

ALIBABA CLOUD

阿里云

DataV数据可视化  
蓝图编辑器使用说明

文档版本：20210809

 阿里云

## 法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

# 通用约定

格式	说明	样例
 危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 危险 重置操作将丢失用户配置数据。
 警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	 警告 重启操作将导致业务中断，恢复业务时间约十分钟。
 注意	用于警示信息、补充说明等，是用户必须了解的内容。	 注意 权重设置为0，该服务器不会再接受新请求。
 说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击设置> 网络> 设置网络类型。
<b>粗体</b>	表示按键、菜单、页面名称等UI元素。	在结果确认页面，单击 <b>确定</b> 。
Courier字体	命令或代码。	执行 <code>cd /d C:/window</code> 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	<code>bae log list --instanceid</code> <i>Instance_ID</i>
[ ] 或者 [a b]	表示可选项，至多选择一个。	<code>ipconfig [-all -t]</code>
{ } 或者 {a b}	表示必选项，至多选择一个。	<code>switch {active stand}</code>

# 目录

1.蓝图编辑器与节点编程的对比差异	05
2.配置蓝图编辑器	06
3.蓝图编辑器详细功能介绍	11
4.调试预览指南	18
5.逻辑节点配置	22
5.1. 概述	22
5.2. 全局节点	24
5.3. 流程控制	26
5.4. 数据处理	30
5.5. 输入设备	37
6.配置表单组	40
7.常见问题	46
7.1. 如何使用Tab列表控制组件显隐	46
7.2. 如何配置数据筛选	52
7.3. 如何动态控制组件样式	58
7.4. 如何动态控制组件高亮	65
7.5. 如何在请求数据接口时传递动态参数	69
7.6. 如何通过合并请求进行数据分发	75
7.7. 时间戳联动组件查询	80
7.8. 如何实现跨屏联动	84
7.9. 开发者自定义组件如何使用蓝图编辑器	89
7.10. 如何使用时间器触发数据定时更新	93
7.11. 如何展示多个实时数据相加结果	98

# 1. 蓝图编辑器与节点编程的对比差异

本文档介绍新版蓝图编辑器与旧版节点编程之间的关系和差异。

节点编程已经升级为蓝图编辑器，旧版本的大屏交互配置从节点编程迁移到新版蓝图编辑器后产生的变化如下表格所示：

节点编程	蓝图编辑器	备注
画布中添加的触发器	升级为分支判断逻辑节点。	升级后将触发器名称保留在节点名称中。
画布中添加的转换器	升级为串行数据处理节点逻辑节点。	升级后将转换器名称保留在节点名称中。
左侧图层列表	升级为导入节点栏，并且只展示已在画布编辑器中被导出到蓝图编辑器的图层。已导出的图层在后期如果取消导出后会标注红色感叹号，且配置的交互均为无效。	无
左侧规则器模板	<ul style="list-style-type: none"><li>管理：统一在DataV控制台首页 &gt; 我的数据 &gt; 代码片段管理中进行管理代码片段。</li><li>添加：在逻辑节点右侧配置面板中的处理方法数据编辑框内右键单击，可将之前保存为模板的代码片段导入直接应用。</li></ul>	无
节点编程界面右下角的预览页	蓝图编辑器预览与画布编辑器的预览功能合并。	无
节点编程界面右上角的应用功能键	应用键功能和大屏预览页功能合并，实时生效，无需再单击应用键。	无
查看日志	升级为调试预览指南。	无

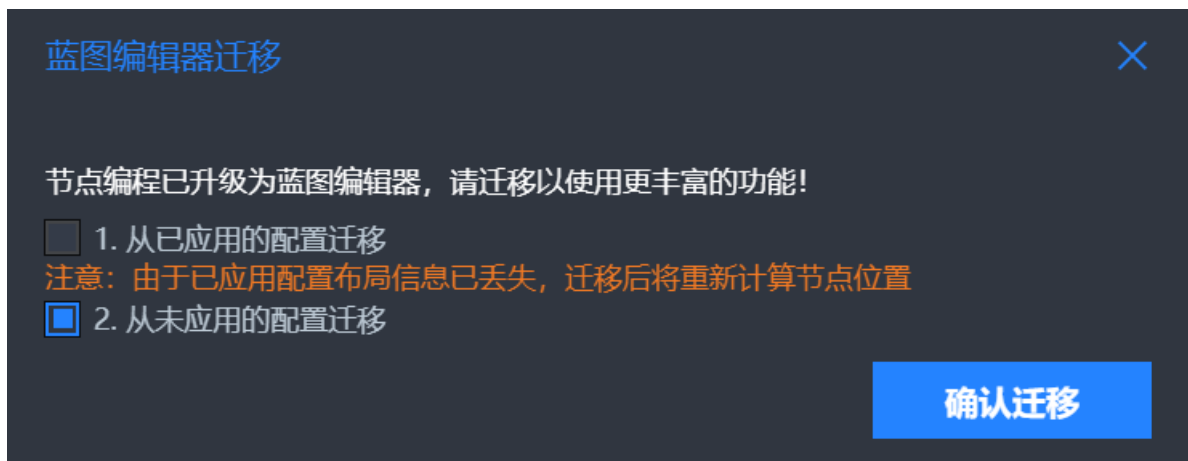
## 2.配置蓝图编辑器

蓝图编辑器一般称为visual programming或者flow based programming，即通过可视化连线的方式，定义图层与图层之间的交互行为。本文为您介绍Dat aV蓝图编辑器的使用方法，帮助您自由管理可视化应用中多个组件之间的交互关系。

### 版本更新迁移说明

节点编程已经升级为蓝图编辑器，首次进入蓝图编辑器时，您需要将旧版本的交互配置迁移至蓝图编辑器中，以使用更丰富的功能。

在蓝图编辑器迁移对话框中，按照以下说明选择迁移方式，完成后单击**确认迁移**即可。



- **从已应用的配置迁移**

如果您的可视化应用已经应用了旧版本的交互配置，可以选择此方式。


 **注意** 由于已应用配置的布局信息已丢失，迁移后将重新计算节点位置。

- **从未应用的配置迁移**

如果可视化应用没有应用过旧版本的交互配置，可以选择此方式。

### 蓝图编辑器的优势

- 蓝图编辑器区别于之前的回调ID，可以保证交互和数据的实时性和同步性。
- 蓝图编辑器支持数据请求合并和数据分发的功能。
- 蓝图编辑器可模块化拆分，专注单个的交互链路，不需要考虑代码的整理和规范，只需要专注于业务规则和交互需求即可。
- 蓝图编辑器支持原厂组件和第三方组件。该功能对开发者型用户较为便利。

 **注意** 用户自行开发的第三方组件，必须是在遵循开发者规范的前提下，才能在Dat aV中的蓝图编辑器页面无缝衔接使用。

### 前提条件

在开始配置蓝图编辑器之前，请先准备好组件的交互需求，并将需要交互的组件在画布编辑器中搭建完成，详情请参见[添加资产](#)。



## 操作步骤

1. 登录Datav控制台。
2. 在我的可视化页面中，选择一个可视化应用，单击编辑。
3. 在画布编辑器页面，单击顶部工具栏的蓝图编辑器图标。



4. 在蓝图编辑器页面，将左侧的导入节点和逻辑节点拖入画布中。

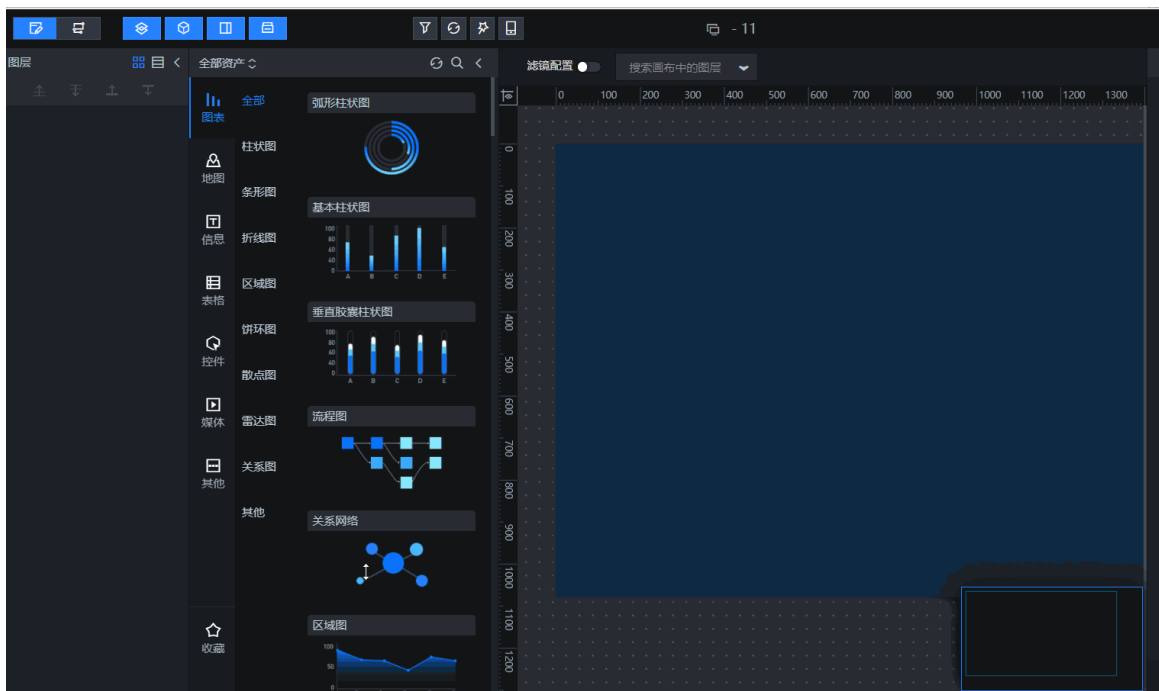
**说明** 如果左侧没有导入节点，可返回画布编辑器，选中图层，右键导出到蓝图编辑器。

5. 根据交互逻辑进行节点连线。
6. 在右侧的配置面板中，完成逻辑节点配置，包括节点名称、上下游事件、上下游动作以及处理方法等。  
添加组件、组件连线、逻辑节点配置的详细操作方法，请参见[操作示例](#)和[蓝图编辑器详细介绍](#)。
7. 配置完成后，单击页面右上角的预览（）图标，查看可视化应用交互效果。
8. 预览成功后，单击页面右上角的发布（）图标，发布可视化应用。

## 操作示例

下面以Tab列表和通用标题组件为例，为您演示蓝图编辑器的具体使用方法。

1. 整理Tab列表和通用标题组件的交互需求。  
例如单击Tab列表中的某一列，可将该列表项中的文本直接显示在通用标题组件上。
2. 在可视化画布编辑器页面，搭建所需要的Tab列表和通用标题组件。



3. 在图层栏内，分别右键单击Tab列表和通用标题组件，选择导出到蓝图编辑器。



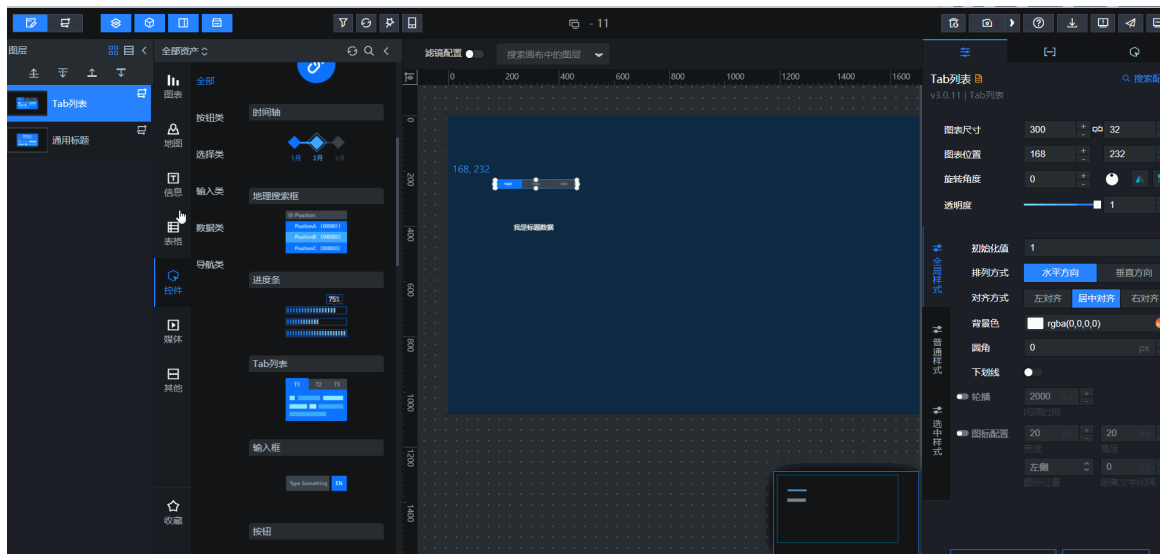
4. 成功导出后，单击画布编辑页面左上角的蓝图编辑器图标。




5. 在蓝图编辑器页面，将导入节点面板中的Tab列表和通用标题，拖至画布上并连线。

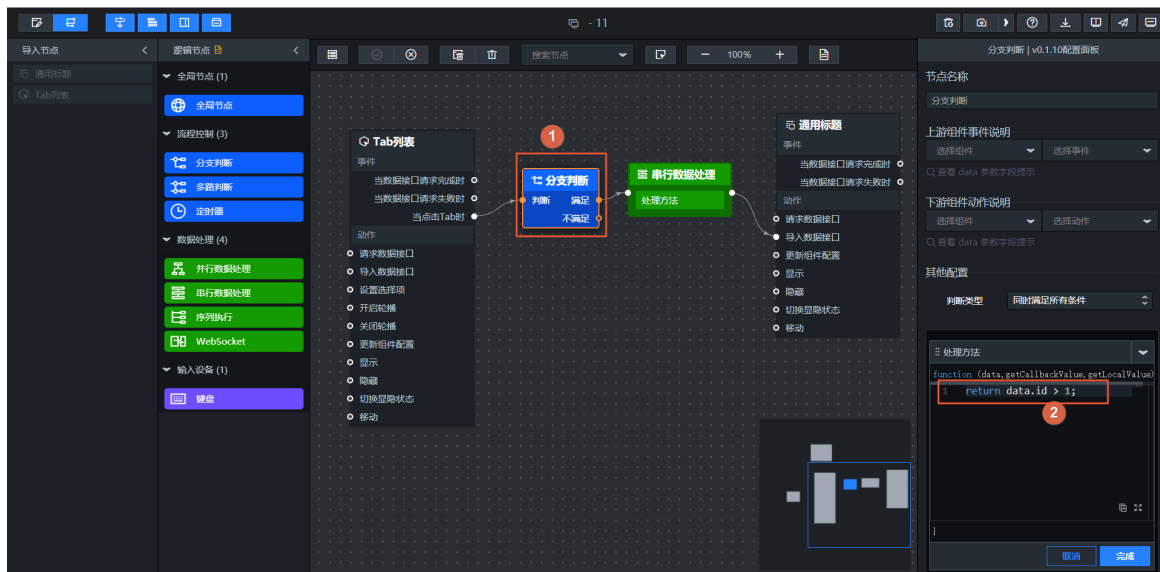
? **说明** 每次在两个节点之间连线都会默认自动在连线之间添加一个串行数据处理节点。您如果不需要使用该逻辑节点可以自行删除。





注意 如果导入节点面板不显示，可单击上方工具栏的导入节点图标（），显示导入节点面板。

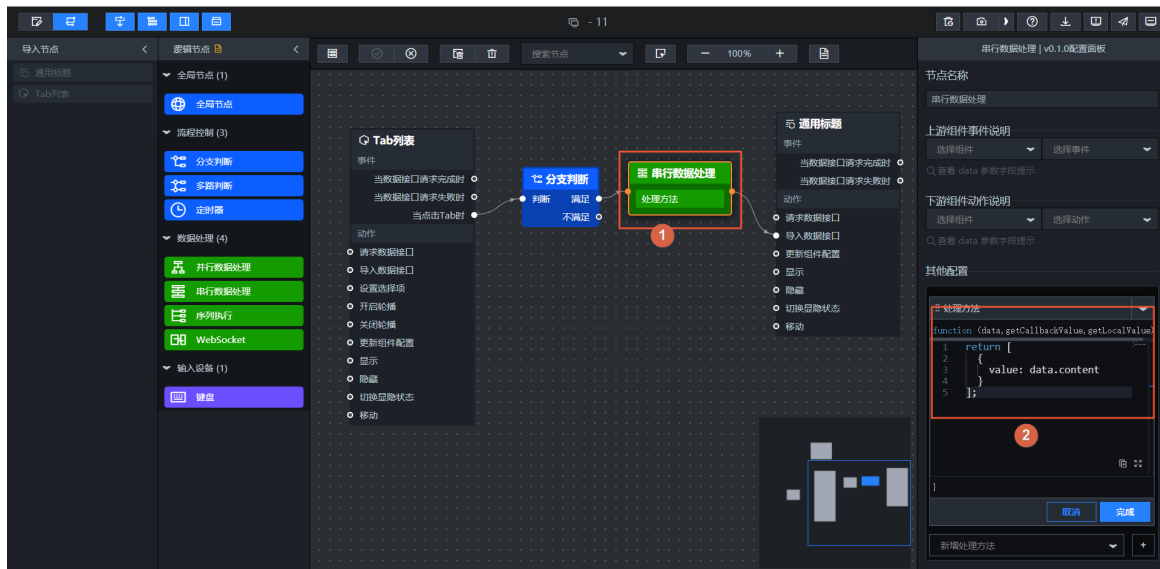
- 6. 在逻辑节点面板中，将分支判断节点拖动到画布上并连线到串行数据处理节点之前。
- 7. 在右侧分支判断配置面板中，配置具体上下游情况和处理方式，设置单击可执行的触发判断条件。



上图中的示例触发判断条件如下。


```
return data.id>1;
```

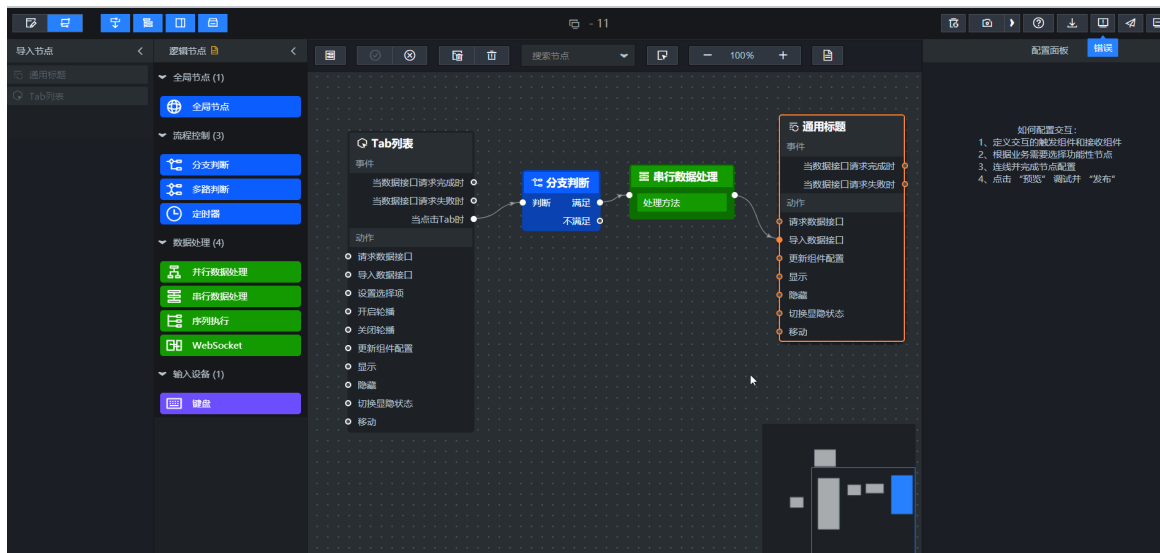
- 8. 以同样的方式，配置右侧的串行数据处理节点（此步骤作用是将列表的数据格式转换成标题的数据格式）。



上图中的示例转换格式的规则如下。

```
return [  
  {  
    value:data.content  
  }  
];
```

- 9. 单击右上角的预览图标 (  )，在预览页面进行交互操作（例如单击列表某一格等），查看交互效果。



如果需求中有多个组件需要配置交互链路，则重复以上几步直至满足全部交互需求。

- 10. 配置并预览成功后，单击左上角的发布图标 (  )。选择发布样式后，即可在线展示具有交互功能的可视化应用。

## 3. 蓝图编辑器详细功能介绍

本文为您介绍蓝图编辑器在使用过程中常用的功能详情，可以帮助您快速上手蓝图编辑器。

### 导出到蓝图编辑器

只有当组件导入到蓝图编辑器后，才可以为该组件配置交互。

在画布编辑器内，右键单击左侧图层栏或中间画布区的组件，选择**导出到蓝图编辑器**，即可将对应组件导出到蓝图编辑器中。




导出成功后，切换到**蓝图编辑器**页面，可在**导入节点列表**中查看对应的节点。列表内所有节点都可供后续配置交互使用。

### 取消导出到蓝图编辑器

对于已经导入到蓝图编辑器中的组件，如果不再需要对该组件配置交互，可将该组件取消导出到蓝图编辑器。

 **注意** 取消导出后，组件已经配置的相关交互都不可用。

在蓝图编辑器页面，单击左上角的画布编辑器图标（）。在画布编辑器页面，右键单击左侧图层栏或中间画布区的对应组件，选择取消导出到蓝图编辑器。




被取消导出的节点，在蓝图编辑器画布中置灰且有红色叹号。此时该节点是处于不可用状态，但是可以被重新导入并恢复之前配置。



## 在蓝图编辑器内定位

对于已经导入到蓝图编辑器中的组件，如果需要快速定位到该组件在蓝图编辑器中的位置，可使用蓝图编辑器定位功能实现。

在蓝图编辑器页面，单击左上角的画布编辑器图标（）。在画布编辑器页面，右键单击左侧图层栏或中间画布区的对应组件，选择在蓝图编辑器内定位。



定位成功后，系统会自动跳转到蓝图编辑器页面，定位并选中对应组件。

## 显示或隐藏配置栏面板

Datav蓝图编辑器的配置栏包括导入节点面板、逻辑节点面板、右侧面板以及工具栏，分别对应以下四个功能图标。单击对应图标，可控制各配置栏的显示或隐藏。



- **导入节点**：在导入节点面板中，可以查看从画布编辑器导入的所有组件节点。单击节点拖动到画布中，可添加节点。在画布中选中某个节点，右键单击该节点，选择**从画布中移除**，可移除该节点。

**注意** 移除节点时，会同时移除选中的节点和边，且无法恢复，请谨慎操作。

- **逻辑节点**：在逻辑节点面板中，单击节点拖动到画布中，添加逻辑节点，并在右侧配置面板中进行配置，详情请参见**逻辑节点配置**。在画布中选中某个节点，右键单击该节点，选择**删除**，可删除该节点。

**注意** 删除节点时，会同时删除选中的节点和边，且无法恢复，请谨慎操作。

- **右侧面板**：右侧面板主要展示蓝图编辑器内两部分配置面板的内容：默认的蓝图交互效果配置面板样式和逻辑节点的配置面板样式。

- 单击画布空白处，在右侧面板可设置蓝图配置效果的生效时机和超时时间功能。

参数	说明
生效时机	<p>设置蓝图配置交互后在预览时的生效时机，选择的时机不同可能影响组件默认数据与交互之间的先后关系。时机可选：</p> <ul style="list-style-type: none"> <li>■ 页面载入时：页面刚开始加载时。</li> <li>■ 页面初始化完成时：所有组件都渲染出来后。</li> <li>■ 数据加载完成时：所有组件的数据都返回后。</li> </ul>
超时时间	<p>配置完成后，蓝图生效时机可能因某些原因触发失败，如出现组件加载失败，组件渲染错误，接口数据返回失败等情况。如组件数据加载失败或组件渲染失败时，您可以自定义设置<b>超时时间</b>，强制使蓝图交互效果生效的时间，单位为秒。</p>

- 在画布中单击某个**逻辑节点**，在右侧面板可配置该节点的名称、上下游组件节点、以及处理方法等，详情请参见[逻辑节点配置](#)。
- 工具栏**：工具栏位于画布上方，包含适配画布、节点启用、禁用、清空画布、删除不可用节点、搜索、切换单选框选以及画布的缩放等快捷操作。



操作	说明
适配画布	单击工具栏中的 <b>适配画布</b> 图标（  ），将画布中的节点自适应到画布中间显示。
停用/启用	在画布中，右键单击某个节点，选择 <b>停用/启用</b> ，或者选中节点后，单击工具栏中的 <b>停用/启用</b> 图标（  ），暂时停用或启用交互链路的某个节点、某个图层或某个规则组。
清空画布	<p>单击工具栏中的<b>清空画布</b>图标（），删除当前画布中所有节点和连线，一键还原空白画布配置。</p> <p> <b>警告</b> 删除后，节点或连线都将无法恢复，请谨慎操作。</p>
删除不可用节点	<p>当您不再使用被取消导出到<b>蓝图编辑器</b>的组件节点时，可以单击工具栏中的<b>删除不可用节点</b>图标（），删除当前无用节点，并清除该节点的所有交互配置。</p> <p> <b>警告</b> 删除后，节点或连线都将无法恢复，请谨慎操作。</p>
搜索节点	单击工具栏中的 <b>搜索节点</b> 下拉列表，选择一个节点，可在画布中快速定位并居中显示该节点。

操作	说明
切换框选模式	<p>单击工具栏中的切换框选模式图标 ()，可切换画布选择模式为单选或框选。</p> <ul style="list-style-type: none"> <li>单选模式：鼠标单击一个节点或连线进行选中。Windows用户使用快捷键Ctrl (Mac用户使用Command) 外加鼠标单击，实现节点或连线的多选功能。</li> <li>框选模式：使用鼠标，在画布中框选一个或多个节点或连线，进行选中。</li> </ul>
缩放画布	<p>单击工具栏中的+ 或- 图标，或者Windows使用快捷键Ctrl (Mac用户使用Command) 同时鼠标滚轮，可放大或缩小画布。您也可以输入具体的百分比，实现精确缩放。</p>
新手引导	<p>单击工具栏中的新手引导图标 ()，系统弹出蓝图编辑器的引导弹框，帮助您快速了解蓝图编辑器的用法。</p>

## 画布操作

在蓝图编辑器的画布中，您可以完成节点连线和移动、复制节点、选中节点关联边、选中锚点关联边、在画布编辑器中选中等操作。

操作	说明
移动画布	<p>在单选模式下，单击画布空白处移动，即可移动画布。</p>
缩放画布	<p>Windows用户使用快捷键Ctrl (Mac用户使用Command) 外加鼠标滚轮，可缩放画布。</p>
移动节点	<p>选择某个节点拖动即可。</p>
连线和删除连线	<p>您可以使用连线功能触发两个节点之间的数据交互。单击源节点的锚点，拖出一条连线，连接到目标节点的锚点上。</p> <div style="background-color: #e6f2ff; padding: 5px; margin: 5px 0;"> <p> <b>说明</b> 源节点事件只能抛出，所以是起始锚点；目标节点动作只能被触发，所以是目标锚点。</p> </div> <p>右键单击某一条连线，选择删除可删除该连线。</p>
变更连线终点	<p>您可以单击选中连线的终点，拖动连线的终点到其他目标节点的锚点上。</p>
记录日志和取消记录日志	<p>右键单击某一条连线，选择记录日志/取消记录日志，可选择是否在调试预览界面内的蓝图日志&amp;报错面板中记录经过这条交互链路的所有日志。</p> <div style="background-color: #e6f2ff; padding: 5px; margin: 5px 0;"> <p> <b>说明</b> 支持多选连线记录日志或取消记录日志，在画布空白处可支持单击右键取消所有日志记录。</p> </div>
启用和停用	<p>右键单击画布中某个节点、连线或表单组，选择启用/停用，可暂时启用或停用对应的节点、连线或表单组。</p>
在画布编辑器中选中	<p>右键单击画布中某个导入节点，选择在画布编辑器中选中，即可返回画布编辑器页面并选中该组件。</p>





操作	说明
选中关联边	右键单击画布中某个逻辑节点，选择选中关联边，即可选中当前逻辑节点左右两侧关联的边线。
选中上下游链路	右键单击画布中某个导入节点，选择选中上下游链路，即可选中当前节点关联的上下游节点。
在画布中移除	<p>右键单击画布中某个导入节点，选择在画布中移除，可将该节点从画布中移除。</p> <p> <b>注意</b> 移除节点时，会同时移除选中的节点和边，且无法恢复，请谨慎操作。</p>
删除	<p>右键单击画布中某个逻辑节点或连线，选择删除，可将该节点或者连线从画布中删除。</p> <p> <b>注意</b> 删除节点时，会同时删除选中的节点和边，且无法恢复，请谨慎操作。</p>
复制	右键单击画布中某个逻辑节点，选择复制，可复制该节点。除上下游事件和动作外，复制的节点继承源节点的其他所有配置，包括节点名称、处理方法等。
编组	<p>右键单击画布中某个节点或表单组，选择编组，可生成一个编组框。</p> <ul style="list-style-type: none"> <li>您可以调整编组框的大小，并将您需要放在一个编组框的多个节点移动到框内，后续只需要拖动编辑框即可同时移动所有编组框内的节点。</li> <li>您可以双击编组框左上角的注释，自定义修改编组框的名称。</li> <li>您可以右键单击编组框最上栏，单击解除编组即可解除编组功能。</li> </ul> <p> <b>说明</b> 解除编组后不可恢复，请谨慎解除。</p>

## 4. 调试预览指南

本文介绍DataV可视化应用在调试预览界面下的详细功能。

DataV支持两种可视化应用预览的模式：**正常预览**和**调试预览**。

- **正常预览**：在编辑器右上方单击预览图标（），选择**正常预览**时，DataV不会记录预览可视化应用时的报错和日志。
- **调试预览**：在编辑器右上方单击预览图标（），选择**调试预览**时，DataV将记录在预览可视化应用时组件和蓝图内的交互日志和报错信息。

### 说明

- 调试预览功能仅支持PC端可视化应用，移动端可视化应用不支持**调试预览**。
- 进入**调试预览**界面后，默认开启日志和报错记录。


### 蓝图日志&报错功能介绍

**使用前提**：要使用蓝图日志&报错功能，首先需要在蓝图编辑器界面内，将需要被监听的连线选中，并右键单击记录日志。



在**调试预览**界面产生交互行为时，如果命中记录日志的交互，则记录这次连线交互的上下游节点关系和交互触发时产生的数据快照。

在进入**调试预览**界面后，单击左侧边缘的**开启调试**，选择**蓝图日志&报错**界面，在当前可进行如下操作：

- **开始记录/暂停记录**：单击记录  图标，切换到**开始记录**即可监听蓝图日志和报错或**暂停记录**模式即可关闭监听。

**说明** 为了避免日志刷新时影响可视化应用的性能，建议用户在预览调试可视化应用时关闭数据的自动更新，或增加数据自动更新时长。

- **清空日志**：单击清空日志图标，即可清除当前界面所有日志或报错内容。
- **日志记录**：每次新的交互后，将在原来的日志记录的最上方新增一条记录。单击打开某个日志记录，选择

日志记录中的连线，右侧会弹出数据快照框，框内将进一步说明报错时的数据、回调ID（如有）、页面临时变量（如有）以及报错堆栈等情况。



- 记录交互日志如遇到报错时，连线会被标记为红色。单击红色连线，在数据快照框内展示错误堆栈。



- 记录交互日志没有报错时，连线为蓝色。

**说明** 在连线左侧的逻辑节点中，若使用了getLocalValue和getCallbackValue，获取到了回调ID和临时变量，才会记录在localValue和callbackValue的数据快照中。

- **复制节点ID**：双击记录内的节点，可复制节点ID。可以将复制的ID在蓝图编辑器界面中进行搜索和定位。



### 组件日志&报错功能介绍

在进入调试预览界面后，单击左侧边缘的开启调试，选择组件日志&报错界面，在当前可进行如下操作：

- **开始记录/暂停记录**：单击记录 图标，切换到开始记录即可监听组件日志和报错或暂停记录模式即可关闭监听。

**说明** 为了避免日志刷新时影响可视化应用的性能，建议用户在预览调试可视化应用时关闭数据的自动更新，或增加数据自动更新时长。

- **清空日志**：单击清空日志图标，即可清除当前界面所有日志或报错内容。
- **日志记录**：当组件请求发生错误时，面板内会将原来的记录的最上方新增一条报错记录。单击选择一个报错记录，右侧会弹出**数据快照框**，框内将进一步说明报错时的数据、回调ID（如有）、页面临时变量（如有）以及报错堆栈情况。



组件日志记录的常见报错类型

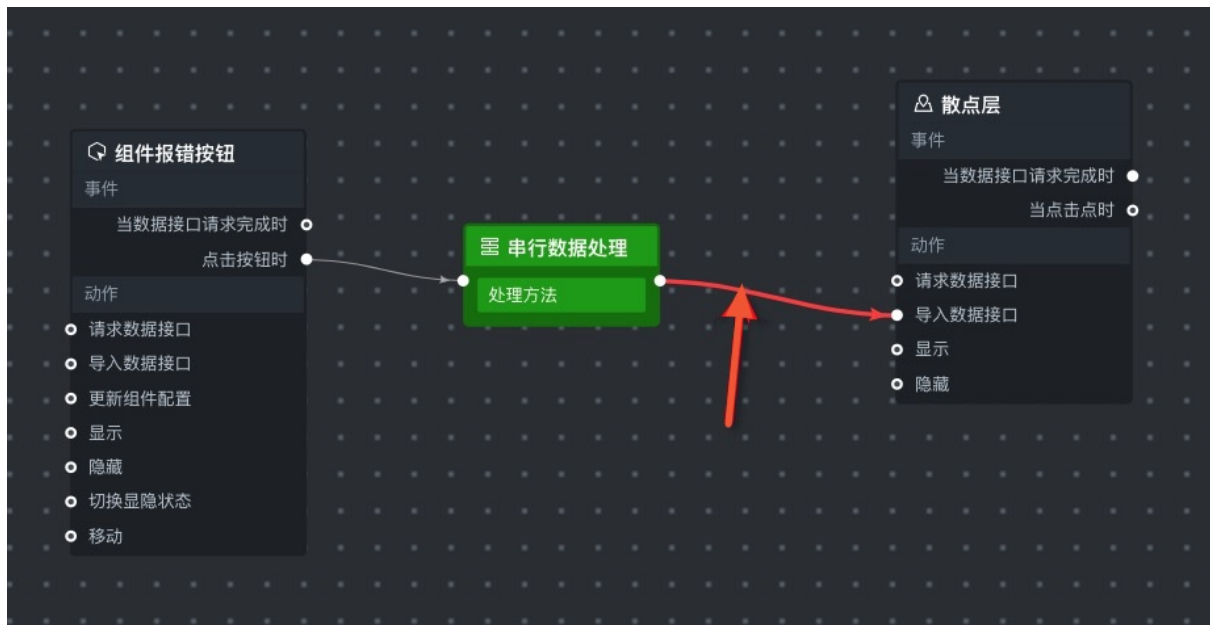
报错类型	类型说明
数据源报错	组件请求数据源发生错误时，产生该报错。数据源的请求可能发生在组件首次加载时、组件自动更新时、组件回调ID触发更新时、蓝图触发请求数据源动作时。展示报错的组件ID、数据源ID、错误描述。
过滤器报错	组件请求数据源成功返回数据后，经过过滤器时产生报错，将记录为过滤器报错。展示报错的组件ID、数据源ID、过滤器ID、错误描述。
组件内部报错	组件渲染或执行组件方法时产生的报错。选择组件内部报错，展示报错的组件ID、组件方法。
节点内部报错	蓝图节点内代码出现问题时，导致的节点初始化报错等。

**说明** 当调用组件方法时的参数不符合组件声明时，将产生警告信息。

- **复制组件ID**：双击报错条目内的组件名称，可复制ID到剪贴板。
- **搜索组件ID**：将复制到的组件名称，在编辑器中搜索框粘贴即可快速查询到该组件，并根据报错提示去修改组件的配置。
- **筛选组件报错日志**：在输入框内输入某个组件名称、组件ID、过滤器ID或报错信息等，即可快速筛选出当前报错面板下所有与输入信息相关联的报错条目。

### 调试窍门

以终为始，用最小粒度记录日志。如遇到当目标组件的交互效果未产生时，可从目标组件的动作开始，往回反查每一步的传入数据是否正确。



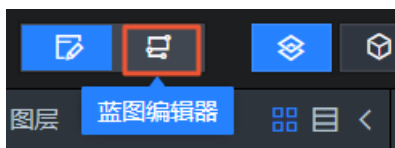
# 5.逻辑节点配置

## 5.1. 概述

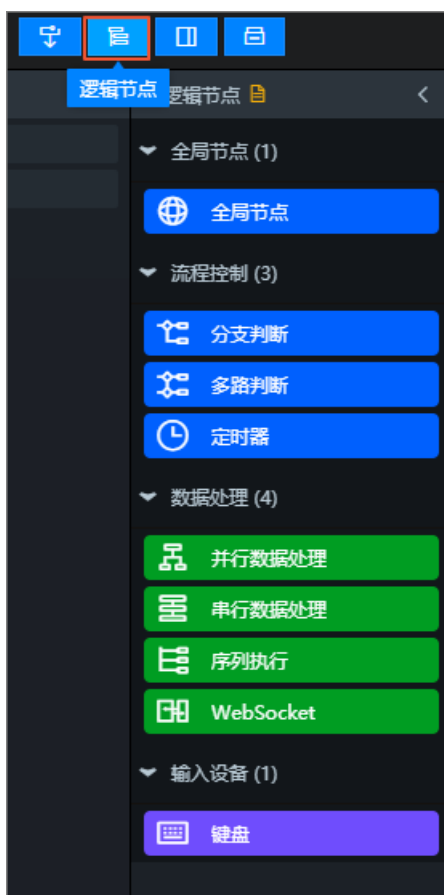
逻辑节点可以帮助您设置组件和组件之间的交互逻辑，实现大屏内各个组件的交互。本文介绍在蓝图编辑器中，配置逻辑节点的方法。

### 使用逻辑节点

1. 登录DataV控制台。
2. 在我的可视化页面中，选择一个可视化应用，单击编辑。
3. 在画布编辑器页面，单击顶部工具栏的蓝图编辑器图标。



4. 在蓝图编辑器页面中，单击顶部工具栏的逻辑节点图标。



**说明**

- 如果逻辑节点面板已经显示，可忽略此步骤。
- 在配置逻辑节点前，请确保您已经添加了对应的导入节点。如果还未添加，请参考[配置蓝图编辑器](#)进行添加。

5. 在逻辑节点面板中，将需要使用的逻辑节点拖入中间画布中。




逻辑节点包括了全局节点、流程控制类节点、数据处理类节点以及输入设备类节点。各节点的使用方法以及参数详情，请分别参见[全局节点](#)、[流程控制](#)、[数据处理](#)和[输入设备](#)。

### 配置逻辑节点

在画布中单击逻辑节点，即可在右侧的配置面板中配置节点交互逻辑（全局节点不需要配置）。各节点的共同配置如下。



参数	说明
节点名称	自定义逻辑节点的名称，不同的逻辑节点，图标也不同。重命名后可根据图标分辨节点类别和功能。
上游事件说明	选择上游组件和事件，并查看参数说明。
下游组件动作	选择下游组件和动作，并查看参数说明。

参数	说明
处理方法	<p>新增并配置处理方法，实现各节点的逻辑交互功能。支持的操作如下：</p> <ul style="list-style-type: none"> <li>• <b>新建</b>：单击+，新增一个处理方法。</li> <li>• <b>重命名</b>：单击重命名（）图标，修改处理方法名称。</li> <li>• <b>创建代码片段</b>：处理方法保存后，单击创建代码片段（）图标，保存当前处理方法到控制台代码片段管理处，实现代码片段的复用。</li> <li>• <b>导入代码片段</b>：在处理方法编辑框内，右键单击空白处，在导入代码片段列表中，选择已保存的代码片段进行导入。</li> </ul> <p> <b>注意</b> 输入设备类节点和WebSocket节点不需要配置处理方法。</p>

## 5.2. 全局节点

全局节点可以帮助您在蓝图编辑器中，设置组件的初始化状态，并使用页面的回调ID和临时变量等。本文介绍在蓝图编辑器中，配置全局节点的方法。

添加全局节点到画布中，可查看全局节点支持的事件和动作。添加方式请参见[使用逻辑节点](#)



类型	事件/动作	说明
事件	全部组件初始化完成	可视化应用上所有组件初始化完成后，抛出的事件，无参数。
	数据加载完成	可视化应用上所有组件数据接口加载完成后，抛出的事件，无参数。



类型	事件/动作	说明
动作	设置回调id	<p>与画布编辑器的交互面板联通，可设置回调ID。回调ID的变化，会引起监听该回调ID的组件重新请求数据。回调值仅支持字符串、数字、单层对象、单层数组，不支持嵌套。参数举例说明如下。</p> <pre>{   data: {     // 回调ID变量名。     name: "productName",     // 回调ID值。     value: "DataV"   } }</pre> <p>取值方式</p> <p>在编辑处理方法时，可通过 <code>getCallbackValue("productName")</code> 取值，详情请参见 <a href="#">配置蓝图编辑器</a>。</p>
	设置页面临时变量	<p>页面级的全局变量，支持复杂的嵌套数据，用作简单的数据存储。参数举例说明如下。</p> <pre>{   data: {     // 临时变量名     name: "product",     // 临时变量值     value: {       "productName": "DataV",       "companyName": "Alibaba"     }   } }</pre> <p>取值方式</p> <p>在编辑处理方法时，可通过 <code>getLocalValue("product")</code> 取值。</p> <p><b>说明</b> 如果一个事件的连线同时触发设置页面临时变量和其他动作，则优先执行设置页面临时变量动作。</p>
	更新全部组件	<p>刷新可视化应用，所有组件的所有数据接口重新请求数据，重新渲染，无参数。</p>

## 5.3. 流程控制

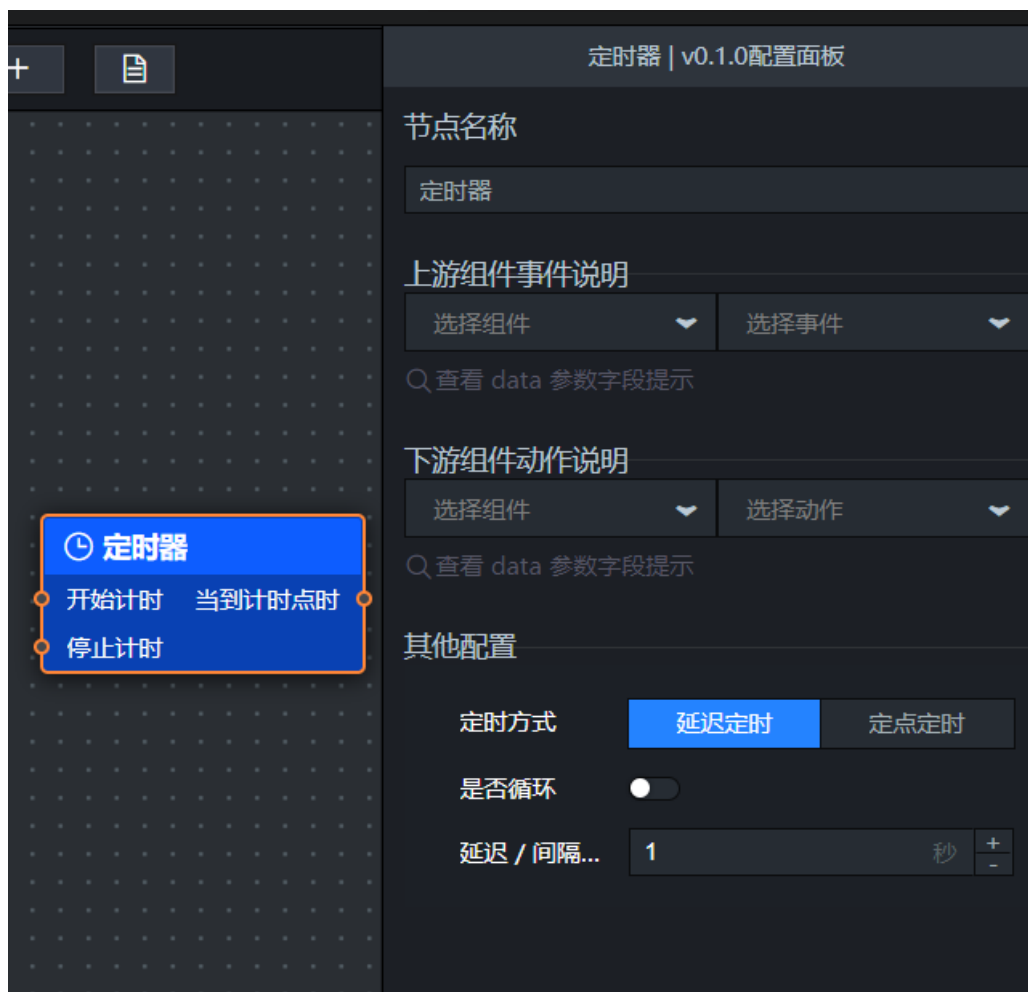
流程控制内的逻辑节点包括定时器、分支判断和多路判断。本文介绍在蓝图编辑器中，配置流程控制类节点的方法。

### 定时器

定时器节点支持延迟定时、定点定时、循环延迟定时和循环周期定时。

**使用场景：**定时器适用于需要定时的场景需求。当时间计数到达配置项设定的时间点时，定时器节点抛出当时计时结束时事件，输出上游节点的输出结果，触发后续动作。

添加定时器节点至画布中，可查看定时器节点支持的事件/动作，以及配置参数。添加方式请参见[使用逻辑节点](#)。



事件/动作参数说明

事件/动作	说明
开始计时	启动定时器。
停止计时	停止计时器，下次触发开始计时时，将重新计时。一般用在循环计时中。

事件/动作	说明
当到计时点时	到达计时点时抛出事件，触发下游节点执行动作。循环计时情况下，将循环抛出该事件。

### 配置项说明

表格中仅提供其他配置中的参数说明，其他参数配置请参见[公共参数说明](#)。

参数	说明
定时方式	支持延时定时和定点定时两种方式。
是否循环	是否需要重复计时。当定时方式为定点定时时，是否循环开启后，会配置定点周期，自定义设置定点周期内的起始时间和间隔时间。
延迟/间隔时间	以秒为单位进行倒计时。仅当定时方式为延时定时时有效。
定点时间	以秒为单位进行倒计时，到达具体的标准时间点，抛出事件。仅当定时方式为定点定时，且循环计时关闭时有效。

**输出结果：**不改变上游节点的输出结果。触发计时器时，输出上游节点的输出结果。

## 分支判断

分支判断节点属于If-Else判断条件节点。当满足设定条件时，抛出满足事件，不满足设定条件，抛出不满足事件。

**使用场景：**例如，根据开关状态触发两个图层的显隐效果场景。可以通过分支判断节点来判断当前开关的状态。处于打开状态，则显示图层A，隐藏图层B；处于关闭状态，则显示图层B，隐藏图层A。

添加分支判断节点至画布中，可查看分支判断节点支持的事件/动作，以及配置参数。添加方式请参见[使用逻辑节点](#)。



事件/动作参数说明

事件/动作	说明
判断	输入上游节点的输出结果，用于条件判断。
满足	上游节点输出的结果，满足设定条件。
不满足	上游节点输出的结果，不满足设定条件。

配置项说明

表格中仅提供其他配置中的参数说明，其他参数配置请参见[公共参数说明](#)。

参数	说明
判断类型	<ul style="list-style-type: none"> <li>同时满足所有条件：当上游节点的输出结果满足所有设定的条件时，抛出满足事件；只要有一个条件不满足，则抛出不满足事件。</li> <li>满足任一条件：当上游节点输出结果满足任一设定的条件时，抛出满足事件；全部不满足抛出不满足事件。</li> </ul>
处理方法	编写JavaScript函数体，返回结果为Boolean型。return true为满足该处理方法的条件，return false为不满足该处理方法的条件，可新增叠加。

输出结果：不改变上游节点的输出结果。触发分支判断时，输出满足条件的上游节点的输出结果。

## 多路判断

多路判断节点属于Case-When节点。通过对上游节点的输出结果进行判断，触发第一个满足条件的下游节点执行对应动作。

**使用场景：**例如，根据数字输入框内，当前的输入值设置地图散点的颜色。可通过多路判断节点，判断当前的输入值处于哪个范围，进而触发散点颜色的设置。例如在配置项中设置，当数值大于100时，为红色；50~100之间，为黄色；小于50，为蓝色。

添加多路判断节点至画布中，可查看多路判断节点支持的事件/动作，以及配置参数。添加方式请参见[使用逻辑节点](#)。



#### 事件/动作参数说明

事件/动作	说明
判断	输入上游节点的输出结果，用于多路判断。
case-N	处理方法。满足当前处理方法，则抛出该事件。可在配置面板中添加多个处理方法，添加后，多路判断节点中显示您添加的处理方法；不同处理方法可连接不同的下游节点，实现多路判断。
满足默认条件	满足默认条件，则抛出该事件。

#### 配置项说明

表格中仅提供其他配置中的参数说明，其他参数配置请参见[公共参数说明](#)。

参数	说明
处理方法	编写JavaScript函数体，返回结果为Boolean型。return true为满足该处理方法的条件，return false为不满足该处理方法的条件。满足后即抛出对应事件，且不再执行后续处理方法。

**输出结果：**不改变上游节点的输出结果。触发多路判断时，输出满足条件的上游节点的输出结果。

## 5.4. 数据处理

数据处理类节点包括并行数据处理节点、串行数据处理节点、序列执行节点和WebSocket节点。本文介绍在蓝图编辑器中，配置数据处理类节点的方法。

### 并行数据处理节点

并行数据处理节点，是使用并行方式来处理多个事件，各事件之间互不影响。

使用场景：并行数据处理节点在数据分发场景中使用较多。例如一个组件接口返回的数据为 `{name: '蓝图编辑器', version: 'v1.0'}`，分发到2个通用标题组件中，一个取的是name字段，另一个取的是version字段。可以使用该节点新增两个处理方法，一个为 `return [{value: data.name}]`，另一个为 `return [{value: data.version}]`。

添加并行数据处理节点至画布中，可查看并行数据处理节点支持的事件或动作，以及配置参数。添加方式请参见[使用逻辑节点](#)。



#### 事件或动作参数说明

事件或动作	说明
处理方法	并行的数据处理方法。可在配置面板中添加多个处理方法，添加后，并行数据处理节点中显示您添加的处理方法；不同处理方法可连接不同的上游节点，实现数据并行处理。

#### 配置项说明

参数	说明
处理方法	编写JavaScript函数体，返回结果可为任意类型。

表格中仅提供其他配置中的参数说明，其他参数配置请参见[公共参数说明](#)。

输出结果：每个处理方法对相应上游节点的输出结果进行计算后，得到的输出结果。

## 串行数据处理节点

串行数据处理节点，是使用串行方式来处理一个事件。

使用场景：例如，小数0.835要转换成整数百分比83%，可经过：单位转换（83.5）->取整（83）->添加字符串后缀（83%），一系列串行操作完成。

添加串行数据处理节点至画布中，可查看串行数据处理节点支持的事件或动作，以及配置参数。添加方式请参见[使用逻辑节点](#)。



### 事件或动作参数说明

事件或动作	说明
处理方法	串行的数据处理方法。可在配置面板中添加多个处理方法，添加后， <b>串行数据处理</b> 节点中显示您添加的处理方法；各方法共同实现数据处理。

### 配置项说明

参数	说明
处理方法	编写JavaScript函数体，返回结果可为任意类型。

表格中仅提供其他配置中的参数说明，其他参数配置请参见[公共参数说明](#)。

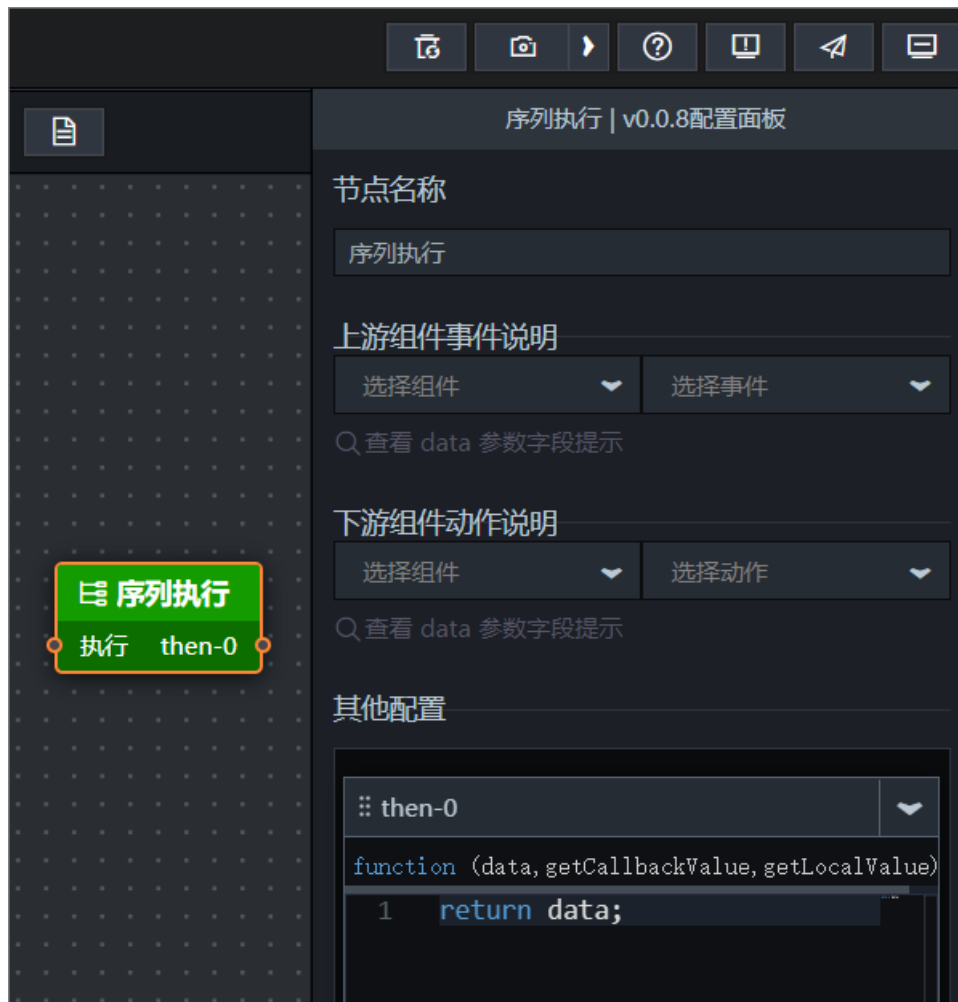
输出结果：上游节点的输出结果经过所有处理方法计算后，得到的输出结果。

## 序列执行节点

当您对下游节点的动作执行有顺序要求时，可使用**序列执行节点**，保证动作从上到下依次执行。

使用场景：例如，当您需要切换不同场景的数据面板时，每个场景有很多数据面板，先把场景A相关的数据面板切出，再把场景B相关的数据面板切进。可使用**序列执行**节点，对不同场景的数据面板进行分组，再按照先A后B的顺序执行。

添加**序列执行**节点至画布中，可查看**序列执行**节点支持的事件或动作，以及配置参数。添加方式请参见[使用逻辑节点](#)。



### 事件或动作参数说明

事件或动作	说明
执行	输入上游节点的输出结果，用于该节点的结果计算。
数据处理	序列执行节点的数据处理方法。可在配置面板中添加多个处理方法，添加后，序列执行节点中显示您添加的处理方法；各方法按照顺序进行数据处理。

### 配置项说明

参数	说明
处理方法	编写JavaScript函数体，返回结果支持任意类型。可叠加，每个处理方法独立计算，输入均为上一个节点的输出结果，输出为每个处理方法自己的计算结果，相互不影响。



表格中仅提供其他配置中的参数说明，其他参数配置请参见[公共参数说明](#)。

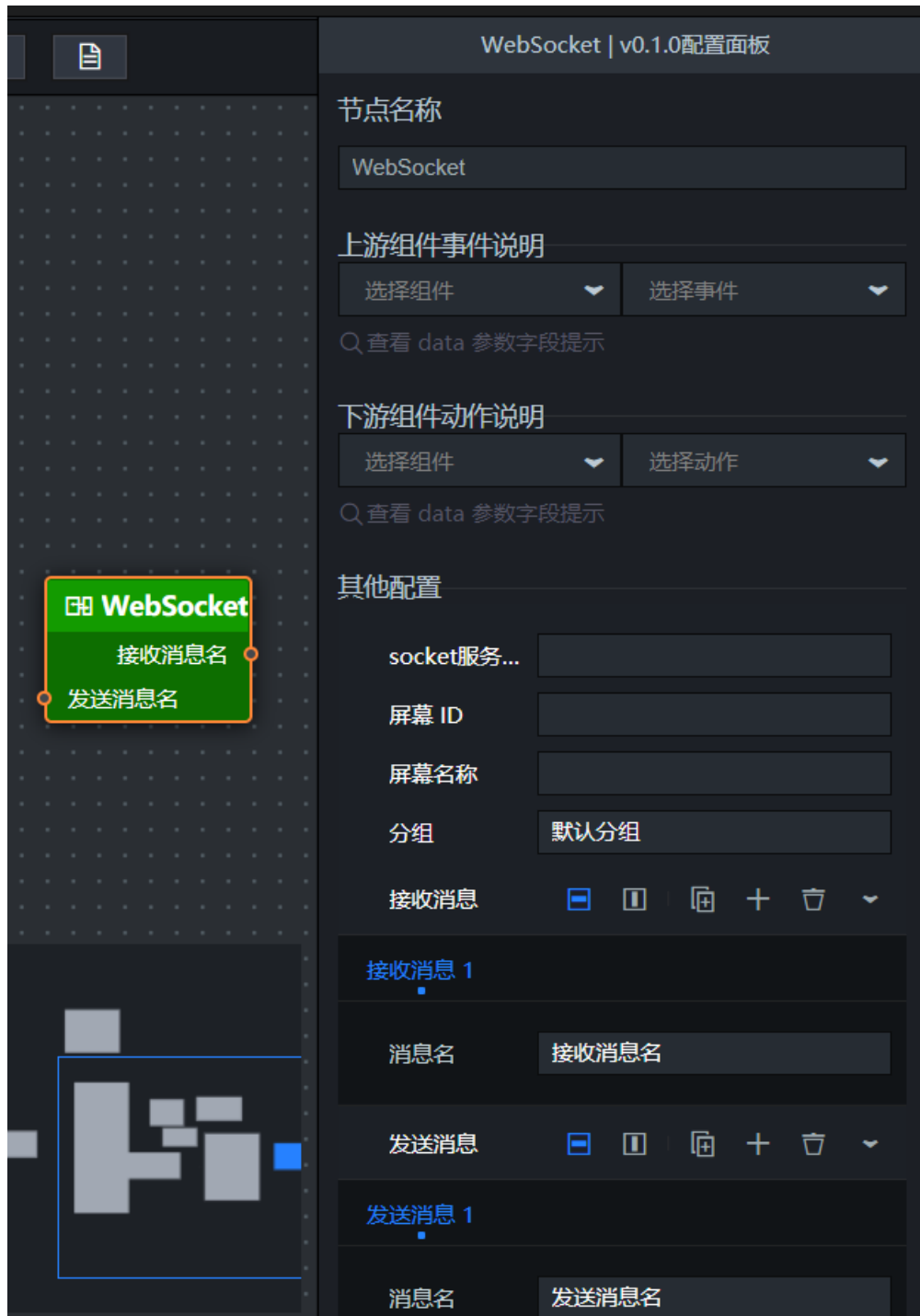
输出结果：每个处理方法按照顺序，对上游节点的输出结果进行计算后，得到的输出结果。

## WebSocket节点

**WebSocket**节点为屏间通信节点。每个消息由消息名称和数据组成，消息名称在配置项中自定义，数据为上一个节点的输出结果。

使用场景：**WebSocket**节点用于多端之间的命令和数据传输。例如大屏与移动端、大屏与触摸屏端的数据传输等。

添加**WebSocket**节点至画布中，可查看**WebSocket**节点的配置参数。添加方式请参见[使用逻辑节点](#)。



配置项说明

参数	说明
socket 服务地址	socket后端服务的地址。
屏幕ID	该WebSocket 节点所在屏幕的ID号，自定义输入。
屏幕名称	该WebSocket 节点所在屏幕的名称，自定义输入。

参数	说明
分组	<b>WebSocket</b> 节点消息只在同socket服务下的同分组中进行广播。一般同一项目约定一个分组名称。
接收消息	该 <b>WebSocket</b> 节点接收来自其他端的消息名称。  单击右侧的  或  图标，添加或删除一个接受消息。单击  或  图标配置多个接收消息的排列样式。单击  图标，即可复制当前选中接收消息配置内容并新增一个同样配置的接收消息。
发送消息	该 <b>WebSocket</b> 节点发送到其他端的消息名称。  单击右侧的  或  图标，添加或删除一个发送消息。单击  或  图标配置多个发送消息的排列样式。单击  图标，即可复制当前选中发送消息配置内容并新增一个同样配置的发送消息。

表格中仅提供其他配置中的参数说明，其他参数配置请参见[公共参数说明](#)。

输出结果：无。

## 自建WebSocket节点服务说明

如果您已经创建了WebSocket节点服务，那么您无需编写任何前端代码，即可将服务接入DataV提供的WebSocket节点中。前提是您的Websocket服务符合以下格式。

配置示例



- 注册消息

WebSocket节点发送消息格式如下。

```
{
  event: "register",
  data: {
    sid: 407194, //屏幕的ID
    name: "数据源受控模式测试", //屏幕的名称
    group: "默认分组" //默认分组
  },
  callback: "callback_15832235175585251131307383912" //当前注册时间戳，自动生成
}
```

WebSocket节点服务获取注册消息之后，需返回如下消息才能成功注册。

```
{
  event: "callback_15832235175585251131307383912", //返回与之前同样的注册时间戳
  data: {
    isError: false, //置为false
    data: "ok"
  }
}
```

- 发送消息

WebSocket节点发送消息格式如下。

```
{
  event: 'broadcast',
  data: {
    event: "发送出去的消息1",          //发送消息名称
    data: {}                          //data为任意格式
  }
}
```

- 接收消息

WebSocket节点接收消息格式如下。

```
{
  event: "broadcast_接收到的消息1",    //broadcast_${接收消息名称}
  data: {}                             //data为任意格式
}
```

## 5.5. 输入设备

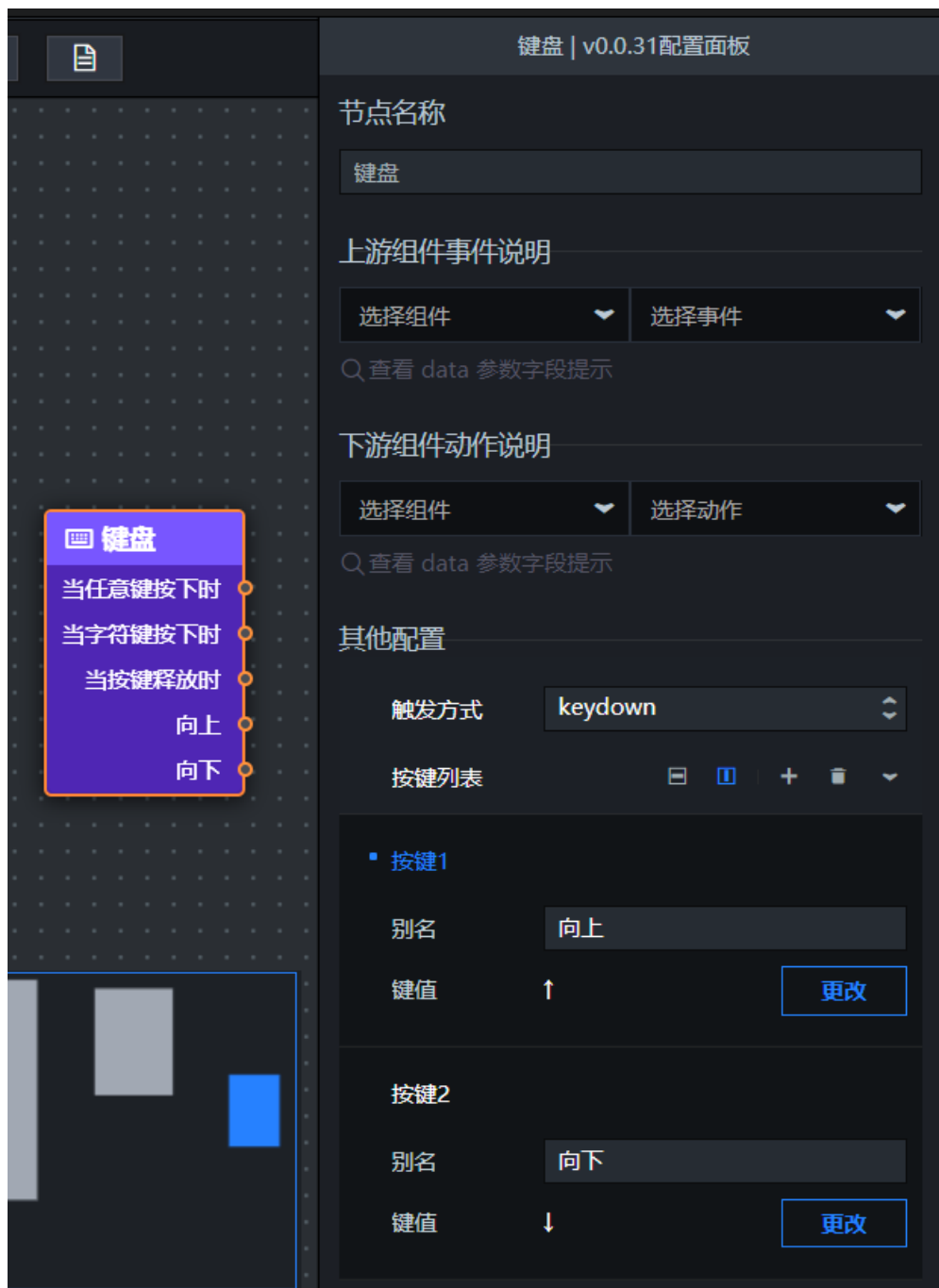
本文介绍在蓝图编辑器中，配置输入设备类节点的方法。目前，输入设备内的逻辑节点只包括键盘。

### 键盘

**键盘**节点用于监听键盘设备。

**使用场景：**键盘节点用于需要键盘参与交互的场景。例如，使用上下左右方向键控制一张图片的移动。

添加**键盘**节点至画布中，可查看**键盘**节点支持的事件和动作，以及配置参数。添加方式请参见[使用逻辑节点](#)。



事件/动作参数说明

事件/动作	说明
当任意键按下时	按下键盘任意键时，执行对应动作。
当字符键按下时	按下键盘的字符键时，执行对应动作。
当按键释放时	释放按键时，执行对应动作。
向上和向下	系统默认添加的按键名称，可自定义增加或删除按键，详情请参见 <a href="#">配置项说明</a> 。

## 配置项说明

表格中仅提供其他配置中的参数说明，其他参数配置请参见[公共参数说明](#)。

参数	说明
触发方式	<p>单击下拉框，选择一种键盘的触发方式，可选：</p> <ul style="list-style-type: none"> <li>• <b>keydown</b>：当任意键按下时触发事件。</li> <li>• <b>keypress</b>：当字符键按下时触发事件。</li> <li>• <b>keyup</b>：当任意键释放时触发事件。</li> </ul>
按键列表	<p>单击右侧的或图标，即可自定义添加或删除一个新按键，添加后的新按键会显示在按键列表节点中。单击或图标配置多个按键的排列样式。单击图标，即可复制当前选中按键配置内容并新增一个同样配置的按键。</p>
别名	自定义输入按键别名，用于按键的命名。例如，向上。
键值	单击右侧的更改，自定义修改按键的触发键。修改后，当在键盘上按下对应按键时，会触发相应的动作。例如， <b>Shift+ ↑</b> 。

输出结果：无。

## 6.配置表单组

如果您是DataV专业版及以上用户，可以将多个高级交互组件进行编组，形成一个表单组。通过表单组，您可以在蓝图编辑器中实现交互筛选功能。本文为您介绍表单组的配置方法。

### 建立表单组

1. 使用空白模板创建可视化应用，并添加几个高级交互组件。
2. 在图层栏或画布中，单击选中多个高级交互组件，或者拖拉鼠标，将想要归为一类的组件选中。



3. 单击鼠标右键，选择成表单组，将这几个组件添加到一个表单组中。

表单组建立之后，可双击组标题重命名。

表单组后与普通成组功能的区别为：表单组建立后会额外增加清空按钮和提交按钮，并且表单组没有限制，可以多次生成。



表单组建立之后，表单组外围及内部新生成的清空按钮和提交按钮会自动导出到蓝图编辑器。

**说明** 表单组本质上是个快捷操作，主要在蓝图编辑器配置中使用。在创建表单组的同时，DataV会自动绑定提交按钮和清空按钮与表单组之间的连线与交互，只需配置表单提交事件和表单清空事件的交互即可。

### 取消表单组

选择需要取消的表单组，单击鼠标右键，选择取消成组即可。





注意

- 取消表单组后，该组里面的组件将全部回归到未成组状态。各组件相互独立，不再以组合为单位进行拖拉移动等操作。
- 取消表单组后，清空按钮和提交按钮不会消失，只能手动删除。

### 清空按钮交互配置

表单组建立后，在蓝图编辑器页面的画布中，可以看到如下图所示的清空按钮的交互配置参数。



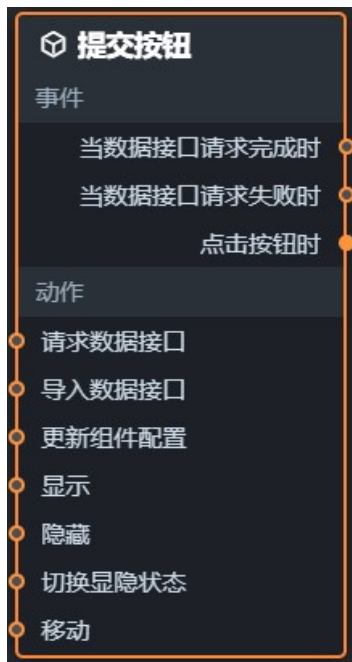
类型	事件/动作	说明
事件	当数据接口请求完成时	数据接口请求返回，抛出经过过滤器处理后的事件，同时抛出处理后的JSON格式的数据。
	当数据接口请求失败时	数据接口请求失败时（请求失败的情况可能是：网络问题或接口报错等）返回并经过过滤器处理后抛出的事件，同时抛出处理后的JSON格式的数据。

类型	事件/动作	说明
	点击按钮时	当单击按钮时抛出的事件，同时抛出该按钮的内部数据。
动作	请求数据接口	重新请求服务端数据，上游转换器或图层节点抛出的数据将作为参数。例如，清空按钮配置了API数据源为 <code>http://api.test</code> ，传到请求接口描述动作的数据为 <code>{id:'1'}</code> ，则最终请求接口为 <code>http://api.test?id=1</code> 。
	导入数据接口	按组件绘制格式处理数据后，导入组件，重新绘制。不需要重新请求服务端数据。
	更新组件配置	动态更新组件的样式配置。需要首先在组件的配置面板中，单击复制配置到剪贴板，获取组件配置数据。再根据需要，在蓝图编辑器配置页面的数据处理节点中，更改对应样式的字段值。
	显示	显示组件，不需要参数。
	隐藏	隐藏组件，不需要参数。
	切换显隐状态	<p>切换组件显示或者隐藏。</p> <p>参数示例：</p> <pre> {   // 显示为true，隐藏为false。   "status": true,   // 显示动画。   "animationIn": {     // 动画方式，可选fade，不填无动画。     "animationType": "fade",     // 显示延时，单位为ms。     "animationDuration": 1000,     // 显示动画函数，可选     linear easeInOutQuad easeInOutExpo。     "animationEasing": "linear"   },   // 隐藏动画。   "animationOut": {     // 动画方式，可选fade，不填无动画。     "animationType": "fade",     // 隐藏延时，单位为ms。     "animationDuration": 1000,     // 隐藏动画函数，可选     linear easeInOutQuad easeInOutExpo。     "animationEasing": "linear"   } }                     </pre>

类型	事件/动作	说明
	移动	<p>将组件移动到指定位置。</p> <p>参数示例：</p> <pre> {   // 移动方式。绝对定位：by，相对定位：to。   "positionType": "by",   // 指定位置。x坐标，y坐标。   "attr": {     "x": 0,     "y": 0   },   // 动画方式。   "animation": {     "enable": false,     // 动画延时。     "animationDuration": 1000,     // 动画曲线。可选值为：     linear easeInOutQuad easeInOutExpo。     "animationEasing": "linear"   } }                     </pre>


### 提交按钮交互配置

提交按钮的交互配置参数如下，与清空按钮类似，详情请参见[清空按钮交互配置](#)。



### 表单组应用操作示例

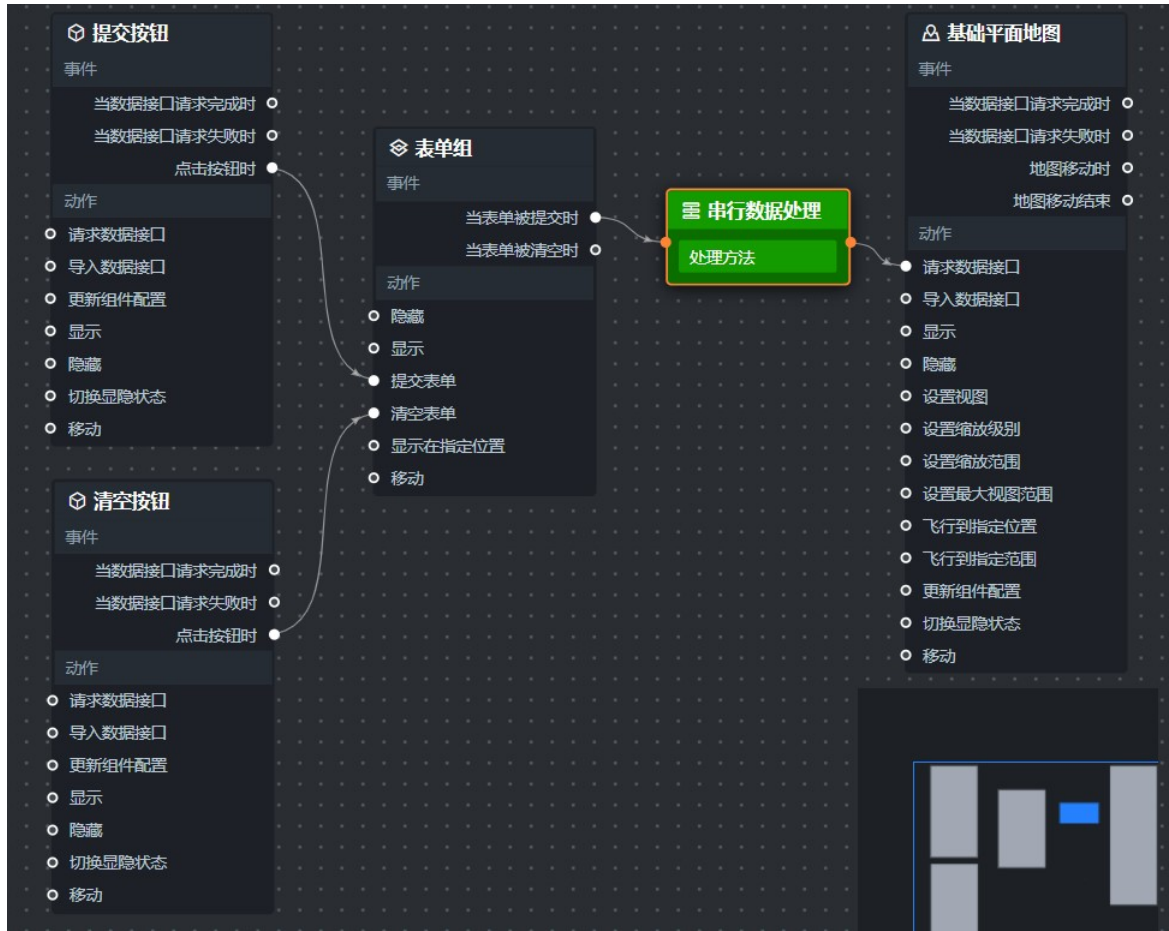
以下示例以高级交互组件中的单选框、输入框，以及地图组件中的基础平面地图为例，为您演示表单组在蓝图编辑器的具体使用方法。

1. 在画布编辑器页面，搭建所需要的单选框、输入框和基础平面地图组件。
2. 同时选中单选框和输入框组件，单击鼠标右键，选择成表单组。
3. 同时选中单选框、输入框和基础平面地图，单击鼠标右键，选择导出到蓝图编辑器。
4. 确认导出后，单击顶部左侧菜单栏的蓝图编辑器图标（）。
5. 在蓝图编辑器页面，查看自动生成的表单组交互配置。

Dat aV会自动生成与编辑器页面中的表单组对应的交互配置，表单组两边清空按钮和提交按钮，分别绑定了清空表单和提交表单的动作。当单击了对应的按钮时，会触发对应的表单响应事件。



6. 在左侧的导入节点面板中，将基础平面地图组件拖到画布中。
7. 将表单组事件中的当表单被提交时事件，与地图上的请求数据接口或者导入数据接口动作连线，即可触发地图上被筛选后的内容更新。



8. 添加其他连线 and 数据处理节点，并进行节点配置。
9. 配置完成后预览并发布，即可在可视化应用项目中实现表单组的交互效果。  
详情请参见[操作示例](#)。

## 7. 常见问题

### 7.1. 如何使用Tab列表控制组件显隐

本文档以Tab列表控制区域图和基本柱状图的显隐为例，为您介绍使用Dat aV蓝图编辑器功能控制组件显隐的方法。

#### 前提条件

准备好交互需求。

本案例的交互需求为：当您单击Tab列表中不同的页签时，会切换为不同的组件显示对应的数据。

#### 背景信息


本文档为您提供以下两种解决方案。

- 通过分支判断节点设置触发条件控制组件的显示和隐藏，详情请参见下文的[通过分支判断节点控制](#)。
- 通过串行数据处理节点结合组内轮播功能控制组件的显示和隐藏，详情请参见下文的[通过串行数据处理节点控制](#)。

#### 理解案例交互

当前案例实现需要您将交互转成节点和连线的关联关系，需要使用的组件有满足需求的事件和动作才能进行连线配置，否则就无法实现。


- **组件**：组件包含两种能力，事件抛出能力和动作执行能力。本案例使用了翻牌器组件之间的联动。
  - **Tab列表**：具备抛出请求接口返回数据的能力。
  - **区域图&基本柱状图**：属于被联动组件，具有接收数据并执行动作的能力。
- **蓝图编辑器**：通过运用逻辑节点编排的方式，将上游组件的交互/事件与下游组件的执行动作绑定，从而实现页面内组件间的交互联动。

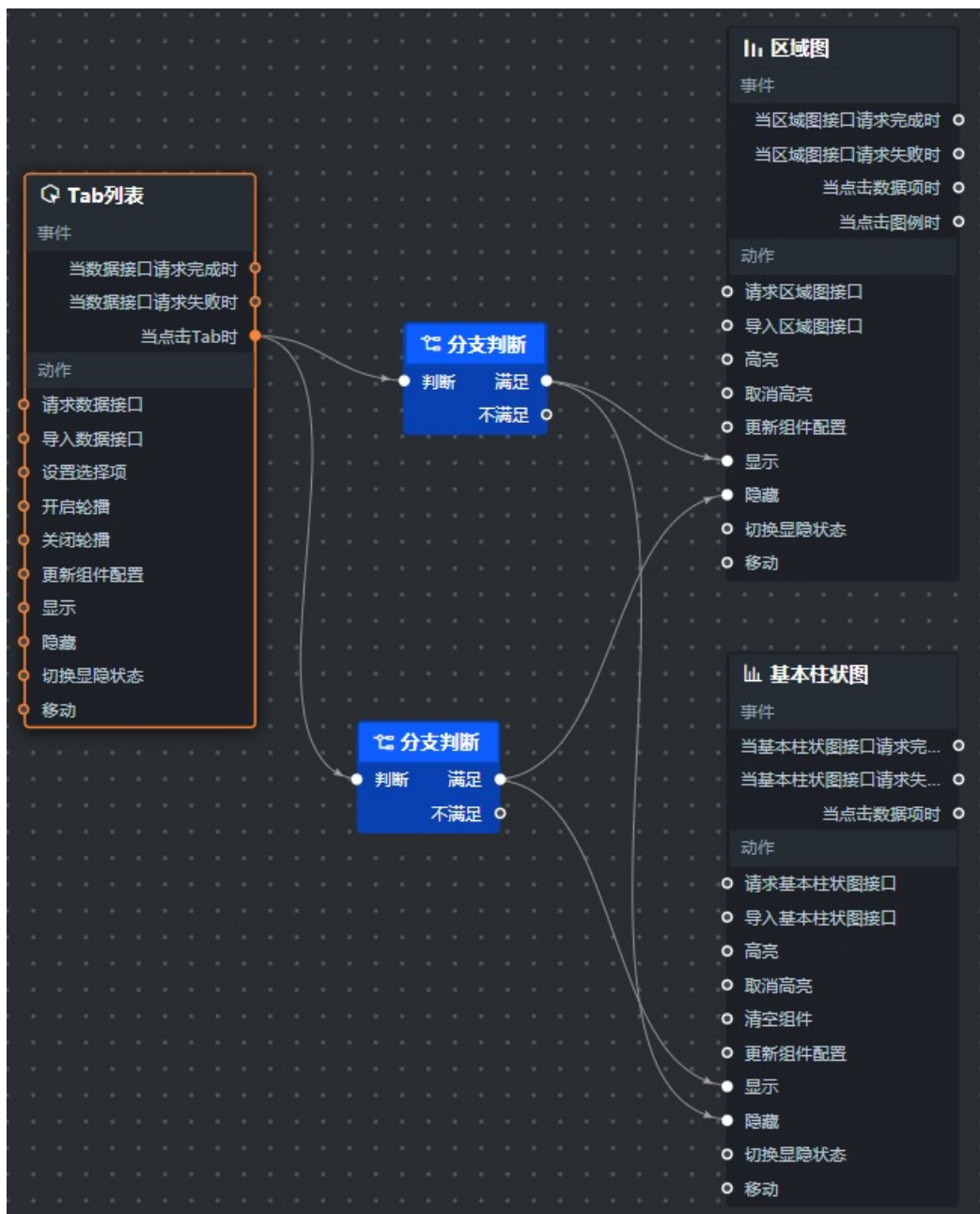
 **说明** 蓝图编辑器配置完成后需要在预览页查看配置效果。

#### 通过分支判断节点控制

1. 参考画布编辑器[添加资产](#)，在画布编辑器页面搭建所需要的Tab列表、区域图和基本柱状图组件。
2. 单击画布编辑器图层栏内三个组件，右键单击选择导出到蓝图编辑器。



3. 在画布编辑器页面左上角，单击蓝图编辑器图标（），切换到蓝图编辑器配置页面。
4. 在蓝图编辑器配置页面，将左侧导入节点栏内的Tab列表、区域图和基本柱状图节点拖至画布中。从逻辑节点栏内拖动添加两个分支判断节点到画布中，并按照如下图所示的样式进行连线。



5. 配置画布中两个分支判断内的处理方法，将处理方法分别重命名为判断是否为tab1和判断是否为tab2。

- i. 单击与区域图关联的分支判断逻辑节点，进入右侧配置面板，选择面板内其他配置栏下方的判断是否为tab1处理方法，单击右侧箭头打开脚本编辑区域。
- ii. 在脚本编辑区域，输入代码，完成后单击完成。

当前处理方法的示例代码如下。

```
return data.id == 1;
```



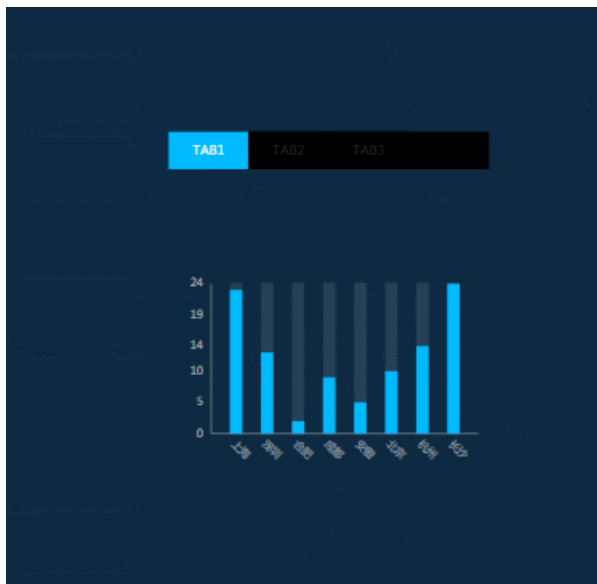
iii. 使用同样的方式配置判断是否为tab2处理方法。

当前处理方法的示例代码如下。

```
return data.id == 2;
```

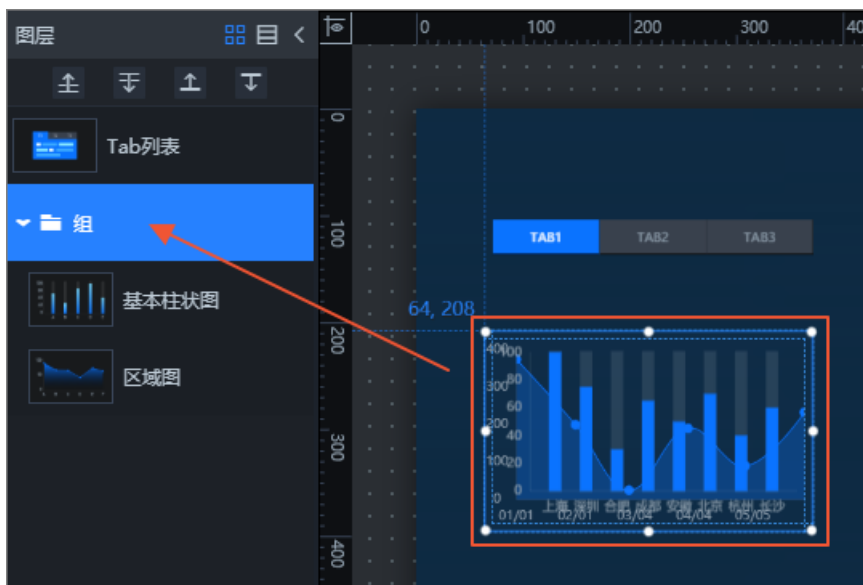
② 说明 该触发判断条件下，实现的交互是如果Tab列表的参数 id=1 的情况下，参数传递给两个分支判断节点后会使得与区域图组件关联的处理方法为true（满足）状态，与基本柱状图组件关联的处理方法为false（不满足）状态，配置为true的时候才会触发区域图执行显示和基本柱状图执行隐藏的相应动作。

6. 处理方法都配置完成后，单击预览即可查看Tab组件控制区域图和基本柱状图组件显隐的交互效果。



### 通过串行数据处理节点控制

1. 参考添加资产，在可视化画布编辑器页面搭建所需要的Tab列表、区域图和基本柱状图组件。
2. 将区域图和基本柱状图重叠放置，并全部选中，右键单击成组。




3. 单击成组的对象，在右侧的配置栏中，打开组内对象轮播开关，并将基础设置中的触发方式设置为事

件触发。

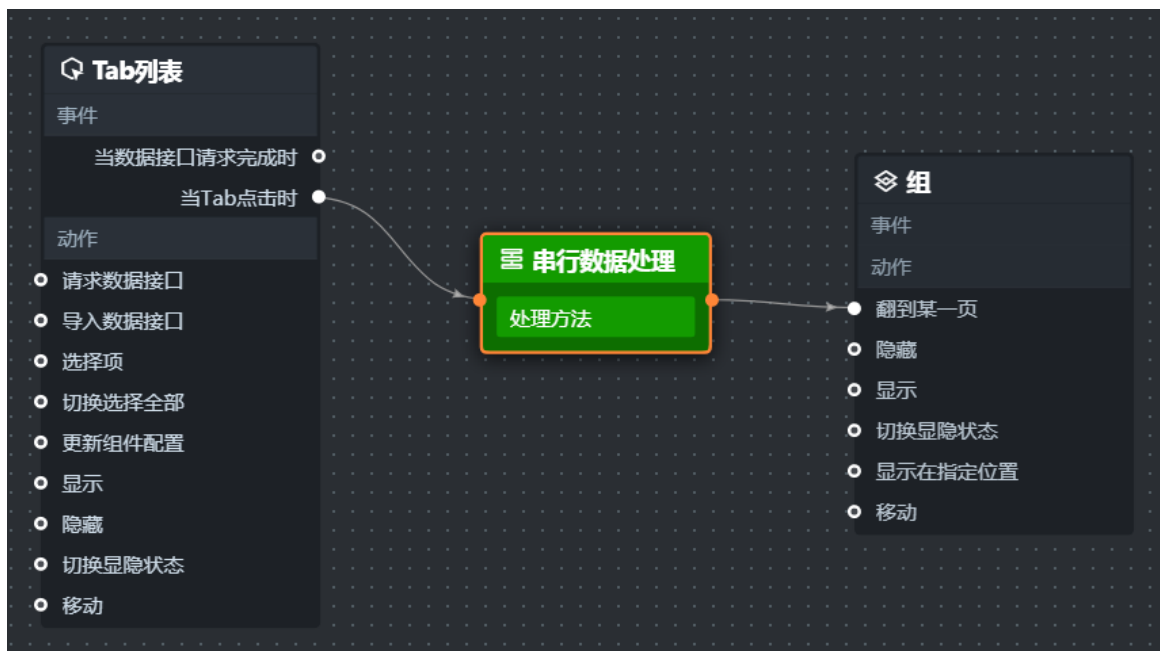


4. 单击画布编辑器图层栏内Tab列表组件和组，右键选择导出到蓝图编辑器。



5. 单击画布编辑器页面左上角，单击**蓝图编辑器**图标 (  )，切换到蓝图编辑器配置页面。

6. 在蓝图编辑器配置页面，将左侧导入节点栏内的**Tab列表**和**组**节点拖至画布上，并按照如下图所示的样式连线。



7. 连线中间会自动添加一个串行数据处理节点。

8. 配置画布中串行数据处理节点内的处理方法。

- i. 单击与串行数据处理逻辑节点，进入右侧配置面板，选择面板内其他配置栏下方的处理方法，单击右侧箭头打开脚本编辑区域。
- ii. 在脚本编辑区域，输入代码，完成后单击完成。

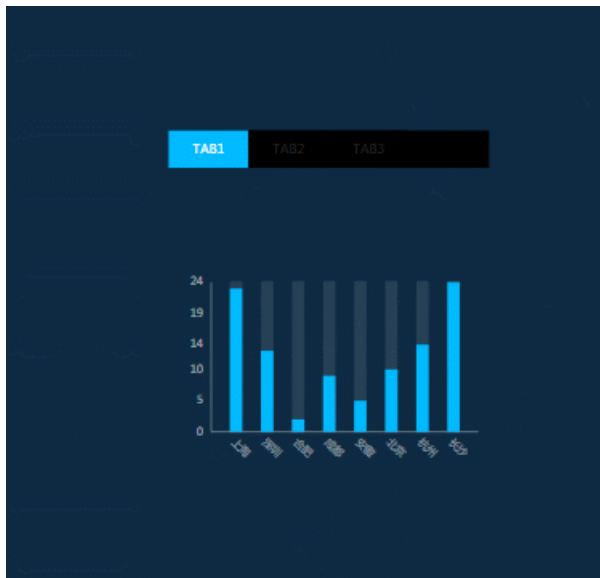
当前处理方法的示例代码如下。

```
return {
  index: data.id - 1
};
```



**说明** 该触发判断条件下，实现的交互是当Tab列表组件当Tab点击时事件内的{id:"1"}参数传递给串行数据处理节点后变成(index:0)并传递给组作为query参数并立即执行相应动作。组内的组件索引是以0开始的，所以要在串行数据处理时进行相应加工。

9. 处理方法都配置完成后，单击预览即可查看Tab组件控制组的组内翻页，实现组件显隐的交互效果。



## 7.2. 如何配置数据筛选

本文档以基本柱状图控制轮播列表组件数据更新为例，为您介绍蓝图编辑器配置的数据筛选功能。

### 前提条件

准备好您的交互需求。

本案例的交互需求为：当用户单击基本柱状图中某一个柱子时，轮播列表筛选数据后展示当前柱子对应的数据。

### 理解案例交互

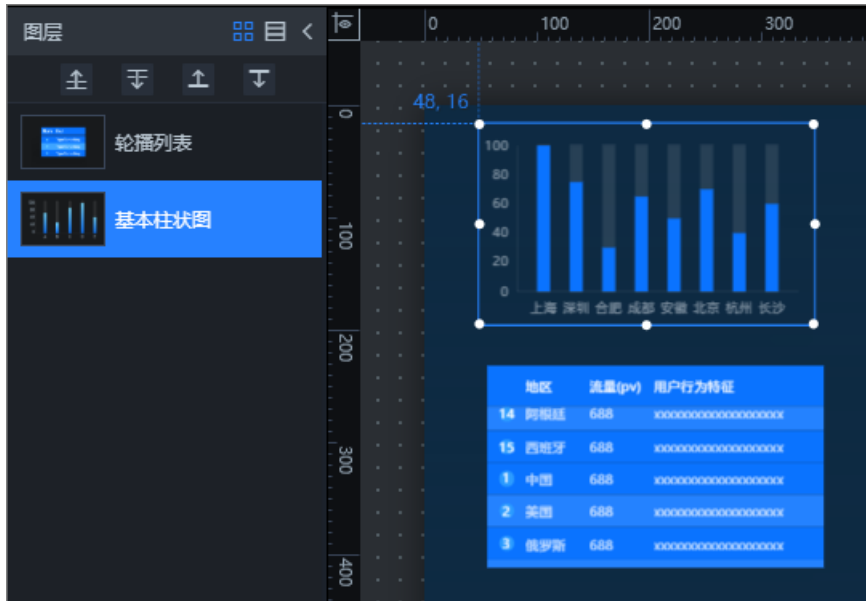
当前案例实现需要用户将交互转成节点和边的关联关系，需要使用的组件有满足需求的事件和动作才能进行配置，否则就无法实现。

- **组件**：组件包含两种能力，事件抛出能力和动作执行能力。本案例使用了基本柱状图和轮播列表组件之间的联动。
  - **基本柱状图**：具备抛出点击事件的能力。
  - **轮播列表**：属于被联动组件，具有接收数据并执行动作的能力。
- **蓝图编辑器**：通过运用逻辑节点编排的方式，将上游组件的交互/事件与下游组件的执行动作绑定，从而实现页面内组件间的交互联动。

**说明** 蓝图编辑器配置完成后需要在预览页查看配置效果。


### 配置案例交互

1. 参考画布编辑器**添加资产**，在画布编辑器页面搭建所需要的基本柱状图和轮播列表组件。



- 单击画布编辑器图层栏内基本柱状图和轮播列表组件，右键选择导出到蓝图编辑器。



- 在画布编辑器页面左上角，单击蓝图编辑器图标（），切换到蓝图编辑器配置页面。
- 在蓝图编辑器配置页面，将左侧逻辑节点栏内的全局节点和串行数据处理节点添加到画布中。
- 将串行数据处理配置面板中的处理方法重命名为设置轮播全量数据。  
具体操作方法请参见[串行数据处理节点](#)。
- 将全局节点的数据加载完成事件与设置轮播全量数据处理方法左侧的圆点相连，再将设置轮播全量

数据处理方法右侧圆点与全局节点的设置临时变量动作连线，最终连线样式如下图。



7. 配置画布中串行数据处理内设置轮播全量数据的处理方法。

- i. 单击串行数据处理逻辑节点，进入右侧配置面板，选择面板内其他配置栏下方的设置轮播全量数据处理方法，单击右侧箭头打开脚本编辑区域。
- ii. 在脚本编辑区域，输入代码，完成后单击完成。

处理方法内需要设置临时变量（示例中为carouselData，可自定义），当前输入代码如下。

```
return {
  data: [{
    name: "carouselData",
    value: [
      {
        "name": "上海项目1",
        "value": 111,
        "type": "上海"
      },
      {
        "name": "上海项目2",
        "value": 222,
        "type": "上海"
      },
      {
        "name": "深圳项目1",
        "value": 111,
        "type": "深圳"
      },
      {
        "name": "深圳项目2",
        "value": 222,
        "type": "深圳"
      },
      {
        "name": "合肥项目1",
```

```
    "value": 111,
    "type": "合肥"
  },
  {
    "name": "合肥项目2",
    "value": 222,
    "type": "合肥"
  },
  {
    "name": "成都项目1",
    "value": 111,
    "type": "成都"
  },
  {
    "name": "成都项目2",
    "value": 222,
    "type": "成都"
  },
  {
    "name": "安徽项目1",
    "value": 111,
    "type": "安徽"
  },
  {
    "name": "安徽项目2",
    "value": 222,
    "type": "安徽"
  },
  {
    "name": "北京项目1",
    "value": 111,
    "type": "北京"
  },
  {
    "name": "北京项目2",
    "value": 222,
    "type": "北京"
  }
]
}]
}
```

8. 在蓝图编辑器配置页面，将左侧导入节点栏内的基本柱状图、轮播列表节点拖至画布中。
9. 将基本柱状图的当点击数据项时事件与轮播列表的导入数据接口动作连线。连线后线段内会自动会添加一个串行数据处理逻辑节点。
10. 将新增加的串行数据处理配置面板中的处理方法重命名为数据筛选。  
具体操作方法请参见[串行数据处理节点](#)。

最终连线样式如下图。



11. 配置画布中串行数据处理内数据筛选的处理方法。

- i. 单击串行数据处理逻辑节点，进入右侧配置面板，选择面板内其他配置栏下方的数据筛选处理方法，单击右侧箭头打开脚本编辑区域。
- ii. 在脚本编辑区域，输入代码，完成后单击完成。

同样需要选取临时变量carouselData，并从该数据中过滤出对应的城市数据，输入的转换条件如下。

```

:: 数据筛选
function handler(data,
1  const allData = getLocalValue("carouselData");
2  return allData ? allData.filter(d => {
3    |   return d.type == data.x;
4  }) : [];
    
```

```

const allData = getLocalValue("carouselData");
return allData ? allData.filter(d => {
  return d.type == data.x;
}) : [];
    
```

12. 切换回Dat aV画布编辑器页面，配置轮播列表组件右侧配置页面中的全局配置项。



表格行数设置为柱图对应数据的显示数量，并开启空值隐藏行开关，将轮播列表的空行隐藏，使得组件更加美观。



13. 修改轮播列表的配置面板中的自定义列配置项，将轮播列表每一列的字段名等内容与全局转换器内的字段名相对应。



14. 组件样式和逻辑节点中的处理方法全部配置完成后，单击编辑器页面右上角的预览图标，查看基本柱状图控制轮播列表组件数据筛选的交互效果。



### 7.3. 如何动态控制组件样式

本文档以Tab列表动态控制数字翻牌器组件样式更新为例，为您介绍蓝图编辑器动态控制组件样式的方法。

#### 前提条件

准备好您的交互需求。

本案例的交互需求为：使用Tab列表动态控制数字翻牌器数值的颜色。

#### 背景信息

本案例的场景为：数字翻牌器在项目中经常使用动态数据源，用户需求为根据数据源返回值的大小，使用不同的颜色展示该数值。

本案例实现的功能：使用**Tab列表**组件模拟实际场景的动态数据源（如API、数据库等），根据**数字翻牌器**接收到的value值大小，切换展示不同的数据样式。

## 理解案例交互

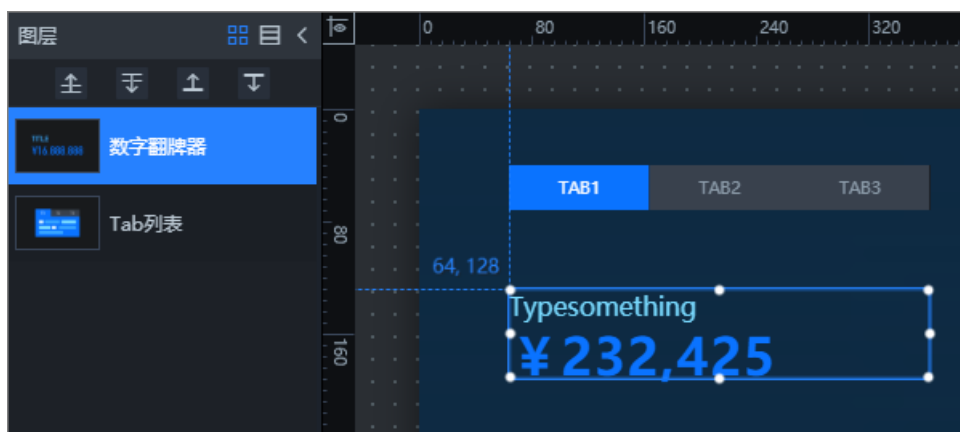
当前案例实现需要用户将交互转成节点和边的关联关系，需要使用的组件有满足需求的事件和动作才能进行配置，否则就无法实现。

- **组件**：组件包含两种能力，事件抛出能力和动作执行能力。本案例使用了翻牌器组件之间的联动。
  - **Tab列表**：具备抛出请求接口返回数据的能力。
  - **数字翻牌器**：属于被联动组件，具有接收数据并执行动作的能力。
- **蓝图编辑器**：通过运用逻辑节点编排的方式，将上游组件的交互/事件与下游组件的执行动作绑定，从而实现页面内组件间的交互联动。

 **说明** 蓝图编辑器配置完成后需要在预览页查看配置效果。

## 配置案例交互

1. 参考画布编辑器**添加资产**，在画布编辑器页面搭建所需要的**Tab列表**和**数字翻牌器**组件。



2. 在画布编辑器内，配置**Tab列表**组件右侧**数据**面板中的**配置数据源**中的**静态数据**数据源内容，修改**Tab**数据内容为两条。




3. 修改数字翻牌器的配置面板中的标题配置项。



4. 样式配置完成后，单击画布编辑器图层栏内数字翻牌器和Tab列表组件，右键选择导出到蓝图编辑器。



5. 在画布编辑器页面左上角，单击蓝图编辑器图标（），切换到蓝图编辑器配置页面。
6. 在蓝图编辑器配置页面，将左侧导入节点栏内的Tab列表和数字翻牌器节点拖至画布中，并按照如下图所示的样式进行连线。



您需要进行两次连线，如上图所示。

- 将Tab列表的当Tab点击时事件与数字翻牌器的导入数据接口动作连线。
  - 将数字翻牌器的当数据接口请求完成时事件与数字翻牌器的更新组件配置动作连线。
7. 连线完成后会自动添加两个串行数据处理逻辑节点到连线中，分别将两个串行数据处理节点配置面板中的处理方法重命名为更新翻牌器value和配置数字颜色，如上图所示。  
具体操作方法请参见[串行数据处理节点](#)。
  8. 配置画布中更新翻牌器value处理方法。
    - i. 单击与Tab列表关联的串行数据处理逻辑节点，进入右侧配置面板，选择面板内其他配置栏下方的更新翻牌器value处理方法，单击右侧箭头打开脚本编辑区域。

- ii. 在脚本编辑区域，输入代码，完成后单击完成。

当前处理方法的示例代码如下，用于模拟动态数据更新翻牌器的value值。

```
return [{
  value: data.id == 1 ? 30 : 50
}];
```



? 说明 在配置Tab列表点击事件动态改变数字翻牌器数据这一步骤时，如果遇到实际场景数字翻牌器已配置动态数据源的情况，可跳过这一步。

9. 配置画布中配置数字颜色处理方法。

- i. 单击蓝图编辑器左上角的画布编辑器图标 (  )，切换返回画布编辑器页面。单击数字翻牌器组件，在右侧的配置面板中，单击下方的复制配置到剪贴板。

- ii. 将复制的内容粘贴到任意的代码编辑器中，观察数据层级关系，并找到您需要更新的组件配置字段。

```

"global": {
  "textStyle": {
    "fontFamily": "Microsoft Yahei"
  },
  "arrangement": "top",
  "distance": 0
},
"title": {
  "content": "根据value修改数字颜色",
  "textStyle": {
    "fontSize": 32,
    "color": "rgba(255,255,255,0.6)",
    "fontWeight": "normal"
  }
},
"counter": {
  "fontFamily": "Microsoft Yahei",
  "justifyContent": "flex-start",
  "prefix": {
    "content": "¥",
    "textStyle": {
      "fontFamily": "Microsoft Yahei",
      "color": "#ffffff",
      "fontSize": 48,
      "fontWeight": "normal"
    }
  },
  "numbers": {
    "textStyle": {
      "color": "#00c0ff",
      "fontSize": 48,
      "fontWeight": "bolder"
    },
    "marginRight": 0,
    "backgroundColor": "rgba(51,51,51,0)",
    "backgroundRadius": 6,
    "separatingBackground": false,
    "digit": 0,
    "rounding": true,
    "decimal": 2,
    "separatingChart": true,
    "separatingSymbol": ",",
    "decimalSymbol": ".",
    "fixedWidth": 0,
    "sameDataFlip": false,
    "duration": 1000,
    "increment": false
  },
  "suffix": {
    "content": "元",
    "textStyle": {

```

- iii. 根据上图的数据结构，编写要传入数字翻牌器的样式配置脚本。

```

{
  "counter": {
    "numbers": {
      "textStyle": {
        "color": "#18F8B8"//将要修改的数据颜色
      }
    }
  }
}

```

- iv. 切换回蓝图编辑器界面，单击另一个串行数据处理逻辑节点，进入右侧配置面板，选择面板内其他配置栏下方的配置数字颜色处理方法，单击右侧箭头打开脚本编辑区域。

v. 在脚本编辑区域，输入代码，完成后单击完成。

当前处理方法的示例代码如下，用于在数据接口请求完成时根据value值大小配置数字颜色样式。

```
return data[0].value - 40 >= 0 ? {
  "counter": {
    "numbers": {
      "textStyle": {
        "color": "#18F8B8"//将要修改的数据颜色
      }
    }
  }
} : {
  "counter": {
    "numbers": {
      "textStyle": {
        "color": "#00C0FF"
      }
    }
  }
}
```

```
1 return data[0].value - 40 >= 0 ?
2   {
3     "counter": {
4       "numbers": {
5         "textStyle": {
6           "color": "#18F8B8"//将要修改的数据颜色
7         }
8       }
9     }
10
11
12   } : {
13     "counter": {
14       "numbers": {
15         "textStyle": {
16           "color": "#00C0FF"
17         }
18       }
19     }
20   }
21 }
```

10. 当前两个处理方法都配置完成后，单击蓝图编辑器页面右上角的预览图标，查看Tab列表动态控制数字翻牌器组件样式的交互效果。





## 7.4. 如何动态控制组件高亮

本文以基本柱状图组件之间控制柱体高亮为例，为您介绍使用蓝图编辑器功能动态控制组件高亮的方法。

### 前提条件

准备好您的交互需求。

本案例的交互需求为：单击基本柱状图1的柱子时，控制自身以及基本柱状图2相关联的柱子高亮显示。


### 背景信息

本案例的适用场景：当有多个数据关联组件时，选择某项数据，通过此功能可以让相关组件的对应数据项高亮，可以直观地看到数据情况。

### 理解案例交互

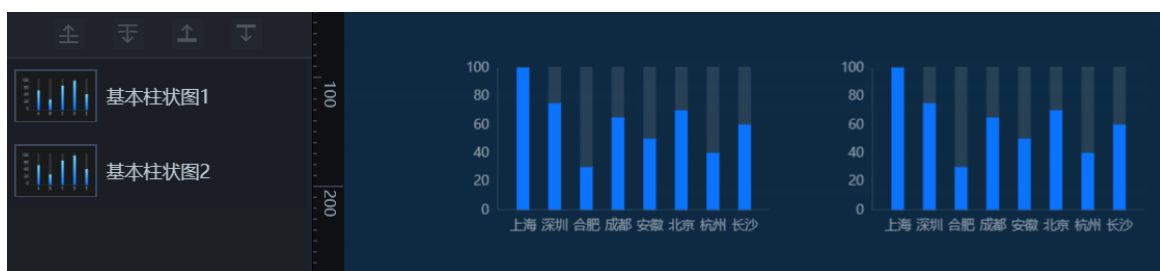
当前案例实现需要用户将交互转成节点和边的关联关系，需要使用的组件有满足需求的事件和动作才能进行配置，否则就无法实现。

- **组件**：组件包含两种能力，事件抛出能力和动作执行能力。本案例使用了基本柱状图组件之间的联动。
  - **基本柱状图1**：这个组件具备抛出单击事件的能力。
  - **基本柱状图2**：属于被联动组件，具有接收数据并执行动作的能力。
- **蓝图编辑器**：通过运用逻辑节点编排的方式，将上游组件的交互/事件与下游组件的执行动作绑定，从而实现页面内组件间的交互联动。

 **说明** 蓝图编辑器配置完成后需要在预览页查看配置效果。


### 配置案例交互

1. 参考画布编辑器**添加资产**，在画布编辑器页面搭建所需要的两个基本柱状图组件。



2. 单击画布编辑器图层栏内两个基本柱状图组件，右键选择导出到蓝图编辑器。



3. 在画布编辑器页面左上角，单击蓝图编辑器图标（），切换到蓝图编辑器配置页面。
  4. 在蓝图编辑器配置页面，将左侧导入节点栏内的基本柱状图1和基本柱状图2节点拖至画布中。
  5. 将基本柱状图1的当点击数据项时事件与基本柱状图2的高亮动作连线。
  6. 连线后，线段内会自动添加一个串行数据处理逻辑节点，将配置面板中的处理方法重命名为目标高亮。
- 具体操作方法请参见[串行数据处理节点](#)。
7. 将串行数据处理逻辑节点的右侧圆点与基本柱状图1的高亮动作连线。
- 最终连线的结果如下图所示。



连线1：实现关联柱状图高亮的效果。

连线2：实现自身柱状图高亮的效果。

- 单击**串行数据处理**进入逻辑节点的配置界面，在选中上下游组建的事件和动作，单击查看data参数字段提示处，在左侧弹窗中查看上下游组件内的数据定义和数据示例内容。查看并根据需求选中实现柱体高亮所需要的字段。



数据示例：

```

{
  "data": {},
  "options": {
    "style": {
      "stroke": "#f00",
      "fill": ""
    },
    "selectMode": "single",
    "cancelHighlightFirst": false
  }
}
    
```

**说明** 我们要完成的高亮操作是需要填充高亮，因此将数据示例中的stroke改为fill。

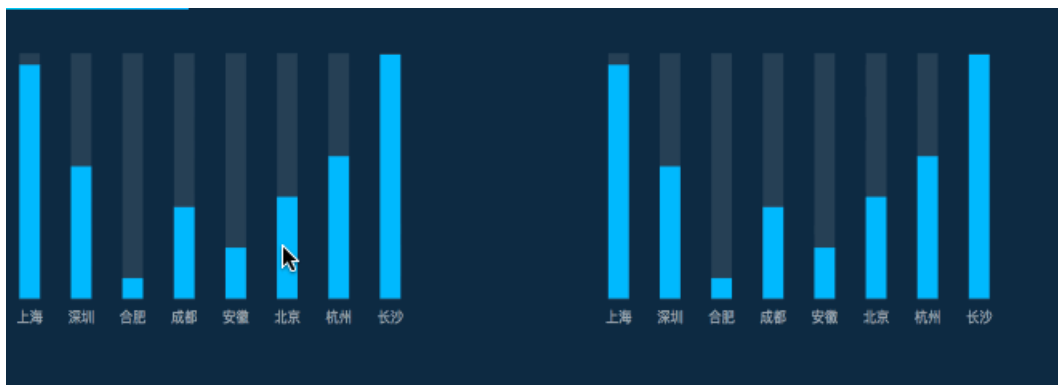
9. 配置画布中串行数据处理内目标高亮这个处理方法。
  - i. 单击串行数据处理逻辑节点，进入右侧配置面板，选择面板内其他配置栏下方的目标高亮处理方法，单击右侧箭头打开脚本编辑区域。

- ii. 在脚本编辑区域，输入代码，完成后单击完成。

当前处理方法的示例代码如下。

```
return {
  data: {
    "x": data.x//需要填充的数据选项
  }, options: {
    style: {
      fill: "3DE7C9"//填充方式及颜色
    },
    "selectMode": "single",
    "cancelHighlightFirst": true//是否先取消之前高亮
  }
}
```

10. 处理方法配置完成后，单击蓝图编辑器配置页面右上角的预览图标，预览交互效果。



## 7.5. 如何在请求数据接口时传递动态参数

本文档为您介绍使用蓝图编辑器功能，在请求数据接口时传递动态参数，实现Tab列表和基本柱状图联动的方法。

### 前提条件

准备好您的交互需求。

本案例的交互需求为：当单击Tab列表的某一页签时，基本柱状图传入API数据源内对应的id字段值作为参数，实现基本柱状图的数据联动效果。

### 背景信息

旧版本的DataV使用回调ID的方法来配置图表数据联动。但是在回调ID配置完成后，并不能直观动态地看到回调ID的传递者和接收者。新版本的DataV为您提供了蓝图编辑器功能，蓝图编辑器通过连线将上下游串联起来，使得数据链路更清晰。

本案例的实际应用场景为：图表联动、图表下钻、服务调用、数据报警等需要通过请求数据接口时带上动态参数的联动效果展示。

### 理解案例交互

当前案例实现需要用户将交互转成节点和边的关联关系，需要使用的组件有满足需求的事件和动作才能进行配置，否则就无法实现。

- **组件**：组件包含两种能力，事件抛出能力和动作执行能力。本案例使用了Tab列表和基本柱状图组件之

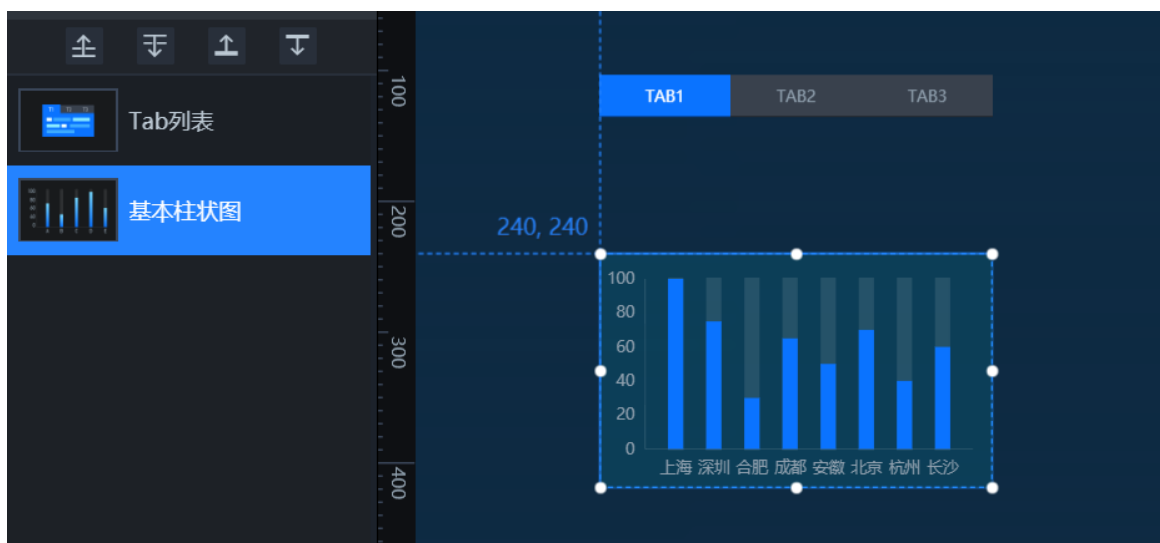
间的联动。

- Tab列表：这个组件具备抛出点击事件的能力。
- 基本柱状图：属于被联动组件，具有接收数据并执行动作的能力。
- 蓝图编辑器：通过运用逻辑节点编排的方式，将上游组件的交互/事件与下游组件的执行动作绑定，从而实现页面内组件间的交互联动。

说明 蓝图编辑器配置完成后需要在预览页查看配置效果。

## 配置案例交互

1. 参考画布编辑器添加资产，在画布编辑器页面搭建所需要的基本柱状图和Tab列表组件。



2. 配置Tab列表和基本柱状图的样式和数据。
  - i. 单击Tab列表右侧的数据面板中，将静态数据源中的内容修改成两条。


- ii. 将数据源内容配置为如下所示的数据。



```
[
  {
    "id": 1,
    "content": "销售额"
  },
  {
    "id": 2,
    "content": "数量"
  }
]
```

- iii. 单击基本柱状图，在右侧的数据面板中，将数据源配置为如下所示的API数据。




 **注意** API数据源的格式为http://api.test，并且需要与基本柱状图的默认字段相匹配。

- 3. 组件配置完样式后，单击画布编辑器图层栏内基本柱状图和Tab列表组件，右键选择导出到蓝图编辑器。





4. 在画布编辑器页面左上角，单击**蓝图编辑器**图标（），切换到蓝图编辑器配置页面。
5. 在蓝图编辑器配置页面，将左侧节点栏内的**Tab列表**和**基本柱状图**节点拖至画布中。
6. 将**Tab列表**的**当Tab点击时**事件与**基本柱状图**的**请求基本柱状图**接口动作连线。
7. 连线中会自动添加一个**串行数据处理**逻辑节点，将逻辑节点内的处理方法重命名为**动态传参**。  
具体操作方法请参见[串行数据处理节点](#)。  
最终连线后效果如下图所示。



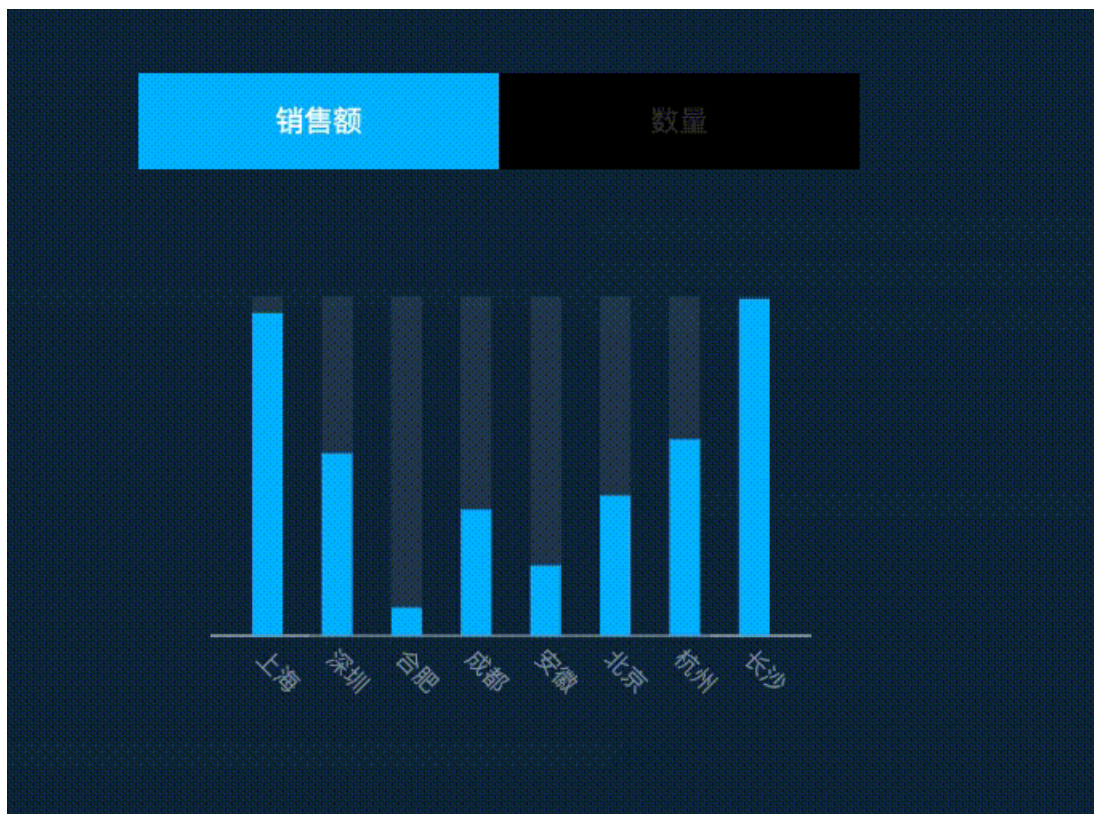
8. 配置画布中串行数据处理内动态传参这个处理方法。
  - i. 单击串行数据处理逻辑节点，进入右侧配置面板，选择面板内其他配置栏下方的动态传参处理方法，单击右侧箭头打开脚本编辑区域。
  - ii. 在脚本编辑区域，输入代码，完成后单击完成。

当前处理方法的示例代码如下。

```
return {
  id:data.id //柱图api将添加参数id:data.id，重新请求数据。
};
```

**说明** 当单击Tab时，触发数据请求。柱图API将添加参数 `id:data.id`，重新请求数据。例如 `data.id` 为1，请求API将变为 `http://api.test?id=1`。

9. 处理方法配置完成后，单击蓝图编辑器页面右上角的预览图标，查看Tab列表控制基本柱状图在请求数据接口时传递动态参数的交互效果。



## 7.6. 如何通过合并请求进行数据分发

本文档以通用标题组件请求数据，并分发给数字翻牌器和多行文本组件，完成数据更新为例，介绍如何使用蓝图编辑器功能，通过合并请求进行数据分发。

### 前提条件

准备好您的交互需求。

本案例的交互需求为：通用标题组件请求数据，并将请求到的数据分发给数字翻牌器及多行文本组件。

### 背景信息

本案例的实际应用场景为：多个组件共用一个数据请求，产生单个组件请求数据，并分发给其他组件。

**说明** 本文档适用于多个组件的数据可以通过API或SQL数据源等一次都获取到时，减少数据请求次数。

### 理解案例交互

当前案例实现需要用户将交互转成节点和边的关联关系，需要使用的组件有满足需求的事件和动作才能进行配置，否则就无法实现。


- **组件**：组件包含两种能力，事件抛出能力和动作执行能力。本案例使用了翻牌器组件之间的联动。
  - **通用标题**：组件具备抛出请求接口返回数据的能力。
  - **数字翻牌器&多行文本**：属于被联动组件，具有接收数据并执行动作的能力。
- **蓝图编辑器**：通过运用逻辑节点编排的方式，将上游组件的交互/事件与下游组件的执行动作绑定，从而实现页面内组件间的交互联动。

 说明 蓝图编辑器配置完成后需要在预览页查看配置效果。

### 配置案例交互

1. 在画布编辑器页面，搭建所需要的通用标题、多行文本和数字翻牌器组件，详情请参见[添加资产](#)。
2. 在数字翻牌器的配置面板中，清空标题名、前缀内容和后缀内容配置项。




 注意 此步骤为可选步骤，您可以根据需求自定义标题名、前缀或后缀内容。

3. 将通用标题组件放置于画布外，并修改组件的数据内容。



本案例中，通用标题组件配置的数据如下。


```
[
  {
    "num": "123",
    "text": "这是多行文本组件",
    "value": "接收数据组件"
  }
]
```

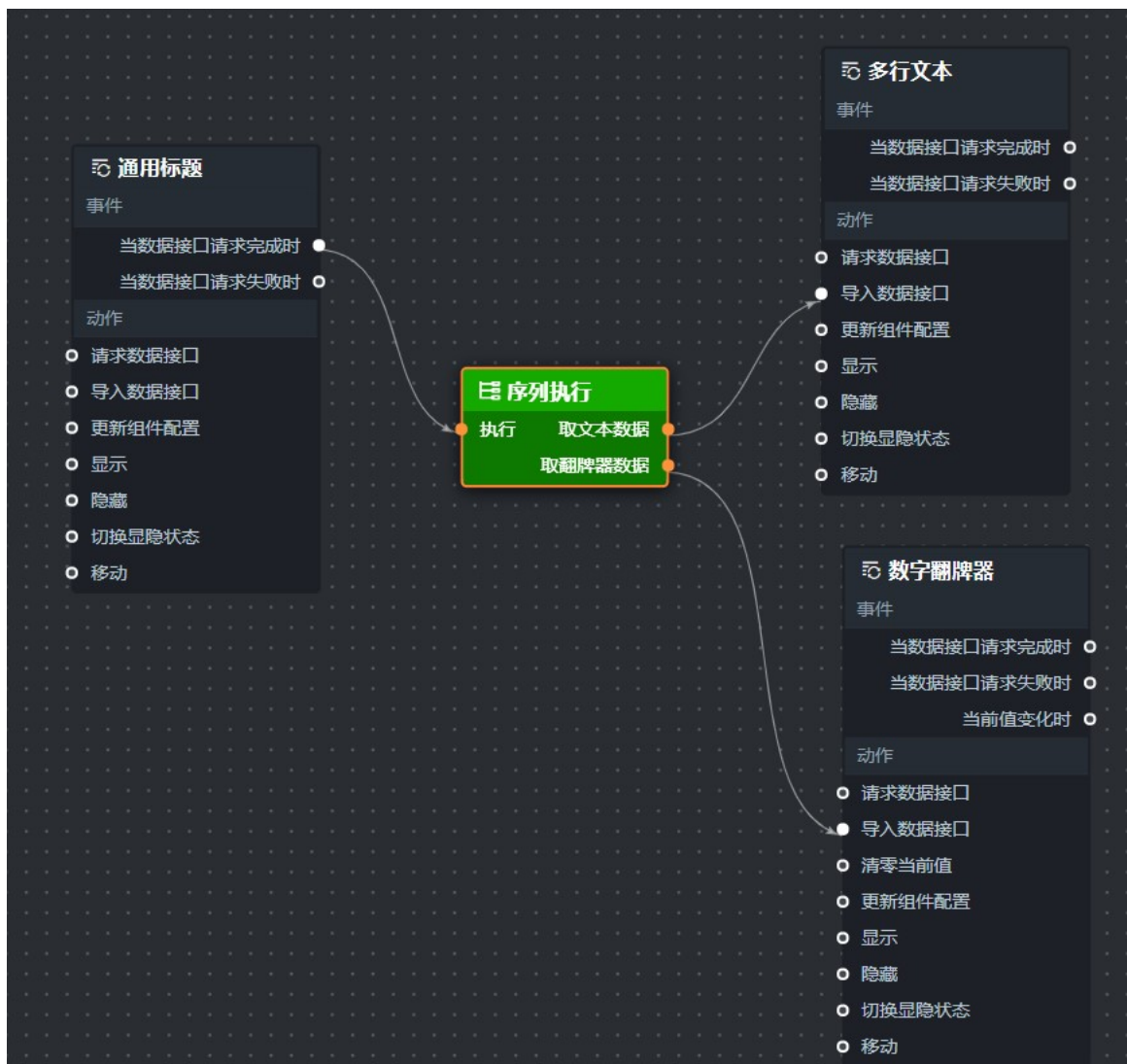
 **注意** 本案例内的通用标题组件只用于请求数据，放置于画布外，在预览发布时不会显示在大屏上。



4. 组件配置完成后，单击画布编辑器图层栏内三个组件，右键选择导出到蓝图编辑器。



5. 在画布编辑器页面左上角，单击**蓝图编辑器**图标（），切换到蓝图编辑器配置页面。
6. 在蓝图编辑器页面，将左侧导入节点栏内的**通用标题**、**多行文本**和**数字翻牌器**节点拖至画布中。
7. 拖动添加一个**序列执行**逻辑节点到画布中，在节点配置面板中新增一个**处理方法**后将**序列执行**内的两个**处理方法**重命名为**取文本数据**和**取翻牌器数据**。  
具体操作方法请参见**序列执行节点**。
8. 将画布中的多个节点按如下样式连线。



本案例需要进行两次连线：

- 将通用标题的当数据接口请求完成时事件，与序列执行节点左侧的执行动作相连，再将节点右侧的取文本数据处理方法与多行文本的导入数据接口动作连线。
- 将通用标题的当数据接口请求完成时事件，与序列执行节点左侧的执行动作相连，再将节点右侧的取翻牌器数据处理方法与多行文本的导入数据接口动作连线。

9. 配置画布中序列执行内的取文本数据和取翻牌器数据的处理方法。

- i. 单击序列执行逻辑节点，进入右侧配置面板，选择面板内其他配置栏下方的取文本数据处理方法，单击右侧箭头打开脚本编辑区域。
- ii. 在脚本编辑区域，输入代码，完成后单击完成。

当前处理方法的示例代码如下。

```
return [{
  value:data[0].text
}];
```

iii. 使用同样的方式配置取翻牌器数据处理方法。

当前处理方法的示例代码如下。

```
return [{
  value:data[0].num
}];
```

**说明** 通用标题内的当数据接口请求完成时事件，事件中包含该组件所有数据。数字翻牌器和多行文本组件的导入数据接口动作接收到上游传过来的内容后，会用新导入的数据覆盖组件原本的数据。序列执行节点负责将上游抛出的事件内容进行加工处理，把处理后的数据直接给数字翻牌器和多行文本组件使用。

10. 处理方法都配置完成后，单击蓝图编辑器配置页面右上角的预览图标，查看数据分发效果。



## 7.7. 时间戳联动组件查询

本文档以时间器组件和通用标题之间联动查询组件，为您介绍使用蓝图编辑器功能通过时间戳参数联动页面内与时间戳相关的组件数据查询。

### 前提条件

准备好您的交互需求。

本案例的交互需求为：实现在页面上每隔3s动态显示当前时间戳的功能。

### 理解案例交互

当前案例实现需要用户将交互转成节点和边的关联关系，需要使用的组件有满足需求的事件和动作才能进行配置，否则就无法实现。

- **组件**：组件包含两种能力，事件抛出能力和动作执行能力。本案例使用了时间器和通用标题组件之间的联动。
  - **时间器**：这个组件具备抛出当前时刻的能力。
  - **通用标题**：属于被联动组件，具有接收数据并执行动作的能力。
- **蓝图编辑器**：通过运用逻辑节点编排的方式，将上游组件的交互/事件与下游组件的执行动作绑定，从而实现页面内组件间的交互联动。

**说明** 蓝图编辑器配置完成后需要在预览页查看配置效果。

### 配置案例交互



1. 参考画布编辑器**添加资产**，在画布编辑器页面搭建所需要的时间器和通用标题组件。



2. 在画布编辑器内，配置**时间器**组件右侧**配置面板**，打开面板的**定时回调设置**，配置回调抛出间隔为3s抛出一次当前时间。




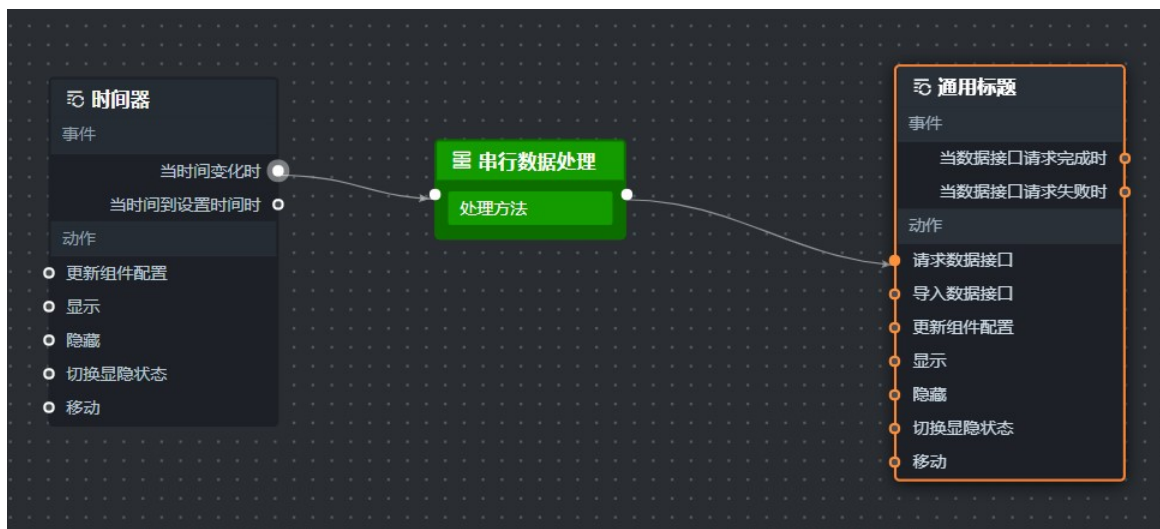
3. 在画布编辑器内，配置**通用标题**组件右侧**数据数据**面板，单击面板中的**配置数据源**，配置以数据库查询为例的样式，并编写SQL。如下图SQL内，开头的变量表示回调变量名，在真实的请求中会被替换掉，具体回调内容详情可参见[DataV回调ID实现图表联动功能](#)。



4. 单击画布编辑器图层栏内时间器和通用标题组件，右键选择导出到蓝图编辑器。



5. 在画布编辑器页面左上角，单击蓝图编辑器图标（），切换到蓝图编辑器配置页面。
6. 在蓝图编辑器配置页面，将左侧节点栏内的时间器和通用标题节点拖至画布中。
7. 将时间器的当前时间变化时事件与通用标题的请求数据接口动作连线。
8. 连线中会自动添加一个串行数据处理逻辑节点。  
具体操作方法请参见[串行数据处理节点](#)。  
最终连线的结果如下图所示。



9. 配置画布中串行数据处理内的处理方法。

- i. 单击串行数据处理逻辑节点，进入右侧配置面板，选择面板内其他配置栏下方的处理方法，单击右侧箭头打开脚本编辑区域。
- ii. 在脚本编辑区域，输入代码，完成后单击完成。

当前处理方法的示例代码如下。

```
return {
  start_time:
    new Date(data.time).valueOf() - 60 * 1000, // 当前时间前 60s
  end_time:
    new Date(data.time).valueOf() // 当前时间
}
```

- a. 时间器组件在设置了定时触发后，每3s会抛出一次当时间变化时的事件，事件中包含time: 当前时间。
- b. 通用标题组件的请求数据接口动作接收到上传过来的内容后，将上游内容比如 x: 1作为发起请求时的 query参数，并发起请求。
- c. 串行数据处理节点负责将上游抛出的事件内容进行加工处理，传递给通用标题组件作为请求时的 query参数，也就是替换掉第二步操作中填写的:变量名。

10. 处理方法配置完成后。

- i. 时间器组件的{time: "Mon Oct 28 2019 15:47:29 GMT+0800 (China Standard Time)"}
- ii. 通用标题 执行sql查询之前，会将SELECT :start\_time as value1, :end\_time as value2替换为SELECT 1572248800098 as value1, 1572248860098 as value2, 最终执行sql并返回结果到前端。
- iii. 在编辑页调试时，可以在编辑页URL上添加querv参数，比如 [http://datav.alibaba-inc.com/admin/screen/123456?start\\_time=1572248800098&end\\_time=1572248860098](http://datav.alibaba-inc.com/admin/screen/123456?start_time=1572248800098&end_time=1572248860098) 进行查询结果调试，最终展示结果以预览页为准。

单击蓝图编辑器配置页面右上角的预览图标，预览交互效果。

## 7.8. 如何实现跨屏联动

本文档以配置两个不同可视化应用上的Tab列表和通用标题之间联动为例，为您介绍如何使用蓝图编辑器实现跨屏联动效果。

## 前提条件

准备好您的交互需求。

本案例的交互需求为：主屏幕单击Tab列表来触发副屏幕通用标题的修改。

## 背景信息

本案例的适用场景：当您有多个可视化应用，并希望多个可视化应用内的组件能展示相互联动的效果。

## 理解案例交互

当前案例实现需要用户将交互转成节点和边的关联关系，需要使用的组件有满足需求的事件和动作才能进行配置，否则就无法实现。

- 组件：组件包含两种能力，事件抛出能力和动作执行能力。本案例使用了Tab列表和通用标题组件之间的联动。
  - Tab列表：这个组件具备点击事件的能力。
  - 通用标题：属于被联动组件，具有接收数据并执行动作的能力。
- 蓝图编辑器：通过建立两个屏幕的WebSocket服务，将上游组件的交互/事件与下游组件的执行动作绑定，从而实现单击主屏幕的Tab列表触发副屏幕通用标题的修改。


### 说明

- WebSocket服务格式的详情请参见[自建WebSocket节点服务说明](#)。
- 蓝图编辑器配置完成后需要在预览页查看配置效果。

## 配置案例内主屏幕交互

1. 参考画布编辑器[添加资产](#)，在主屏幕的画布编辑器页面搭建所需要的Tab列表组件。
2. 单击画布编辑器图层栏内Tab列表组件，右键选择[导出到蓝图编辑器](#)。



3. 在画布编辑器页面左上角，单击**蓝图编辑器**图标（），切换到蓝图编辑器配置页面。
4. 在蓝图编辑器配置页面，将左侧导入节点栏内的**Tab列表**节点和逻辑节点栏内的**串行数据处理**和**WebScket**节点一起拖至画布中。
5. 配置画布中**串行数据处理**内的处理方法。
  - i. 单击**串行数据处理**逻辑节点，进入右侧**配置面板**，选择面板内其他配置栏下方的处理方法，单击右侧箭头打开脚本编辑区域。
  - ii. 在脚本编辑区域，输入代码，单击**完成**。

当前处理方法的示例代码如下。

```
return [{"value":data.content}];
```


6. 配置画布中**WebScket**逻辑节点。
  - i. 单击**WebScket**逻辑节点，进入右侧**配置面板**，参见**WebScket配置项说明**配置面板中的参数内容。
  - ii. 在**socket服务地址**区域，输入测试地址。  
 由于当前可视化应用是主屏幕，属于发送端，所以需要添加一个**发送消息**。
  - iii. 添加一个**发送消息**，并将消息名命名为下一个。
7. 在蓝图编辑器配置页面，将画布中的这三个节点按下图样式连线。




## 配置案例内副屏幕交互

1. 参考画布编辑器[添加资产](#)，在副屏幕的画布编辑器页面搭建所需要的通用标题组件。
2. 单击画布编辑器图层栏内通用标题组件，右键选择导出到蓝图编辑器。

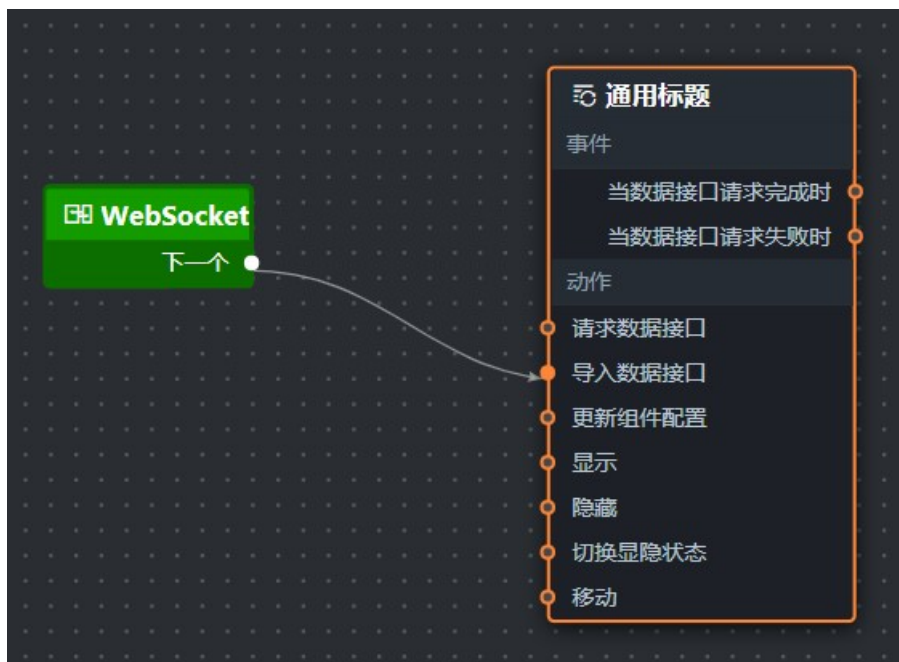


3. 在画布编辑器页面左上角，单击**蓝图编辑器**图标（），切换到**蓝图编辑器配置**页面。
4. 在**蓝图编辑器配置**页面，将左侧**导入节点**栏内的**通用标题**节点和**逻辑节点**栏内的**WebScoket**节点一起拖至画布中。

 **说明** 因为主屏幕内socket发出的消息已经处理，并且处理后的数据能够导入到副屏幕内的通用标题中，所以在副屏幕内不用再处理数据格式。

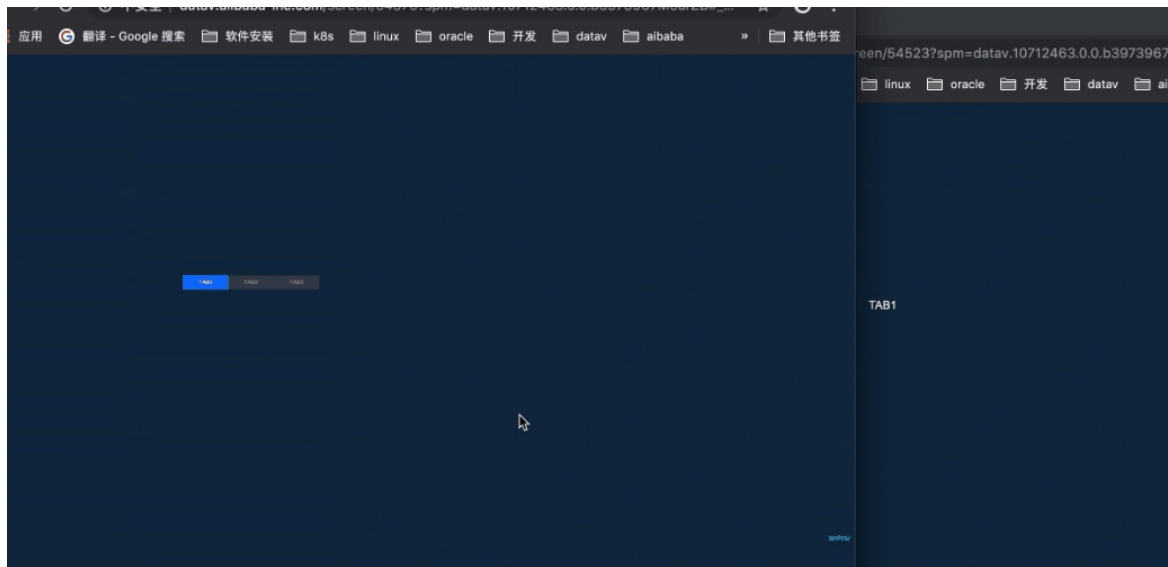
5. 配置画布中**WebScoket**逻辑节点。
  - i. 单击**WebScoket**逻辑节点，进入右侧**配置面板**，参见**WebScoket配置项说明**配置面板中的参数内容。
  - ii. 在**socket服务地址**区域，输入和主屏幕相同的地址。  
由于当前可视化应用是主屏幕属于接收端，所以需要添加一个**接收消息**。
  - iii. 添加一个**接收消息**，并将消息名命名为下一个。
6. 将画布中的这两个节点按下图样式连线。





**注意** 主屏幕和副屏幕的分组要一致，因为副屏幕是接收端，所以配置接收消息即可。

7. 单击蓝图编辑器配置页面右上角的预览图标，查看跨屏联动的交互效果。



## 7.9. 开发者自定义组件如何使用蓝图编辑器

本文档示例为开发者的自定义组件提供功能开发的参考，主要介绍三个蓝图编辑器内容的实现方法：事件触发、导入数据接口、添加交互动作。

### 前提条件

本文档功能实现需要用户操作使用桌面版开发工具，详情请参见开发工具使用说明中的[桌面版开发工具使用说明](#)。

### 背景信息

本文档适用用户在开发自定义组件的时候，为了使得自定义组件能同样使用Dat aV专业版中蓝图编辑器的功能，需要使用与蓝图编辑器一致的规则开发自定义组件。在使用开发者工具创建完成组件后，本地的组件包文件夹下会生成 *index.js* 和 *package.json* 文件，本文三点内容的实现需要在这两个文件中修改内容来操作。

## 理解交互功能

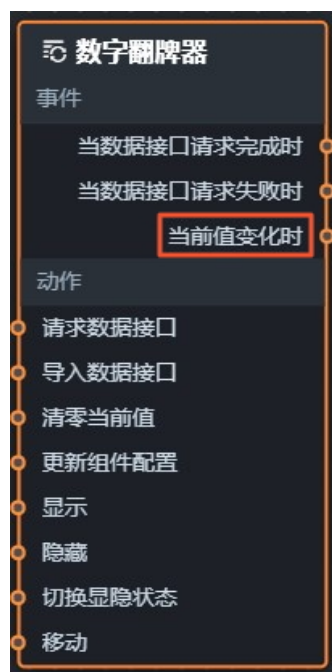
当前案例实现需要用户将交互转成节点和边的关联关系，需要使用的组件有满足需求的事件和动作才能进行配置，否则就无法实现。

- **组件**：组件包含两种能力，事件抛出能力和动作执行能力。本案例使用了数字翻牌器组件来举例说明。
- **蓝图编辑器**：通过运用逻辑节点编排的方式，将上游组件的交互/事件与下游组件的执行动作绑定，从而实现页面内组件间的交互联动。


 **说明** 蓝图编辑器配置完成后需要在预览页查看配置效果。

## 事件触发

在自定义组件中实现蓝图编辑器中事件触发功能，以下图数字翻牌器组件的当前值变化事件为例。



1. 在使用开发者工具创建完成组件后，打开 *package.json* 文件。
2. 在 *package.json* 文件中搜索 *dat av* 字段下的 *events* 字段。

 **说明** 每一个蓝图编辑器中的事件必须要和自定义组件中 *events* 字段中定义的事件保持一致，否则编辑器无法识别。

3. 在当前 *events* 字段中定义事件，自定义修改 *event-name* 事件名下的描述和 *value* 字段名下的描述。

```
"events": {
  "event-name": {
    "description": "当值发生变化时",
    "fields": {
      "value": {
        "description": "值"
      }
    }
  }
},
```

① 说明 事件名和字段名都可自定义多个，自定义设置越多，蓝图编辑器节点页面显示事件越多。

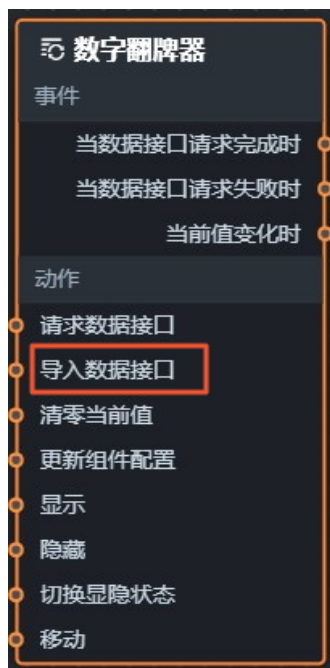
4. 打开 *index.js* 文件，设置代码中的 *emit* 的事件名，如下方所示。

```
this.emit('event-name', obj);
```

① 说明 *emit* 中的事件名要和 *package.json* 文件中的事件名保持一致，传递参数必须是 *object* 类型，可以传递多个字段。

## 导入数据接口

在自定义组件中实现蓝图编辑器中导入数据接口动作，以下图数字翻牌器组件的导入数据接口动作为例。



1. 在使用开发者工具创建完成组件后，打开 *index.js* 文件。
2. 在 *index.js* 文件中搜索 *dat av* 字段下的 *render* 字段。
3. 在当前 *render* 字段中自定义设置导入数据接口动作代码逻辑。

```
render: function(data){//data 通过 --->(导入数据借口==组件数据相应结果) 得到的数据。  
  
  //在此处进行组件的逻辑编写  
  
}
```

❓ 说明 动作逻辑都可自定义多个，自定义设置越多，蓝图编辑器节点页面显示动作越多。

### 添加交互动作

在自定义组件中实现蓝图编辑器中多个交互动作，以下图数字翻牌器组件的显示和隐藏动作为例。



1. 在使用开发者工具创建完成组件后，打开package.json文件。
2. 在package.json文件中搜索dat av字段下的publicHandler字段。
3. 在当前publicHandler字段中自定义设置显示和隐藏的描述，但是需要与index.js文件中的方法名保持一致。

```
"publicHandler": {  
  "show": {  
    "description": "显示组件"  
  },  
  "hide": {  
    "description": "隐藏组件"  
  }  
},
```

② 说明 动作逻辑都可自定义多个，自定义设置越多，蓝图编辑器节点页面显示动作越多。

4. 打开 `index.js` 文件，设置代码中的 `container` 的内容，如下方所示。

```
show() {
  this.container.show(); //显示代码，开发者自定义
}
hide() {
  this.container.hide(); //隐藏代码，开发者自定义
}
```

```
this.container.show(); //显示代码，开发者自定义
```

```
this.container.hide(); //隐藏代码，开发者自定义
```

② 说明 `container` 中的字段名要和 `package.json` 文件中的字段名保持一致。

## 7.10. 如何使用时间器触发数据定时更新

本文档以时间器组件控制通用标题组件数据更新为例，为您介绍使用蓝图编辑器功能控制时间器触发通用标题数据定时更新的方法。

### 前提条件

准备好您的交互需求。

本案例的交互需求为：当时间器内的时间到达2019-08-19 17:20:00时（具体时间根据项目实际情况自定义），通用标题组件数据更新显示为活动开始。

### 背景信息

本案例的适用场景为双十一等场景活动的定时触发。

### 理解案例交互

当前案例实现需要用户将交互转成节点和边的关联关系，需要使用的组件有满足需求的事件和动作才能进行配置，否则就无法实现。

- **组件**：组件包含两种能力，事件抛出能力和动作执行能力。本案例使用了翻牌器组件之间的联动。
  - **时间器**：组件具备抛出请求接口返回数据的能力。
  - **通用标题**：属于被联动组件，具有接收数据并执行动作的能力。
- **蓝图编辑器**：通过运用逻辑节点编排的方式，将上游组件的交互/事件与下游组件的执行动作绑定，从而实现页面内组件间的交互联动。

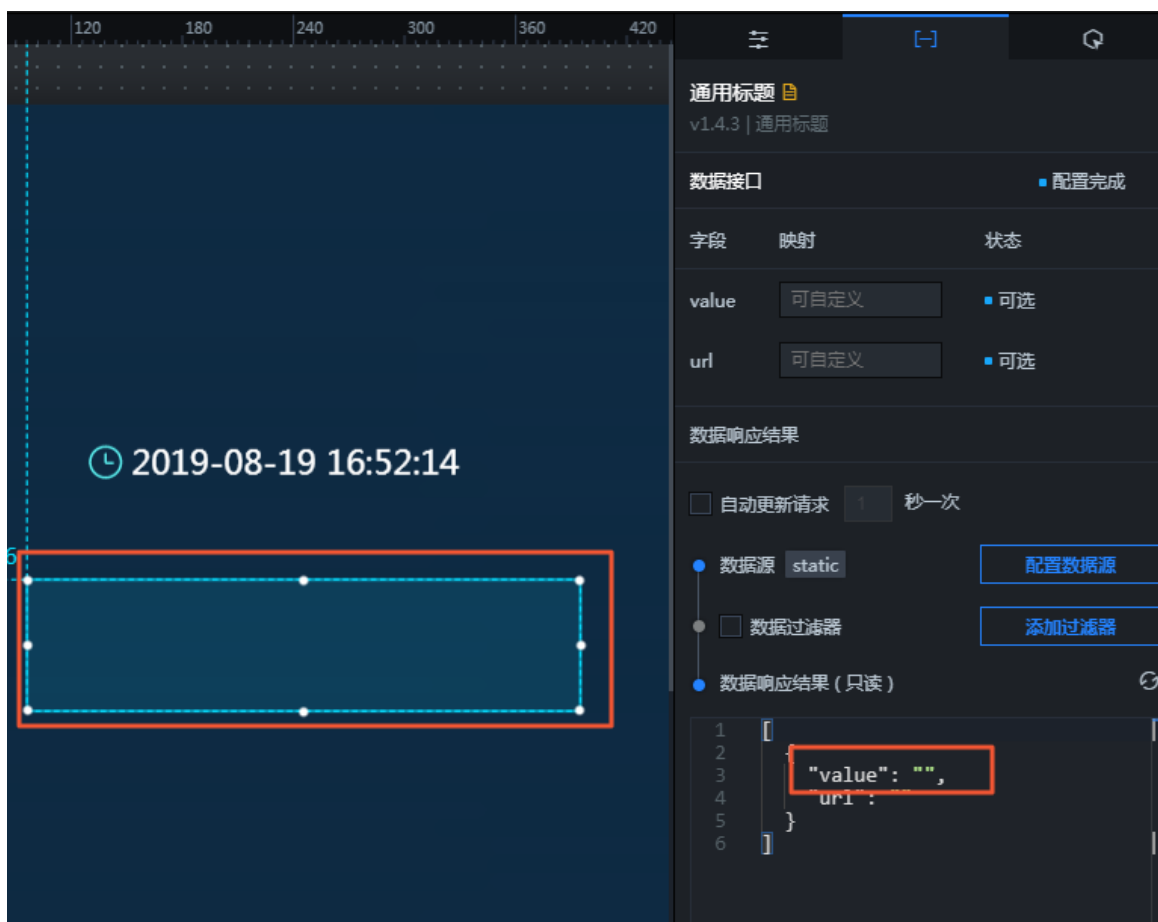
② 说明 蓝图编辑器配置完成后需要在预览页查看配置效果。

### 配置案例交互

1. 参考画布编辑器[添加资产](#)，在画布编辑器页面搭建所需要的时间器和通用标题组件。



2. 在通用标题组件的数据面板中，清空value字段的数据。



🔍 说明 此步骤为可选，表示在活动未开始时，可视化应用上不显示通用标题的内容。您也可以自定义内容，显示在可视化应用上。


3. 在时间器组件的配置面板中，单击回调配置，打开定点回调设置，设置定点抛出时间的为当前案例中交互所需的时间。



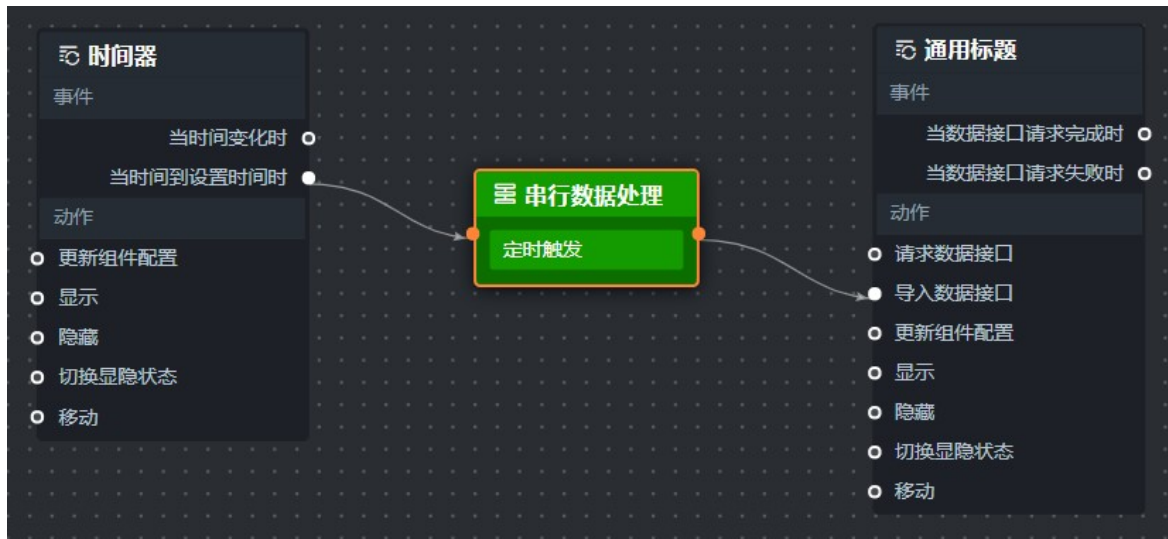
本案例设置定点抛出时间为2019-08-19 17:20:00。

- 单击画布编辑器图层栏内时间器和通用标题组件，右键选择导出到蓝图编辑器。



5. 在画布编辑器页面左上角，单击**蓝图编辑器**图标（），切换到蓝图编辑器配置页面。
6. 在蓝图编辑器页面，将左侧导入节点栏内的**时间器**和**通用标题**节点拖至画布中。
7. 将**时间器**的**当时间到设置时间时**事件与**通用标题**的**导入数据接口**动作连线。
8. 连线中会自动添加一个**串行数据处理**逻辑节点，将配置面板中的**处理方法**重命名为**定时触发**。  
具体操作方法请参见[串行数据处理节点](#)。  
最终连线的结果如下图所示。



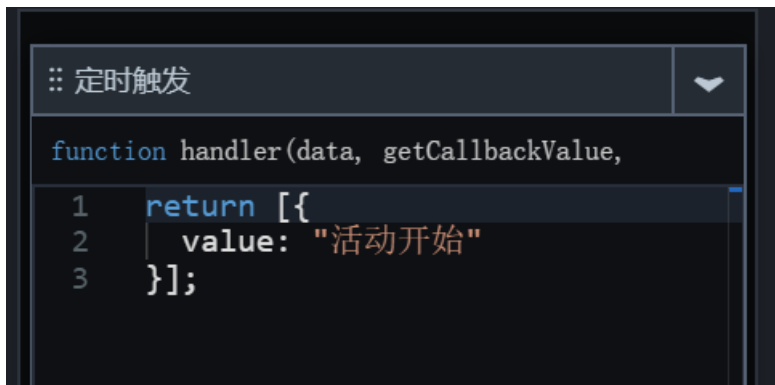


9. 配置画布中串行数据处理内定时触发这个处理方法。

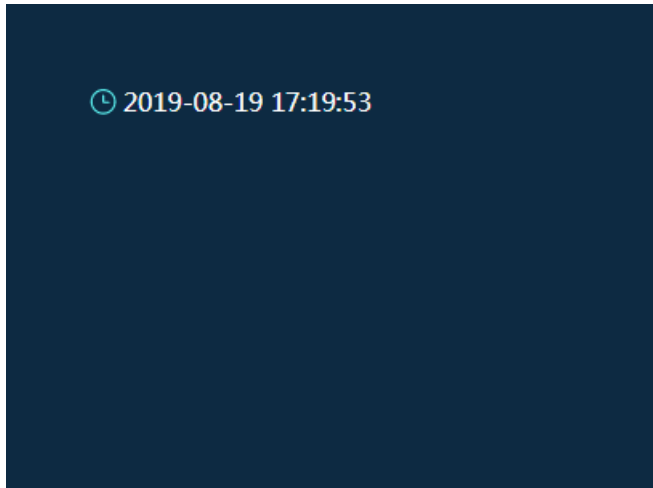
- i. 单击串行数据处理逻辑节点，进入右侧配置面板，选择面板内其他配置栏下方的定时触发处理方法，单击右侧箭头打开脚本编辑区域。
- ii. 在脚本编辑区域，输入代码，完成后单击完成。

当前处理方法的示例代码如下。

```
return [{  
  value: "活动开始"  
}];
```



10. 处理方法配置完成后，单击蓝图编辑器配置页面右上角的预览图标，查看时间器控制通用标题数据更新的交互效果。



## 7.11. 如何展示多个实时数据相加结果

本文档以使用目标数字翻牌器展示其他两个数字翻牌器的实时数据相加结果为例，为您介绍如何使用蓝图编辑器功能展示多个实时数据相加的结果。


### 前提条件

准备好您的交互需求。

本案例的交互需求为：将数字翻牌器A的上半年绿色能量（实时请求数据）和数字翻牌器B的下半年绿色能量（实时请求数据）通过蓝图编辑器功能相加，将全年的绿色能量数据显示在数字翻牌器C组件上。

### 背景信息

本案例的适用场景：通过蓝图编辑器功能实现多个实时数据相加。

 **注意** 本案例只适用于实时数据，如果您使用的是静态数据和实时数据相加，只需要在实时数据组件中添加过滤器后再相加即可。

### 理解案例交互

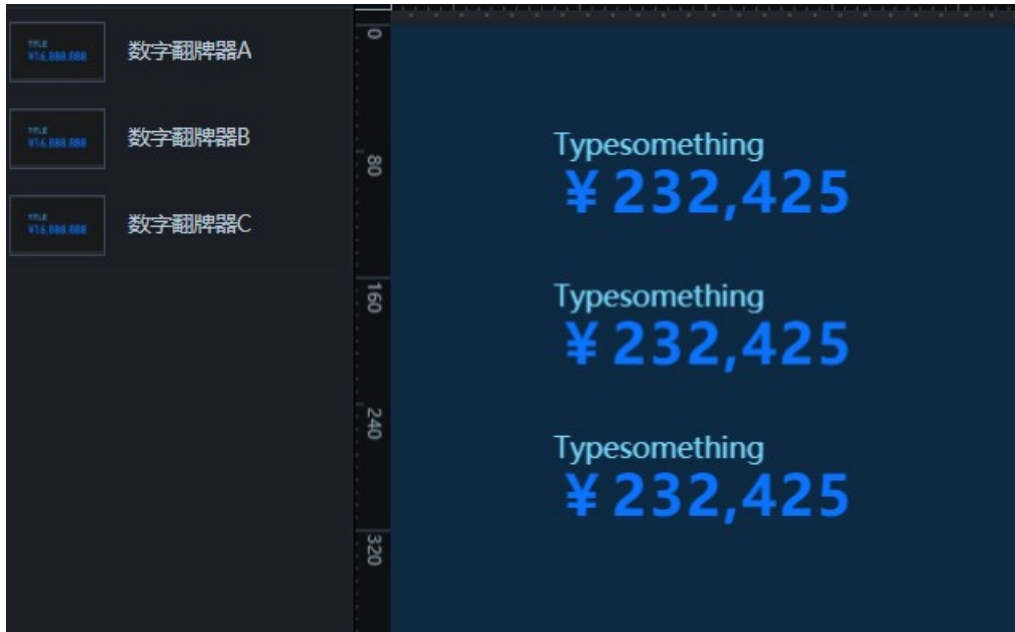
本案例实现需要用户将交互转成节点和边的关联关系，需要使用的组件有满足需求的事件和动作才能进行配置，否则就无法实现。

- **组件**：组件包含两种能力，事件抛出能力和动作执行能力。本案例使用了翻牌器组件之间的联动。
  - **数字翻牌器A&B**：这两个组件具备抛出请求接口返回数据的能力。
  - **数字翻牌器C**：属于被联动组件，具有接收数据并执行动作的能力。
- **蓝图编辑器**：通过运用逻辑节点编排的方式，将上游组件的交互/事件与下游组件的执行动作绑定，从而实现页面内组件间的交互联动。

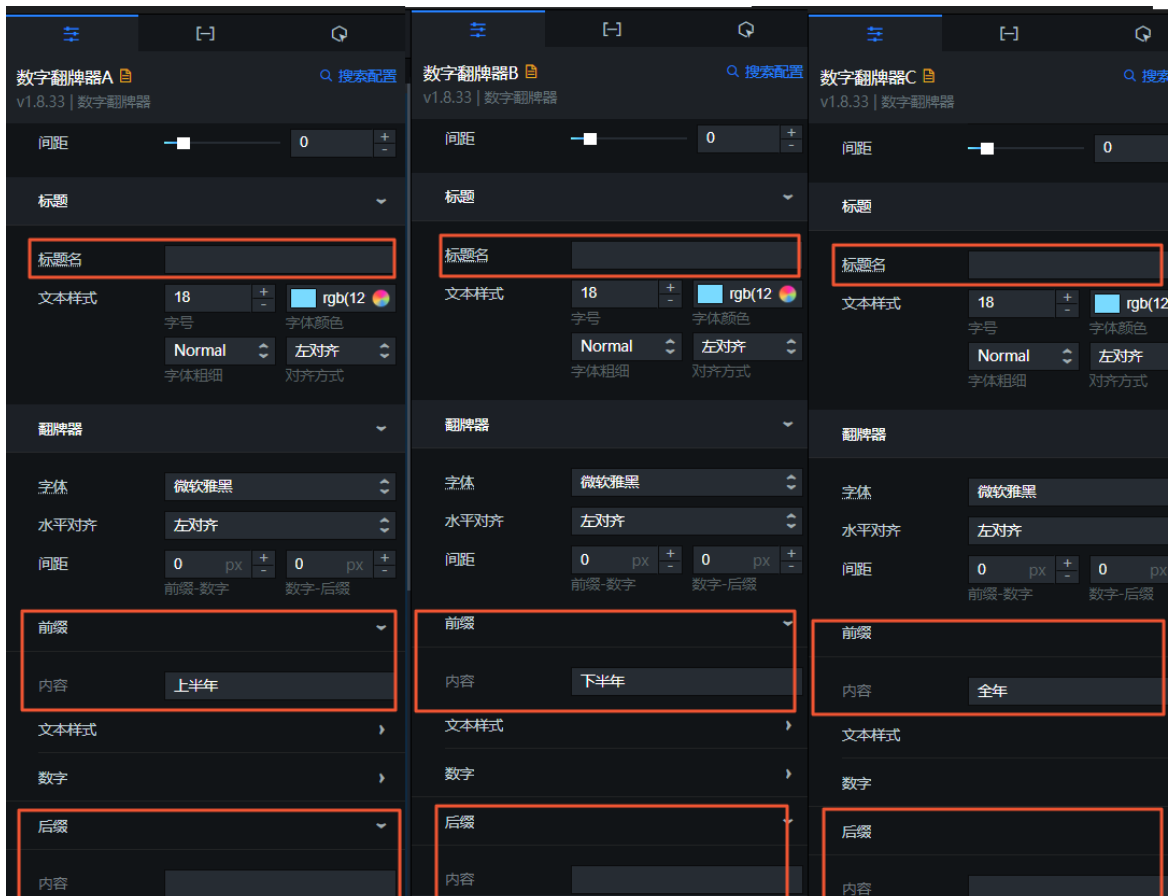
 **说明** 蓝图编辑器配置完成后需要在预览页查看配置效果。

### 配置案例交互

1. 参考画布编辑器 [添加资产](#)，在画布编辑器页面搭建所需要的数字翻牌器A、数字翻牌器B和数字翻牌器C组件。



2. 在数字翻牌器A、数字翻牌器B和数字翻牌器C的配置面板中，清空三者的标题名和后缀内容，并修改前缀内容分别为上半年、下半年和全年。



**注意** 此步骤为可选步骤，您也可以根据实际需求自定义数字翻牌器的标题名、后缀内容和前缀内容。

3. 在数字翻牌器A和数字翻牌器B组件的数据面板中，将数据源类型设置为动态数据类型，例如API、数


数据库等。

具体操作步骤请参见[配置资产数据](#)。最终画布上的组件结构样式如下图所示。



- 单击画布编辑器图层栏内三个数字翻牌器组件，右键选择导出到蓝图编辑器。



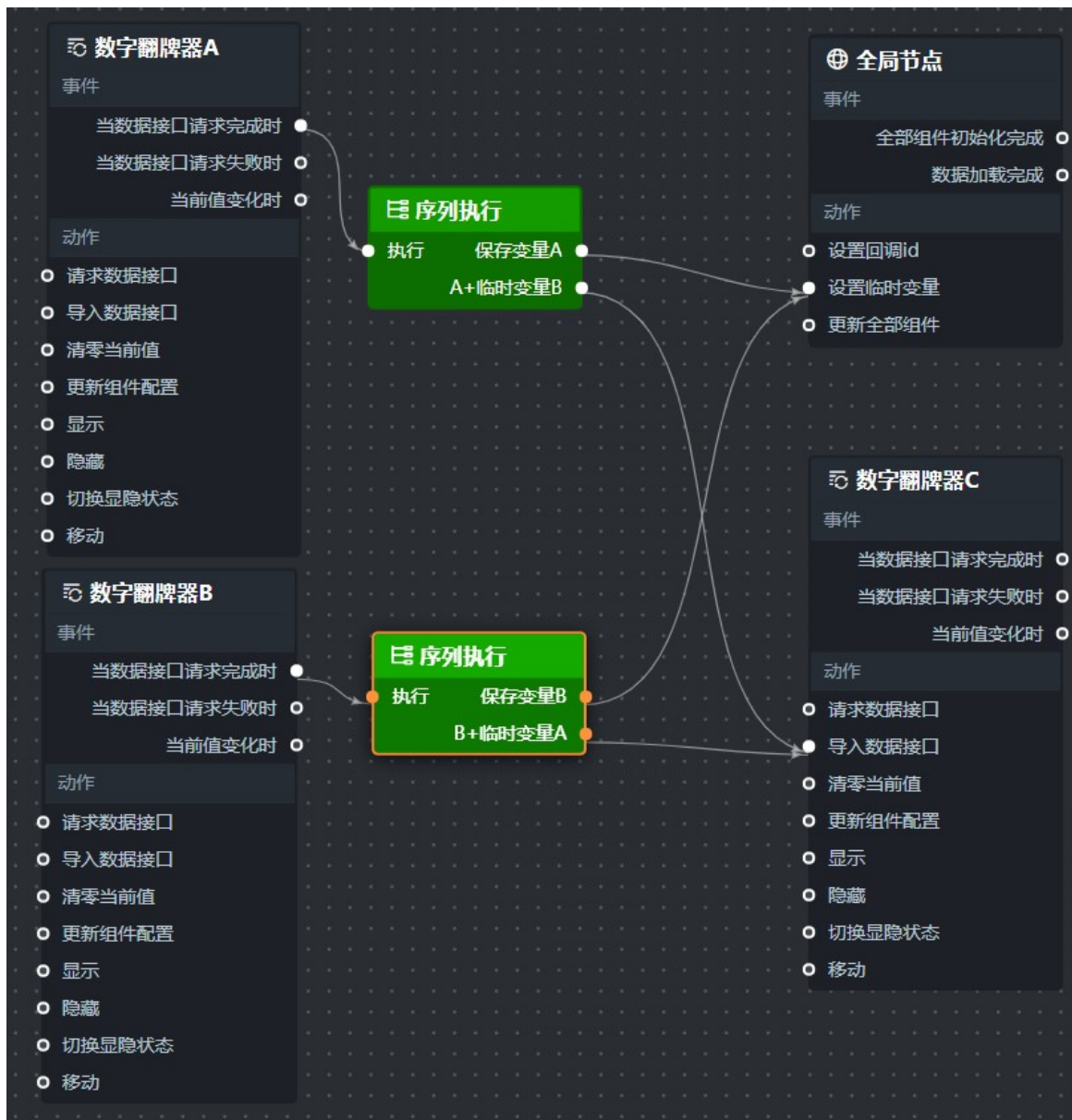
- 在画布编辑器页面左上角，单击[蓝图编辑器](#)图标（），切换到蓝图编辑器配置页面。
- 将左侧导入节点栏内的数字翻牌器A、数字翻牌器B、数字翻牌器C、全局节点和两个序列执行逻辑节点拖至画布。
- 分别将两个序列执行节点中配置面板中的处理方法重命名为保存变量A和保存变量B。  
具体操作方法请参见[序列执行节点](#)。
- 将数字翻牌器A的当数据接口请求完成时事件与对应的序列执行逻辑节点中的执行动作连线。  
同理数字翻牌器B也按照上述数字翻牌器A的方法连线。
- 将序列执行逻辑节点中的保存变量A与全局节点的设置临时变量动作连线。

同理保存变量B也按照上述保存变量A的方法连线。

10. 在画布中的两个序列执行逻辑节点的配置面板中，各自新增一个新的处理方法，分别命名为A + 临时变量B和B + 临时变量A。然后将这两个新增的处理方法都与数字翻牌器C的导入数据接口动作连线。

具体操作方法请参见[序列执行节点](#)。

最终连线的结果如下图所示。



11. 配置画布中序列执行内的保存变量A和A + 临时变量B的处理方法。
  - i. 单击与数字翻牌器A关联的序列执行逻辑节点，进入右侧配置面板，选择面板内其他配置栏下方的保存变量A处理方法，单击右侧下拉框打开脚本编辑区域。

- ii. 在脚本编辑区域，输入代码，完成后单击完成。

当前处理方法的示例代码如下，用于保存数字翻牌器A内每次请求到的最新数据。

```
return {
  data:[
    {
      name:"data_a", //变量名可自定义，不重名即可
      value:data[0].value || 0
    }
  ]
};
```

- iii. 使用同样的方式配置A + 临时变量B处理方法。

当前处理方法的示例代码如下，可实现将当前数字翻牌器A内请求到的数据与另外一份保存的数据相加后，导入数字翻牌器C的组件接口中。

```
let data_b = getLocalValue('data_b') || 0;
let res = data && data.length !== 0 ? data[0].value + data_b : data_b
console.log(data_b);
return [{ value: res }];
```

- 12. 使用同样的方式配置另一个序列执行逻辑节点内的保存变量B和B+ 临时变量A处理方法。

- o 保存变量B内的代码如下，用于保存数字翻牌器B内每次请求到的最新数据。

```
return {
  data:[
    {
      name:"data_b", //变量名可自定义，不重名即可
      value:data[0].value || 0
    }
  ]
};
```

- o B+ 临时变量A内代码如下，可实现将当前数字翻牌器B内请求到的数据与另外一份保存的数据相加后，导入数字翻牌器C的组件接口中。

```
let data_a = getLocalValue('data_a') || 0;
let res = data && data.length !== 0 ? data[0].value + data_a : data_a
console.log(data_a);
return [{ value: res }];
```

- 13. 处理方法都配置完成后，单击蓝图编辑器配置页面右上角的预览图标，预览交互效果。

