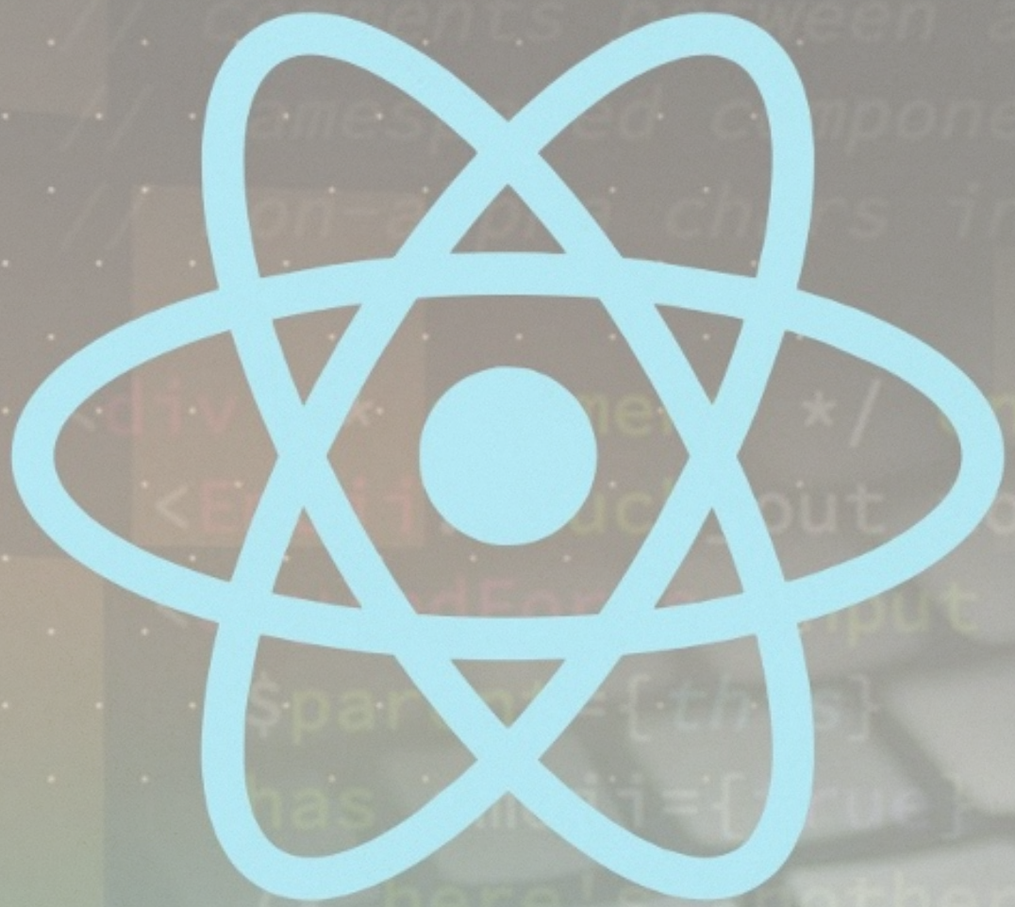

React 业务开发讨论

——包罗强

2019.11.22

目录

前言

很多编码问题或实现方案没有唯一答案，只有
针对特定应用场景和团队约定的最佳实践

以下话题旨在提出论点和方向，具体细节实现
和落地需要团队共同努力完善

业务开发周期

1. 需求分析
2. 组件设计与编写
3. 数据流设计
4. 组件串联
5. 联调测试
6. Code review

组件-设计

1. 按页面布局初步设计

2. 设计文件组织结构

3. 按功能模块再次拆分（使用或开发公共组件）

4. 按业务逻辑串联模块

组件-代码

- 代码整洁，拒绝重复
- props 类型定义
- 控制代码行数（300-500以内）
- 无状态组件与容器组件
- 组件、代码性能优化（除非必要，勿过早优化）

状态管理与数据流

UI = f(state)

The layout on the screen

Your build methods

The application state

store 设计

- 使所有数据具有响应式更新能力
- 尽量不使用全局变量保存数据
- store 保存所有公共数据
- store 结构扁平化
- 组件内 state 保存局部状态

数据流

- 自上而下，少量容器组件，尽可能多的无状态组件
  - 通过 props 传递数据流
  - 无状态组件较多，易复用，副作用少
  - 可能有较多不必要渲染
- 细粒度 connect 组件
  - 状态依赖准确，组件重新渲染控制好
  - 组件状态独立，只与 store 进行数据交互，灵活
- 最佳实践?

公共组件

公共组件（工具函数）

- 无副作用
- 注释、文档
- 简易接口
- 可扩展性
- 单一职责
- 让队友知道

公共组件开发

最佳实践

每个人都有自己的最佳实践

最佳实践

- 基于特定功能、业务场景的最佳实现方案
- 便于 copy，不易出错
- 各模块代码相似，维护方便
- 持续完善，技术产出

最佳实践形成

- 特定、重复业务场景（Modal、分页、表格、搜索）
- 分享交流
- 不断完善优化
- 有效产出（如公共组件、文章等）

总结

- 优先组件设计和封装
- 数据流思维实现功能
- 思考总结最佳实践
- 一切为了效率和产出
-

问题&讨论

- 代账改造中发现可编写很多公共组件（功能或样式），正在开发，后续分享

- 很多组件是在开发过程中发现功能重复，遇到这种情况及时提取公共组件，重构代码，比如记账打印功能

- 现有部分公共组件随业务发展扩展性和质量不能满足需求，需要维护完善

谢谢观看！