正课:

1. ES5

数组API

2. ES6:

...

\*\*\*\*\*Promise

1. ES5:

数组API:

判断: 数组中的元素是否符合条件

every(): 判断数组中所有元素是否都符合条件

some(): 判断数组中是否包含符合条件的元素

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

//this->window

//回调函数会在每个元素上自动调用一次

//每次调用时:

//elem可自动获得当前元素值

//i 可自动获得当前位置

//arr 可自动获得当前.前的数组对象

return 判断条件

})

必须所有元素经过回调函数的检查，都返回true，every整体才返回true

遍历: 对数组中每个元素依次执行相同的操作

forEach(): 对原数组中每个元素执行相同的操作

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

arr[i]=新值

})

map(): 取出原数组中每个元素值，执行相同操作后，放入新数组中。原数组保持不变!

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

return 新值

});

过滤和汇总:

过滤: 选取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 sub=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成一个最终的结果

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

//prev: 保存截止目前，临时的汇总值

//将当前元素的值，和prev累加，再返回到prev中

prev<=return prev+elem;

}, base);

2. ES6:

在不改变原理的基础上，纯简化了代码

let: 代替var声明变量:

特点:

1. 不会被声明提前

let会检查在当前作用域内，let之前不允许提前使用该变量

2. 相同作用域内不允许重复let第二个变量

3. 仅在当前块作用域内有效

问题: 块内的变量即使不执行，也会被提前到块外部，污染外部！

解决: 用let代替var，将变量限定在块内

let为js增加了块作用域: if else else if for while do...while switch..case

原理: 匿名函数自调

何时: 只要声明变量，一律用let

用let+for实现闭包结构:

for(let i=0;i<xxx.length;i++){  
 //(function(i){  
 循环体  
 //})(i);  
 }

参数增强:

1. 默认值:

.indexOf(val, fromi=0){

//fromi=fromi||0;

}

强调: 带默认值的参数必须放结尾

2. rest: 剩余: 代替arguments

arguments的问题:

1. 不是数组,而是类数组对象! 用不了数组API

2. arguments只能获得全部参数! 无法有选择的获得部分参数

3. arguments名字太长!

解决: rest

何时: 今后，只要代替arguments获得不确定个数的参数值时，都用rest

如何: 在定义函数时的参数列表中:

function fun(参数1, 参数2, ...数组名)

兼容:

类数组对象转为数组:

var arr=Array.prototype.slice.call(arguments)

//arguments.slice()//完整复制arguments

强调: rest的参数数组必须放在参数列表结尾

3. spread: 散播: 代替apply 打散数组类型的参数

为什么: apply的主要目的不是打散数组类型参数，而是替换函数中的this。只不过，可以顺便打散数组类型参数

何时: 如果专门只希望打散数组类型参数，不附加其它操作时。

如何: 在调用函数时的参数值()内

函数(...数组)

兼容: 函数.apply(null,数组)

箭头函数: 对一切回调函数和匿名函数的简化

1. 去function换箭头=>

2. 如果只有一个参数，可省略()

3. 如果只有一句话，可省略{}

如果仅有的一句话还是return，则可省略return

特点: 箭头函数内外共用this!

优: 不用bind，也能让内外this统一

问题: 有时，反而希望内外this不通用

比如: btn.onclick=function(){ ... this->btn ... }

()=>{ ... this->window ...} 错误

e=>{ ... e.target->btn ...} 正确

模板字符串: 简化的是大段字符串复杂字符串的拼接

何时: 只要复杂的字符串拼接，都用模板字符串

如何:

整个字符串被` `包裹

在``内支持换行,"",'',而不会发生字符冲突

需要动态拼接表达式的值: ${表达式}