JS一般般的网页重构可以使用Node.js做些什么

这篇文章发布于 2016年06月7日，星期二，00:18，归类于 Web综合。 阅读 53399 次, 今日 10 次 73 条评论

一、非计算机背景前端如何快速了解Node.js?

做前端的应该都听过Node.js，偏开发背景的童鞋应该都玩过。

对于一些没有计算机背景的，工作内容以静态页面呈现为主的前端，可能并未把玩过Node.js，且很有可能对Node.js都没有一个比较立体的认识——知道这玩意可以跑服务，构建很多前端工具，看上去很厉害的样子，但是，可能就仅限于此了。

“那可否三言两语概括Node.js的林林总总呢？”
“不可！”
“那怎么办？”
“那就六言四语！”

首先，要知道，Node.js一个JavaScript运行环境(runtime)，没错，就是用来运行Javascript. 以前JavaScript只能在浏览器这个小世界里称王称霸。很多前端小伙伴可能就JS这门程序语言熟一点，其他C++, .net之类的就呵呵了。如果是过去，如果浏览器一觉醒来灭绝了，很多人就会失业。就像食物单一的物种一旦这种食物没了，就坐等灭绝是一个道理。

但是，现在，不要担心了，Node.js让JavaScript变成杂食的了，也就是除了网页行为，可以和其他C++等语言一样，构建服务，操作文件等等。

我们应该都使用过`.exe`后缀的文件，双击一下，就可以潜伏个病毒什么的；我们可能还使用过`.bat`后缀的批处理文件，一点击，文件夹里面的图片全部重命名；那么如果是`.js`后缀的文件呢（假设你的系统已经安装了Node.js环境），双击一下则……当当当当……会打开编辑器看到JS代码，双击是没有用的！

我们可以打开命令行工具，cd到指定目录，然后输入（假设JS文件名为test.js）：

```
node test
```

然后`test.js`里面的代码就可以欢快地跑起来啦！

对于“页面仔”而言，了解这么多就够了！

1. 安装后Node.js环境；

2. 用我们蹩脚的JS写一个蹩脚处理的 `.js` 文件；
3. `node` 执行下。

简简单单三部曲，我们就变身成了具有开发味道的前端从业人员了。

二、蹩脚JS下的Node.js初体验

绝大数厂子都是小厂，很大部分小厂都只有一个前端，很多前端的JS其实都一般般。

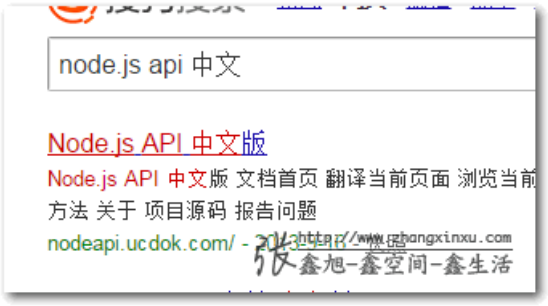
圈子里面经常把“前端解决方案”挂在嘴边的，实际上都是有前端团队的，因为有团队，才能显价值。

“前端解决方案”是好，但是，如果真正关心行业发展，应该知道，能够在一个大团队里面玩耍的实际上是小部分人，有很多很多的小伙伴都是孤军奋战，这套东西说不定反而阻碍了敏捷和灵活；有很多很多的小伙伴在二三四线城市，是野生的前端开发，底子不够，这套庞杂的东西很难驾驭；有很多很多的项目就是几个静态活动页面，没必要回回使用高射炮打蚊子。

此时，往往需要的是定制化很强的小而美的处理。有同学可能会疑虑，哎呀，我JS水平比较菜，自造工具这种事情会不会有点挑大梁啊。实际上，即使你JS一般般，借助Node.js构建一些小工具提升自己的前端开发效率这种事情，完全不在话下。

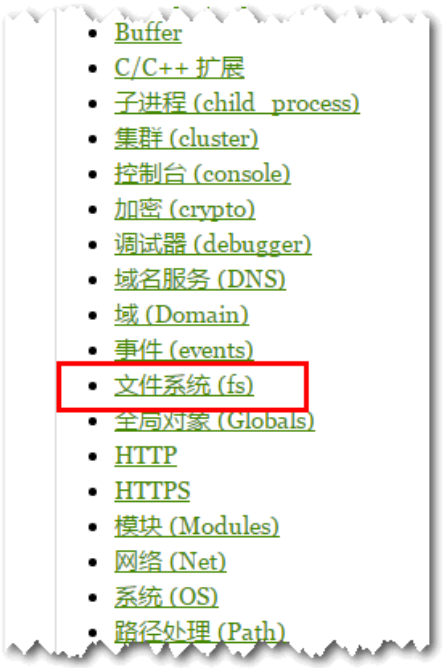
前端这东西，有个博尔特都认同的特点，就是上手快！

首先，我们需要一份Node.js API文档，我们使用“动物搜索”，搜一下：



就第一个吧，进入会看到一长排的API列表内容：

不要怕，我们只需要这一个就可以，没错，就一个文件系统(fs)！其他都不需要管，那些都是资深玩家玩的：



点击去，又是洋洋洒洒，一群API：

不要怕，我们只需要……淡定，不是一个，是若干个常规的增删读写重命名文件就可以了。

好了，然后只需要一点蹩脚的JS，我们就可以玩起来了。

玩什么呢？容我看集动漫想一想……

设计师给的图标重命名

勤劳的设计师送来了香饽饽的小图片素材，但是，连接字符是下划线`_`，恰巧，此时，前端童鞋的处女病发错，其他自己处理的图片全部是短横线`-`连接的，这里图标全是下划线受不了，想要全部替换为短横线，怎么办？

如果就一两个图标还好，大不了手动改改，但是，要是如上截图，设计师一口气给了57个图标，我去，要改到头皮发麻了吧~倒不是时间问题，而是重复劳动带来的那种枯燥和不愉悦会影响工作的激情，而且这种劳动用完就没了，无法复用，且不能作为业绩（我可以5分钟完成100个文件的重命名，有个卵用~）。

此时，Node.js就可以闪亮登场了，有了Node.js环境，我们只要寥寥几行JS代码，就可以完全秒杀了，很简单，读取文件夹里面的所有图片，然后把名称里面所有的下划线`_`替换成短横线`-`，假设我们的`.js`文件和需要处理的小图标文件夹结构如下：

underscore2dash.js内容如下：

```
// 引入fs文件处理模块
var fs = require("fs");
// 现在我们要关心的是'icons'文件夹
// 我们不妨用变量表示这个文件夹名称，方便日后维护和管理
var src = 'icons';

// API文档中中找到遍历文件夹的API
```

```
// 找到了，是fs.readdir(path, callback)
// 文档中有叙述：
// 读取 path 路径所在目录的内容。 回调函数 (callback) 接受两个参数 (err, files) 其中 files 是一个存储目录中所包含的文件名称的数组
// 因此：
fs.readdir(src, function(err, files) {
    // files是名称数组，因此
    // 可以使用forEach遍历哈，此处为ES5 JS一点知识
    // 如果不清楚，也可以使用for循环哈
    files.forEach(function(filename) {
        // 下面就是文件名称重命名
        // API文档中找到重命名的API，如下
        // fs.rename(oldPath, newPath, callback)
        // 下面，我们就可以依葫芦画瓢，确定新旧文件名称：
        var oldPath = src + '/' + filename, newPath = src + '/' + filename.replace(/_/g, '-');
        // 重命名走起
        fs.rename(oldPath, newPath, function(err) {
            if (!err) {
                console.log(filename + '下划线替换成功!');
            }
        })
    });
});
```

window系统举例，我们使用cmd或者PowerShell，在对应文件夹目录下执行下该JS文件：

```
node underscore2dash
```

结果：

此时的文件夹的图片们：

此处的文件名批量替换不仅适用于图片，实际上适用于任意格式的文件。

当前，对命名的批量处理不仅仅如此，还包括统一前缀（例如`icon_*`），此时只要把`newPath =`后满的代码改成`src + '/icon_' + filename`。或者非开发需求，比方说批量下载的小视频名称从1依次往后排，则……还是自己处理下吧，`forEach`方法第二个参数是数组序号值，可以直接拿来用，就当课后作业了，看好你哟！

本文件夹批量处理例子，抛开详尽的注释，差不多10行出头JS代码，用到的JS方法也都是非常非常基本的，对吧，数组遍历forEach和字符替换`replace`方法，其他就是套API走套路，就算我老婆（非IT领域）亲自上阵，也都可以弄出来。简单，而且有意思。

我强烈建议大学的程序开发入门课程就学JavaScript，跑web网页，跑Node.js, 简单且所见即所得，容易激发学习的乐趣，要比枯燥不知干嘛用的C语言更适合科普和入门。

三、蹩脚JS下的Node.js初体验二周目

我们写页面实际的开发需求肯定不知文件批量重命名这么简单，我知道有一个需求点，尤其经常写静态原型页面的小伙伴一定感兴趣的。

就是HTML页面也能够如动态语言，如php一样，各个模块可以直接`include`进来。现在普遍存在这样

一个问题，某项目，重构人员哗啦啦编写了20多个静态页面，但是，由于HTML无法直接include公用的头部底部和侧边栏，导致，这20个页面的头尾都是独立的，一般头部内容发生了变更，呵呵，估计就要求助编辑器来个批量替换什么的了。

这是不是痛点？显然是！凡事痛点都是可以做出贡献体现自己价值的地方。

没错，我们工作就是切切页面，我们的JS勉强可以扶上墙，但，就是这样的我们，只要你有这个心，意识到问题所在，同时知道Node.js可以帮你做到这一点，一个实用的工具其实已经完成了一半。参照API文档，东拼拼，西凑凑，肯定可以弄出一个至少自己用得很high的东西，剩下的一半就这么简简单单续上了。

实例示例示意

有一个原始的HTML页面，头尾都使用了类似下面代码的标准HTML5 `import` 导入：

```
<link rel="import" href="header.html">
```

但是，实际上，`rel="import"` 和 `include` 是完全不一样的概念，`import` 进来实际上是个独立的 `document` ！不过这是后话了，反正我们又不是直接浏览这个页面，因此，大家不必在意这个细节。

HTML几个文件结构关系如下示意：

此时，我们就可以借助Node.js以及我们那一点点JS知识，把 `rel="import"` 这行HTML替换成对应的导入的HTML页面内容。

原理其实很简单：

1. 读import-example.html页面；
2. `href="header.html"` 这行HTML替换成 `header.html` 的内容；
3. 监控import-example.html页面，一有变化，即时生成；
4. done!

下面为本例子的JS代码import.js：

```
// 引入fs文件处理模块
var fs = require("fs");

// 测试用的HTML页文件夹地址和文件名称
var src = 'import', filename = 'import-example.html';

var fnImportExample = function(src, filename) {
    // 读取HTML页面数据
    // 使用API文档中的fs.readFile(filename, [options], callback)
    fs.readFile(src + '/' + filename, {
        // 需要指定编码方式，否则返回原生buffer
        encoding: 'utf8'
    }, function(err, data) {
        // 下面要做的事情就是把
        // <link rel="import" href="header.html">
        // 这段HTML替换成href文件中的内容
        // 可以求助万能的正则
        var dataReplace = data.replace(/<link\srel="import"\shref="(.*)">/gi, function(mat
chs, m1) {
            // m1就是匹配的路径地址了
```

```
            // 然后就可以读文件了
            return fs.readFileSync(src + '/' + m1, {
                encoding: 'utf8'
            });
        });

        // 由于我们要把文件放在更上一级目录，因此，一些相对地址要处理下
        // 在本例子中，就比较简单，对../进行替换
        dataReplace = dataReplace.replace(/"\.\.\//g, '"');

        // 于是生成新的HTML文件
        // 文档找一找，发现了fs.writeFile(filename, data, [options], callback)
        fs.writeFile(filename, dataReplace, {
            encoding: 'utf8'
        }, function(err) {
            if (err) throw err;
            console.log(filename + '生成成功！');
        });
    });
};

// 默认先执行一次
fnImportExample(src, filename);

// 监控文件，变更后重新生成
fs.watch(src + '/' + filename, function(event, filename) {
    if (event == 'change') {
        console.log(src + '/' + filename + '发生了改变，重新生成...');
        fnImportExample(src, filename);
    }
});
```

我们此时node run一下：

```
node import
```

结果：

此时的文件夹：

箭头所指就是新生成的HTML页面，此时的内容是：

我们访问此页面：

连广告都显示良好！

此时，node实际上是实时监控原始HTML是否发生变化的，文档中的`fs.watch()`方法，例如，我们把图片地址的mm1换成mm2，则：

此时页面变成了：

于是乎，一个随时自动编译import导入HTML页面的小工具的雏形就好了。

页面重构的小伙伴，就不要担心20多个原型页面公用部分修改一次要改20多处的问题了，直接将公用的模块import进来，20多个页面分分钟编译为HTML页面完全体。

现在，我们再回过头看上面的HTML支持模块引入的小工具，就是几个简单的Node.js API和几行简单的JS. 我们又不是开源就自己用用，很多复杂场景根本就不要去考虑，所以，这么简单就足够了！

四、结束语

当项目比较小的时候，当团队成员比较少的时候，当开发同学不鸟你的时候，此时，要发扬自己动手，丰衣足食的精神。

开发时候遇到痛点，或者感觉自己在做重复劳动的时候，想想看，是不是可以花点时间捣腾出一个Node.js的小脚本。

不要以为npm仓库里面的那些工具好像很Diao很难搞，其实呢，也就是一点点核心加上应付各种场景弄出来的。由于我们是自娱自乐，追求的是敏捷高效，专注于眼前任务功能，所以，我们只要把核心弄出来就好，而这些核心往往就几行JS代码+几个fs API就可以了。

蚂蚁虽小，咬人也疼。所以，不要觉得自己JS比较菜，搞不来，就几行JS代码，你不动手搞一搞你怎么就确定呢？

写CSS为主的前端想要往后发展，没有比本文介绍内容更适合学习和入门的了。

Try it!

> 《CSS世界》签名版独家发售，包邮，可指定寄语，点击显示购买码

（本篇完） // 想要打赏？点击这里。有话要说？点击这里。

« 基于用户行为的图片等资源预加载

基于clip-path的任意元素的碎片拼接动效 »

猜你喜欢

- 使用electron构建跨平台Node.js桌面应用经验分享
- 博闻强识：了解CSS中的@ AT规则
- 万岁，浏览器原生支持ES6 export和import模块啦！
- jQuery之replace字符串替换实现不同尺寸图片切换
- JavaScript实现http地址自动检测并添加URL链接
- 翻编-JavaScript有关的10个怪癖和秘密
- 小tip：我是如何初体验uglifyjs压缩JS的
- 我是如何实现electron的在线升级热更新功能的？
- Stylus-NodeJS下构建更富表现力/动态/健壮的CSS
- 高富帅seajs使用示例及spm合并压缩工具露脸
- windows系统下批量删除OS X系统.DS_Store文件

分享到： 1

标签： @import, imports, node, nodejs, replace

发表评论（目前73条评论）

名称 (必须)

邮件地址(不会被公开) (必须)

网站

提交评论

« 先前评论

1. 梁秋生说道：
2018年06月25日 08:56

大神，你不能一时写Javascript，一时就写JavaScript，不统一，强迫症的人会很难受的。

回复

2. **datouxia说道：**
2018年04月30日 15:56

没用过这个，但我试过用Apache的一个名叫服务器包含（ssi）的东东，非常类似于php的include的使用方式，最终表现跟张老师的也差不多，就是没有实时更新（不过我另外通过一个超级简单（开始部署也不简单的，不过部署完就超eeeeeasy了）方法解决了实时刷新http://www.browsersync.cn/，就是这个网站（之前试过livereload插件的，但是最终因为翻墙和sublime安装等等问题放弃了））。就是这个ssi配置的话需要稍微一点apache的知识，不过这个完成可以找教程搞定，就是好像会有一个问题，每次新建一个站点（通过自定义域名，使用apche访问的，不是使用127.0.0.1那种方式）会删除掉之前自己开启ssi那个代码。所以这个方法不如张老师这个好用

回复

3. **eowyn说道：**
2018年04月27日 18:07

这文分明就是讲给我听的，受教受教，茅塞顿开啊，突然觉得node好神奇

回复

4. **lixingyang说道：**
2017年03月16日 15:56

感谢科普，说的明白清楚，

回复

5. 惶惶不可终日说道：
2017年01月5日 22:54

张老师的博客太实用了

回复

6. loki说道：
2016年12月2日 11:15

如果没有你，我可能已经放弃重构了。

回复

7. Leo说道：
2016年11月2日 14:01

其实。。我想说。。MAC的OS系统自带满足 重命名那个需求

回复

8. PHP程序员雷雪松说道：
2016年09月28日 09:57

厉害，很实用的工作技巧！！！

回复

9. YJH说道：
2016年09月7日 00:14

来过~

回复

10. 博主肯定是被开发同学嫌弃过说道：
2016年08月6日 09:45

博主肯定是被开发同学嫌弃过

回复

11. xiao鱼仔说道：
2016年08月3日 15:27

写CSS为主的前端想要往后发展

回复

> xiao鱼仔说道：
> 2016年08月3日 15:29
>
> 艾玛我话还没说完就提交评论了。
>
> 此文太适合入门
>
> 回复

> hahahah说道：
> 2016年09月14日 17:41
>
> 最好的入门就是自己亲手去玩玩，nodejs入门资料那么多。。。
> 自己照着随便动动手，1，2小时学会搭个简单的webserver玩玩不就入门了，
> 楼主发文启发读者是好事，不过读者如果指望看看文章就入门，那这个门就真是难入了
>
> 应用技术入门都不难，自己动手玩玩，一下午不管啥技术都入门了
>
> 回复

12. 哈哈哈说道：
2016年08月3日 09:53

无论做什么工作，学习作者这样的处理工作的思路都是有益的，既然重复性的工作会影响到工作的激情，为什么不设法去改造这种流程呢，发明创造 应该也是这样来的。应该好好跟作者学习这种发现问题解决问题的精神，态度！

回复

13. 花花花说道：
2016年08月1日 18:02

膜拜博主

回复

14. MM说道：
2016年07月29日 17:58

说真的，我们这些天天拜读你帖子的人其实应该给你颁发一个奖项，叫互联网雷锋奖，哈哈

回复

15. KK说道：
2016年07月25日 15:00

再也找不到这么好玩的科普了。

回复

16. 业界大牛说道：
2016年07月18日 17:09

我用c 沙埔

回复

17. laraxia说道：
2016年07月15日 17:44

真是实用。。。。。博主太伟大！！！！！！！！！！膜拜~~~~

回复

18. 我要给你生猴子说道：
2016年07月14日 15:38

那个文件重命名的例子，真是相见恨晚啊！！！！！！

回复

« 先前评论

最新文章
» 常见的CSS图形绘制合集
» 粉丝群第1期CSS小测点评与答疑
» 分享三个纯CSS实现26个英文字母的案例
» 小tips: 纯CSS实现打字动画效果
» CSS/CSS3 box-decoration-break属性简介
» CSS :placeholder-shown伪类实现Material Design占位符交互效果
» 从天猫某活动视频不必要的3次请求说起

» CSS vector-effect与SVG stroke描边缩放
» CSS ::backdrop伪元素是干嘛用的?
» 周知：CSS -webkit-伪元素选择器不再导致整行无效

今日热门

» 常见的CSS图形绘制合集 (190)
» 未来必热：SVG Sprite技术介绍 (119)
» 粉丝群第1期CSS小测点评与答疑 (115)
» HTML5终极备忘大全（图片版+文字版） (93)
» 让所有浏览器支持HTML5 video视频标签 (86)
» Selectivizr-让IE6~8支持CSS3伪类和属性选择器 (82)
» CSS3下的147个颜色名称及对应颜色值 (79)
» 视区相关单位vw, vh..简介以及可实际应用场景 (76)
» 写给自己看的display: flex布局教程 (76)
» 小tips: 纯CSS实现打字动画效果 (76)

今年热议

» 《CSS世界》女主角诚寻靠谱一起奋斗之人 (76)
» 不借助Echarts等图形框架原生JS快速实现折线图效果 (64)
» 看，for..in和for..of在那里吵架！ (60)
» 是时候好好安利下LuLu UI框架了！ (47)
» 原来浏览器原生支持JS Base64编码解码 (35)
» 妙法攻略：渐变虚框及边框滚动动画的纯CSS实现 (33)
» 炫酷H5中序列图片视频化播放的高性能实现 (31)
» CSS scroll-behavior和JS scrollIntoView让页面滚动平滑 (30)
» windows系统下批量删除OS X系统.DS_Store文件 (26)
» 写给自己看的display: flex布局教程 (26)

猜你喜欢

- 使用electron构建跨平台Node.js桌面应用经验分享
- 博闻强识：了解CSS中的@ AT规则
- 万岁，浏览器原生支持ES6 export和import模块啦！
- jQuery之replace字符串替换实现不同尺寸图片切换
- JavaScript实现http地址自动检测并添加URL链接
- 翻编-JavaScript有关的10个怪癖和秘密
- 小tip：我是如何初体验uglifyjs压缩JS的
- 我是如何实现electron的在线升级热更新功能的?
- Stylus-NodeJS下构建更富表现力/动态/健壮的CSS
- 高富帅seajs使用示例及spm合并压缩工具露脸
- windows系统下批量删除OS X系统.DS_Store文件