基于 react 实现前端组件化开发的研究与实现

敖　曼
（上海外服信息技术有限公司，上海 200120）

摘要：随着软件行业的迅速发展，WebApp 的功能及逻辑越来越复杂。用户们也不再满足于软件仅仅能够达到他们的基本业务需求，对于用户体验有越来越高的要求。文章基于 React 技术对组件化开发进行了研究分析，阐述如何通过组件化开发更快更好地产出让用户满意的 WebApp。

关键词：WebApp；React；组件化
中图分类号：TP311　　文献标识码：B　　文章编号：2096-9759（2021）07-0077-04

0 引言

随身前端的飞速发展，前端所需要实现的业务逻辑越来越复杂，一个页面可能会有几百上千行代码。在这种情况下代码的可读性和可扩展性非常差，维护起来异常困难。类似的场景在开发过程中并不少见。

在这种情况下，组件化开发可以通过拆分功能封装组件，实现各个模块单独维护，能够很大程度上解决以上问题。

1 组件化

组件化思想并不是前端的专利，很多后台流行语言很早就有类似的应用场景与先例。我们可以将样式以及其对应的功能看作一个整体，然后将这个整体放在任何地方使用，并且都具有相同的功能以及样式。由此实现了代码的复用性、灵活性，从而提高项目或系统的设计以及开发效率。这就是组件化的基本思想。

我们在多个项目中均使用了组件化进行前端的开发，将常用的组件用组件化方式设计并开发了一套模板，为之后的开发带来了便利。图 1 为按钮组件的样例。

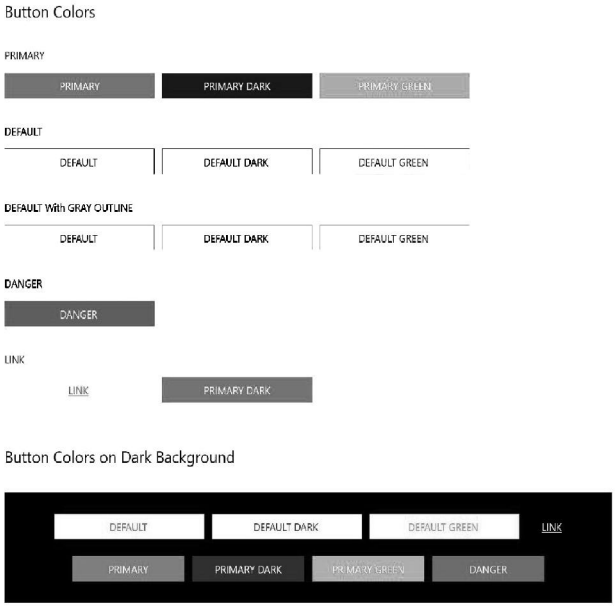

图 1 按钮组件的样例

2 React

2.1 React 简介

React 是 Facebook 开元在 GitHub 上的 JavaScript 库。它会把用户界面抽象成一个一个的组件，如按钮 Button、日历组件 Calendar、对话框组件 Dialog 等。开发人员可以通过这些组件得到功能完善、交互性丰富的页面。通过引入 JSX 语法。复用组件变得非常容易，同时也能保证组件结构的清晰。有了这层抽象的组件，React 把代码和真实渲染的目标隔离开来。

2.2 Virtual DOM

在 React 中，render 执行的结果得到的并不是真正的 DOM 节点，结果仅仅是轻量级的 JavaScript 对象，我们称之为 virtual DOM。虚拟 DOM 是 React 的一大亮点，具有 batching（批处理）和高效的 Diff 算法。这让我们可以无需担心性能问题，并且能够随时刷新整个页面，由虚拟 DOM 来确保只对界面上真正变化的部分进行相关的 DOM 操作。每当 React 检测到数据变化时，React 便会重新构建整个 DOM 树。

2.3 React 组件的生命周期

本文主要是基于 React 实现第二种形式的组件化开发。本质上 React 会关注各元素的构成。React 组件即为元素组件。这个组件元素被描述为纯粹的 JSON 对象，这样做的好处是可以使用类或者是方法的形式来构建它。React 组件有三个部分组成，它们分别是属性（props）、状态（status）以及生命周期。如图 2 所示。

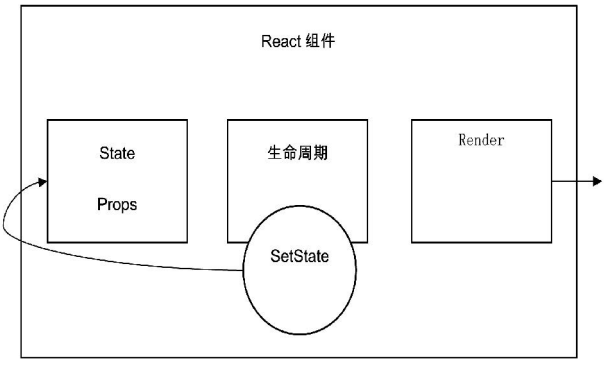

图 2 React 组件示意图

React 组件都有自己的状态，而且可以接收参数。并且如果接收到的参数或者状态发生了改变，React 组件会检测到此类变化，然后执行相应的生命周期方法，最终完成 React 组件的整体渲染。这一整个过程就是 React 组件化的生命周期的概述。

3 基于 React 的组件实际应用

3.1 组件的设计方法

React 组件支持面向对象的设计方法，React 也解决了频

收稿日期：2021-03-18
作者简介：敖曼（1975-），女，上海人，本科，研究方向：计算机。

繁操作 DOM 树的问题。可以将 WebApp 看作由一个个的组件组合而成。一些大型组件拥有很多相似的功能，可以将它们看作是由一些基本组件组合而成。例如这些组件可以是文字、select 框、单选、复选、输入框等组合而成。不同的大型组件都是由这些基础组件根据不同的排列组合而构成的。

以前为了减少大型组件的重复编码，往往在服务器端生成组件的前端代码。

现在基于 React，我们可以在前端开发组件，并且复用在各个项目中。根据大型组件可以分解为基本组件的这种思想。项目中的组件可以由多级组件组合而成。

React 组件需要通过接口和外部通讯。所以需要设计交互的数据结构。本文采用 JSON 进行数据交互。下面本文将基于一个项目中的一个 Tabs 组件来看看 React 组件化是如何实现的。

3.2 基础组件设计

以内部前端模板项目为例介绍下基础组件设计。首先要确认设计的整体风格，随后根据 UI 设计规范[1]编写组件样式等。参照规范进行整体的样式编写使页面风格统一。大致分为三块尺寸、交互、视觉。尺寸即是规范各个组件的高度宽度等。交互即是呈现各个组件与用户交互的方式及展现形式。视觉即是配色风格例如色系、高亮、字体高亮变色、字体放大等。如图 3 所示。

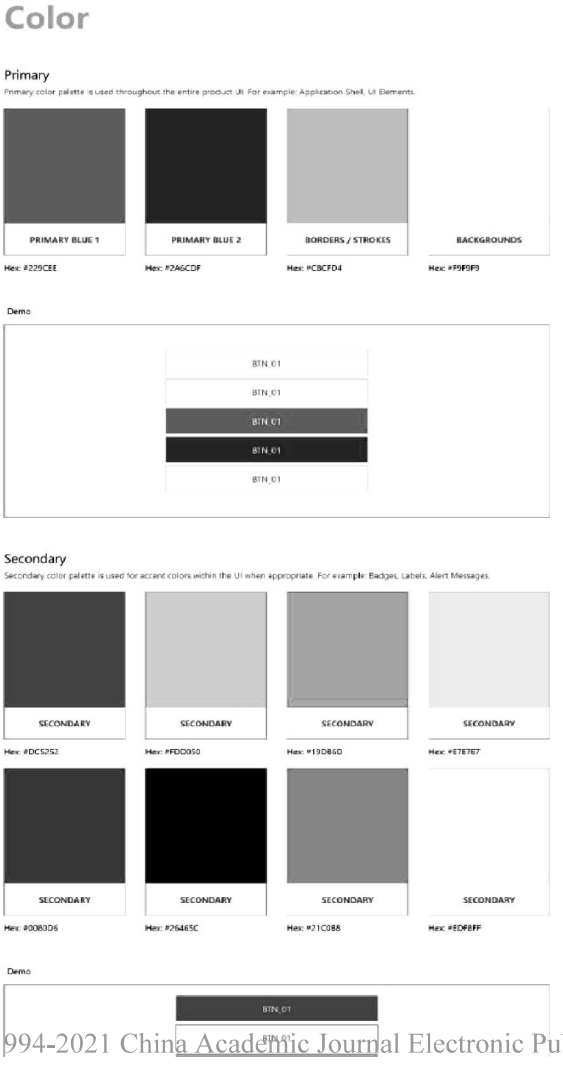

图 3 基础组件设计风格

遵循 UI 设计规范进行基础组件的开发。基础组件包括最基础的按钮、输入框、单选框、多选框等。图 4 为内部前端模板项目中的一些设计规范示例。

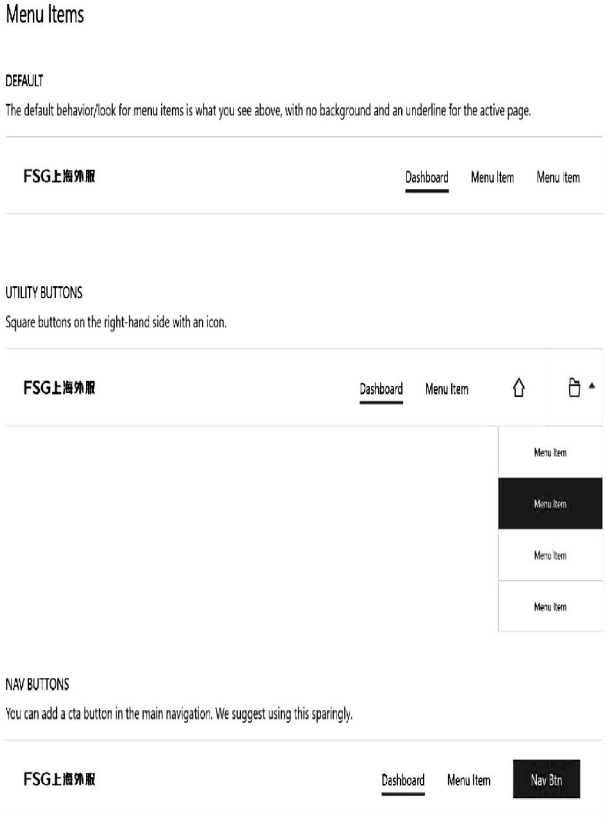

图 4 设计规范

3.3 通过栅格化提高页面规范性

栅格化是在一个有限的、固定的平面上，用水平线和垂直线（虚拟的线，“参考线”），将平面划分成有规律的一系列“格子”（虚拟的格子），并依托这些格子、或以格子的边线为基准线，来进行有规律的版面布局。遵循 UI 设计规范开发了栅格系统，栅格系统可以规范所有组件的尺寸。

布局的栅格化系统，基于行（row）和列（col）来定义信息区块的外部框架，以保证页面的每个区域能够稳健地排布起来。

下面简单介绍一下它的工作原理：

(1) 通过 row 在水平方向建立一组 column (简写 col)；

(2) 内容应当放置于 col 内，并且只有 col 可以作为 row 的直接元素；

(3) 栅格系统中的列是指 1 到 24 的值来表示其跨越的范围。例如，三个等宽的列可以使用 .col-8 来创建；

(4) 如果一个 row 中的 col 总和超过 24，那么多余的 col 会作为一个整体另起一行进行排列。

通过栅格系统我们可以方便地将组件进行布局，大大提高了网页的规范性。在栅格系统下，页面中所有组件的尺寸都是有规律的。这对于大型网站的开发和维护来说，能节约不少成本。

基于栅格进行设计，可以让整个网站各个页面的布局保持一致。这能增加页面的相似度，提升用户体验[2]。

3.4 基于 React 的 Tabs 组件的设计和实现

3.4.1 Tabs 组件介绍

React 组件的生命周期，大致可以分为挂载、渲染和卸载这几个阶段。渲染组建后更新数据，需要去重新渲染组件，直至卸载。下面将借助在内部前端模板项目中实际开发过的一个 Tabs 组件来介绍如何基于 React 实现 Tabs 组件，讲述如何进行渲染、挂载、卸载以及数据更新。

首先我们看一下内部前端模板项目中 Tabs 组件的大致结构。Tabs 组件分解成各个子组件。

大致结构为如图 5 所示：

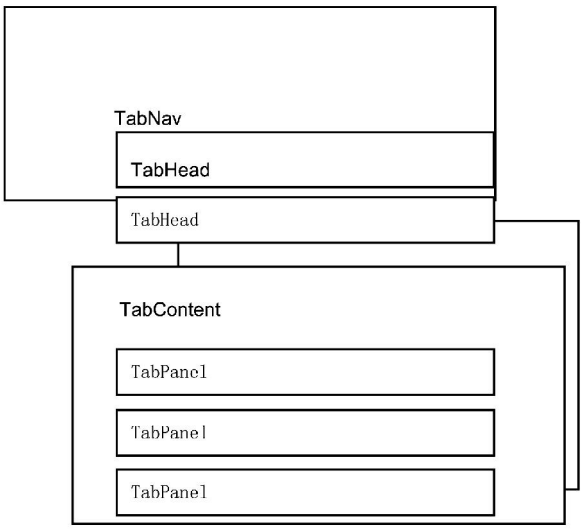

图 5 Tabs 组件结构示意图

上图在组件内只显示定义内容的子组件集合。TabNav 组件功能切换内容的点击按钮，TabContent 组件功能为内容展示区域。TabNav 内部的 TabHead 对应每一个 TabPanel 组件的 props。props 是 React 让组件直接交互的一种机制，大致和方法的传参相似。通过每一个子组件的 props 让其在 Tabs 组件内进行拼装。这种方式的好处是调用方式简洁，将复杂的逻辑留给组件去实现。

3.4.2 组件集合渲染过程

下面以内部前端模板项目中 Tabs 组件代码为例介绍一下组件的渲染过程。主要展示子组件集合是如何进行渲染的。

```
getTabPenals（）{
    const {classPrefix，activeIndex，panels，isActive}=this.props;
    return React.Children.map(panels,(child))=>{
        if(!child){return;}
        const order=parseInt(child.props.order,10);
        const isActive = activeIndex===order;

        return React.cloneElement(child,{
            classPrefix,
            isActive,
            children:child.props.children,
            key:'tabPanel-${order}',
        });
    });
}
```

这里涉及了一个子组件的概念 React 中有一个重要的内置的 prop——children。它代表组件的子组件合集，children 可以是数组类型。

React.children 是 React 官方提供的一系列操作 children 的方法。他提供很多方法使我们可以更方便的处理子组件。提供的方法有 map、forEach、count 等类似实用函数。

渲染上述组件的过程为通过 React.children.map 方法遍历子组件，将 order (渲染顺序)、isActive (是否激活 tab)、children (Tabs 组件中传下的 children) 和 key 利用 React 中的 cloneElement 方法克隆到 TabPanel 组建中，最后返回 TabPanel 组件的集合。以上就完成了 Tabs 组件拼装的整个过程。

3.4.3 Tabs 组件的生命周期

Tabs 组件完成后就来看一下组件的生命周期具体是如何工作的，它主要涉及了挂载、卸载、渲染以及数据更新。它们之间的关系如图 6 所示。

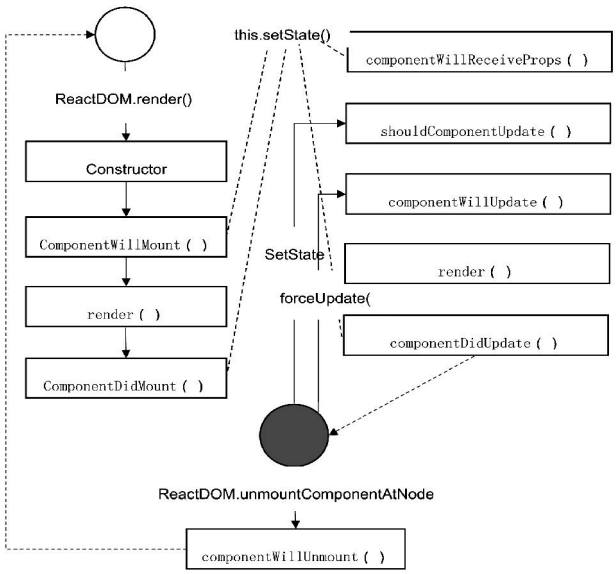

图 6 组件生命周期的流程图

图 6 为一张组件生命周期的流程图。

组件挂载相关的方法有 componentWillMount、componentDidMount。

组件更新相关的方法有 componentWillReceiveProps、shouldComponentUpdate、componentWillUpdate、componentDidUpdate。

组件卸载相关方法有 componentWillUnmount。

组件的挂载是最为基本的一个行为，他的主要目的是组件状态的初始化从图 5.4.3-1 中可以看出 componentWillMount 与 com-

ponentDidMount 方法会在 Render 方法前后分别执行。它们代表了渲染前后的两个时刻。这些方法都会在组件挂载时运行一次。

图 7 住房贷款利息信息的录入界面

组件的卸载只有一个 componentWillUnmount 方法。通常在此执行清理方法。比如清除事件或者清除定时器等。

组件数据更新可以理解为父组件向下传递 props 或者组件自身执行 setState 方法时发生的一系列更新。组件自身 state 发生了改变。会依次执行 shouldComponentUpdate、componentWillUpdate、componentDidUpdate。在 shouldComponentUpdate 方法中，他接收 props 和 state。在项目中往往在其中添加一些判断条件，来控制是否需要跟新组件，返回 false 则不会再向下执行生命周期方法。shouldComponentUpdate 方法可以决定组件是否需要重新渲染，主要用于组件的性能优化。

componentWillUpdate 和 componentDidUpdate 这两个生命周期方法比较容易理解，跟组件挂载的两个方法相似，它们也分别在更新过程中渲染前后执行。

以上及阐述了 Tabs 组件的整个生命周期是如何工作的。

4 敏捷开发项目中的例子

2018 年底，由于专项扣除政策。需要新上线需要快速开发相关功能用于个税抵扣，云平台个税抵项目扣应运而生。

由于此项目的紧迫性、需求多变性，需要多次迭代相关功能用于个税抵扣。所以我们使用敏捷开发，下面是敏捷项目中如何融入前端组件化开发的例子。

此项目目标是实现企业员工可通过外服云平台在线提交专项附加扣除和税延养老保险的申报，各项申报信息的修改、查询，个税抵扣金额的明细查询以及当前月个税抵扣金额的查询。实现企业 HR 可通过外服云平台查看本公司通过外服云平台或小程序提交的专项附加扣除和税延养老保险抵扣申报的员工每月个税抵扣的金额。

由于项目时间的紧迫性，和需求的不确定性，我们直接借助内部前端模板项目中的 react 组件库进行前端的开发。

图 7 为项目中住房贷款利息信息的录入界面。

图中所示的页面由按钮组件、单选框组件、日期选择组件等组合而成，开发迅速整体风格统一。整个借助于组件库我们迅速的开发出功能完善、界面美观的程序。

通过这个项目，我们可以发现组件化带来的好处，即代码的复用性、灵活性以及提高项目或系统设计的开发效率。

5 结语

随着前端技术的不断发展，前端在程序开发中的比重日益提升。以传统的前端开发方式开发 WebApp 会产生许多困难。本文通过研究 React 技术，基于 React 实现前端组件化开发，为 WebApp 中的前端开发带来了新的思路，实现了代码的复用性、灵活性，从而提高项目或系统的设计和开发的效率，并且能够提升用户体验，整体提升了 WebApp 的质量。

参考文献：

[1] Johnson J.认知与设计:理解 UI 设计准则[M].张一宁译.人民邮电出版社,2011.
[2] Garrett J J.用户体验要素:以用户为中心的产品设计[M].范晓燕译.机械工业出版社,2011.

（上接第 76 页）随着通信技术的飞速发展，基于 RTP 的图像远程传输系统可以应用于宽带多媒体卫星通信业务中，在窄带无线视频远程传输的基础上，已广泛应用于矿产勘查、野外作业、防震减灾等领域，实时传输协议是提高图像质量的关键技术。

参考文献：

[1] 张瑞, 汤心溢. 基于激光照明的远距离视觉信息采集系统设计[J]. 红外技术, 2019, 41(02):57-60.
[2] 曹婷, 来智勇, 黄铝文.作物生长图像远程采集系统的设计与实现[J]. 农机化研究, 2020, 042(001):80-85.
[3] 杜文略, 李红薇, 高越. 水下试验图像数据采集存储系统的设计与实现[J]. 电子器件, 2019, 042(003):733-739.
[4] 宫毅,刘学焕, 李一鸣,等. 基于软件虚拟专用网络技术的医学影像远程传输系统的建立与测试[J].实用放射学杂志, 2020, 036(006):977-979,1014.
[5] 刘海洋, 胡晓峰, 雷旭. 基于图形集群的远程实时渲染系统研究[J]. 系统仿真学报, 2019, 31(005):886-892.
[6] 罗哲明, 杨媛媛.一种互联网环境下的医学图像自适应传输系统实现 [J]. 现代电子技术, 2020, v. 43; No. 559 (08): 13-15+19.
[7] 王志强,李介夫,王珊,等. 电力作业远程监控系统中视频的优化传输方法[J].电力科学与技术学报,2020,035(001):169-175.
[8] 梁国壮, 安在秋, 田涵雷,等. 基于网络协议的无线远程数据传输系统设计[J]. 现代电子技术, 2020, 043(006):23-26.
[9] 赵宏伟, 宋云峰, 王会峰,等. 激光基准桥梁挠度成像检测系统[J]. 激光与红外, 2019, 49(003):282-290.
[10] 秦晓明. 嵌入式拖拉机视频监控系统设计—基于 DSP 和视频压缩编码[J]. 农机化研究, 2020,042(006):255-259.