MontageJS 对象

MontageJS在JavaScript对象继承方式上做了一层简单封装。类型通过 `constructor` 方法构造。构造器有一个 `prototype` 。 `prototype` 又有一个 `constructor` 方法。当在对象上执行 `instanceof` 和 `new` 时就跟JavaScript对象一样。

例如，下面的A和B两个例子是同等的：

例A：JavaScript (ECMAScript 5)

```
function Penguin() {
    Bird.call(this);
}

Penguin.prototype = Object.create(Bird.prototype);

Penguin.prototype.constructor = Penguin;

Penguin.prototype.fly = function () {
    return Bird.prototype.fly.call(this);
};

Object.defineProperty(Penguin.prototype, "habitat", {
    get: function () {
        return this._habitat;
    },
    set: function (value) {
        this._habitat = value;
    }
});

Penguin.staticMethod = function () {};
```

例B：MontageJS

```
var Penguin = Bird.specialize({
    constructor: {
        value: function Penguin() {
            this.super();
        }
    },
    fly: {
        value: function () {
            return this.super();
        }
    },
    _habitat: {value: null},
    habitat: {
        get: function () {
            return this._habitat;
        },
        set: function (value) {
            this._habitat = value;
        }
    }
}, {
    staticMethod: {
        value: function () {}
    }
});
```

MontageJS的构造器使用 `specialize` 方法接收2个参数，第一个参数接受ECMAScript 5属性描述器来构造新的类，以及类的属性。第二个参数是可选参数，用来定义类方法。它使用 `Object.create` 来继承父类。使用 `Object.defineProperty` 来描述新类的属性。总的来说，这个方式提供了一个相对于原生Javascript更简易的，不易出错的方式来定义类。

MontageJS方法

不过MontageJS也提供一些新增功能。在任何的 `Montage` 方法中， `super(...args)` 会调用父类中同名的方法。同样，在get方法中使用 `super()` 会获取父类同名属性，在set方法中使用 `super(value)` 会设置父类同名属性。

在这个例子中， `Type` 类实现了一个 `id` get方法，返回值每访问一次加1。 `Subtype` 重写了 `id` get方法，在父类返回值前加一个下划线。

```
var ids = new WeakMap();
var nextId = 0;
var Type = Montage.specialize({
    id: {
        get: function () {
            if (!ids.has(this)) {
                ids.set(this, nextId++);
            }
            return ids.get(this);
        }
    }
});

var Subtype = Type.specialize({
    id: {
        get: function () {
            return "_" + this.super();
        }
    }
});
```

ontageJS还对ES5 property-descriptor做了一些其它的小修改。Javascript对象的任何属性如果不特殊指定都会出现在 `enumerable` 迭代中并且可 `writable` 和 `configurable` 。但是在MontageJS对象中如果属性名以下划线开始或者属性是一个方法，这两种属性都不会出现在迭代中。

扩展Javascript模型

[illegible]

通过 `Montage.specialize` 增强JavaScript对象继承方式好处是构造方法可以继承父类构造方法，但是原型链是并行分开的。这样我们就有机会重写类的 `Montage.specialize` ， `defineProperties` ，和 `defineProperty` ，所有继承这个类的子类都会调用相应的方法。Montage实现了 `specialize` ， `defineProperties` 和 `defineProperty` 默认功能，类重写这些方法可以为所有继承于该类的子类的构造函数实现钩子模式。

```
var Type = Montage.specialize({
}, {
    specialize: {
        value: function () {
            return this.super();
        }
    },
    defineProperty: {
        value: function (object, descriptor) {
            return this.super(object, descriptor);
        }
    }
});
```

为了方便调试，你可以在构造方法中设置类型名，类型名会被存储在 `prototype` 属性中。因为把 `prototype` 属性设置为类型名，在原型链中就不是缺省的类型名（数字ID名），在调试时就知道当前对象类型。

```
var Type = Montage.specialize({
    constructor: {
        value: function Type() {
            this.super();
        }
    }
});

var Subtype = Type.specialize({
```

```
    constructor: {
        value: function Subtype() {
            return this.super();
        }
    }
});
```