JavaScript深入浅出

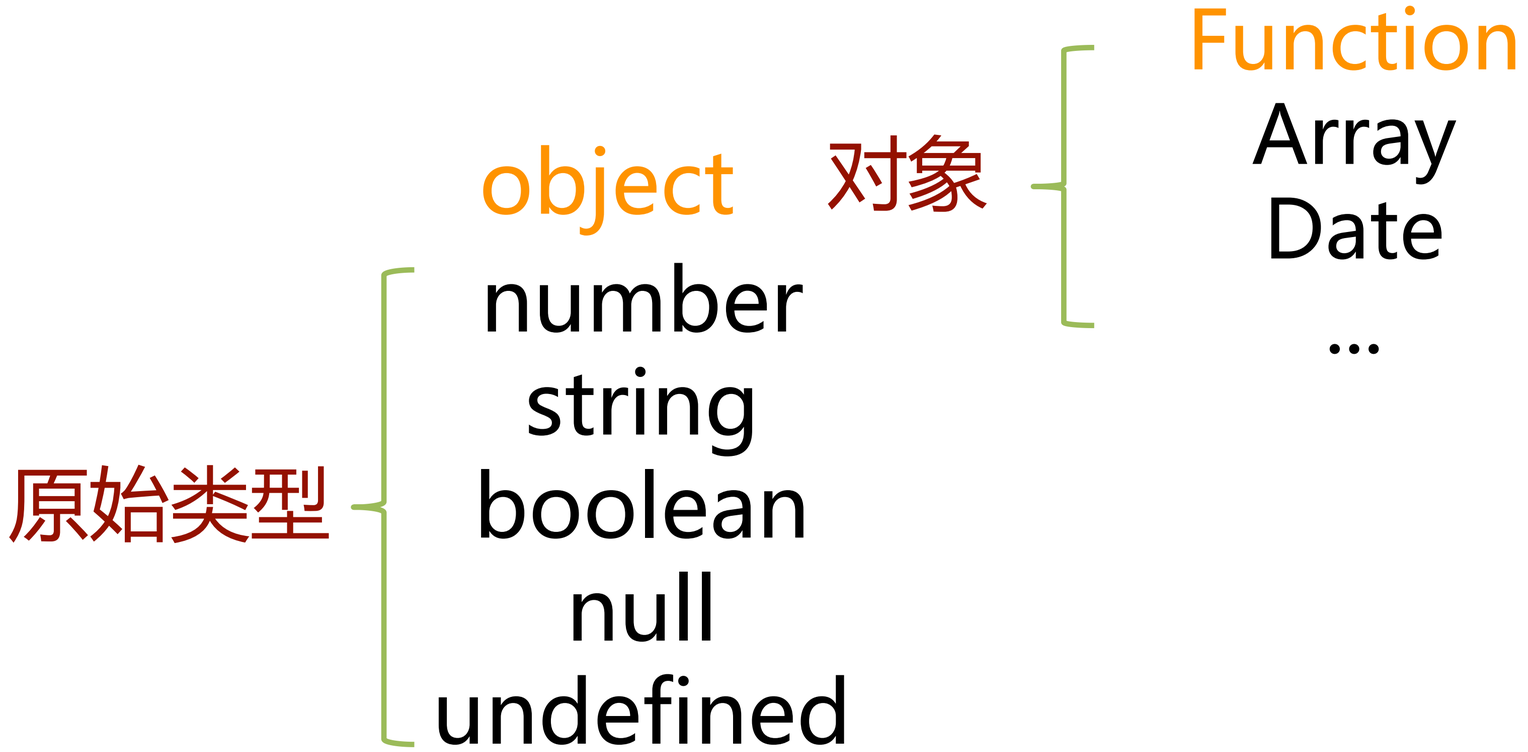
第1章 数据类型 by @Bosn

1 六种数据类型

弱类型特性

```
var num = 32;

num =  “this is a string” ;

    32 + 32          //   64

“32”  +  32          //  “3232”

“32”  -  32          //    0
```

数据类型


object
对象
Function
Array
Date
...
原始类型
number
string
boolean
null
undefined

2 隐式转换

+ 和 -

```
var x =    'The answer is    ' + 42;
var y = 42 +    ' is the answer'   ;
```

```
num - 0
num +   ''
```

```
"37"   - 7   // 30
"37"   + 7  // 377
```

巧用+/-规则转换类型

a == b

"1.23" == 1.23

0 == false

null == undefined

new Object() == new Object()

[1, 2] == [1, 2]

a === b

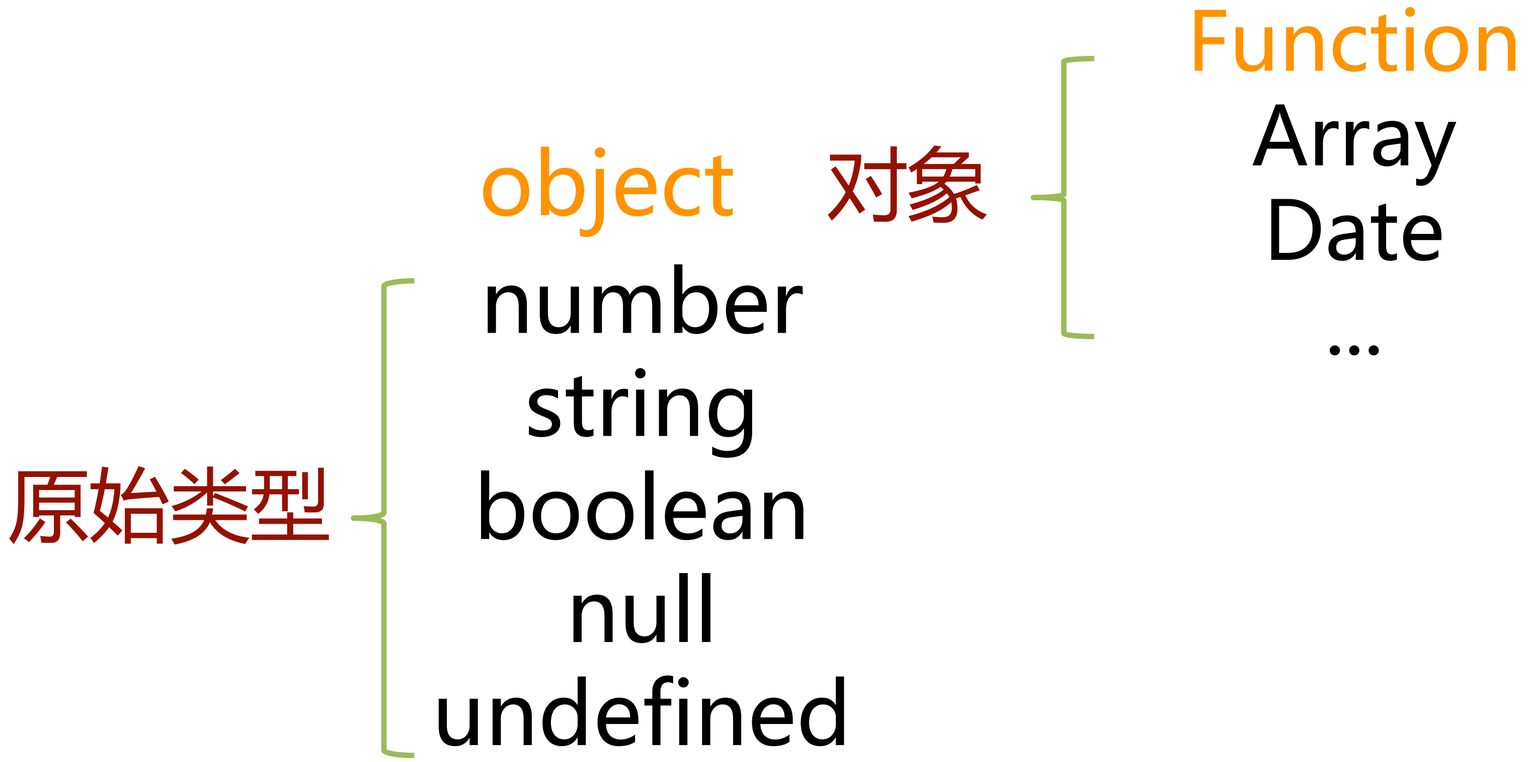
- 类型不同，返回false
- 类型相同
  - NaN ≠ NaN
  - new Object ≠ new Object
  - null === null
  - undefined === undefined

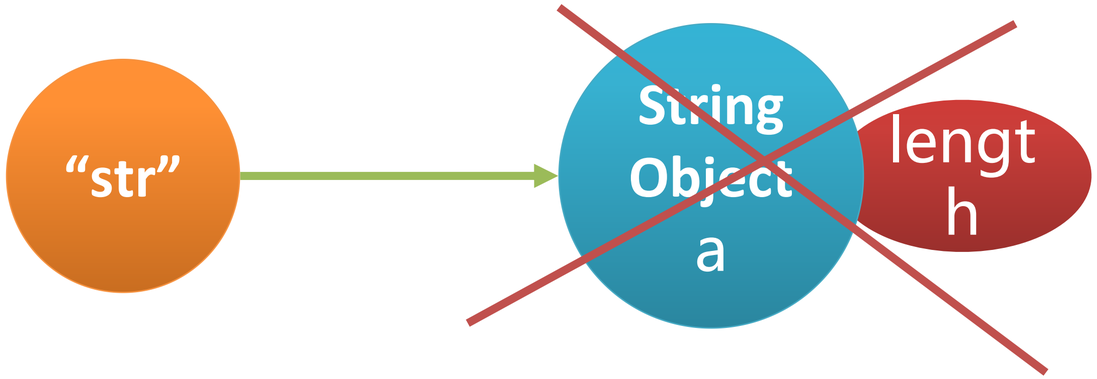
a == b

- 类型相同，同===
- 类型不同，尝试类型转换和比较:
  - null == undefined 相等
  - number == string 转number 1 == “1.0" // true
  - boolean == ？ 转number 1 == true // true
  - object == number | string 尝试对象转为基本类型 new String('hi') == ‘hi’ // true
  - 其它：false

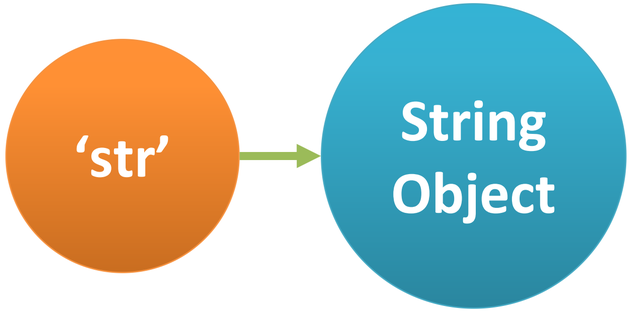
3 包装对象

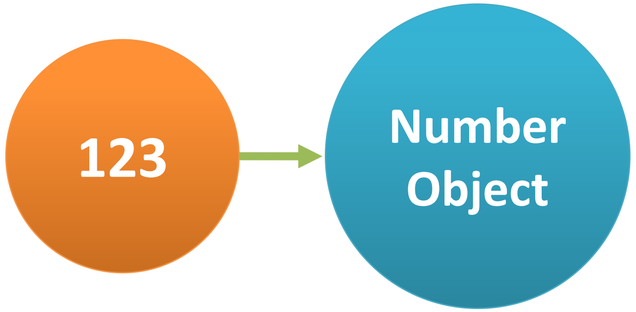
数据类型

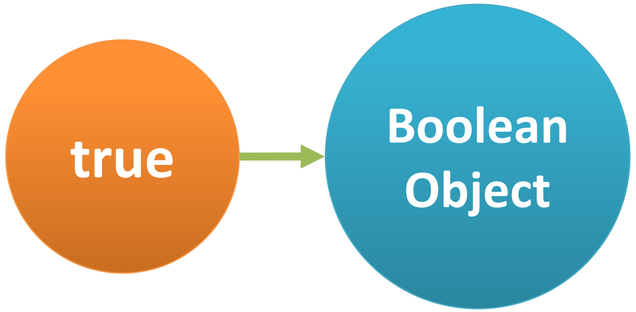

object
对象
Function
Array
Date
...
原始类型
number
string
boolean
null
undefined




```
var a = "string";
alert(a.length);
a.t = 3;
alert(a.t);
```







4 类型检测

类型检测

typeof

instanceof

Object.prototype.toString

constructor

duck type

```
typeof 100          "number"
typeof true         "boolean"
typeof function     "function"

typeof(undefined)   "undefined"
typeof new Object() "object"
typeof  [1 ,  2]    "object"
typeof NaN          "number"
typeof null         "object"
```

typeof null === “object”

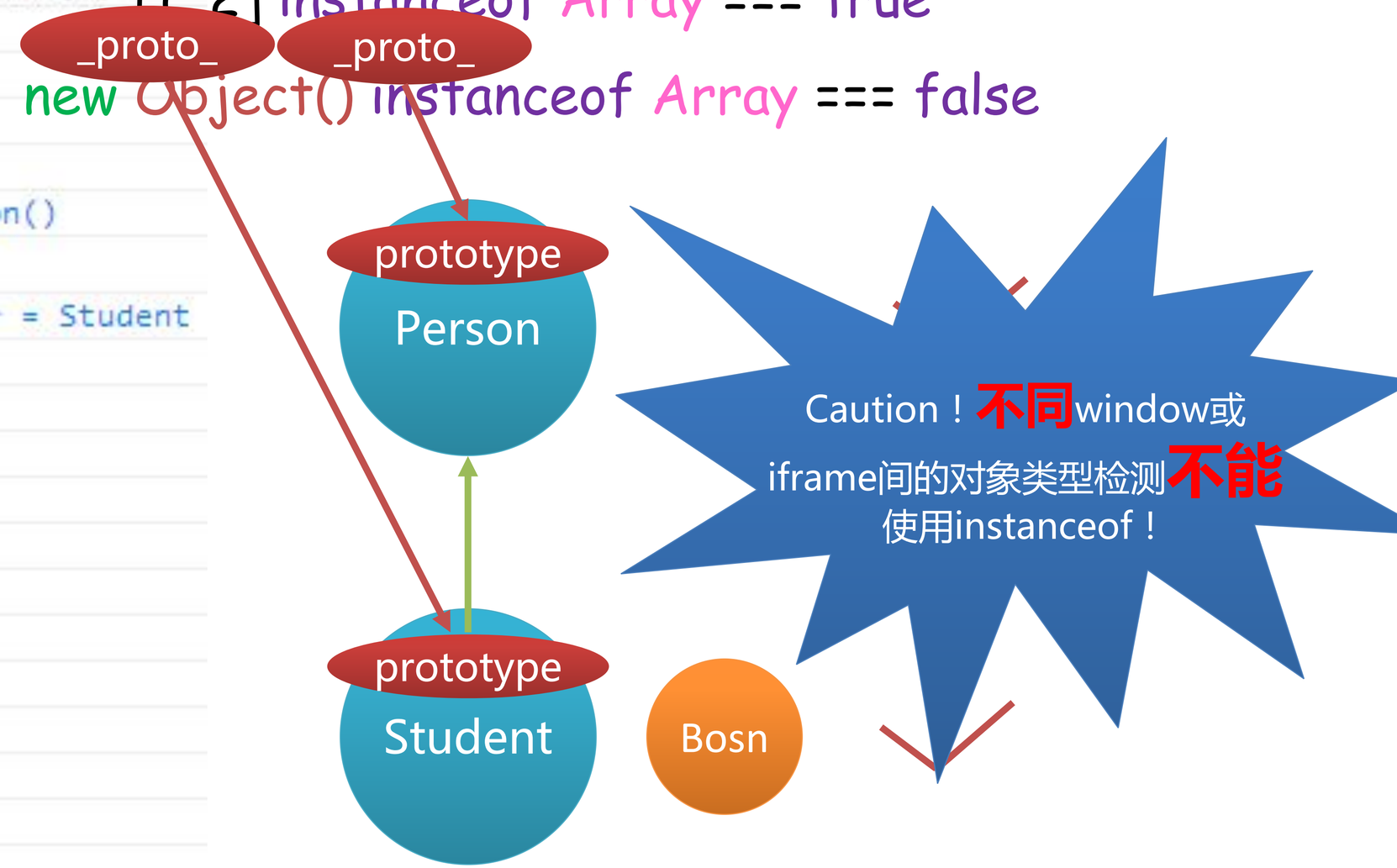
obj instanceof Object

instanceof

```
> function Person(){}
< undefined
> function Student(){}
< undefined
> Student.prototype = new Person()
< ▶ Person
> Student.prototype.constructor = Student
< function Student(){}
> var bosn = new Student()
< undefined
> bosn instanceof Student
< true
> var one = new Person()
< undefined
> one instanceof Person
< true
> one instanceof Student
< false
> bosn instanceof Person
< true
```

[1, 2] instanceof Array === true

new Object() instanceof Array === false



Object.prototype.toString

```
Object.prototype.toString.apply([]); === "[object Array]";
Object.prototype.toString.apply(function(){}); === "[object Function]";
Object.prototype.toString.apply(null); === "[object Null]"
Object.prototype.toString.apply(undefined); === "[object Undefined]"
```

IE6/7/8 Object.prototype.toString.apply(null) 返回" [object Object]"

类型检测

typeof

instanceof

Object.prototype.toString

constructor

duck type

类型检测小结

typeof

适合基本类型及function检测，遇到null失效。

[[Class]]

通过{}.toString拿到，适合内置对象和基元类型，遇到null和undefined失效(IE678等返回[object Object])。

instanceof

适合自定义对象，也可以用来检测原生对象，在不同iframe和window间检测时失效。

5 实践

Practise 1

```
1 == '1'
1 === '1'
1 + '2' === '1' + 2
1 + '2' === '1' + new Number(2)
1 + true === false + 2
1 + null == undefined + 1
'a' - 'b' == 'b' - 'a'
```

```
> 1 == '1'
true
> 1 === '1'
false
> 1 + '2' === '1' + 2
true
> 1 + '2' === '1' + new Number(2)
true
```

```
> 1 + true === false + 2
true
> 1 + null == undefined + 1
false
> 'a' - 'b' == 'b' - 'a'
false
```

Practise 2

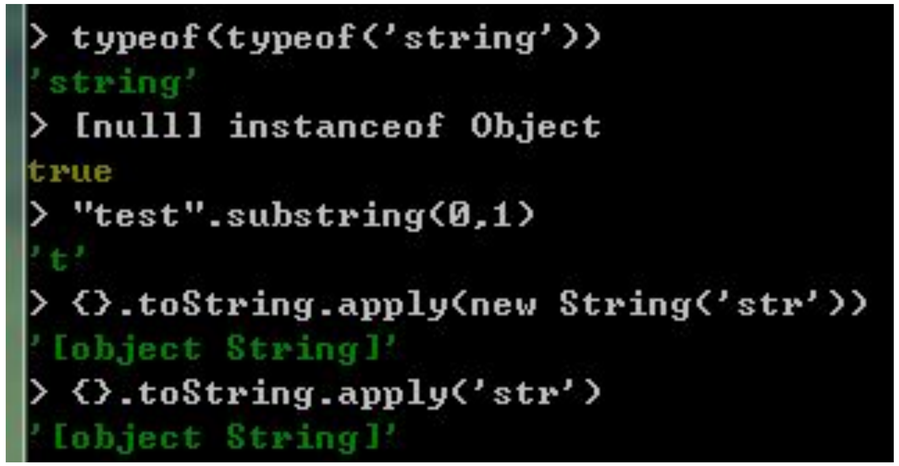
typeof(typeof(‘string’))

[null] instanceof Object

“test” .substring(0 ,1)

{}.toString.apply(new String(‘str’));

{}.toString.apply(‘str’);



```
5 - "4"

5 + "4"

+!{}[true]

+[1]

+[1, 2]

7 - "a"

7 / 0
```

...

5 + "4"

5 + null

4 == "4.00"

4 === "4.00"

null == undefined

0 == false

0 == null

null == false

谢谢