Vue3Project DAY03

切换顶部导航，更新列表

切换到某一个类别，相当于拿到类别ID后重新发送请求，更新列表。

实现步骤：

1. 通过watch监听active的变化。
2. 监听到变化后，获取选中的类别ID，发送请求加载列表。
3. 更新列表。

实现列表的触底加载下一页

实现步骤：

1. 添加组件：van-list。
2. 补充相关变量和方法： loading finished onLoad()
3. 在onLoad中处理触底业务流

Web应用的缓存设计方案

当应用需要数据时，一般会向服务端发请求，但是有一些数据：数据量不小、全局基本没有什么变化、访问还挺频繁，这些数据如果每次都需要发送请求去服务端获取，将对服务端构成不小的压力。这时可以考虑做一下客户端**缓存**。

缓存是一种实现业务时的数据存储手段，主要业务流程如下：

1. 发送请求，获取一组数据，拿到数据后，存起来。
2. 当再次需要这些数据时，先去缓存中找，找得到；就直接用，找不到在发请求。

考虑缓存何时更新

如果缓存不更新，则每次从缓存中获取的数据都是旧数据，所以应该给应用数据提供一个合理的缓存更新机制。

1. 将缓存存入vuex，刷新页面就会更新。
2. 将缓存存入sessionStorage（关了浏览器就没了），下次访问就是新数据。
3. 将缓存存入localStorage（永久存储），每次加载时都会访问旧数据，所以在此处需要判断是否符合缓存的更新条件，如果符合更新条件，则需要抛弃现有数据，再去获取一份新的。例如以下更新条件：
    1. 判断当前缓存中的音乐列表是否是早上10点以后的新数据，如果不是，则更新缓存。
    2. 定时器，每隔5分钟更新一次。
    3. 实现列表的下拉刷新。（由用户控制手动更新）

实现列表的下拉刷新

```
<!-- 电影列表 -->
<van-pull-refresh
                  success-text="加载成功"
                  v-model="refreshing"
                  @refresh="onRefresh">
    <van-list
              v-if="movieList"
              v-model:loading="loading"
              :finished="finished"
              finished-text="没有更多了"
              @load="onLoad">
        <movie-item
                    :movie="item"
                    v-for="item in movieList" :key="item.id">
        </movie-item>
    </van-list>
</van-pull-refresh>
```

显示电影详情页

业务需求： 当用户在首页点击其中某一个列表项时，跳转到详情页。并且携带选中项的电影ID一起跳转。到详情页后显示该电影的详情信息。

实现步骤：

1. 准备好电影详情页静态页面。
2. 配置路由，当访问：`/movie-detail/:id` 时，跳转到详情页。（传递选中的电影ID）
3. 在详情页中接收参数id，带着该电影id，发送http请求，加载电影详情数据。
4. 在页面中完成数据的显示。