目录

[String 3](#_Toc2955415)

[存储问题 3](#_Toc2955416)

[substr/slice/subtring 3](#_Toc2955417)

[substr 3](#_Toc2955418)

[slice 3](#_Toc2955419)

[substring 3](#_Toc2955420)

[String和toString 3](#_Toc2955421)

[object 4](#_Toc2955422)

[构造方法 4](#_Toc2955423)

[new Object() 4](#_Toc2955424)

[assign 4](#_Toc2955425)

[create 4](#_Toc2955426)

[is 5](#_Toc2955427)

[defineProperty/defineProperties/getOwnPropertyDescriptor 5](#_Toc2955428)

[getPrototypeOf/setPrototypeOf 6](#_Toc2955429)

[getOwnPropertyNames/getOwnPropertySymbols/keys/entries/values/for..in 7](#_Toc2955430)

[freeze/preventExtensions/seal/isExtensible/isFrozen/isSealed 7](#_Toc2955431)

[其他 8](#_Toc2955432)

[set/get 8](#_Toc2955433)

[冷门 9](#_Toc2955434)

[in 9](#_Toc2955435)

[a[a] 9](#_Toc2955436)

[Object.length/object.length 10](#_Toc2955437)

[属性操作 10](#_Toc2955438)

[原型方法 10](#_Toc2955439)

[hasOwnProperty/propertyIsEnumerable 10](#_Toc2955440)

[isPrototypeOf 11](#_Toc2955441)

[toLocaleString/toString/valueOf 11](#_Toc2955442)

[function 11](#_Toc2955443)

[new Function(args1, args2, args3) 11](#_Toc2955444)

[Foo.x 11](#_Toc2955445)

[冷门属性 11](#_Toc2955446)

[name 11](#_Toc2955447)

[prototype 12](#_Toc2955448)

[内置构造函数 12](#_Toc2955449)

[匿名函数 12](#_Toc2955450)

[arguments 13](#_Toc2955451)

[callee 13](#_Toc2955452)

[caller 13](#_Toc2955453)

[Array 14](#_Toc2955454)

[isArray 14](#_Toc2955455)

[遍历函数 14](#_Toc2955456)

[特殊 14](#_Toc2955457)

[Array.prototype 14](#_Toc2955458)

[[] 14](#_Toc2955459)

[JSON 14](#_Toc2955460)

[parse 14](#_Toc2955461)

[Number 15](#_Toc2955462)

[数字范围 15](#_Toc2955463)

[精度问题 15](#_Toc2955464)

[NaN 15](#_Toc2955465)

[Math 16](#_Toc2955466)

[random 16](#_Toc2955467)

# String

## 存储问题

和是不是js语言无关。

UTF-8编码：

一个英文字符等于一个字节(B)，一个中文（含繁体）等于三个字节。

Unicode编码：

一个英文等于两个字节，一个中文（含繁体）等于两个字节。

## substr/slice/subtring

### substr

第一个参数为起始位置，第二个参数为截取的长度

第一个参数为负数则从末尾往前找，第二个参数为负数则返回空字符串

此方法不会修改原字符串，只是查找并返回查找的结果

### slice

第二个参数为结束位置，返回的结果不包括结束为止。

### substring

类似slice, 不能为负数,否则返回空字符串。

## String和toString

toString不能转换null,undefined（用的话会报错）。可以写一个数字，代表进制，对应进制字符串。String可以转null,undefined(变成‘null’等)，不能写进制。

# object

## 构造方法

### new Object()

如果给定值是 null 或 undefined，将会创建并返回一个空对象，否则，将返回一个与给定值对应类型的对象。

new Obejct(1,2) => Number 1

new Obejct([1,2]) => Array [1,2]

new Object(1:2) => 报错

### assign

使用源对象的[[Get]]和目标对象的[[Set]]，

会跳过那些值为 null 或 undefined 的源对象。

继承属性和不可枚举属性是不能拷贝的

原始类型会被包装为对象,但是只有字符串的包装对象才可能有自身可枚举属性。

“abc” => { "0": "a", "1": "b", "2": "c" }

### create

Object.create(proto[, propertiesObject])

### is

#### Object.is(value1, value2);

Object.is的行为方式与三等号相同，但是对于NaN和-0和+0进行特殊处理，所以最后两个不相同，而Object.is（NaN，NaN）将为 true。

Object.is（NaN，NaN）为 true

Object.is(+0,-0)为false

### defineProperty/defineProperties/getOwnPropertyDescriptor

修改描述符属性： 用defineProperties覆盖设置。

#### Object.defineProperty(obj, prop, descriptor)

创建在属性枚举期间显示的属性

descriptor: configurable/enumerable/value/writable/get/set: 默认都是false或者undefined

(configurable: 对象的属性是否可以被删除，以及除writable特性外的其他特性是否可以被修改。如果是false,删除无效，重新设置报错)

value和get不能混合使用。(会报错，就是数据描述符和存取描述符不能混用，但他们都有configurable和enumerable)

o.a = 1;

// 等同于 :

Object.defineProperty(o, "a", {

value : 1,

writable : true,

configurable : true,

enumerable : true

});

#### defineproperties

Object.defineProperties(obj, {

'property1': {

value: true,

writable: true

},

// etc. etc.

});

#### length

length 属性初始为 non-configurable，non-enumerable 以及 writable。

#### getOwnPropertyDescriptor

只能获取，修改没有效果，比如d.enumerable = true;

d = Object.getOwnPropertyDescriptor(o, "foo");

// d {

// configurable: true,

// enumerable: true,

// get: /\*the getter function\*/,

// set: undefined

// }

### getPrototypeOf/setPrototypeOf

返回其原型的对象。如果没有继承属性，则返回 null 。

在 ES5 中，如果参数不是一个对象类型，将抛出一个TypeError异常。在 ES2015 中，参数会被强制转换为一个 Object。

### getOwnPropertyNames/getOwnPropertySymbols/keys/entries/values/for..in

#### getOwnpropertyNames/keys/entries/values/

不包括原型链中的。

#### for in

会枚举其原型链上的属性

不会枚举constructor和\_\_proto\_\_，这两个相当于隐藏对象

#### getOwnPropertySymbols

console.log(obj); // { a : 1, [Symbol("foo")]: 2 }

Object.getOwnPropertySymbols(obj); // [Symbol(foo)]

### freeze/preventExtensions/seal/isExtensible/isFrozen/isSealed

#### freeze

非对象参数将被视为要被冻结的普通对象，并被简单地返回。

> Object.freeze(1)

TypeError: 1 is not an object // ES5 code

> Object.freeze(1)

1 // ES2015 code

要使对象不可变，需要递归冻结每个类型为对象的属性

#### seal

密封，可写性描述（writable）为可写（true）的属性的值仍然可以被修改。

#### preventExtensions

不可扩展对象的属性可能仍然可被删除修改,只是不能增加属性

#### Object.isExtensible(obj)

Object.preventExtensions，Object.seal 或 Object.freeze 方法都可以标记一个对象为不可扩展

在 ES5 中，如果参数不是一个对象类型，将抛出一个 TypeError 异常。在 ES6 中， non-object 参数将被视为一个不可扩展的普通对象，因此会返回 false 。

## 其他

### set/get

var test = {

age: null,

    get age (){

        return age;

    },

    set age (value){

        if(value > 100) age= new Date().getFullYear() - value;

        else age = value;

    }

};

或者

function Person() {

    var age = new Date().getFullYear() - 18;

    Object.defineProperty(this, "age", {

        get: function () { return new Date().getFullYear() - age; },

        set: function (value) { return value; }

    });

}

## 冷门

### in

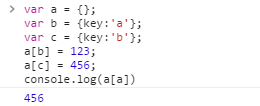
'a' in a

返回一个属性是否直接在对象中（不包括原型）。返回boolean。

不能嵌套！只判断最外层。

### a[a]

一个对象作为一个对象的属性的时候，这个属性就成了 [object Object]



### Object.length/object.length

Object.length = 1

({a:'a'}).length = undefined

### 属性操作

delete obj.name

不能删除原型链中的属性。

或者obj[‘name’] = undefined;

## 原型方法

### hasOwnProperty/propertyIsEnumerable

temp.hasOwnProperty(tmp)

判断一个对象是否有这个属性。直接用[]判断也可以。

不包括原型链上的属性



#### obj.propertyIsEnumerable(prop)

### isPrototypeOf

prototypeObj.isPrototypeOf(object)

测试一个对象是否存在于另一个对象的原型链上。

### toLocaleString/toString/valueOf

# function

## new Function(args1, args2, args3)

## Foo.x

定义后能改变函数中this对应的值

区分function Foo() {

var x = 5;

console.log(this.x)

}

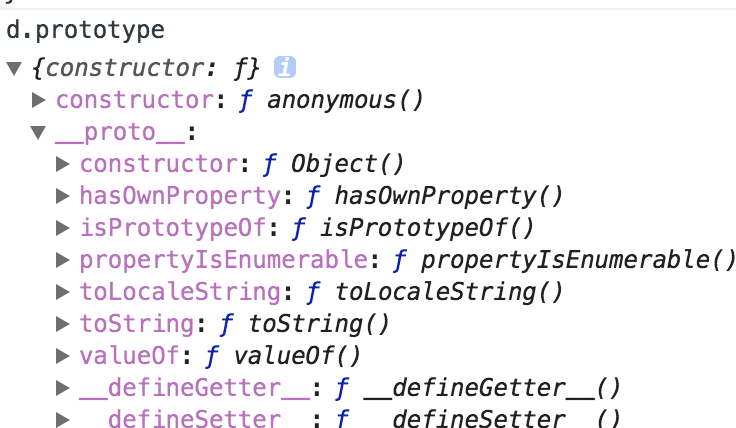
## 冷门属性

### name

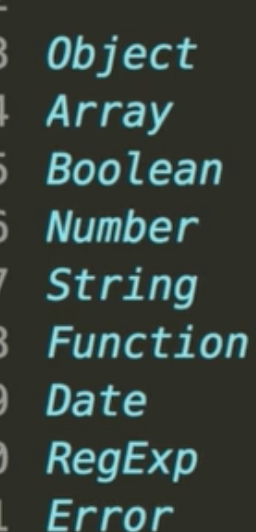
返回函数名。Foo.name;

## prototype

自带两个属性



## 内置构造函数



## 匿名函数

(function {// code})();可以被执行, 而function {// code}();却会报错

(1). 首先, 要清楚两者的区别:     (function {// code})是表达式, function {// code}是函数声明.       
       (2). 其次, js"预编译"的特点:     **js在"预编译"阶段, 会解释函数声明, 但却会忽略表式**.       
       (3). 当js执行到function() {//code}();时, 由于function() {//code}在"预编译"阶段已经被解释过, js会跳过function(){//code}, 试图去执行();, 故会报错;       
    当js执行到(function {// code})();时, 由于(function {// code})是表达式, js会去对它求解得到返回值, 由于返回值是一 个函数, 故而遇到();时, 便会被执行.

 函数转换为表达式的方法并不一定要靠分组操作符()，我们还可以用void操作符，~操作符，!操作符……

## arguments

arguments这个对象不能显式创建，arguments对象只有函数开始时才可用。就是说用在函数创建时。

arguments对象并不是一个数组，但是访问单个参数的方式与访问数组元素的方式相同

function test() {  
        var s = "";  
        for (var i = 0; i < arguments.length; i++) {  
            alert(arguments[i]);  
            s += arguments[i] + ",";  
        }  
        return s;  
    }  
    test("name", "age")

### callee

callee 属性是 arguments 对象的一个成员，它表示对函数对象本身的引用，这有利于匿名函数的递归或者保证函数的封装性，

## caller

返回调用这个函数的函数。

该属性的常用形式arguments.callee.caller替代了被废弃的 arguments.caller.

# Array

## isArray

空数组也为true。

## 遍历函数

在forEach等遍历函数里不支持传入异步函数，会当成同步函数处理。用for循环没这个问题。

## 特殊

### Array.prototype

Array.prototype 是一个数组

### []

[].length = 0

# JSON

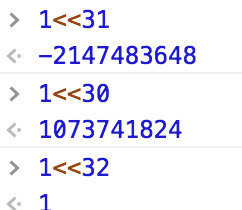
## parse

不能是单引号

键必须用双引号包裹

# Number

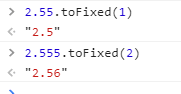
## 数字范围

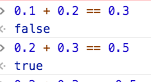


js的数字是int，也就是说占4个字节（B），每个字节8位（b），所以2^32,第一位表示符号，所以是-2^31 到 2^31-1(1<<30都是1000。。，不是最大的1111。。。)

为什么10000。。是最小的负数，而不是表示0？ 因为计算机用的补码表示数字。

## 精度问题





## NaN

Number.NaN => 一个特殊的值，与所有值都不相等，包括它自己。在Number对象里面。无法使用 for/in 循环来枚举 NaN 属性，也不能用 delete 运算符来删除它。

判断： isNaN() 全局函数。

# Math

## random

返回的数小数点后一般有16-18位。