位是0，被舍弃掉了

大数危机和最大安全数

由于M(mantissa)的固定长度是52位，再加上省略的一位

1.1111...111 （小数点后共52个1） 即 $2^{53} - 1 = 9007199254740991$

所以最大安全数是`Number.MAX_SAFE_INTEGER = 9007199254740991`

这是JS最多能表示的精度，长度是16。但是长度为16的依然有很大一部分不安全

比最大安全数大的数字会怎样呢？

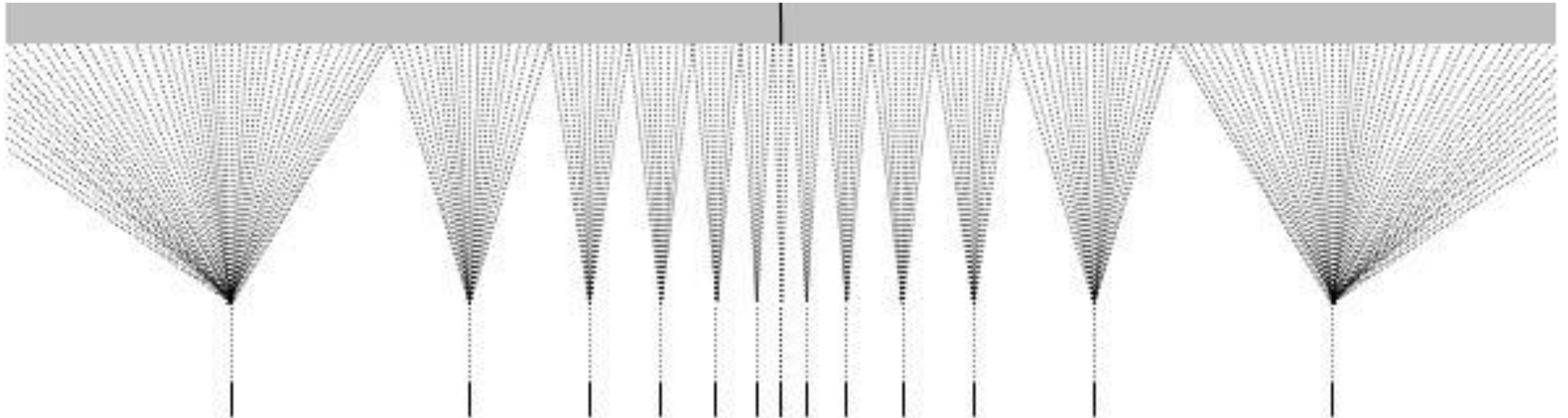
首先比最大安全数大的数字智能在指数E上搞事情。

$2^{52} * M = 2^{52} * 1.xxxx(52\text{位})$ 就是一个能精确表示的整数

$2^{53} * M = 2^{52} * M * 2 = \text{整数} * 2$, 所以 $[2^{53}, 2^{54}]$ 只能一个顶俩

$2^{54} * M = 2^{52} * M * 4 = \text{整数} * 4$, 所以 $[2^{53}, 2^{55}]$ 只能一个顶四

...



那么整数大于9007199254740991会出现什么情况呢？

由于E的最大值是1023(why not 1024)

最大数 $1.111111... * 2^{1023} = (2 - 2^{-52}) * 2^{1023} = 2^{1024} - 2^{971}$

= Number.MAX_VALUE

所以能最大数也是最大整数。并非有些论坛上的 $2^{1024} - 1$

介于 $9007199254740992 < \text{Num} < 1.7976931348623157e+308$ 已经不安全了，因为精度已经无法表示只能从指数E上增加

当指数位全部是1的时候(特殊值)，IEEE规定这个浮点数可用来表示3个特殊值，分别是正无穷，负无穷，NaN。具体的，小数位不为0的时候表示NaN；小数位为0时，当符号位s=0时表示正无穷，s=1时候表示负无穷

Thank You

```
1 | the result of running an experiment
2 class Scientist::Result
3   # An Array of candidate Observations.
4   attr_reader :candidates
5   # The control Observation to which the rest are compared.
6   attr_reader :control
7   # An experiment.
8   attr_reader :experiment
9   # An Array of observations which didn't match the control. See note above.
10  attr_reader :ignored
11  # An Array of observations which didn't match the control.
12  attr_reader :mismatched
13  # An Array of Observations in execution order.
14  attr_reader :observations
15  # Internal: Create a new result.
16  #
17  # experiment - the Experiment this result is for
18  # observations - an Array of Observations, in execution order
19  # control - the control Observation
20  #
21  def initialize(experiment, observations = [], control = nil)
22    @experiment = experiment
23    @observations = observations
24    @control = control
25    @candidates = observations - [control]
26    evaluate_candidates
27  end
28  freeze
29  end
30
31  # Public: the experiment's context
32  def context
33    experiment.context
34  end
35
36  # Public: the name of the experiment
37  def experiment_name
38    experiment.name
39  end
40
41  # Public: was the result a match between all tests?
42  def matched?
43  end
44
45  lib/scientist/result.rb 1:1
```