1. 内置对象

JavaScript中的对象分为三种：**自定义对象、内置对象、浏览器对象**

前面两种对象是js基础内容，属于ECMAScript;第三个对象属于我们JavaScript独有的 JSAPI时学习

内置对象就是指js语言自带的一些对象，这些对象拱开发者使用，并提供了一些常用的或是最基本的而必要的功能（属性和方法）
内置对象：Math Date Array String等

2. 查文档

2.1. MDN

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

Mozilla开发者网络（MDN）提供了有关开放网络技术（Open Web)的信息，包括HTML、css和万维网及HTML5应用的API

2.2. 如何学习对象中的方法

(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

3. Math对象

3.1. Math三个取整方法

```
// 三个取整方法
        // 向下取整    往小了取
        console.log(Math.floor(-1.1));

        // 向上取整     往大了取
        console.log(Math.ceil(-1.6));

        // 四舍五入     其他数字都是四舍五入，但是.5特殊，它往大了取
        console.log(Math.round(-1.4));    //-1
        console.log(Math.round(-1.5));    //-1
        console.log(Math.round(-2.5));    //-2
        console.log(Math.round(5.5));     //6
        console.log(Math.round(-1.6));    //-2
```

3.1. Math.random()

Math对象随机数方法 random()返回一个随机的小数 0=<x<1
这个方法里面不跟参数

```
// 我们想要得到两个数之间的随机整数，并且包含这两个整数
        // Math.floor(Math.random()*(max-min+1))+min;
        function getRandom(min,max){
            return Math.floor(Math.random()*(max-min+1))+min;
        }
        console.log(getRandom(1,10));

        // 随机点名
        var arr=['张三','李四','王五','孙六','赵七'];
        console.log(arr[getRandom(0,arr.length-1)]);
```

4. 日期对象

4.1. 使用日期对象

Date()日期对象是一个构造函数 必须使用new来调用创建我们的日期对象

```
        var date=new Date();
        console.log(date);
        // 如果没有参数，返回当前系统的当前时间
        // 参数常用的写法 数字型：2019,10,01 或者是字符串型 '2019-10-1 9:8:8'
        var date1=new Date(2021,11,5);
        console.log(date1);    //返回的是十二月 Dec
        var date2=new Date('2021-11-5 11:43:34');
        console.log(date2);
```

4.2. 日期格式化

我们想要得到2019-8-8 8:8:8 格式的日期，该怎么办
需要获取日期指定的部分，所以我们要动手得到这种格式

（图：日期格式化）

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()	获取当前分钟	dObj.getMinutes()
getSeconds()	获取当前秒钟	dObj.getSeconds()

// 格式化日期年月日

```
var date = new Date();
console.log(date.getFullYear()); //返回当前日期的年
console.log(date.getMonth()+1);     //返回的月份小一个月     记得月份加1
console.log(date.getDate());        //返回的是几号
console.log(date.getDay());      //周一返回的是1，周六返回的是6    但是周日返回的是0
//我们写一个2021年11月5日 星期五
var year =date.getFullYear();
var month=date.getMonth()+1;
var dates=date.getDate();
var arr=['星期日','星期一','星期二','星期三','星期四','星期五','星期六'];
var day=date.getDay();
console.log('今天是'+year+'年'+month+'月'+dates+'日'+arr[day]);
```

// 要求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());
```

4.3. 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
我们经常利用总的毫秒数来计算时间，因为它更精确

```
// 获得date的总的毫秒数(时间戳)，不是当前时间的毫秒数，而是距离1970年1月1日过了多少毫秒
    // 通过valueOf()
    // 通过getTime()
    var date=new Date();
    console.log(date.valueOf());
    console.log(date.getTime());

    // 简单的写法(最常用的写法)
    var date1=+new Date();    //返回的就是总的毫秒数
    console.log(date1);

    //H5新增的获得总的毫秒数的方法
    console.log(Date.now());
```

5. 数组对象

5.1. 数组对象的创建

创建数组的两种方式:

- 字面量方式
- new Array()

// 利用字面量创建数组

```
    var arr=[1,2,3];
    console.log(arr[2]);

    // 利用new Array()
    var arr1=new Array();    //创建了一个空的数组
    var arr1=new Array(2);   //这个2表示数组的 长度为2，里面有2个空的数组元素
    var arr1=new Array(1,2); //等价于[1,2] 这样写表示里面有两个数组元素1,2
    console.log(arr1);
```

5.2. 检测是否为数组

```
// 检测是否为数组
        // (1)instanceof    运算符 它可以用来检测是否为数组
        var arr = [];
        var obj = {};
        console.log(arr instanceof Array);
        console.log(obj instanceof Array);

        // (2) Array.isArray(参数);
        console.log(Array.isArray(arr));
        console.log(Array.isArray(obj));
```

5.3. 添加删除数组元素的方法

（图：添加删除数组元素的方法）

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，把数组长度减 1 无参数、修改原数组	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	并返回第一个元素的值

```
// 1.push() 可以在我们数组的末尾添加一个或多个数组元素 push 推
var arr=[1,2,3];
arr.push(4,'海绵宝宝');
console.log(arr);
// 注意：
// push可以给数组追加新的元素
// push 小括号里参数直接写数组元素就可以了
// push完毕后，返回的结果是新数组的长度
// 原数组也会发生变化

// 2. unshift   可以在我们数组的开头 添加一个或多个数组元素
arr.unshift('开头','second');
console.log(arr);
// 注意：
// unshift可以在数组的前面追加新的元素
// unshift()参数直接写数组元素就可以了
// unshift完毕后，返回的结果是新数组的长度
// 原数组也会发生变化

// 3.pop()它可以删除数组的最后一个参数
arr.pop();
console.log(arr.pop());
console.log(arr);
// 注意：
// pop是可以删除数组的最后一个元素，记住一次只能删除一个元素
// pop()是没有参数的
// pop()完毕之后返回的是删除的那个元素
// 原数组也会发生变化

// 4.shift() 它可以删除数组的第一个元素
console.log(arr.shift());
console.log(arr);
// 注意：
// pop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
// pop()是没有参数的
// pop()完毕之后返回的是删除的那个元素
// 原数组也会发生变化
```

5.4. 数组排序

```
// 数组排序
        // 翻转数组
        var arr = [1, 2, 3];
        arr.reverse();
        console.log(arr);

        // 数组排序(冒泡排序)
        var arr1 = [3, 2, 41, 653, 28, 3, 1];
        arr1.sort(function (a, b) {
            // return a - b; //按照升序的顺序排列
            return b - a; //按照降序的顺序排列
        });
        console.log(arr1);
```

5.5. 数组索引方法

(图：数组索引方法)

方法名	说明	返回值
indexOf()	数组中查找给定元素的第一个索引	如果存在返回索引号 如果不存在，则返回-1。
lastIndexOf()	在数组中的最后一个的索引，	如果存在返回索引号 如果不存在，则返回-1。

```
        // 返回数组元素索引号方法：indexOf(数组元素)    作用就是返回该数组元素的索引号
        // 它只返回第一个满足条件的索引号

        // 它如果在该数组里面找不到该元素，则返回的是-1
        var arr=[1,2,34,4,6,'海绵宝宝'];
        console.log(arr.indexOf('海绵宝宝'));
        console.log(arr.indexOf('派大星'));

        // 返回数组元素索引号方法：lastIndexOf(数组元素)    作用就是从后面开始查找 返回该数组元素的索

        console.log(arr.lastIndexOf('海绵宝宝'));
        console.log(arr.lastIndexOf('派大星'));
```

5.6. 数组转化为字符串

(图：数组转换为字符串)

方法名	说明	返回值
toString()	把数组转换成字符串，逗号分隔每一项	返回一个字符串
join('分隔符')	方法用于把数组中的所有元素转换为一个字符串。	返回一个字符串

```
// 数组转换为字符串
        // toString()
        var arr=[1,2,3];
        console.log(arr.toString());   //1,2,3

        // join(分隔符)
        var arr1=['green','blue','yellow','red'];
        console.log(arr1.join());
        console.log(arr1.join('-'));
        console.log(arr1.join('&'));
```

6. 字符串对象

6.1. 基本包装类型

为了方便操作基本数据类型，JavaScript还提供了三个特殊的引用类型：String、Number、Boolean

基本包装类型就是把简单数据类型包装称为复杂数据类型，这样基本数据类型就有了属性和方法

例如：

```
        var str='海绵宝宝';
        console.log(str.length);
        // 对象才有属性和方法 复杂数据类型才有属性和方法
        // 简单数据类型为什么会有length属性呢
        // 基本包装类型：就是把简单数据类型包装成为了复杂数据类型
        // 以上两句代码相当于：
        var temp=new String('海绵宝宝');
        str=temp;
        temp=null;
```

6.2. 字符串的不可变

指的是里面的值不可变，虽然看上去可以改变内容，但其实是地址变了，内存中开辟了一个新的内存空间

6.3. 根据字符返回位置

字符串所有的方法，都不会修改字符串本身(字符串是不可变的)，操作完成后会返回一个新的字符串

（图：根据字符返回位置）

方法名	说明
indexOf('要查找的字符', 开始的位置)	返回指定内容在元字符串中的位置， 如果找不到就返回 -1，开始的位置是 index 索引号
lastIndexOf()	从后往前找，只找第一个匹配的

```
// 根据字符返回位置    str.indexOf('要查找的字符',[起始的位置])
        var str='改革春风吹满地，吹吹牛皮';
        console.log(str.indexOf('吹'));      //4
        console.log(str.indexOf('吹',5));    //8
```

6.4. 根据位置返回字符（重点）

```
// 根据位置返回字符
// (1)charAt(index)根据位置返回字符
var str='海绵宝宝';
console.log(str.charAt(1));

// 遍历所有的字符
for (var i=0;i<str.length;i++){
    console.log(str.charAt(i));
}
// (2)charCodeAt(index)    返回相应索引号的字符ASCII值  目的：判断用户按下了哪个键
console.log(str.charCodeAt(0));

// (3)str[index]  H5新增
console.log(str[0]);
```

6.5. 字符串操作方法（重点）

（图：字符串的操作方法）

方法名	说明
concat(str1,str2,str3...)	concat() 方法用于连接两个或多个字符串。拼接字符串，等效于+，+更常用
substr(start,length)	从start位置开始（索引号） ， length 取的个数 重点记住这个
slice(start, end)	从start位置开始，截取到end位置，end取不到 (他们俩都是索引号)
substring(start, end)	从start位置开始，截取到end位置，end取不到 基本和slice 相同 但是不接受负值

```
// (1)concat('字符串1','字符串2'...)
var str='海绵宝宝';
console.log(str.concat('派大星'));
// (2)substr('截取的起始位置','截取几个字符')
var str1='改革春风吹满面';
console.log(str1.substr(2,2));    //第一个2是索引号的2 从第几个开始 第二个2是取几个字符
```

6.6. 其他方法

```
// (1)替换字符 replace('被替换的字符','替换为的字符')    它只会替换第一个字符
var str='海绵宝宝派大星';
console.log(str.replace('派大星','章鱼哥'));

// 有一个字符串'heigognrogirtgnhgduhgiorgyhruhnvgaofjndioghgoiahigyhohgaojjgfoi'，要求吧
var str1='heigognrogirtgnhgduhgiorgyhruhnvgaofjndioghgoiahigyhohgaojjgfoi';
while (str1.indexOf('g')!==-1){
    str1=str1.replace('g','*')
}
console.log(str1);

// (2)字符替换为数组 split('分隔符')    前面我们学过join把数组转换为字符串
var str2='red,blue,green,yellow,pink';
console.log(str2.split(','));
```

7. 简单类型和复杂类型

简单类型又叫做基本数据类型或者值类型，复杂类型又叫做引用类型

值类型：简单数据类型/基本数据类型，在存储变量中存储的是值本身，因此叫做值类型

string number boolean undefined null

引用类型：复杂数据类型，在存储时变量中存储的仅仅是地址（引用），因此叫做引用数据类型，通过new关键字创建的对象（系统对象、自定义对象），如Object Array Date等

8. 堆和栈

堆栈空间分配区别：
栈（操作系统）：由操作系统自动分配释放存放函数的参数值，局部变量的值等。其操作方式类似于数据结构中的栈；
简单数据类型存放到栈里面

堆(操作系统)：存储复杂类型（对象），一般由程序员分配释放，若程序员不释放，由垃圾回收机制回收。
复杂数据类型存放到堆里面

注意：JavaScript中没有堆栈的概念，通过堆栈的方式可以更容易理解代码的执行方式

9. 简单类型的内存分配

值类型：（简单数据类型）string number boolean undefined null

值类型变量的数据直接存放在变量（栈空间）中 里面直接开辟一个空间，存放的是值

10. 简单类型传参

函数的形参也可以看做是一个变量，当我们把一个值类型变量作为参数传递给函数的形参时，其实是把变量在栈里面的值复制了一份给形参，那么在方法内部对形参做任何改变，都不会影响到外部变量

```
function fn(a){
        a++;
        console.log(a);
}
var x=120;
fn(x);
console.log(x)
//11 10
```

11. 复杂类型传参

```
// 复杂数据类型传参
        function Person(name){      //x=p
            this.name=name;         //2.这个输出什么    刘德华
        }
        function f1(x){
            console.log(x.name);    //3.这个输出什么    张学友
            x.name='张学友';
            console.log(x.name);
        }
        var p=new Person('刘德华');
        console.log(p.name);        //1.这个输出什么    刘德华
        f1(p);
        console.log(p.name);        //4.这个输出什么    张学友
```

函数的形参也可以看做是一个变量，当我们把引用类型变量传递给形参时，其实是把变量在栈空间里保存的堆地址复制给了形参，形参和实参实际上保存的是同一个堆地址，所以操作的是同一个对象