正课:

1. ES5

1. ES5

保护对象:

1. 保护属性:

2. 保护对象结构: 3个级别:

1. 防扩展: 禁止给当前对象添加新属性

Object.preventExtensions(obj)

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: 1. Object.preventExtensions(obj)

2. 自动将所有属性的configurable都设置为false

3. 冻结: 在兼具密封的基础上，又禁止修改属性值

Object.freeze(obj)

原理: 1. Object.seal(obj)

2. 又自动将所有属性的writable都设置为false

Object.create(): 仅有父对象，也可创建子对象

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,{

//为child添加新自有属性

//语法相当于defineProperties

属性名:{

value:属性值,

开关: true/false,

... : ...

}

})

原理: 1. 创建一个空对象

2. 让空对象继承father

3. 为新对象添加自有属性

替换this:

何时: 如果一个函数调用时，其中的this不是想要的，就可替换为想要的对象.

如何: 3种:

1. 调用函数，并临时替换一次this:

call(): 要求传入函数的参数，是分别传入

apply(): 要求传入函数的参数，必须放在一个数组中整体传入。

何时: 如果参数值是放在一个数组中的

原理: 先打散数组为单个值，再注射进函数中

2. 基于原函数，创建一个一模一样的新函数，并永久绑定this:

var 新函数=原函数.bind(替换this的对象, 其他参数...)

1. 创建一个和原函数一模一样的新函数

2. 永久替换函数中的this为指定对象

何时: 替换回调函数中的this

数组API:

1. 查找元素: 查找一个指定元素在数组中的位置

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

用法同str.indexOf()

2. 判断: 判断数组中的元素是否符合要求:

1. 判断数组中的元素是否都符合要求:

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

//val->当前正在遍历到的元素值

//i->当前正在遍历的位置

//arr->当前正在遍历的数组对象

return 判断条件;

})

依次判断arr中每个元素，是否都符合“判断条件”的要求

执行过程: every自动拿着回调函数，在每个元素上执行一次

每次执行时，都自动传入: 当前元素值，当前位置, 当前数组对象

每次执行回调函数后，都返回对当前元素的判断结果。

只有所有元素经过判断都返回true时，every才返回true

2. 判断数组中是否包含符合要求的元素:

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

return 判断条件

})

3. 遍历: 依次对数组中每个元素执行相同的操作

1. 对原数组中的元素执行相同的操作：

arr.forEach(function(elem,i,arr){

操作

})

会自动拿着回调函数，在每个元素上执行一次！

2. 复制出原数组中的元素，执行相同操作后，放入新数组中，原数组保持不变。

var 新数组=arr.map(function(elem, i, arr){

return 新元素值;

})

map: 3件事:

1. 创建一个新的空数组

2. 自动在每个元素上调用一次回调函数，将返回的新值，放入新数组中相同位置

3. 返回新数组

4. 过滤和汇总:

1. 过滤: 复制出原数组中符合条件的元素，组成新数组返回

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

return 判断条件;

})

执行过程: filter会自动拿着回调函数在每个元素上执行一次

每次执行时，如果当前元素符合“判断条件”要求，就复制到新数组中。

2. 汇总: 将数组中的所有元素，经过统计，得出一个最终结论

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

return 将当前元素的内容汇总到prev中

},start);

//start: 开始的基础值

//prev: 截止目前位置，之前的元素的临时汇总值