前端工程化（es6模块化）

1. 模块化概述

**传统开发模式的主要问题**： ① 命名冲突 ②文件依赖

**通过模块化解决上述问题：**

**概念**：模块化就是把单独功能封装到一个模块（文件）中，模块之间相互隔离，但是可以通过特定的接口公开内部成员，也可以依赖别的模块

**好处：**方便代码的重用，从而提高开发效率，并方便后期的维护

二、模块化规范

**1．浏览器端模块化规范（已不推荐使用）**

① AMD (Require.js)

② CMD(sea.js)

**2.服务器端的模块化规范(Commonjs)**

模块分为单文件模块与包

模块成员导出： module.exports 与 exports

模块成员导入： require(‘模块标识符‘)

**3.大一统的模块化规范（ES6模块化）**

**定义：**每个js文件都是一个独立的模块

导入成员使用import关键字

暴露模块成员使用exports关键字

三、Node.js中通过babel体验ES6模块化

**1.配置模块化规范（babel）**

①npm install –save-dev @babel/core @babel/cli @babel/preset-env @babel/node

②npm install –save @babel/polyfill

③项目根目录下创建文件，文件名为:babel.config.js并将文件配置文件内容为以下代码：

Const presets = [

[“@babel/env”,{

Targets: {

Edge: “17”,

Firefox: “60”,

Chrome: “67”,

Safari: “11.1”

}

}

]

];

Module.exports={presets};

④可以用npx babel-node命令运行index.js文件

**2.ES6模块规范的导入导出**

①默认导入导出方式

默认导出：export default{默认导出的成员}

默认导入：import 自定义名 from ‘模块标识符’

② 按需要导入导出

按需导出：export let 名= 值

按需导入： import {名(对应导出的名)} from ‘模块标识符’

③ 直接导入模块

Import ‘模块标识符’