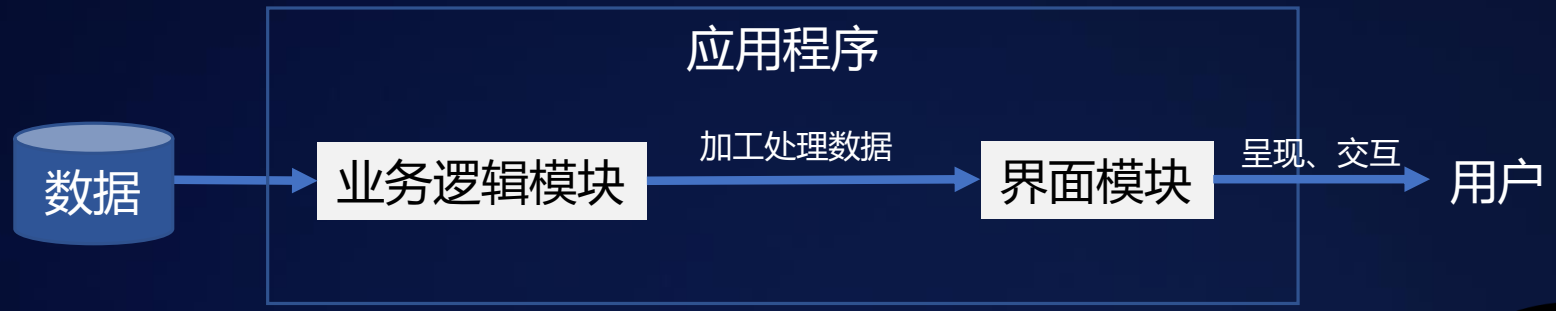
华为分布式日历应用开发实践

华为产品中的分布式日历应用

1、分布式应用架构
- 遵循MVVM
- 一套架构适配多设备形态

2、分布式数据服务
- 三步实现日程同步

应用架构要满足两点要求



日历应用特点

- 界面：易变更，产品定制性强
- 业务逻辑：日历数据计算逻辑复杂，但多产品通用

界面开发要简单、业务逻辑实现要健壮高效

HarmonyOS 应用原生支持JS与Java两种语言

Android 应用

Java

界面　业务逻辑

Java：界面命令式开发，能实现复杂的界面逻辑，但不够简单、灵活

RN、Vue 应用

Javascript

界面　业务逻辑

JS：复杂运算速度较慢,代码的隐私性差

HarmonyOS 日历应用

Javascript FA

界面

简单

声明式开发，纯粹做界面

组件间接口

Java PA

业务逻辑

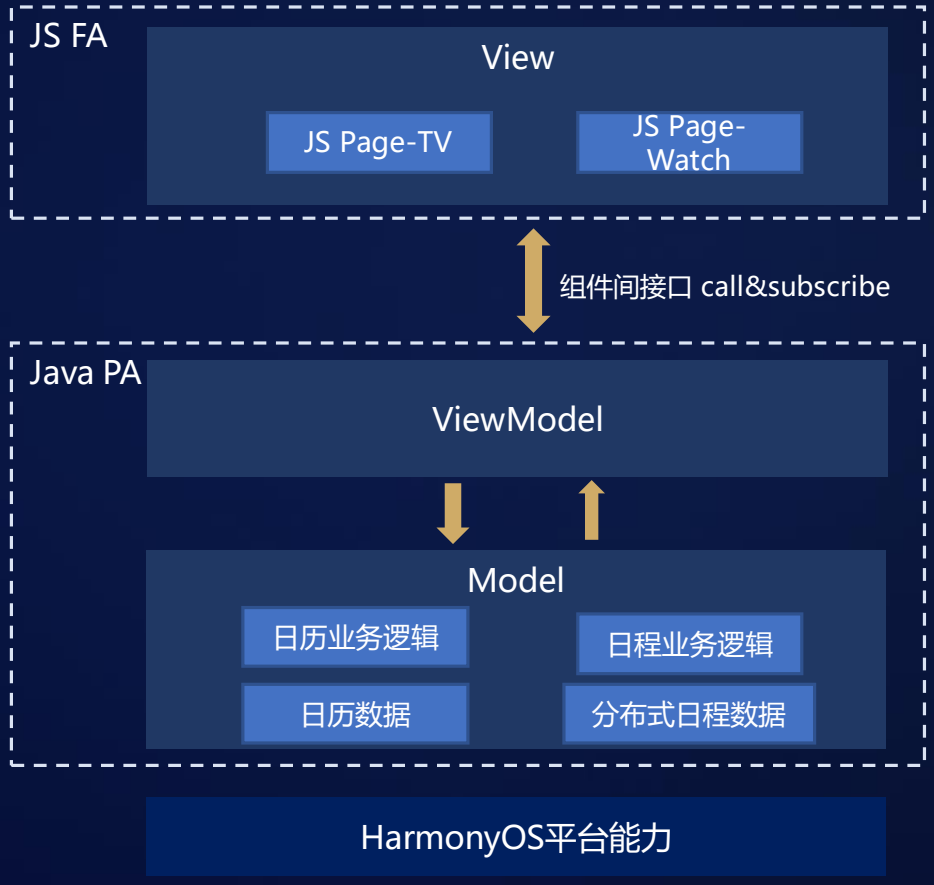
高效健壮

可使用Java强大类库及高性能
可复用、可独立测试

- 支持JS、Java混合编程，各取所长

一套架构适配多设备形态

- MVVM分层，多设备界面适配只需要修改View层

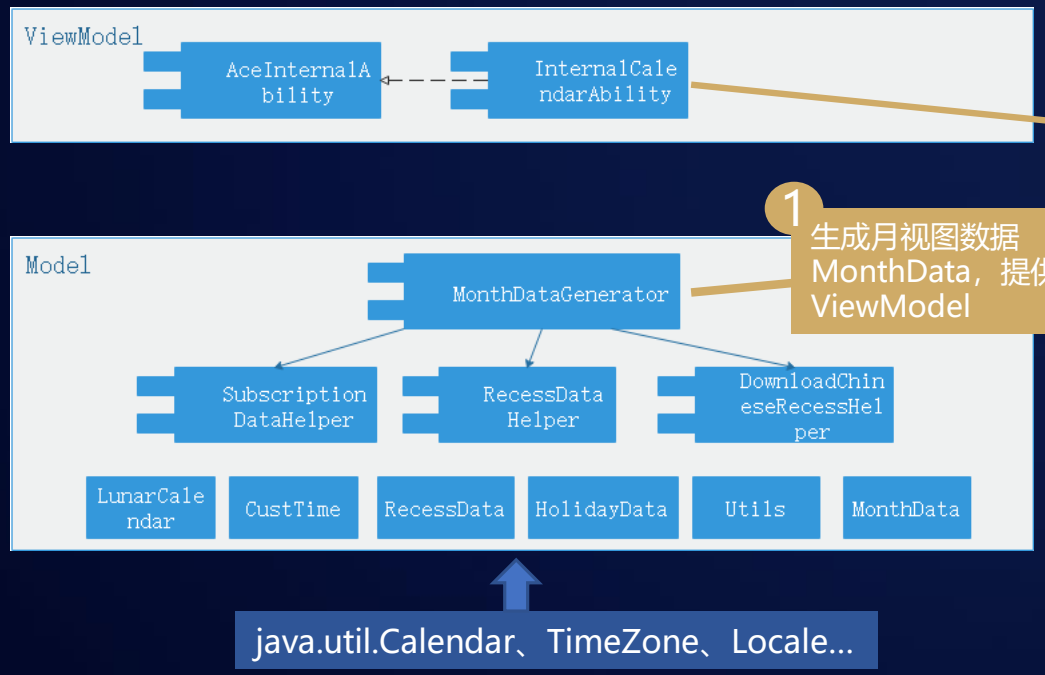


View视图层：
- 多JS Page针对不同设备进行深度UI定制

ViewModel视图模型层：
- JS与Java的数据绑定

Model模型层：
- 业务逻辑与数据
- 分布式日程数据

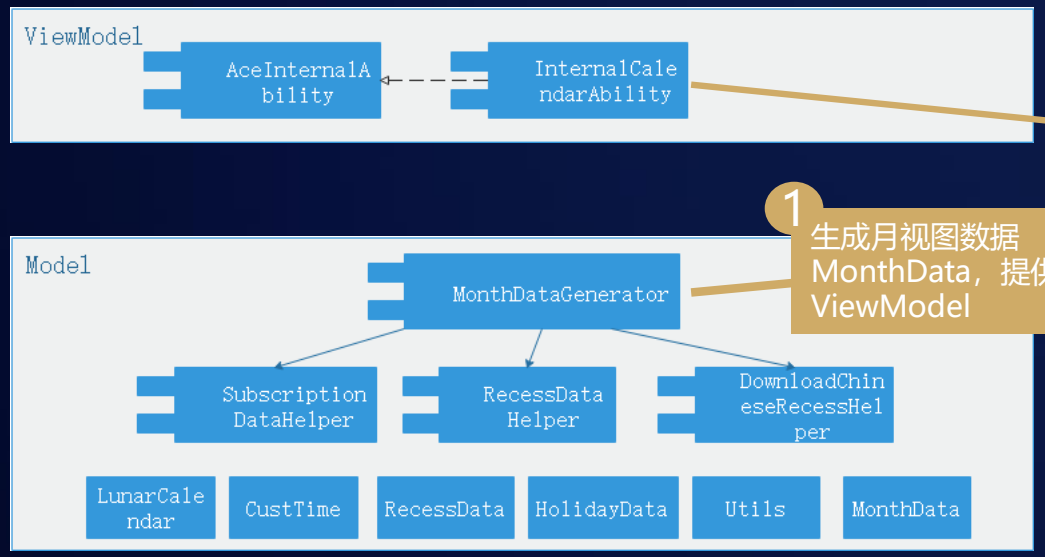
Model层数据经过ViewModel传递到View层界面



1 生成月视图数据MonthData，提供给ViewModel

```java
public class InternalCalendarAbility extends AceInternalAbility
        implements AceInternalAbility.AceInternalAbilityHandler {
    @Override
    public boolean onRemoteRequest(int code, MessageParcel data, MessageParcel reply,
                                   MessageOption option)
        throws RemoteException {
        LogUtil.info(TAG, msg: "onRemoteRequest, code: " + code);
        if (code == MONTH_VIEW_DATA) {
            // 1001月视图数据请求
            String jsonStr = data.readString();
            MonthData monthData = dataGenerator.getMonthData(jsonStr, mContext);
            reply.setCapacity(CAPACITY * CAPACITY_SIZE);
            reply.writeString(ZSONObject.toZSONString(monthData));
        } else if (code == GET_CURRENT_DAY_DATA) {
            // 1002当前天数据请求
```

Model层数据经过ViewModel传递到View层界面

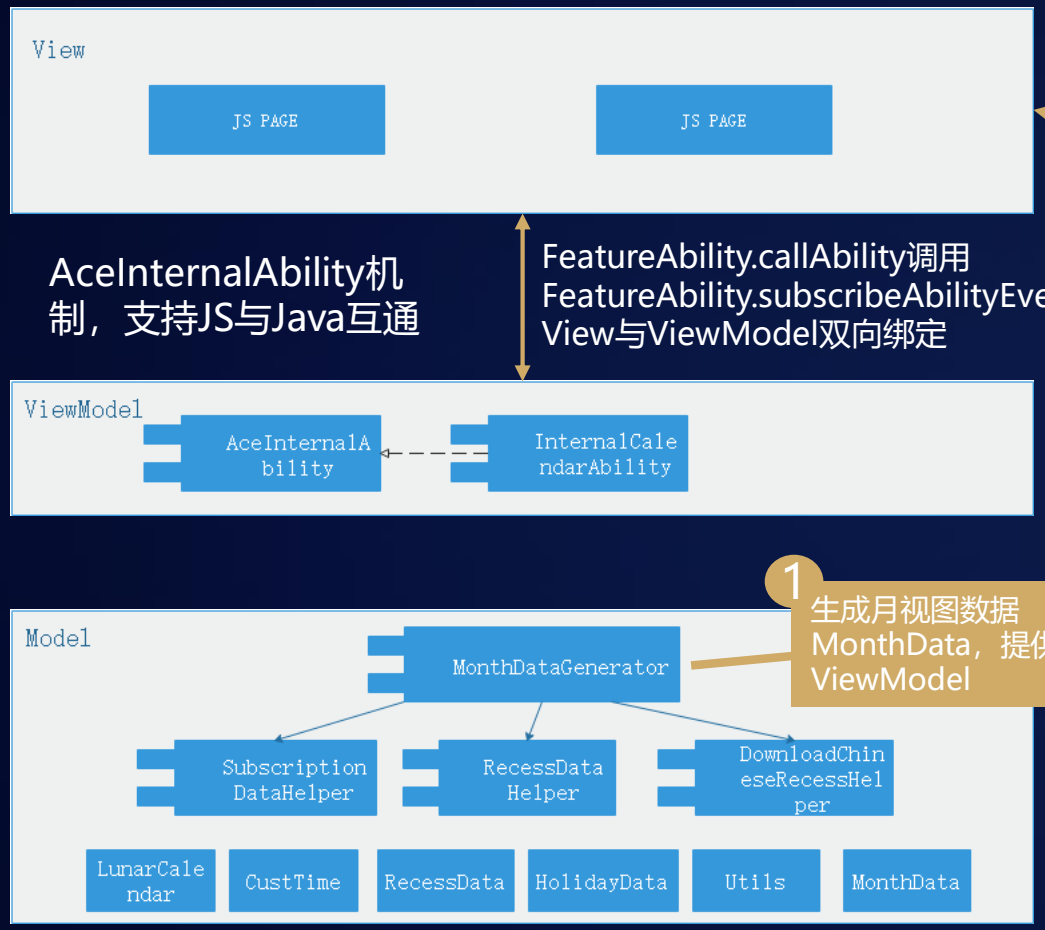


1 生成月视图数据MonthData，提供给ViewModel

```java
public class InternalCalendarAbility extends AceInternalAbility
        implements AceInternalAbility.AceInternalAbilityHandler {
    @Override
    public boolean onRemoteRequest(int code, MessageParcel data, MessageParcel reply,
                                   MessageOption option)
        throws RemoteException {
        LogUtil.info(TAG, msg: "onRemoteRequest, code: " + code);
        if (code == MONTH_VIEW_DATA) {
            // 1001月视图数据请求
            String jsonStr = data.readString();
            MonthData monthData = dataGenerator.getMonthData(jsonStr, mContext);
            reply.setCapacity(CAPACITY * CAPACITY_SIZE);
            reply.writeString(ZSONObject.toZSONString(monthData));
        } else if (code == GET_CURRENT_DAY_DATA) {
            // 1002当前天数据请求
```

2 ViewModel层序列化MonthData

Model层数据经过ViewModel传递到View层界面



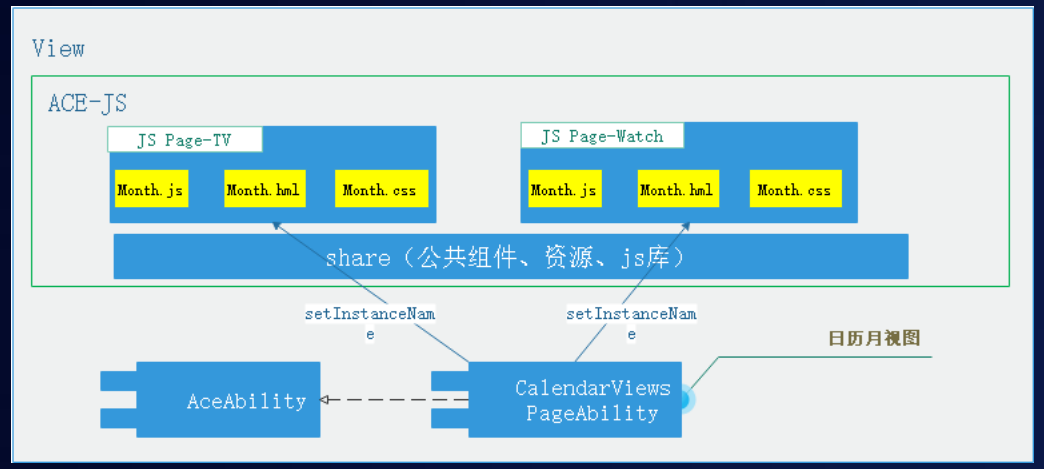
1 生成月视图数据MonthData，提供给ViewModel

3 通过FeatureAbility通信接口返回MonthData

```
public class InternalCalendarAbility extends AceInternalAbility
        implements AceInternalAbility.AceInternalAbilityHandler {
    @Override
    public boolean onRemoteRequest(int code, MessageParcel data, MessageParcel reply,
                                   MessageOption option)
        throws RemoteException {
        LogUtil.info(TAG, msg: "onRemoteRequest, code: " + code);
        if (code == MONTH_VIEW_DATA) {
            // 1001月视图数据请求
            String jsonStr = data.readString();
            MonthData monthData = dataGenerator.getMonthData(jsonStr, mContext);
            reply.setCapacity(CAPACITY * CAPACITY_SIZE);
            reply.writeString(ZSONObject.toZSONString(monthData));
        } else if (code == GET_CURRENT_DAY_DATA) {
            // 1002当前天数据请求
```

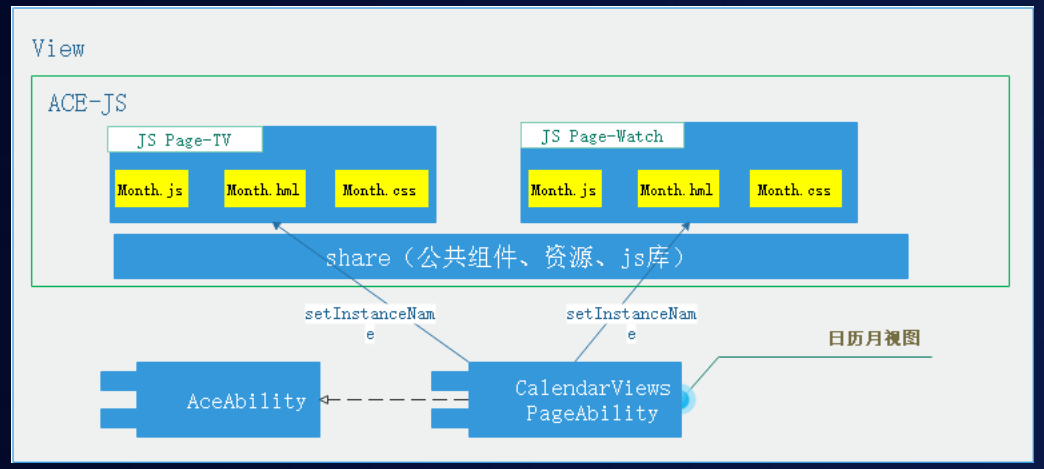
2 ViewModel层序列化MonthData

View层做到多产品界面深度定制



1、PageAbility、页面指定启动

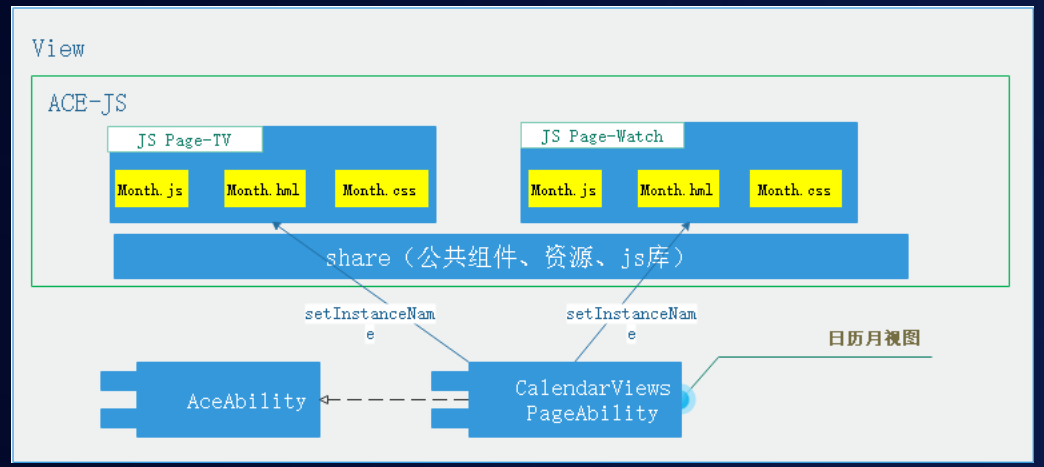
View层做到多产品界面深度定制




```
<div class="container">
    <div class="container_watch">
        <div class="month_title_container">
            <text class="month_title">{{ currentMonth + 1 }} {{ $t('strings.month') }}</text>
        </div>
        <div class="calendar_container">
            <calendar id="calendar" class="calendar_tv" onselectedchange="selectedChange" dateadapter={{calendarData}}>
        </div>
    </div>
</div>
```

1、PageAbility、页面指定启动

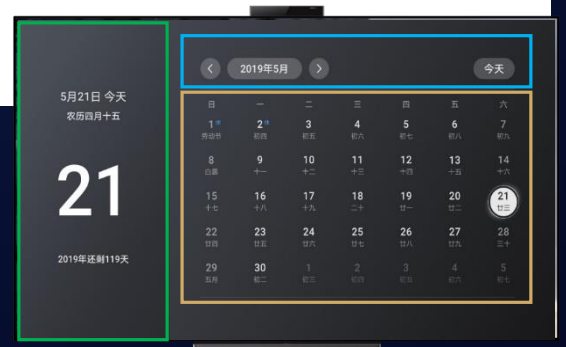
View层做到多产品界面深度定制



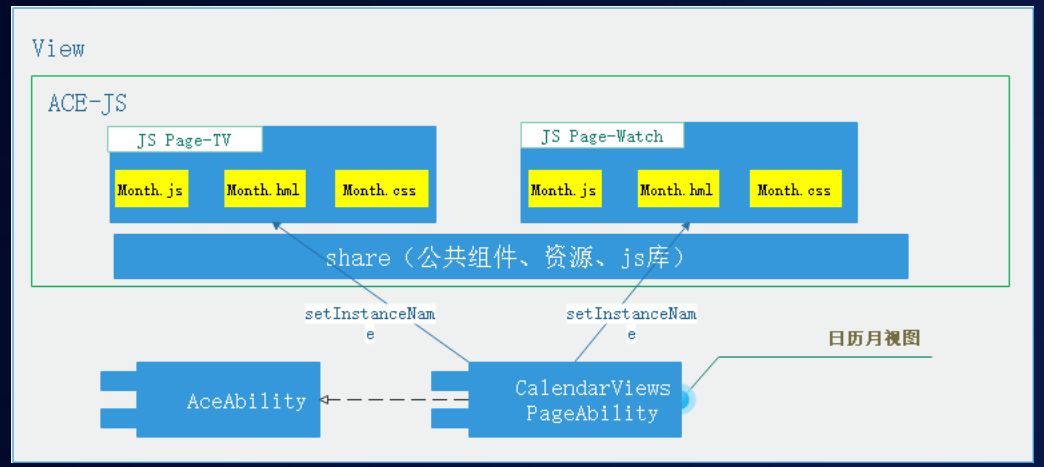
1、PageAbility、页面指定启动

```
<div class="container">
    <div class="container_watch">
        <div class="month_title_container">
            <text class="month_title">{{ currentMonth + 1 }} {{ $t('strings.month') }}</text>
        </div>
        <div class="calendar_container">
            <calendar id="calendar" class="calendar_tv" onselectedchange="selectedChange" dateadapter={{calendarData}}>
        </div>
    </div>
</div>
```


```
<div class="container" style="background:linear-gradient(#2E3033, #1A1A1A)">
    <div class="date_detail_left" style="background:linear-gradient(to bottom right, #00000000, #66000000)">
        <!--日期详情部分-->
        <...>
    </div>
    <div class="date_calendar_right">
        <div class="right-container">
            <!--日期选择部分-->
            <div class="up_right">
                <...>
            </div>
            <!--日期部分-->
            <div class="down_right">
                <div><divider class="divider"></divider></div>
                <calendar id="calendar" onselectedchange="selectedChange" dateadapter={{calendarData}}
                    showlunar="{{this.lunarcalendar}}" offdays="{{this.offday}}" date="{{todayDate}}"> </calendar>
            </div>
        </div>
    </div>
</div>
```



View层做到多产品界面深度定制



1、PageAbility、页面指定启动

```
<div class="container">
    <div class="container_watch">
        <div class="month_title_container">
            <text class="month_title">{{ currentMonth + 1 }} {{ $t('strings.month') }}</text>
        </div>
        <div class="calendar_container">
            <calendar id="calendar" class="calendar_tv" onselectedchange="selectedChange" dateadapter={{calendarData}}>
        </div>
    </div>
</div>
```


```
<div class="container" style="background:linear-gradient(#2E3033, #1A1A1A)">
    <div class="date_detail_left" style="background:linear-gradient(to bottom right, #00000000, #66000000)">
        <!--日期详情部分-->
        <...>
    </div>
    <div class="date_calendar_right">
        <div class="right-container">
            <!--日期选择部分-->
            <div class="up_right">
                <...>
            </div>
            <!--日期部分-->
            <div class="down_right">
                <div><divider class="divider"></divider></div>
                <calendar id="calendar" onselectedchange="selectedChange" dateadapter={{calendarData}}
                    showlunar="{{this.lunarcalendar}}" offdays="{{this.offday}}" date="{{todayDate}}"> </calendar>
            </div>
        </div>
    </div>
</div>
```

2、calendar组件
按需声明式开发

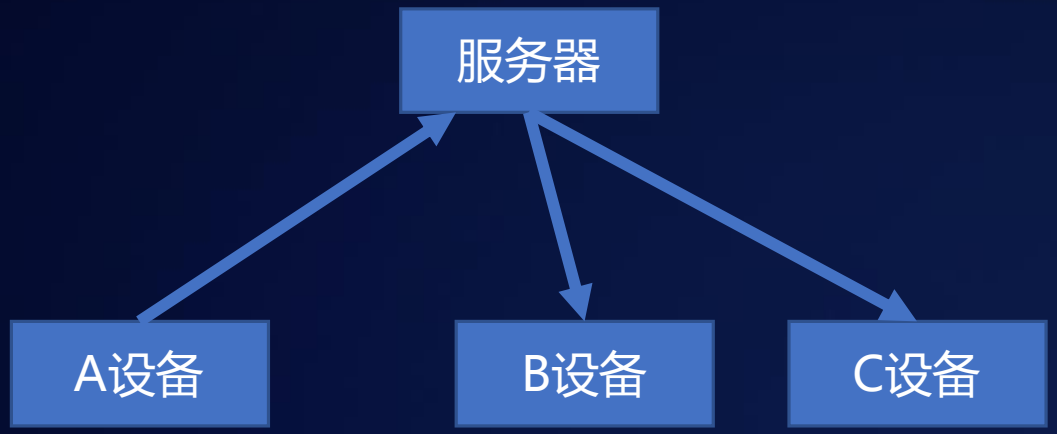
多套HML设计支持产品深度定制

快速完成新设备形态拓展

分布式数据服务实践

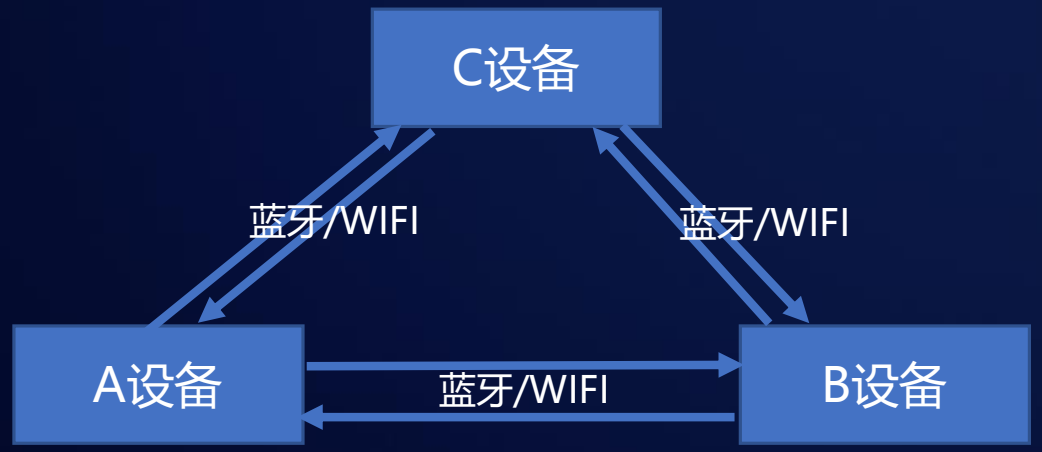
分布式数据

分布式特性 传统方式实现痛点



云侧同步方案

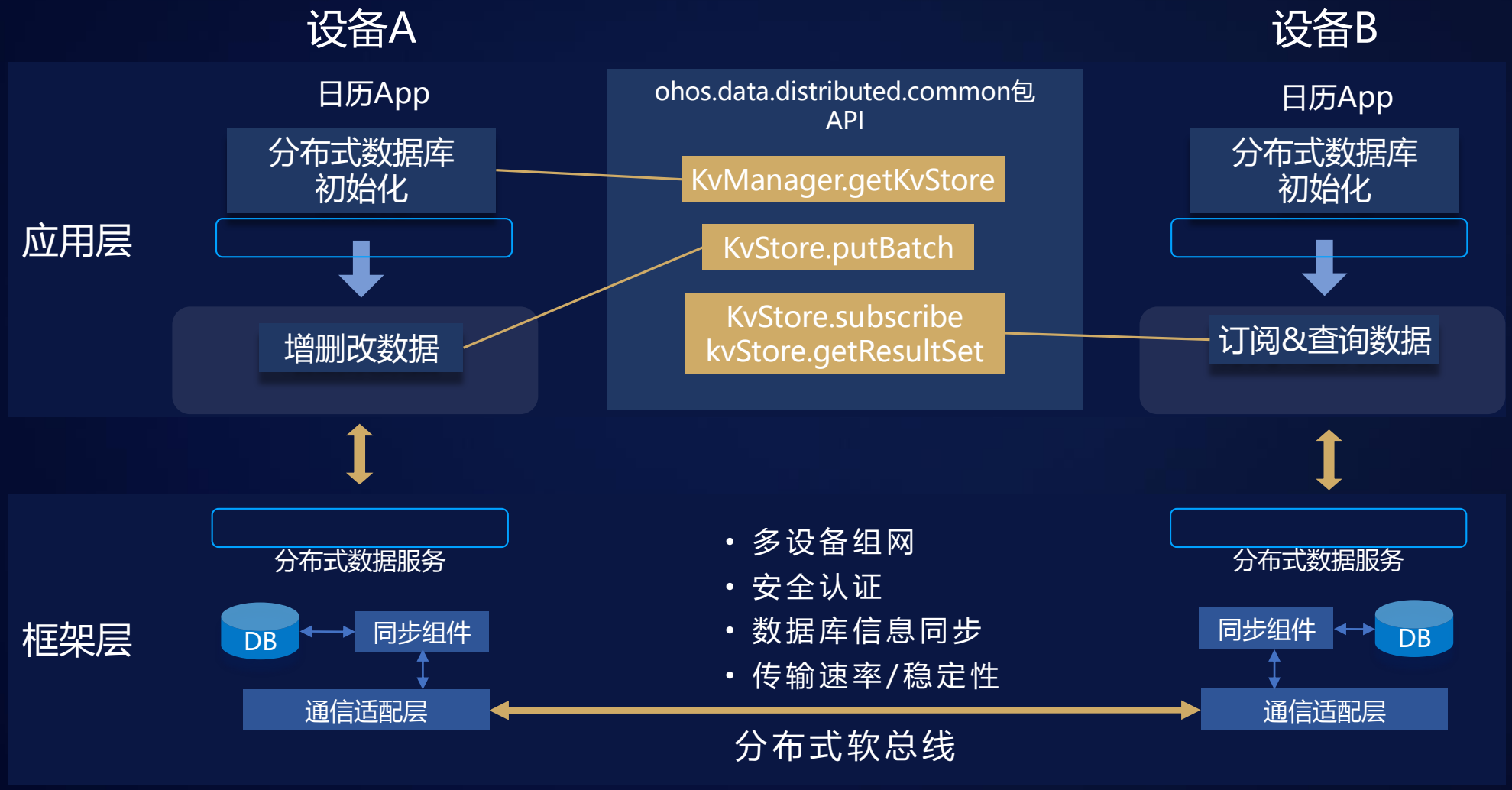
- 依赖远端的服务器，开发/维护成本高
- 受限于数据流量及网络带宽，不适用于构建超级终端场景



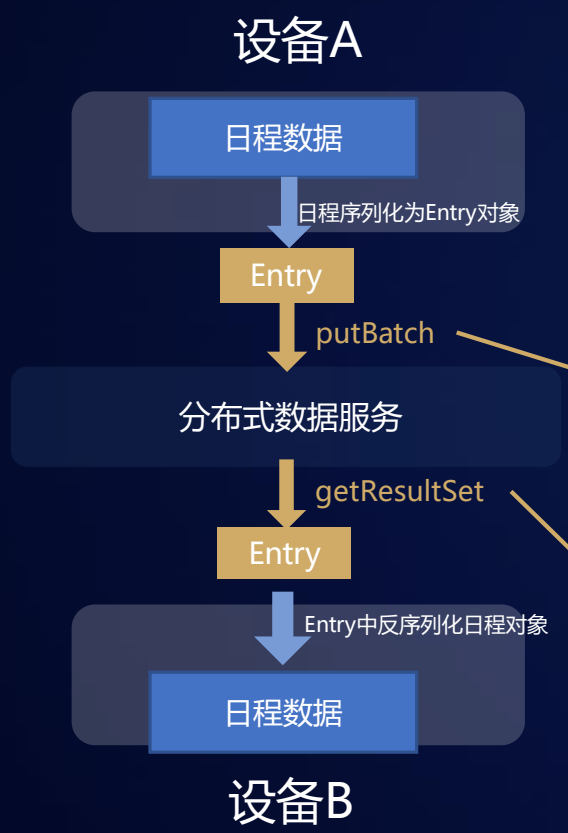
P2P近场共享

- 传输硬件强相关，考虑多样传输协议
- 面临1对多的组网、同步策略、多版本兼容性，技术难度高，编码量大

日历分布式数据服务实践



使用 HarmonyOS 分布式数据服务只需要3步



1、设备A，B 分别初始化分布式数据库KVStore

```
manager = kvManagerFactory.createKvManager(config);
options = new Options();
options.setCreateIfMissing(true).setEncrypt(false).
        setBackup(false).setAutoSync(true).setKvStoreType(KvStoreType.SINGLE_VERSION);
kvStore = manager.getKvStore(options, "singleCalendarDatabase");
```

2、设备A 将日程数据序列化为Entry，并写入KvStore

```
for (HwSyncEvent event : events) {
    // 将日程列表转换为Entry列表
    insertList.add("calendar_" + event.eventUuid, new Entry(Value.get(gson.toJson(event))));
}
kvStore.putBatch(insertList);  // 将所有日程信息插入KV数据库
```

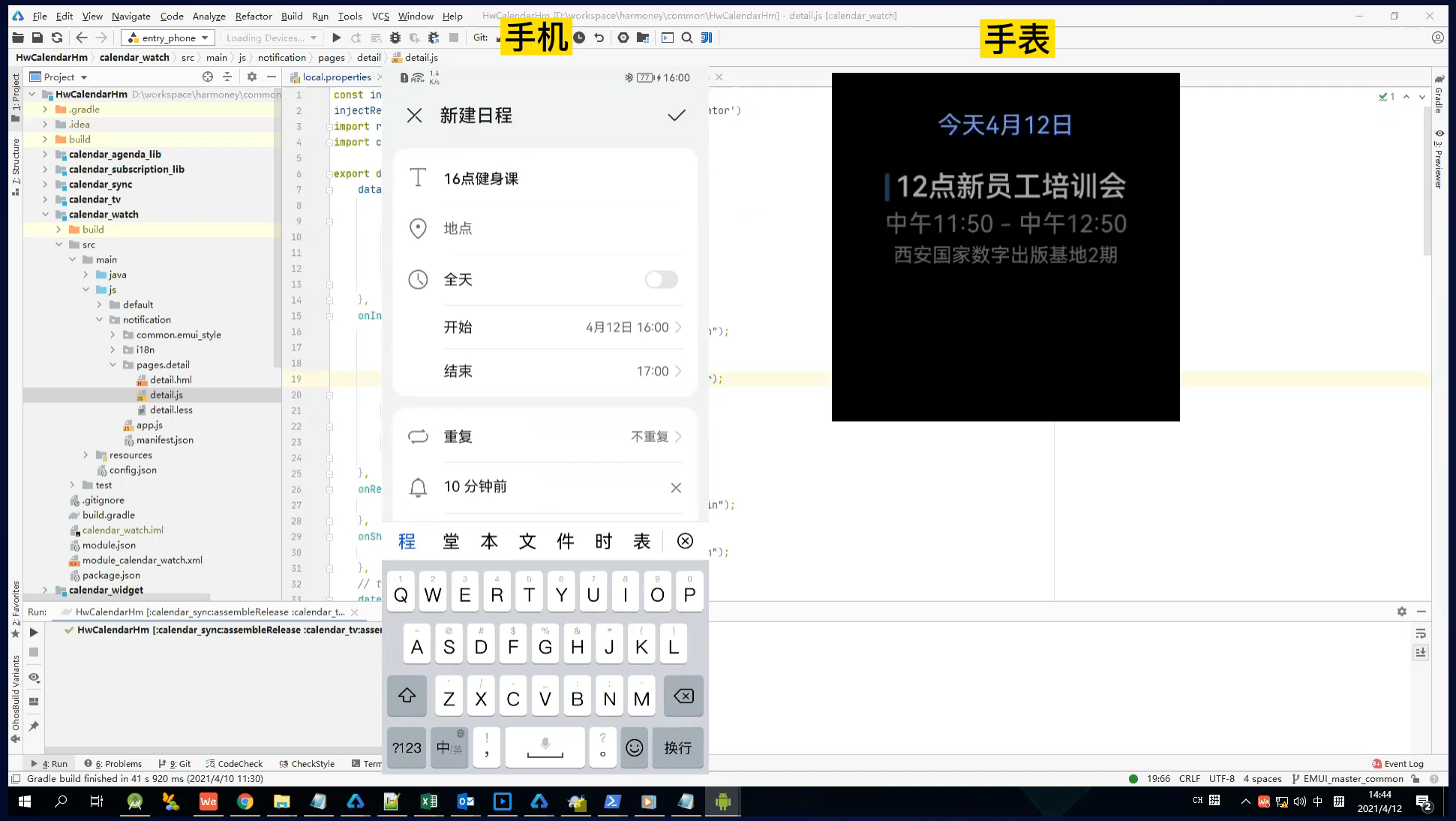
3、设备B 订阅及查询日程数据

```
kvStore.subscribe(SubscribeType.SUBSCRIBE_TYPE_REMOTE, new KvStoreObserverClient(mContext));
```

```
resultSet = kvStore.getResultSet("calendar_" + uuid);
if (resultSet.goToNextRow()) {
    HwSyncEvent hwSyncEvent = new Gson().fromJson(resultSet.getEntry().getValue().getString(), HwSyncEvent.class);
    return hwSyncEvent;
}
```

开发者可以像操作本地数据库一样，不用关心数据如何同步

日历分布式日程管理 效果展示

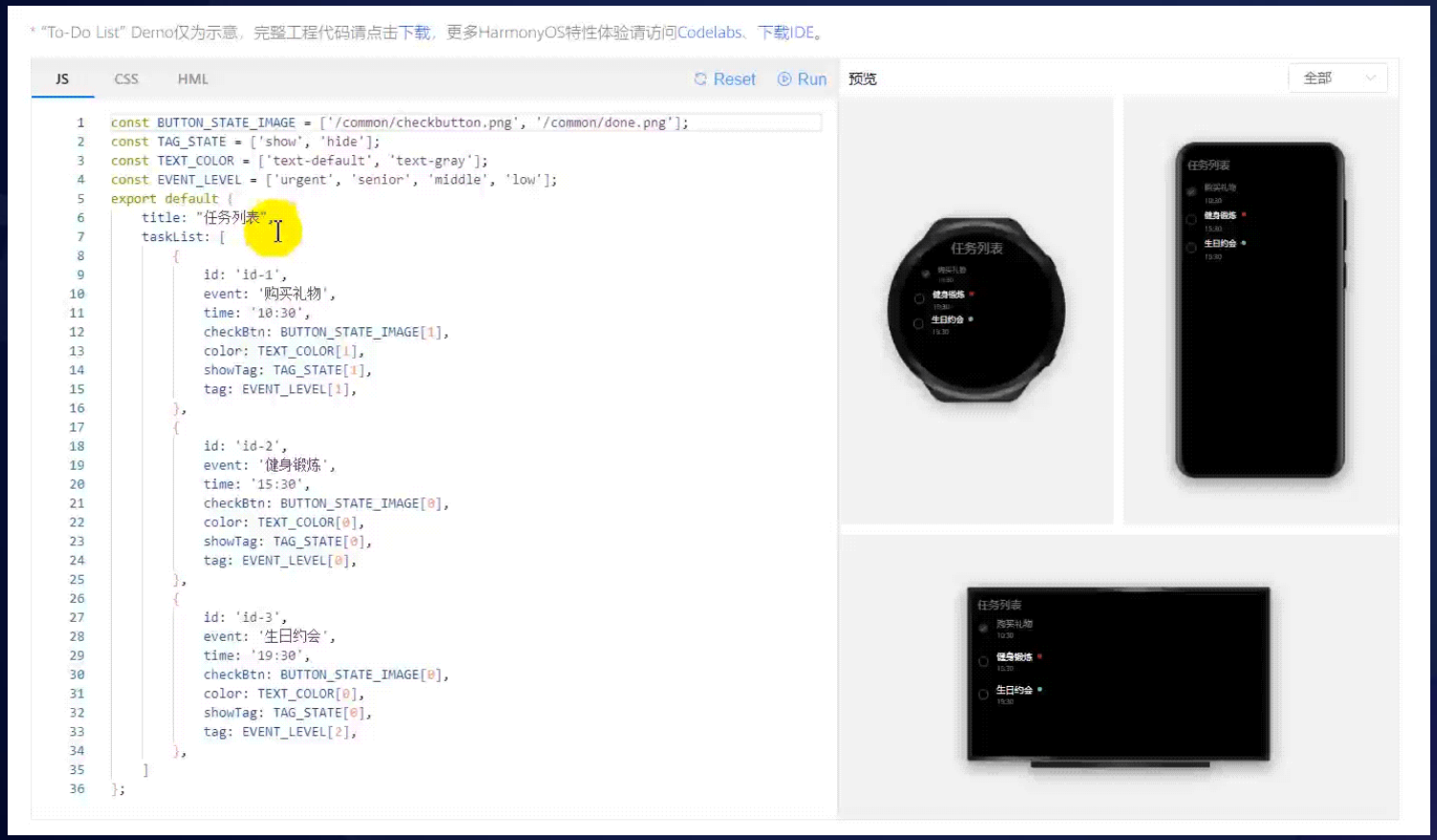


总结

1.UI与业务解耦，JS与Java混合编程，一套架构快速完成全场景设备的界面定制

2.使用分布式数据服务能力KvStore接口，三步简单高效地完成分布式特性

扫码在线体验HarmonyOS应用开发




```
const BUTTON_STATE_IMAGE = ['/common/checkbutton.png', '/common/done.png'];
const TAG_STATE = ['show', 'hide'];
const TEXT_COLOR = ['text-default', 'text-gray'];
const EVENT_LEVEL = ['urgent', 'senior', 'middle', 'low'];
export default {
    title: "任务列表",
    taskList: [
        {
            id: 'id-1',
            event: '购买礼物',
            time: '10:30',
            checkBtn: BUTTON_STATE_IMAGE[1],
            color: TEXT_COLOR[1],
            showTag: TAG_STATE[1],
            tag: EVENT_LEVEL[1],
        },
        {
            id: 'id-2',
            event: '健身锻炼',
            time: '15:30',
            checkBtn: BUTTON_STATE_IMAGE[0],
            color: TEXT_COLOR[0],
            showTag: TAG_STATE[0],
            tag: EVENT_LEVEL[0],
        },
        {
            id: 'id-3',
            event: '生日约会',
            time: '19:30',
            checkBtn: BUTTON_STATE_IMAGE[0],
            color: TEXT_COLOR[0],
            showTag: TAG_STATE[0],
            tag: EVENT_LEVEL[2],
        },
    ]
};
```

HarmonyOS应用开发在线体验

HarmonyOS开发者微信公众号