插件接入仪表盘主题开发者指南

背景

为了更好的展示效果及用户体验，飞书多维表格在仪表盘的能力上进行了升级，增加了仪表盘主题，针对不同的主题设计了相应的配色方案，为了能够让仪表盘插件在不同主题下呈现更好的展示效果，需要开发者根据以下这个文档稍作调整。

相关概念

仪表盘主题

仪表盘主题

外观设置

- 仪表盘新增功能，可用于快速切换当前仪表盘的视觉风格，包括背景色、字体颜色等
- 仪表盘主题的**优先级大于**外层的外观配置

环境配置

加入群聊

曹诚 邀请你加入飞书群，快点击https://go.larkoffice.com/join-chat/149k05a7-b1be-47b8-8f69-74a2a92cdf4e加入吧！

版本环境

❗ 内测过程版本不稳定，请随时关注下方版本变更情况。

- 引入内测版本 SDK：

```
pnpm install https://lf3-static.bytednsdoc.com/obj/eden-cn/jjjpceh7nulojvhj/@lark-base-open-js-sdk-0.4.1-beta.5.tgz

// 仪表盘统一入口
import { dashboard } from '@lark-base-open/js-sdk';
```

- 在 Base 文档链接 URL 后方拼接如下查询参数：

```
vb=1.0.4.9969&v=1.0.15.2486&isDashboardV2=true
```

接入步骤

引入全局 CSS

在顶层组件中引入下方代码：

```
import '@lark-base-open/js-sdk/dist/style/dashboard.css';
```

仪表盘主题定义

```
// 主题相关数据结构定义
interface IDashboardTheme {
    colorThemeId: string; // 当前主题唯一标识
    theme: ThemeMode; // 深浅色
    /** 当前主题下图表背景颜色 */
    chartBgColor: string;
    /** 当前主题下字体颜色色板，css token 值 */
    labelColorTokenList: string[];
    /** 当前主题下的系列色板，hex 值 */
    themePalette: Hex[];
}
```

- `colorThemeId`：主题唯一标识
- `theme`：当前主题是深色还是浅
- `chartBgColor`：代表图表在当前主题下的背景颜色，你需要**将该值赋给插件的内容区域**。
- ~~`globalThemeBgColor`：外层主题的全局背景色，作用在图表下方。~~
- `labelColorTokenList`： 代表原生图表中的字体色板，适用于单色值的业务场景，如倒计时插件：

- `themePalette`：代表**当前主题下的系列色板**，适用于多色值的业务场景，如地图插件：

获取主题信息

使用 `dashboard.getTheme` 方法获取当前主题信息：

```
// 以 react hook 为例
const [themeConfig, setThemeConfig] = useState();

const theme = await dashboard.getTheme();
setThemeConfig(theme);
```

监听主题切换

> ❗ **务必移除原代码块中 `bridge.onThemeChange` 相关逻辑，否则会造成冲突**

使用 `dashboard.onThemeChange` 方法监听当前主题切换，响应数据将返回切换后的主题信息：

```
// 以 react hook 为例
const [themeConfig, setThemeConfig] = useState();

dashboard.onThemeChange(res => {
    setThemeConfig(res.data);
});
```

接入主题背景色

以 React 为例，一个仪表盘插件需要由 `内容区域` + `配置面板` 共同组成：

```
const App = () => {
    // ...

    return (
        // 内容区域
        <Content />
        {
        // 配置面板
            isConfig ? <ConfigPanel /> : null
        }
    )
}
```

内容区域

`chartBgColor` 代表图表在当前主题下的背景颜色，**你需要将该值赋给插件的内容区域**。

```
const Content = () => {
    // ...
    const [currentTheme, setCurrentTheme] = useState();

    useEffect(() => {
        const theme = await dashboard.getTheme();
        setCurrentTheme(theme);
    }, [])

    return (<div style={{ background: currentTheme.chartBgColor }}></div>)
}
```

配置面板

请勿给配置面板设置任何背景颜色，直接继承外部容器的背景色即可，以防万一**你可以手动将配置面板的背景色设置为透明**：

```
const ConfigPanel = () => {
    // ...

    return (<div style={{ background: 'transparent' }}></div>)
}
```

接入主题色板

如果你的插件仅需要消费单个颜色，消费 `labelColorTokenList` 即可：

```
// 简单示意
const ColorPicker = () => {
    const [currentTheme, setCurrentTheme] = useState();
    // 当前字体色值
    const [lableColor, setLableColor] = useState();

    useEffect(() => {
        const theme = await dashboard.getTheme();
        setCurrentTheme(theme);
    }, [])

    const handleSelectColor = (labelToken) => {
        setLableColor(labelToken);
    }

    return (
        <div>
            {
                (currentTheme.labelColorTokenList ?? []).map(labelToken => {
                    return (<div style={{ background: labelToken }} onClick={()
  => handleSelectColor(labelToken)}></div>)
                })
            }
        </div>
    )
}

```

同理，如果你的业务场景中需要消费一系列的值，消费 `themePalette` 即可，该色板会随着仪表盘主题的切换而变化。

详细示例 Demo

https://github.com/Lark-Base-Team/count-down-theme

FAQ