数组、字符串

一. 目标

- 能够判断一个变量是否是数组
- 能够使用数组的pop/push/shift/unshift方法
- 能够模拟数组的sort方法的实现
- 能够使用数组的splice/indexOf等方法
- 能够清空数组
- 理解什么是基本包装类型
- 能够转换字符串中字母的大小写
- 能够截取字符串的指定部分
- 能够替换和截取字符串

二. 数组

2.1 数组的创建方式【重要】

- 创建方式
  - 方式一：数组字面量

```
//语法：
var 数组名 = [数据,数据,数据];  //数组字面量
//代码：
var names = ['张三','李四','王五','赵六'];
```

  - 方式二：Array类型

```
//语法：
var 数组名 = new Array(数据,数据,数据);
//代码：
var names = new Array('张三','李四','王五','赵六');
```

- 数组是一个特殊的对象
  - 数组的类型：Array

```
var arr = [];
console.log(arr instanceof Array); //true
```

  - 数组特殊在哪里？

```
数组中的元素是有序的（元素的下标从0开始）
```

2.2 数组常用的方法 【重要】

- toString()、valueOf()
  - 数组名.toString();

```
/*
    功能：数组转换为字符串，逗号分隔每一项
    参数：无
    返回值：字符串 string
    不会改变数组本身
*/
var arr = ['张三','李四','王五'];
var result = arr.toString();
console.log(result);//张三,李四,王五
```

  - 数组名.valueOf();

```
/*
    功能：返回数组对象本身(原始值)
    参数：无
    返回值：数组 Array
    不会改变数组本身
*/
var arr = ['张三','李四','王五'];
var result = arr.valueOf();
console.log(result);// ["张三", "李四", "王五"]
```

- push()、pop();
  - 数组名.push(数据,数据,数据...);

```
/*
    功能：向数组的尾部添加一个或多个数据
    参数：任意数据
    返回值：数组改变后的长度 number
    会改变数组本身
*/
var arr = ['张三','李四','王五'];
console.log(arr);   //改变前  ['张三','李四','王五'];
var len = arr.push('赵六','陈七');
console.log(len);   //长度  5
console.log(arr);   //改变后  ["张三", "李四", "王五", "赵六", "陈七"]
```

  - 数组名.pop();

```
/*
    功能：删除数组中的最后一个数据
    参数：无
    返回值：删除的最后一个数据
    会改变数组本身
*/
var arr = ['张三','李四','王五'];
console.log(arr);   //改变前  ['张三','李四','王五'];
var data = arr.pop();
console.log(data);  //王五
console.log(arr);   //改变后  ["张三", "李四"]
```

- unshift()、shift();
  - 数组名.unshift(数据,数据,数据...);

```
/*
    功能：向数组的首部部添加一个或多个数据
    参数：任意数据
    返回值：数组改变后的长度 number
    会改变数组本身
*/
var arr = ['张三','李四','王五'];
console.log(arr);   //改变前  ['张三','李四','王五'];
var len = arr.unshift('赵六','陈七');
console.log(len);   //长度  5
console.log(arr);   //改变后   ["赵六", "陈七", "张三", "李四", "王五"]
```

  - 数组名.shift();

```
/*
    功能：删除数组中的最前面的一个数据
    参数：无
    返回值：删除的最前面的一个数据
    会改变数组本身
*/
var arr = ['张三','李四','王五'];
console.log(arr);   //改变前  ['张三','李四','王五'];
var data = arr.shift();
console.log(data);  //张三
console.log(arr);   //改变后  ['李四','王五'];
```

- reverse()、sort()
  - 数组名.reverse();

```
/*
    功能：反转
    参数：无
    返回值：返回排序好的数组
    会改变数组本身
*/
var arr = ['张三','李四','王五'];
console.log(arr);   //改变前  ['张三','李四','王五'];
arr.reverse();
console.log(arr);   //改变后  ["王五", "李四", "张三"]
```

  - 数组名.sort(compareFunction)

```
/*
    功能：排序
    参数：compareFunction 回调函数，控制排序的规则。可有可无。默认按照Unicode编码排序
    返回值：返回排序好的数组
    会改变数组本身
*/
//排序方式一：默认
var arr = [11,22,44,111,222];
arr.sort();  //默认排序
console.log(arr);   //[11, 111, 22, 222, 44]  按照Unicode编码排序

//排序方式二：升序（从小到到）
var arr = [11,22,44,111,222];
arr.sort(function(a,b){
  return a-b;
});
console.log(arr);   // [11, 22, 44, 111, 222] 从小到大

//排序方式三：降序（从大到小）
var arr = [11,22,44,111,222];
arr.sort(function(a,b){
  return b-a;
});
console.log(arr);   // [222, 111, 44, 22, 11] 从大到小
```

- concat()、slice()、splice()
  - 数组名.concat(其他数组);

```
/*
    功能：联合其他数组
    参数：其他数组 Array 必填
    返回值：返回一个新的组合好的数组  Array
    不会改变数组本身
*/
var arr1 = [22,33];
var arr2 = [44,55];
var result = arr1.concat(arr2);
console.log(result); //[22, 33, 44, 55]
```

- 数组名.slice(startIndex,endIndex);

```
/*
    功能：截取
    参数：
        startIndex 起始位置   number 必填
        endIndex   终止位置   number  可有可无
    返回值：返回一个新的数组  Array
    不会改变数组本身
*/
var arr = [22,33,44,55,66,77];
var result1 = arr.slice(1);     //[1,arr.length-1]
var result2 = arr.slice(2,4);  //[2,4);
console.log(result1); //[33, 44, 55, 66, 77]
console.log(result2);  //[44,55]
```

- 数组名.splice(startIndex,count,数据,数据,数据...);

```
/*
    功能：增、删、修改（替换）
    参数：
        startIndex 起始位置    number 必填
        count      删几个      number  必填
        数据,数据,数据... 新的数据    任意类型    可有可无
    返回值：增、删、修改（替换）后的数组
    会改变数组本身
*/
//删除
var arr = [22,33,44,55,66,77];
arr.splice(1,1);
console.log(arr); //[22, 44, 55, 66, 77]

//增加
var arr = [22,33,44,55,66,77];
arr.splice(1,0,'三三','小三');
console.log(arr); //[22, "三三", "小三", 33, 44, 55, 66, 77]

//修改
var arr = [22,33,44,55,66,77];
arr.splice(1,1,'三三');
console.log(arr); // [22, "三三", 44, 55, 66, 77]
```

- indexOf()、lastIndexOf()
    - 数组名.indexOf(数据)、数组名.lastIndexOf(数据);;

```
/*
    功能：根据数据找数据在数组中位置（从前往后找 与从后向前找）
    参数：
        数据，任意类型，必填
    返回值：数据的下标
    不会改变数组本身
*/
var arr = [22,33,44,55,22,66];
var index1 = arr.indexOf(22);
console.log(index1);  //0
var index2 = arr.lastIndexOf(22);
console.log(index2);  //4
```

- every()、filter()、forEach()、map()、some()

  > 扩展，查手册

- join()

  - 数组名.join(separator);

```
/*
    功能：用指定的符合拼接数组中的每一个数据
    参数：
        separator，字符串，可选
    返回值：字符串  string
    不会会改变数组本身
*/
var arr = [22,33,44];
var result1 = arr.join();
var result2 = arr.join('|');
console.log(result1);  //22,33,44
console.log(result2);  //22|33|44
```

2.3 清空数组的方式【重要】

```
var arr = [22,33,44,55];
// 方式1 推荐
arr = [];
// 方式2
arr.length = 0;
// 方式3
arr.splice(0, arr.length);
```

三. 字符串

3.1 字符串的不可变性【了解】

```
var str = 'abc';
str = 'hello';
// 当重新给str赋值的时候，常量'abc'不会被修改，依然在内存中
// 重新给字符串赋值，会重新在内存中开辟空间，这个特点就是字符串的不可变
// 由于字符串的不可变，在大量拼接字符串的时候会有效率问题
```

3.2 字符串包装对象【重要】

```
// 普通字符串
var str = 'abc';   //普通字符串不是对象
var len = str.length; //但是为什么可以像对象一样使用点出东西？
//答：str在点的时候，在内存中会创建一个字符串包装对象并把引用赋值给str,等str调用完length之后，这个在内存中创建的临时对象会断开与str之间的引用并从内存中删除。str重新指向普通字符串'abc'。
alert(len); //3

//把字符串包装成对象 → 基本包装类型
var strObj = new String('abc');  //把字符串包装成对象
var len = strObj.length;  //因为是对象，所以可以点出东西。
alert(len); //3
```

3.3 字符串常用的方法 【重要】

> 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)，操作完成会返回一个新的字符串
> 以下方法，可以自己尝试查手册或文档使用

```
// 1 字符方法
charAt()          //获取指定位置处字符
charCodeAt()      //获取指定位置处字符的ASCII码
str[0]            //HTML5，IE8+支持 和charAt()等效  重点
// 2 字符串操作方法
concat()          //拼接字符串，等效于+，+更常用
slice()           //从start位置开始，截取到end位置，end取不到 重点
substring()       //从start位置开始，截取到end位置，end取不到
substr()          //从start位置开始，截取length个字符
// 3 位置方法
indexOf()         //返回指定内容在元字符串中的位置  重点
lastIndexOf()     //从后往前找，只找第一个匹配的
// 4 去除空白
trim()            //只能去除字符串前后的空白
// 5 大小写转换方法
toUpperCase()     //转换大写 重点
toLowerCase()     //转换小写 重点
// 6 其它
search()
replace()  重点
split(s)   重点
fromCharCode()
// String.fromCharCode(101, 102, 103);   //把ASCII码转换成字符串
```