webpack

使用 webpack 打包 TS 项目是推荐做法，webpack 可以将多个模块文件打包到一个文件中。当然不只处理 TS、也可以处理 sass、less、vue、react 等文件。

#初始环境

创建 package.json 用于管理项目及扩展包

```
yarn init -y
```

下面安装安装打包需要的软件

```
yarn add -D typescript webpack webpack-cli ts-loader webpack-dev-server
```

- webpack 核心文件、webpack-cli 命令行工具，用于实现 webpack 核心功能
- ts-loader 用于处理 typescript 的 ts-loader 软件
- webpack-dev-server 在开发阶段启动访问地址为 localhost:3000 服务，修改时网页自动刷新的热更新效果

接下来创建 typescript 的配置文件 tsconfig.js

```
tsc --init
```

#目录结构

首先看一下项目的目录结构

```
├── public
|   ├── dist
|   |   └── app.js     最终编译文件
|   └── index.html                模板文件
├── package.json                          项目配置
├── src                                                  项目源文件
```

```
|   └── index.ts                          项目入口文件
├── tsconfig.json                         typescript 配置文件
├── webpack.config.js   webpack 配置文件
└── yarn.lock                                                  锁定项目安装后的软件包版本，其他人安装我们项目时可以使用相同的版本，保证正常运行

```

#文件说明

下面介绍项目中的主要文件内容

#webpack.config.js

webpack.config.js

```
const path = require('path')

module.exports = {
    //程序打包的起点即入口文件
    entry: './src/index.ts',

    //配置 webpack 如何去输出,webpack 会将打包到一个文件中，从而减少网络请求
    output: {
        //最终编译文件与目录
        filename: 'app.js',
        //输出目录的绝对路径，所以要使用 path 模块的 resolve 方法处理,node.js已经内置了path模块
        //参数一__dirname为当前文件的路径，参数二dist为与参数一组合后的目录
        path: path.resolve(__dirname, 'public/dist'),
        //使用webpack-dev-server运行编译时，即热更新时的静态文件前缀
        //因为编译内容是存在内存中的，不会真正创建文件，设置publicPath值决定以什么路径引用文件
        //值是相对于 public/index.html 的路径，需要以 / 结尾
        //需要安装 webpack-dev-server
        publicPath: '/dist/',
    },

    //项目中的不同类型模块的处理规则
    module: {
        rules: [
            {
```

```
                //test定义文件检测，满足后才处理，下面定义文件是否为 ts 或tsx，则满中本规则，然后进行处理
                test: /\.tsx?$/,
                //use定义处理器，下面是使用 ts-loader 将 ts或 tsx 文件编译成 javascript
                use: 'ts-loader',
                //exclude排除node_modules 目录中的文件处理
                exclude: /node_modules/,
            },
        ],
    },

    //配置模块如何解析
    resolve: {
        //在使用 import Hd from Util 等代码时，如果不添加扩展名，webpack 按以下扩展名顺序匹配文件
        extensions: ['.tsx', '.ts', '.js'],
    },
}

```

#package.json

package.json 项目配置文件

```
{
    "name": "houdunren.com",
    "version": "1.0.0",
    "main": "index.js",
    "license": "MIT",
    "scripts": {
        "dev": "webpack-dev-server --mode=development",
        "build": "webpack --mode=production"
    },
    "devDependencies": {
        "ts-loader": "^9.2.6",
        "typescript": "^4.4.3",
        "webpack": "^5.56.0",
        "webpack-cli": "^4.8.0",
        "webpack-dev-server": "^4.3.0"
```

```
    }
}

```

通过运行以下命令查看不同的执行结果

- development 是开发阶段
- production 是生产环境使用，会对代码进行更好的压缩与优化

```
yarn run webpack --mode=development
yarn run webpack --mode=production

```

#tsconfig.json

tsconfig.json 是对 typescript 的配置，TypeScript 中使用 ES6 模块，并编译 js 为 ES6

> 更多有关 TS 的课程已经在后盾人网站 (opens new window)录制了，大家可以去线上学习

```
...
"target": "ES6",
"module": "ES6"
....

```

#编译测试

修改 public/index.html 内容如下

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <title>后盾人-向军大叔</title>
    </head>
    <body>
        <h2>houdurnen.com</h2>
        <script src="./dist/app.js"></script>
    </body>
</html>
```

```
12
```

然后在命令行执行 `yarn run dev`，会看到以下执行结果

```
$ webpack-dev-server --mode=development
<i> [webpack-dev-server] Project is running at:
<i> [webpack-dev-server] Loopback:
http://localhost:8080/
<i> [webpack-dev-server] Loopback:
<i> [webpack-dev-server] On Your Network (IPv4):
http://192.168.31.156:8080/
<i> [webpack-dev-server] On Your Network (IPv4):
<i> [webpack-dev-server] On Your Network (IPv6): http://[fe80::1]:8080/
<i> [webpack-dev-server] Content not from webpack is served from
'/Users/hd/test/ts/public' directory

```

在浏览器中访问 http://localhost:8080/ ，这时修改内容后，浏览器也会同步热更新
项目开发完毕后可以执行 `yarn run build` 生成体积更小的编译文件，文件地址为 public/app.js

#项目示例

下面来体验一下 typescript 结合 ES6 的模块的开发过程
index.html 项目访问主文件

```
<!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>后盾人</title>
        <script src="dist/app.js"></script>
    </head>
    <body></body>
</html>

```

src/user.ts user 模块文件

```
export namespace User {
```

```
    export let name: string = '后盾人'
}
export namespace Member {
    let name: string = 'houdunren.com'
}

```

src/index.ts 项目入口文件

```
import { User, Member } from './user'
console.log(User.name);

```

最后执行编译命令生成 dist/bundle.js 编译文件，然后再浏览器中访问 index.html

```
yarn run build

```