day15-javascript ES5&ES6

笔记本： 凡尘：二阶段
创建时间： 2017/8/27 13:00 **更新时间：** 2017/9/1 14:43
作者： 437389128@qq.com

严格模式

001、回顾

1、回顾ES5中的严格模式

2、如何开启严格模式

3、回顾数组方法　　回顾字符串方法

4、回顾作用域&作用域链
　　a、什么是作用域？

　　变量或者函数的可访问范围

　　作用域是一个对象，存放当前区块里面的变量和函数

　　作用域有两种：
　　全局作用域（window对象）
　　局部作用域（函数内部）

　　b、什么是作用域链？（面试必问）

　　1、简单说就是作用域集合

　　2、当前作用域 -> 父级作用域 -> ... -> 全局作用域　　形成的作用域链条

002、严格模式做了哪些限制

1、变量必须使用var定义　（很有帮助，强制规范代码）

　　2、禁止this关键字指向全局对象（window）

　　3、禁止删除变量

```
/*
    js的执行过程

    1、预编译　（变量提升=》将var定义的变量和使用function定义的函数提升到当前作用域的最上方）

    2、执行代码（从最上方开始（赋值语句或者执行语句））
*/
```

003、神奇的块级作用域（重）

1、什么是块级作用域？
使用 {} 括起来的区域 称之为块级作用域（{}对象除外）

比如：

```
if(){ }
for(){  }
switch(){}
...
```

详情请点击：http://blog.csdn.net/huangjq36sysu/article/details/51085674
（说明：变量提升总结有错误，请忽略）

2、块级作用域的特点
在块级作用中 使用let 定义的变量或者函数 在 {}外部 不可访问

demo:

```
if(true){
     let a = 10;
}
console.log(a);  //ReferenceError: a is not defined
```

let&const(重)

1、let用于声明块级变量

```
if(false){
    let a = 10; //只能在当前的代码块{}中使用
}
console.log(a);//报错

for(let i=0; i<4; i++){

}
console.log(i); //报错
```

注意：
1、同一个{}中不能使用let重复定义一个变量
demo:

```
let a = 10;
let a = 5; //报错
```

2、let不会进行变量提升

2、const用于声明常量（不可改变的变量）

```
const PI = 3.14;
PI = 3.1415926; //报错(Assignment to constant variable.)

HOST_NAME
```

字符串API扩展

1、字符串太长需要换行怎么办？
常规解决方案：

```
var a = '<div>'+
          '<span>'+num+'</span>'+
        'div>';
```

ES6神器：

```
var b = `
          <div>
               <span></span>
          </div>
        `;
```

2、字符串拼接太麻烦怎么办？

ES6神器（字符串模板）：

```
var phone = 18200000000;
var intro = `my name is pine, my phone is ${phone}`; //注意是以`开头和结尾的哦，不是'
console.log(intro); //${phone}被替换成18200000000
```

说明：

1、使用 `` 反单因号 代替 '' 或者 ""
2、使用 ${变量} 实现变量拼接

3、includes 字符串搜索
之前使用indexOf进行查找，利用的是indexOf方法的特性，找打了返回下标，找不到返回-1，所以每次你需要这么写：

```
var str = 'abcd';
if( str.indexOf('c') > -1 ){}  //需要判断是否>-1
```

ES6神器：includes方法

```
str.includes(查找的内容);  找到返回true，找不到返回false
demo:
     var str = 'good method!';
     str.includes('method'); //true
```

4、判断首尾 startsWith endsWith
startsWith用于判断是否位于头部，endsWith判断是否位于尾部，可以说这两个方法是includes方法的扩展；

```
demo:
     let  str = 'how are you?';
     str.startsWith('how');//true
     str.endsWith('?');//true
```

5、repeat 字符串重复（懒人福利）
str.repeat(n); 将字符串重复n次（n是整数）
demo:

```
let str = 'money';
str.repeat(2); // 'moneymoney'
```

对象（非常重要）

1、什么是对象
万物皆对象
一间教室 一个人 一棵树 。。。

一个人：

对象的属性（静态描述）

名字：pine
肤色： 黑色 、 黄色 白色
头发：短发 长发 光头
身高：175 165
体重：180 120 150

对象的方法（function） （也是属性）

睡觉
吃饭
打痘痘

动作 => 功能 => function(){}

2、js怎么定义对象？

JSON数据格式：由数组和对象组成

```
{
    "name": "wenhao"
}
```

1、JSON格式（常用）

```
var person = {
    name: '张三'
};
```

2、构造函数方式（基本不用）

```
var dog = new Object();
dog.name = '小黄毛';
```

3、什么是对象的属性和方法？

4、怎么访问对象的属性和方法？

1、如果属性名确定的话（不是变量）

```
var person = {
    name: 'pine'
};
person.name
person['name']
```

2、不知道属性名是什么（属性名是变量）

```
var person = {
    name: 'pine',
    age: 18
};

var attr = 'age';
person[attr] //person['age']
```

解构赋值（重）

1、什么是解构赋值？

解构赋值可将数组的元素或对象的属性赋予给另一个变量，该变量的定义语法与数组字面量或对象字面量很相似。

2、解构数组
现有数组：var arr = [13,22,34];
如果我们希望用3个变量得到数组中对应的值，我们会这么写：

```
var first = arr[0],
    second = arr[1],
    third = arr[2];
```

解构赋值：

```
var [first,second,third] = arr; //first=13  second = 22  third = 34
```

实际场景剖析：

```
<input type="text" class="username">
<input type="text" class="phone">
```

如果我们想用变量得到username和phone
我们可以这么写：

```
<script>
    var [uname,phone] = document.querySelectorAll('input');
</script
```

3、解构对象（先了解）

```
var classRoom = {
    id: '1609',
    personNum: 34
};

var {id,personNum} = classRoom;
console.log(id,personNum);// id=1609  personNum=34
```

实例场景：

1、用变量接收对象中的数据

```
var result = {
    status: 1,
    data: [10000,8000,5000]
};

var {status,data} = result;
```

详情请查看：http://www.csdn.net/article/2015-07-07/2825149-es6-in-depth-destructuring
博客推荐：https://segmentfault.com/a/1190000002920859（详细）
博客推荐：http://es6.ruanyifeng.com/#docs/destructuring（阮一峰）

实战

1、交换两个变量的值

=>函数

1、什么是箭头函数？
用箭头定义的函数，类似于匿名函数

写法简单

2、简单箭头函数（代码块中只有一行代码）

```
var fn = function(a){
    return a*a;
}
```

箭头函数：

```
var fn = (a) => a*a;
```

3、多个参数箭头函数

```
var add = function(a,b){
    return a+b;
}
```

箭头函数：

```
var fn = (a,b) => a+b;
```

4、代码块中有多行代码

```
var getSum = function(arr){
    var sum = 0;
    for(var i=0,len=arr.length; i<len; i++){
        sum += arr[i];
    }
    return sum;
}
```

箭头函数：

```
var fn = (arr) => {
    var sum = 0;
    for(var i=0,len=arr.length; i<len; i++){
        sum += arr[i];
    }
    return sum;
}
```

实战演示：

```
var arr = [23,45,1,2];
/*arr.sort(function(a,b){
    return a - b;
});*/
arr.sort( (a,b) => a-b );
console.log(arr);

/*arr.map(function(v){
    return v+1;
});*/
arr = arr.map( v => v+1 );
console.log(arr);
```

this关键字/bind方法(重)

this的存在场景
1、全局范围内this　不在函数中指的就是window对象
2、function中的this
1、普通函数调用 (开启严格模式　是 undefined，否则是window)
2、对象的
3、事件的

谁调用，this就指向谁

```
div.onclick  = function(){
    console.log(this); // div   因为是div调用了onclick方法
}

var person = {
    name: 'pine',
    eat: function(){
         console.log(this); //当person调用时指向person对象
    }
};
person.eat(); // eat方法  暂时  只能由person调用
```

bind

bind:ECMAScript 5.1 (或仅 ES5) 是ECMAScript(基于JavaScript的规范)标准最新修正。其中，新增了一个名叫bind 函数扩展方法()。

bind这个方法应用在哪里那？

bind事件委托，更改this指向

扩展

```
set集合，本质上就是对数组的一种包装
let imgs = new Set();
imgs.add(1);
imgs.add(1);
imgs.add(5);
imgs.add("5");
imgs.add(new String("abc"));
imgs.add(new String("abc"));
console.log(imgs.size)
打印的结果:
1  5  '5'  'abc'  'abc'
```

创建构造函数的时候js执行了那些操作

```
var str = new String();
1、在内存中开辟了一块空间

2、把this的指向指向了它
```

预习

```
1、运动的原理是什么（元素是怎么动起来的）？
```

2、定时器（复习）

3、实现小球在一个盒子中乱撞（边界处理）