Vue3+TS项目实战

王红元 coderwhy

tsconfig.json文件

- tsconfig.json是用于配置TypeScript编译时的配置选项：
  - https://www.typescriptlang.org/tsconfig
- 我们这里讲解几个比较常见的：

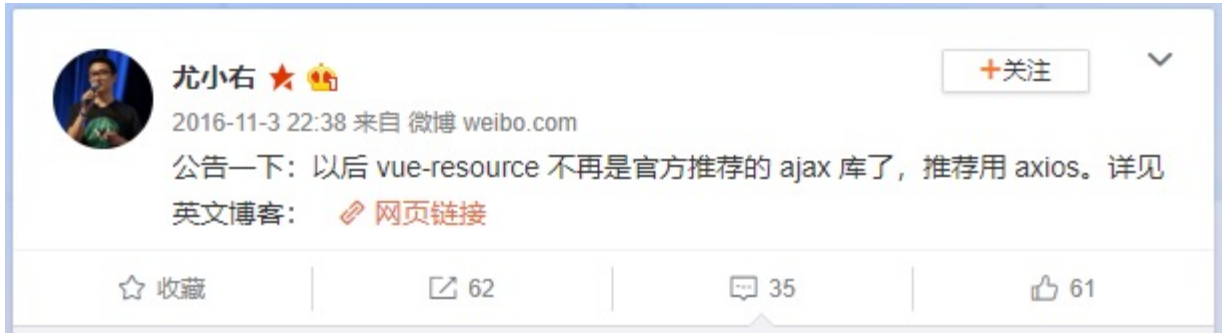
```
{
  "compilerOptions": {
    // 目标代码
    "target": "esnext",
    // 生成代码使用的模块化
    "module": "esnext",
    // 打开所有的严格模式检查
    "strict": true,
    "allowJs": false,
    "noImplicitAny": false,
    // jsx的处理方式(保留原有的jsx格式)
    "jsx": "preserve",
    // 是否帮助导入一些需要的功能模块
    "importHelpers": true,
    // 按照node的模块解析规则
    // https://www.typescriptlang.org/docs/handbook/module-resolution.html#module-resolution-strategies
    "moduleResolution": "node",
    // 跳过对整个库进行类似检测，而仅仅检测你用到的类型
    "skipLibCheck": true,
```

tsconfig.json文件

```
    // 可以让es module 和 commonjs相互调用
    "esModuleInterop": true,
    // 允许合成默认模块导出
    // import * as react from 'react': false
    // import react from 'react': true
    "allowSyntheticDefaultImports": true,
    // 是否要生成sourcemap文件
    "sourceMap": true,
    // 文件路径在解析时的基本url
    "baseUrl": ".",
    // 指定types文件需要加载哪些(默认是都会进行加载的)
    // "types": [
    //   "webpack-env"
    // ],
    // 路径的映射设置,类似于webpack中的 alias
    "paths": {
      "@/*": ["src/*"]
    },
    // 指定我们需要使用到的库(也可以不配置,直接根据target来获取)
    "lib": ["esnext", "dom", "dom.iterable", "scripthost"]
  },
  "include": [...
  ],
  "exclude": ["node_modules"]
}
```

认识axios

- 为什么选择axios? 作者推荐和功能特点



- 功能特点:
  - 在浏览器中发送 XMLHttpRequests 请求
  - 在 node.js 中发送 http请求
  - 支持 Promise API
  - 拦截请求和响应
  - 转换请求和响应数据
  - 等等

- 补充: axios名称的由来? 个人理解
  - 没有具体的翻译.
  - axios: ajax i/o system.

axios请求方式

- 支持多种请求方式:
  - axios(config)
  - axios.request(config)
  - axios.get(url[, config])
  - axios.delete(url[, config])
  - axios.head(url[, config])
  - axios.post(url[, data[, config]])
  - axios.put(url[, data[, config]])
  - axios.patch(url[, data[, config]])
- 有时候, 我们可能需求同时发送两个请求
  - 使用axios.all, 可以放入多个请求的数组.
  - axios.all([]) 返回的结果是一个数组，使用 axios.spread 可将数组 [res1,res2] 展开为 res1, res2

常见的配置选项

- 请求地址
  - url: '/user',
- 请求类型
  - method: 'get',
- 请根路径
  - baseURL: 'http://www.mt.com/api',
- 请求前的数据处理
  - transformRequest:[function(data){}],
- 请求后的数据处理
  - transformResponse: [function(data){}],
- 自定义的请求头
  - headers:{'x-Requested-With':'XMLHttpRequest'},
- URL查询对象
  - params:{ id: 12 },
- 查询对象序列化函数
  - paramsSerializer: function(params){ }
- request body
  - data: { key: 'aa'},
- 超时设置s
  - timeout: 1000,
- 跨域是否带Token
  - withCredentials: false,
- 自定义请求处理
  - adapter: function(resolve, reject, config){},
- 身份验证信息
  - auth: { uname: '', pwd: '12'},
- 响应的数据格式 json / blob /document /arraybuffer / text / stream
  - responseType: 'json',

axios的实例和拦截器

- 为什么要创建axios的实例呢?
  - 当我们从axios模块中导入对象时, 使用的实例是默认的实例.
  - 当给该实例设置一些默认配置时, 这些配置就被固定下来了.
  - 但是后续开发中, 某些配置可能会不太一样.
  - 比如某些请求需要使用特定的baseURL或者timeout或者content-Type等.
  - 这个时候, 我们就可以创建新的实例, 并且传入属于该实例的配置信息.
- axios的也可以设置拦截器：拦截每次请求和响应
  - axios.interceptors.request.use(请求成功拦截, 请求失败拦截)
  - axios.interceptors.response.use(响应成功拦截, 响应失败拦截)

axios+ts请求库封装

- 代码较多，这里不给出截图，见课堂代码

区分不同环境

- 在开发中，有时候我们需要根据不同的环境设置不同的环境变量，常见的有三种环境：
  - 开发环境：development；
  - 生产环境：production；
  - 测试环境：test；
- 如何区分环境变量呢？常见有三种方式：
  - 方式一：手动修改不同的变量；
  - 方式二：根据process.env.NODE_ENV的值进行区分；
  - 方式三：编写不同的环境变量配置文件；