**名词解释**

任何一对花括号{ }中的语句都属于一个块，在花括号里面用let定义的所有变量在花括号外都是不可见的，我们称之为**块级作用域**

从数组和对象中提取值，对变量进行赋值，这被称为**解构**

解构不成功，变量的值就等于**undefined**

解构赋值给我们一种新的变量赋值方式，主要可以利用数组解构赋值和对象解构赋值。它的用途包括：交换变量值，提取函数返回值，函数参数定义，默认值设定等等。

**Let**

声明变量只在块级作用域起作用，适合在for循环使用，也不会出现变量提升现象。同一个代码块内，不可重复声明相同变量，不可重复声明函数内的参数。

**Const**

也是用于声明一个常量，并必须赋值，声明后不可修改，跟let一样，只在块级作用域起作用，不可重复声明同一个变量，不会变量提升，声明引用类型的常量时，要注意是传址赋值

**this**

永远指向函数运行时所在的对象，而不是函数创建时所在的对象()=>{}箭头函数，函数体内的this对象，就是定义时所在的对象，而不是使用时所在的对象匿名函数和不处于任何对象中的函数，This指向windowcall, apply, with指的This是谁就是谁。普通函数调用，函数被谁调用，This就指向谁

**call & apply**

可以用来代替另一个对象调用一个方法, 可将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象,如果没有提供 thisObj 参数，那么 Global 对象被用作 thisObj。**都可以改变this指向**.

**语法： apply(thisObj，[argArray])**

**语法： call(thisObj，arr1，arr2,arr3,…)**接收两个参数：一个是在其中运行函数的作用域，另一个是参数数组。

**例子：**

Math.max,call(this,1,2,3,4) // this指代Math

**javaScript链式调用原理**

就是jQuery节点在调用api后都会返回节点自身

实际上链式操作仅仅是通过对象上的方法最后  return this 把对象再返回回来，对象当然可以继续调用方法啦，所以就可以链式操作

优点：

节省代码量，代码看起来更优雅

为了更好的异步体验

**Promise**

Promise是一个让开发者更合理、更规范地用于处理异步操作的对象，它有三种状态：初始化、操作成功、操作异常。使用实例方法：then( ) 和  catch( ) 来绑定处理程序；还提供了类方法：Promise.all( ) 和 Promise.race( )

之所以能够使用链式调用，是因为then方法和catch方法调用后，都会返回promise对象。

**getPrototypeOf** 获取一个对象的prototype属性

**setPrototypeOf** 设置一个对象的prototype属性

// 首先定义一个Animal类(函数)  
function Animal() {}  
  
//函数都有一个预属性protoype对象  
Animal.**prototype** = {  
 //给prototype对象添加一个showName属性  
 **showName**: 'sdf',  
 say() {  
 **console**.log('hello')  
 }  
}  
//实例化Animal对象，赋值给变量a  
var a = new Animal()  
//调用类的say方法  
a.say(); // 打印hello  
  
// 获取a对象的prototype属性  
Object.getPrototypeOf(a) // {showName:'sdf...}  
// 设置一个对象的prototype属性  
Object.**setPrototypeOf**(a, {**showName**: '修改了'})

面向对象编程有几个关键步骤：

1、构造函数；

2、给prototype对象添加属性和方法；

3、实例化；

4、通过实例化后的对象调用类的方法或者属性。

**rest**

...values（三个点+变量名）；其次，values是一个数组rest参数必须是函数的最后一个参数，后面不能再跟其他参数结合数组使用，把数组的元素用逗号分隔开来，组成一个序列例如 var arr=[2,3] function wf(a,b){} wf(...arr)

**箭头函数**

a => a

第一个a代表传进去的参数；后面的a表示函数体

(a,b)=>{return a+b}

如果参数超过1个需要用()括起来

函数体语句超过1条需要用{}括起来，并使用return语句返回

**For…of**

1.写法比for循环简洁很多；2.可以用break来终止整个循环，或者continute来跳出当前循环，继续后面的循环；3.结合keys( )获取到循环的索引，并且是数字类型，而不是字符串类型。而且for...in更适合用于对象的遍历