JavaScript设计模式

大纲

- 基础
- 封装
- 继承
- 单体模式
- 链式调用
- 接口模式
- 其他常用设计模式

基础

匿名函数

```
(function (){
    /*
        doSomething
    */
})();
```

```
(function () {
    /*
        doSomething
    */
}());
```

闭包

```
function init() {
    var name = "JavaScript";
    function displayName() {
        alert(name);
    }
    displayName();
}
init();
```

闭包

```javascript
function makeFunc() {
    var name = "JavaScript";
    function displayName() {
        alert(name);
    }
    return displayName;
}

var myFunc = makeFunc();
myFunc();
```

构造器

```
var Anim = function () {

};

Anim.prototype.start = function() {

};

Anim.prototype.stop = function() {

};
```

封装

```
var Book = function(isbn, title, author) {
    this.setIsbn(isbn);
    this.setTitle(title);
    this.setAuthor(author);
};

Book.prototype = {
    checkIsbn: function(isbn) {

    },
    getIsbn: function() {
        return this.isbn;
    },
    setIsbn: function() {
        if(!this.checkIsbn(isbn)) throw new Error('Book: Invalid ISBN.');
        this.isbn = isbn;
    },
    getTitle: function() {
        return this.title;
    },
    setTitle: function(title) {
        this.title = title || 'No title specified';
    },
    getAuthor: function() {
        return this.author;
    },
    setAuthor: function(author) {
        this.author = author || 'No author specified';
    },
    display: function() {

    }
};
```

```
var Book = function(isbn, title, author) {
    this.setIsbn(isbn);
    this.setTitle(title);
    this.setAuthor(author);
};

Book.prototype = {
    checkIsbn: function(isbn) {

    },
    getIsbn: function() {
        return this._isbn;
    },
    setIsbn: function() {
        if(!this.checkIsbn(isbn)) throw new Error('Book: Invalid ISBN.');
        this._isbn = isbn;
    },
    getTitle: function() {
        return this._title;
    },
    setTitle: function(title) {
        this._title = title || 'No title specified';
    },
    getAuthor: function() {
        return this._author;
    },
    setAuthor: function(author) {
        this._author = author || 'No author specified';
    },
    display: function() {

    }
};
```

还有什么写法?

```
var Book = function(newIsbn, newTitle, newAuthor) {

    var isbn, title, author; //私有属性

    function checkIsbn(isbn) {} //私有方法

    this.getIsbn = function() { return isbn; } //公开特权方法

    this.setIsbn = function(newIsbn) {
        if(!this.checkIsbn(newIsbn)) throw new Error('Book: Invalid ISBN.');
        isbn = newIsbn;
    }

    this.getTitle = function() { return title; }

    this.setTitle = function(newTitle) {
        title = newTitle || 'No title specified';
    }

    this.getAuthor = function(Author) { return author; }

    this.setAuthor = function(newAuthor) {
        author = newAuthor || 'No author specified';
    }

    this.setIsbn(newIsbn);
    this.setTitle(newTitle);
    this.setAuthor(newAuthor);
};

Book.prototype = {
    display: function() {} //公开非特权方法
};
```

优点是显而易见的，缺点是什么？

- 耗费更多内存
- 私有属性和方法无法继承

```javascript
var Book = (function() {

    var numOfBooks = 0; //静态私有属性

    function checkIsbn(isbn) {} //静态私有方法

    return function(newIsbn, newTitle, newAuthor) {
        var isbn, title, author; //私有属性

        this.getIsbn = function() { return isbn; }; //公开特权方法
        this.setIsbn = function(newIsbn) {
            if(!checkIsbn(newIsbn)) throw new Error('Book: Invalid ISBN.');
            isbn = newIsbn;
        };
        this.getTitle = function() { return title; };
        this.setTitle = function(newTitle) {
            title = newTitle || 'No title specified';
        };
        this.getAuthor = function() { return author;};
        this.setAuthor = function(newAuthor) {
            author = newAuthor || 'No author specified';
        };

        numOfBooks++;
        if(numOfBooks > 50) throw new Error('Book: Only 50 instances of Book can be created.');

        this.setIsbn(newIsbn);
        this.setTitle(newTitle);
        this.setAuthor(newAuthor);
    };

}());

Book.convertToTitleCase = function(inputString) {}; //静态公开方法

Book.prototype = {
    display: function() {} //公开非特权方法
};
```

继承

类式继承

```
function Person(name) {
    this.name = name;
}

Person.prototype.getName = function() {
    return this.name;
};

function Author(name, books) {
    Person.call(this, name);
    this.books = books;
}

Author.prototype = new Person();
Author.prototype.constructor = Author;
Author.prototype.getBooks = function() {
    return this.books;
}
```

```
function extend(subClass, superClass) {
    var F = function() {};
    F.prototype = superClass.prototype;
    subClass.prototype = new F();
    subClass.prototype.constructor = subClass;
}

function Person(name) {
    this.name = name;
}

Person.prototype.getName = function() {
    return this.name;
};

function Author(name, books) {
    Person.call(this, name);
    this.books = books;
}

extend(Author, Person);

Author.prototype.getBooks = function() {
    return this.books;
}
```

```
function extend(subClass, superClass) {
    var F = function() {};
    F.prototype = superClass.prototype;
    subClass.prototype = new F();
    subClass.prototype.constructor = subClass;

    subClass.superclass = superClass.prototype; //增加superClass属性
    if(superClass.prototype.constructor === Object) {
        superClass.prototype.constructor = superClass;
    }
}

function Person(name) {
    this.name = name;
}

Person.prototype.getName = function() {
    return this.name;
}

function Author(name, books) {
    Author.superclass.constructor.call(this, name);
    this.books = books;
}

extend(Author, Person);
Author.prototype.getBooks = function() {
    return this.books;
};

Author.prototype.getName = function() {
    var name = Author.superclass.getName.call(this);
    return name + ', Author of ' + this.getBooks().join(', ');
}
```

原型式继承

```
function clone(object) {
    function F() {}
    F.prototype = object;
    return new F;
}

var Person = {
    name: 'default name',
    getName: function() {
        return this.name;
    }
};

var Author = clone(Person);
Author.books = [];
Author.getBooks = function() {
    return this.books;
}
```

思考：有什么问题?

```
var authors = [];

authors[0] = clone(Author);
authors[0].name = 'Dustin Diaz';
authors[0].books[0] = 'JavaScript Design Patterns';

authors[1] = clone(Author);
authors[1].name = 'Ross Harmes';
authors[1].books[0] = 'JavaScript Language';
```