正课:

1. 自定义继承:

2. 静态方法:

3. ES5:

1. 自定义继承: 见PDF: 1\_自定义继承.pdf

2. 静态方法:

什么是: 不需要创建子对象，就可用构造函数直接调用的方法

问题: 原型对象中的方法，只能被当前类型的子对象使用

何时: 2种:

1. 如果一个函数，不确定将来使用它的对象的类型时

2. 如果一个函数在调用时，不需要提前创建子对象

如何: 构造函数.静态方法=function(){ ... }

调用时: 构造函数.静态方法()

强调: 调用静态方法之前不需要创建子对象

3. ES5:

1. 严格模式:

什么是: 比普通js运行机制要求更严格的模式

为什么: js语言有广受诟病的很多缺陷

何时: 今后所有js，都要运行在严格模式下.

如何: 在当前代码段的顶部: "use strict";

严格模式要求:

1. 禁止给未声明的变量赋值:

2. 静默失败升级为错误:

静默失败: 执行不成功，也不报错

3. 禁用了arguments.callee:

arguments.callee专门在调用函数时，获得当前函数本身

何时: 专门用于递归调用中获得当前函数

为什么禁用: 不再建议使用递归

因为: 递归重复计算量太大！效率极低

解决: 几乎所有的递归都可用循环代替

4. 普通函数调用中的this不再指向window

而是undefined

2. 保护对象:

问题: JS中对象毫无自保能力

解决: ES5提供了一系列措施保护对象

如何:

1. 保护对象的单个属性

ES5将对象的属性，分为2大类:

1. 命名属性: 可用.直接访问的属性

又分为: 2类:

1. 数据属性: 实际存储属性值的属性

2. 访问器属性: 不实际存储属性值，专门提供对另一个数据属性的自定义保护！

2. 内部属性: 实际存在，但不能用.访问到的属性

保护数据属性:

ES5将一个数据属性，又定义为一个缩微的小对象

属性: {

value: 实际存储属性值,

writable: true/false, //控制是否可修改

enumerable: true/false, //控制是否可被for in遍历

//只能防住for in，但是用.依然可访问

configurable: true/false //控制是否可删除该属性

//还控制是否可修改另外两个特性

//一旦改为false，不可逆！

//所以，修改其它两个开关时都会同时修改configurable:false,作为双保险。

}

获取一个属性的四大特性: 了解

var miniObj=

Object.getOwnPropertyDescriptor(obj, "属性名");

修改一个属性的四大特性:

Object.defineProperty(obj,"属性名",{

开关: true/false,

... : ...

})

同时修改多个属性的四大特性:

Object.defineProperties(obj,{

属性名:{

开关:true/false,

... : ...

},

... : {

}

})

问题: 开关不能用自定义规则保护属性

解决: 用访问器属性:

访问器属性:

何时: 只要用自定义规则保护对象属性时

如何:

前提: 先有一个隐藏的数据属性实际存储数据

再添加一个访问器属性来保护隐藏的数据属性:

Object.defineProperty(obj,"访问器属性名",{

get:function(){ return this.受保护的数据属性 },

set:function(value){

//value可接住将来要赋的新值

//如果value符合条件

//才this.受保护的数据属性=value

//否则

//报错

},

enumerable:true, //让访问器属性代替数据属性抛头露面

configurable:false

})

访问器属性何时起作用:

访问器属性的用法和普通属性是一样的

只不过:

1. 当试图获取访问器属性的值时，自动调用get()

2. 当试图修改访问器属性的值时，自动调用set()，并传入新值作为参数value

2. 保护对象的结构