1.NPM的扩展使用

(1)npm init : 初始化一个Node.js项目——创建必需的package.json文件

npm init -y：创建必需的package.json文件

(2)npm help json：查看package.json文件中可用的所有条目

(3)npm install xxx：下载并安装指定的扩展依赖模块

npm install xxx **--save**：下载并安装指定的依赖模块，并声明为“运行时依赖”

npm install xxx **--save-dev**：下载并安装指定的依赖模块，并声明为“开发时依赖”

npm install: 根据package.json中的“**运行时依赖**”和“**开发时依赖**”列表进行指定模块的下载

(4)npm run xxx：执行当前目录下package.json中的scripts所定义的命令；若xxx是test/start/stop/rebuild之一，可以省略掉run简写为 npm xxx，即npm rum start等价于npm start

=================================

package.json中的特殊用法：

{

"scripts":{ 定义可以使用npm run执行的命令

"test": "node ./test.js",

"start": "node ./main.js"

},

"dependencies":{ 定义当前项目的“运行时依赖”

"express": "^4.3.0",

"mysql": "^2.16.0"

},

"devDependencies":{ 定义当前项目的“开发时依赖”

"scss"： "^1.15.0",

"gulp": "2.0.0"

}

}

|  |
| --- |
| 运行时依赖：指当前项目在部署到服务器上供客户端请求时需要使用到的第三方模块 |
| 开发时依赖：指当前项目在程序员开发阶段（在最终运行之前）需要使用到的第三方模块 |

2.模块的使用

注意：在JS中使用全局变量/函数，会自动成为全局对象的成员——称为“**全局对象的污染问题**”；后续重名的变量/函数会覆盖之前的变量/函数。 实际开发中，应努力避免使用全局变量/函数。

解决方案：—— 模块(Module)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **CommonJS中的模块规范** | **ES6中的模块规范** | |
| 导出 | //x.js  module.exports = {  age: 20,  work: function(){ }  }  //每个模块只能导出一次 | //x.js  export var age = 20;  export function work(){}  //每个模块可以导出多个内部成员 | //x.js  export default {  age: 20,  work: function(){}  }  //每个模块只能导出一个默认对象 |
| 导入 | //y.js  const obj = require('./x');  console.log(obj.age);  obj.work(); | //y.js  import **{age, work}** from './x';  console.log(age);  work(); | //y.js  import **obj** from './x';  console.log(obj.age);  obj.work(); |
| 书写位置 | 导入/导出可以写在顶级或内部 | 导出/导入只能写在最顶级 | |
| 适用场合 | 服务器端Node.js中使用；  浏览器中不能使用 | 服务器端Node.js不支持 —— 这是客户端浏览器中的JS模块规范！—— 目前所有浏览器都不能直接支持此规范！ | |

3.Webpack的使用

**官网：**www.webpackjs.com

**作用：**把多个静态资源文件(脚本/样式/图片等)“打包”为一个/少数几个静态资源文件 —— 减少客户端请求次数/数据传输量，提高Web访问效率。

**使用方法：**先使用传统方法写完一个Web项目(若干js/css/图片...)，再使用Webpack把静态资源打包，在HTML中引入打包后的资源文件让客户端请求。

**使用步骤：**

(0)按照传统方法编写Web项目

(1)下载并安装Webpack所需要的所有模块

npm i webpack --save-dev

(2)在项目根目录下创建Webpack主配置文件

webpack.config.js 注意文件位置/名称

(3)运行Webpack，读取配置文件，执行打包工作

node ./node\_modules/webpack/bin/webpack.js

(4)修改HTML文件，只引入打包后得到的打包文件即可