新的变量声明方式

es5中如果有两个相同的变量不会报错，会有变量冲突在循环中使用var来定义变量会使变量提升

```
console.log(on);//变量会提升 打印未定义
var on=0
```

let声明的变量不允许重复声明防止变量冲突

let的块级作用域

只要有大括号就会产生一个块级作用域，所以let的作用域是独立的不会影响到其他地方

```
for (let index = 0; index < array.length; index++) {
   let a=0

}
console.log(a);//在循环外面获取不到a
```

同样也适用于分支语句

```
if (true) {
    let num =1
}
console.log(num);//分支语句同样适用
```

let没有变量提升 要求书写时要先定义变量再使用变量

```
console.log(on); 不会提升直接报错
let on=0
```

const定义常量 简单来说就是不能改变的量 拥有let所有的特点 但是值不能改变

规范：所有字母为大写可以结合下下划线来区分不同的单词

```
const ADD_CART=''
```

对象是指向的的地址，修改属性不会去修改地址

```
const ADD_CART={
    name:'no',
    age:18
}
ADD_CART.name='jack' //对象是指向的的地址，修改属性不会去修改地址
```

重新赋值会改变地址 不合法会报错 /报错

```
ADD_CART={
    username:'ddd'//重新赋值会改变地址 不合法会报错
}
```

默认参数

函数的新参位置直接赋值 （如果有实参传入就以实参为基准，如果没有，就以直接赋值的数为默认参数）

es5中去判断

```
function suhai(user){
            user=user||'世界'
            console.log(`你好${user}`);
        }
        suhai('')
```

直接使用默认参数

```
function suhai(user='sh'){      //会成为默认参数

            console.log(`你好${user}`);
        }
        suhai()
```

箭头函数

再es6中可以简化函数的定义方式

再es5中会这样定义

```
     let map=  function(user,use2){
            return user +user2
        }

        map('s','j')
```

如果用es6

用箭头函数

```
//改成箭头函数
        let mapw=(user1,user2)=>{
            return user1+user2
        }
```

//只有一个值 可以省略括号//一个值都没有必须写上括号

```
let kks=data=>{
            return `${data}`
        }
```

如果只有return 可以省略大括号和return

```
let dahh=num=> 'dahh'
```

箭头函数不能当作构造函数使用

箭头函数不存在arguments

this指向问题

箭头函数外面指向谁箭头函数就指向谁