20190723课堂笔记

一、Math

> 数学函数：但是它不是一个函数，它是一个对象，对象中存储了很多操作数字的属性方法，因此被称为数学函数

```
console.log(typeof Math); //=>"object"
console.dir(Math);
/*
 * Math = {
 *     PI:3.141592653589793,
 *     abs:function(){[native code]},
 *     ceil:function(){[native code]},
 *     ...
 * }
 *
 * Math.abs();
 * Math.PI;
 */
```

Math中常用的属性和方法

1. Math.abs([number value])

> 获取绝对值（绝对值永远是正数或者零）

```
console.log(Math.abs(-12.5)); //=>12.5
console.log(Math.abs(12)); //=>12
console.log(Math.abs(0)); //=>0
// 传递的不是数字类型的值：先基于Number()转换为数字再处理
console.log(Math.abs('-1')); //=>1
console.log(Math.abs('-1px')); //=>NaN
console.log(Math.abs(true)); //=>1
```

2. Math.ceil / floor([number value])

> 把一个数向上取整 / 向下取整

```
console.log(Math.ceil(12)); //=>12
console.log(Math.ceil(12.1)); //=>13
console.log(Math.ceil(12.9)); //=>13
console.log(Math.ceil(-12.1)); //=>-12
console.log(Math.ceil(-12.9)); //=>-12

console.log(Math.floor(12)); //=>12
console.log(Math.floor(12.1)); //=>12
console.log(Math.floor(12.9)); //=>12
console.log(Math.floor(-12.1)); //=>-13
console.log(Math.floor(-12.9)); //=>-13
```

3. Math.round()

> 四舍五入

```
console.log(Math.round(12));  //=>12
console.log(Math.round(12.1)); //=>12
console.log(Math.round(12.5)); //=>13  正数中.5属于入
console.log(Math.round(12.9)); //=>13
console.log(Math.round(-12.1)); //=>-12
console.log(Math.round(-12.5)); //=>-12  负数中.5属于舍
console.log(Math.round(-12.9)); //=>-13
```

4. Math.max / min ([val1],[val2],...)

> 获取一堆数中的最大值和最小值

```
console.log(Math.max(12, 5, 68, 23, 45, 3, 27)); //=>68
console.log(Math.min(12, 5, 68, 23, 45, 3, 27)); //=>3

//思考题：如何基于Math.max/min获取数组中的最大值最小值？
Math.max([12, 5, 68, 23, 45, 3, 27]); //=>NaN 此处是只传第一个值，是个数组，和内置的语法要求不符
```

5. Math.sqrt / pow()

> sqrt：给一个数开平方
>
> pow：计算一个数的多少次幂

```
console.log(Math.sqrt(9)); //=>3  符合N*N=M 这样的M才能整开平方
console.log(Math.sqrt(-9)); //=>NaN 负数开不了平方
console.log(Math.pow(2, 10)); //=>1024
```

6. Math.random()

> 获取0~1之间的随机小数

```
for (let i = 1; i <= 10; i++) {
    console.log(Math.random());
}
/*
 * 0.09453770227521763
 * 0.06700581113042259
 * 0.10092020814995206
 * ...
 */
```

扩展：获取 [n~m] 之间的随机整数

> 包含n也包含m
>
> n<m

```
Math.round(Math.random()*(m-n)+n)
```

数组及数组中常用的方法

> 数组是对象数据类型的，它属于特殊的对象

```
let ary = [12, 23, 34, 45];
console.log(typeof ary); //=>"object"
console.dir(ary);
/*
 * ary = {
 *    0:12,
 *    1:23,
 *    2:34,
 *    3:45,
 *    length:4
 * }
 *
 * 数字作为索引（KEY 属性名）
 * length代表长度
 *
 * ary[0] 根据索引获取指定项的内容
 * ary.length 获取数组的长度
 * ary.length-1 最后一项的索引
 */
```

数组中常用的方法

- 方法的作用和含义
- 方法的实参（类型和含义）
- 方法的返回值
- 原来的数组是否会发生改变

1.实现数组增删改的方法

> 这一部分方法都会修改原有的数组

`push`

```
/*
 * push : 向数组末尾增加内容
 * @params
 *    多个任意类型
 * @return
 *    新增后数组的长度
 */
let ary = [10, 20];
let res = ary.push(30, 'AA');
// 基于原生JS操作键值对的方法，也可以向末尾追加一项新的内容
ary[ary.length] = 40;
console.log(res, ary); //=>4 [10,20,30,'AA',40]
```

`unshift`

```
/*
 * unshift : 向数组开始位置增加内容
 * @params
 *    多个任意类型
 * @return
 *    新增后数组的长度
 */
let ary = [10, 20];
let res = ary.unshift(30, 'AA');
```

```
console.log(res, ary); //=>4
[30,'AA',10,20]

// 基于原生ES6展开运算符，把原有的ARY克隆一份，在新的数组中创建第一项，其余的内容使用原始ARY中的信息即可，也算实现了向开始追加的效果
ary = [100, ...ary];
console.log(ary); //=>
[100,30,'AA',10,20]
```

shift

```
/*
 * shift : 删除数组中的第一项
 * @params
 * @return
 *    删除的那一项
 */
let ary = [10, 20, 30, 40];
let res = ary.shift();
console.log(res, ary); //=>10  [20, 30, 40]

// 基于原生JS中的DELETE，把数组当做普通的对象，确实可以删除掉某一项内容，但是不会影响数组本身的结构特点（length长度不会跟着修改）,真实项目中杜绝这样的删除使用
delete ary[0];
console.log(ary); //=>
{1:30,2:40,length:3}
```

pop

```
/*
 * pop : 删除数组中的最后一项
 * @params
 * @return
 *    删除的那一项
 */
let ary = [10, 20, 30, 40];
let res = ary.pop();
console.log(res, ary); //=>40
[10,20,30]

// 基于原生JS让数组数组长度干掉一位，默认干掉的就是最后一项
ary.length--; //=>ary.length = ary.length - 1;
console.log(ary);
```

splice

```
/*
 * splice : 实现数组的增加、删除、修改
 * @params
 *    n,m 都是数字   从索引n开始删除m个元素（m不写，是删除到末尾）
 * @return
 *    把删除的部分用新数组存储起来返回
 */
let ary = [10, 20, 30, 40, 50, 60, 70, 80, 90];
```

```
let res = ary.splice(2, 4);
console.log(res, ary); //=>[30, 40, 50, 60]  [10, 20, 70, 80, 90]

// 基于这种方法可以清空一个数组，把原始数组中的内容以新数组存储起来（有点类似数组的克隆：把原来数组克隆一份一模一样的给新数组）
/* res = ary.splice(0);
console.log(res, ary);//=>[10, 20, 70, 80, 90] [] */

// 删除最后一项和第一项
ary.splice(ary.length - 1);
ary.splice(0, 1);
console.log(ary);
```

```
/*
 * splice : 实现数组的增加、修改
 * @params
 *    n,m,x  从索引n开始删除m个元素，用x占用删除的部分
 *    n,0,x  从索引n开始，一个都不删，把x放到索引n的前面
 * @return
 *    把删除的部分用新数组存储起来返回
 */
let ary = [10, 20, 30, 40, 50];
let res = ary.splice(1, 2, '珠峰培训', '哈哈哈');
```

```
console.log(res, ary); //=> [20,30] [10,'珠峰培训','哈哈哈', 40, 50]

// 实现增加
ary.splice(3, 0, '呵呵呵');
console.log(ary); //=>[10, "珠峰培训", "哈哈哈", "呵呵呵", 40, 50]

// 向数组末尾追加
ary.splice(ary.length, 0, 'AAA');

// 向数组开始追加
ary.splice(0, 0, 'BBB');
```

2.数组的查询和拼接

> 此组学习的方法，原来数组不会改变

`slice`

```
/*
 * slice : 实现数组的查询
 * @params
 *    n,m 都是数字 从索引n开始，找到索引为m的地方（不包含m这一项）
 * @return
 *    把找到的内容以一个新数组的形式返回
 */
let ary = [10, 20, 30, 40, 50];
let res = ary.slice(1, 3);
console.log(res); //=>[20,30]
```

```
11
12 // m不写是找到末尾
13 res = ary.slice(1);
14 console.log(res); //=>[20, 30, 40, 50]
15
16 // 数组的克隆，参数0不写也可以
17 res = ary.slice(0);
18 console.log(res); //=>[10, 20, 30, 40,
   50]
19
20 // 思考：1.如果n/m为负数会咋地，如果n>m了会咋
   地，如果是小数会咋地，如果是非有效数字会咋地，如
   果m或者n的值比最大索引都会咋地？ 2.这种克隆方式
   叫做浅克隆，可以回去先看看深度克隆如何处理！
```

`concat`

```
1  /*
2   * concat : 实现数组拼接
3   * @params
4   *    多个任意类型值
5   * @return
6   *    拼接后的新数组（原来数组不变）
7   */
8  let ary1 = [10, 20, 30];
9  let ary2 = [40, 50, 60];
10 let res = ary1.concat('珠峰培训', ary2);
11 console.log(res);
```

3.把数组转换为字符串

> 原有数组不变

`toString`

```
/*
 * toString : 把数组转换为字符串
 * @params
 * @return
 *    转换后的字符串，每一项用逗号分隔（原来数组不变）
 */
let ary = [10, 20, 30];
let res = ary.toString();
console.log(res); //=>"10,20,30"
console.log([].toString()); //=>""
console.log([12].toString()); //=>"12"
```

`join`

```
/*
 * join : 把数组转换为字符串
 * @params
 *    指定的分隔符（字符串格式）
 * @return
 *    转换后的字符串（原来数组不变）
 */
let ary = [10, 20, 30];
let res = ary.join('');
console.log(res); //=>"102030"

res = ary.join();
```

```
console.log(res); //=>"10,20,30"

res = ary.join('|');
console.log(res); //=>"10|20|30"

res = ary.join('+');
console.log(res); //=>"10+20+30"
console.log(eval(res)); //=>60  eval把字符串变为JS表达式执行
```

4.检测数组中的是否包含某一项

`indexOf / lastIndexOf / includes`

```
/*
 * indexOf / lastIndexOf : 检测当前项在数组中第一次或者最后一次出现位置的索引值（在IE6~8中不兼容）
 * @params
 *    要检索的这一项内容
 * @return
 *    这一项出现的位置索引值（数字），如果数组中没有这一项，返回的结果是-1
 * 原来数组不变
 */
let ary = [10, 20, 30, 10, 20, 30];
console.log(ary.indexOf(20)); //=>1
console.log(ary.lastIndexOf(20)); //=>4

// 想验证ARY中是否包含'珠峰培训'
if (ary.indexOf('珠峰培训') === -1) {
```

```
    // 不包含
}
// 也可以直接使用ES6新提供的includes方法判断
if (ary.includes('珠峰培训')) {
    // 包含：如果存在返回的是TRUE
}
```

5.数组的排序或者排列

`reverse`

```
/*
 * reverse : 把数组倒过来排列
 * @params
 * @return
 *    排列后的新数组
 * 原来数组改变
 */
let ary = [12, 15, 9, 28, 10, 22];
ary.reverse();
console.log(ary); //=>[22, 10, 28, 9, 15, 12]
```

`sort`

```
/*
 * sort : 实现数组的排序
 * @params
 *    可以没有，也可以是个函数
 * @return
 *    排序后的新数组
```

```
 * 原来数组改变
 */
let ary = [7, 8, 5, 2, 4, 6, 9];
ary.sort();
console.log(ary); //=>[2, 4, 5, 6, 7, 8, 9]

// SORT方法中如果不传递参数，是无法处理10以上数字排序的(它默认按照每一项第一个字符来排，不是我们想要的效果)
/* ary = [12, 15, 9, 28, 10, 22];
ary.sort();
console.log(ary); //=> [10, 12, 15, 22, 28, 9] */

// 想要实现多位数正常排序，需要给SORT传递一个函数，函数中返回 a-b 实现升序，返回 b-a 实现降序
ary = [12, 15, 9, 28, 10, 22];
// ary.sort(function(a,b){ return a-b; });
ary.sort((a, b) => a - b);
console.log(ary);
```

6.遍历数组中每一项的方法

`forEach`

```
/*
 * forEach：遍历数组中的每一项内容
 * @params
 *     回调函数
```

```
 * @return
 *
 * 原来数组不变
 */
let ary = [12, 15, 9, 28, 10, 22];

/* // 基于原生JS中的循环可以实现
for (let i = 0; i < ary.length; i++) {
    // i:当前循环这一项的索引
    // ary[i]:根据索引获取循环的这一项
    console.log('索引: ' + i + ' 内容: ' + ary[i]);
} */

ary.forEach((item, index) => {
    // 数组中有多少项，函数就会被默认执行多少次
    // 每一次执行函数：item是数组中当前要操作的这一项，index是当前项的索引
    console.log('索引: ' + index + ' 内容: ' + item);
});
```

`map`

`filter`

`find`

`reduce`

`some`

`every`

.....

Array.prototype 在控制台查看数组中所有提供的方法，可以基于MDN网站去查询方法的用法