前端模块管理器简介

作者： 阮一峰

日期： 2014 年 9 月 14 日

来源：http://www.ruanyifeng.com/blog/2014/09/package-management.html

模块化结构已经成为网站开发的主流。

制作网站的主要工作，不再是自己编写各种功能，而是如何将各种不同的模块组合在一起。

浏览器本身并不提供模块管理的机制，为了调用各个模块，有时不得不在网页中，加入一大堆 script 标签。这样就使得网页体积臃肿，难以维护，还产生大量的 HTTP 请求，拖慢显示速度，影响用户体验。

为了解决这个问题，前端的模块管理器（package management）应运而生。它可以轻松管理各种 JavaScript 脚本的依赖关系，自动加载各个模块，使得网页结构清晰合理。不夸张地说，将来**所有**的前端 JavaScript 项目，应该都会采用这种方式开发。

最早也是最有名的前端模块管理器，非 RequireJS 莫属。它采用 AMD 格式，异步加载各种模块。具体的用法，可以参考我写的教程。Require.js 的问题在于各种参数设置过于繁琐，不容易学习，很难完全掌握。而且，实际应用中，往往还需要在服务器端，将所有模块合并后，再统一加载，这多出了很多工作量。

今天，我介绍另外四种前端模块管理器：Bower，Browserify，Component 和 Duo。它们各自都有鲜明的特点，很好地弥补了 Require.js 的缺陷，是前端开发的利器。

需要说明的是，这篇文章并不是这四种模块管理器的教程。我只是想用最简单的例子，说明它们是干什么用的，使得读者有一个大致的印象，知道某一种工作有特定的工具可以完成。详细的用法，还需要参考它们各自的文档。

Bower

Bower 的主要作用是，为模块的安装、升级和删除，提供一种统一的、可维护的管理模式。

首先，安装 Bower。

```
$ npm install -g bower
```

然后，使用 bower install 命令安装各种模块。下面是一些例子。

```
# 模块的名称
$ bower install jquery
# github 用户名/项目名
$ bower install jquery/jquery
# git 代码仓库地址
$ bower install git://github.com/user/package.git
# 模块网址
$ bower install http://example.com/script.js
```

所谓"安装"，就是将该模块（以及其依赖的模块）下载到当前目录的 bower_components 子目录中。下载后，就可以直接插入网页。

```
<script src="/bower_componets/jquery/dist/jquery.min.js">
```

bower update 命令用于更新模块。

```
$ bower update jquery
```

如果不给出模块的名称，则更新所有模块。

bower uninstall 命令用于卸载模块。

```
$ bower uninstall jquery
```

注意，默认情况下，会连所依赖的模块一起卸载。比如，如果卸载 jquery-ui，会连 jquery 一起卸载，除非还有别的模块依赖 jquery。

Browserify

Browserify 本身不是模块管理器，只是让服务器端的 CommonJS 格式的模块可以运行在浏览器端。这意味着通过它，我们可以使用 Node.js 的 npm 模块管理器。所以，实际上，它等于间接为浏览器提供了 npm 的功能。

首先，安装 Browserify。

```
$ npm install -g browserify
```

然后，编写一个服务器端脚本。

```
var uniq = require('uniq');
var nums = [ 5, 2, 1, 3, 2, 5, 4, 2, 0, 1 ];
console.log(uniq(nums));
```

上面代码中 uniq 模块是 CommonJS 格式，无法在浏览器中运行。这时，Browserify 就登场了，将上面代码编译为浏览器脚本。

```
$ browserify robot.js > bundle.js
```

生成的 bundle.js 可以直接插入网页。

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

Browserify 编译的时候，会将脚本所依赖的模块一起编译进去。这意味着，它可以将多个模块合并成一个文件。

Component

Component 是 Express 框架的作者 TJ Holowaychuk 开发的模块管理器。它的基本思想，是将网页所需要的各种资源（脚本、样式表、图片、字体等）编译后，放到同一个目录中（默认是 build 目录）。

首先，安装 Component。

```
$ npm install -g component@1.0.0-rc5
```

上面代码之所以需要指定 Component 的版本，是因为 1.0 版还没有正式发布。

然后，在项目根目录下，新建一个 index.html。

```
<!DOCTYPE html>
<html>
  <head>
    <title>Getting Started with Component</title>
    <link rel="stylesheet" href="build/build.css">
  </head>
  <body>
    <h1>Getting Started with Component</h1>
    <p class="blink">Woo!</p>
    <script src="build/build.js"></script>
  </body>
```

```
</html>
```

上面代码中的 build.css 和 build.js，就是 Component 所要生成的目标文件。

接着，在项目根目录下，新建一个 component.json 文件，作为项目的配置文件。

```
{
  "name": "getting-started-with-component",
  "dependencies": {
    "necolas/normalize.css": "^3.0.0"
  },
  "scripts": ["index.js"],
  "styles": ["index.css"]
}
```

上面代码中，指定 JavaScript 脚本和样式表的原始文件是 index.js 和 index.css 两个文件，并且样式表依赖 normalize 模块（不低于 3.0.0 版本，但不高于 4.0 版本）。这里需要注意，Component 模块的格式是"github 用户名/项目名"。

最后，运行 component build 命令编译文件。

```
$ component build
   installed : necolas/normalize.css@3.0.1 in 267ms
       build : resolved in 1221ms
       build : files in 12ms
       build : build/build.js in 76ms - 1kb
       build : build/build.css in 80ms - 7kb
```

在编译的时候，Component 自动使用 autoprefixer 为 CSS 属性加上浏览器前缀。

目前，Component 似乎处于停止开发的状态，代码仓库已经将近半年没有变动过了，官方也推荐优先使用接下来介绍的 Duo。

Duo

Duo 是在 Component 的基础上开发的，语法和配置文件基本通用，并且借鉴了 Browserify 和 Go 语言的一些特点，相当地强大和好用。

首先，安装 Duo。

```
$ npm install -g duo
```

然后，编写一个本地文件 index.js。

```
var uid = require('matthewmueller/uid');
var fmt = require('yields/fmt');

var msg = fmt('Your unique ID is %s!', uid());
window.alert(msg);
```

上面代码加载了 uid 和 fmt 两个模块，采用 Component 的"github 用户名/项目名"格式。

接着，编译最终的脚本文件。

```
$ duo index.js > build.js
```

编译后的文件可以直接插入网页。

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

Duo 不仅可以编译 JavaScript，还可以编译 CSS。

```
@import 'necolas/normalize.css';
@import './layout/layout.css';

body {
  color: teal;
  background: url('./background-image.jpg');
}
```

编译时，Duo 自动将 normalize.css 和 layout.css，与当前样式表合并成同一个文件。

```
$ duo index.css > build.css
```

编译后，插入网页即可。

```
<link rel="stylesheet" href="build.css">
```

（完）