JS知识笔记

--2021-2-18

# 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

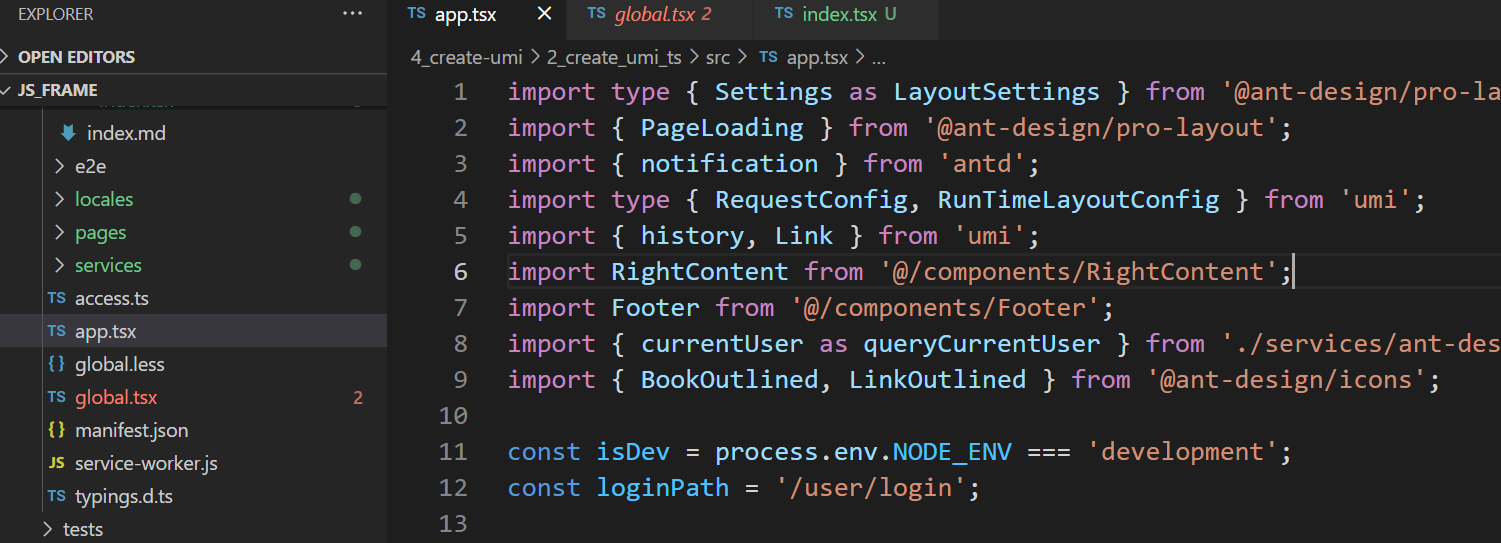
Babel: JavaScript 编译器

## JS库

Jquery

Angular vue react

直接引用js库，有缺点：

1. 不利于模块化管理；  
   改进之后的：  
   
2. 初次加载需要临时编译，速度慢：  
   [file:///D:/code/js\_workpalace/study\_react/src/10\_special/8\_dragAndDrop/2\_html/4\_veryNice.html](file:///D:\code\js_workpalace\study_react\src\10_special\8_dragAndDrop\2_html\4_veryNice.html)
3. 引入了很多非必要依赖  
   大型项目的插件之间存在间接依赖, 本地开发时依赖非常大。  
   a -> b -> c

<script src = “./a.js” />  
<script src = “./b.js” />

<script src = “./c.js” />

# 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/)

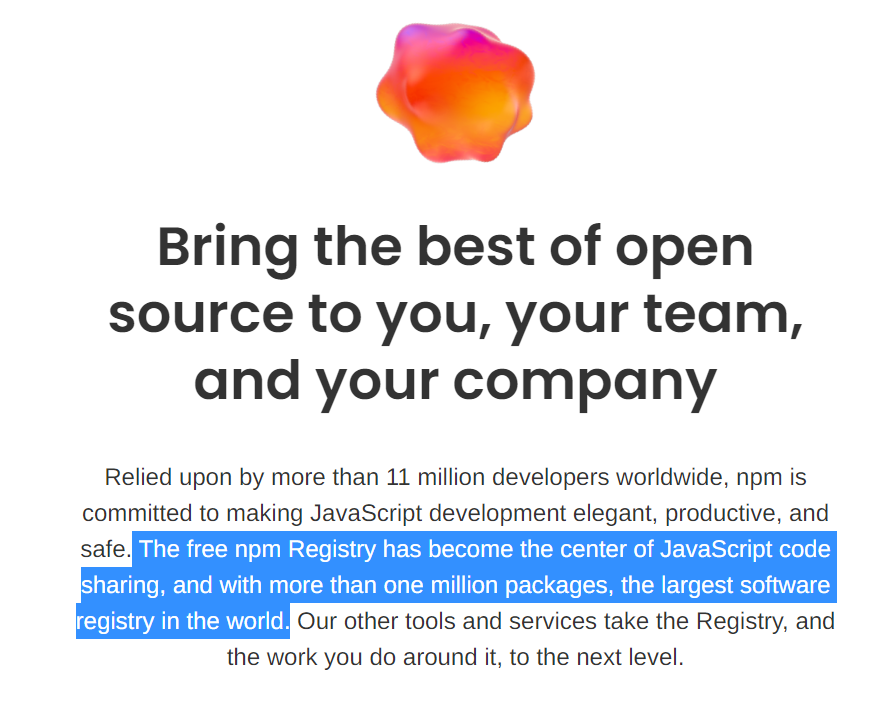


## 补充NodeJs

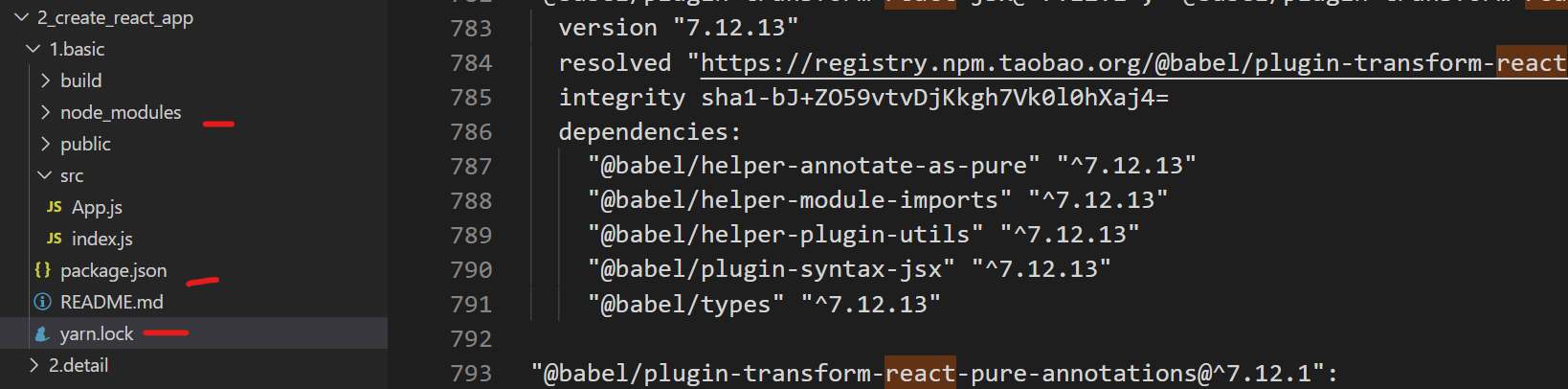
[Nodejs官网](http://nodejs.cn/)



**Npm:** 集成于Node.js中的标准软件包管理器。

****

1. 无需在线引用依赖，全部下载到本地。
2. 可以自动解析并管理间接依赖, 打包之后进行了很多优化。
3. 代码编写更简洁。



对比于java

|  |  |  |
| --- | --- | --- |
|  | js | java |
| 依赖管理、更新依赖 | # npm install  # yarn | # gradle  # maven |
| 打包 | # npm run build / yarn build  # webpack | # gradle build  # mvn build / mvn pacakge |

## 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)

webpack 只能理解 JavaScript 和 JSON 文件，这是 webpack 开箱可用的自带能力。loader 让 webpack 能够去处理其他类型的文件，并将它们转换为有效 模块，以供应用程序使用，以及被添加到依赖图中。

* [插件(plugin)](https://webpack.docschina.org/concepts/#plugins)

loader 用于转换某些类型的模块，而插件则可以用于执行范围更广的任务。包括：打包优化，资源管理，注入环境变量。

* [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。

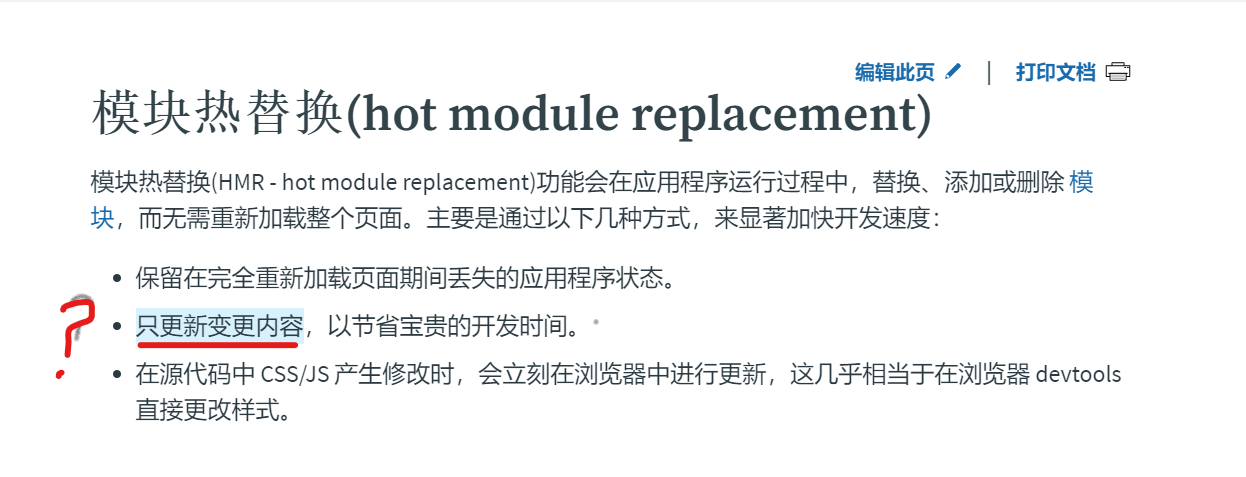
代码：09\_devServer

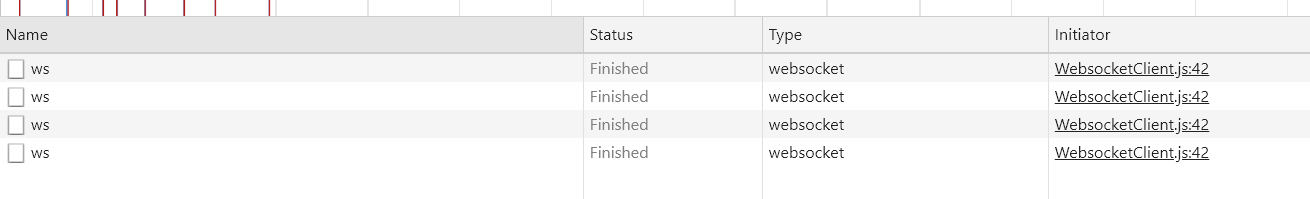
### HMR

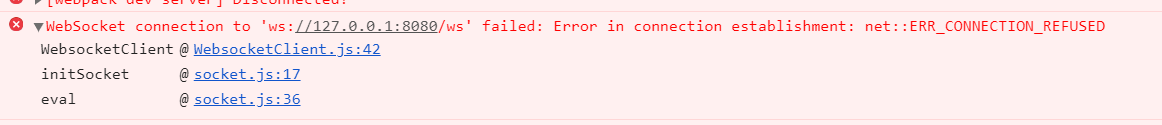
webpac-dev-server支持Hot Module Replacement，即模块热替换（也可以不使用webpack-dev-server, 而通过webpack其它配置实现）:

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 |

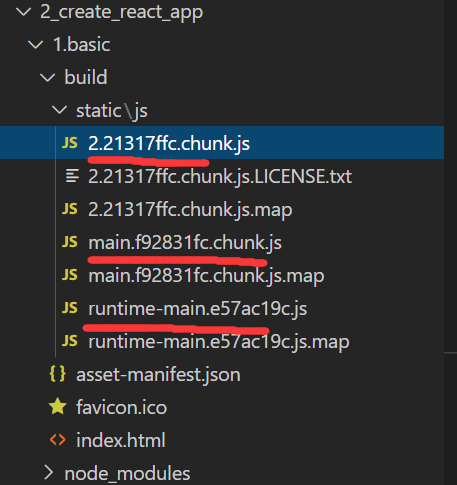
## 开发及打包性能优化

### 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

## Webpack缺点

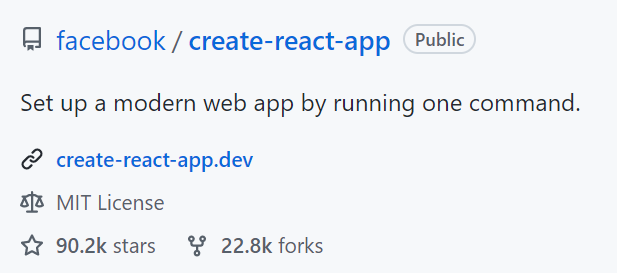
1. 要能形成一个完整可用的脚手架，需要很多的配置：loader，plugins…
2. 工程很大时，npm run build非常慢；此时即便只修改了一行代码，也需要重写build生成新的静态文件，重新上传。  
   ps: 开发模式下没问题，webpack-dev-server非常方便开发。

# 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/)

创建应用：

* npx create-react-app my-app

或者

* npm install -g create-react-app
* create-react-app my-app

## eject弹射出webpack配置

## react\_app\_rewired更改配置

## create-react-app的缺点

只完成了打包配置，对于我这样的菜鸟而言，做一个完整工程还不够方便：

1. 一些必须的功能：  
   路由：

<HashRouter>

      <Route path="/app" component={App}></Route>

      <Route path="/study" component={Study}></Route>

</HashRouter>

<http://192.168.1.6:3000/#/app>

<http://192.168.1.6:3000/#/study>

Layout:

<Layout>

<Header>Header</Header>

<Layout>

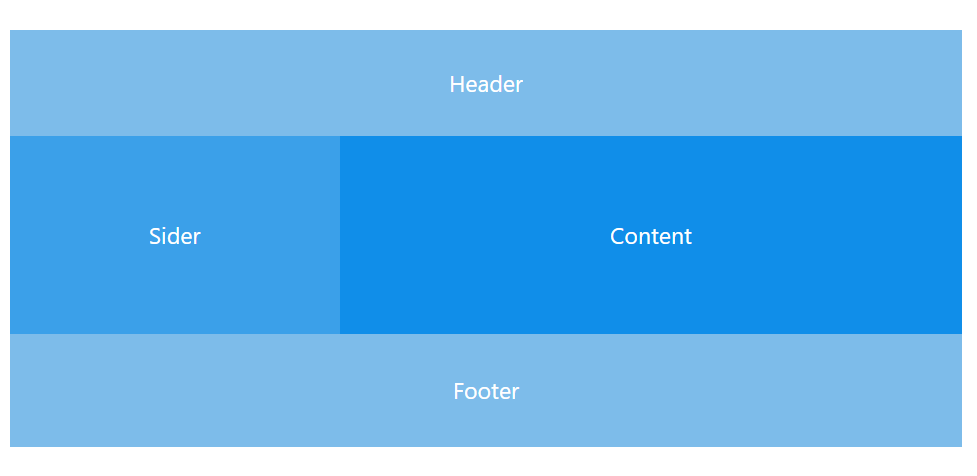
<Sider>Sider</Sider>

<Content>Content</Content>

</Layout>

<Footer>Footer</Footer>

</Layout>



Menus:

<Menu

defaultSelectedKeys={['1']}

defaultOpenKeys={['sub1']}

mode="inline"

theme="dark"

inlineCollapsed={this.state.collapsed}

>

<Menu.Item key="1" icon={<PieChartOutlined />}>

Option 1

</Menu.Item>

<Menu.Item key="2" icon={<DesktopOutlined />}>

Option 2

</Menu.Item>

<Menu.Item key="3" icon={<ContainerOutlined />}>

Option 3

</Menu.Item>

<SubMenu key="sub1" icon={<MailOutlined />} title="Navigation One">

<Menu.Item key="5">Option 5</Menu.Item>

<Menu.Item key="6">Option 6</Menu.Item>

<Menu.Item key="7">Option 7</Menu.Item>

<Menu.Item key="8">Option 8</Menu.Item>

</SubMenu>

<SubMenu key="sub2" icon={<AppstoreOutlined />} title="Navigation Two">

<Menu.Item key="9">Option 9</Menu.Item>

<Menu.Item key="10">Option 10</Menu.Item>

<SubMenu key="sub3" title="Submenu">

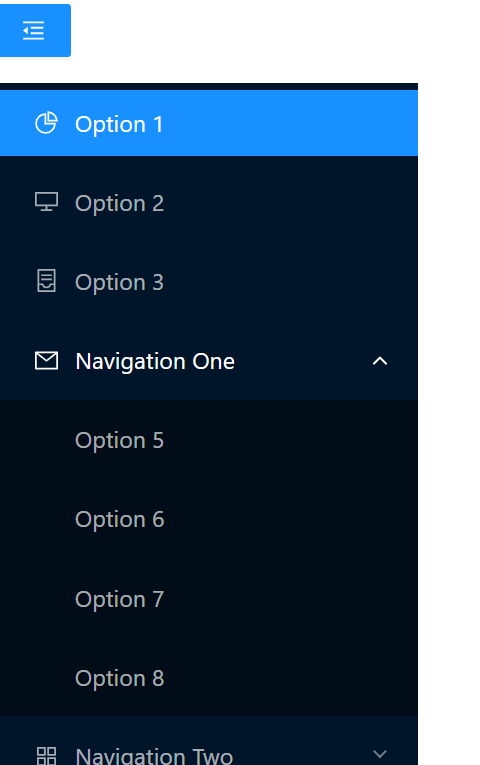
<Menu.Item key="11">Option 11</Menu.Item>

<Menu.Item key="12">Option 12</Menu.Item>

</SubMenu>

</SubMenu>

</Menu>



当前登录用户的管理：头像、昵称、注销等。

Ajax请求，404页面，登录超时/无权限跳转

1. 非必须功能
2. 国际化语言切换
3. 搜索功能
4. 小铃铛消息通知
5. 登录注册界面
6. 手机验证码、微博、微信账号登录…

<https://preview.pro.ant.design/>

# UmiJs

-- 2021/6/14 端午节 松江泖港

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





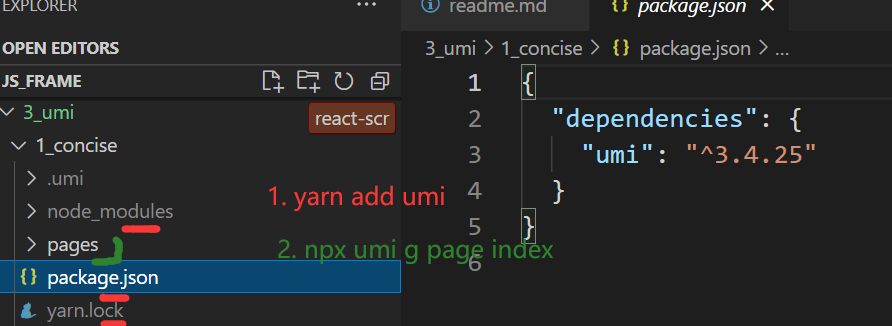
## Hello World

### 最简单的方式

# cd my-app

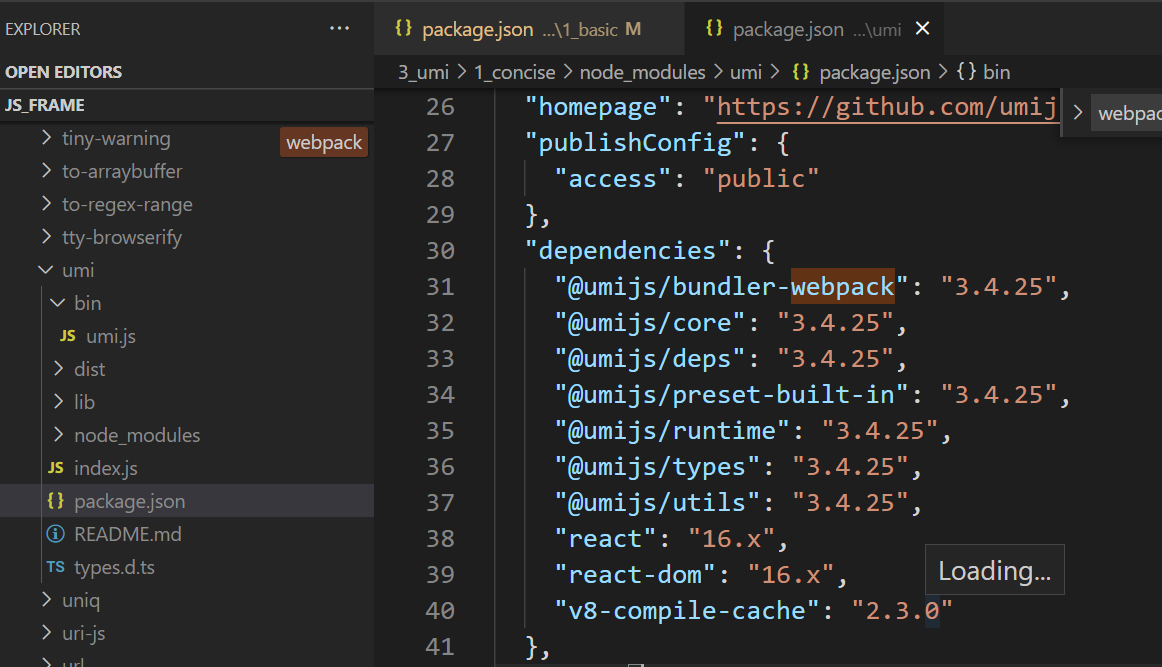
# yarn add umi

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

# npx umi dev  


[UMI官方介绍为什么要用它](https://umijs.org/zh-CN/docs)

插件集概念：



### 脚手架

上面的创建方法只是玩玩而已，用来大概了解下原理，真正使用还是的靠脚手架生产更多的默认配置

[umi脚手架创建工程](https://umijs.org/zh-CN/docs/getting-started)

# cd my-app

$ 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', name: ‘足球王子’ },

],

}

[**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/>

## UmiJs缺点

刚入门就学习这个，可能会比较懵圈。

# DvaJs

## 是什么

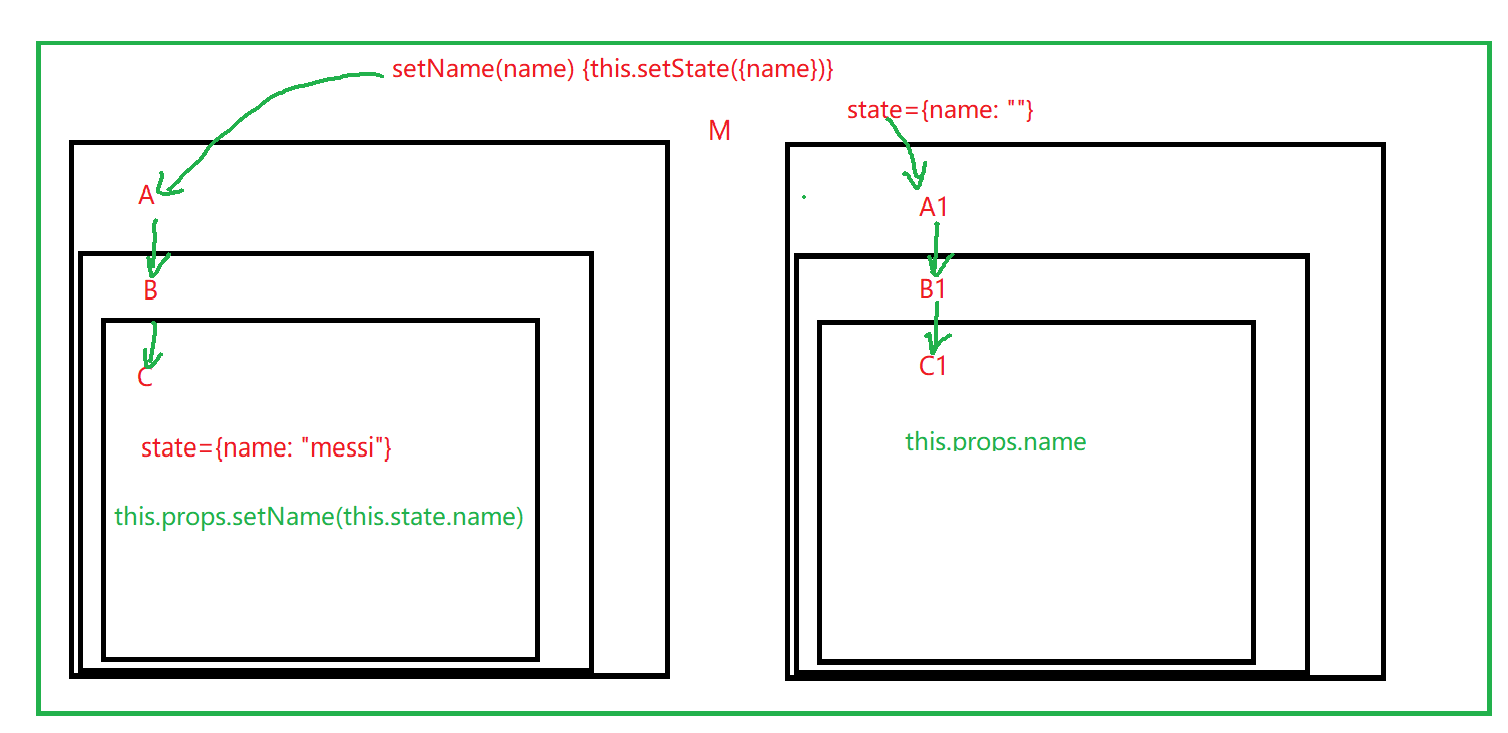
[dvajs官方文档](https://dvajs.com/guide/getting-started.html#%E4%BD%BF%E7%94%A8-antd)



巡检机器人 Lts Roadhog lu-probe

[roadhog - npm介绍](https://www.npmjs.com/package/roadhog)





dva 首先是一个基于 redux 和 redux-saga 的数据流方案，然后为了简化开发体验，dva 还额外内置了 react-router 和 fetch，所以也可以理解为一个轻量级的应用框架。



## Hello World

$ npm install dva-cli -g

$ dva new dva-quickstart

## 功能

待续…

But，文档   
Last Updated: 7/26/2018, 9:58:42 PM

# dva new my-project

都提示过时了，是否要使用 create-umi去创建ant-design-pro，所以感觉是不是过时了，这玩意儿。

[千锋教育\_最新DvaJS视频精讲（全网首发，没看就是错过一个亿） - B站](https://www.bilibili.com/video/BV1s7411v73j?p=5&spm_id_from=pageDriver)

这个教程我没看懂，差评

1. 老师不讲原理，上来就算方法
2. 不站在学生的角度循序渐进
3. 没有图解

比如redux设计的也同样很复杂麻烦，但是人家尚硅谷张天禹老师就讲的非常好懂

# Ant design pro

# create-umi my-app

# 未来展望

## Esbuilder

## Vite