Webpack

静态模块打包工具

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

目录

Contents

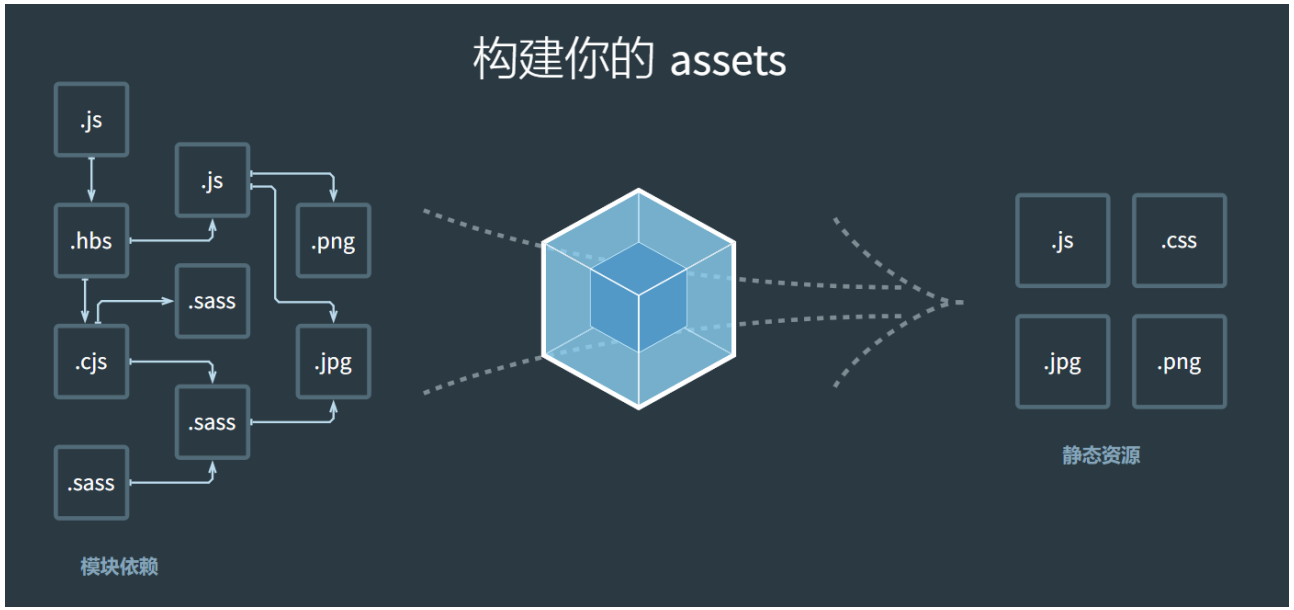
什么是 Webpack?

定义：

> 本质上，**webpack** 是一个用于现代 JavaScript 应用程序的 *静态模块打包工具*。当 webpack 处理应用程序时，它会在内部从一个或多个入口点构建一个 依赖图(dependency graph)，然后将你项目中所需的每一个模块组合成一个或多个 *bundles*，它们均为静态资源，用于展示你的内容。

静态模块：指的是编写代码过程中的，html，css，js，图片等固定内容的文件

图解：



注意：　只有和入口有直接/简介的引入关系的模块，才会被打包

为什么要学 Webpack?

原因：把静态模块内容，压缩，整合，转译等（前端工程化）

- ✓ 把 less / sass 转成 css 代码
- ✓ 把 ES6+ 降级成 ES5
- ✓ 支持多种模块文件类型，多种模块标准语法

问题：为何不学 vite ？

因为：很多项目还是基于 Webpack 来进行构建的，所以还是要掌握 Webpack 的使用

使用 Webpack

需求：封装 utils 包，校验用户名和密码长度，在 index.js 中使用，使用 Webpack 打包

步骤：

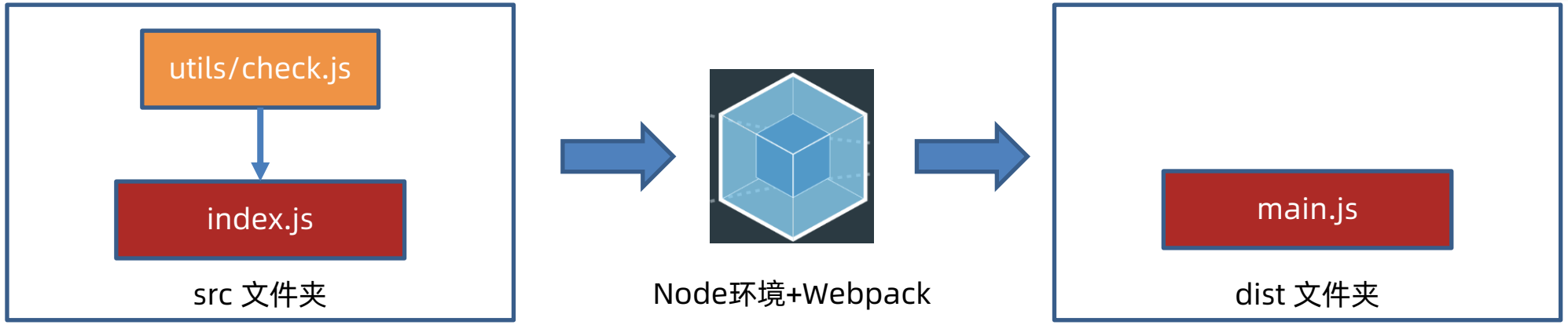
1. 新建项目文件夹，初始化包环境
2. 新建 src 源代码文件夹（书写代码）
3. 下载 webpack webpack-cli 到项目（版本独立）
4. 项目中运行工具命令，采用自定义命令的方式（局部命令）
5. 自动产生 dist 分发文件夹（压缩和优化后，用于最终运行的代码）

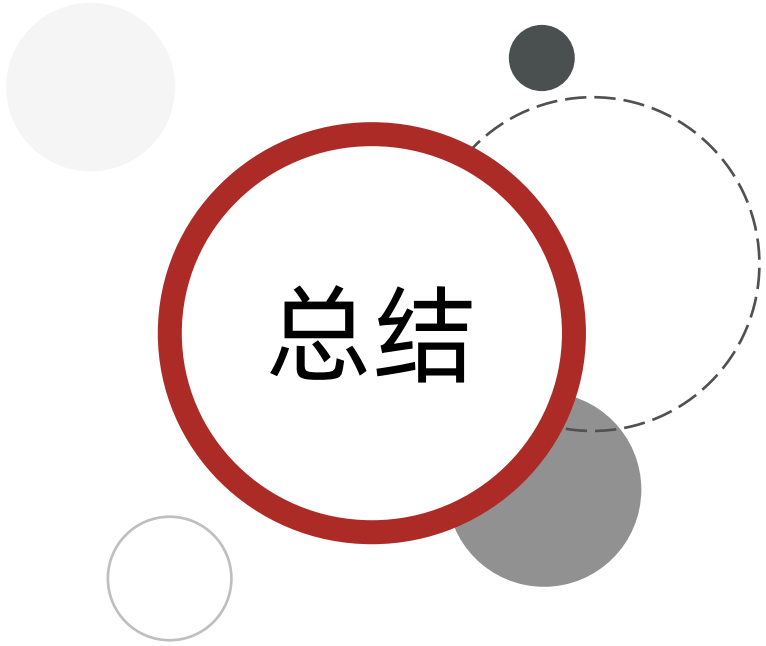
package.json

```
npm i webpack webpack-cli --save-dev
```

```
"scripts": {
  "build": "webpack"
},
```

```
npm run build
```





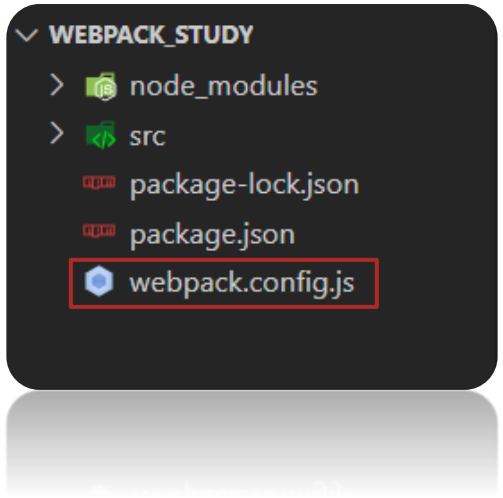
1. Webpack 有什么用？
   - 压缩，转译，整合，打包我们的静态模块
2. Webpack 怎么用？
   - 初始化环境，编写代码
   - 安装，配置自定义命令
   - 打包体验
3. 如何运行 package.json 里的自定义命令？
   - npm run 自定义命令
4. Webpack 默认入口和出口？
   - src/index.js 和 dist/main.js

修改 Webpack 打包入口和出口

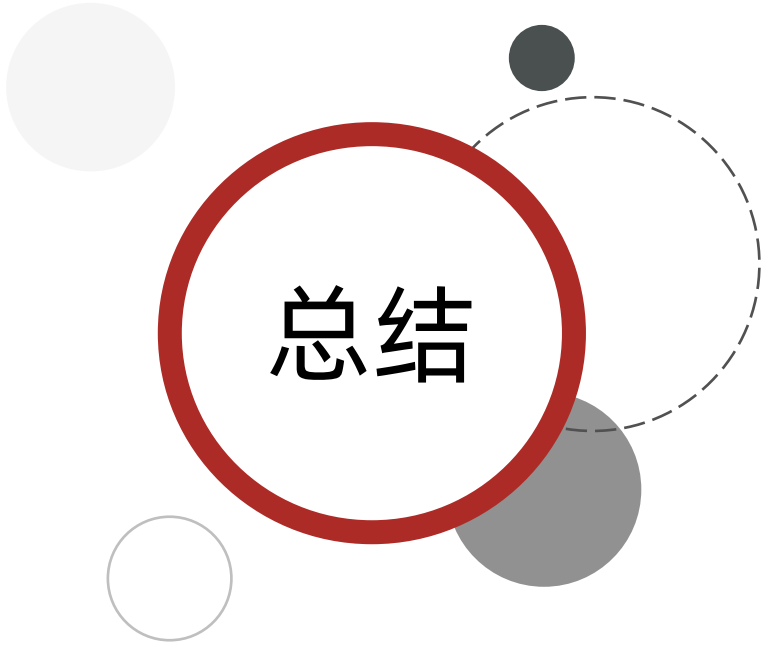
Webpack 配置：影响 Webpack 打包过程

步骤：

1. 项目根目录，新建 webpack.config.js 配置文件
2. 导出配置对象，配置入口，出口文件路径
3. 重新打包观察




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

module.exports = {
  entry: path.resolve(__dirname, 'src/main.js'),
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'app.js',
  },
};
```



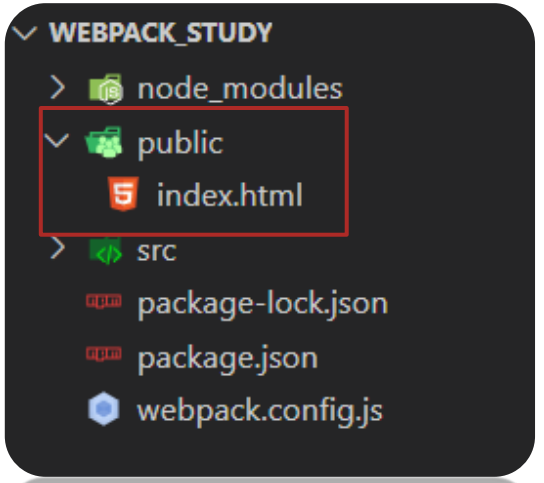
1. 如何影响 Webpack 打包过程？
   - 查文档，新建配置文件和配置属性

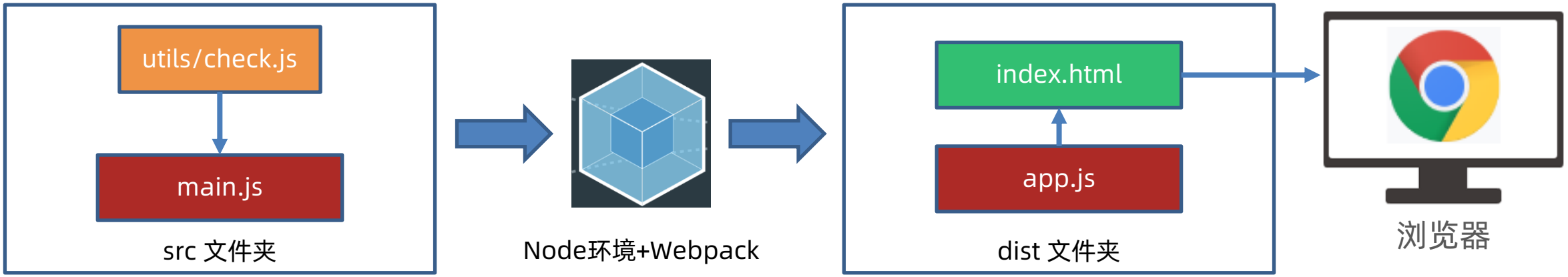
案例 注册用户 - 长度判断

需求：点击注册按钮，判断用户名和密码长度是否符合要求

步骤：

1. 新建 public/index.html 准备网页模板
2. 核心代码写在 src/main.js
3. 打包 js 代码
4. 手动复制 index.html 到 dist 下，引入打包后的 js，运行 dist/index.html 查看效果





自动生成 html 文件

插件 html-webpack-plugin： 在 Webpack 打包时生成 html 文件

步骤：

1. 下载 html-webpack-plugin 本地软件包

```
npm i html-webpack-plugin --save-dev
```

2. 配置 webpack.config.js 让 Webpack 拥有插件功能
3. 指定以 public/index.html 为模板复制到 dist/index.html，并自动引入其他打包后资源

```
// ...
const HtmlWebpackPlugin = require('html-webpack-plugin')

module.exports = {
  // ...
  plugins: [new HtmlWebpackPlugin({
    template: path.resolve(__dirname, 'public/index.html')
  })]
};
```

打包 css 模块

加载器 css-loader：解析 css 代码

加载器 style-loader：把解析后的 css 代码插入到 DOM

```
npm i css-loader style-loader --save-dev
```

步骤：

1. 准备 css 文件引入到 src/main.js 中（压缩转译处理等）
2. 下载 css-loader 和 style-loader 本地软件包
3. 配置 webpack.config.js 让 Webpack 拥有该加载器功能
4. 打包后运行 dist/index.html 观察效果

```
// ...

module.exports = {
  // ...
  module: {
    rules: [
      {
        test: /\.css$/i,
        use: ["style-loader", "css-loader"],
      },
    ],
  },
};
```

注意：Webpack 默认只识别 js 和 json 文件内容

打包 less 模块

加载器 less-loader：把 less 代码编译为 css 代码，还需要依赖 less 软件包

步骤：

1. 准备 less 样式并引入到 src/main.js 中
2. 下载 less 和 less-loader 本地软件包
3. 配置 webpack.config.js 让 Webpack 拥有功能
4. 打包后运行 dist/index.html 观察效果

```
npm i less less-loader --save-dev
```

```
// ...

module.exports = {
  // ...
  module: {
    rules: [
      // ...
      {
        test: /\.less$/i,
        use: ["style-loader", "css-loader", "less-loader"]
      }
    ],
  },
};
```

打包图片

资源模块：Webpack5 内置了资源模块的打包，无需下载额外 loader

步骤：

1. 准备图片素材到 src/assets 中
2. 在 index.less 中给 body 添加背景图
3. 在 main.js 中给 img 标签添加 logo 图片
4. 配置 webpack.config.js 让 Webpack 拥有打包图片功能
5. 打包后运行 dist/index.html 观察效果

```
// ...

module.exports = {
  // ...
  module: {
    rules: [
      // ...
      {
        test: /\.(png|jpg|jpeg|gif)$/i,
        type: 'asset',
        generator: {
          filename: 'assets/[hash][ext]'
        }
      }
    ],
  },
};
```

注意：判断临界值默认为 8KB

- ✓ 小于 8KB 文件会被转成 data URI（base64字符串）
- ✓ 大于 8KB 文件会被复制到 dist 下

babel 编译器

```
npm i babel-loader @babel/core @babel/preset-env -D
```

babel 定义：是一个 JavaScript 语法编译器，将采用 ECMAScript 2015+ 语法编写的代码转换为向后兼容的 JavaScript 语法，以便能够运行在当前和旧版本的浏览器或其他环境中

babel-loader：让 Webpack 可以使用 babel 转译 JavaScript 代码

步骤：

1. 编写一段映射数组元素，每个数值 + 1 的代码（要求用箭头函数）
2. 下载 babel babel-loader core 本地软件包
3. 配置 webpack.config.js 让 Webpack 拥有功能
4. 打包运行后 dist/index.html 观察效果

模块	作用
@babel/core	Js 编译器，分析代码
@babel/preset-env	babel 预设，规则
babel-loader	让 webpack 翻译 js 代码

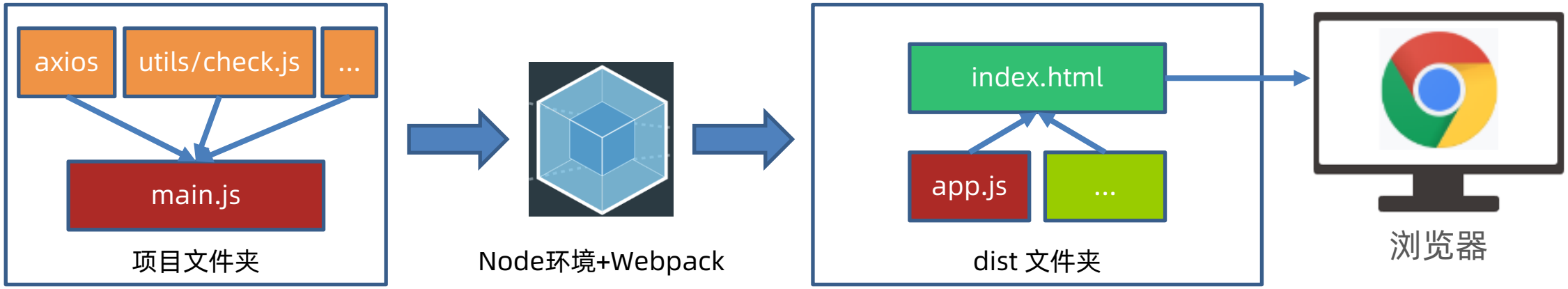
```
// ...

module.exports = {
  // ...
  module: {
    rules: [
      // ...
      {
        test: /\.m?js$/,
        exclude: /(node_modules|bower_components)/,
        use: {
          loader: 'babel-loader',
          options: {
            presets: ['@babel/preset-env']
          }
        }
      }
    ],
  },
};
```

案例 注册用户 - 完成功能

需求：点击注册按钮，提交用户名和密码到服务器完成注册

步骤：

1. 使用 npm 下载 axios
2. 引入到 src/main.js 中编写业务实现
3. 打包后运行 dist/index.html 观察效果



Webpack 开发服务器

webpack-dev-server：快速开发应用程序

作用：启动 Web 服务，打包输出源码在内存，并检测代码变化热更新到网页

```
npm i webpack-dev-server --save-dev
```

步骤：

1. 下载 webpack-dev-server 软件包到当前项目
2. 配置自定义命令，并设置打包的模式为开发模式
3. 使用 npm run dev 来启动开发服务器，试试热更新效果

```
"scripts": {
  "build": "webpack",
  "dev": "webpack serve --mode=development"
},
```

打包模式

打包模式：告知 Webpack 使用相应模式的内置优化

分类：

模式名称	模式名字	特点
开发模式	development	调试代码，实时加载，模块热替换等
生产模式	production	压缩代码，资源优化，更轻量等

```
// ...

module.exports = {
  // ...
  mode: 'production'
};
```

设置：

方式1：在 webpack.config.js 配置文件设置 mode 选项

方式2：在 package.json 命令行设置 mode 参数

```
"scripts": {
  "build": "webpack --mode=production",
  "dev": "webpack serve --mode=development"
},
```

注意：命令行设置的优先级高于配置文件中的，推荐用命令行设置

开发环境调错 - source map

source map：可以准确追踪 error 和 warning 在原始代码的位置

问题：代码被压缩和混淆，无法正确定位源代码位置（行数和列数）

设置：webpack.config.js 配置 devtool 选项

```
// ...

module.exports = {
  // ...
  devtool: 'inline-source-map'
};
```

inline-source-map 选项：把源码的位置信息一起打包在 js 文件内

注意：source map 仅适用于开发环境，不要在生产环境使用（防止被轻易查看源码位置）

解析别名 alias

解析别名：配置模块如何解析，创建 import 或 require 的别名，来确保模块引入变得更简单

例如：

原来路径如下：

```
import {checkUserName, checkPassword} from '../src/utils/check.js'
```

配置解析别名：在 webpack.config.js 中设置

```
// ...

module.exports = {
  // ...
  resolve: {
    alias: {
      MyUtils: path.resolve(__dirname, 'src/utils'),
      '@': path.resolve(__dirname, 'src'),
    },
  }
};
```

```
import {checkUserName, checkPassword} from 'MyUtils/check.js'
```

```
import {checkUserName, checkPassword} from '@/utils/check.js'
```

传智教育旗下高端IT教育品牌