数组去重

数据结构 Set，类似于数组，但是成员的值都是唯一的，没有重复的值。可以用于数组去重：

```
[...new Set([1, 2, 3, 2])]; // [1, 2, 3]
```

```
Array.from(new Set([1, 2, 3, 2])); // [1, 2, 3]
```

也可以使用传统方法：

```
function unique(arr) {
  let res = [];
  arr.forEach(item => {
    if (!res.includes(item)) {
      res.push(item);
    }
  })

  return res;
}
let arr = [1, 2, 3, 2];
console.log(unique(arr)); // [1, 2, 3]
```

时间复杂度比较

```
// 生成一个成员数很多的并且有重复的数组
let arr = [];
for (let i = 0; i < 50 * 10000; i++) {
  arr.push(i);
}
for (let i = 50 * 10000; i > 0; i--) {
  arr.push(i);
}

function unique(arr) {
  let res = [];
  arr.forEach(item => {
    if (!res.includes(item)) {
      res.push(item);
    }
  })
  return res;
}
```

```
function unique2(arr) {
  let res = [];
  for (let i = 0, len = arr.length; i < len; i++) {
    if (!res.includes(arr[i])) {
      res.push(arr[i]);
    }
  }
  return res;
}

console.time("Set 结合 ...");
[...new Set(arr)]; // Set 结合 ...: 33.26708984375 ms
console.timeEnd("Set 结合 ...");

console.time("Set 结合 Array.from");
Array.from(new Set(arr));// Set 结合 Array.from: 33.0068359375 ms
console.timeEnd("Set 结合 Array.from");

console.time("传统方法结合 forEach 循环");
unique(arr); // 传统方法结合 forEach 循环: 19628.690185546875 ms
console.timeEnd("传统方法结合 forEach 循环");

console.time("传统方法结合 for 循环");
unique2(arr); // 传统方法结合 for 循环: 19563.662109375 ms
console.timeEnd("传统方法结合 for 循环");
```

for 循环比 forEach 循环的时间复杂度稍小，因为 forEach 循环每一次都需要创建一个新的函数，创建函数需要时间也需要占用内存。

Set 结合 `...` 以及 Set 结合 `Array.from` 都比传统方法时间复杂度低很多。