Vue.js与Webpack

1 vue.js研究

1.1 vue.js介绍

1、vue.js是什么？

Vue (读音 /vjuː/，类似于 **view**) 是一套用于构建用户界面的**渐进式框架**。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。

渐进式框架：Progressive，说明vue.js的轻量，是指一个前端项目可以使用vue.js一两个特性也可以整个项目都用vue.js。

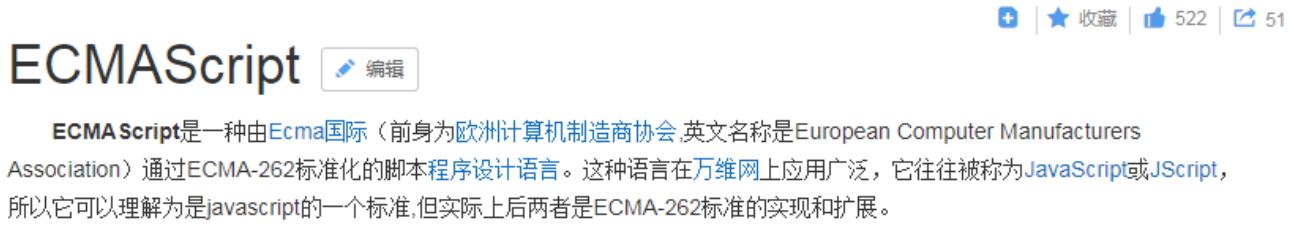
自底向上逐层应用：作为渐进式框架要实现的目标就是方便项目增量开发。

参考：https://cn.vuejs.org/v2/guide/

2、Vue.js与ECMAScript

Vue **不支持** IE8 及以下版本，因为 Vue 使用了 IE8 无法模拟的 ECMAScript 5 特性。

什么是ECMAScript?



ECMAScript（简称ES）是一种规范，我们平常所说的Js/Javascript是ECMAScript的实现，早期主要应用的ES3，当前主流浏览器都支持ES5、ES6，ES8已于2017年发布。

ES6：http://www.ecma-international.org/ecma-262/6.0/

ES7：http://www.ecma-international.org/ecma-262/7.0/

3、Vue.js的使用

1）在html页面使用script引入vue.js的库即可使用。

2）使用Npm管理依赖，使用webpack打包工具对vue.js应用打包。

大型应用推荐此方案。

3）Vue-CLI脚手架
使用vue.js官方提供的CLI脚本架很方便去创建vue.js工程雏形。

4、vue.js有哪些功能？

1）声明式渲染

Vue.js 的核心是一个允许采用简洁的模板语法来声明式地将数据渲染进 DOM 的系统。

比如：使用vue.js的插值表达式放在Dom的任意地方， 差值表达式的值将被渲染在Dom中。

2）条件与循环

dom中可以使用vue.js提供的v-if、v-for等标签，方便对数据进行判断、循环。

3）双向数据绑定

Vue 提供v-model 指令，它可以轻松实现Dom元素和数据对象之间双向绑定，即修改Dom元素中的值自动修改绑定的数据对象，修改数据对象的值自动修改Dom元素中的值。

4）处理用户输入

为了让用户和你的应用进行交互，我们可以用 `v-on` 指令添加一个事件监听器，通过它调用在 Vue 实例中定义的方法

5）组件化应用构建

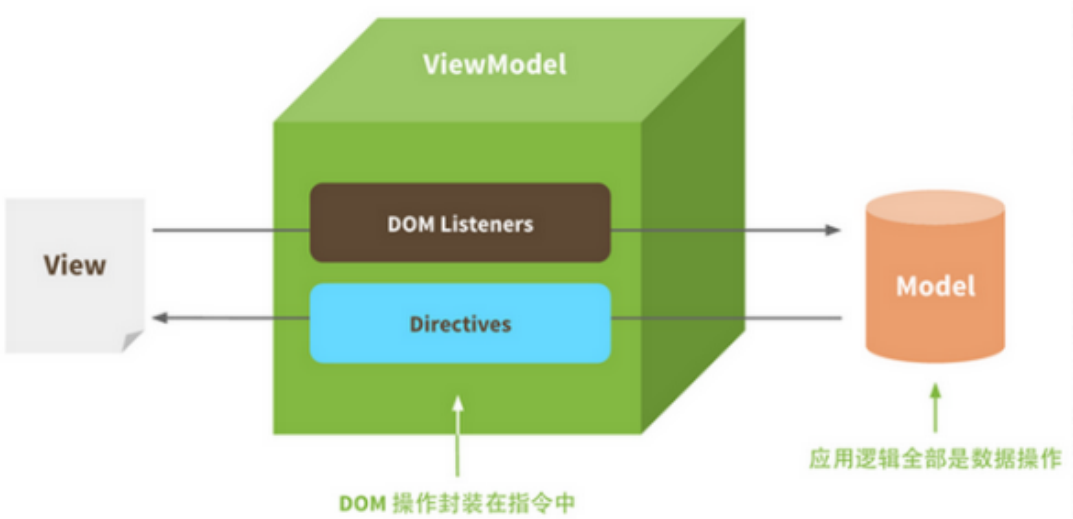
vue.js可以定义一个一个的组件，在vue页面中引用组件，这个功能非常适合构建大型应用。

1.2 vue.js基础

1.2.1 MVVM模式

vue.js是一个MVVM的框架，理解MVVM有利于学习vue.js。

- MVVM拆分解释为：
  - Model:负责数据存储
  - View:负责页面展示
  - View Model:负责业务逻辑处理（比如Ajax请求等），对数据进行加工后交给视图展示
- MVVM要解决的问题是将业务逻辑代码与视图代码进行完全分离，使各自的职责更加清晰，后期代码维护更加简单
- 用图解的形式分析Ajax请求回来数据后直接操作Dom来达到视图的更新的缺点，以及使用MVVM模式是如何来解决这个缺点的
- Vue中的 MVVM



从上图看出，VM(ViewModel)可以把view视图和Model模型解耦合，VM的要做的工作就是vue.js所承担的。

1.2.2 入门程序

本次测试我们在门户目录中创建一个html页面进行测试，正式的页面管理前端程序会单独创建工程。

在门户目录中创建vuetest目录，并且在目录下创建vue_01.html文件

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>vue.js入门程序</title>
    <script src="/js/vue/vue.min.js"></script>
</head>
<body>
<div id="app">
{{name}}
    <!-- 在Vue接管区域中使用Vue的系统指令呈现数据
    这些指令就相当于是MVVM中的View这个角色 -->
</div>
</body>
<script>
    // 实例化Vue对象
    //vm :叫做MVVM中的 View Model
    var VM = new Vue({
        el:"#app",//表示当前vue对象接管app的div区域
        data:{
            name:'传智播客'// 相当于是MVVM中的Model这个角色
        }

    });

</script>

</html>
```

代码编写步骤：

1、定义html，引入vue.js

2、定义app div，此区域作为vue的接管区域

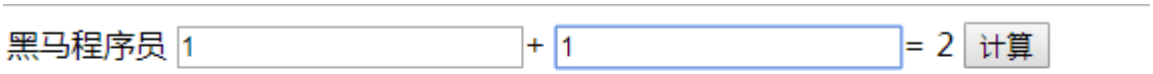
3、定义vue实例，接管app区域。

4、定义model（数据对象）

5、VM完成在app中展示数据

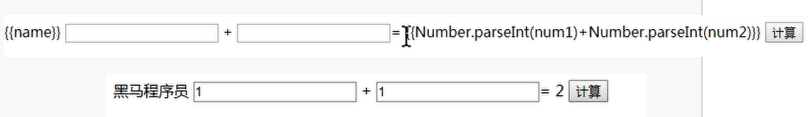
1.2.3 1+1=2

实现效果：



代码如下：

```
<!DOCTYPE html>
<html lang="en" xmlns:v-on="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="UTF-8">
    <title>vue.js入门程序</title>

</head>
<body>
<div id="app">
    <!--{{name}}解决闪烁问题使用v-text-->
<a v-bind:href="url"><span v-text="name"></span></a>
<input type="text" v-model="num1">+
<input type="text" v-model="num2">=
<span v-text="result"></span>
   <!-- <span v-text="Number.parseInt(num1)+Number.parseInt(num2)"></span>-->
    <!--{{num1+num2}}-->
<!--<input type="text" v-model="result">-->
    <button v-on:click="change">计算</button>
    <!-- 在Vue接管区域中使用Vue的系统指令呈现数据
    这些指令就相当于是MVVM中的View这个角色 -->
</div>
</body>
<script src="/js/vue/vue.min.js"></script>
<script>
    // 实例化Vue对象

    //vm :叫做MVVM中的 View Model
```



```
    var VM = new Vue({
        el:"#app",//表示当前vue对象接管app的div区域
        data:{
            name:'黑马程序员',// 相当于是MVVM中的Model这个角色
            num1:0,
            num2:0,
            result:0,
            url:'http://www.itcast.cn'
        },
        methods:{
            change:function(){
                this.result = Number.parseInt(this.num1)+Number.parseInt(this.num2)
                alert(this.result)
            }
        }

    });

</script>
</html>
```

本例子学习了：

1、v-model： 输入功能

1、在表单控件或者组件上创建双向绑定 2、v-model仅能在如下元素中使用：

```
input
select
textarea
components（Vue中的组件）
```

2、解决插值表达式闪烁问题，使用v-text 网速越慢也明显闪烁

v-text可以将一个变量的值渲染到指定的元素中,它可以解决插值表达式闪烁的问题

3、v-on绑定一个按钮的单击事件

4、v-bind

```
1、作用：
    v-bind可以将数据对象绑定在dom的任意属性中。
    v-bind可以给dom对象绑定一个或多个特性，例如动态绑定style和class

  2、举例：
      <img v-bind:src="imageSrc">
      <div v-bind:style="{ fontSize: size + 'px' }"></div>

  3、缩写形式
      <img :src="imageSrc">
      <div :style="{ fontSize: size + 'px' }"></div>
```

1.2.4 v-if和v-for

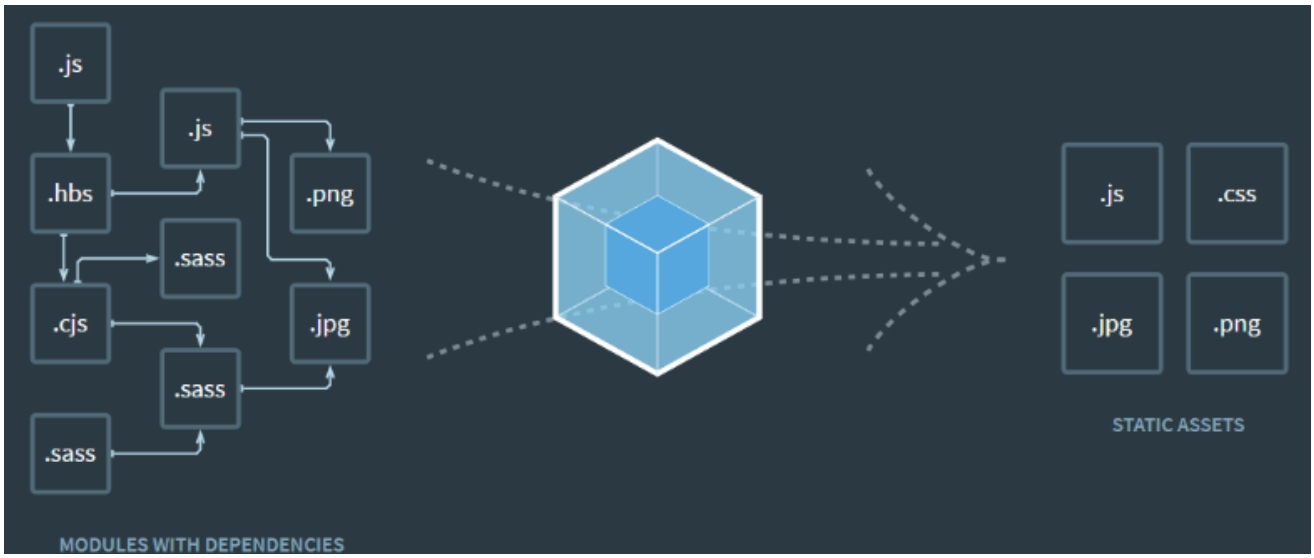
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="/js/vue/vue.min.js"></script>
</head>
<body>
<div id="app">
    <ul>
        <!--只显示偶数行-->
        <li v-for="(item,index) in list" :key="index" v-if="index % 2==0">{{index}}-{{item}}
</li>                                                  下标，可以省略
        <li v-for ="(value,key) in user">{{key}}-{{value}}</li>
        <li v-for="(item,index) in userlist" :key="item.user.uname">
            <div v-if="item.user.uname=='heima'" style="background: chartreuse"><!--名称为heima的
加背景色-->
                {{index}}-{{item.user.uname}}-{{item.user.age}}
            </div>
            <div v-else="">
                {{index}}-{{item.user.uname}}-{{item.user.age}}
            </div>
        </li>
    </ul>
</div>
</body>

<script>
    new Vue({
        el:'#app',
        data:{
            list:[1,2,3,4,4],
            user:{uname:'itcast',age:10},
            userlist:[
                { user:{uname:'itcast',age:10}},
                { user:{uname:'heima',age:11}}
            ]
        }
    });
</script>
</html>
```

1.3 webpack入门

使用vue.js开发大型应用需要使用webpack打包工具，本节研究webpack的使用方法。

1.3.1 webpack介绍

Webpack 是一个前端资源的打包工具，它可以将js、image、css等资源当成一个模块进行打包。



从图中我们可以看出，Webpack 可以将js、css、png等多种静态资源 进行打包，使用webpack有什么好处呢？

1、模块化开发

程序员在开发时可以分模块创建不同的js、 css等小文件方便开发，最后使用webpack将这些小文件打包成一个文件，减少了http的请求次数。

webpack可以实现按需打包，为了避免出现打包文件过大可以打包成多个文件。

2、 编译typescript、ES6等高级js语法

随着前端技术的强大，开发中可以使用javascript的很多高级版本，比如：typescript、ES6等，方便开发，webpack可以将打包文件转换成浏览器可识别的js语法。

3、CSS预编译

webpack允许在开发中使用Sass 和 Less等原生CSS的扩展技术，通过sass-loader、less-loader将Sass 和 Less的语法编译成浏览器可识别的css语法。

webpack的缺点：

1、配置有些繁琐

2、文档不丰富

1.3.2 安装webpack

1.3.2.1 安装Node.js

webpack基于node.js运行，首先需要安装node.js。

收藏 1349 175

node.js 编辑

和jre差不多

Node.js是一个Javascript运行环境(runtime environment)，发布于2009年5月，由Ryan Dahl开发，实质是对Chrome V8引擎进行了封装。Node.js对一些特殊用例进行优化，提供替代的API，使得V8在非浏览器环境下运行得更好。

V8引擎执行Javascript的速度非常快，性能非常好。[1] Node.js是一个基于Chrome JavaScript运行时建立的平台， 用于方便地搭建响应速度快、易于扩展的网络应用。Node.js 使用事件驱动， 非阻塞I/O 模型而得以轻量和高效，非常适合在分布式设备上运行数据密集型的实时应用。

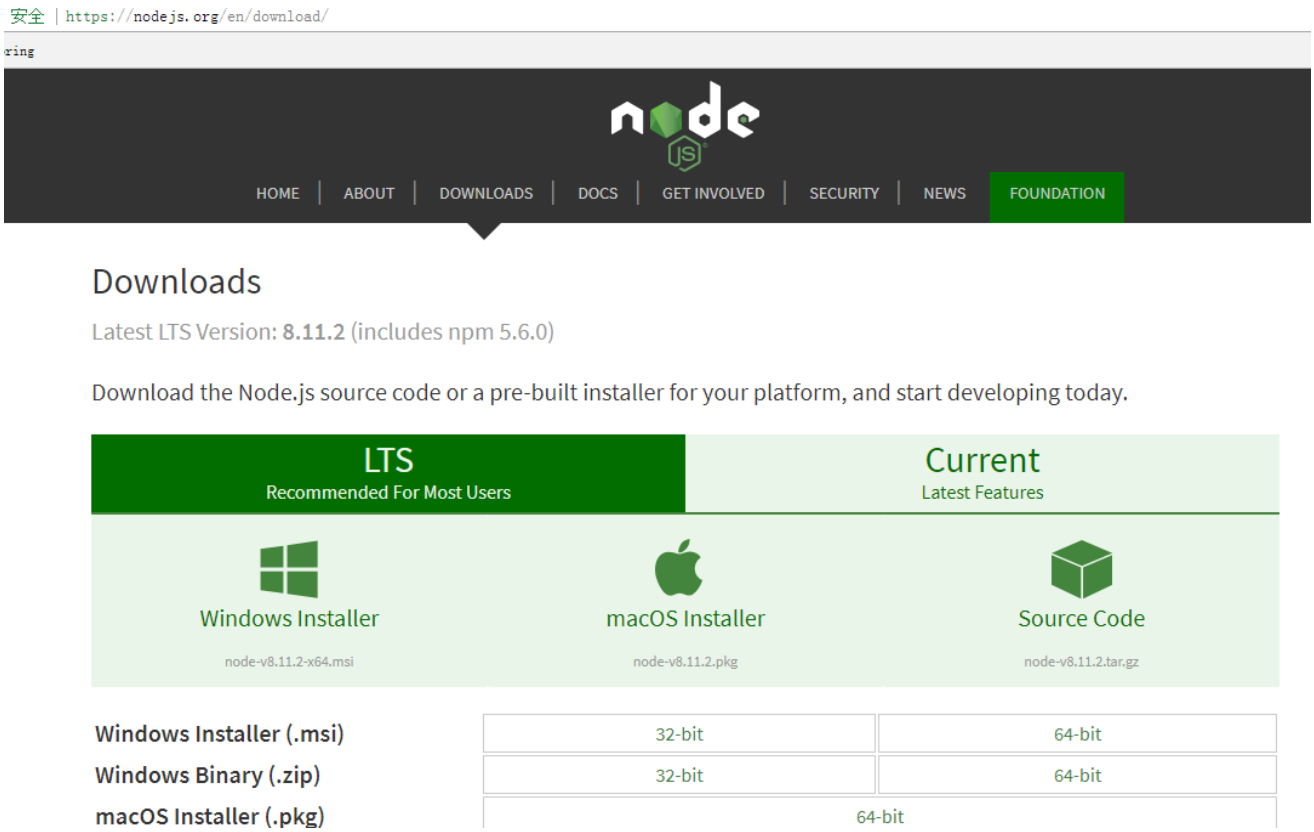
为什么会有node.js？

传统意义上的 JavaScript 运行在浏览器上，Chrome 使用的 JavaScript 引擎是 V8，Node.js 是一个运行在服务端的框架，它的底层就使用了 V8 引擎，这样就可以使用javascript去编写一些服务端的程序，这样也就实现了用 javaScript去开发 Apache + PHP 以及 Java Servlet所开发的服务端功能，这样做的好处就是前端和后端都采用 javascript，即开发一份js程序即可以运行在前端也可以运行的服务端，这样比一个应用使用多种语言在开发效率上要高，不过node.js属于新兴产品，一些公司也在尝试使用node.js完成一些业务领域，node.js基于V8引擎，基于事件驱动机制，在特定领域性能出色，比如用node.js实现消息推送、状态监控等的业务功能非常合适。

下边我们去安装Node.js：

1、下载对应你系统的Node.js版本:

https://nodejs.org/en/download/



推荐下载LTS版本，本教程安装9.4.0。

2、选安装目录进行安装

默认即可

安装完成检查PATH环境变量是否设置了node.js的路径。

3、测试

在命令提示符下输入命令

```
node -v
```

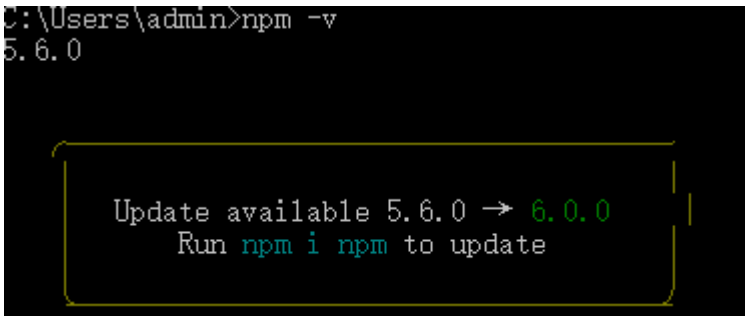
会显示当前node的版本

1.3.2.2 安装NPM

1、自动安装NPM

npm全称Node Package Manager，他是node包管理和分发的工具，使用NPM可以对应用的依赖进行管理，NPM的功能和服务端项目构建工具maven差不多，我们通过npm 可以很方便地下载js库，打包js文件。

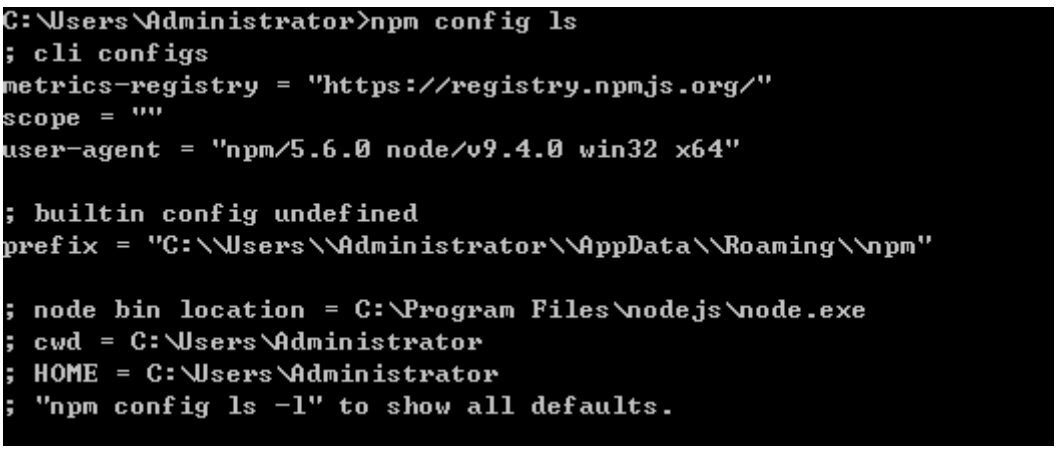
node.js已经集成了npm工具，在命令提示符输入 npm -v 可查看当前npm版本



2、设置包路径

包路径就是npm从远程下载的js包所存放的路径。

使用 npm config ls 查询NPM管理包路径（NPM下载的依赖包所存放的路径）



换一个地方

NPM默认的管理包路径在C:/用户/[用户名]/AppData/Roming/npm/node_meodules，为了方便对依赖包管理，我们将管理包的路径设置在单独的地方，本教程将安装目录设置在node.js的目录下，创建npm_modules和npm_cache，执行下边的命令：

本教程安装node.js在D:\Program Files\nodejs下所以执行命令如下：

npm config set prefix "C:\Program Files\nodejs\npm_modules"

npm config set cache "c:\Program Files\nodejs\npm_cache"

此时再使用 npm config ls 查询NPM管理包路径发现路径已更改

```
C:\Users\Administrator>npm config ls
; cli configs
metrics-registry = "https://registry.npmjs.org/"
scope = ""
user-agent = "npm/5.6.0 node/v9.4.0 win32 x64"

; userconfig C:\Users\Administrator\.npmrc
cache = "c:\\Program Files\\nodejs\\npm_cache"
prefix = "c:\\Program Files\\nodejs\\npm_modules"

; builtin config undefined

; node bin location = C:\Program Files\nodejs\node.exe
; cwd = C:\Users\Administrator
; HOME = C:\Users\Administrator
; "npm config ls -l" to show all defaults.
```

3、安装cnpm

npm默认会去国外的镜像去下载js包，在开发中通常我们使用国内镜像，这里我们使用淘宝镜像

下边我们来安装cnpm：

有时我们使用npm下载资源会很慢，所以我们可以安装一个cnmp(淘宝镜像)来加快下载速度。

输入命令，进行全局安装淘宝镜像。

npm install -g cnpm --registry=https://registry.npm.taobao.org

安装后，我们可以使用以下命令来查看cnpm的版本

cnpm -v

```
C:\Users\Administrator>cnpm -v
cnpm@6.0.0 (D:\Program Files\nodejs\npm_modules\node_modules\cnpm\lib\parse_argv.js)
npm@6.1.0 (D:\Program Files\nodejs\npm_modules\node_modules\cnpm\node_modules\npm\lib\npm.js)
node@9.4.0 (D:\Program Files\nodejs\node.exe)
npminstall@3.6.2 (D:\Program Files\nodejs\npm_modules\node_modules\cnpm\node_modules\npminstall\lib\index.js)
prefix=D:\Program Files\nodejs\npm_modules
win32 x64 6.1.7601
registry=https://registry.npm.taobao.org
```

nrm ls 查看镜像已经指向taobao

```
C:\Users\Administrator>nrm ls

  npm ----- https://registry.npmjs.org/
  cnpm --- http://r.cnpmjs.org/
* taobao - https://registry.npm.taobao.org/
  nj ----- https://registry.nodejitsu.com/
  rednpm - http://registry.mirror.cqupt.edu.cn/
  npmMirror  https://skimdb.npmjs.com/registry/
  edunpm - http://registry.enpmjs.org/
```

使nrm use XXX切换 镜像

如果nrm没有安装则需要进行全局安装：cnpm install -g nrm

4、非连网环境安装cnpm

从本小节第3步开始就需要连网下载npm包，如果你的环境不能连网在老师的资料文件下有已经下载好的webpack相关包，下边是安装方法。

1）配置环境变量

NODE_HOME = C:\Program Files\node js (node.js安装目录)

在PATH变量中添加：%NODE_HOME%; %NODE_HOME%\npm_modules;

node.js本身的命令位置

下载的全局node模块及其命令位置
node.js根目录还有一个nodemodules，那是默认一起安装的npm模块

2）找到npm包路径

根据上边的安装说明npm包路径被设置到了node.js安装目录下的npm_modules目录。

可以使用npm config ls查看。

拷贝课程资料中的 npm_modules.zip到node.js安装目录，并解压npm_modules.zip覆盖本目录下的npm_modules文件夹。

3）完成上边步骤测试

cnpm -v

1.3.2.3 安装webpack

1、连网安装

webpack安装分为本地安装和全局安装：

本地安装：仅将webpack安装在当前项目的node_modules目录中，仅对当前项目有效。

全局安装：将webpack安装在本机，对所有项目有效，全局安装会锁定一个webpack版本，该版本可能不适用某个项目。全局安装需要添加 -g 参数。

进入webpacktest测试目录目录，运行：

1）本地安装：

只在我的项目中使用webpack，需要进行本地安装，因为项目和项目所用的webpack的版本不一样。
本地安装就会将webpack的js包下载到项目下的npm_modeuls目录下。

在门户目录下创建webpack测试目录webpacktest01：

npm install --save-dev webpack 或 cnpm install --save-dev webpack

npm install --save-dev webpack-cli (4.0以后的版本需要安装webpack-cli)

2）全局安装加-g，如下：　任意目录下

全局安装就将webpack的js包下载到npm的包路径下。

npm install webpack -g 或 cnpm install webpack -g

3）安装webpack指定的版本：

本教程使用webpack3.6.0，安装webpack3.6.0：

进入webpacktest测试目录，运行：cnpm install --save-dev webpack@3.6.0

全局安装：npm install webpack@3.6.0 -g或 cnpm install webpack@3.6.0 -g

2、非连网安装

参考上边“非连网环境安装cnpm”描述，将课程资料中的 npm_modules.zip到node.js安装目录，并解压npm_modules.zip覆盖本目录下的npm_modules文件夹。

说明：已执行“非连网环境安装cnpm”下的操作不用重复执行。

测试：

在cmd状态输入webpack，出现如下提示说明 webpack安装成功

```
C:\Users\Administrator>webpack
No configuration file found and no output filename configured via CLI option.
A configuration file could be named 'webpack.config.js' in the current directory.
Use --help to display the CLI options.
```

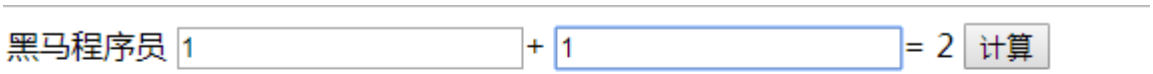
1.3.3 入门程序

通过本入门程序体会webpack打包的过程及模块化开发的思想。

1.3.3.1 需求分析

通过入门程序实现对js文件的打包，体会webpack是如何对应用进行模块化管理。

对上边1+1=2的例子使用webpack进行模块化管理



1.3.3.2 定义模块

创建webpacktest01目录，将vue.min.js及vue_02.html拷贝到目录下。

1、定义model01.js 用于被导出

在webpacktest01目录下创建model01.js

将本程序使用的加法运算的js方法抽取到一个js文件，此文件就是一个模块

只抽取一个文件

或者可以传入整个文件

```
// 定义add函数
function add(x, y) {
    return x + y
}
// function add2(x, y) {
//     return x + y+1
// }
// 导出add方法

module.exports.add = add;
// module.exports ={add,add2};//如果有多个方法这样导出
// module.exports.add2 = add2//如果有多个方法也可以这样导出
```

2、定义main.js 需要引用其他模块

在webpacktest01目录下创建main.js，main.js是本程序的js主文件，包括如下内容：

1、在此文件中会引用model01.js模块

2、引用vue.min.js（它也一个模块）

3、将html页面中构建vue实例的代码放在main.js中。

main.js的代码如下

```
var {add} = require('./model01.js');
var Vue = require('./vue.min');
var VM = new Vue({
    el:"#app",//表示当前vue对象接管app的div区域
    data:{
        name:'黑马程序员',// 相当于是MVVM中的Model这个角色
        num1:0,
        num2:0,
        result:0,
        url:'http://www.itcast.cn'
    },
    methods:{
        change:function(){
            //这里使用了导入的model01.js文件中的add方法
            this.result = add(Number.parseInt(this.num1),Number.parseInt(this.num2))
            alert(this.result)
        }

    }
```

```
});
```

1.3.3.3 打包测试

上边将mode01.js模块及main.js主文件编写完成，下边使用webpack对这些js文件进行打包

1、进入程序目录，执行webpack main.js build.js ，这段指令表示将main.js打包输出为 build.js文件

执行完成，观察程序目录是否出现build.js。

2、在html中引用build.js

```
<!DOCTYPE html>
<html lang="en" xmlns:v-on="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="UTF-8">
    <title>vue.js入门程序</title>
</head>
<body>
<div id="app">
    <!--{{name}}解决闪烁问题使用v-text-->
<a v-bind:href="url"><span v-text="name"></span></a>
<input type="text" v-model="num1">+
<input type="text" v-model="num2">=
<span v-text="result"></span>
    <!--{{num1+num2}}-->
<!--<input type="text" v-model="result">-->
    <button v-on:click="change">计算</button>
    <!-- 在Vue接管区域中使用Vue的系统指令呈现数据
    这些指令就相当于是MVVM中的View这个角色 -->
</div>
</body>
<script src="build.js"></script>
<script>

</script>
</html>
```

3、测试总结

测试功能：

1）输入任意加数，其和会自动计算

2）点击“计算”会调用model01.js中的add方法

总结：

webpack可以将js分模块开发，开发完成对各模块代码打包成一个统一的文件。

前端模块开发的思想和服务端模块开发的思想是一致的，有利于多人协助开发。

1.3.4 webpack-dev-server

webpack-dev-server开发服务器，它的功能可以实现热加载 并且自动刷新浏览器。

创建一个新的程序目录，这里我们创建webpacktest02目录，将webpack入门程序的代码拷贝进来，并在目录下创建src目录、dist目录。

将main.js和model01.js拷贝到src目录。

vue.min.js

1.3.4.1安装配置

1、安装webpack-dev-server

使用 webpack-dev-server需要安装webpack、 webpack-dev-server和 html-webpack-plugin三个包。

cnpm install webpack@3.6.0 webpack-dev-server@2.9.1 html-webpack-plugin@2.30.1 --save-dev

安装完成，会发现程序目录出现一个package.json文件，此文件中记录了程序的依赖。

没有联网的同学拷贝老师提供的node_modules.zip到webpacktest02目录下，解压到node_modules目录下。

2、配置webpack-dev-server

在package.json中配置script

```
  "scripts": {
    "dev": "webpack-dev-server --inline --hot --open --port 5008"
  },
```

--inline：自动刷新

--hot：热加载

--port：指定端口

--open：自动在默认浏览器打开

--host：可以指定服务器的 ip，不指定则为127.0.0.1，如果对外发布则填写公网ip地址

此时package.json的文件内容如下：

```
{
  "scripts": {
    "dev": "webpack-dev-server --inline --hot --open --port 5008"
  },
  "devDependencies": {
    "html-webpack-plugin": "^2.30.1",
    "webpack": "^3.6.0",
    "webpack-dev-server": "^2.9.1"
  }
}
```

devDependencies：开发人员在开发过程中所需要的依赖。

scripts：可执行的命令

1.3.4.2 配置webpack.config.js

在webpacktest02目录下创建 webpack.config.js， webpack.config.js是webpack的配置文件。在此文件中可以配置应用的入口文件、输出配置、插件等，其中要实现热加载自动刷新功能需要配置html-webpack-plugin插件。

html-webpack-plugin的作用是根据html模板在内存生成html文件，它的工作原理是根据模板文件在内存中生成一个index.html文件。

1、配置模板文件

将原来的vue_02.html作为模板文件，为了和内存中的index.html文件名区别，注意将vue_02.html中的script标签去掉，内容如下：

```
<!DOCTYPE html>
<html lang="en" xmlns:v-on="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="UTF-8">
    <title>vue.js入门程序</title>
</head>
<body>
<div id="app">
    <!--{{name}}解决闪烁问题使用v-text-->
<a v-bind:href="url"><span v-text="name"></span></a>
<input type="text" v-model="num1">+
<input type="text" v-model="num2">=
<span v-text="result"></span>
    <!--{{num1+num2}}-->
<!--<input type="text" v-model="result">-->
    <button v-on:click="change">计算</button>
    <!-- 在Vue接管区域中使用Vue的系统指令呈现数据
    这些指令就相当于是MVVM中的View这个角色 -->
</div>
</body>
</html>
```

2、配置 html-webpack-plugin

在webpack.config.js中配置html-webpack-plugin插件

```
var htmlwp = require('html-webpack-plugin');
module.exports={
    entry:'./src/main.js',  //指定打包的入口文件
    output:{
        path : __dirname+'/dist',  // 注意：__dirname表示webpack.config.js所在目录的绝对路径
        filename:'build.js'         //输出文件
    },
    plugins:[
        new htmlwp({
            title: '首页',  //生成的页面标题<head><title>首页</title></head>
            filename: 'index.html', //webpack-dev-server在内存中生成的文件名称，自动将build注入到这个页面底部，才能实现自动刷新功能
            template: 'vue_02.html' //根据index1.html这个模板来生成(这个文件请程序员自己生成)
        })
    ]
}
```

1.3.4.3 启动

启动文件：

1、进入 webpacktest02目录，执行npm run dev

2、使用webstorm，右键package.json文件，选择“Show npm Scripts”

打开窗口：

双击 dev。

注意：dev就是在package.json中配置的webpack dev server命令。

发现启动成功自动打开浏览器。

修改src中的任意文件内容，自动加载并刷新浏览器。

1.3.4.4 debug调试

使用了webpack之后就不能采用传统js的调试方法在chrome中打断点。

webpack将多个源文件打包成一个文件，并且文件的内容产生了很大的变化，webpack提供devtool进行调试，devtool是基于sourcemap的方式，在调试时会生成一个map文件，其内容记录生成文件和源文件的内容映射，即生成文件中的哪个位置对应源文件中的哪个位置，有了sourcemap就可以在调试时看到源代码。

配置如下：

1、在webpack.config.js中配置：

```
devtool: 'eval-source-map',
```

webpack.config.js部分内容如下：

```
var htmlwp = require('html-webpack-plugin');
module.exports={
    entry:'./src/main.js',  //指定打包的入口文件
    output:{
        path : __dirname+'/dist',  // 注意：__dirname表示webpack.config.js所在目录的绝对路径
        filename:'build.js'          //输出文件
    },
    devtool: 'eval-source-map',
    ......
```

2、在js中跟踪代码的位置上添加debugger

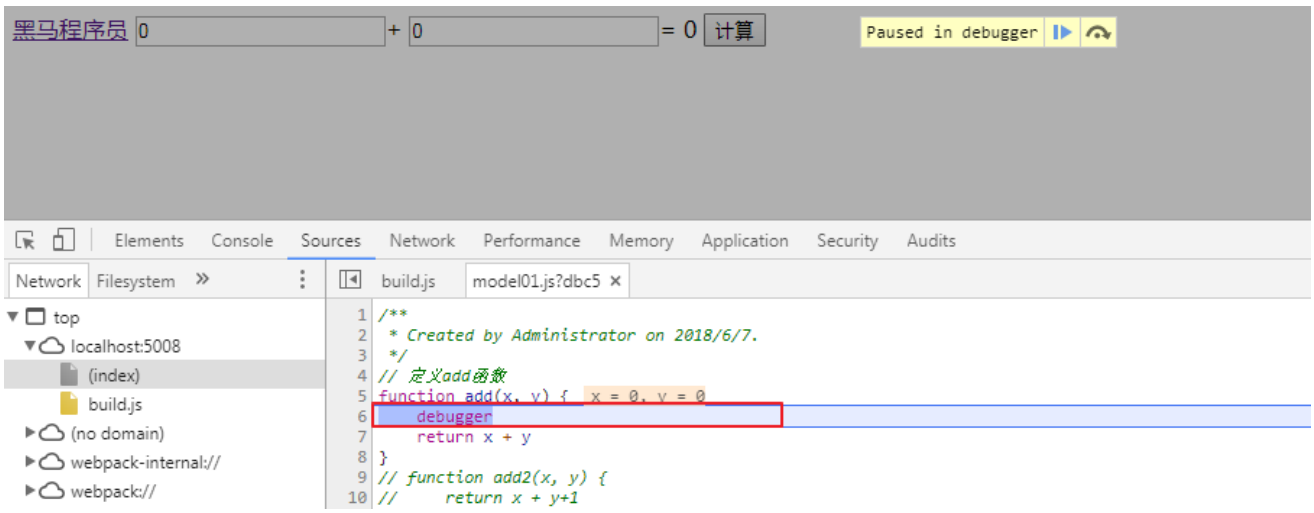
一个例子：

在add方法中添加debugger

```
// 定义add函数
function add(x, y) {
    debugger
    return x + y
}
```

启动应用，刷新页面跟踪代码：

点击"计算" 即进入debugger代码位置，此时可以使用chrome进行调试了。



1、vue.js研究--入门程序

1）编写html，引入vue.min.js（vue.js的类库）

2）编写视图部分的代码（用户界面，只负责展示）

3）编写VM（viewModel）及其中Model

4）刷新页面运行程序，vue.js(VM)部分实现将model中的数据在view中展示

2、vue.js研究-常用指令

1）v-model

实现双向数据绑定：

a、由模型数据绑定到Dom对象，模型数据的值改变，Dom对象的值自动改变

b、由Dom对象绑定到模型数据，Dom对象的值改变，模型数据就改变

2）v-text

v-text可以将一个变量的值渲染到指定的元素中,它可以解决插值表达式闪烁的问题

3) v-on

监听用户事件

4）v-bind

v-bind可以将数据对象绑定在dom的任意属性中。

v-bind可以给dom对象绑定一个或多个特性，例如动态绑定style和class

5) v-if和v-for

v-for可以循环遍历数组、v-if进行判断

2、安装webpack

1) webpack依赖node.js，首先安装node.js 类似jdk

2) 安装npm node.js自动继承

npm全称Node Package Manager，他是node包管理和分发的工具。

node.js使用npm安装我们所依赖的js包。

通过npm安装webpack。

npm的工作原理：去远程下载所依赖的js包。

3）cnpm

cnpm 代替了npm，可以从国内的镜像下载js包。

如果没有连网使用老师提供的npm_modules.zip,解压到npm_modules目录的位置（前边设置了npm_modules的目录在nodejs安装目录下）

4）安装webpack

本地安装: 将webpack安装到指定应用程序的目录下

进入这个目录，执行npm install --save-dev webpack 或 cnpm install --save-dev webpack

全局安装：将webpack安装npm默认的依赖包的目录（前边配置在了nodejs的安装目录下）

在任意目录，执行npm install -g webpack 或 cnpm install -g webpack

建议指定webpack的版本进行安装

全局安装：npm install webpack@3.6.0 -g或 cnpm install webpack@3.6.0 -g

本发安装：cnpm install --save-dev webpack@3.6.0 或

npm install --save-dev webpack@3.6.0

1、webpack入门程序--打包过程及webpack开发过程

1）分模块去定义js

在js中要导出将来要被打包的方法 `module.exports`

2）定义main.js入口文件（主文件）

在此文件，导入引用的js文件

```
var {add} = require("./model01")
```

可以在main.js中使用导入的js方法。

3）使用webpack 命令打包js

4）在html上引用打包生成的js文件

2、推荐使用webpack-dev-server开发服务器，它的功能可以实现热加载 并且自动刷新浏览器。

1）安装webpack-dev-server

cnpm install webpack@3.6.0 webpack-dev-server@2.9.1 html-webpack-plugin@2.30.1 --save-dev

安装完成自动创建package.json（记录了本程序所依赖的包信息）

安装完成自动创建node_modules（存放了本程序所依赖的包）

2）在package.json中配置script（运行命令）

```
"scripts": {
  "dev": "webpack-dev-server --inline --hot --open --
port 5008"
},
```

3）配置webpack.config.js

webpack的配置文件，配置了入口文件、输入文件的路径、依赖的插件。

4）使用webpack的命令去运行程序

npm run dev

5) 使用 debugger进行调试

在要打断点的代码处加debugger

2、创建CMS前端工程结构

前端工程结构基于vue-cli脚手架工具创建的，在基础上进行二次封装

拷贝资料文件夹下的xc-ui-pc-sysmanage.7z到UI工程目录，并进行解压

运行程序：

开发使用：npm run dev

打包使用：npm run build

NRM：NPM源管理工具

Cha0res 关注
0.1 2016.08.16 23:35* 字数 22 阅读 295 评论 0 喜欢 1

安装NRM

```
$ npm install -g nrm --registry=https://registry.npm.taobao.org
```

列出现有源

```
; nrm ls
```

切换到taobao

```
; nrm use taobao
```

添加npm源

```
; nrm add <registry> <url> [home]
```

移除npm源

```
; nrm del <registry>
```

测速

```
; nrm test
```