# NodeJS入门 & Webpack基本使用

1. 课程介绍

* NodeJS介绍（了解）
* Node的安装（**掌握**）
* 常用的dos命令（**掌握**）
* Node的模块化导入导出（**掌握**）
* Npm（**重点掌握**）
* Yarn（**重点掌握**）
* Webpack介绍（了解）
* Webpack的5个核心（掌握）
* Webpack的使用步骤（掌握）
* 核心loader（**重点掌握**）

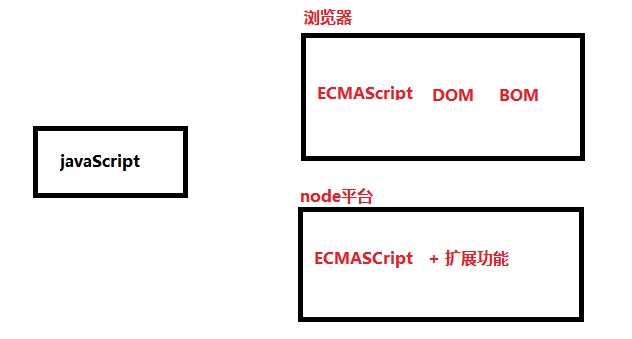
1. 什么是NodeJS？（了解）

## 2.1 简介

NodeJs就是一个可以让JavaScript脱离浏览器还能执行的平台,并且这个平台对JavaScript功能进行了增强，让JavaScript具备服务器端编程语言的能力。

可以理解为： NodeJs = JavaScript执行环境 + JavaScript扩展的功能。

## 2.2 JS运行在浏览器和Node环境的区别？



## 2.3 为什么要学习NodeJS?

现在主流前端工具，框架，大部分都是基于Node的。

# 3. Node的安装

## 3.1 官方文档

<http://nodejs.cn/> （Nodejs中文网）

## 3.2 下载安装步骤（重点掌握）

去官方下载地址： <http://nodejs.cn/download/> 下载对应版本，点击下一步安装即可。

安装完成，在dos中运行命令： **node -v** 如果看到版本号，证明安装成功。

**注意：默认安装c盘，英文目录，不建议修改。**



4. 常用DOS命令（掌握）

## 4.1 Dos介绍（了解）

使用window系统有两种方式:

1）可视化的界面(窗口),

2）DOS系统

DOS命令是windows系统中自带的命令行操作系统。

## 4.2 常用命令（掌握）

进入dos系统：  **window + r ，然后输入 cmd， 然后点击回车。**

**常用dos命令：**

**盘符:**  // 切换盘符

**cd 目录**  // 切换目录

**cd /**  // 回到根目录

**dir**  // 显示当前文件夹中的文件和目录列表

**cls**  // 清除屏幕

**Ctrl + c**  // 中断执行

1. Node的模块化（掌握）

## 5.1 js如何在node环境运行？（掌握）

首先，我们得知道js文件如何运行在node环境中，只需要：

在dos或 终端 中运行： **node JS文件的名字**

## **5.2 模块化介绍（了解）**

模块: 一个具有特定功能的文件就是一个模块，模块之间可能存在一定的依赖关系，使用模块可以很好的把这些依赖关系整合起来。比如jquery的轮播插件依赖与jquery核心模块。

模块之间可以相互导入和导出。

## 5.3 模块的导入和导出（掌握）

导出模块： **module.exports = 要导出的东西**

导入模块：  **let 变量 = require(‘路径’)**

注意：require(‘路径’)有三种写法

1. 如果引入的模块是自己写的文件，写相对路径。
2. 如果引入的模块，是node内置的，直接写模块名。
3. 如果引入的模块，是node的第三方模块（npm下载的），直接写模块名。
4. npm命令(重点掌握)

## 6.1 包描述文件package.json（掌握）

每个项目的根目录下面，一般都有一个package.json文件，定义了这个项目所需要的各种模块，以及项目的配置信息（比如名称、版本、许可证等元数据）。

**npm**是node的包管理工具（命令）。

创建包描述文件： **npm init**  然后根据引导操作即可。

**package.json核心字段：**

name: 名字

scripts: 可执行脚本

dependencies: 运行依赖(项目上线运行，需要的依赖模块)

devDependecies: 开发依赖(只是开发阶段需要，运行不需要的依赖模块)

## 6.2 npm常用命令（重点掌握）

**局部（本地）安装：（安装到当前目录中）**

**npm init**  // 创建包描述文件package.json

**npm i 模块名**  // 本地安装模块（本地安装就是安装在当前目录）

**npm i 模块名 -D** // 本地安装模块且记录到开发依赖（默认记录到运行依赖）

**npm i 模块名@版本号** // 安装指定版本的模块

**npm i**  // 一次性安装package.json中记录的所有依赖模块（包含运行依赖和开发依赖）

**全局安装：（安装到c盘中）**

**npm i 模块名 -g** 或 **npm i -g 模块名**

**其他命令：**

**npm remove 模块名** // 删除模块

**注意：**

1. 全局安装必须配置环境变量，默认npm的环境变量已经自动配置了。
2. 如果npm下载源比较慢，建议切换下载源为淘宝镜像
3. 查看当前下载源

**npm config get registry**

1. 切换下载源

**npm config set registry https://registry.npm.taobao.org**

1. yarn（重点掌握）

yarn也是依赖管理工具。在企业开发中，npm和yarn二选一。

安装yarn： **npm i yarn -g**

**yarn常用命令：**

**局部（本地）安装：（安装到当前目录中）**

**yarn init**  // 创建包描述文件package.json

**yarn add 模块名**  // 本地安装模块（本地安装就是安装在当前目录）

**yarn add 模块名 -D** // 本地安装模块且记录到开发依赖（默认记录到运行依赖）

**yarn add 模块名@版本号** // 安装指定版本的模块

**yarn**  // 一次性安装package.json中记录的所有依赖模块（包含运行依赖和开发依赖）

**全局安装：（安装到c盘中）**

**yarn global add 模块名**

**其他命令：**

**yarn remove 模块名** // 删除模块

**注意：**

1. 全局安装必须配置环境变量，yarn的环境变量必须手动配置。

配置地址：C:\Users\Administrator\AppData\Local\Yarn\bin

环境变量： 电脑（计算机）-》右键属性-》高级系统设置-》环境变量-》Path

把yarn的全局安装目录粘贴进去即可。

1. 如果yarn模块下载源比较慢，建议切换下载源为淘宝镜像
2. 查看当前下载源

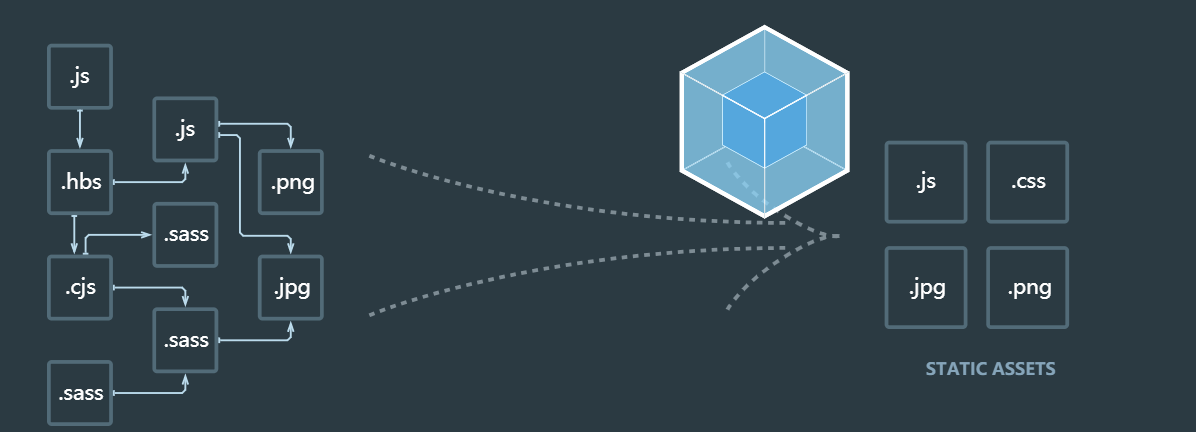
**yarn config get registry**

1. 切换下载源

**yarn config set registry https://registry.npm.taobao.org**

1. webpack简介【了解】

webpack是一个现代 JavaScript 应用程序的静态模块打包器(module bundler)，是一种前端资源构建工具，在webpack看来，前端所有资源文件（js、json、css、img、less、typeScript、等等】）都会作为模块处理。它将根据模块的依赖关系进行静态分析，打包生成对应的静态资源（bundle）。



当 webpack 处理应用程序时，它会递归地构建一个依赖关系图(dependency graph)，其中包含应用程序需要的每个模块，然后将所有这些模块打包成一个或多个 bundle。

1. webpack的5个核心概念【掌握】

## 9.1 Entry

入口起点(entry point)指示 webpack 应该使用哪个模块，来作为构建其内部依赖图的开始。

可以通过在 [webpack 配置](https://www.webpackjs.com/configuration)中配置 entry 属性，来指定一个入口起点（或多个入口起点）

## 9.2 Output

output 属性告诉 webpack 在哪里输出它所创建的 bundles，以及如何命名这些文件，默认值为 ./dist。基本上，整个应用程序结构，都会被编译到你指定的输出路径的文件夹中。

## 9.3 Loader

loader 让 webpack 能够去处理那些非 JavaScript 文件（**webpack 自身只理解 JavaScript**）。loader 可以将所有类型的文件转换为 webpack 能够处理的有效[模块](https://www.webpackjs.com/concepts/modules)，然后你就可以利用 webpack 的打包能力，对它们进行处理。

## 9.4 Plugins

loader 被用于转换某些类型的模块，而插件则可以用于执行范围更广的任务。插件的范围包括，从打包优化和压缩，一直到重新定义环境中的变量。[插件接口](https://www.webpackjs.com/api/plugins)功能极其强大，可以用来处理各种各样的任务。

## 9.5 Mode

通过选择 **development**或 **production**之中的一个，来设置 mode 参数，你可以启用相应模式下的 webpack 内置的优化

|  |  |
| --- | --- |
| **选项** | **选项** |
| development | 会将 process.env.NODE\_ENV 的值设为 development。启用 NamedChunksPlugin 和 NamedModulesPlugin。代码本地调试环境 |
| production | 会将 process.env.NODE\_ENV 的值设为 production。启用 FlagDependencyUsagePlugin, FlagIncludedChunksPlugin, ModuleConcatenationPlugin, NoEmitOnErrorsPlugin, OccurrenceOrderPlugin, SideEffectsFlagPlugin 和 UglifyJsPlugin。优化线上环境。 |

1. webpack使用步骤【掌握】

前提条件，保证node版本在10.x以上。

## 10.1 使用步骤

1）创建项目文件夹webpack-demo，初始化生成 **package.json**

|  |
| --- |
| **yarn init**  **// 创建 package.json** |

1. 本地安装 **webpack 和 webpack-cli**

|  |
| --- |
| **yarn add webpack@5.12.0 webpack-cli@4.5.0 -D // 本地安装** |

## 10.2 使用步骤

1）在项目根目录创建 **webpack.config.js**文件。

1. 项目根目录创建**src**文件夹，里面创建文件 **index.html** 和 **index.js**。
2. **Index.js** 编写如下代码 （自己随便写一点代码）

|  |
| --- |
| // 随便写一点JavaScript代码哦~~~  const sum = (num1, num2) => num1 + num2;  console.log(sum(1, 2)) |

1. **Webpack.config.js** 文件配置如下

|  |
| --- |
| /\*\*   \* webpack配置文件   \*/  // 引入path模块，path是node内置的模块 主要用于处理路径拼接  const path = require('path')  // 把webpack的配置暴露出去  module.exports = {    // 1. 入口    entry: './src/index.js',    // 2. 出口    output: {      path: path.resolve(\_\_dirname, 'dist'), // 输出路径 必须是绝对路径      filename: 'bundle.js'  // 输出的文件名    },    // 3. 配置loader    module: {      // 配置loader规则    },    // 4. 插件    plugins: [],    // 5. 模式    mode: 'development' // 开发模式 development 生产模式 production  } |

1. 修改package.json

|  |
| --- |
| "scripts": {      "build": "webpack"    } |

1. 执行打包命令

**yarn build** 或 **npm run build**

6）在 **index.html**中，引入打包后的结果 **bundle.js**, 浏览器打开**index.html**开效果

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>    <meta charset="UTF-8">    <meta http-equiv="X-UA-Compatible" content="IE=edge">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Document</title>  </head>  <body>    <script src="../dist/bundle.js"></script>  </body>  </html> |

1. 核心loader【重点】

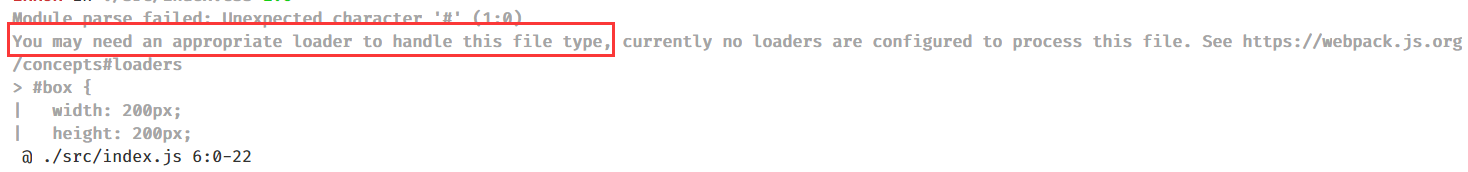
loader 让 webpack 能够去处理那些非 JavaScript 文件（webpack 自身只理解 JavaScript）。loader 可以将所有类型的文件转换为 webpack 能够处理的有效[模块](https://www.webpackjs.com/concepts/modules)。

## 11.1 加载css的loader

创建一个 **index.css**, 然后再 **index.js** 中使用**node**的模块化写法引入。

|  |
| --- |
| // 引入index.css  require('./index.css') |

**直接运行打包命令： yarn build 或 npm run build. 发现报错如下**



模块解析失败，你可能需要一个合适的loader处理这个类型的文件。这就是因为webpack不能打包非js文件，需要loader把它处理成webpack能够处理的模块。

下面，我们来学习如何配置loader【所有的loader配置都只有2步哦】。

1. 下载对应的loader 【第一步】

|  |
| --- |
| **yarn add style-loader css-loader -D** |

1. 配置loader【第二步】

|  |
| --- |
| // loader配置    module: {      rules: [        {          test: /\.css$/,  // 匹配所有.css文件          use: [            'style-loader', // 创建style标签 把css插入html的head内            'css-loader' // 解析加载css到js中（把css打成字符串）。          ]        }      ]    }, |

1. 今日作业
2. node模块导入导出代码。
3. npm命令【10遍】
4. yarn命令【10遍】
5. webpack打包代码和第一个loader配置【走通】
6. 面试题
7. npm常用命令有哪些？
8. yarn常用命令有哪些？
9. Npm 和 yarn 有什么区别？
10. 扩展阅读

node中文网： <http://nodejs.cn/learn>

yarn官网： <https://yarn.bootcss.com/>

webpack官网：<https://webpack.docschina.org/>