AnchorModel 文档

`AnchorModel` 用于描述某个节点上的“锚点”信息，使连线 (Edge) 能够准确地与节点对应的锚点进行连接。它包含锚点的相对位置、方向、可连接的类型、UI外观、协作锁等元数据。通过在节点上设置多个 `AnchorModel`，就能灵活配置节点可以连线的“端点”。

一、AnchorModel 基础字段

字段名	类型	说明
id	`String`	锚点的唯一标识 (如 `"anchor-1"`)
position	`Position`	锚点所在边：`left`、`right`、`top`、`bottom`
ratio	`double`	在对应边上的相对比例，取值 **0.0 ~ 1.0**，0表示左/上端，1表示右/下端
direction	`AnchorDirection`	锚点的连线方向： - `inOnly` (只能当目标) - `outOnly` (只能当源) - `inout` (既可做源也可做目标)
maxConnections	`int?`	允许的最大连接数 (如 `1` 代表只可连接1条线)，为空时不限制
acceptedEdgeTypes	`List<String>?`	只允许连接某些类型的 `Edge`（可空表示无限制）
shape	`AnchorShape`	锚点形状 (circle, diamond, square, custom)
arrowDirection	`ArrowDirection`	箭头方向(可选)：`inward` / `outward` / `none`

二、协作与锁定信息

字段名	类型	说明
locked	`bool`	是否锁定该锚点 (锁定后不允许编辑/移动)
version	`int`	版本号 (便于协作或Undo/Redo)
lockedByUser	`String?`	锁定者ID (多人协作场景下区分谁锁定了锚点)

三、UI / 交互相关信息 (可选)

字段名	类型	说明
plusButtonColorHex	`String?`	用于绘制 `Plus` 创建按钮的颜色（如自动生成连线或新节点时）

字段名	类型	说明
plusButtonSize	`double?`	`Plus` 按钮图标大小
iconName	`String?`	如果需要在UI渲染锚点图标，可在这里指定图标名称 (由业务/插件决定如何解析)
data	`Map<String, dynamic>`	存储任意自定义数据（脚本、统计、权限标记等）

四、约束与自动修正

Ratio 自动 clamp

```
ratio = (ratio < 0.0 ? 0.0 : (ratio > 1.0 ? 1.0 : ratio));
```

若构造时传入的 ratio < 0.0 会自动设为 0.0，>1.0 则设为 1.0。
同理，从 JSON 中解析 ratio 字段后也进行 clamp。

五、关键方法

(1) `copyWith`

不可变更新方法，创建并返回新的锚点实例：

```
final updatedAnchor = anchor.copyWith(ratio: 0.7);
```

(2) `toJson` & `fromJson`

序列化与反序列化，支持网络传输或数据存储:

```
final json = anchor.toJson();
final anchor = AnchorModel.fromJson(json);
```

(3) `getAnchorOffset`

获取指定锚点相对于节点中心的偏移量：

```
Offset getAnchorOffset(AnchorModel anchor) {
    final ratio = anchor.ratio.clamp(0.0, 1.0);
    final offset = anchor.position.getOffset(width, height);
    return Offset(x + offset.dx, y + offset.dy);
}
```

(4) `AnchorModel` 的构造方法

```
final anchor = AnchorModel(
  id: 'anchor-1',
  position: Position.left,
  ratio: 0.5,
);
```

六、使用场景

1. 锚点位置更新

```
final updatedAnchor = anchor.copyWith(ratio: 0.7);
```

2. 锚点形状更新

```
final updatedAnchor = anchor.copyWith(shape: AnchorShape.diamond);
```

3. 锚点连线方向更新

```
final updatedAnchor = anchor.copyWith(arrowDirection:
ArrowDirection.outward);
```

4. 锚点协作锁定

```
final updatedAnchor = anchor.copyWith(locked: true);
```

5. 锚点数据更新

```
final updatedAnchor = anchor.copyWith(data: {'key': 'value'});
```

6. 锚点UI样式更新

```
final updatedAnchor = anchor.copyWith(style: AnchorStyle(colorHex:
'#00FF00'));
```

七、示例 JSON

```
{
  "id": "anchor-1",
  "position": "left",
  "ratio": 0.5,
  "direction": "inout",
  "maxConnections": 1,
  "acceptedEdgeTypes": ["flow"],
  "shape": "diamond",
  "arrowDirection": "outward",
  "locked": false,
  "version": 1,
  "lockedByUser": null,
  "plusButtonColorHex": "#FF0000",
  "plusButtonSize": 16.0,
  "iconName": "custom_icon",
  "data": {"key": "value"}
}
```

八、单元测试建议

- 测试构造方法默认值与边界值
- 测试copyWith方法的不可变性
- 测试JSON序列化与反序列化是否一致
- 测试getAnchorOffset返回准确的坐标值

九、总结

AnchorModel 以不可变数据结构实现，便于扩展、维护及序列化。通过合理设计状态与数据的边界，能够很好地适用于复杂工作流或绘图应用。