JavaScript进阶

---Array数组

河北师范大学软件学院
Software College of Hebei Normal University

内容提纲

- **数组的创建和基本操作（增删改查）**
- **稀疏数组与多维数组**
- **数组的方法和相关高阶函数**

数组的创建和基本操作

创建数组的方式

- 通过字面量的方式直接创建，直接量中的值可以是任意的表达式
- 通过Array构造函数来创建数组对象，注意传递的参数

```
var arr1 = [1,2,3,"4"];
var arr2 = new Array(5);
console.log(arr2);
for(var i=0;i<arr2.length;i++){arr2[i] = i;}
var arr3 = new Array(1,2,3,4);//多个参数
console.log(arr1,arr2,arr3);
```

思考：new Array(“5”)返回什么

思考：demo1中的异步初始化数组

```
▶(5) [undefined × 5]
▶(4) [1, 2, 3, "4"] ▶(5) [0, 1, 2, 3, 4] ▶(4) [1, 2, 3, 4]
```


参见实例demo1 创建数组实例

数组的创建和基本操作

- 数组元素的增删改查的基本操作

```
var a = ["hello"];
a[1] = 3.14;//增：直接添加数组元素，通过方法添加元素参见后续章节
a[2] = "world";
console.log("删除a[2]前的数组a",a);
delete a[2];//删：思考此时数组长度是2还是3？如何彻底删除？
console.log("删除a[2]后的数组a",a);
a[0] = "XX";//改：修改数组元素a[0]
console.log(a[0]);//查:看数组中的元素，索引从0开始
```

```
var i=2,b=[];
b[i]=3;
b[i+1]="YY";
b[b[i]] = b[0];
console.log(b);//输出什么？
```

思考左侧案例

参见实例demo2 Part1

数组的创建和基本操作

- 数组相对于普通对象的特别之处
  - 数组是对象的特殊形式，可以为数组添加对象属性，对于0至2的32次方之外的数，将作为普通对象的键来对待
  - 数组特别之处在于，当使用使用2的32次方以内的非负整数作为属性名时（包括类型转换的数字），数组会自动维护其length属性，作为数组的元素，而不是数组对象的属性

```
var a = [];
a[-1.23] = true; //创建一个名为“-1,23”的属性
a["100"] = 0;    // 数组的第101个元素
a[1.00] = "Hi"; //和a[1]相当
console.log(a.length);
console.log(a);
```

```
101
▶ (101) [undefined × 1, "Hi", undefined × 98, 0, -1.23: true]
```


参见实例demo2 Part2

内容提纲

- 数组的创建和基本操作（增删改查）
- 稀疏数组与多维数组
- 数组的方法和相关高阶函数

稀疏数组与多维数组

稀疏数组

- 稀疏数组是包含从0开始的不连续索引的数组，length值大于实际定义的元素的个数
- 遍历稀疏数组时，注意的跳过无元素项的问题

```
var a1 = [,"abc"];
console.log(a1.length);
for(var i in a1){
    console.log(i,a1[i]);//输出几个元素?
}

var a2 = new Array(3);
console.log(a2.length);//长度为几?
var a3 = [,,];
console.log(a3.length);//长度为几?
```


参见实例demo3 稀疏数组

稀疏数组与多维数组

- 多维数组（矩形数组、交错数组）

  - JS中可以通过包含数组的数组来模拟多维数组

```
var table = new Array(5);
for(var i=0;i<table.length;i++){
    table[i] = new Array(5);//若是table[i] = new Array(i);
}

for(var row=0;row<table.length;row++){
    for(var col=0;col<table[row].length;col++){
        table[row][col]=row*col;
    }
}
var product = table[2][4];
```

练习：优化代码，for循环合并

参见实例demo4 多维数组（矩形、交错）

内容提纲

- 数组的创建和基本操作（增删改查）
- 稀疏数组与多维数组
- **数组的方法和相关高阶函数**

数组的方法

- 数组静态方法（构造器函数对象的方法）
  - Array.from(...)、Array.isArray(...)、Array.of(...)等（具体参见实例）
- 数组原型方法（添加和删除元素-破坏性）
  - Array.prototype.shift()　Array.prototype.unshift(elem1?,elem2?,...)
  - Array.prototype.pop()　Array.prototype.push(elem1?,elem2?,...)
  - Array.prototype.splice(start,deleteCount?,elem1?,elem2?)

```
function foo(){
    console.log(Array.isArray(arguments));
    //console.log(arguments.pop());//这样是否能调用？数组与类数组对象
    console.log(Array.prototype.pop.call(arguments));
}
foo(3,2,5);
```


参见实例demo5 静态方法和添加删除元素的原型方法

数组的原型方法

- 数组原型方法（排序和颠倒元素顺序-破坏性）
  - Array.prototype.reverse()
  - Array.prototype.sort(compareFunction？) //回调函数的写法，思考冒泡排序
- 数组原型方法（合并、切分和连接-非破坏性）
  - Array.prototype.concat(arr1?,arr2?,...)
  - Array.prototype.slice(begin?,end?) //注意参数的正负，注意不要和splice混淆了
  - Array.prototype.join(separator?) //注意返回的类型
- 数组原型方法（值的查找-非破坏性）
  - Array.prototype.indexOf(searchValue,startIndex?)
  - Array.prototype.lastIndexOf(searchElement,startIndex?) //注意方向和起始点

参见实例demo6 数组原型方法

数组相关的高阶函数

- 数组原型方法（迭代-非破坏性-检测方法）
  - Array.prototype.forEach(callback,thisValue?)
  - Array.prototype.every(callback,thisValue?) //若有不满足的，立即返回false，不再后续迭代
  - Array.prototype.some(callback,thisValue?) //若有满足的，立即返回true，不再后续迭代
- 数组原型方法（迭代-非破坏性-转换方法）
  - Array.prototype.map(callback,thisValue?)
  - Array.prototype.filter(callback,thisValue?)
- 数组原型方法（迭代-非破坏性-归约方法）
  - Array.prototype.reduce(element,initialValue?)
  - Array.prototype.reduceRight(callback,initialValue?)

参见实例demo7 数组相关的高阶函数 迭代案例

Thank You!