**恒生电子股份有限公司**

**HUI Vue3 组件库**

**架构设计说明书**

<技术平台总部>

2024年 3月

**目录**

[1. 引言 4](#_Toc162370718)

[1.1 编写目的 4](#_Toc162370719)

[1.2 背景 4](#_Toc162370720)

[1.3 术语与缩略语 4](#_Toc162370721)

[1.4 引用文件 5](#_Toc162370722)

[2. 架构目标与设计约束 6](#_Toc162370723)

[2.1 功能性需求 6](#_Toc162370724)

[2.1.1 组织结构 6](#_Toc162370725)

[2.1.2 业务功能 6](#_Toc162370726)

[2.1.3 业务流程 7](#_Toc162370727)

[2.1.4 业务数据 9](#_Toc162370728)

[2.2 非功能性需求 9](#_Toc162370729)

[2.2.1 性能需求 9](#_Toc162370730)

[2.2.2 安全性需求 10](#_Toc162370731)

[2.2.3 可维护性需求 10](#_Toc162370732)

[2.2.4 可测性需求 10](#_Toc162370733)

[2.2.5 其他非功能性需求 11](#_Toc162370734)

[2.3 系统周边与交互 11](#_Toc162370735)

[2.4 开发与运行环境限制 12](#_Toc162370736)

[2.4.1 公司技术选型约定 12](#_Toc162370737)

[2.4.2 运行环境约定 12](#_Toc162370738)

[2.4.3 开发环境约定 13](#_Toc162370739)

[3. 架构设计 14](#_Toc162370740)

[3.1 逻辑视图 14](#_Toc162370741)

[3.1.1 子系统划分 14](#_Toc162370742)

[3.2 开发视图 16](#_Toc162370743)

[3.2.1 功能需求映射与组件识别 16](#_Toc162370744)

[3.2.2 功能需求矩阵 16](#_Toc162370745)

[3.2.3 各组件说明 17](#_Toc162370746)

[3.2.4 技术选型 17](#_Toc162370747)

[3.2.5 各组件对应的代码工程与开发技术 18](#_Toc162370748)

[3.3 运行视图 18](#_Toc162370749)

[3.3.1 Table组件运行态架构设计 19](#_Toc162370750)

[3.3.2 全局定制与主题动态运行的架构设计 19](#_Toc162370751)

[3.3.3 组件支持键盘操作的架构设计 19](#_Toc162370752)

[3.4 数据视图 20](#_Toc162370753)

[3.4.1 Table 组件数据视图 20](#_Toc162370754)

[3.4.2 数据存储与分布 21](#_Toc162370755)

[3.4.3 Table 组件数据管理策略 23](#_Toc162370756)

[4.关键功能模块设计 27](#_Toc162370757)

[4.1 全局定制与主题功能模块设计 27](#_Toc162370758)

[4.2 全键盘方案功能模块设计 28](#_Toc162370759)

[4.2.1 功能性需求 28](#_Toc162370760)

[4.2.2 非功能性需求 28](#_Toc162370761)

[4.2.3 子系统的组件划分、组件协作图 28](#_Toc162370762)

[4.2.4 本CBB与各层组件的协作图 28](#_Toc162370763)

[4.3 Table 组件方案设计 28](#_Toc162370764)

[4.3.1 功能性需求 28](#_Toc162370765)

[4.3.2 非功能性需求 28](#_Toc162370766)

[4.3.3 子系统的组件划分、组件协作图 28](#_Toc162370767)

[4.3.4 重点功能的实现协作图 28](#_Toc162370768)

[5. 非功能性架构设计 30](#_Toc162370769)

[5.1 性能设计 30](#_Toc162370770)

[5.2 可靠性设计 32](#_Toc162370771)

[5.2.1 异常处置设计 32](#_Toc162370772)

[5.3 安全设计 33](#_Toc162370773)

[5.4 可维护性设计 33](#_Toc162370774)

[5.4.1 安装升级 33](#_Toc162370775)

[5.4.2 排查辅助 33](#_Toc162370776)

[5.4.3 生产环境测试设计 34](#_Toc162370777)

[5.5 可测试性设计 34](#_Toc162370778)

[5.5.1 测试数据设计 34](#_Toc162370779)

[5.5.2 单元测试和模块测试的可测性设计 34](#_Toc162370780)

[5.5.3 系统验证测试的可测性设计 35](#_Toc162370781)

[5.6 跨平台性设计 35](#_Toc162370782)

[5.7 其他非功能性设计 35](#_Toc162370783)

[6. 模板修订记录 36](#_Toc162370784)

# 引言

## 编写目的

本文档编写的目的是指导软件开发人员在开发组件的过程中按照设计方案进行开发，本说明书的预期 读者为软件开发人员，软件测试人员和项目评审人员。

## 背景

公司的 HUI vue2 版本的组件库已经支撑公司业务 6 年左右的时间了，产品的生命周期已经逐步走到尽头

vue3 版本在 2021 年进行了发版，新版本的周边生态已经成熟。此外 vue3 带来的模块化和 composition api 组合式的组件编程方式。此外 vue3就虚拟 DOM 树结构和差异对比算法进行了优化，减少了静态内容的重复创建，引入了块级的更新策略。渲染性能平均提升了 1.3 倍左右（官网数据）。根据公司现有业务的需求，组件库适配 vue3 版本已水到渠成。

适配 vue3 版本的组件目的是满足业务方对 vue3 组件需求以及提升 组件库的性能。

## 术语与缩略语

|  |  |
| --- | --- |
| **词汇** | **解释** |
| *Vue3* | 一个 mvvm 的 javascript ui 框架，为第三个版本 |
| 全局定制 | 根据配置设置组件的字体大小，以及 padding 和line-height |
| 键盘快捷键支持 | 组件支持业务方配置快捷键 |
| css 变量 | 一种在 CSS 中定义和复用样式值的机制，使得样式的维护和修改更加灵活、高效。 |
| tree sharking | 一种通过静态分析代码来消除无用代码的优化技术，以减小最终打包的 JavaScript 文件大小。 |
| 虚拟滚动 | 一种优化技术，通过只渲染可视区域内的元素并动态更新，来提高大数据列表的渲染性能。 |
| xss攻击 | 一种通过在网页中注入恶意脚本来窃取用户信息、劫持用户操作或破坏网站的安全漏洞。 |
| JSX | 一种在 JavaScript 中编写类似 HTML 的语法扩展，用于描述组件的结构和外观，提高了代码的可读性和开发效率 |
| 自定义 render | 一种允许开发者自定义组件渲染逻辑的技术 |
| css in js | 种将 CSS 样式直接写入 JavaScript 代码中的技术，通过 JavaScript 的能力来管理和生成样式，提供了更好的样式封装、动态性和可维护性。 |
| composition api | vue3 允许开发者使用函数式的方式组织和重用逻辑，从而更灵活地构建组件。 |

## 引用文件

<todo>: 列举全键盘、全局换肤、主题定制等全局相关方案

|  |  |  |
| --- | --- | --- |
| **文档名称** | **版本号** | **文档日期** |
|  |  |  |
|  |  |  |

# 架构目标与设计约束

## 功能性需求

### 组织结构

## 非功能性需求

### 性能需求

### 安全性需求

### 可维护性需求

### 可测性需求

## 系统周边与交互

## 开发与运行环境限制

### 运行环境约定

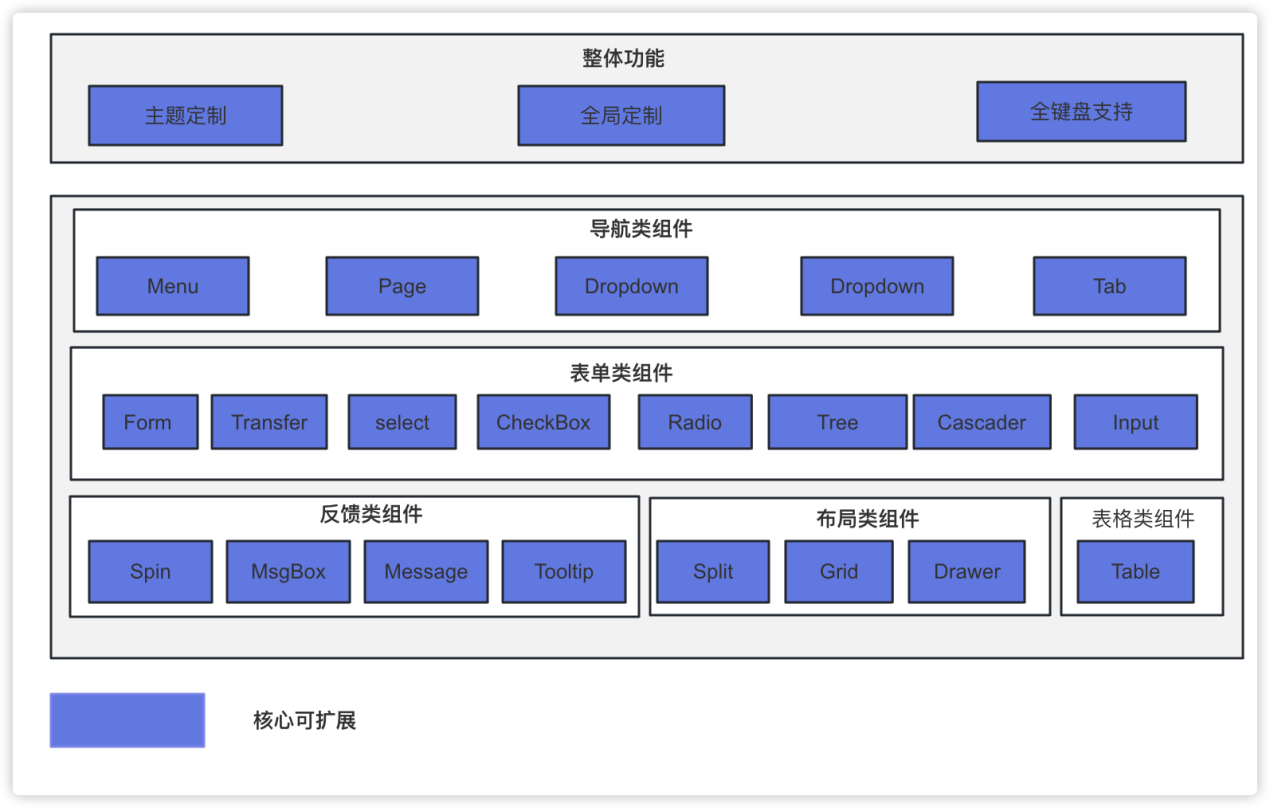
|  |  |  |
| --- | --- | --- |
| **运行环境** | **约束类型** | **说明** |
| WebView/浏览器 | CEF /Chrome 95+ | 95 代表内核版本为95 |
| firefox 103+ |  |
| QtWebEngine 6.3.0 | *内核为chromium95* |
| 360信创浏览器 | *内核为chromium95* |
| webview2 1.0.945.30 + | *内核为Chromium 95.0.4638.69* |

### 开发环境约定

|  |  |  |
| --- | --- | --- |
| **工具分类** | **工具名称** | **版本** |
| 构建工具 | roullup |  |
| 开发语言 | Javascript/typescript | ES2015+ |
| 版本管理工具 | git |  |
| 单测工具 | vitest | v1.4.0+ |
| 组件 UI 测试工具 | cypress | v13+ |
| 其他 |  |  |

# 架构设计

## 逻辑视图



### 子系统划分

## 开发视图

### 功能需求矩阵

### 各组件说明

### 技术选型



### 各组件对应的代码工程与开发技术

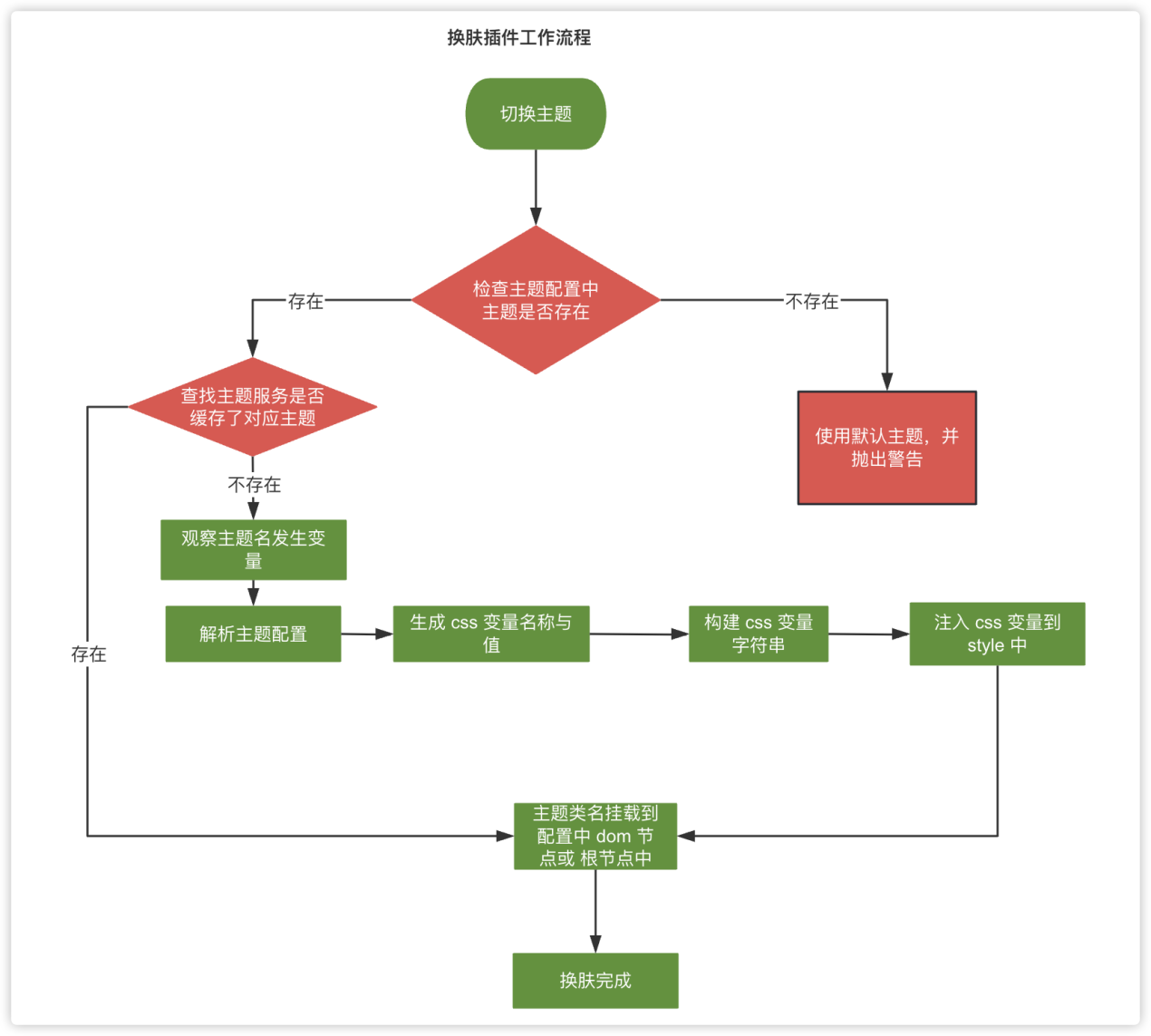
## 运行视图

### Table组件运行态架构设计

### 主题切换运行的设计

主题换肤通过 vue 插件的形式进行全局依赖注入，提供 composition api 进行换肤操作。

换肤运行流程如下：



组件换肤支持用户调用 api 的形式进行换肤，大致原理是利用 vue 的 provide inject ，进行全局注入。

用户调用换肤插件提供的 compostion api 进行换肤操作，并支持动态注入皮肤主题。

对已经注入的皮肤进行缓存，减少主题解析流程。

主题配置分为 color,font模块。配置是将 css自定义变量与 js 做映射。

### 全局定制运行的架构设计

### 组件支持键盘操作运行视图

## 数据视图

### Table 组件数据视图

### 数据存储与分布

### Table 组件数据管理策略

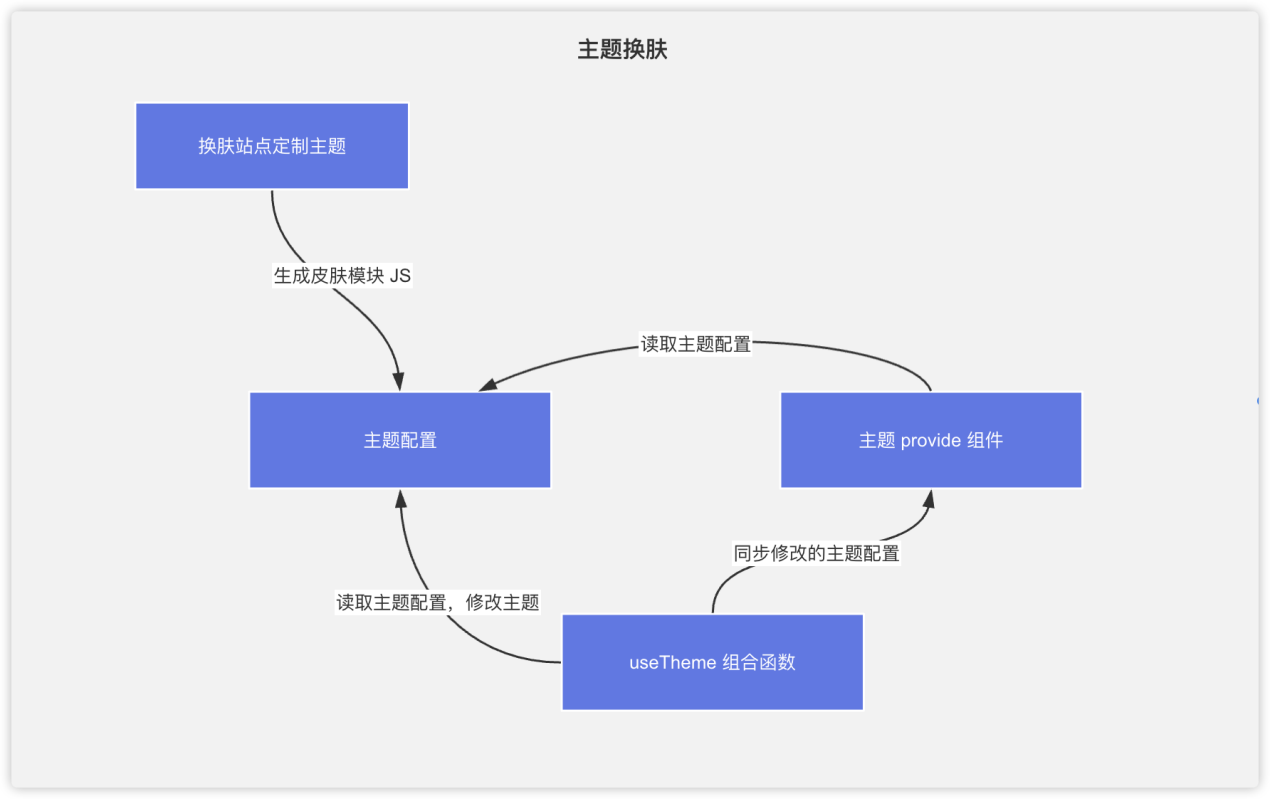
#### 数据存储机制

#### 数据同步机制

# 4.关键功能模块设计

## 主题换肤模块设计

主题换肤由三部分组成如下所示：



换肤站点：换肤站点主要是自定义主题预览，生成自定义皮肤模块。

主题配置：主题配置为主题全局配置支持自定义主题。

主题配置的接口定义 大致如下

interface Theme{

defaultTheme:string,

themes: ThemeDefinition

}

interface ThemeDefinition{

color:{

[propName: string]: string;

}

fontSize:string

}

主题 provide 组件：为提供主题上下文的组件，用来设置页面全局的主题样式。并支持主题嵌套。

useTheme组合函数用于动态切换主题，修改主题配置，以及获取当前主题。

主题配置默认预制 light,dark 主题，css 变量的映射参考主题换肤功能边界说明的 css 变量表。

## 全局定制

## 全键盘方案功能模块设计

### 功能性需求

### 非功能性需求

### 子系统的组件划分、组件协作图

### 本CBB与各层组件的协作图

## Table 组件方案设计

### 功能性需求

### 非功能性需求

支持

### 子系统的组件划分、组件协作图

### 重点功能的实现协作图

# 非功能性架构设计

## 性能设计

## 可靠性设计

### 异常处置设计

## 安全设计

## 可维护性设计

### 安装升级

### 排查辅助

### 生产环境测试设计

## 可测试性设计

### 测试数据设计

### 单元测试和模块测试的可测性设计

### 系统验证测试的可测性设计

## 跨平台性设计

## 其他非功能性设计

# 模板修订记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 批准日期 | 修订说明 | 编制部门 | 编制人员 |
| V1.0 | 2022/08/03 | 新增 | 产品技术管理总部 | 朱龙、寇旭、杨珍良、黄华、陆旭、张磊、赖毓泉 |