webpack

模块打包工具

目录

Contents

学习目标

Learning Objectives

1. 能够说出webpack的作用
2. 能够掌握webpack基本配置
3. 能够知道webpack开发服务器运作过程
4. 能够熟练webpack中文文档

目录

Contents

webpack基本概念

问题1: 以前写完的网站, 文件很多, 体积很大?

问题2: 有没有一种自动整合, 压缩, 剔除无用代码技术?

1.0_为什么要学习webpack

toutiao

大小:	143 MB (150,646,067 字节)
占用空间:	190 MB (199,401,472 字节)
包含:	36,071 个文件，4,896 个文件夹

dist

大小:	2.58 MB (2,707,998 字节)
占用空间:	2.64 MB (2,772,992 字节)
包含:	38 个文件，3 个文件夹

是不是让浏览器加载资源更快呀?

小结

我们为什么学习webpack

1. 减少文件数量
2. 缩减代码体积
3. 提高浏览器打开的速度

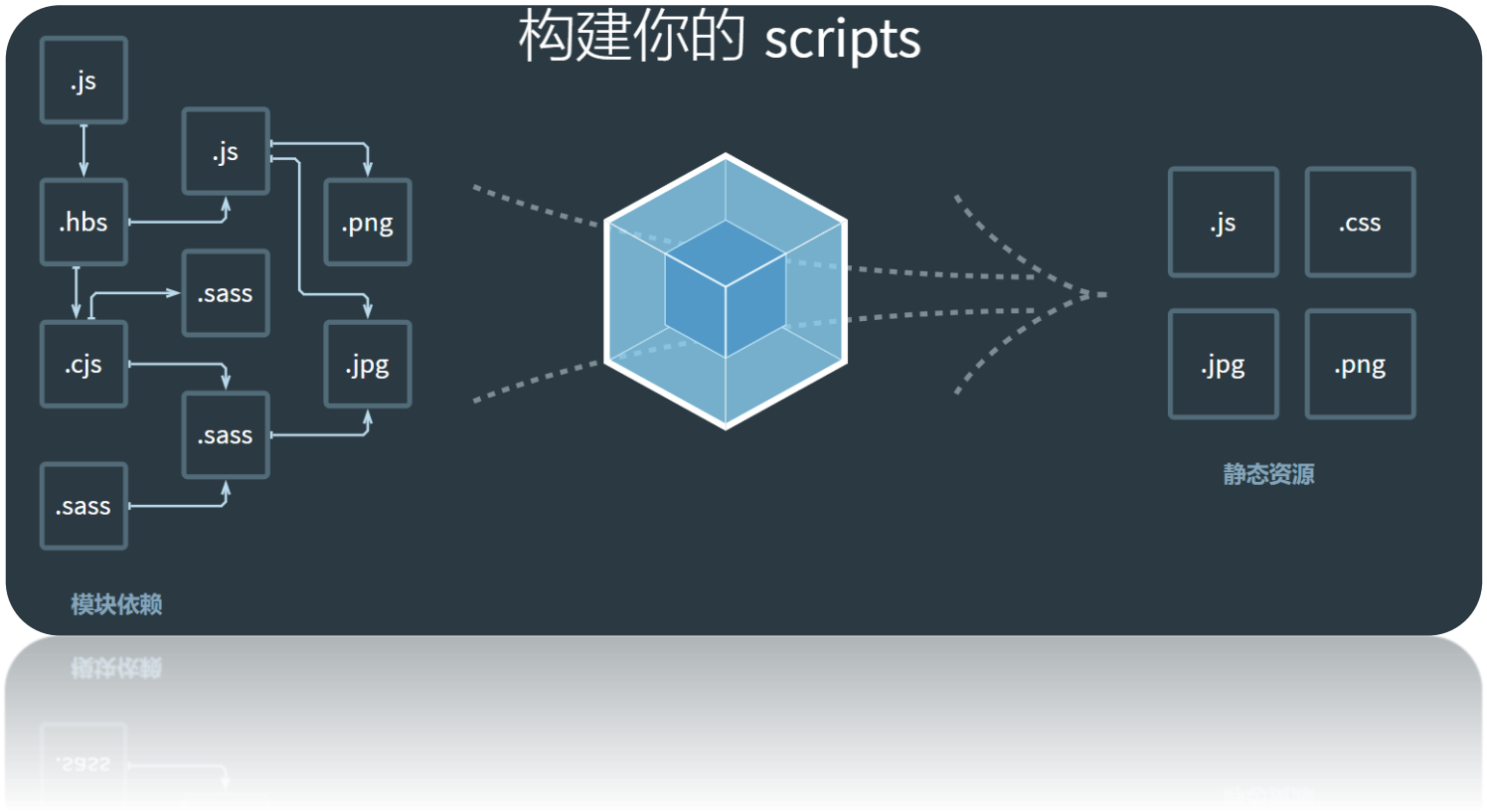
问题1: 我们手动能否压缩代码呢?

问题2: 能否找个模块来, 翻译, 压缩, 打包合并代码?

1.1_webpack基本概述

webpack本质是, 一个第三方模块包, 用于分析, 并打包代码

- 支持所有类型文件的打包
- 支持less/sass => css
- 支持ES6/7/8 => ES5
- 压缩代码, 提高加载速度



小结

什么是webpack呢? 作用是什么? 目的是?

1. 它是一个模块包
2. 识别代码, 翻译, 压缩, 整合打包
3. 提高打开网站的速度

目录

Contents

webpack使用步骤

问题1: webpack既然是个模块包, 需要下载吗?

问题2: 在一个文件夹里下载包? 需要不需要有人记录呢?

2.0_环境准备

1. 初始化包环境

```
yarn init
```

2. 安装依赖包

```
yarn add webpack webpack-cli -D
```

3. 配置scripts（自定义命令）

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

使用webpack需要做哪些准备工作?

1. 初始化文件夹包环境, 得到package.json文件
2. 下载webpack等模块包
3. 在package.json自定义命令, 为打包做准备

问题: 如何把2个js文件, 打包整合到一起并压缩呢?

2.1_案例: webpack基础使用

案例

webpack基础使用

需求： 2个js文件 ->打包成1个js文件

分析：

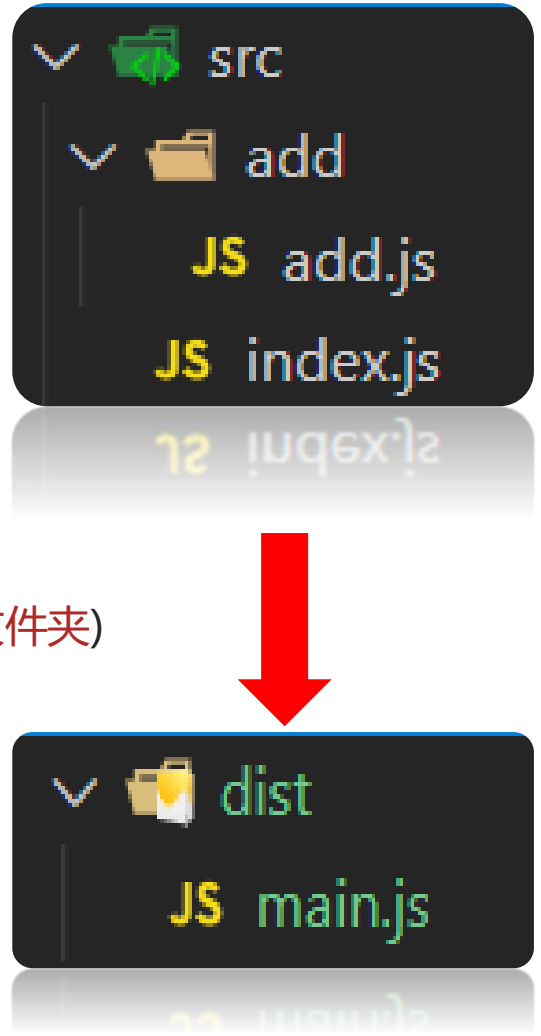
①：新建src下的资源

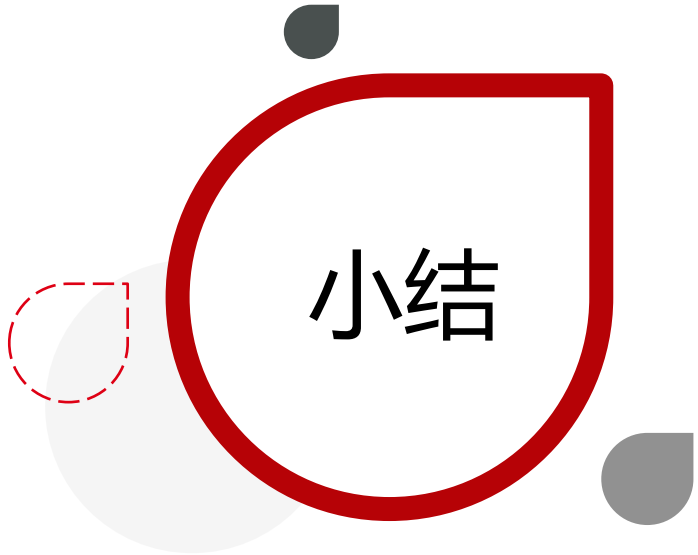
②：add.js – 定义求和函数并导出

③：index.js – 引入add模块并使用函数输出结果

④：执行 `yarn build` 自定义命令, 进行打包 (确保终端路径在src的父级文件夹)

⑤：打包后默认生成dist和main.js, 观察其中代码





webpack如何使用

1. 默认src/index.js – 打包入口文件
2. 需要引入到入口的文件才会参与打包
3. 执行package.json里build命令, 执行webpack打包命令
4. 默认输出dist/main.js的打包结果

问题: 如果以后代码增加了, 如何再打包呢?

2.2_案例: webpack再次打包

案例

webpack再次打包

需求：代码更多后, 如何打包呢?

分析：

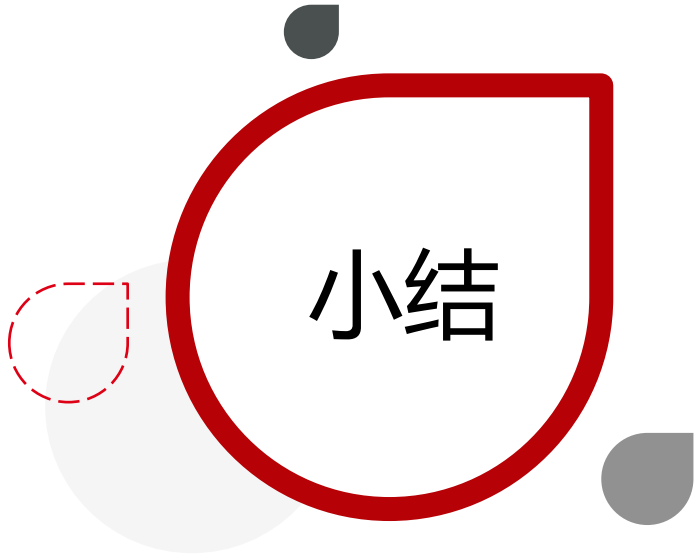
①：src下新建tool/tool.js

②：定义数组求和函数导出

③：index.js – 引入tool模块的函数并使用, 打印结果

④：执行 `yarn build` 自定义命令, 进行打包 (确保终端路径在src的父级文件夹)

⑤：打包后默认生成dist和main.js, 观察其中代码



代码增加后, 如何打包呢?

1. 确保在src/index.js引入和使用
2. 重新执行yarn build打包命令

目录

Contents

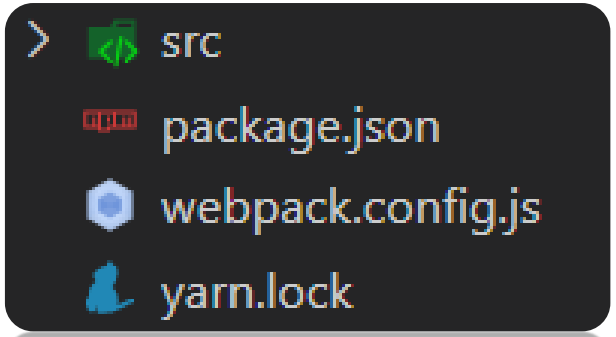
webpack的配置

问题: 能否更改webpack打包默认的入口和出口?

3.0_webpack-入口和出口

配置文档: https://webpack.docschina.org/concepts/#entry

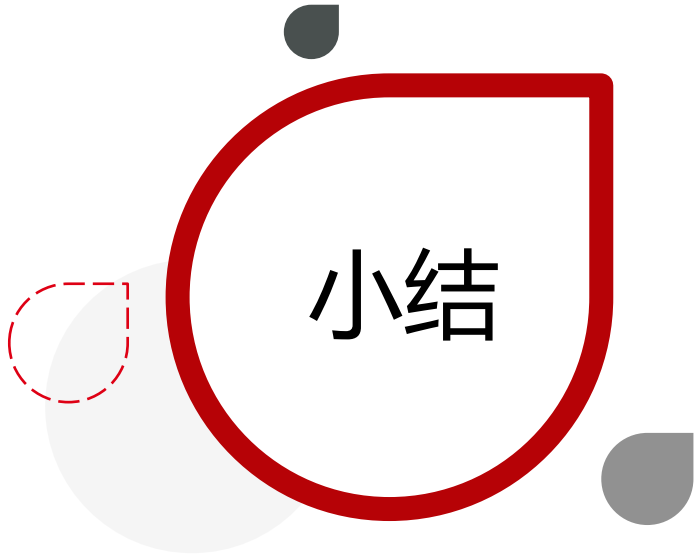
1. 新建webpack.config.js
2. 填入配置
3. 修改入口文件名
4. 打包观察效果




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

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

注意: webpack基于node, 所以导出, 遵守CommonJS规范

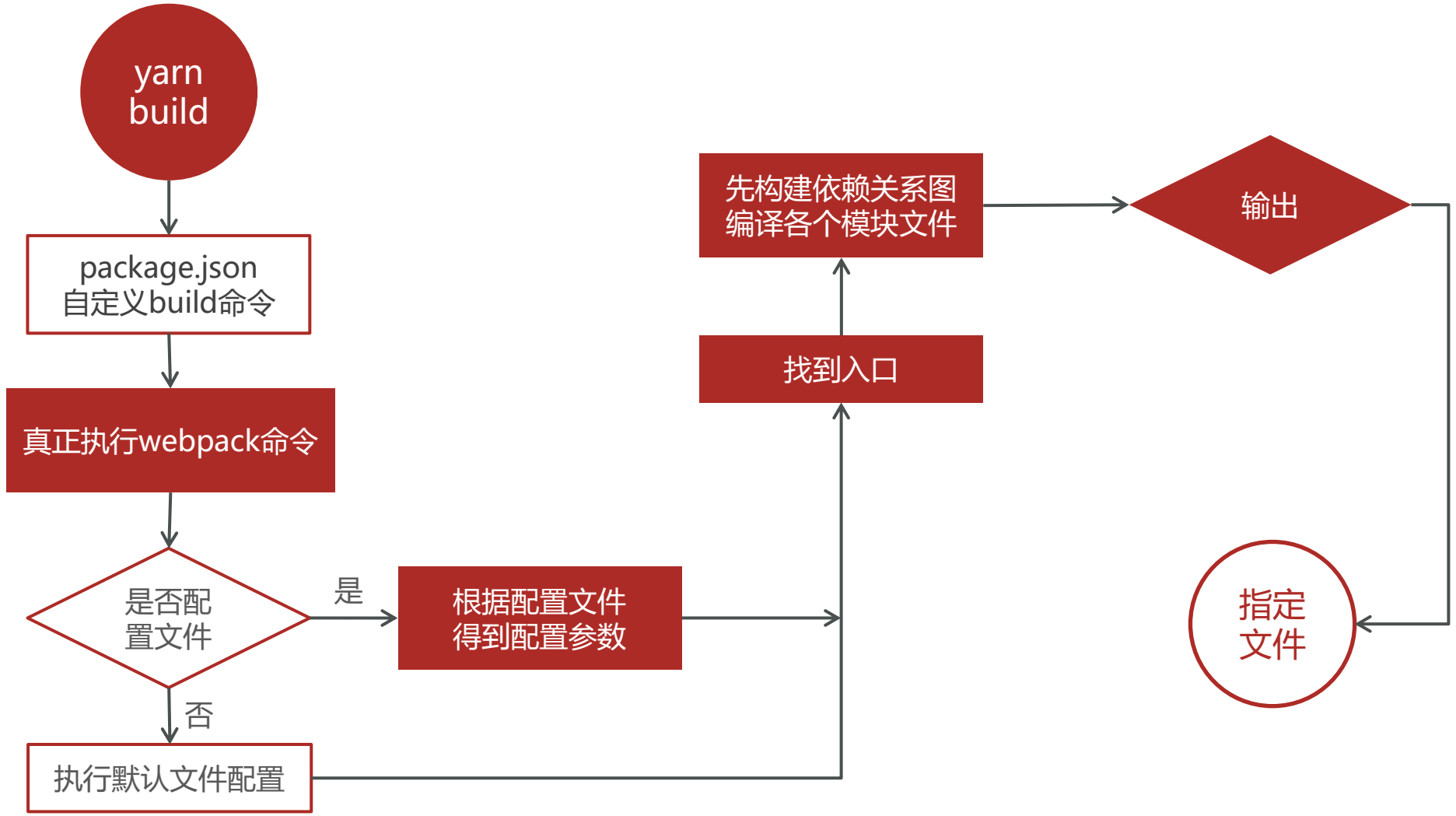


如何修改webpack入口和出口

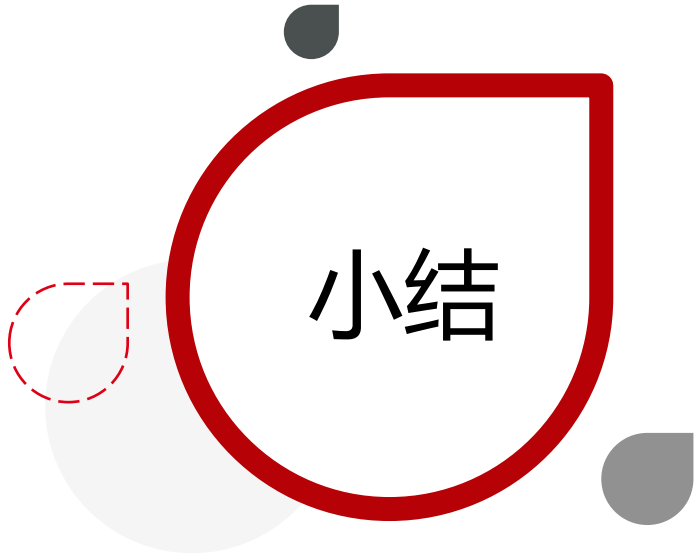
1. 新建webpack.config.js (webpack默认配置文件名)
2. 通过entry设置入口文件路径
3. 通过output对象设置出口路径和文件名

问题: 从`yarn build`开始都走了哪些流程呢?

3.1_yarn build执行流程图



重点: 所有要被打包的资源都要跟入口产生直接/间接的引用关系



一句话总结下`yarn build`后干了什么

1. 执行webpack命令, 找到配置文件, 入口和依赖关系,

   打包代码输出到指定位置

问题: 打包后的js能不能应用于网页上呢?

3.2_案例-隔行变色

案例

webpack打包代码, jquery实现功能

需求：新建项目, yarn下载jquery, 然后模块化引入到js中, 编写jq代码实现隔行变色

效果如下:

- 我是第1个li
- 我是第2个li
- 我是第3个li
- 我是第4个li
- 我是第5个li
- 我是第6个li
- 我是第7个li
- 我是第8个li
- 我是第9个li

3.2_案例-隔行变色

案例

webpack打包代码, jquery实现功能

步骤:

①: 从0准备环境, 初始化包环境, 下载webpack和webpack-cli包, 配置自定义命令build

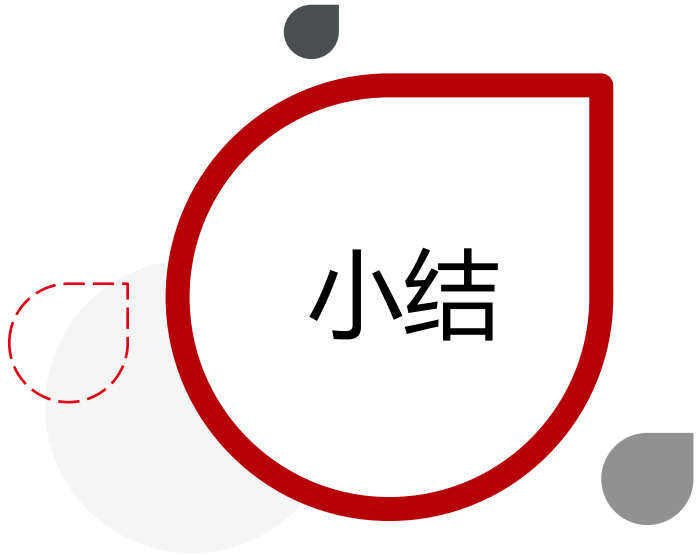
②: yarn下载jquery, 新建public/index.html,准备一些li标签

③: src/main.js 引入jquery, 编写功能代码

④: 执行打包命令

⑤: 复制public/index.html到dist/, 然后引入打包后的js, 运行网页观察效果

重点: webpack打包后的js引入到html中使用



用yarn下的包, 如何作用在前端项目中?

1. 借助webpack, 把模块和代码打包
2. 把js文件引入到html执行查看效果

问题: 打包后还得自己手动建立html文件, 累不累?

3.3_html-webpack-plugin插件

配置文档: https://webpack.docschina.org/plugins/html-webpack-plugin/

1. 下载插件

```
yarn add html-webpack-plugin -D
```

2. webpack.config.js添加配置

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

module.exports = {
    // ...其他配置
    plugins: [
        new HtmlWebpackPlugin({
            template: './public/index.html'
        })
    ]
}
```

重要: webpack就像一个人, webpack.config.js是人物属性, 给它穿什么装备它就干什么活

如何让webpack打包时, 自动生成html文件呢?

1. 依赖html-webpack-plugin插件, yarn下载此插件
2. 在webpack.config.js配置写入即可

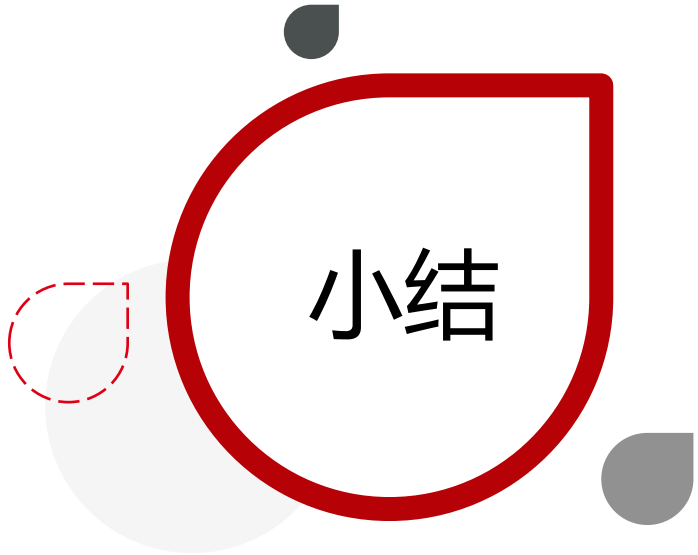
问题: 写个css文件, 去除li的圆点, 让webpack一起打包?

3.4_webpack打包css文件

目标：编写css代码, 让webpack打包

1. 新建 - src/css/index.css
2. 编写去除li圆点样式代码
3. (重要) 一定要引入到入口才会被webpack打包
4. 执行打包命令观察效果

报错: 因为webpack默认只能处理js文件



为什么webpack打包css文件会报错呢?

1. webpack默认只能识别js类型的文件

问题: 那如何才能让webpack识别和打包css文件呢?

3.5_webpack-使用加载器

css-loader 文档: https://webpack.docschina.org/loaders/css-loader/
style-loader文档: https://webpack.docschina.org/loaders/style-loader/

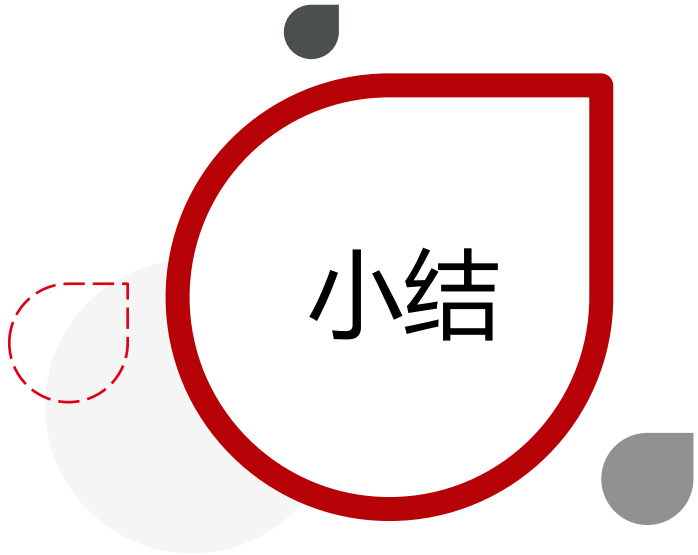
css-loader 让webpack能处理css类型文件

style-loader 把css插入到DOM中

1. 下载加载器
2. webpack.config.js配置
3. 打包观察效果

```
yarn add css-loader style-loader -D
```

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



webpack如何支持css打包？打包后样式在哪里？如何生效？

1. 依赖css-loader和style-loader
2. css代码被打包进js文件中
3. style-loader会把css代码插入到head下style标签内

问题: webpack能否识别打包less类型文件呢?

3.6_webpack处理less文件

less-loader文档: https://webpack.docschina.org/loaders/less-loader/

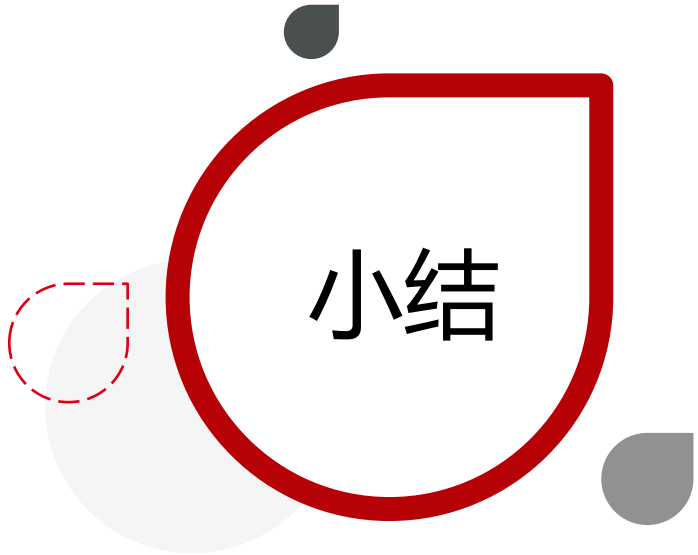
less-loader作用: 识别less文件

less 作用: 将less编译为css

1. 新建src/less/index.less – 设置li字体大小
2. 把index.less引入到入口处
3. 下载加载器来处理less文件
4. webpack.config.js针对less配置
5. 打包观察效果

```
yarn add less less-loader -D
```

```
module: {
    rules: [
      // ...其他配置
      {
        test: /\.less$/,
        use: [ "style-loader", "css-loader", 'less-loader']
      }
    ]
}
```



webpack如何支持less打包？需要注意什么？

1. 依赖less-loader和less模块包
2. 转换css后还需要css-loader和style-loader的处理

问题: 网页里不能没有图片吧, 那webpack能打包不?

3.7_webpack处理图片文件

1. 在src/assets/准备图片文件
2. 在index.less对body设置背景图片
3. 在入口导入图片文件, 设置到img标签插入到body
4. 打包观察效果

报错: 因为webpack无法自己处理图片文件

3.7_webpack处理图片文件

webpack5, 使用asset module技术实现字体文件和图片文件处理, 无需配置额外loader
文档: https://webpack.docschina.org/guides/asset-modules/

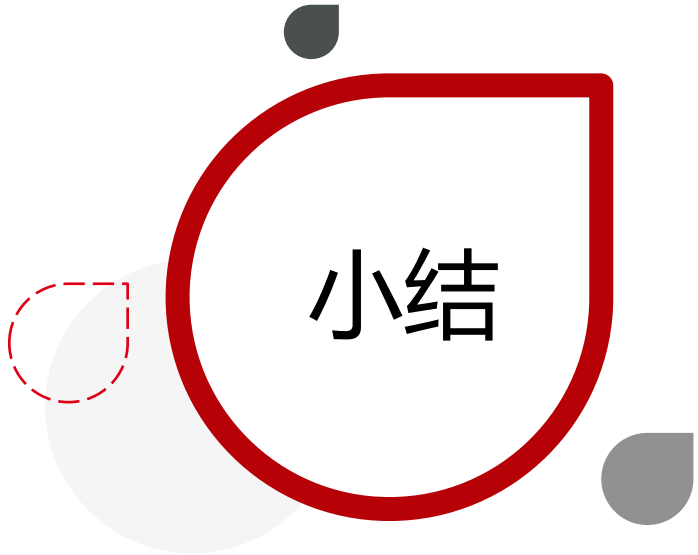
以前用url-loader和file-loader来处理

现在webpack.config.js – 针对图片文件设置type: “assets“

```
module: {
    rules: [
      // ...其他规则
      {
        test: /\.(png|jpg|gif|jpeg)$/i,
        type: 'asset'
      }
    ]
}
```

再次打包观察效果

小于8KB文件, 文件转base64打包在js中, 大于8KB, 文件自动命名输出到dist下, 打包观察效果和2图区别



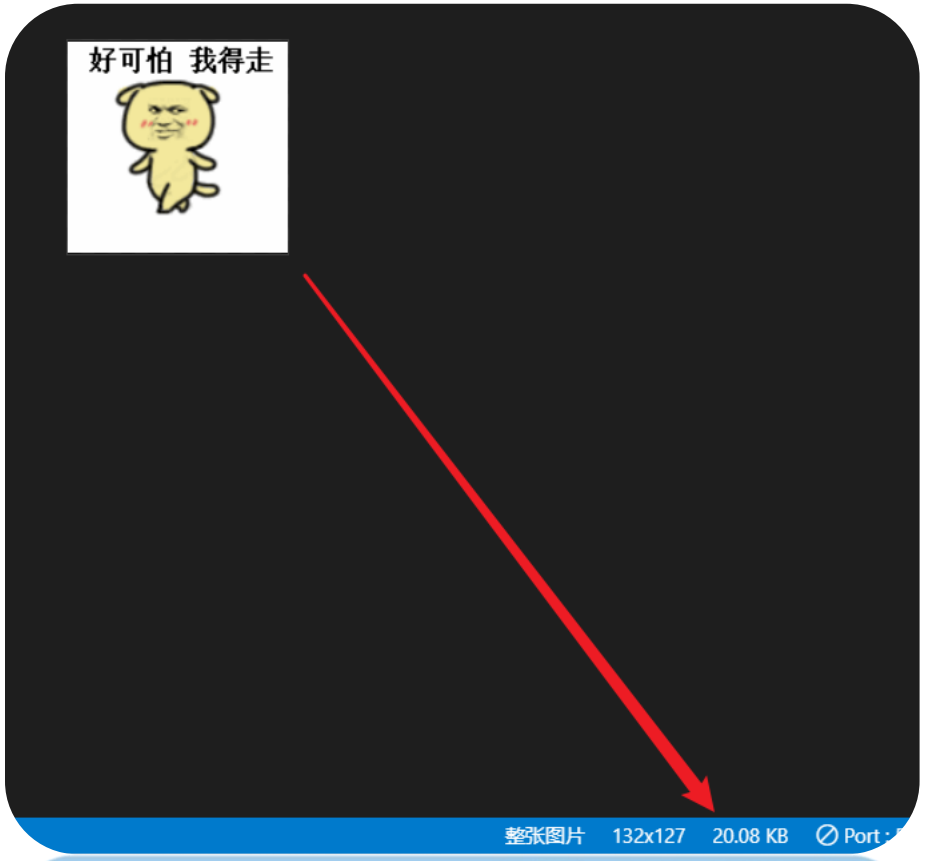
webpack如何支持图片打包? 对图片有哪2种处理方案?

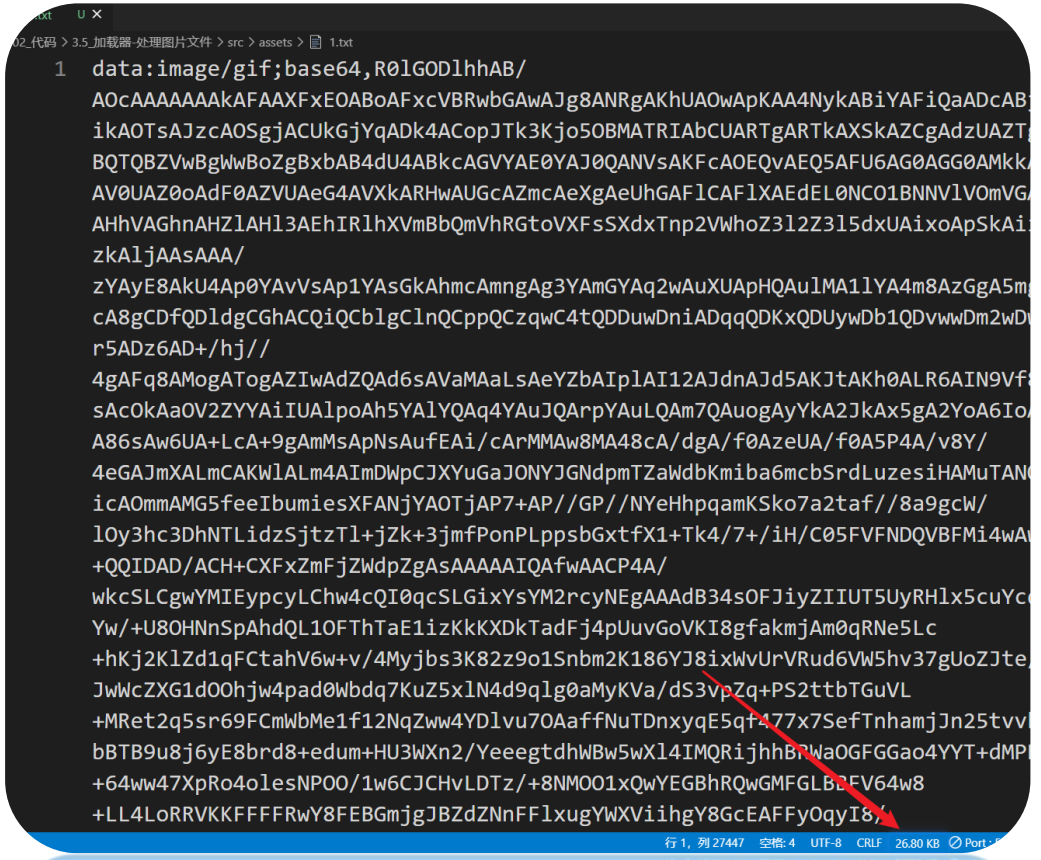
1. webpack5, 在rules里, 针对图片文件设置type: asset
2. 小于8KB转base64字符串进js里, 大于8KB输出文件

问题: 为什么8KB上下的图片要区分呢?

3.8_webpack加载文件优缺点

- 图片翻译成了base64, 放到了js文件中
  - 好处: 浏览器不用发请求了，直接可以读取, 速度快
  - 坏处: 图片太大，再转`base64`就会让图片的体积增大 30% 左右, 得不偿失





26.8KB

小结

图片转base64打包进js中好处和坏处是什么?

1. 好处是减少浏览器发送的请求次数, 读取图片速度快
2. 坏处是图片过大, 转base64占空间会多30%左右

问题: webpack能不能处理字体图标的字体文件呢?

3.9_webpack处理字体图标

1. src/assets 下 放入fonts字体相关文件夹(预习资料里)
2. src/main.js 引入 assets/fonts/iconfont.css
3. src/main.js 创建一个i标签, 使用字体图标标签添加到body上
4. 添加针对字体文件的加载器规则, 使用asset/resource(直接输出文件并配置路径)
5. 打包后运行网页观察效果

```
module: {
  rules: [
    // ...其他配置
    { // webpack5默认内部不认识这些文件，所以当做静态资源直接输出即可
      test: /\.(eot|svg|ttf|woff|woff2)$/,
      type: 'asset/resource',
      generator: {
        filename: 'font/[name].[hash:6][ext]'
      }
    }
  ]
}
```

小结

webpack如何处理图标字体文件呢?

1. 在webpack.configjs的rules里针对字体图标文件类型设置asset/resource, 直接输出到dist下

问题1: 我们以前写完的代码, 是如何做兼容低版本浏览器的?

问题2: 写的太多了, 手动修改, 工作量太大了..

3.10_webpack对JS语法降级

babel官网: https://www.babeljs.cn/
babel-loader文档: https://webpack.docschina.org/loaders/babel-loader/

babel: 一个javascript编译器, 把高版本js语法降级处理输出兼容的低版本语法

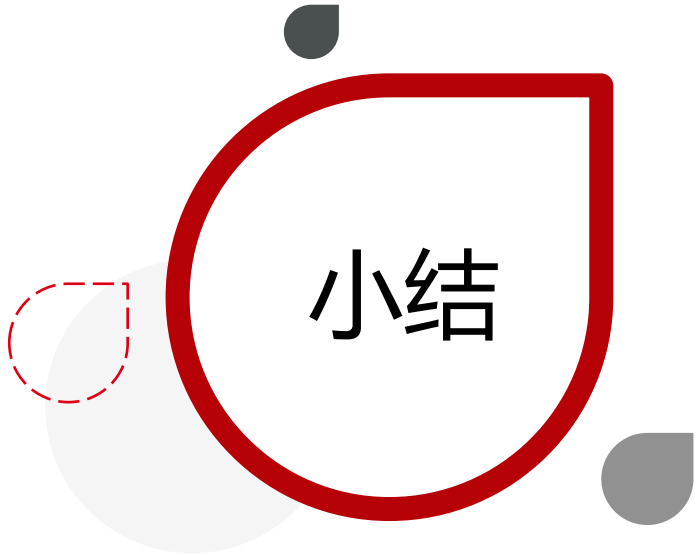
babel-loader: 可以让webpack转译打包的js代码

1. 在src/main.js – 定义箭头函数, 并打印箭头函数变量 (千万不能调用, 为了让webpack打包函数, 看降级效果)
2. 下载加载器

```
yarn add babel-loader @babel/
```

3. 配置到webpack.config.js上
4. 打包观察是否降级

```
module: {
  rules: [
    // ...其他配置
    {
      test: /\.js$/,
      exclude: /(node_modules|bower_components)/,
      use: {
        loader: 'babel-loader',
        options: {
          presets: ['@babel/preset-env']
        }
      }
    }
  ]
}
```



webpack如何帮助我们降低js版本语法的呢?

1. 借助babel-loader和babel编译器, 给webpack配置上

目录

Contents

webpack开发服务器

问题1: 每次写完代码, 想看效果, 是不是都要重新打包?

问题2: 累不累, 麻烦不麻烦, 浪费宝贵的开发时间不?

4.0_webpack开发服务器

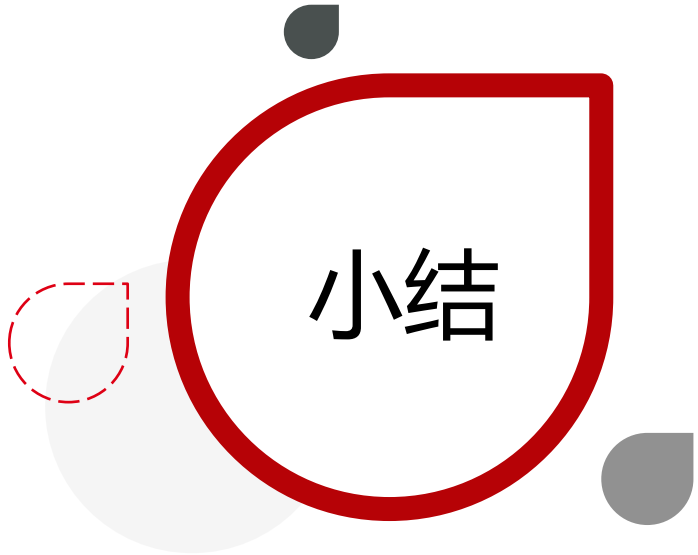
问题: 每次修改代码, 重新 yarn build 打包, 才能看到最新的效果, 实际工作中, 打包 yarn build 非常费时 (30s - 60s)

原因:

- 从0构建依赖
- 磁盘读取对应的文件到内存, webpack开始加载
- 再用对应的 loader 进行处理
- 将处理完的内容, 输出到磁盘指定目录

解决: 起一个开发服务器, 缓存一些已经打包过的内容, 只重新打包修改的文件, 最终运行在内存中给浏览器使用

总结: webpack开发服务器, 把代码运行在内存中, 自动更新, 实时返回给浏览器显示



为什么要使用webpack开发服务器呢?

1. 打包在内存中, 速度快
2. 自动更新打包变化的代码, 实时返回给浏览器显示

问题: webpack开发服务器这么香, 怎么用啊?

4.1_webpack-dev-server模块使用

webpack-dev-server文档: https://webpack.docschina.org/configuration/dev-server/

1. 下载模块包

```
yarn add webpack-dev-server -D
```

2. 自定义webpack开发服务器启动命令serve – 在package.json里

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

3. 启动当前工程里的webpack开发服务器

```
yarn serve
```

4. 重新编写代码, 观察控制台和浏览器是否自动打包和更新

小结

如何使用webpack开发服务器实时打包我们代码呢?

1. 确保下载了webpack-dev-server到工程中
2. 在package.json-配置自定义命令, 然后启动即可
3. webpack-dev-server会给我们一个地址+端口浏览器访问即可看到在内存中打包的index.html页面

问题1: webpack开发服务器能不能改变它的端口号呢?

问题2: 比如我的幸运数字是3000

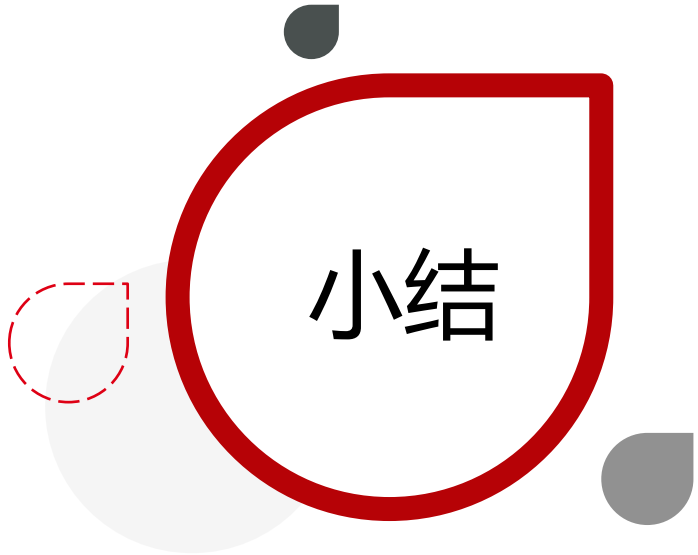
4.2_webpack-dev-server配置

webpack-dev-server配置文档: https://webpack.docschina.org/configuration/dev-server/#devserverafter

1. 在webpack.config.js中添加如下配置即可

```
module.exports = {
  // ...其他配置
  devServer: {
    port: 3000 // 端口号
  }
}
```

2. 重新启动webpack开发服务器观察效果



如何修改webpack开发服务器的配置呢?

1. 去文档查找配置项的名字
2. 在webpack.config.js – devServer选项里添加

今日总结

1. 什么是webpack
2. 自定义命令在哪里
3. webpack构建的流程是什么
4. webpack的插件和加载器作用是什么, 你都用过哪些
5. webpack如何做热更新服务
6. 想学好webpack-会查文档-会看文档 不可少

今日作业

1. 把今日课上的配置从0再来一遍
2. (附加)可以自己调研如何把css, 独立打包提取成一个css文件在dist下
3. (附加)可以调研如何把vue文件让webpack打包使用(百度vue-loader官网)

传智教育旗下高端IT教育品牌