TCL

创意感动生活
The Creative Life

javascript模块化开发

JavaScript开发经历的几个过程

1. 过程式的JavaScript
2. 面向对象的JavaScript
3. 面向模块的JavaScript

什么是模块

模块就是实现特定功能的一组方法
只要把不同的函数（以及记录状态的变量）简单地放在一起，就算是一个模块。

```
function m1(){
	//...
}

function m2(){
	//...
}
```

常用模块创建方法

使用"立即执行函数"，可以达到不暴露私有成员的目的。

```
(function(window){
        var _count = 0; //私有变量
        var m1 = function(){
              //...
        };
        var m2 = function(){
              //...
        };
          //暴露模块名
        window.module = {
              m1 : m1,
              m2 : m2
        };
 })(window);
```

使用上面的写法，外部代码无法读取内部的_count变量。

```
console.info(module._count); //undefined
```

外部调用模块里面的方法：

```
module.m1()
```

开发中经常遇到的问题

1、命名冲突

我们从一个简单的习惯出发。我做项目时，常常会将一些通用的、底层的功能抽象出来，独立成一个个函数，比如

```
function each(arr) {
  // 实现代码
}

function log(str) {
  // 实现代码
}
```

把这些函数统一放在 util.js 里。需要用到时，引入该文件就行。

其他人想定义一个 each 方法遍历对象，但页头的 util.js 里已经定义了一个，那就只能换 eachObject；

为了避免这种情况，大家提出了新的解决方案，参照 Java 的方式
——命名空间

下面是一段来自 Yahoo! 的一个开源项目，YUI2。

```
if (org.cometd.Utils.isString(response)) {
  return org.cometd.JSON.fromJSON(response);
}
if (org.cometd.Utils.isArray(response)) {
  return response;
}
```

新的问题：过长的命名空间难于记忆，代码过于复杂

2、烦琐的文件依赖

基于 util.js，我开始开发 UI 层通用组件，这样项目组同事就不用重复造轮子了。
其中有一个组件是 dialog.js，使用方式很简单。

```
<script src="util.js"></script>

<script src="dialog.js"></script>

<script>

  org.CoolSite.Dialog.init({ /* 传入配置 */ });

</script>
```

如果其他人或者其他页面需要用dialog.js就必须引用util.js

一个项目里面会有十多或者几十个组件，你完全不知道你想用的那个组件会依赖那个js文件

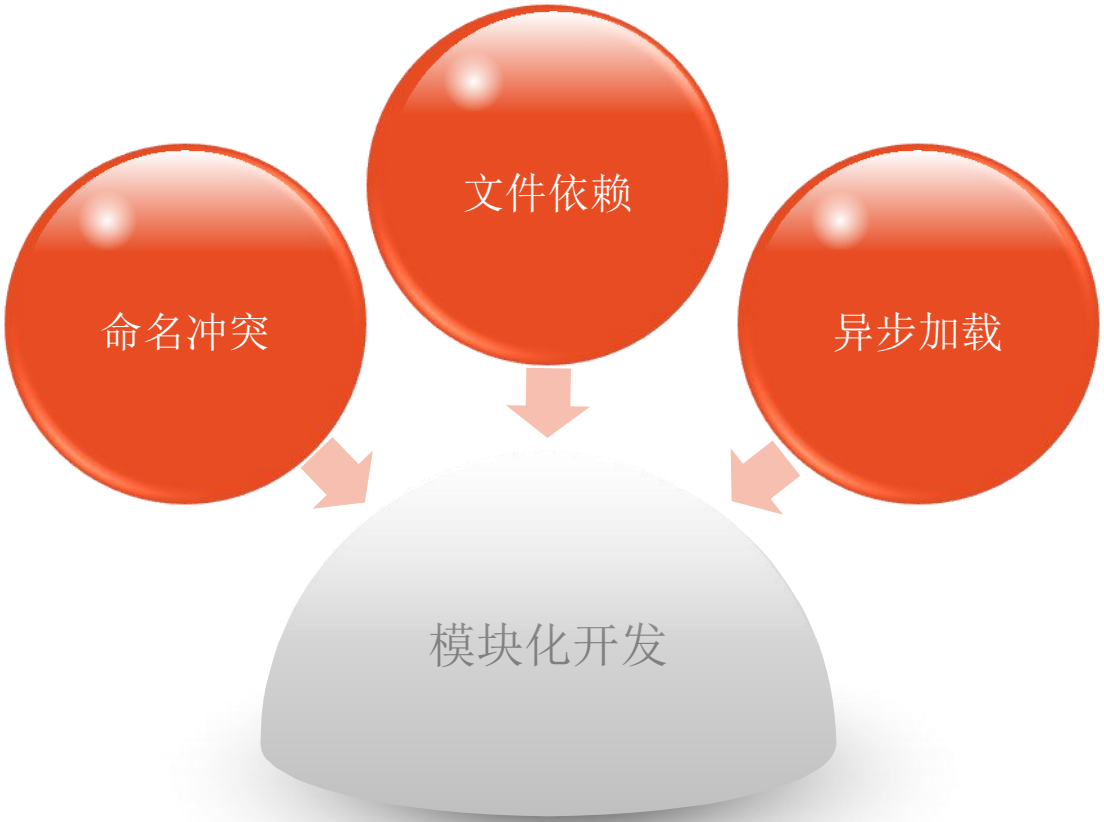
3、javascript加载

一般浏览器用<script>标签引入js，js文件会顺序（串行）加载顺序执行，但是，串行加载js文件的数量有限制，会阻塞后面的代码加载和执行

使用异步加载js，可以让js并行加载，经常会使用下面这种方式：

```
var head = document.getElementsByTagName('head')[0],
    script = document.createElement('script');
script.src = url;
head.appendChild(script);
```

不能保证执行顺序， 谁先加载完就先执行谁，现有解决方案 LABJS


文件依赖
命名冲突
异步加载
模块化开发

模块的规范

有了模块，我们就可以更方便地使用别人的代码，想要什么功能，就加载什么模块。

但是，这样做有一个前提，那就是大家必须以同样的方式编写模块(es6新特性自带模块定义方法)

一般一个文件作为一个模块, 文件名作为模块名。

目前，通行的Javascript模块规范共有两种：

CMD（Common Module Definition， Commonjs模块定义）
AMD（Asynchronous Module Definition，异步模块定义）

（浏览器端使用模块化开发都需要用script标签来引入模块加载器）

CommonJS规范写法（nodejs）

模块定义

```
//math.js
exports.add = function(n1, n2) {
    return n1+n2;
};
```

模块使用

```
//app.js
var math = require('math');

math.add(2,3); // 5
```

（nodejs会自动给文件加上模块的头和尾）
上面代码会自动转换成

```
define(function(require, exports, module){
    exports.add = function(n1, n2) {
        return n1+n2;
    };
});
```

AMD加载模块写法（浏览器端对js异步加载）

模块定义

```
//math.js
define(function(){
    var add = function(n1, n2){
        return n1+n2;
    };
    return {
        add:add
    };
});
```

模块使用

```
//app.js
require(['math'], function (math, other) {

    math.add(2, 3);

});
```

AMD模块加载的使用(requirejs)

如果一个模块还依赖其他模块，那么define()函数的第一个参数，必须是一个数组，指明该模块的依赖性

```
//util.js
 define( function(){
        var util = {
             doSomething:function(){
                  alert('...');
             }
        };
        return util;
 });

 //使用
 require(['dialog'], function(dialog){
        dialog.open();
 });
```

```
//dialog.js
 define(['util'], function(util){

        var dialog = {
             open:function(){
                  util.doSomething();
             }
        };
        return dialog;
 });
```

当require()函数加载dialog这个模块的时候，就会先加载util.js文件。不需要模块的使用者手动去加载util.js文件

模块的命名

1,默认使用文件路径加文件名来作为模块命引用：

--js/lib/util.js
--js/app.js

```
// lib/util.js
define(function(){return util;});

//app.js
require(['lib/util'], function(util){
    util.doSomething();
});
```

页面加载app.js, 根据依赖的模块名查找文件并加载lib/util.js

2,自定义模块名(不建议)：

```
// lib/util.js
define('util', [依赖的其他模块], function(){return util;});

//app.js
require(['util'], function(util){
    util.doSomething();
});
```

注意：util.js 必须要先于app.js加载

CMD模块加载写法（seajs，requirejs2.0）

```
define(function (require) {
    var dependency1 = require('dependency1'),
        dependency2 = require('dependency2');
    return function () {};
});
```

通过 Function.prototype.toString() 来解析require的模块名，在内部将上述 define 调用转换成这种形式

```
define(['require', 'dependency1', 'dependency2'], function (require) {
    var dependency1 = require('dependency1'),
        dependency2 = require('dependency2');
    return function () {};
});
```

谢　谢！

TCL集团股份有限公司

www.tcl.com

@TCL创意感动生活