1.CommonJS和ES中的模块系统

node.js 每个JS文件都是一个模块对象，所有“全局变量/函数”都是模块对象内部的局部变量——避免污染

模块：Module，用于封装特定的数据和操作，防止污染全局对象

|  |  |  |  |
| --- | --- | --- | --- |
|  | **CommonJS规范** | **ES6规范** | |
| 导出内容 | x.js  module.exports = {  age: 10,  print: function(){}  } | export var age=10;  export function print(){} | export default {  age: 10,  print: function(){}  } |
| 导入内容 | y.js  const obj = require('./x') | import {age,print} from './x' | import obj from './x'  //obj为变量名 |
| 导出数量 | 只能导出一个对象 | 可以导出多个变量或方法(完整的一个对象) | 默认导出只能有一个 |
| 使用限制 | 可以是最顶层或内部代码  //y.js  //var obj=require(‘./x’);  function(){  var obj=require(‘./x’)  } | 只能是最顶层代码 | |
| 支持情况 | 服务器端技术：可以被Node.js解释器直接理解 | 客户端技术：目前还不能被各大浏览器直接支持，只能使用编译器进行预处理转换为标准JS代码 | |

Deno:(node.js升级版本) ES6

Webpack：具备把ES模块规范转换为当前浏览器能够理解的ES5代码的能力。

2.Webpack

项目的自动化构建工具

**手册：**<https://www.webpackjs.com/>

**概念**：Webpack 是一个现代 JavaScript 应用程序的静态模块打包器。当 webpack 处理应用程序时，它会递归地构建一个依赖关系图，其中包含应用程序需要的每个模块(可能是JS/CSS/图片等)，然后将所有这些模块打包成一个资源包(bundle)文件。

**用途**：资源打包后，再供客户端一次性加载，减少请求次数和请求数据总大小，从而提高访问效率。

**注意**：Webpack本身用于打包静态资源，通常的使用场景是：项目初步开发完成后，运行Webpack(是一个NPM扩展包)，把需要处理的静态资源打包，把得到的打包文件和HTML文件一起发布到WEB服务器(可以是Node / PHP / Java等)上供客户端访问。

**核心概念：SSR S**erver **S**ide **R**ender**(服务器端渲染)**

(1)entry：入口，指定资源包的入口文件

(2)output：输出，指定最终输出的资源包文件

(3)loader：加载器，让Webpack可以处理非JS文件，如CSS/图片等

(4)mode：运行模式，可以指定为production或者development

**使用过程**：

(1)创建客户端项目，编写HTML/CSS/JS/图片等文件

(2)创建项目的描述文件：package.json

npm init

(3)下载webpack及其依赖的模块，并添加为当前项目的“开发阶段依赖模块”

npm i webpack --save-dev

(4)创建webpack主配置文件： webpack.config.js

module.exports = {

mode: 'production',

entry: './src/index.js',

output: {

path: \_\_dirname + '/dist',

filename: 'bundle.js'

},

module: { },

plugins: [ ]

}

(5)运行webpack，根据主配置文件对静态资源进行打包

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

或者配置为scripts： "build": "webpack"，用 npm run build方式运行