02.模块打包和配置

学习要点：

1.模块打包

2.配置文件

本节课我们来开始学习一下将包含模块的 JS 文件进行打包以及简化的配置；

一. 模块打包

1. 我们创建 02 目录，在其创建约定俗成的两个子目录 src 和 dist；
2. 其中：src 表示源文件，dist 表示生成发布的文件；
3. 在 src 中建立两个 js 文件，用 node 环境支持的 CommonJS 规范的模块化；

```
//module.js
module.exports = {
    'Mr.Lee'
};

//index.js
const name = require('./module.js').name;
console.log(name);
```

4. 建立 02.html 文件，引入 src 中的 index.js 文件，但浏览器是无法运行的；
5. 此时，我们需要进行打包来解决两个问题：浏览器兼容和导入导出 js 文件的合并；
6. 使用打包命令，注意目录的问题，如果在根目录带上子目录路径；

```
webpack ./02/src/index.js -o ./02/dist/bundle.js --mode=development
```

PS：只要打包入口文件即可，依赖的文件会自动合并；

PS：development 表示开发模式，production 表示生产模式，压缩成一行；

二. 配置文件

1. 打包一次的命令太过于冗长，且特别容易出错，所以要对这些路径参数进行存储；
2. 我们可以创建 package.json 文件，来配置 scripts 属性来部署生成路径；

```
npm init -y          //生成配置文件命令
```

3. 然后在 scripts 属性里添加子属性：build，属性值具体如下：

```
"build" : "webpack ./02/src/index.js -o ./02/dist/bundle.js
                                                --mode=development"
```

4. 然后使用 npm 命令自动执行这个属性值的路径；

```
npm run build
```

5. 如果是比较简单的打包，package.json 还行，当参数越发复杂维护将变得困难；
6. Webpack 还提供了一个 webpack.config.json 配置文件，解决这个问题；
7. 由于我们是子目录 01,02 这种，完全可以直接存放子目录中即可；
8. 也就是说，配置文件不一定非要存放在根目录，可以根据自己目录结构进行调整；

```
/*
    webpack 构建时，会自动读取此文件
 */

//获取当前路径
const path = require('path');

module.exports = {
    //入口文件
    entry: './src/index.js',

    //出口文件
    output: {
        //文件名
        filename : 'bundle.js',
        //路径，要绝对路径
        path : path.resolve(__dirname, './dist')
    },

    //生成模式
    mode : "development"
};
```

PS：在哪个目录，就进入哪个目录，直接执行命名：webpack，即执行打包；