# 单例模式

这个是我学习的第一个设计模式。

单例模式的定义是： 保证一个类有且仅有一个实例，并提供一个访问它的全局访问点。

基于上面的解释，我们发现全局变量都是符合“单例模式”的。 ## 应用场景

其实只要你有需求是全局出现一次的都可以用，比如全局的蒙层，你不希望页面同时出现两个蒙层，那么很显然你可以使用单例模式。 类似的例子还有全局缓存，线程池等。 ## 实现 下面是一个最简单的单例模式的实现：

function Singleton() {  
 // 你的业务逻辑  
}  
Singleton.getInstance = function(...args) {  
 if (!Singleton.instance) {  
 Singleton.instance = new Singleton();  
 }  
 return Singleton.instance;  
};  
Singleton.instance = null;  
  
// test  
const a = Singleton.getInstance();  
const b = Singleton.getInstance();  
  
console.log(a === b); // true

注意上面的例子是用Singleton.getInstance来实例化的。

我们继续改造一下代码，用闭包来实现，这样我们可以使用new来实例化：

const Singleton = (function () {  
 let instance = null;  
  
 return function () {  
 if (instance) {  
 return instance;  
 }  
 // 你的业务逻辑  
  
 return instance = this;  
 }  
})();  
  
// test  
const a = new Singleton();  
const b = new Singleton();  
  
console.log(a === b); // true

## 扩展

* 如果我想写一个高阶函数，高阶函数接受一个函数为参数，返回一个新的函数，这个新的函数和原函数功能一样，但是多了单例的效果，如何实现？