正课:

1. ES5

2. ES6

1. ES5

保护对象:

保护属性:

属性分为2大类:

命名属性: 可用.直接访问到的属性

——服务器端语言 公有public

保护命名属性: 分为2大类:

数据属性: 实际保存属性值的属性

如何保护: 每个数据属性都有四大特性:

{

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遍历到

问题: 隐藏并不彻底, 依然可用.直接访问！

configurable: true/false,

控制着: 1. 是否可删除该属性

2. 能否修改其它特性

强调: 一旦改为false，不可逆！

}

如何获取四大特性:

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

如何修改四大特性:

Object.defineProperty(obj,"属性名",{

特性名: 值,

... : ...

})

问题: 每次只能修改一个属性的四大特性

解决: Object.defineProperties(obj,{

属性名:{

特性名:值,

... : ...

},

属性名:{

... : ...

}

})

问题: 保护是固定的，有限的几种

访问器属性: 不实际存储属性值，仅对其它数据属性提供保护！

何时: 1. 用自定义规则保护属性时

2. 计算属性(vue)/虚拟属性(mongodb)

包含: 四大特性: {

get(){ return this.受保护的数据属性},

set(val){

//val自动收到要赋的新值

if(val符合规则)

才this.受保护的数据属性=val

else

报错！

},

enumerable:true/false,

configurable:true/false

}

其中: 访问器属性的用法和普通属性完全一样

但是: 试图读取属性值时，自动调用get方法

试图为属性赋值时，自动调用set方法

set方法中的val会自动接到要赋的新值

如何定义访问器属性: 访问器属性只能通过Object.defineProperty或defineProperties创建

内部属性: 不可用.访问的属性

——服务器端语言 私有 private

比如: .class .\_\_proto\_\_

防篡改: 禁止删除/添加属性

3个级别：

1. 防扩展: 禁止给对象添加新属性

Object.preventExtensions(obj)

原理: 每个对象中都有一个内部属性extensible=true

控制着是否可为对象添加新属性

preventExtensions()其实是将extensible=false

2. 密封: 在兼具防扩展基础上，又禁止删除现有属性

Object.seal(obj)

原理: 不但将extensible=false

且自动修改所有属性的configurable为false

3. 冻结: 在兼具密封基础上，又禁止修改一切属性值

Object.freeze(obj)

原理: 在密封的基础上，进一步修改所有属性的writable为false!

Object.create:

什么是: 创建一个子对象，继承指定父对象，并为子对象扩展自有属性。

何时: 只有父对象，也想创建子对象时

如何: var child=Object.create(father,{

//Object.defineProperties

bao:{

value:"LV",

writable:true,

enumerable:true,

configurable:false

}

})

Object.assign() 鄙视: 复制或合并对象

复制: var clone=Object.assign({},obj)

将obj中的所以属性和属性值合并到空对象中,并返回

问题: 浅克隆: 仅复制数据属性的值

如果属性是引用类型对象，则仅复制对象地址

解决: 深克隆: var clone=JSON.parse(JSON.stringify(obj))

合并: 将多个原对象，合并为一个新对象，原对象保持不变

var newObj=Object.assign({},obj1,obj2,...)

何时: 如果页面需要的数据，来自于多个接口方法返回的多个对象。都要将多个对象，先合并为一个对象，用一次响应返回！

call apply bind:

如果函数中的this不是想要的，都可用call,apply,bind替换函数中的this为想要的任何对象

call apply:

什么是: 在调用函数时，临时替换函数中的this

何时: 如果调用函数时，临时替换this

如何:

fun.call(obj,参数值列表)

调用fun，临时替换fun中的this为obj，并传递参数给fun

fun.apply(obj,[参数值数组])

调用fun，临时替换fun中的this为obj，并传递参数给fun

强调: 唯一差别: 参数的形式上:

call,要求参数单独传入

apply,要求所有参数值必须放在一个数组中整体传入

bind:

什么是: 基于原函数，创建一个新函数，并永久绑定其中的this为指定对象

何时: 专门用于替换回调函数中的不想要的this

数组API:

判断: 判断数组中的元素是否符合条件

.every(): 判断是否数组中每个元素都符合要求

.some(): 判断数组中是否包含符合要求的元素

仅以.every()举例:

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

//elem当前元素值, i 当前位置, arr 当前数组对象

return 判断条件

})

用判断条件去检查每个元素，除非所有元素都满足要求，才返回true。

遍历API: 对数组中每个元素执行相同的操作

.forEach() 对原数组中每个元素执行相同操作,直接修改原数组

何时: 对原数组中每个元素执行相同操作时

如何: arr.forEach(function(elem,i,arr){

arr[i]=新值

})

.map() 取出原数组中每个元素，执行相同操作后，放入新数组中返回，原数组保持不变

何时: 希望原数组保持不变，返回新数组时

如何: arr.map(function(elem,i,arr){

return 新值

})

过滤和汇总:

过滤: 选取出原数组中符合条件的元素，组成新数组返回，原数组保持不变。

var subArr=arr.filter(function(elem,i,arr){

return 判断条件

})

汇总: 将数组中每个元素的值汇总统计出一个最终的结论

var result=arr.reduce(function(prev,elem,i,arr){

//prev: 截止当前位置之前的临时汇总值

return prev+elem;

})