## webpack的基本使用

#### 1.创建列表隔行变色项目

①新建项目空白目录，并运行 npm init -y命令，初始化包管理配置文件 package.json

②新建src源代码目录

③新建src -> index.html首页

④初始化首页基本的结构

⑤运行npm install jquery -S命令，安装jQuery

⑥通过模块化的形式，实现列表隔行变色效果

#### 2.在项目中安装和配置 webpack

①运行npm install webpack webpack-cli -D命令，安装webpack 相关的包

②在项目根目录中，创建名为webpack.config.js的webpack配置文件

③在webpack的配置文件中，初始化如下基本配置：

module.exports={

mode:'development' // mode 用来指定构建模式

}

④在package.json配置文件中的scripts 节点下，新增 dev脚本如下：

"scripts"：{

“dev"："“webpack"//script 节点下的脚本，可以通过npm run执行

}

⑤在终端中运行npm run dev 命令，启动webpack 进行项目打包。

#### 3.配置打包的入口与出口

webpack的4.x版本中默认约定：

·打包的入口文件为src -> index.js

·打包的输出文件为dist -> main.js

如果要修改打包的入口与出口，可以在webpack.config.js中新增如下配置信息：

const path=require（'path'）//导入node.js中专门操作路径的模块module.exports={

entry:path.join（\_\_dirname，'./sre/index.js'），//打包入口文件的路径

output:{

path:path.join（\_\_dirname，’./dist’），//输出文件的存放路径 filename:'bundle.js'//输出文件的名称

}

}

#### 4.配置 webpack的自动打包功能

①运行npm install webpack-dev-server -D命令，安装支持项目自动打包的工具

②修改package.json -> scripts中的dev命令如下：

"scripts"：{

“dev”："webpack-dev-server"//script节点下的脚本，可以通过npm run执行

}

③将src->index.html中，script 脚本的引用路径，修改为"/buldle.js\*

④运行npm run dev命令，重新进行打包

⑤在浏览器中访问http://localhost：8080地址，查看自动打包效果

#### 5.配置html-webpack-plugin 生成预览页面

①运行npm install html-webpack-plugin -D命令，安装生成预览页面的插件

②修改webpack.config.is文件头部区域，添加如下配置信息：

//导入生成预览页面的插件，得到一个构造函数

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

const htmlPlugin=new HtmlWebpackPlugin（{//创建插件的实例对象 template:'./src/index.htm1'，//指定要用到的模板文件

filename:‘index.htm1.//指定生成的文件的名称，该文件存在于内存中，在目录中不显示

})

③修改webpack.config.js文件中向外暴露的配置对象，新增如下配置节点：

module.exports={

plugins:[ htmlPlugin ]//plugins 数组是webpack 打包期间会用到的一些插件列表

#### 6.配置自动打包相关的参数

//package.json中的配置

//--open 打包完成后自动打开浏览器页面

//--host配置IP地址

//--port配置端口

"scripts"：{

"dev"："webpack-dev-server --open --host 127.0.0.1 --port 8888"

），

#### 2.4 webpack中加载器的基本使用

##### 1.打包处理css文件

①运行npm i style-loader css-loader -D命令，安装处理css文件的loader

②在webpack.config.is的module -> rules数组中，添加1oader规则如下：

//所有第三方文件模块的匹配规则

module:{

rules:[

{test:/\.csss/，use:['style-loader'，'css-loader']}

]

其中，test表示匹配的文件类型，use 表示对应要调用的 loader

注意：

·use 数组中指定的1oader顺序是固定的

·多个1oader的调用顺序是：从后往前调用

##### 2.打包处理less文件

①运行npm i less-loader less -D命令

②在webpack.config.js的module -> rules数组中，添加loader规则如下：

//所有第三方文件模块的匹配规则

module:{

rules:[

{test:/\.less/，use:['style-loader'，'css-loader'，'1ess-loader']}

]}

##### 3.打包处理scss文件

①运行npm i sass-loader node-sass-D命令

②在webpack.config.js的module->rules 数组中，添加loader规则如下：

//所有第三方文件模块的匹配规则

module:{

rules:[

{test:/\.scss/，use:['style-loader'，'css-loader'，'sass-loader']}

]

}

##### 4.配置 postCSS自动添加css的兼容前缀

①运行npm i postcss-loader autoprefixer -D命令

②在项目根目录中创建 postcss的配置文件 postcss.config.js，并初始化如下配置：

const autoprefixer = require（'autoprefixer'）//导入自动添加前缀的插件module.exports={

plugins:I autoprefixer]//挂载插件

③在webpack.config.js的module->rules数组中，修改css的1oader规则如下：

module:{

rules:[

{test:/\.css$/，use:I'style-loader'，'css-loader'，'postcss-loader']}

]}

##### 5.打包样式表中的图片和字体文件

①运行npm i url-loader file-loader -D命令

②在webpack.config.js的module->rules数组中，添加1oader规则如下：

module:{

rules:[{

test:/\.jpg|png|gif|bmplttf|eot|svg|woff|woff2$/，

use:'url-loader？limit=16940’

}]

}

其中？之后的是1oader的参数项。

1imit用来指定图片的大小，单位是字节（byte），只有小于1imit 大小的图片，才会被转为base64图片

##### 6.打包处理js文件中的高级语法

①安装babe1转换器相关的包：npm i babel-loader @babel/core @babel/runtime -D

②安装babe1语法插件相关的包：npm i @babel/preset-env @babel/plugin-transform-runtime @babel/plugin-proposal-class-properties -D

③在项目根目录中，创建babe1配置文件babel.config.js并初始化基本配置如下：

module.exports={

presets:[‘@babel/preset-env']，

plugins:['@babel/plugin-transform-runtime'，'@babel/plugin-proposal-

class-properties']

④在webpack.config.js的module->rules数组中，添加1oader规则如下：

//exclude 为排除项，表示babe1-loader不需要处理 node modules中的js文件

{test:/\.js$/，use:'babel-loader'，exclude:/node\_modules/}

##### 3.3 webpack中配置vue 组件的加载器

①运行npm i vue-loader vue-template-compiler -D命令

②在webpack.config.js配置文件中，添加vue-loader的配置项如下：

const VueLoaderPlugin=require（"vue-loader/lib/plugin'）

module.exports = {

rules:[

//...其它规则

{test:/\.vue$/，use:'vue-loader'}

]

}，

plugins:[

//..其它插件

new VueLoaderPlugin（）//请确保引入这个插件！

]

##### 3.4在webpack 项目中使用vue

①运行npm i vue -S安装vue

②在src->index.js入口文件中，通过import vue from'vue'来导入vue 构造函数

③创建vue的实例对象，并指定要控制的e1区域

④通过render 函数渲染App根组件

//1.导入Vue构造函数

import vue from 'vue'

//2.导入App根组件

import App from ’./components/App.vue'

const vm= new vue（{

//3.指定vm实例要控制的页面区域

e1：'#app’，

//4.通过render函数，把指定的组件渲染到e1区域中

render: h =>h（App）

}）