JavaScript 语言精粹四

...往往会把一件完整的东西化成无数的形象，如凹凸镜一般，正面望去，一片模糊 - 威廉·莎士比亚《理查二世》

起步：

- 多数编程语言中，继承都是一个重要的主题
- 基于类的语言中（如JAVA）,继承提供两个重要的服务
  - 代码复用，显著减少软件开发成本
  - 类继承可以引入一套类型系统规范，减轻工作量
- JavaScript 是一门弱类型语言，不需类型转换，无需关注对象继承关系
- JavaScript 提供一套更丰富的代码复用模式，可模拟基于类的模式
- JavaScript 是一门基于原型的语言，意味着对象直接从其他对象继承

伪类

- javascript 中某些复杂语法看起来像那些基于类的语言，它不直接让对象从其它对象继承，反而是使用构造器函数产生对象
  - 当函数对象被创建时，Function 构造器产生的函数对象会运行：

```
this.prototype = { constructor: this };
```

  - 新对象被赋予 prototype 属性，（值是包含constructor属性，且属性值为该新函数的对象）
  - prototype 用以存放继承特征javascript 中每个函数都包含 prototype 属性
  - 当采用构造器调用模式时，（使用 new调用），函数的执行方式会改变

- 定义一个构造器，并扩充它的原型

```
var Man = function (name) {
  this.name = name;
};
Man.prototype.getName = function () {
  return this.name;
};
Man.prototype.says = function () {
  return this.saying || "";
};
```

- 构造实例

```
var myName=new Man('poo');
var name=myName.getName();
console.log(name);// poo
```

- 还可以构造另一个伪类继承 Man,

```
var Dog=function(name){
  this.name=name;
  this.saying='aHa';
}
Dog.prototype=new Man();//替换它的prototype
Dog.prototype.wangwang=function(n){
  var i,txt='';
  for(i=0;i<n;i++){
    s=s?s+='~':'汪'
  }
  return txt;
}
var yourDog=new Dog('花花')
var says=yourDog.says();// aHa
var wangwang=yourDog.wangwang(3);//汪~~
```

- 使用构造器有一个严重的危害，调用构造器函数时，若没在前面加上 new,则**this 不会绑定到新对象上**，反而绑定到了全局，破坏了全局变量环境
- 为了降低这个风险，构造器函数约定俗成首字母大写。

对象说明符

- 通常构造器要接受一些参数，为避免记错顺序引起的问题
- 一般都会使用一个简单的对象说明符来描述
- 这样就不必在意它的顺序了，而且也更易于阅读使用。例如:

```
var myObj=maker(a,b,c,d);
//---以上可写为
var myObj=maker({
  isa:a,
  isb:b,
  isc:c,
  isd:d
})
```

原型

- 基于原型的继承相比基于类的继承概念上更简单——即：新对象可以继承旧对象的属性
- 例如，使用对象字面量构造一个对象

```
var man = {
  name: "poo",
  getName: function () {
    return this.name;
```

```
  },
  says: function () {
    return this.saying || "";
  },
};
```

- 再利用Object.create方法构造出其他实例

```
var myFriends = Object.create(man);
myFriends.name = "tomato";
myFriends.says = "Hi~  potato";
myFriends.getName = function () {
  return this.name + " " + this.says;
};
```

- 通过定制新对象，来指定与基于的基本对象区别的方式，叫做差异化继承。

函数化

- 继承模式缺点在于数据私密性不强，因为对象中的属性都是可见的
- 若想构造私有属性，那么可以使用 应用模块模式
- 从构造一个可以生成对象的函数（它并不需要使用 new 操作符，因此命名不需大写）
    1. 创建新对象。使用Object.create方法
    2. 定义私有实例变量和方法，是函数中通过var语句定义的普通变量
    3. 给当前新对象扩充方法，使这个方法拥有访问参数及(2)中var声明的普通变量的特权
    4. 返回当前新对象。

例如：

```
var constructor=function(spec,my){
  var that,//-其他的私有实例变量
  my=my||{};
  //共享变量和函数添加到my中
  that=//一个新对象
  //添加给that的特权方法
  return that;
}
```

> spec 对象包含构造器需要构造一个新实例的所有信息。spec 的内容可能会被复制到私有变量，或被其他函数改变，或方法在需要的时候可以访问 spec。

- my对象是一个为继承链中的构造器所提供私密共享的容器，my 对象可以选择性使用，若 my 为空，则创建一个 my 对象（ my=my||{ } ）
- 然后声明该对象私有的实例变量和方法，构造器的变量和内部函数都变成该实例的私有成员，内部函数可访问spec my that及其他变量。
- 然后通过赋值语句，给my添加共享私有成员

```
my.member=value
```

- 然后构造新对象，将其赋值给`that`,再进行扩充,后边我就迷糊了，还没看懂。

部件

- 可以利用一套部件把对象组装出来，比如构造一个给对象添加时间处理的函数
- 给对象添加`on`方法，`fire`方法和私有事件注册表对象

```
var eventuality = function (that) {
  var registry = {};
  /**
   * 在一个对象上触发一个事件
   * 可以是一个含事件名称的字符串
   * 或是一个 包含事件名的type属性的 对象
   * 通过 on 方法注册的事件处理程序中匹配事件名称的函数会被调用
   */
  that.fire = function (event) {
    var array,
      func,
      handler,
      type = typeof event === "string" ? event : event.type;
    //若该事件存在一组事件，则遍历且顺序执行
    if (registry.hasOwnProperty(type)) {
      array = registry[type];
      for (let i = 0; i < array.length; i++) {
        handler = array[i];
        //每个处理程序中包括一个方法和一组可选参数
        //若该方法是字符串形式的，则找到该函数
        func = handler.method;
        if (typeof func === "string") {
          func = this[func];
        }
        //调用此处理程序，若该条包括参数，则传递，否则传递该对象
        func.apply(this, handler.parameters || [event]);
      }
    }
    return this;
  };
  //注册一个事件，构造一条处理程序条目，将其插入处理程序数组
  //若此类型事件不存在则构造
  that.on = function (type, method, parameters) {
    var handler = {
      method: method,
      parameters: parameters,
    };
    if (registry.hasOwnProperty(type)) {
      registry[type].push(handler);
    } else {
      registry[type] = [handler];
```

```
    }
    return this;
  };
  return that;
};
```

- 可在任何单独对象上调用eventuality,授予其事件处理方法，也可在that返回前在构造器函数中调用它。

```
eventuality(that);
```

- 当前这种方式中：
    - 一个构造器函数可从一套部件中组装出对象
    - 因为JavaScript弱类型语言的原因，所以无需去了解对象在类型系统中的继承关系
    - 若想eventuality访问该对象私有状态，则可把私有成员集合my传递进去

总结：

> 在《JavaScript 精粹》这本书中，本章节描述的太贵晦涩难懂，本章节我会后续再通读几遍，然后重新组织下语言。