内置对象

- JavaScript 中的对象分为3种：自定义对象 、内置对象、 浏览器对象
- 内置对象就是指 JS 语言自带的一些对象，这些对象供开发者使用，并提供了一些常用的或是最基本而必要的功能
- JavaScript 提供了多个内置对象：Math、 Date 、Array、String等

查阅文档

学习一个内置对象的使用，只要学会其常用成员的使用即可，我们可以通过查文档学习，可以通过MDN/W3C来查询

MDN: https://developer.mozilla.org/zh-CN/

1. 查阅该方法的功能
2. 查看里面参数的意义和类型
3. 查看返回值的意义和类型
4. 通过 demo 进行测试

Math对象

Math 对象不是构造函数，它具有数学常数和函数的属性和方法。跟数学相关的运算（求绝对值，取整、最大值等）可以使用 Math 中的成员。

通过MDN上的Math对象 里面的 max 方法我们可以通过练习来熟悉

```
    <script>
        // Math数学对象，不是一个构造函数，所以我们不需要new 来调用，而是直接使用里面的属性和方法即可

        //Math.PI    圆周率      是静态属性，使用中不带括号
        console.log(Math.PI);

        var re = Math.max(1,5,23,123,0);
        //Math.max()   函数返回一组数中的最大值       是方法，需要带括号。
        console.log(re);
    </script>
```

```
    <script>
        function getMaxOfArray(numArray) {
            return Math.max.apply(null, numArray);
        }
        var a = getMaxOfArray([1,0,23,123,1231234,5454545]);
        console.log(a);
    </script>
```

封装自己的数学对象

通过函数的封装，我们也可以构建自己的数学对象。

```
<script>
        var myMath = {
            PI: 3.141592653,
            max: function() {
                var max = arguments[0];
                for (var i = 1; i < arguments.length; i++) {
                    if (arguments[i] > max) {
                        max = arguments[i];
                    }
                }
                return max;
            },
            min: function() {
                        var min = arguments[0];
                        for (var i = 1; i < arguments.length; i++) {
                            if (arguments[i] < min) {
                                min = arguments[i];
                            }
                        }
                        return min;
                    }
        }
        console.log(myMath.PI);
        console.log(myMath.max(1, 5, 9));
        console.log(myMath.min(1, 5, 9));
    </script>
```

Math的绝对值和三个取整方法

Math.floor() 向下取整

Math.ceil() 向上取整

Math.round() 四舍五入，就近取整

Math.abs() 绝对值

```

    <script>
        //绝对值
        console.log(Math.abs(1));
        console.log(Math.abs(-1));
        console.log(Math.abs('-1'));    //隐式转换，会把字符串 -1 转换为数字型
        console.log(Math.abs('string'));       //NaN

        // round()，四舍五入，但是.5特殊，会往大了取。
        console.log(Math.round(1.1));       //1
        console.log(Math.round(1.5));       //2
        console.log(Math.round(1.9));       //2
        console.log(Math.round(-1.1));      //-1
        console.log(Math.round(-1.5));      //-1

```

```
        //向下取整
        console.log(Math.floor(1.1));       //1
        console.log(Math.floor(1.9));       //1

        //向上取整
        console.log(Math.ceil(1.1));        //2
        console.log(Math.ceil(1.9));        //2
    </script>
```

随机数方法 random()

函数返回一个浮点数，伪随机数在范围从**0到**小于**1**，也就是说，从0（包括0）往上，但是不包括1（排除1）。 0 =< x <1

```
    <script>
        console.log(Math.random());
    </script>
```

例子：

```
    <script>
        function getRandomInt(min, max) {
            min = Math.ceil(min);
            max = Math.floor(max);
            return Math.floor(Math.random() * (max - min)) + min; //不含最大值，含最小值
        }
        var g = getRandomInt(1,10);
        console.log(g);
    </script>
```

案例:随机点名

```
    <script>
        function getRandomInt(min, max) {
            min = Math.ceil(min);
            max = Math.floor(max);
            return Math.floor(Math.random() * (max - min)) + min; //不含最大值，含最小值
        }
        var arr = ['张三','李四','王二麻子'];
        var arrR = getRandomInt(0,2);   //通过索引值，进行判定
        console.log(arr[arrR]);
        // 优化方法
        var arr1 = ['张三','李四','王二麻子','甲','乙','丙','丁']
        var arrR1 = getRandomInt(0,arr1.length-1);
        console.log(arr1[arrR1]);
    </script>
```

Date()对象

- Date 对象和 Math 对象不一样，他是一个构造函数，所以我们需要实例化后才能使用
- Date 实例用来处理日期和时间

获取当前时间必须实例化

使用 `new`；来实例化对象

```
    <script>
        var now = new Date();
        console.log(now);
    </script>
```

Date()构造函数的参数

如果括号里面有时间，就返回参数里面的时间。例如日期格式`字符串`为 `‘2019-5-1’`，可以写成 `new Date('2019-5-1')` 或者 `new Date('2019/5/1')`

1. 如果Date()不写参数，就返回当前时间

```
    <script>
        var now = new Date();
        console.log(now);
    </script>
```

2. 如果Date()里面写参数，就返回括号里面输入的时间

    1.
```
    <script>
        // 2.参数常用的写法 数字型 2019,10,1  字符串型 '2019-10-1 8:8:8' 时分秒
        // 如果Date()里面写参数，就返回括号里面输入的时间
        var data = new Date(2019,10,1);
        console.log(data);  // Fri Nov 01 2019 00:00:00 GMT+0800 (中国标准时间)  ,  返回的是11月不是10月

        var data2 = new Date('2019-10-1 8:8:8');
        console.log(data2); // Tue Oct 01 2019 08:08:08 GMT+0800 (中国标准时间) ,有时间值的
    </script>
```

日期格式化

我们想要 2019-8-8 8:8:8 格式的日期，要怎么办?

需要获取日期指定的部分，所以我们要手动的得到这种格式

方法名	说明	代码
getFullYear()	获取当年	dObj.getFullYear()
getMonth()	获取当月(0-11)	dObj.getMonth()
getDate()	获取当天日期	dObj.getDate()
getDay()	获取星期几(周日0到周六6)	dObj.getDay()
getHours()	获取当前小时	dObj.getHours()
getMinutes()	获取当前小时	dObj.getMinutes()
getSeconds()	获取当前秒钟	dObj.gerSeconds()

```
    <script>
        var now = new Date();
        console.log(now.getFullYear()); //获取当前年份
        console.log(now.getMonth()+1);    //获取月份（0-11）,返回的月份小一个月 记得月份 +1
        console.log(now.getDate()); //返回的是几号
        console.log(now.getDay());  //周一返回1 周6返回六 周日返回0
        console.log(now.getHours());    //获取当前小时
        console.log(now.getMinutes());  //获取当前分钟数
        console.log(now.getSeconds());  //获取当前秒钟数
    </script>
```

例子：使用Date()对象，写一个当前时间。

```
    <script>
        var now = new Date();
        var year = now.getFullYear();
        var month = now.getMonth() + 1;
        var day = now.getDate();
        var hours = now.getHours();
        var minutes = now.getMinutes();
        var seconds = now.getSeconds();
        console.log('今天是'+year+'年'+month+'月'+day+'日'+hours+'点'+minutes+'分'+seconds+'秒');

    </script>
```

案例：封装一个函数返回当前的时分秒，格式 08:08:08

```
    <script>
        // 封装一个函数返回当前的时分秒    格式 08:08:08
        function getTime() {
            var time = new Date();
            var h = time.getHours();
            h = h < 10 ? '0'+h : h;
            var m = time.getMinutes();
            m = m < 10 ? '0'+m : m;
            var s = time.getSeconds();
            s = s < 10 ? '0'+s : s;
            return h+':'+m+':'+s;

```

```
        }
        console.log(getTime());
    </script>
```

获取日期的总的毫秒形式

Date对象是基于1970年1月1日 (世界标准时间) 起的毫秒数，获得Date总的毫秒数(时间戳)。

注意：不是当前时间的毫秒数，而是距离1970年1月1号过了多少毫秒数

1、通过valueOf()、getTime() 两种方法来获取

```
// 使用途径，可以用来生成订单数(少量订单)，因为不同时间的毫秒数是不一样的。
var date = new Date();
console.log(date.valueOf());     //是现在时间距离1970.1.1 总的毫秒数
console.log(date.getTime());
```

2、+new Date() 较为简单的写法

```
var date1 = +new Date();     // +new Date()    返回的也是总毫秒数
console.log(date1);
```

3、HTML5新增的方法，Date.now() 获取总毫秒数。

```
console.log(Date.now());     //H5新增的方法，也是获取距离1970年1月1号的总毫秒数。
```

案例：倒计时效果

思路：

1、截止的时间 减去 现在的时间 = 剩余的时间 （倒计时）

2、使用时间戳实现以下效果，用户输入时间的总毫秒数减去现在的总毫秒数，得到剩余时间的毫秒数。

3、把剩余的时间毫秒数转换为 天、时、分、秒 (需要将时间戳转换为时分秒)

公式如下：

- d = parseInt(总秒数/60/60/24); //计算天数
- h = parseInt(总秒数/60/60%24); //计算小时
- m = parseInt(总秒数/60%60);   //计算分数
- s = parseInt(总秒数%60);   //计算当前秒数

```
    <script>
        function demo(time) {
            var nowTime = +new Date();           //获取现在距离1970年的总毫秒数
            var input = +new Date(time);         //获取活动截止日期距离1970年的总毫秒数
            var times = (input - nowTime) / 1000;    //活动截止日期 - 现在时间 = 距离活动截止的倒计时
            var d = parseInt(times/60/60/24);  //计算天数
            d = d < 10 ? '0' + d : d;
            var h = parseInt(times/60/60%24);  //计算小时
            h = h < 10 ? '0' + h : h;
```

```
            var m = parseInt(times/60%60);     //计算分数
            m = m < 10 ? '0' + m : m;
            var s = parseInt(times%60);     //计算当前秒数
            s = s < 10 ? '0' + s : s;
            return d + '天' + h + '时' + m + '分' + s + '秒';
        }
        console.log(demo('2022-3-18 18:00:00'));
        var date = new Date();
        console.log(date);
    </script>
```

数组对象

数组对象的创建

创建数组对象的两种方式

- 字面量方式
- new Array()

字面量方式

```
    <script>
        var arr = [1,2,3,4];
    </script>
```

利用 new Array() 创建数组

```
    <script>
        var arr1 = new Array(); //创建了一个空数组
        var arr2 = new Array(2);    //一个2,代表了数组长度为2，但是里面的数组元素为空
        var arr3 = new Array(2,3)   //等价于 [2,3]  ,里面有两个数组元素 2,3
        console.log(arr1);
        console.log(arr2);
        console.log(arr3);
    </script>
```

检测是否为数组

- `instanceof` 运算符，可以判断一个对象是否属于某种类型
- `Array.isArray()` 用于判断一个对象是否为数组，isArray() 是 HTML5 中提供的方法

```
    <script>
        var arr = [1, 23];
        var obj = {};
        // instanceof 运算符，可以判断一个对象是否属于某种类型
        console.log(arr instanceof Array); // true
        console.log(obj instanceof Array); // false
        // Array.isArray()  用于判断一个对象是否为数组，isArray() 是 HTML5 中提供的方法
        console.log(Array.isArray(arr));   // true
        console.log(Array.isArray(obj));   // false
    </script>
```

例子：通过检测函数输入的值是否为数组，使用 instanceof 或者 Array.isArray() 来判定。

原代码：

```
    <script>
        // 通过翻转数组来举例，如果函数输入的不是一个数组，就不会实现我们想要的数组翻转的效果。
        function reverse(arr) {
            var newArr = [];
            for (var i = arr.length - 1; i >= 0; i--) {
                newArr[newArr.length] = arr[i];
            }
            return newArr;
        }
        var arr1 = reverse([1, 3, 4, 6, 9]);
        console.log(arr1);
        // 通过此例子可以看出，我们需要修改代码，来判定输入的值，是否为一个数组
        var arr2 = reverse('qwe','asd','zxv');
        console.log(arr2);

    </script>
```

添加数组判定 后的代码：

```
    <script>
        function reverse(arr) {
            if(arr instanceof Array){
                var newArr = [];
                for (var i = arr.length - 1; i >= 0; i--) {
                    newArr[newArr.length] = arr[i];
                }
                return newArr;
            }else{
                return '函数参数要求必须为数组格式[1,2,3,4]';
            }
        }
        console.log(reverse(1,2,3,4,5));
    </script>
```

添加删除数组元素

方法名	说明	返回值
push(参数1...)	末尾添加一个或多个元素，注意修改原数组	并返回新的长度
pop()	删除数组最后一个元素，一次只能删除一个元素	返回它删除的元素的值
unshift(参数1...)	向数组的开头添加一个或更多元素，注意修改原数组	并返回新的长度
shift()	删除数组的第一个元素，数组长度减1，无参数，修改原数组	并返回第一个元素

push(参数1...) 末尾添加一个或多个元素，注意修改原数组,并返回新的长度

```
    <script>
        // 1.push() 在我们数组的末尾，添加一个或者多个数组元素 push 推
        var arr = [1, 2, 3];
        arr.push('哈哈哈', '秦晓');
        console.log(arr.push('哈哈哈', '秦晓'));
        console.log(arr);
        console.log(arr.push(4, '秦晓'));
        console.log(arr);
        // push 完毕之后，返回结果是新数组的长度
    </script>
```

1、push() 是可以给数组追加新的元素

2、push() 参数直接 写数组元素

3、push() 完毕之后，返回的结果是 新数组的长度

4、原数组也会发生变化

unshift(参数1...) 向数组的开头添加一个或更多元素，注意修改原数组,并返回新的长度

```
    <script>
        // unshift 在我们数组的开头 添加一个或者多个数组元素
        var arr = [1, 2, 3];
        arr.unshift('blue');
        console.log(arr.unshift('red'));
        console.log(arr);
    </script>
```

1、unshift( ) 是可以给数组前面追加新的元素

2、unshift( ) 参数直接写 数组元素

3、unshift( ) 完毕之后，返回的结果是 新数组的长度

4、原数组也会发生变化

pop() 删除数组最后一个元素，一次只能删除一个元素 返回它删除的元素的值

```
    <script>
        //pop()  删除数组最后一个元素，一次只能删除一个元素  返回它删除的元素的值  ,不加参数
        var arr = [1,2,3,4,5];
        console.log(arr.pop());
    </script>
```

1、pop() 是可以删除数组的最后一个元素，一次只能删除一个元素

2、pop() 没有参数

3、pop() 完毕之后，返回的结果是 删除的那个元素

4、原数组也会发生变化

shift() 删除数组的第一个元素，数组长度减1，无参数，修改原数组， 并返回第一个元素

```
    <script>
        // shift() 它剋删除数组的第一个元素,一次只能删除一个元素
        var arr = [1,2,3,4,5];
        arr.shift(); //不加参数
        console.log(arr);
    </script>
```

1、shift() 是可以删除数组的第一个元素，记住一次只能删除一个元素

2、shift()没有参数

3、shift()完毕之后，返回的结果是，删除的那个元素

4、原数组会发生变化

案例：筛选数组

有一个包含工资的数组[1500,1200,2000,2100,1800],要求把数组中工资超过2000的删除，剩余的放到新数组里面

```
    <script>
        var arr = [1500, 1200, 2000, 2100, 1800];
        var newArr = [];
        for (var i = 0; i < arr.length; i++) {
            if (arr[i] < 2000) {
                newArr[newArr.length] = arr[i];
            }
        }
        console.log(newArr);
    </script>
```

使用push() ,添加新数组

```
    <script>
        var arr = [1500, 1200, 2000, 2100, 1800];
        var newArr = [];
        for (var i = 0; i < arr.length; i++) {
            if (arr[i] < 2000) {
                newArr.push(arr[i]);
            }
        }
        console.log(newArr);
    </script>
```

数组的排序

方法名	说明	是否修改原数组
reverse()	颠倒数组中元素的顺序，无参数	该方法会改变原来的数组，返回新数组
sort()	对数组的元素进行排序	该方法会改变原来的数组，返回新数组

reverse()

```
 // 1、翻转数组
        var arr = ['red','blue','green'];
        arr.reverse();
        console.log(arr);   // ['green', 'blue', 'red']
```

sort()

sort() 对于个位数的使用

```
// 2、数组排序(冒泡排序)
        var arr1 = [3,4,1,2,9];
        arr1.sort();
        console.log(arr1);  //[1, 2, 3, 4, 9]
```

sort()对于双位数的排序，因为sort()，进行排序的判定是从`第一个数字`进行判定的，所以在进行多位数判定时会出错。所以需要用到方法。会使用就行

出错案例：

```
var arr2 = [2,31,23,41,9,0];
arr2.sort();
console.log(arr2);
```

解决方法

```
var arr2 = [2,31,23,41,9,0];
arr2.sort(function(a,b){
    return a - b; //升序的顺序排列
    return b - a; //降序的排列
});
console.log(arr2);
```