1. 内存泄漏

①正确使用闭包可避免内存泄漏

②var的变量可能存在全局污染，造成内存泄漏，而let是块级作用域，不会造成泄漏

1. 箭头函数中This指代

①普通函数下的this:

* 在普通函数中的this总是代表它的直接调用者，在默认情况下，this指的是window，
* 在严格模式下,没有直接调用者的函数中的this是 undefined使用
* call,apply,bind(ES5新增)绑定的,this指的是 绑定的对象

②箭头函数中的this:

* 箭头函数没有自己的this, 它的this是继承而来; 默认指向在定义它时所处的对象(宿主对象),
* 而不是执行时的对象, 定义它的时候,可能环境是window,也有可能是其他的。

1. 作用域链

**①变量作用域**

在JavaScript中全局变量的作用域比较简单，它的作用域是全局的，在代码的任何地方都是有定义的。然而函数的参数和局部变量只在函数体内有定义。另外局部变量的优先级要高于同名的全局变量,也就是说当局部变量与全局变量重名时，局部变量会覆盖全局变量。

**②函数作用域**

在JavaScript中变量的作用域，并非和C、Java等编程语言似得，在变量声明的代码段之外是不可见的，我们通常称为块级作用域，然而在JavaScript中使用的是函数作用域（变量在声明它们的函数体以及这个函数体嵌套的任意函数体都是有定义的）

③**作用域链**

在JavaScript中，函数也是对象，实际上，JavaScript里一切都是对象。函数对象和其它对象一样，拥有可以通过代码访问的属性和一系列仅供JavaScript引擎访问的内部属性。其中一个内部属性是[[Scope]]，该内部属性包含了函数被创建的作用域中对象的集合，这个集合被称为函数的作用域链，它决定了哪些数据能被函数访问。

当一个函数创建后，它实际上保存一个作用域链，并且作用域链会被创建此函数的作用域中可访问的数据对象填充。

1. 原型链

**①什么是原型链？**

简单理解就是原型组成的链，对象的\_\_proto\_\_它的是原型，而原型也是一个对象，也有\_\_proto\_\_属性，原型的\_\_proto\_\_又是原型的原型，就这样可以一直通过\_\_proto\_\_想上找，这就是原型链，当向上找找到Object的原型的时候，这条原型链就算到头了。

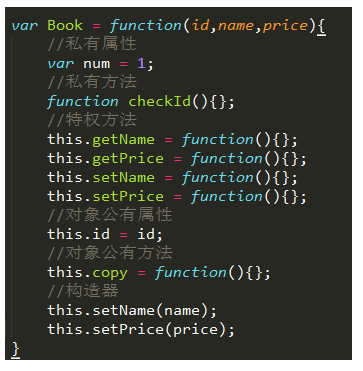
**②原型对象和实例之间有什么作用呢？**

通过一个构造函数创建出来的多个实例，如果都要添加一个方法，给每个实例去添加并不是一个明智的选择。这时就该用上原型了。

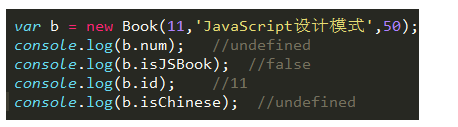
在实例的原型上添加一个方法，这个原型的所有实例便都有了这个方法。

1. 封装

①封装通俗的说，就是我有一些秘密不想让人知道，就通过私有化变量和私有化方法，这样外界就访问不到了。然后如果你有一些很想让大家知道的东西，你就可以通过this创建的属性看作是对象共有属性和对象共有方法，这样别人知道你的公共的东西，不止如此，你还可以访问到 类或对象自身的私有属性和私有方法。外面的公共的方法和属性，和内部的私有属性和方法都可以访问到，都有特权，因此就叫做特权方法。



如上图所示，封装了一个函数，函数内部存在局部变量和局部，外部无法直接调用

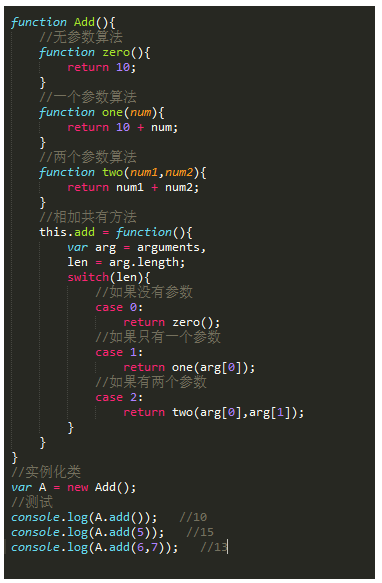


在调用了封装函数之后，

1. 多态

多态就是通过对传递的参数判断来执行逻辑，即可实现一种多态处理机制

      下面就是这个例子，通过多态类，调用add运算方式，根据不同参数做运算



1. 移动端屏幕适配（移动端必考）
2. 在页面中引入flexible.js，flexible会在<html>标签上增加一个data-dpr属性和font-size样式，达到屏幕自适应的效果
3. 把视觉稿中的px转换成rem
4. 字体不使用rem的方法
5. flexible.js会先去获取页面上[name="viewport"]或[name="flexible"]的meta标签，如果有就直接根据获取到的值来判断，如果没有，会自己创建一个meta标签
6. 有了<meta>标签之后，就可以动态改写data-dpr和font-size两个属性的值，因此也就达到了适配的效果。