[设计模式之工厂模式](http://www.cnblogs.com/TomXu/archive/2012/02/23/2353389.html)

**介绍**

与创建型模式类似，工厂模式创建对象（视为工厂里的产品）时无需指定创建对象的具体类。

工厂模式定义一个用于创建对象的接口，这个接口由子类决定实例化哪一个类。该模式使一个类的实例化延迟到了子类。而子类可以重写接口方法以便创建的时候指定自己的对象类型。

这个模式十分有用，尤其是创建对象的流程赋值的时候，比如依赖于很多设置文件等。并且，你会经常在程序里看到工厂方法，用于让子类类定义需要创建的对象类型。

**正文**

var Car = (function () {  
 var Car = function (model, year, miles) {  
 this.model = model;  
 this.year = year;  
 this.miles = miles;  
 };  
 return function (model, year, miles) {  
 return new Car(model, year, miles);  
 };  
})();  
  
var tom = new Car("Tom", 2009, 20000);  
var dudu = new Car("Dudu", 2010, 5000);

不好理解的话，我们再给一个例子：

var productManager = {};  
  
productManager.createProductA = function () {  
 console.log('ProductA');  
}  
  
productManager.createProductB = function () {  
 console.log('ProductB');  
}  
   
productManager.factory = function (typeType) {  
 return new productManager[typeType];  
}  
  
productManager.factory("createProductA");

如果还不理解的话，那我们就再详细一点咯，假如我们想在网页面里插入一些元素，而这些元素类型不固定，可能是图片，也有可能是连接，甚至可能是文本，根据工厂模式的定义，我们需要定义工厂类和相应的子类，我们先来定义子类的具体实现（也就是子函数）：

var page = page || {};  
page.dom = page.dom || {};  
//子函数1：处理文本  
page.dom.Text = function () {  
 this.insert = function (where) {  
 var txt = document.createTextNode(this.url);  
 where.appendChild(txt);  
 };  
};  
  
//子函数2：处理链接  
page.dom.Link = function () {  
 this.insert = function (where) {  
 var link = document.createElement('a');  
 link.href = this.url;  
 link.appendChild(document.createTextNode(this.url));  
 where.appendChild(link);  
 };  
};  
  
//子函数3：处理图片  
page.dom.Image = function () {  
 this.insert = function (where) {  
 var im = document.createElement('img');  
 im.src = this.url;  
 where.appendChild(im);  
 };  
};

那么我们如何定义工厂处理函数呢？其实很简单：

page.dom.factory = function (type) {  
 return new page.dom[type];  
}

使用方式如下：

var o = page.dom.factory('Link');  
o.url = 'http://www.cnblogs.com';  
o.insert(document.body);

至此，工厂模式的介绍相信大家都已经了然于心了，我就不再多叙述了。

**总结**

**什么时候使用工厂模式**

以下几种情景下工厂模式特别有用：

1. 对象的构建十分复杂
2. 需要依赖具体环境创建不同实例
3. 处理大量具有相同属性的小对象

**什么时候不该用工厂模式**

不滥用运用工厂模式，有时候仅仅只是给代码增加了不必要的复杂度，同时使得测试难以运行下去。