# **JavaScript 设计模式（策略模式，桥接模式）**

****1. 策略模式****

**定义一系列的算法，把每一个算法封装起来，并且使它们可相互替换。**

举个简单的例子吧。  
例如组团去郊游，然后去到动物园，进入动物园需要买票  
对于小朋友春游来说，校巴进入动物园，买票的方式是，老师统一收钱，然后交给收费站。  
对于其他做巴士的人来说，买票的方式是，就由工作人员上车一个一个收钱。

var Bus = function() {

this.fee = function() {

throw new Error('此处由子类实现方法');

};

};

var MiniBus = function() {};

MiniBus.prototype = new Bus();

MiniBus.prototype.fee = function() {

console.log('小巴在付款！收费人员上车逐个收费');

};

var SchoolBus = function() {};

SchoolBus.prototype = new Bus();

SchoolBus.prototype.fee = function() {

console.log('校巴在付款！老师收钱，交给收费站');

};

var TollStation = function() {

this.bus = null;

this.setBus = function(b) {

this.bus = b;

};

this.fee = function () {

this.bus.fee();

};

};

// 让校巴进入收费站

var tollStation1 = new TollStation();

var schoolBus = new SchoolBus();

tollStation1.setBus(schoolBus);

tollStation1.fee();

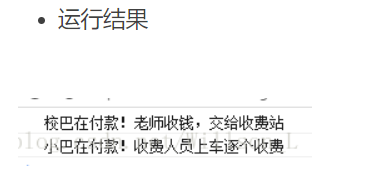
// 让小巴进入收费站

var tollStation2 = new TollStation();

var miniBus = new MiniBus();

tollStation2.setBus(miniBus);

tollStation2.fee();



****2. 桥接模式****

将抽象部分与它的实现部分分离，使它们都可以独立地变化。它是一种对象结构型模式

var Bus = function() {

this.fee = function() {

throw new Error('此处由子类实现方法');

};

};

var MiniBus = function() {};

MiniBus.prototype = new Bus();

MiniBus.prototype.fee = function() {

console.log('小巴在付款！收费人员上车逐个收费');

};

var SchoolBus = function() {};

SchoolBus.prototype = new Bus();

SchoolBus.prototype.fee = function() {

console.log('校巴在付款！老师收钱，交给收费站');

};

var TollStation = function() {

this.bus = null;

this.name = '';

this.money = '';

this.setBus = function(b) {

this.bus = b;

};

this.fee = function () {

console.log('巴士进入' + this.name + '收费站，每人' + this.money + '元');

this.bus.fee();

};

};

var ZooTollStation = function() {};

ZooTollStation.prototype = new TollStation();

ZooTollStation.prototype.money = 10;

ZooTollStation.prototype.name = '动物园';

var PlantTollStation = function() {};

PlantTollStation.prototype = new TollStation();

PlantTollStation.prototype.money = 20;

PlantTollStation.prototype.name = '植物园';

// 让校巴进入动物园收费站

var tollStation1 = new ZooTollStation();

var schoolBus = new SchoolBus();

tollStation1.setBus(schoolBus);

tollStation1.fee();

console.log('-------------------------------------');

// 让小巴进入动物园收费站

var tollStation2 = new ZooTollStation();

var miniBus = new MiniBus();

tollStation2.setBus(miniBus);

tollStation2.fee();

console.log('-------------------------------------');

// 让校巴进入植物园收费站

var tollStation1 = new PlantTollStation();

var schoolBus = new SchoolBus();

tollStation1.setBus(schoolBus);

tollStation1.fee();

console.log('-------------------------------------');

// 让小巴进入植物园收费站

var tollStation2 = new PlantTollStation();

var miniBus = new MiniBus();

tollStation2.setBus(miniBus);

tollStation2.fee();

console.log('-------------------------------------');



定义一系列的算法，在不同的环境下，可以替换不同的算法，就是桥接。

而策略是在一样的环境。

# **JS常用设计模式**

**<https://www.cnblogs.com/imwtr/p/9451129.html>**

**<https://zhuanlan.zhihu.com/p/68165406>**

**<https://segmentfault.com/a/1190000012547750>**