# js设计模式

## 单例模式（Singleton）

专业解释：

单例模式是一种确保在任何情况下一个类仅有一个实例，并提供全局访问点的设计模式。它主要用于控制对全局唯一资源的访问。

通俗理解：

就好比一座城市里只有一座供水站，不论你需要从哪里取水，都只能通过这个供水站来获取，而且全市人民共用的是同一座供水站。

> 优点：

节约系统资源，对于那些需要频繁创建和销毁的对象，单例模式可以提高系统性能。

提供了全局访问点，对象间通信更简单。

> 缺点：

不适用于多线程环境，可能会引起线程安全问题。

单例对象的状态，对于依赖它的对象可能不易处理。

单例模式在系统中引入全局状态，增加了测试的复杂性。

// 1.使用一个变量来记录单例实例是否已经被创建。

let singleton;

const getSingleton = (function () {

  return function (Constructor, ...args) {

    if (!singleton) {

      singleton = new Constructor(...args);

    }

    return singleton;

  };

})();

// 2.使用闭包来创建私有变量。

const Singleton1 = (function () {

  let instance = null;

  class Singleton {

    constructor() {

      if (instance) {

        return instance;

      }

      instance = this;

    }

  }

  return Singleton;

})();

// 3.使用Object.defineProperty来创建具有私有变量的单例。

class Singleton2 {

  static instance;

  constructor() {

    if (Singleton2.instance) {

      return Singleton2.instance;

    }

    Singleton2.instance = this;

  }

}

## 抽象工厂模式（Abstract Factory）

专业解释：

抽象工厂模式提供一个接口用于创建一系列相关或相互依赖的对象，而无需指定具体类。客户端使用此接口选择所需的产品族中的产品对象。

通俗理解：

设想一个汽车工厂不仅能生产各种类型的车（如轿车、SUV等），还能生产配套的轮胎和内饰。客户只要告诉工厂要哪种类型的车，工厂就会相应地提供整套适合的汽车部件。