状态管理开发顺序与说明

为了确保良好的开发体验与模块间的依赖有序，推荐以下的**状态管理**编写顺序：

---

1. 先开发 CanvasState

1. **CanvasState**
   - 定义：记录画布偏移 (offset)、缩放 (scale) 等信息
   - 提供：`copyWith` 等更新方式
2. **CanvasStateNotifier** (或其他状态管理)
   - 实现 `setOffset(Offset)`, `setScale(double)`, `panBy(dx, dy)`, `zoomTo(...)` 等方法
   - 在 `canvas_interaction_manager` 中监听空白处的**拖拽/滚轮**事件，进而更新 CanvasState
3. **目的与价值**
   - 让画布平移/缩放逻辑先行稳定
   - 提供基础支撑，随后处理节点/连线时能准确定位坐标

---

2. 然后开发 NodeState

1. **NodeState**
   - 可包含 `List<NodeModel>` + 一些选中信息 (e.g. `Set<String> selectedNodeIds`)
   - 提供 CRUD 方法：`addNode`, `updateNode`, `deleteNode`
   - 可支持选中逻辑：`selectNode`, `toggleSelectNode` 等
2. **NodeStateNotifier**
   - 实现上述方法，以**不可变方式**更新 NodeState
   - 与 UI 或交互层 (node_interaction_manager) 配合，监听**节点拖拽、点击**等事件
3. **目的与价值**
   - 解决节点本身的增删改查；
   - 结合 CanvasState -> Node 在画布坐标中的移动/渲染就能完成基本功能
   - 选中机制(单选、多选)也可初步实现

---

3. 最后开发 EdgeState

1. **EdgeState**
   - 存储 `List<EdgeModel>` + （可选） `selectedEdgeId`
   - 处理：`addEdge`, `updateEdge`, `deleteEdge`, `selectEdge`
2. **EdgeStateNotifier**
   - 让交互逻辑（edge_interaction_manager）可以**拖拽连线**，创建/删除/更新 edge
   - 与 NodeState 协同：如果节点被移除/移动，需要同步更新连线
3. **目的与价值**
   - 在节点/画布都能正常运作后，连线将成为**完整**绘图/工作流编辑器关键组件
   - 拥有可管理的 EdgeState，可实现：
     - 拖拽锚点 => 建立或更新连线
     - 选中连线高亮 / 菜单 / 删除等

总体流程

1. **CanvasState** (管理画布偏移、缩放)
2. **NodeState** (管理节点增删改、选中)
3. **EdgeState** (管理边的增删改、连接关系)

依此顺序能先稳定基础 (画布)、再处理节点 (场景中最常见操作)，最后扩展到边 (较复杂的交互逻辑依赖节点)。完成这三大状态后，便可组合成一套**可操作可视化**的画布编辑器或流程图应用。