第一节

  1》webpack是什么

    本质上，webpack 是一个现代 JavaScript 应用程序的静态模块打包器(module bundler)

  2》webpack能干什么

    模块化打包:

    css变成一行

    html变成一行

    js变成一行

    去除注释

    去除多余引号

    可以编译其他自定义的"后缀名"

第二节

  1》全局安装

    npm install webpack -g

    npm install webpack-cli -g （webpack4的脚手架）

  2》生成package.json

    npm init 初始化一个项目

  3》局部安装 生成node-modules

    npm install webpack -S （--save）

    npm install webpack-cli -S

  4》打包 : webpack

    默认entry ：src/index.js 默认入口

    默认output ：dist/main.js 执行webpack命令后默认生成dist/main.js

    (那么如何配置入口和出口呢？)

    可以自己定义出口和入口

    压缩： webpack --mode production (默认打包生产环境的变成一行)

    未压缩：webpack --mode development

第三节

  1》wbepack本身是打包js的，但是如果想要打包css，html，图片...怎么办？必须要进行相关配置

  2》配置

    1>webpack四个核心概念

      入口(entry)

      输出(output)

      loader: webpack只支持js,不支持css，图片...,如果让webpack支持使用loader

      插件(plugins)：有效的打包或者压缩css，js，html，图片

    2>配置文件

    命名：webpack.config.js

第四节

https://segmentfault.com/a/1190000006178770#articleHeader3 入门 Webpack，看这篇就够了

  一、本地服务器(dev-server)(热插拔)

  安装：

  npm install webpack-dev-server -S

  配置：

  webpack.config.js:

  devServer:{

    contentBase:"",

    inline

    port

    hot

  }

  配置本地服务器后怎么运行？webpack-dev-server

  也可以在package.json 配置运行命令

  "scripts":{

    "dev":"webpack-dev-server --open --inline" 自动刷新，实时打开

  }

  devServer:{

    contentBase:'./public', //本地服务器所加载的页面所在的目录

    proxy: {

      '/api': {

        target: ' https://segmentfault.com',

        // https://segmentfault.com/a/1190000018704305

        pathRewrite: { '^/api': '' },

        changeOrigin: true,     // target是域名的话，需要这个参数，

        secure: true,          // 设置支持https协议的代理

      }

    },

    port:8080,

    hot:true, // 实时刷新

    inline:true,

},

第五节

一、loader ： 加载器，加载东西

二、loader的配置

  1》引入css需要

  style-loader

  css-loader

  打包之前

  test:".css"

  use:['style-loader','css-loader']

下载：

  npm install style-loader css-loader -S

  file-loader css文件引入图片需要file-loader

  npm install file-loader -S

loader列表：

https://blog.csdn.net/keliyxyz/article/details/51649429

第六节

  一、插件

html怎么打包？webpack只打包js文件，如何打包打包html，css？ 需要用到插件

https://github.com/jantimon/html-webpack-plugin#configuration

    html-webpack-plugin

    src :开发阶段（写代码）

      index.html

      index.js

      style.css

      1.jpg

      ...

    public ：生产阶段也要有（项目要上线）

      index.html

      index.js

      style.css

      1.jpg

      ...

  2》使用 webpack.config.js

  1》安装（下载）

    npm install html-webpack-plugin -S

    该插件将为你生成一个 HTML5 文件， 其中包括使用 script 标签的 body 中的所有 webpack 包。

    只需添加插件到你的 webpack 配置如下：

  2》配置(webpack.config.js)

    1>const HtmlWebpackPlugin = require("html-webpack-plugin");

    2>plugins:[

      new HtmlWebpackPlugin({

        template:"./src/index.html", // 把谁生成了，包含什么模板的内容

        filename："a.html // 生成的文件名，不写默认生成输出口目录下的index.html,

        minify:{ // 设置打包后的html是否压缩

        removeAttributeQuotes:true,//去除引号

        removeComments:true,//去除注释

        removeEmptyAttributes:true,//去除空属性

        collapseWhitespace:true//去除空格

        }

      })

    ]

  第七节

  一、html-withimg-loader

    让html内可以支持图片

  1》安装(下载)

    npm install html-withimg-loader -S

  2》配置webpack.config.js

    {

    test:/\.html$/,

    use: 'html-withimg-loader'

    }

第九节

一、css提取?

  插件：

1》下载(安装)

  npm install extract-text-webpack-plugin@next -S

2》配置

  const ExtractTextPlugin = require("extract-text-webpack-plugin");

第十节

一、babel

  核心：

  babel-core

功能：

  babel-loader

  babel-preset-env

  babel-preset-react

二、安装（下载）

  npm install babel-core babel-loader babel-preset-env babel-preset-react -S

三、配置

  v1:

    webpack.config.js

  {

    test:/(\.jsx|\.js)$/,

    use:{

      loader:"babel-loader",

      options:{

      presets:["env",'react']

    }

    },

    exclude:/node\_modules/

  }

  v2:

    新增.babelrc文件

  {

  "presets":['env','react']

  }

第十一节

  引入第三方插件

    npm install jquery

    import $ from 'jquery'

  如何把vue项目上传到git仓库？

  可以先把仓库拿下来git clone XXX

  然后 vue init webpack XXX

  如何配置多个入口？

  多个出口？

a.js b.js c.js ==>  aaa.js

module.exports = {

  entry: ["./aap/a","./aap/b"，"./aap/c" ],

  output: {

    path.resolve(\_\_dirname, "dist")

    filename:"aaa.js"

  }

}

a.js ==>a.js     b.js c.js ===>bbb.js

module.exports = {

  entry: {

    a: "./app/entry-a",

    bbb: ["./app/b", "./app/c"]

  },

  output: {

    path.resolve(\_\_dirname, "dist")

    filename:"[name]-[hash].js"

    // [name] [hash] 用[name]去对应不同的出口文件名

  }

}

entry: "./app/entry", // string | object | array

entry: ["./app/entry1", "./app/entry2"],

entry: {

  a: "./app/entry-a",

  b: ["./app/entry-b1", "./app/entry-b2"]

},

  丢失或者缺少某些模块时，直接把node\_modules删掉，再重新npm install就可以了。

  get not？