文章编号:1009-2552(2018)08-0044-04 DOI:10.13274/j.cnki.hdzj.2018.08.010

Web前端工程化解决方案研究

周 伟[1],郑世珏[2]

(1.华中师范大学信息化办公室,武汉 430079;2.华中师范大学计算机学院,武汉 430079)

摘 要: 随着互联网的快速发展,Web业务日益复杂化,Web前端开发在软件产品开发环节中的作用变得越来越重要。目前Web前端的开发效率低、开发质量差是亟待解决的问题。用软件工程的思想来研究和解决Web前端工程问题,结合已有的Web前端框架和技术,设计出一套行之有效的Web前端工程化解决方案。经过实际项目的实践,解决方案合理可行,实现了高效、高质量的Web前端开发。

关键词: Web前端;软件工程;自动化;vue.js

中图分类号: TP393.09 **文献标识码:** A

Research on Web front-end engineering solutions

ZHOU Wei[1], ZHENG Shi-jue[2]

(1. Information Office, Central China Normal University, Wuhan 430079, China;
2. Computer School, Central China Normal University, Wuhan 430079, China)

Abstract: With the rapid development of the Internet and increasingly complicated Web services, the role of Web front-end development in software product development is becoming more and more important. The low efficiency and poor quality of Web front-end development are now an urgent problem to be solved. This paper uses software engineering methods to do research and solve the Web front-end engineering problems. It combines with the existing Web front-end engineering frameworks and technologies to achieve an effective Web front-end engineering solution. After the actual project practice, the solution proved it is reasonable and feasible, which can achieve efficient, high-quality Web front-end development.

Key words: Web front-end; software engineering; automation; vue.js

0 引言

随着互联网的高速发展,Web前端开发在产品开发环节中的作用变得越来越重要。Web前端开发的发展经历了刀耕火种时期、手工工场时期和工业革命时期。

在刀耕火种时期,前端的概念还未完全明确。此时的典型特征就是页面主要依赖服务端语言,如JSP、ASP、PHP等模版引擎来进渲染,配合上少量的简单的CSS和JavaScript代码,来实现简单的页面展示[1]。

因为页面主要依赖后端模版引擎来实现,所以主要代码依旧存放在后端的代码库中,整个项目也都是以后端开发为主导的。此时的前后端分工并不明确,前端的工作直接交由后端工程师完成。前后端职责划分不清,模版引擎语言和前端语言的杂乱糅合,导致项目的阅读性很差,可维护性大大降低。

2005年Ajax技术的出现,使得前端开发进入了手工工场时期。通过Ajax技术,实现了视觉上局部刷新,不会存在刷新的白屏等待时间,同时对也减少了前后端传递的数据量,大大提高了性能[2]。但此时的前端开发,仍旧处于简单网页开发模式,大量工程师依旧推崇小而美的开发模式,基本没有工程化

收稿日期:2018-02-05
基金项目:国家支撑计划(2015BAK33B00)
作者简介:周伟(1980-),男,博士,工程师,研究方向为网络安全管理、系统开发。

的概念。

随着 Web 2.0 时代的到来，交互功能需求的大大提升，以往的前端框架性能开始无法满足飞速增长的需求。Web 前端开发的工业革命开始，优秀的 JavaScript 框架（Backbone，Vue，React，Angular）纷纷涌现出来。

各类 JavaScript 框架的出现代表着 Web 前端开发不再满足于构建一个简简单单的页面，而更加趋向于开发一个类似于传统软件的 Web 型应用。这些优秀的 JavaScript 框架极大的提高了前端渲染的性能，同时也使开发变的更加敏捷、统一，使 Web 型应用逐渐变成了可能。

但是如果仅仅使用这些框架，却没有使用工程化的开发方法，开发的效率和质量还是难以保障。本文用软件工程的思想来研究和解决 Web 前端工程问题，结合已有的 Web 前端框架和技术，取长补短，总结实现出一套较为全面且行之有效的 Web 前端工程化解决方案。该解决方案包括设计和代码模块化、组件化设计、规范化和自动化。

1 模块化

模块化是以功能块为单位进行程序设计，实现一个独立功能的模块单元[3]。简而言之就是将一个大文件拆分给多个小文件，然后再统一进行拼装和加载，是一种分而治之的思想。

如今的 Web 前端开发中，一个复杂页面所引用的 JavaScript、CSS 的代码有几千行乃至上万行之多。这时候，就需要将 JavaScript，CSS 中的大文件进行模块化划分，划分为一个个具备独立功能的一百行代码左右的模块文件。最后再根据业务功能需要，对这些模块文件进行组装调用即可。

1.1 JavaScript 模块化

JavaScript 由于构建语言之初，创始人 Brendan Eich 只是将其当作一门简单的脚本语言，且仅用了七天时间完成，所以，一开始 JavaScript 在标准上并没有模块系统，直到 2015 年 ES6 标准的提出。这也对前端工程化的发展，造成了巨大的阻碍[4]。

在这之前有许多社区制定了自己的一套模块化加载方案，如 AMD、MDD 等，但是在实施上并没有统一，有些系统并存着几套模块化加载方案。因为每套模块化加载方案的写法并不一致，这样就导致了无法进行合理的统一化开发，很多冗余的代码也一直存在系统中，根本无法剔除。

所幸，ES6 的标准已经相当完善，本方案根据 ES6 的标准，结合 Webpack 这一模块化自动打包工具和 Babel（将 ES6 代码转换成 ES5 代码，使其在大多数浏览器上均可运行，解决兼容性问题的插件）[5]，对 JavaScript 进行模块化的分割。

1.2 CSS 模块化

CSS 在模块化上虽然本身语言支持性比 JavaScript 要好，本身就可以通过 import 引用机制，来进行模块化开发，但是仍然存在一个核心的问题，就是如何解决全局污染问题。

CSS 的选择器无法进行私有化判定，导入一个新的模块后，原有模块的样式很有可能会被全局样式所覆盖。虽然从本身语言上而言，CSS 的样式重写机制是它的一个优势，但是从工程化的角度分析，这个优势却变成了劣势，因为它不利于多人协作的模块化开发。

为了解决这一问题，本方案主要采用了 BEM 风格的命名规约，从规范上去限制 CSS 命名风格，从而规避全局选择器的问题。但是这仅仅只是一个弱约束，假如有人没有遵守这套命名规约，依旧会导致问题。

所以，另一方面从工具的角度上去消灭这种问题，在良好命名规约的基础上，利用自动化工具对 CSS 进行 data 加签，利用随机生成的 data 标签选择器，对每个模块进行选择定义，这样就能保证每个模块经过工具加签后都是私有化的标签选择。保证了一个模块的更改，不会影响其他模块的正常运作。

2 组件化

组件化就是从设计层面上，对用户界面进行拆分，将原本是一个整体的页面，拆分为由首部、内容、底部等组件构成的页面，如图 1 所示。

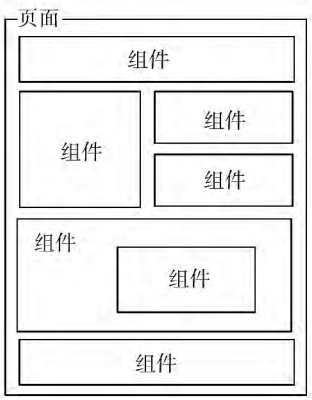

图 1 页面的组件化划分

页面的组件化划分有利于统一管理，出现问题能够进行统一修改和替换。封装良好的每个组件之间也是相互独立的，出现不会互相影响，这对一个大型软件工程项目而言是十分有必要的。

对于 Web 前端开发而言，往往把页面模版（HTML）+页面样式（CSS）+页面逻辑（JavaScript）这三

个部分的代码封装到一起，形成一个功能完备的结构单元，然后将之称为一个组件。实际上，这就是一种从设计层次，对页面的抽象划分工作。其实，假如页面足够简单，那么每一个 DOM 元素都是一个组件[6]。但是随着现在业务的发展，Web 应用观念的提出，简单的 DOM 操作已经无法满足业务需求，所以需要自己对其进行进一步封装，形成一个完备的组件库，才能适应如今的 Web 前端工程化开发。

对比了几种现有的组件化模式，本文最终采用 Vue. js 框架推荐使用的单文件组件形式来实现组件化。

如图 2 所示，将组件结构 HTML + 逻辑 JavaScript + 样式 CSS 写在同一个单页文件中。组件的结构、样式和逻辑都一目了然，修改起来也十分便利。

图 2　Hello. vue 简单实例

3　规范化

规范化是软件开发中必不可少的一环，只有依赖标准进行规范化的管理，软件项目的开发过程才能顺利进行[7-8]。

同样，模块化和组件化的顺利落实离不开规范化。没有规范化去统一管理，很有可能出现各个模块和组件之间代码风格互不一致，代码阅读起来就很费力，造成了理解上的困难，增加了合作的难度。当团队出现人员更替时，需要大量的时间成本来重新熟悉代码，乃至因为无法维护导致的模块重写也是常有的事情。

3.1　目录结构规范

规范的目录结构不仅能够方便日常编码过程中的代码查找，而且对新加入的成员而言，能够使其对整个项目结构一目了然，迅速投入到相应部分的开发中。

如图 3 所示，本文采用 vue-cli 工具，构建一个新项目时，只需要几个简单的 npm 命令就能够快速地生成同样目录结构的项目。

对整个团队而言，所有的项目在目录结构上都应该是大同小异的。以便于不同项目组的开发人员能够快速上手阅读其他组的项目，进行通用模块的代码复用。

图 3　vue-cli 生成的模版目录结构例子

3.2　编码规范

编码规范因为每个团队中人员的不同，也形成了很多种不同的规范方式。往往是在大量的编码过程中，形成的一种良好规约。因此，本文不详细阐述具体的编码规范，主要推荐使用腾讯 alloyteam 的编码规范来统一编码风格。

同时，仅仅依赖文档式的编码规范并不一定能保证规范的正确落实。这时候就需要使用自动化的代码规范化检测工具 ESLint，在编码的过程中立刻产生错误提示，严格限制了代码规范问题。

3.3　接口文档规范

接口文档格式整体采用 markdown，方便提交至仓库时能够清楚的看到接口文档中具体变化了哪些部分。

如图 4 所示，接口采用请求地址 url + 请求参数 request + 返回参数 response 的格式，简洁明了地显示了请求需要传递的数据和接口会返回的数据。

4　自动化

在自动化构建工具未出现之前，前端工程师们往往需要对代码进行手动压缩、合并，利用 PS 对

注册接口

url: /users/register

request:

参数	类型	是否必要	说明
username	String	是	用户名
password	String	是	密码

response:

```
{
    result: '注册成功',
    errorCode: '000000'
}
```

图 4　注册接口文档

CSS sprite 图进行合并等工作。手动去完成这些工作，不仅低效，而且容易出错。代码的每一次改动就需要重复之前所做的工作，极大地拖慢了开发效率。自动化工具的出现不仅降低了出错的可能，同时也大大节约了开发时间。

4.1 自动化构建

如图 5 所示，通过编写 webpack 的 uglifyjs，html-plugin，merge 等插件的配置文件，实现了项目的自动化构建工作，如 JavaScript，CSS，HTML 文件的压缩合并，md5 加签，sprite 图的合成等。

```
var path = require('path')
var utils = require('./utils')
var webpack = require('webpack')
var config = require('../config')
var merge = require('webpack-merge')
var baseWebpackConfig = require('./webpack.base.conf')
var HtmlWebpackPlugin = require('html-webpack-plugin')
var ExtractTextPlugin = require('extract-text-webpack-plugin')
var env = config.build.env

var webpackConfig = merge(baseWebpackConfig, {...})

if (config.build.productionGzip) {...}

if (config.build.bundleAnalyzerReport) {
  var BundleAnalyzerPlugin = require('webpack-bundle-analyzer').BundleAnalyzerPlugin
  webpackConfig.plugins.push(new BundleAnalyzerPlugin())
}

module.exports = webpackConfig
```

图 5 自动化构建配置

开发完成后，配合上 npm 打包命令即可实现前端资源的整合打包，所有前端资源文件被统一打包至 dist 文件夹，上线发布时只需要提供 dist 文件夹即可。

4.2 自动化测试

当修改一个已有模块时，需要对引用该模块的代码进行重新测试，这样才能保证模块修改的正确性。但是依赖手工方式，对引用模块的每个部分都进行测试是十分费时费力的，尤其是当这个模块存在数十个，乃至上百个引用的时候，这种手工测试的方法显示是不可靠的。

所以，如图 6 - 7 所示，本文引入了 QUnit 进行自动化测试，为这个模块编写相应的自动化测试用例，覆盖其引用的可能情况。这样，只需要在修改完模块时，运行一遍自动化测试，只要所有的测试用例都通过了，那么便说明这次修改是可行的。

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>QUnit</title>
</head>

<body>
    <link href="https://code.jquery.com/qunit/qunit-git.css" rel="stylesheet" type="text/css" />
    <script src="https://code.jquery.com/qunit/qunit-git.js"></script>
    <script>
    QUnit.test("hello", function(assert) {
        assert.expect(1);

        var $body = $("body");

        $body.on("click", function() {
            assert.ok(true, "body was clicked!");
        });

        $body.trigger("click");
    });
    </script>
    <h1 id="qunit-header">QUnit Hello World</h1>
    <h2 id="qunit-banner"></h2>
    <ol id="qunit-tests"></ol>
</body>

</html>
```

图 6 自动化测试用例

图 7 测试成功结果

5 结束语

本文给出了一种工程化的 Web 前端开发解决方案。该方案通过模块化和组件化避免重复性的工作，提高开发效率；通过规范化和自动化，自动化完成调试测试等，完成优化工作，提高开发质量。通过多个项目的实践，结果显示该工程化解决方案是可行和值得推广的。

参 考 文 献：

[1] 刘春华. 基于 HTML5 的移动互联网应用发展趋势 [J]. 移动通信，2013(9)：64 - 68.

[2] 王亚楠，吴华瑞，黄锋. 高并发 Web 应用系统的性能优化分析与研究 [J]. 计算机工程与设计，2014(8)：2976 - 2980.

[3] 陈竞艺. 浅析 Web2.0——未来的互联网 [J]. 科技资讯，2011(10)：14.

[4] 辛刚，王清心. 基于 Ajax 的 Java Web 应用的研究与开发 [J]. 山西电子技术，2010(1)：57 - 58.

[5] 夏明忠，夏以轩，李兵元. 软件模块化设计和模块化管理 [J]. 中国信息界，2012(11)：56 - 59.

[6] 戴翔宇. Web 前端工程组件化的分析与改进 [D]. 长春：吉林大学，2016.

[7] 徐頔，朱广华，贾瑶. 基于 VueJs 的 WEB 前端开发研究 [J]. 科技风，2017(14)：69.

[8] 麦冬，陈涛，梁宗湾. 轻量级响应式框架 Vue.js 应用分析 [J]. 信息与电脑：理论版，2017(7)：58 - 59.

责任编辑：薛慧心

(上接第 43 页)

[5] Zhou R，Damerow L，Sun Y，et al. Using colour features of cv. 'Gala' apple fruits in an orchard in image processing to predict yield [J]. Precision Agriculture，2012，13(5)：568 - 580.

[6] 谢祥徐. 基于链表的图像连通区域提取算法 [J]. 数字通信，2012，39(3)：34 - 38.

[7] 陈小娟. 基于水平集的彩色图像分割方法 [J]. 科学技术与工程，2013，13(23)：6756 - 6759，6766.

[8] 齐丽娜，张博，王战凯. 最大类间方差法在图像处理中的应用 [J]. 无线电工程，2006，36(7)：25 - 26.

[9] 陈浩，庞全. 基于 Grab Cut 和八方向链码法的藻类细胞轮廓提取算法 [J]. 机电工程，2010，27(8)：108 - 110.

[10] 许凯. 基于图像识别的苹果果实检测技术 [J]. 实验室研究与探索，2016，35(10)：36 - 39.

[11] 赵文旻，姬长英，李莹莹. 自然场景下成熟苹果的图像识别研究 [J]. 科学技术与工程，2012，12(27)：6889 - 6896.

[12] 石雪强，程新文. 苹果采摘机器人视觉系统的目标提取研究 [J]. 农机化研究，2013，35(10)：46 - 48.

责任编辑：薛慧心