# 1. 总结 (Summary)

TechM基于 React 生态系统构建。涵盖项目搭建、组件化开发、集中式状态管理到声明式路由设计。

UI 框架 (React 18 + TypeScript):。React 进行组件化开发，将复杂的 UI 拆解为独立、可复用的组件。引入 TypeScript 为整个项目提供了静态类型检查，

UI 组件库 (Ant Design 5.x):项目采用了 Ant Design。如 Layout、Card、List、Avatar 等

状态管理 (Redux Toolkit): 对于跨组件共享的应用级状态（如文章详情、评论列表），使用Redux Toolkit。通过 createSlice 和 createAsyncThunk 等工具，简化了 Redux 的样板代码。例如articleStore，它将 reducers和 extraReducers组织在一起。

路由: 应用内的页面导航由 React Router驱动。我们利用其提供的 createBrowserRouter API集中配置所有路由规则，通过在 Layout 组件中使用 <Outlet />，实现了嵌套路由，保留公共页头和页脚，动态渲染匹配当前 URL 的页面内容。

后端与数据交互 (Axios + json-server):用 json-server 模拟后端。前端通过 Axios，所有 API 请求都被封装在 src/utils/request.ts 中，便于统一管理和维护。

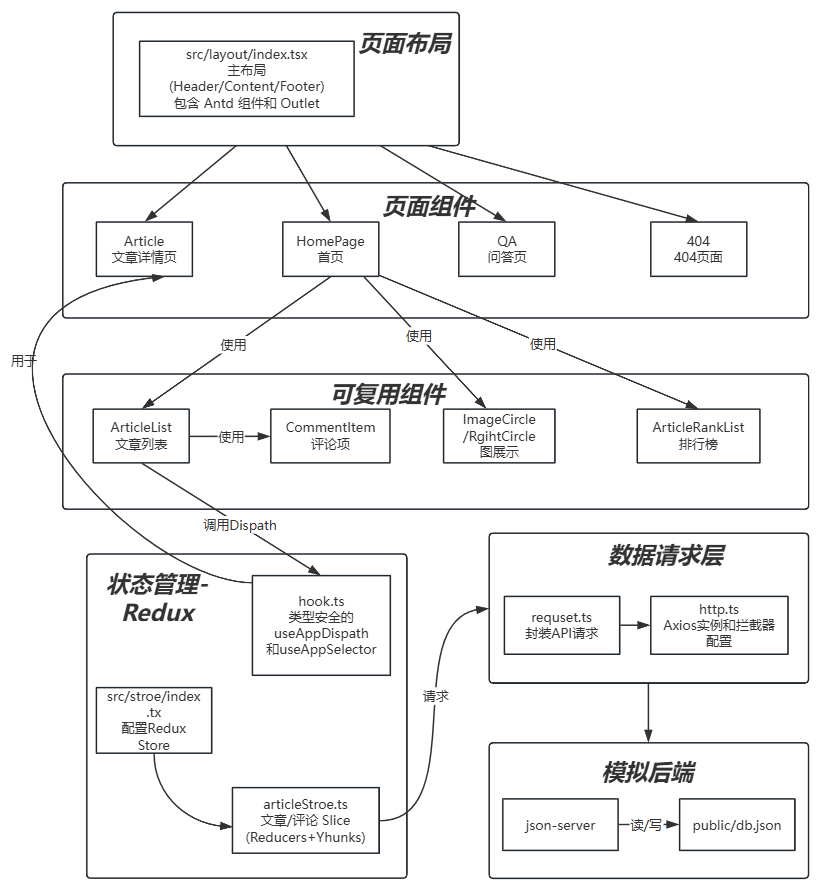
主要功能模块剖析

通用布局模块 (src/layout/index.tsx): Layout 模块 定义了所有页面的共享结构。包含了网站的 Logo、主导航栏、全局搜索框和用户状态（登录/注册），通过 <Outlet/> 展示独立页面的内容

首页模块 (src/pages/HomePage/index.tsx):组织了多个功能性子组件，如横向图片轮播（ImageCircle）、文章列表（ArticleList）和排行榜（ArticleRankList）。通过组件内部的 useState 和 useEffect 管理局部 UI 状态

文章详情模块 (src/pages/Article/index.tsx):使用 useParams从 URL 中动态获取文章 ID，并 dispatch getArticle 这个异步 thunk 来加载文章和评论数据。用户可以进行点赞、收藏等操作，这些交互会 dispatch 相应的 Redux action 来同步更新 UI。评论发布功能（handlePublishComment）异步流程：用户提交表单 -> dispatch addComment thunk -> thunk 调用 API -> 成功后 extraReducers 更新评论列表状态 -> UI 自动刷新。

状态管理模块 (src/store/): store 提供了全局状态的单一数据源。我们按照功能领域划分 slice（如 articleStore），每个 slice 内部封装了其 initialState、同步 reducers 和与异步 thunks 关联的 extraReducers。

相关结构图如下：

# 2. 模块设计详解

## 2.1ArticleStore设计

1. State (数据状态)

articleStore 管理着与文章相关的核心数据。

article: 存储当前页面的主要文章对象。初始值为 null，在数据加载成功后会被填充。

comments: 存储这篇文章对应的评论列表。

2. AsyncThunks (异步操作)

与后端 API 进行通信，获取或提交数据。使用createAsyncThunk

getArticle()根据传入的 articleId，并行地调用 getArticleAPI 和 getCommentListAPI，一次性获取文章和其评论列表。成功时返回包含 article 和 commentList 的对象。如果 API 调用失败，它会通过 rejectWithValue 返回一个标准化的错误信息。

addComment:将一条新评论发送到服务器。它接收评论信息（作者、内容、文章ID），构造一个 MyComment 对象，然后调用 addCommentAPI 将其发送到后端。成功后，它返回新创建的评论对象，以便更新状态。

3. Reducers (同步操作)

toggleLikesComment(commentId):切换某条评论的点赞状态。它使用了一个递归辅助函数 findComment 来在评论列表（包括子回复）中查找指定 ID 的评论，然后将其 ifLike 状态取反，并相应地增减 likeNum。

addCommentLocal(commentInfo):在本地状态中立即添加一条新评论，用于实现“乐观更新”。可以在调用 addComment 这个 thunk 的同时，同步调用 addCommentLocal 来立即更新UI。

toggleArticleMark():切换文章的收藏状态 (ifBookMark) 并更新收藏数 (bookmarks)。

toggleArticleLikes():切换文章的点赞状态 (ifLike) 并更新点赞数 (likeNum)。

4. ExtraReducers :由 createAsyncThunk 或其他 slice 创建的 actions。它将异步操作的结果与 state 连接起来。

getArticle.fulfilled:

getArticle 这个 thunk 成功完成时触发。实现: 将 action.payload（即 getArticle 返回的 { article, commentList } 对象）中的数据分别赋值给 state.article 和 state.comments，完成数据填充。

getArticle.rejected:失败时触发，用于处理错误，例如在控制台打印错误日志。

addComment.fulfilled: addComment 成功将评论提交到服务器后触发。

addComment.rejected:当 addComment 提交失败时触发，用于处理错误。

types.ts: 定义了所有的数据结构（Article, MyComment 等），确保了整个应用的数据类型一致性，是 TypeScript 发挥优势的基础。

index.ts: 用 configureStore 将 articleReducer 注册到全局唯一的 Redux Store 中，使其在 article 这个 key 下可用,导出了 RootState 和 AppDispatch 这两个至关重要的类型。RootState 代表了整个 Redux store 的状态树类型，而 AppDispatch 则包含了所有可 dispatch 的 action 类型（包括 thunks）。

hooks.ts: 创建了类型化的 useAppSelector 和 useAppDispatch hooks。

useAppSelector: 让你在组件中获取 state 时，无需手动声明 state 的类型，就能获得完整的类型提示和检查。

useAppDispatch: 让你在 dispatch thunk 异步 action 时，也能获得正确的类型检查和参数提示。大致工作流程如下