NODE.JS

> node.js是基于v8引擎运行js的环境，它不是一门语言，你可以把它理解为谷歌浏览器，它是工具

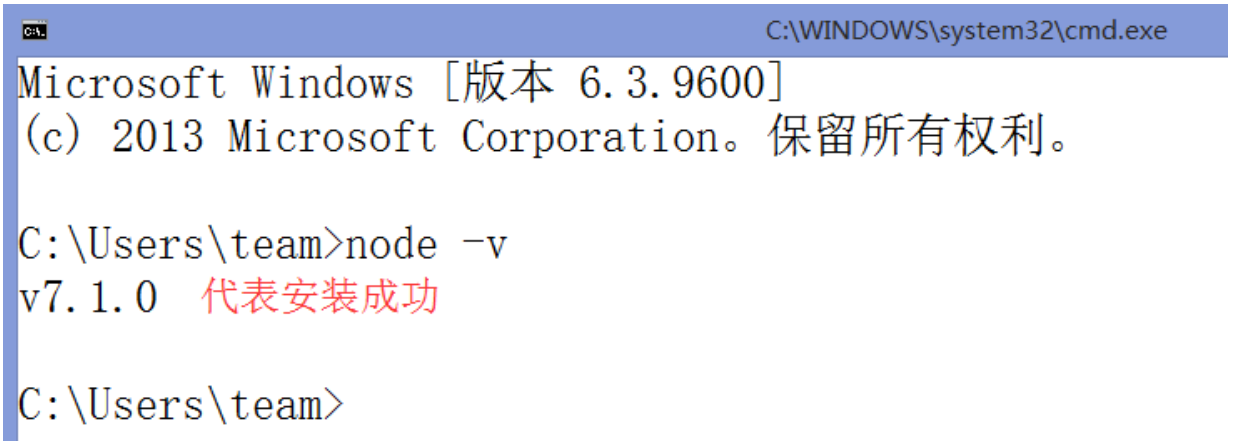
为什么把node称之为后台语言？

> 客户端向服务器端发送请求，服务器端需要编写相关的程序，把客户端请求的内容准备好，然后返回给客户端
>
> 我们可以使用java/php等语言编写这些程序，也同样可以使用js编写这些操作（js是全栈编程语言，它可以写后台的程序了）
>
> js代码写完后，我们要把它运行，此时我们在服务器上安装一个node工具，使用node可以把这些代码执行，从而让其具备相关的功能即可

安装node.js

http://nodejs.cn/ 中文
https://nodejs.org/en/ 英文

安装的时候基本上一路下一步即可，默认情况下，node安装成功后，会把相关的操作命令集成到系统的dos命令中（MAC是终端），以后我们可以在DOS(终端)命令中执行node的命令



node.js的基础知识

基于node环境使用js编写后台程序，相对于传统的后台语言具备了一些优势

- 基于V8引擎渲染解析代码（快）
- 基于事件驱动的非阻塞I/O操作
- 采用的是异步单线程开发
- npm包管理器，是全球最大的开源库生态系统：https://www.npmjs.com/
- 对于前端开发工程师来说，学习成本低，可以快速入手这门技术

NODE中的模块

> node其实就是由很多模块拼起来的
>
> - 内置模块：node环境本身自带的（类似于浏览器也会天生自带一些自己的方法）
> - 自定义模块：开发者自己编写的
> - 第三方模块：别人写好的模块，我们可以下载下来使用（类似于在客户端导入JQ）
>
> 我们需要使用的第三方模块，在npmjs.com中都可以获取到；而模块的下载安装统一使用npm包管理器完成；

NPM以及第三方模块

`npm install xxx -g` 把模块安装在全局环境下
`npm install xxx --global`
浏览器中的全局对象是window，NODE中的全局对象是global

`npm install xxx` 把模块安装在当前操作的项目目录下
`npm install xxx --save-dev` 不仅安装在当前下，并且把安装的信息记录在项目的package.json清单中，生成一条开发环境依赖项
`npm install xxx --save` 和上述操作类似，不过生成的是一条生产环境依赖项

`npm uninstall xxx -g/--save-dev/--save` 相当于安装来说，当前操作为卸载这些模块

`npm install jquery@1.11.3` 在安装这个模块的时候，指定了安装的版本号
…

1、安装在全局和安装在本地项目中的区别

安装在全局

```
C:\Users\team>npm install less --global
C:\Users\team\AppData\Roaming\npm\lessc -> 安装的命令
e_modules\less\bin\lessc
- assert-plus@1.0.0 node_modules\less\node_
plus
- assert-plus@1.0.0 node_modules\less\node_
lus
- assert-plus@1.0.0 node_modules\less\node_
us
- assert-plus@1.0.0 node_modules\less\node_
s
- assert-plus@1.0.0 node_modules\less\node_
us
C:\Users\team\AppData\Roaming\npm 安装在全局下的目录
`-- less@2.7.2
  `-- request@2.82.0
```

安装在全局下，会在安装的全局目录中生成一个文件:

`lessc.cmd ->可以在DOS中执行命令的文件，此时我们就可以在DOS中执行 lessc 这个命令了`

lessc.cmd

```
@IF EXIST "%~dp0\node.exe" (
  "%~dp0\node.exe"  "%~dp0\node_modules\less\bin\lessc" %*
) ELSE (   执行lessc命令，其实就是在node环境下，把指定的JS文件
  @SETLOCAL中的代码给执行了（实现想要的功能）
  @SET PATHEXT=%PATHEXT:;.JS;=;%
  node  "%~dp0\node_modules\less\bin\lessc" %*
)
```

安装在全局环境下的模块可以使用命令来操作，但是只能使用命令操作，如果想把这个安装的模块导入到我们自己的JS代码中使用，则是不可以的

安装在项目中

> npm install less
>
> 安装完成后在当前的项目目录下多了一个文件夹：`node_modules`，此文件夹中包含了我们安装的less模块

安装在本地项目中的模块无法使用命令来操作（默认情况下）；但是可以在当前项目的JS代码中，通过require把它导入进来，然后在js代码中调取模块中的方法，实现一些特殊的处理；

```
1. let lessc= require('less');
2. lessc.render();
```

2、能否有办法即能使用命令也能导入到JS中？

> 真实项目开发的时候，我们很少安装在全局，因为安装在全局可能导致版本的冲突，一般我们都安装在本地项目中

在本地项目中配置模块的运行命令

`npm init -y`
在本地项目中生成一个`package.json`文件：项目的配置文件（命令中不加-y，需要自己在执行的时候一个个的输入配置信息，加-y一切都走默认的信息，比较方便快速）

```
{
  "name": "BASE-INFO",//->项目名称
  "version": "1.0.0", //->项目版本
  "description": "",//->项目的描述
  "main": "index.js",//->项目的入口页面(首页面)
  "dependencies": {//->生产环境依赖模块清单
    "less": "^2.7.2"
  },
  "devDependencies": {},//->开发环境依赖模块清单
  "scripts": {//->项目的命令脚本配置信息
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],//->关键词
  "author": "",//->作者
  "license": "ISC"//->监听模式
}
```

在生成的package.json文件中的`scripts`属性中，配置我们需要运行的命令

```
{
    ...
    "scripts": {
        //->zxt是自己定义的属性名（随便起）
        //->属性值是我们即将要执行的命令（使用的就是less的命令）
        "zxt": "lessc LESS/index.less CSS/index.min.css -x"
    }
    ...
}
```

配置完成后，接下来在当前项目的DOS命令中执行：`npm run zxt`

```
E:\201708\WEEK9\BASE-INFO>npm run zxt

> BASE-INFO@1.0.0 zxt E:\201708\WEEK9\BASE-INFO
> lessc LESS/index.less CSS/index.min.css -x
```

相当于执行run zxt的时候，把zxt的属性值，在DOS命令中给执行了，而属性值就是把某个less编译成css的命令

3、生产环境和开发环境

> 开发环境：项目在本地开发的时候，所需要依赖的模块叫做开发依赖项
> 生产环境：项目开发完成部署到服务器上，所需要依赖的模块叫做生产环境依赖项
>
> less模块，开发的时候需要依赖，项目部署后不需要依赖；开发的时候需要安装less模块，项目上线则不需要安装；

`npm install less --save-dev` 安装less模块并且把安装的信息存放在开发依赖项中

`npm install less --save` 安装less模块并且把安装的信息存放在生产依赖项中

为啥要设置依赖项？
项目如果是多人开发，我们使用git仓库来管理项目代码以及实现团队协作开发

A是其中的一个开发人员，开发这个项目需要用到很多模块，A在自己的电脑上已经把需要的模块都安装在本地项目中了（node_modules）

A在提交自己的代码到git仓库的时候，会忽略node_modules文件夹的提交：因为这个文件中的内容太大了（当前项目增加.gitignore文件）

B从git仓库下载代码，代码都有了，但是开发需要依赖的模块没有，项目无法运行，此时B也需要安装这些模块
1）找到A，手动记录一下需要的模块，然后B自己在本地一个个的安装（太low了）
2）此时体现出我们配置依赖项的好处了，A在他本地安装的时候，把安装的信息都记录到package.json的devDependencies/dependencies这里面，虽然node_modules没有传递到git仓库中，但是package.json传递上去了，B下载完成后，在本地的package.json中可以看到需要依赖的模块信息，此时的B只需要执行：`npm install` 命令，就可以把当前项目需要依赖的模块自动的都安装上，我们把这个操作叫做`跑环境`

> 项目上线也是同样的原理