JS知识笔记

# Html基础

## 简洁html结构

## Es6

[ES6 教程\_菜鸟教程](https://www.runoob.com/w3cnote/es6-tutorial.html)

[ECMAScript 6 入门\_阮一峰](https://es6.ruanyifeng.com/#docs/intro)

ECMAScript 6.0（简称ES6）是JavaScript语言的下一代标准，已经在2015年6月正式发布了。它的目标，是使得JavaScript语言可以用来编写复杂的大型应用程序，成为企业级开发语言。

部分浏览器，特别是IE支持的不好

Babel: JavaScript 编译器

## JS库

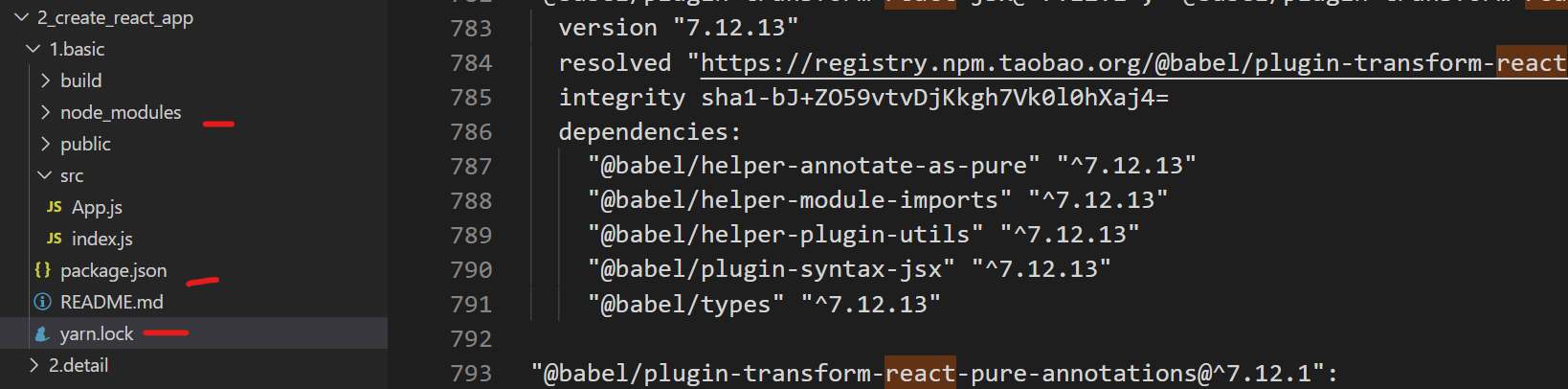
Jquery Angular vue react

## NodeJs

Npm：

*npm is the world's largest software registry. Open source developers from every continent use npm to share and borrow packages, and many organizations use npm to manage private development as well.*

1. 无需在线引用依赖，可以下载到本地。
2. 可以自动解析并管理间接依赖；
3. 写法更简洁。



# Webpack

时间: 2021-1-10 到 2021-1-31, 三个周末

## Reference

[webpack官方中文](https://webpack.docschina.org/concepts/)

[尚硅谷最新版Webpack5实战教程(从入门到精通) - 视频教程 - blibli](https://www.bilibili.com/video/BV1e7411j7T5)

## Webpack是什么

[webpack官方中文网](https://webpack.docschina.org/)



## 基本配置

* [入口(entry)](https://webpack.docschina.org/concepts/#entry)
* [输出(output)](https://webpack.docschina.org/concepts/#output)
* [loader](https://webpack.docschina.org/concepts/#loaders)
* [插件(plugin)](https://webpack.docschina.org/concepts/#plugins)
* [模式(mode)](https://webpack.docschina.org/concepts/#mode)

## webpack-dev-server

### 概念和目的

webpack-dev-server是一个小型的node.js Express服务器,它使用webpack-dev-middleware中间件来为通过webpack打包生成的资源文件提供Web服务。

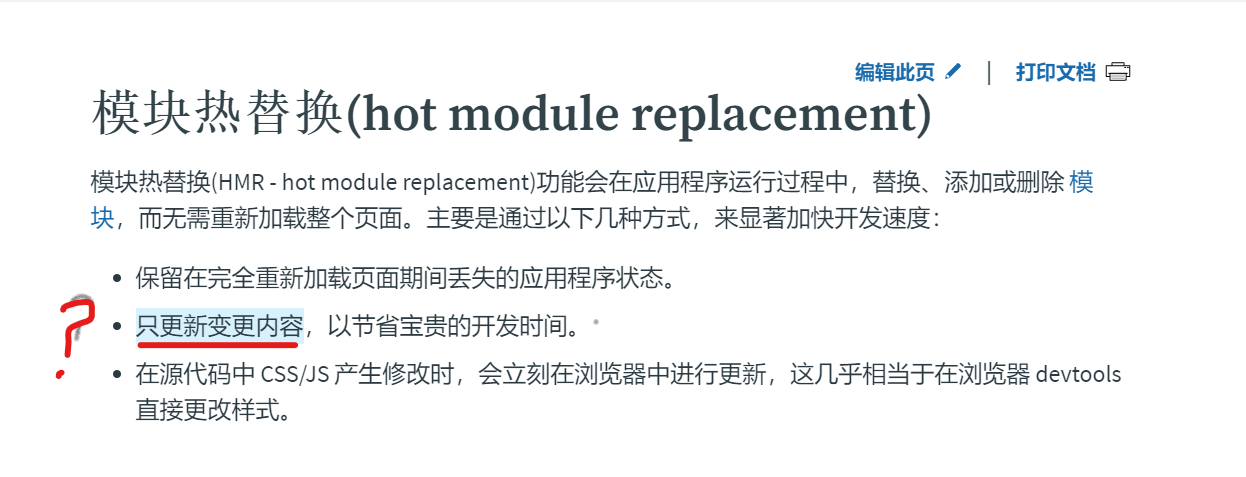
代码：09\_devServer

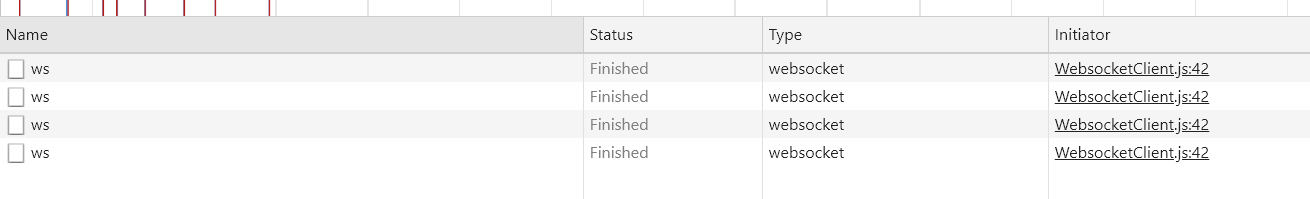
### HMR

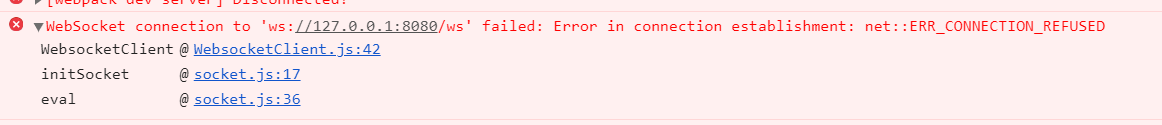
webpac-dev-server支持Hot Module Replacement，即模块热替换（也可以不使用webpack-dev-server, 而通过webpack其它配置实现）:

在前端代码变动的时候无需整个刷新页面，只把变化的部分替换掉。

[模块热替换(hot module replacement) - 官方介绍](https://webpack.docschina.org/concepts/hot-module-replacement/)







[websocket介绍 - 阮一峰的博客](http://www.ruanyifeng.com/blog/2017/05/websocket.html)

不同于http单向协议，webpack是：服务器可以主动向客户端推送信息，客户端也可以主动向服务器发送信息，是真正的双向平等对话，属于服务器推送技术的一种。

### 开发模式下如何解决跨域

只有ip&端口都一样，才不会跨域

**生产环境：**

前端页面：10.143:22333/be-invoker/skynet/page/123

后端接口：10.143:22333/be-invoker/api/page/

10.143:22333/Skynet/service/query-case-list

Web容器，同一个端口22333下

后端接口 + 前端静态资源

本**地开发：**

A: Localhost:3000/xx

浏览器地址栏

B: locahost:22333/be-invoker/

bs.lujs.net/Skynet/service/xxx

后端接口

方式1：后端接口代码里设置允许跨域

方式二：

1. 把后端接口请求指向webpack-dev-server；

2. webpack-dev-server配置代理，&&再进行转发：

A: Localhost:3000/xx

浏览器地址栏

B: locahost:22333/be-invoker/

bs.lujs.net/Skynet/service/xxx

后端接口

A: Localhost:3000/xx

Nodejs express后端服务

代码：36\_devserver

## 开发者模式VS生产者模式

mode: 'development',

mode: 'production',

[模式（Mode）- 官方解释](https://webpack.docschina.org/configuration/mode/)

以下没有特别规定一定如此，只不过不同模式下做的事情倾向于不同，可能对于框架来说更是如此：

|  |  |  |
| --- | --- | --- |
|  | 开发者 | 生产者 |
|  | Webpack-dev-server调试 | Css压缩  Css兼容性处理  Js eslint语法检查  Js压缩  js兼容性处理 |

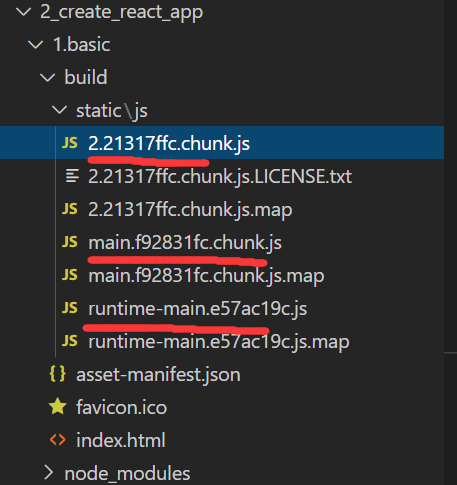
## 开发及打包性能优化

### 缓存

有好处也有坏处。

解释为什么框架编译后的文件要带hash值, chunk是什么？

代码：23\_cache



### Tree\_shaking

通过工具"摇"我们的JS文件，将其中用不到的代码"摇"掉

[B站视频教程](https://www.bilibili.com/video/BV1e7411j7T5?p=24)

[Tree-Shaking性能优化实践 - 原理篇 - 百度外卖大前端 - 掘金](https://juejin.cn/post/6844903544756109319)



### Code split && ddl

**代码分割：**

通过entry多入口方式 && optimization

module.exports = {

    // 单入口方式:

    // entry: './src/js/index.js',

    // 多入口方式:

    entry: {

        "index": './src/js/index.js',

        "study": './src/js/study.js'

    },

………

    optimization: {

        splitChunks: {

            chunks: 'all'

        }

    }

}

从打包中去除在线链接：29\_externals

**Ddl：**

1. externals是可以引用在线cdn链接

2. dll是常用的打包成本地js，这样也就包括node\_modules, 一些自己定义的固有js了

使用插件：add-asset-html-webpack-plugin

[add-asset-html-webpack-plugin使用说明 - npm](https://www.npmjs.com/package/add-asset-html-webpack-plugin)

### 多进程打包

  loader: 'thread-loader',

### 其它

懒加载 & 预加载

Pwa Progressive Web App，渐进式WEB应用

实现离线缓存功能，即使用户手机没有网络，依然可以使用一些离线功能

路径别名：35\_resolve

    resolve: {

        // 这个非常好

        alias: {

            $cssPath: resolve(\_\_dirname, 'src/css')

        },

    }

## 使用react

之前react都是使用脚手架，别人给配置好了webpack，如果自己直接在webpack中使用react呢？

简直是个噩梦,

1. babel版本问题
2. webpack配置babel-loader，而且坑爹的是一些网友给的是webpack4，webpack5写法又不一样  
   本来网上的教程很不错，只是前端相关的东西发展太过迅猛，只相隔了半年有些东西的版本就不对了
3. 还得新建一个文件 .babelrc

特别感谢：

[最小白的webpack+react环境搭建 - 我的天空-老潘 - 博客园](https://www.cnblogs.com/mysky7511/p/9495924.html)

[babel报错](https://blog.csdn.net/weixin_42235173/article/details/90897014)

## 特别注意的坑

Webpack版本兼容性；目前（2021-1-31）

# npm i webpack 安装的是webpack5, 然而不少插件默认配套webpack4，有莫名奇妙的问题时应该考虑npm i xxx@指定版本

# Create-react-app

## 是什么

可以理解为一个插件，或者一个命令；



## 常用功能

[create-react-app官方中文文档](http://www.html.cn/create-react-app/)

## eject弹射出webpack配置

## react\_app\_rewired更改配置

# UmiJs

2021/6/14 端午节

[umijs官方中文教程](https://umijs.org/zh-CN)



## Hello World

1. 最简单的方式

# yarn add umi

# npx umi g page index --typescript --less

# npx umi dev

1. 脚手架

$ yarn create @umijs/umi-app

# 或 npx @umijs/create-umi-app

# yarn

# yarn start

应该是默认生成typescript, less ?

## Umi基础

### 目录结构

### 配置

Umi 在 .umirc.ts 或 config/config.ts 中配置项目和插件

如果.umirc.ts存在，config/config.ts则不会生效；

还能指定多份配置，例如本地再来一份：.umirc.local.ts

[运行时配置](https://umijs.org/zh-CN/docs/runtime-config)

Src/app.tsx

在这里可以修改 clientRender 参数，比如在微前端里动态修改渲染根节点，不渲染到root去；

修改路由，如在最前面添加一个 /foo 路由，

覆写 render。比如用于渲染之前做权限校验，

在初始加载和路由切换时做一些事情。比如用于做埋点统计 比如用于设置标题，

### 路由

**配置路由**

不管是在 .umirc.ts 或 config/config.ts , 只需要

export default {

routes: [

{ exact: true, path: '/', component: 'index' },

{ exact: true, path: '/user', component: 'user' },

],

}

[**约定式路由**](https://umijs.org/zh-CN/docs/convention-routing)

如果没有 routes 配置，Umi 会进入约定式路由模式，然后分析 src/pages 目录拿到路由配置。

感觉非常棒呀！

### Html模板

这里没有create-react-app中的那个public文件家，如果要定义模板

src/pages/document.ejs

### Mock数据

一直想研究下js里面怎么写UT和mock，这小节太好了

[mock数据 - umi](https://umijs.org/zh-CN/docs/mock)

Umi 约定 /mock 文件夹下所有文件为 mock 文件。

更多深度mock, 可能需要参考：<http://mockjs.com/>

# DvaJs

# Ant design pro

# 其它待续

## .gitigore使用