2018.3.26 day-00

课程目录:

### JS数据类型：  
 基本数据类型：number string boolean null undefined  
 Object function...  
 Symbol(es6)

数组方法：数组的变异（能改变原数组） (pop push unshift shift splice reverse sort)  
 indexof lastIndexof concat slice  
  
 forEach filter(过滤) map（映射） some every reduce (includes find)  
  
### node的版本（升级到8.5以上） webStorm(2017版本以上) vscode sublime  
  
##框架和库  
-框架 vue 拥有完整的解决方案 我们写好人家调用我  
-库 jquery underscore zepto animate.css 我们调用他  
  
##渐进式（渐进增强）  
-vue全家桶 vue.js+vue-router+vuex+axios  
-通过组合 完成一个完整的框架

1. *forEach，for，for in,for of 区别***let *arr***=[1,2,3,4,5];  
   ***arr***.**b**=**'bb'**;  
   **for**(**let *i***=0;***i***<***arr***.length;***i***++){*//编程式* ***console***.log(***arr***[***i***]);*//1,2,3,4,5*}  
   *1）forEach 不支持return*

**let *arr***=[1,2,3,4,5];***arr***.forEach(**function** (item,index){*//声明式（不关心如何实现）* ***console***.log(item);  
});

**for**(**let *key* in *arr***){*//key会变成字符串类型,包括数组的私有属性b也可以打印出来* ***console***.log(**typeof *key***);*//6个string* ***console***.log(***key***);*//0,1,2,3,4,b*}

**for** (**let *value* of *arr***){*//支持return 并且是值of数组（不能遍历对象）* ***console***.log(***value***);*//1,2,3,4,5*}

**let *obj***={**school**:**'zfpx'**,**age**:8};*//Object.keys将对象的key作为新的数组  
//['school','age']***for**(**let *val* of *Object***.keys(***obj***)){  
 ***console***.log((***obj***[***val***]));*//得到的是属性值 'zfpx',8  
 //得到属性值有两种方法：一种是obj.aa,一种是obj[aa]*}

1. *filter （想删除某项）是否操作原数组：不 返回结果：过滤后的新数组 回调函数的返回结果：如果返回true 表示这一项放到新数组里*

**let *arr2***=[1,2,3,4,5];  
**let *newArr***=***arr2***.filter(**function** (item,index) {  
 **return** item<5&&item>2;  
});  
***console***.log(***newArr***);*//[3,4] arr2不变*

*三.map（想更新某项） 映射 将原有的数组映射成一个新数组  
//不操作原数组 返回新数组 回调函数中返回什么这一项就是什么***let *arr3***=[1,2,3].map(**function** (item) {  
 *// return 2;//[2,2,2]  
 // return item\*=3;//[3,6,9]* **return `<li>**${item}**</li>`**;*//``是es6中的模版字符串语法 遇到变量用${}取值  
 //'<li>1</li>', '<li>2</li>', '<li>3</li>'*});  
***console***.log(***arr3***.join(**''**));*//<li>1</li><li>2</li><li>3</li>  
  
join() 方法用于把数组中的所有元素放入一个字符串。  
// 元素是通过指定的分隔符进行分隔的。  
// 指定分隔符方法join("#");其中#可以是任意  
  
split()方法：用于把一个字符串分割成字符串数组.  
 // var str="how are you?";  
 // document.write(str.split("")+"<br />");  
 // document.write(str.split(" ")+"<br />");  
 // document.write(str.split("",3)+"<br />");  
// 返回  
// h,o,w, ,a,r,e, ,y,o,u,?  
// how,are,you?  
// h,o,w*

*//4)includes(es6) 返回的是boolean***let *arr4***=[1,2,3,4,55,555];  
***console***.log(***arr4***.includes(5));*//false  
  
//5)find(es6) 返回找到的那一项,不会改变数组 回调函数中返回true表示找到了，找到后停止循环，找不到返回的是undefined  
//indexOf() 查找n是否在数组里存在，如果存在返回这一项索引，如果不存在返回-1  
//toString() 将数组转换成字符串***let *result***=***arr4***.find(**function** (item, index) {*//找到具体的某一项用find* **return** item.toString().indexOf(5)>-1;*//在字符串里找5*})  
***console***.log(***result***);*//55  
  
//6)some 找true 找到true后停止 返回true 找不到返回false  
// every 找false 找到false后停止 返回false  
//将上面find变成some every即可得到结果  
  
//7)reduce 收敛 4个参数 返回的是叠加后的结果 原数组不变 回调函数返回的结果：  
//第一次：prev 代表的是数组的第一项 next 是数组的第二项  
//第二次：prev 是undefined,next是数组的第三项***let *sum***=[1,2,3,4,5].reduce(**function** (prev,next,index,ary) {  
 ***console***.log(***arguments***);  
 *// return 100;//本次的返回值 会作为下一次的prev* **return** prev+next;  
});  
  
**let *sum2***=[{**price**:30,**count**:2},{**price**:30,**count**:3},{**price**:30,**count**:4}].reduce(**function** (prev, next) {  
 *//0+60  
 //60+90  
 //150+120* ***console***.log(prev,next);  
 **return** prev+next.**price**\*next.**count**;  
},0);*//默认指定第一次的prev为0****console***.log(***sum2***);  
  
*//二维数组变成一维数组  
// let arr5=[[1,2,3],[4,5,6],[7,8,9]].join(',').split(',');//[ '1', '2', '3', '4', '5', '6', '7', '8', '9' ]***let *arr5***=[[1,2,3],[4,5,6],[7,8,9]].reduce(**function** (prev, next) {  
 **return** prev.concat(next);  
})  
***console***.log(***arr5***);*//[ 1, 2, 3, 4, 5, 6, 7, 8, 9 ]*

## *1.功能都是改变this指向 call,apply和bind的区别*

①：函数.call(对象,arg1,arg2....)

②：函数.apply(对象，[arg1,arg2,...])

③：var ss=函数.bind(对象,arg1,arg2,....)

通过观察下面的代码，很明显的就可以得出它们三者的区别，仅仅是函数传递的不同以及bind方法可以更加的方便的使用

1. **<script** type="text/javascript"**>**
2. function show(sex){
3. console.log("普通函数"+sex);
4. }
5. var person={
6. name:"aa",
7. age:14
8. };
9. show.call(person,"男");
10. show.apply(person,["女"]);
11. //对于bind来说，用法更加的灵活
12. var ss=show.bind(person,"不明");
13. ss();
14. **</script>**
15. *改变this指向：var —\_this=this;*

box.onclick = function(){

function fn(){

　　　　alert(this); 　//此时的this指向的是window

　}

　fn();

};

改变如下：

box.onclick = function(){

Var \_this=this;

function fn(){

　　　　alert(\_this); 　//此时的\_this指向的是box

　}

　fn();

};

1. *改变this指向：()=> 箭头函数*

*https://blog.csdn.net/yangbingbinga/article/details/61424363*

1. var obj = {
2. say: function () {
3. setTimeout(() => {
4. console.log(this)
5. });
6. }
7. }
8. obj.say(); // obj

此时的 this继承自obj, 指的是定义它的对象obj, 而不是 window!

多层嵌套箭头函数

1. var obj = {
2. say: function () {
3. var f1 = () => {
4. console.log(this); // obj
5. setTimeout(() => {
6. console.log(this); // obj
7. })
8. }
9. f1();
10. }
11. }
12. obj.say()

因为f1定义时所处的函数 中的 this是指的 obj, setTimeout中的箭头函数this继承自f1, 所以不管有多层嵌套,都是 obj