node搭建环境及webpack打包

## Windows下安装nodejs环境

64 位安装包下载地址 : <http://nodejs.cn/>

本文实例以 v0.10.26 版本为例，其他版本类似， 安装步骤：

步骤 1 : 双击下载后的安装包 **v0.10.26**，如下所示：



步骤 2 : 点击以上的Run(运行)，将出现如下界面：



步骤 3 : 勾选接受协议选项，点击 next（下一步） 按钮 ，一直下一步到一下界面，安装完成



检测PATH环境变量是否配置了Node.js，点击开始=》运行=》输入"cmd" => 输入命令"path"，输出如下结果：

PATH=C:\oraclexe\app\oracle\product\10.2.0\server\bin;C:\Windows\system32;

C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;

c:\python32\python;C:\MinGW\bin;C:\Program Files\GTK2-Runtime\lib;

C:\Program Files\MySQL\MySQL Server 5.5\bin;C:\Program Files\nodejs\;

C:\Users\rg\AppData\Roaming\npm

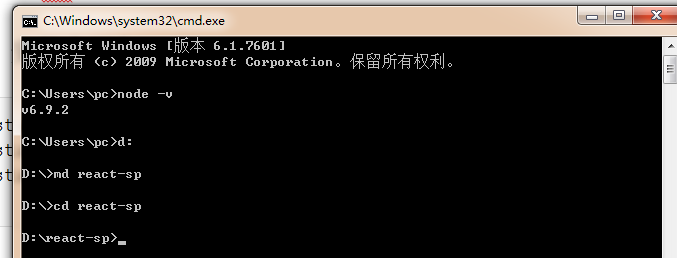
我们可以看到环境变量中已经包含了C:\Program Files\nodejs\

检查Node.js版本

node-version-test

## 通过 npm 使用 React

用管理员权限打开cmd,切换到需要创建项目的路径下，我创建在d盘(随意)



### 第一步、创建根目录

创建一个根目录，目录名为：reactApp，再使用 npm init 初始化，生成 package.json 文件：

md reactApp

cd reactApp/

npm init

name: (reactApp) runoob-react-test

version: (1.0.0)

description: 菜鸟教程 react 测试

entry point: (index.js)

test command:

git repository:

keywords:

author:

license: (ISC)

About to write to /Users/tianqixin/www/reactApp/package.json:

{

"name": "runoob-react-test",

"version": "1.0.0",

"description": "菜鸟教程 react 测试",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC"

}

Is this ok? (yes)

### 第二步、创建文件

接下来我们创建一些必要文件：

cd>index.html

cd>App.jsx

cd>main.js

cd>webpack.config.js

### 第三步、设置编译器，服务器，载入器

打开 **webpack.config.js** 文件添加以下代码:

var config = {

entry: './main.js',

output: {

path:'./',

filename: 'index.js',

},

devServer: {

inline: true,

port: 7777

},

module: {

loaders: [ {

test: /\.jsx?$/,

exclude: /node\_modules/,

loader: 'babel',

query: {

presets: ['es2015', 'react']

}

}]

}

}

module.exports = config;

* **entry:**指定打包的入口文件 **main.js**。
* **output：**配置打包结果，path定义了输出的文件夹，filename则定义了打包结果文件的名称。
* **devServer：**设置服务器端口号为 **7777**，端口后你可以自己设定 。
* **module：**定义了对模块的处理逻辑，这里可以用loaders定义了一系列的加载器，以及一些正则。当需要加载的文件匹配test的正则时，就会调用后面的loader对文件进行处理，这正是webpack强大的原因。

现在打开 **package.json** 文件，找到 **"scripts"** 中的 **"test" "echo \"Error: no test specified\" && exit 1"** 使用以下代码替换：

"start": "webpack-dev-server --hot",

"build": "webpack --config webpack.config.js"

找到 **"**"main": "index.js",**"** 后添加以下代码

"dependencies": {

"react": "^15.4.1",

"react-dom": "^15.4.1"

},

"devDependencies": {

"babel": "^6.5.2",

"babel-core": "^6.21.0",

"babel-eslint": "^6.0.0",

"babel-loader": "^6.2.4",

"babel-plugin-add-module-exports": "^0.1.2",

"babel-plugin-transform-decorators-legacy": "^1.3.4",

"babel-polyfill": "^6.7.2",

"babel-preset-es2015": "^6.6.0",

"babel-preset-react": "^6.5.0",

"babel-preset-stage-0": "^6.5.0",

"webpack": "^1.14.0",

"webpack-dev-server": "^1.16.2"

}

修改后的 **package.json** 文件 内容如下：

$ cat package.json

{

"name": "runoob-react-test",

"version": "1.0.0",

"description": "菜鸟教程 react 测试",

"main": "index.js",

"scripts": {

"start": "webpack-dev-server --hot",

"build": "webpack --config webpack.config.js"

},

"author": "",

"license": "ISC",

"dependencies": {

"react": "^15.4.1",

"react-dom": "^15.4.1"

},

"devDependencies": {

"babel": "^6.5.2",

"babel-core": "^6.21.0",

"babel-eslint": "^6.0.0",

"babel-loader": "^6.2.4",

"babel-plugin-add-module-exports": "^0.1.2",

"babel-plugin-transform-decorators-legacy": "^1.3.4",

"babel-polyfill": "^6.7.2",

"babel-preset-es2015": "^6.6.0",

"babel-preset-react": "^6.5.0",

"babel-preset-stage-0": "^6.5.0",

"webpack": "^1.14.0",

"webpack-dev-server": "^1.16.2"

}

}

### 第四步、index.html

设置 **div id = "app"** 为我们应用的根元素，并引入 **index.js** 脚本文件。

<!DOCTYPE html>

<html>

<head>

<meta charset = "UTF-8">

<title>React App - 菜鸟教程(runoob.com)</title>

</head>

<body>

<div id = "app"></div>

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

</body>

</html>

### 第五步、App.jsx 和 main.js

这是第一个 react 组件。这个组件将输出 **Hello World!!!**。

#### App.jsx 文件代码

import React from 'react';

class App extends React.Component {

render() {

return (

<div>

Hello World!!!<br />

欢迎来到菜鸟教程学习！！！

</div>

);

}

}

export default App;

我们需要引入组件并将其渲染到根元素 **App** 上，这样我们才可以在浏览器上看到它。

#### main.js 文件代码

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App.jsx';

ReactDOM.render(<App />, document.getElementById('app'))

***注意：***

*如果想要组件可以在任何的应用中使用，需要在创建后使用****export****将其导出，在使用组件的文件使用****import****将其导入。*

### 第六步、加载所需要的依赖

npm install --verbose

### 第七步、打包

npm run build

现在我们可以使用 **npm start** 命令来启动服务。**--hot** 命令会在文件变化后重新载入，这样我们就不需要在代码修改后重新刷新浏览器就能看到变化。

### 第八步、运行服务

完成以上配置后，我们即可运行该服务：

npm start

通过浏览器访问 **http://localhost:7777/**，输出结果如下：

