爱创课堂前端培训

# NodeJS第一天

第1天课堂笔记（本课程共6天）

班级：北京前端训练营19期

讲师：李兰波

日期：2019年1月21日

爱创课堂官网 ：[www.icketang.com](http://www.icketang.com)

# NPM

NPM：NodeJS Package Manager NodeJS的模块管理器

NodeJS是模块化开发的。有两种模块。一种是自带的内置核心模块，一种是其它程序人员编写的第三方模块。

NPM是用来管理所有的第三方模块的。

比如 程序员A开发了一个模块a。程序员B开发了一个模块b。他俩分别处于南北半球。但是又正好需要对方开发出来的模块。此时如果他们都上传自己的模块到NPM。那么他们就可以搜索NPM上是否有自己需求的模块。就可以搜索得到。但是如果没有NPM，那么他们只能各自再开发一个对方的模块。就重复开发了。

官网：<https://www.npmjs.com/>

NPM已经集成在NodeJS的安装包中。

查看NodeJS版本：

|  |
| --- |
|  |

查看NPM版本：

|  |
| --- |
|  |

# 环境变量

windows系统中，有一个地方叫做环境变量。它可以监控所有的命令。只要你配置了环境变量。

它是一个配置项，配置的值是一个一个的字符串路径。比如：如果不配置环境变量，那么一个程序只可以在它的安装目录下调用。如果配置了环境变量：将该程序的所在目录字符串放到环境变量中，那么就可以在任意的地方去调用该程序。

举例：安装nodeJS时，我故意将addToPath选项取消。那么安装完毕之后，NodeJS不会将自身的所在路径加入到环境变量中。那么Node命令将只能够在安装目录下调用。

|  |
| --- |
| 1. 在外部调用失败  1. 切换到安装目录下  1. 调用成功 |

现在，我将NodeJS的安装目录 C:\Program Files\nodejs放入环境变量中

打开环境变量的方式： 右击桌面的计算机，选择属性 => 在新打开的面板中找到高级系统设置，弹出一个系统属性面板 => 选择“高级”选项卡（一般会默认） => 在该面板下选择环境变量按钮并点击 => 弹出一个新的面板 从系统变量中找到Path选项=> 如果是win7系统，则弹出的面板中所有的路径字符串都在一行，此时需要在每一个路径字符串之后，添加一个英文半角分号。并将刚刚复制的C:\Program Files\nodejs放在最后。保存即可。 如果是win10 则添加会弹出一个全新的面板。此时粘贴并保存即可。每一个面板都需要点击确定。

此时，打开一个新的命令面板。输入node -v测试是否配置成功。

# NPM的常用命令

## 3.1 下载模块

npm install moduleName1 moduleName2 ……

该命令用于下载第三方模块。下载规则：查看当前文件夹下是否具备node\_modules文件夹 如果有就把模块下载下来并放进去 如果没有node\_modules文件夹 此时会向上一级查找 如果找到就放入 如果没有再向更上一层找。直到最顶层。如果还没有找到，回到当前的下载目录中，创建一个新的node\_modules文件夹并放入。

## 3.2 指定版本

模块是具备版本号的。所以，你可能下载的并不是你想要的版本。因为默认下载最新版本。此时可以使用moduleName@x.x.x这样的方式去下载指定版本

demo:

|  |
| --- |
| 1. npm install jquery@1.7.2 |

## 3.3 查看版本

npm info moduleName

该命令用于查看模块信息

|  |
| --- |
|  |

## 3.4 package.json

这是一个文件。每一个NodeJS的第三方模块都应当具备该文件。该文件是当前模块的配置信息。比如：模块名称、模块描述、模块版本、模块的仓库、模块的启动命令等。

underscore.js模块的配置信息文件：package.json

|  |
| --- |
|  |

package.json内容：

|  |
| --- |
|  |

## 3.5 生成package.json

npm init 该命令用于生成package.json

生成过程：

|  |
| --- |
|  |

## 3.6 添加依赖到package.json

我们在开发之前，要生成一个package.json 因为我们开发的过程中要下载模块，如果没有package.json 则会报警告。如果有该文件，则会将下载的工具库添加到依赖项中。

|  |
| --- |
|  |

## 3.7 重新安装模块

如果有一天你的项目要从一台电脑上移动到另外一台电脑上。此时，我们应该复制并粘贴。但是这样的话，如果项目挺大，依赖的模块挺多。这样会花费较长的时间。于是我们就不需要移动模块文件。也就是不要node\_modules文件夹了。但是没关系，我们可以只带package.json文件过去。此时我们可以运行**npm install**命令。重新下载项目所依赖的模块。

# Express

这是一个服务器框架。

## 4.1 下载

npm install express

|  |
| --- |
|  |

## 4.2 搭建服务器

代码一共三行：

|  |
| --- |
| 1. // 既然要使用express创建服务器 那么就引入进来 2. var express = require("express"); 3. // 创建服务器 4. var app = express(); 5. // 监听端口号 6. app.listen(3000); |

访问：

|  |
| --- |
|  |

## 4.3 静态功能

express唯一内置的中间件 叫做“静态中间件”

它的作用：将某一个目录静态化

被静态化的目录中的内容就可以直接被前端访问。

使用方式:

|  |
| --- |
| 1. // 既然要使用express创建服务器 那么就引入进来 2. var express = require("express"); 3. // 创建服务器 4. var app = express(); 5. // Express唯一内置的中间件 static 该中间件用于静态化目录 6. app.use("/web/", express.static("web")); 7. // 监听端口号 8. app.listen(3000); |

现在将web目录下的内容都静态化了

app.use("/web/", express.static("web"));

以上这条代码中，第一个参数表示访问时的路径 第二个参数是被静态化的目录

如果没有第一个参数：app.use(express.static("web"));

|  |
| --- |
| 1. http://localhost:3000/index.html |

但是此时index.html其实是在服务器根目录下的web目录下。也就是前后端路径表示法不同。

为了统一，我们填写第一个参数：app.use("/web/", express.static("web"));

|  |
| --- |
| 1. http://localhost:3000/web/index.html |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |