1. slice() 方法返回一个新的数组对象，这一对象是一个由 begin 和 end 决定的原数组的浅拷贝（包括 begin，不包括end）。原始数组不会被改变。

a) begin包含当前元素(索引)，end不包含当前元素(索引)

b) 索引顺序，正数是 0 1 2… 负数是 Array.length-1 Array.length-2…

3. splice() 方法通过删除或替换现有元素或者原地添加新的元素来修改数组,并以数组形式返回被修改的内容。此方法会改变原数组。

a) start(索引)、deleteCount(删除个数)、item(可以是多个)

b) deleteCount不填写代表删除索引后面所有元素

c) 索引顺序，正数是 0 1 2… 负数是 Array.length-1 Array.length-2…

4.  concat() 方法用于合并两个或多个数组。此方法不会更改现有数组，而是返回一个新数组。

5. forEach() 方法对数组的每个元素执行一次给定的函数

总结：

1. forEach 其实是一个迭代器，他与 for 循环本质上的区别是 forEach 是负责遍历（Array Set Map）可迭代对象的，而 for 循环是一种循环机制，只是能通过它遍历出数组

2. for需要先知道数组长度再操作，foreach不需要，foreach不能改变在遍历过程中对数组或集合进行修改

3. foreach效率比for高很多，主要原因是for要进行很多次条件判断，另外从数组变量的数据结构来看foreach直接通过结构体中next指针获取下一个值，而for循环需要根据key先进行一次hash才得到值

4. 在固定长度或长度不需要计算的时候for循环效率高于foreach；在不确定长度,或计算长度有性能损耗的时候,用foreach比较方便.并且foreach的时候会锁定集合中的对象.期间不能修改.

    const checkDogs = function (dogsJulia, dogsKate) {

        const dogsJuliaCorrected = dogsJulia.slice();

        dogsJuliaCorrected.splice(0, 1);

        // 未定义删除位数，将删除索引后面所有元素

        dogsJuliaCorrected.splice(-2);

        const dogs = dogsJuliaCorrected.concat(dogsKate);

        console.log(dogs);

        dogs.forEach(function (dog, i) {

            if (dog >= 3) {

                console.log(`Dog number ${i + 1} is an adult, and is ${dog} years old`);

            } else {

                console.log(`Dog number ${i + 1} is still a puppy 🐶`);

            }

        });

    }

    checkDogs([3, 5, 2, 12, 7], [4, 1, 15, 8, 3]);

    checkDogs([9, 16, 6, 8, 3], [10, 5, 6, 1, 4]);

1. Object.entries()方法返回一个给定对象自身可枚举属性的键值对数组，其排列与使用for…in循环遍历该对象时返回的顺序一样(区别在于for-in循环还会枚举原型链中的属性)

2. Object.values()方法返回一个给定对象自身的所有可枚举属性值的数组，值的顺序与使用for...in循环的顺序相同 ( 区别在于 for-in 循环枚举原型链中的属性 )

3. for...of语句在可迭代对象（包括 Array，Map，Set，String，TypedArray，arguments 对象等等）上创建一个迭代循环，调用自定义迭代钩子，并为每个不同属性的值执行语句

总结：

1. 通过Object.entries()方法返回一个带有对象键和值的数组，我们可以解构i和el分别代表对象键和值，因此我们可以console每个玩家的姓名连同进球号码

2. Obejct.value()方法返回一个带有属性值的数组，因此我们循环此数组用于求值

3. for of 语句可迭代数组，因此我们通过此方法求出平均值

    /\*

    让我们继续使用我们的足球投注应用程序！

    1. 循环 game.scored 数组并将每个玩家的姓名连同进球号码一起打印到控制台（例如：“Goal 1: Lewandowski”）

    2. 使用循环计算平均odds并将其记录到控制台（我们已经研究过如何计算平均数，如果您不记得可以去检查）

    3. 将 3 个赔率打印到控制台，但格式很好，就像这样：

        拜仁慕尼黑获胜几率：1.33

        平局几率：3.25

        多特蒙德胜利几率：6.5

    直接从游戏对象中获取团队名称，不要对其进行硬编码（“draw”除外）。 提示：注意赔率和游戏对象如何具有相同的属性名称😉

    奖励：创建一个名为“scorers”的对象，其中包含作为属性得分的球员的姓名，以及作为值的进球数。 在这个游戏中，它看起来像这样：

        {

            Gnarby: 1,

            Hummels: 1,

            Lewandowski: 2

        }

    祝您好运 😀

    \*/

    const game = {

    team1: 'Bayern Munich',

    team2: 'Borrussia Dortmund',

    players: [

        [

        'Neuer',

        'Pavard',

        'Martinez',

        'Alaba',

        'Davies',

        'Kimmich',

        'Goretzka',

        'Coman',

        'Muller',

        'Gnarby',

        'Lewandowski',

        ],

        [

        'Burki',

        'Schulz',

        'Hummels',

        'Akanji',

        'Hakimi',

        'Weigl',

        'Witsel',

        'Hazard',

        'Brandt',

        'Sancho',

        'Gotze',

        ],

    ],

    score: '4:0',

    scored: ['Lewandowski', 'Gnarby', 'Lewandowski', 'Hummels'],

    date: 'Nov 9th, 2037',

    odds: {

        team1: 1.33,

        x: 3.25,

        team2: 6.5,

    },

    };

    // 1.

    for (const [i, players] of game.scored.entries())

    console.log(`Goal ${i + 1}: ${players}`);

    // 2.

    const odds = Object.values(game.odds);

    let average = 0;

    for (const odd of odds) average += odd;

    average /= odds.length;

    console.log(average);

    // Odd of victory Bayern Munich: 1.33

    // Odd of draw: 3.25

    // Odd of victory Borrussia Dortmund: 6.5

    // 奖金

    // 因此，解决方案是循环遍历数组，将数组元素作为对象属性添加，然后在遇到某个元素的新实例时增加计数

    const scorers = {};

    for (const player of game.scored) {

        scorers[player] ? scorers[player]++ : (scorers[player] = 1);

    }

    console.log(scorers);