正课:

1. ES5

2. ES6

1. ES5

保护对象:

什么是: 控制对对象结构和属性值的修改

为什么: js的对象毫无自保能力

何时: 如果限制对一个对象的操作时

如何: 2个层面:

1. 保护单个属性:

ES5规定: 属性分2大类：

1. 内部属性: 保存在对象内部，但不能用.直接访问的属性。比如: class ——不需要保护，因为根本改不了

2. 命名属性: 凡是用.可直接访问到的属性

如何保护命名属性: 又分为两大类:

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，仅提供对另一个数据属性得保护

数据属性: 如何保护:

每个数据属性中，都包含有四大特性:

sname:"Li Lei"

↓ES5

sname{

value: "Li Lei", //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-隐藏

强调: 仅控制for in遍历，不能控制.直接访问

configurable:true/false //控制是否可删除该属性

//控制是否可修改前两个特性

//一旦改为false，不可逆！

}

获得一个属性的四大特性(描述信息):

var attrs=Object.getOwnPropertyDescriptor(obj,"属性名")

修改一个属性的四大特性:

Object.defineProperty(obj,"属性名",{

特性: true/false,

... : ...

})

问题: 一次只能修改一个属性的四大特性

解决: 批量修改多个属性的四大特性:

Object.defineProperties(obj,{

属性名:{

特性名:true/false

... : ...

},

属性名:{

特性名:true/false

... : ...

},

})

问题: 无法使用自定义规则保护属性

访问器属性:

何时: 只要用自定义规则保护属性时

如何:

前提: 先有一个隐藏的数据属性实际存储属性值

定义访问器属性: 不能用直接量，只能用API

Object.defineProperty(obj,"访问器属性名",{

//四大特性:

get:function(){ //每次试图取值时自动调用

return this.隐藏数据属性

},

set:function(val){ //每次试图赋值时自动调用

//val 自动获得要赋的新值

if(val符合要求)

this.隐藏数据属性=val

else

报错！

},

enumerable:true,

configurable:false

})

使用访问器属性: 用法同普通数据属性完全一样！

obj.访问器属性=新值

2. 保护对象结构: 3个级别:

1. 防扩展: 禁止给对象添加新属性

Object.preventExtensions(obj)

原理: 每个对象内部都隐藏着一个内部属性

extensible:true

preventExtensions(obj)将obj的extensible:false

2. 密封: 在兼具防扩展的同时，又禁止删除现有属性

Object.seal(obj)

原理: 2件事: 将obj的extensible:false

将obj的每个属性的configurable:false

3. 冻结: 在兼具密封的同时，又禁止修改所有属性值

何时: 一些公用的配置信息对象

Object.freeze(obj)

原理: 3件事: 将obj的extensible:false

将obj的每个属性的configurable:false

将obj的每个属性的writable:false

Object.create()

什么是: 基于一个现有父对象，直接创建子对象，并为子对象扩展自有属性

3件事: 1. 创建一个新对象

2. 让新对象继承父对象

3. 为新对象扩展自有属性

何时: 如果没有构造函数，也想创建子对象时

如何: var child=Object.create(father)

替换this: .call() .apply() .bind()

何时: 如果一个函数执行时，其中的this不是想要的

如何: 3种方法:

1. 在调用函数时，临时替换this: .call()

函数.call(替换this的对象, 实参1, ... ... )

问题: 临时绑定，只调用一次

2. 基于原函数，为某个对象，创建一个专属的新函数，并永久绑定this为指定对象: .bind()

var 新函数=函数.bind(替换this的对象)

.bind()2件事:

1. 创建一个和原函数一模一样的新函数

2. 永久替换新函数中的this为指定对象

固定场景: 专门用于绑定回调函数中的this

3. 打散数组参数: .apply

何时: 如果函数需要传入多个参数值，但参数是放在数组中给定的。

如何: 函数.apply(替换this的对象,数组)

apply会打散数组为单个值↓

//(替换this的对象,元素1,元素2,...)

数组API:

1. 查找元素在数组中的位置:

var i=arr.indexOf(elem[, fromi])

返回值: elem的位置i, 如果找不到，返回-1

2. 判断: 判断数组的元素是否符合要求

.every(): 判断数组中每个元素是否都符合要求

.some(): 判断数组中是否包含符合要求的元素

都返回bool值

var bool=arr.every(function(val,i,arr){

//function会在每个元素上自动执行一次

//在每个元素上执行时:

//val: 自动获得当前元素值

//i: 自动获得当前位置

//arr: 自动获得当前正在遍历的数组对象

//不能用this！只能用arr->当前数组

return 判断条件

})

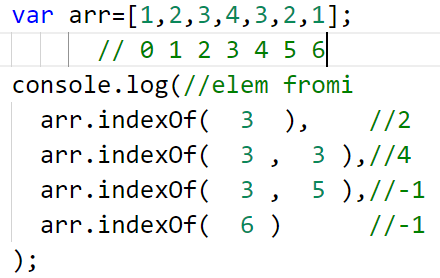
课后任务:

(1)复习: 如何保护对象属性和结构

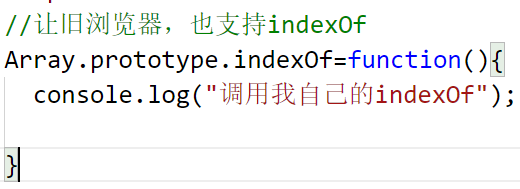
(2)作业: 完成课后练习:  
 题目要求:

1. 让旧浏览器也支持新的数组API

运行效果:



提示:



(3)项目:

题目要求:

运行效果: