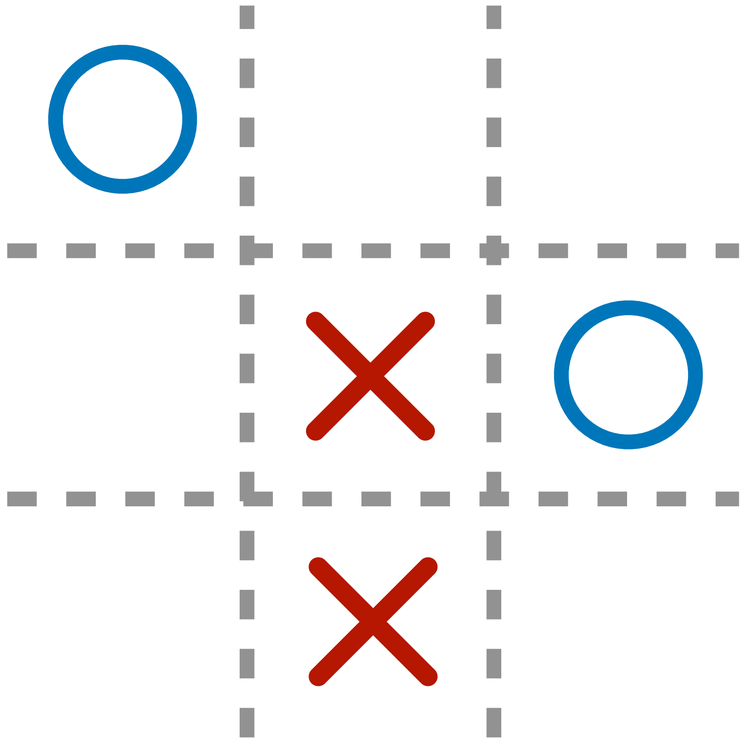
React实战

井字棋



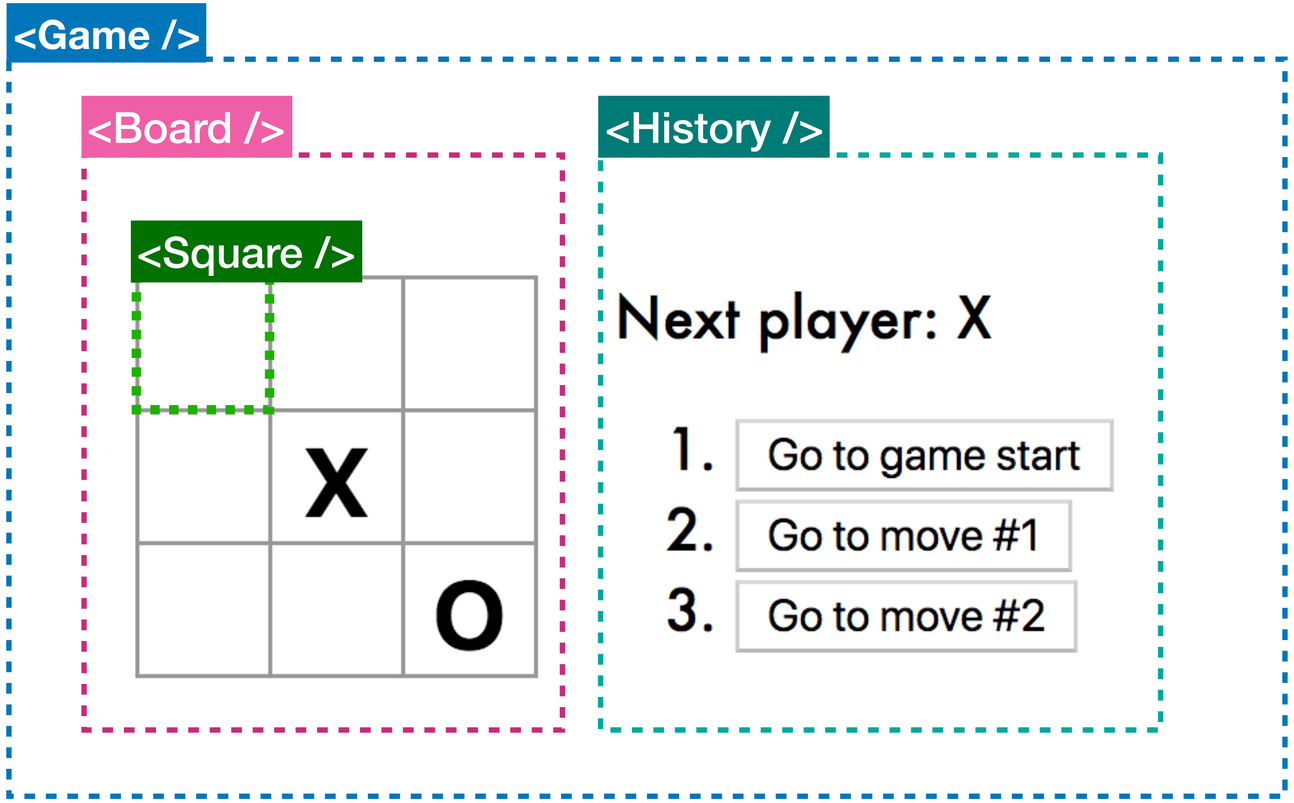
Skipper

项目大纲

Step 1	完成组件构建和基础样式
Step 2	游戏逻辑和数据维护
Step 3	状态提升
Step 4	历史记录数据和跳转
Step 5	官方推荐练习

Step 1 - 完成组件构建和样式

项目组件拆分



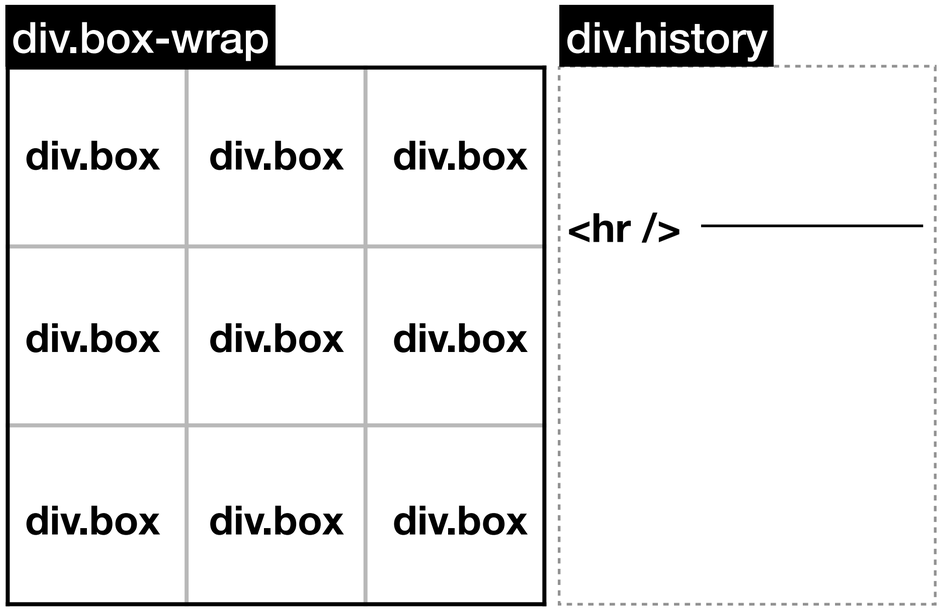
组件名称	作用
Game	整体容器，维护整体状态
Board	游戏面板，提供游戏区域
Square	游戏格子，游戏状态呈现的容器
History	游戏历史控制组件

样式处理

**Board => **

**Square => **

**History => **



div为block元素，无法排在一行，需要使用float实现

.box需要设置float，.box-wrap和.history之间也需要

.box设置为100px * 100px，并且设置1px的边框

.box-wrap设置为300px * 300px，并且设置1px的边框

Step 2 - 游戏逻辑和数据维护

第一次游戏数据设计和维护处理思路

*尚未处理历史记录功能

使用一维数组存储游戏

数据	null	null	null	null	“X”	null	null	null	“O”
下标	0	1	2	3	4	5	6	7	8

对应

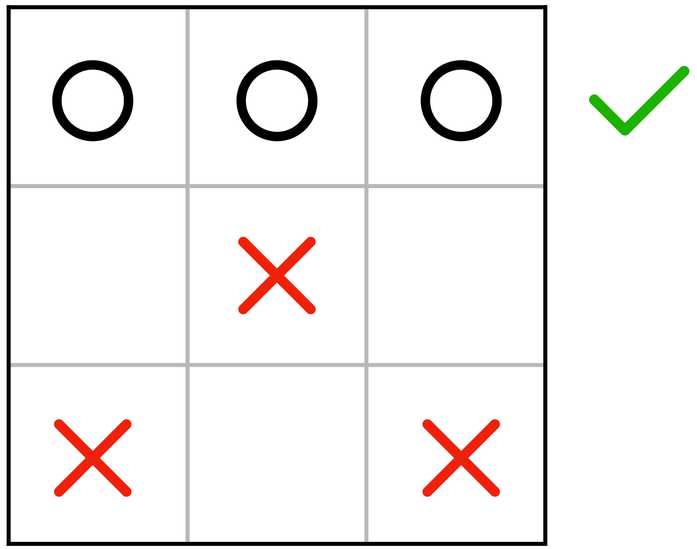
0	1	2
3	X	5
6	7	O

1. 游戏初始化为全为null的一维数组

2. 根据上图所示将数据映射为 3x3 的游戏界面

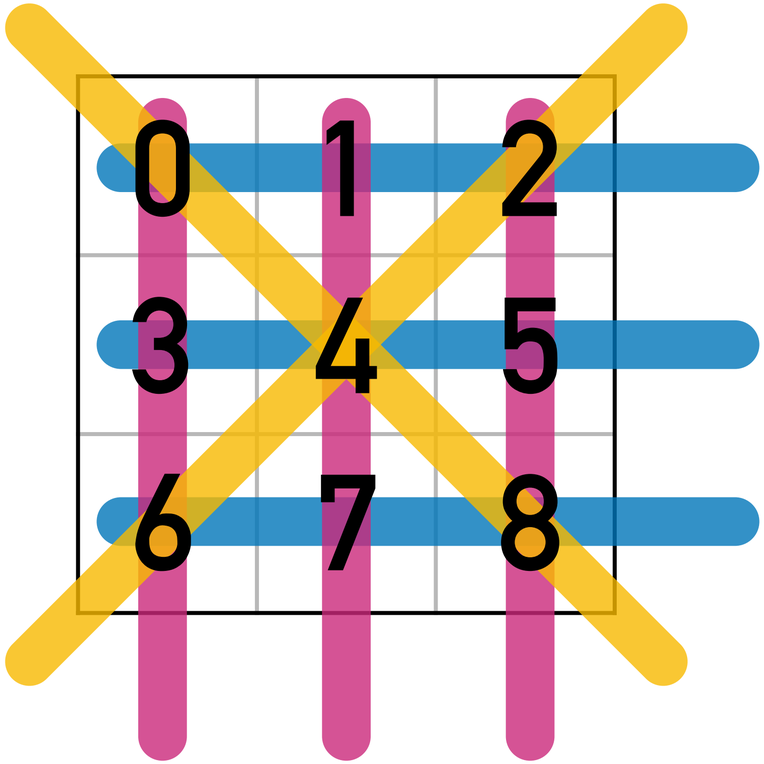
胜利判断

井字棋游戏胜利结果比较简单，可以直接穷举处理



如果某一行/某一列/某对角线全为相同

则可以判断胜利



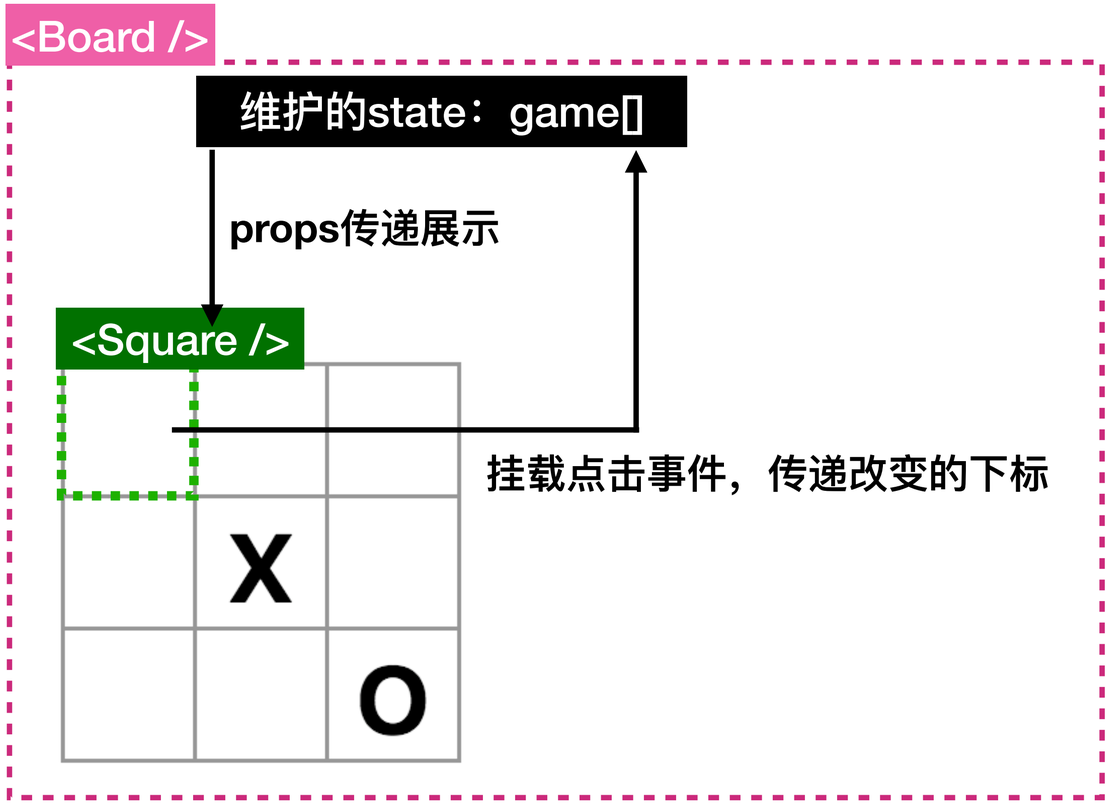
横向成功条件	竖向成功条件	对角线条件
0，1，2	0，3，6	0，4，8
3，4，5	1，4，7	2，4，6
6，7，8	2，5，8	

```
game = [
    “O” , ”O” , ”O”,
    null , ”X” , null,
    ”O” , null , ”O”
]
```

以竖向成功条件0，1，2为例

```
if(
    game[0] === game[1] && game[1] === game[2]
){
    // 成功条件
}
```

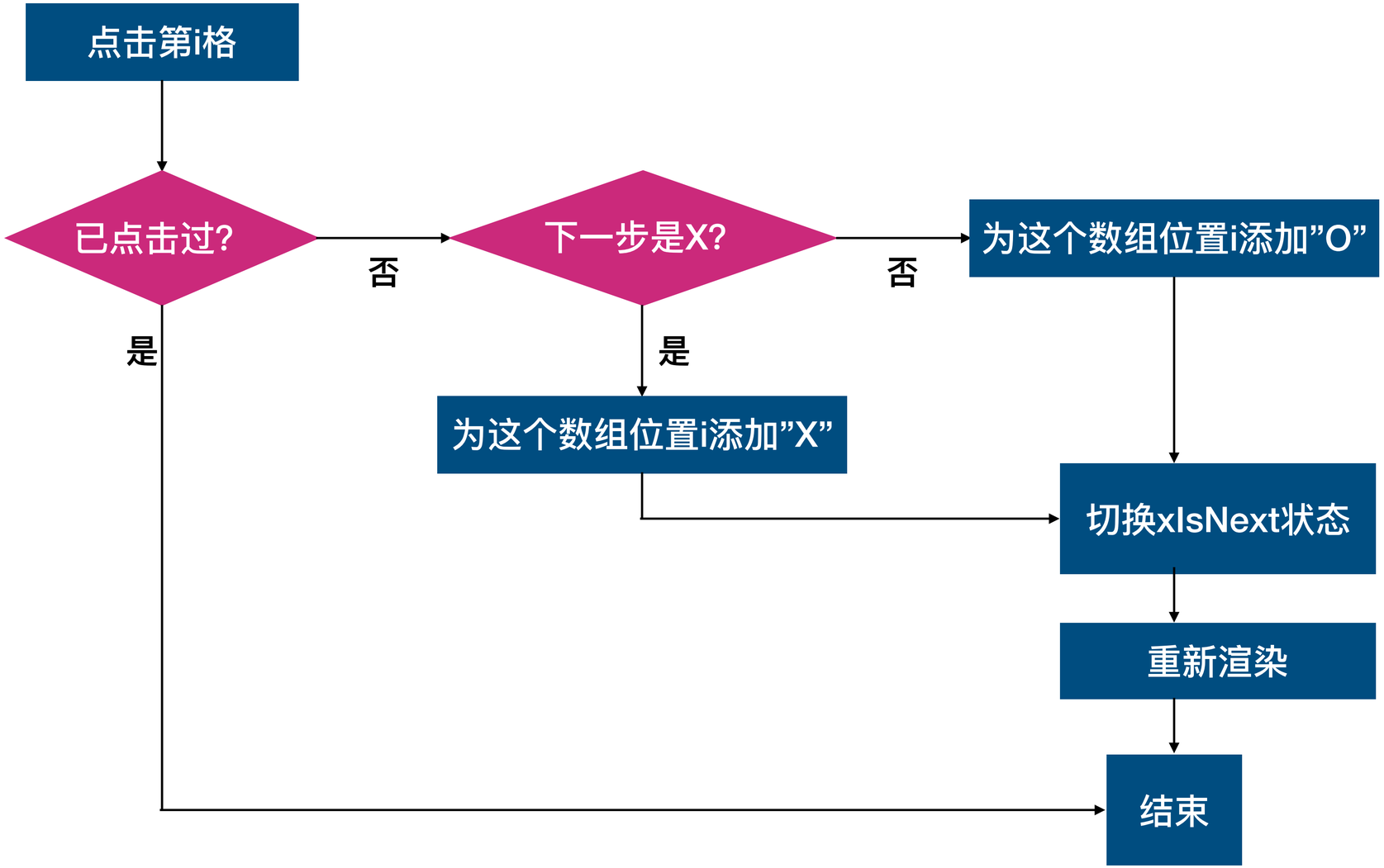
组件和逻辑整体规划



在Board组件中维护如下状态和方法

状态	说明
game	游戏保存
xIsNext	下一步是否为“X”玩家进行
方法	**说明**
handleClick(i)	处理点击事件，要求传入点击的位置
renderSquare(i)	用于创建小方格Square组件

逻辑流程图

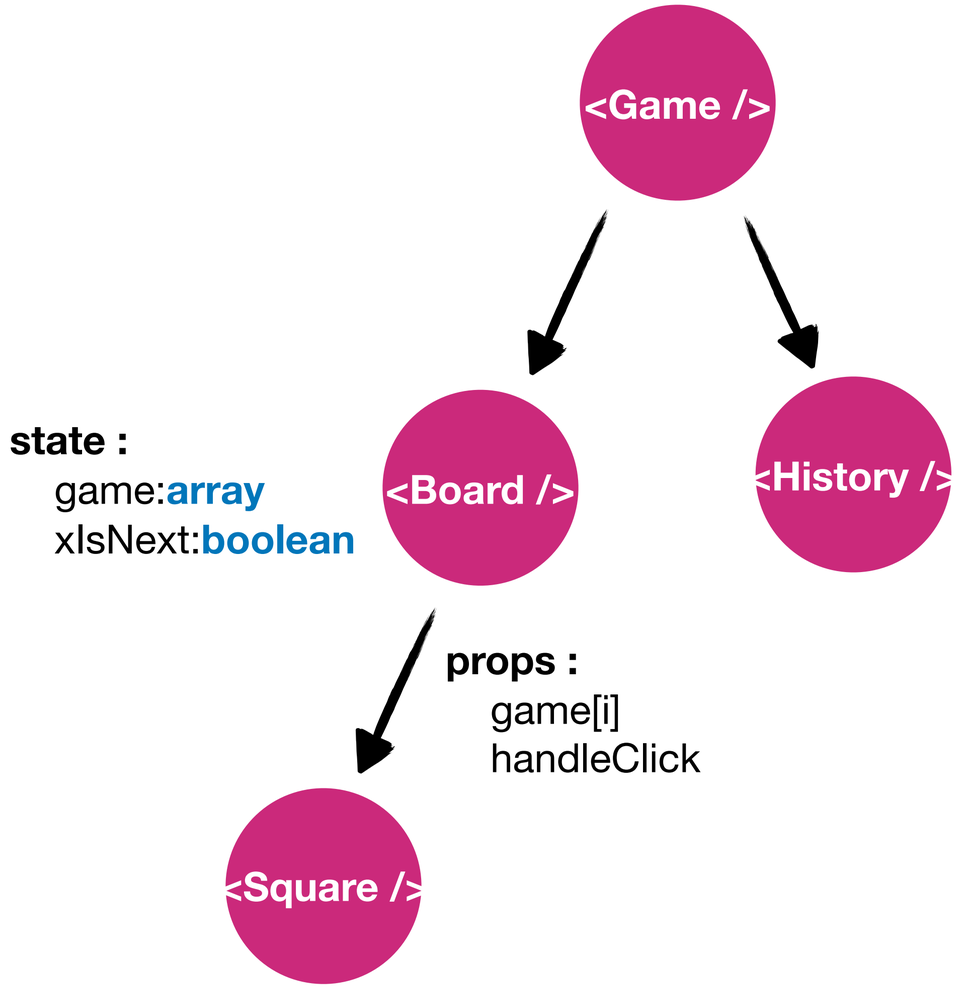


基本逻辑和解决方案

基本逻辑说明	解决方法	代码示范
如果某格下过了，则不能再下	点击时先检查改位置是否为空	if(game[i]) return;
根据游戏状态确定现在是哪个玩家在进行	维护状态xIsNext	if(xIsNext) game[i] = ‘X’
一步游戏完成后，自动更换下一个玩家	维护状态xIsNext	xIsNext = !xIsNext

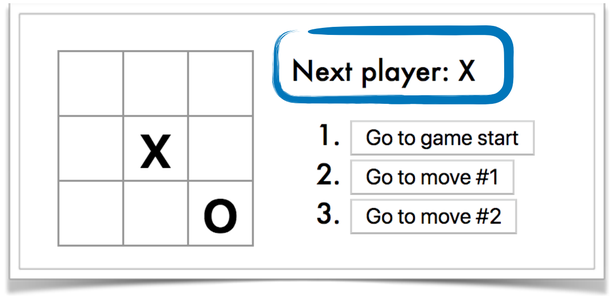
Step3 - 状态提升

组件树分析

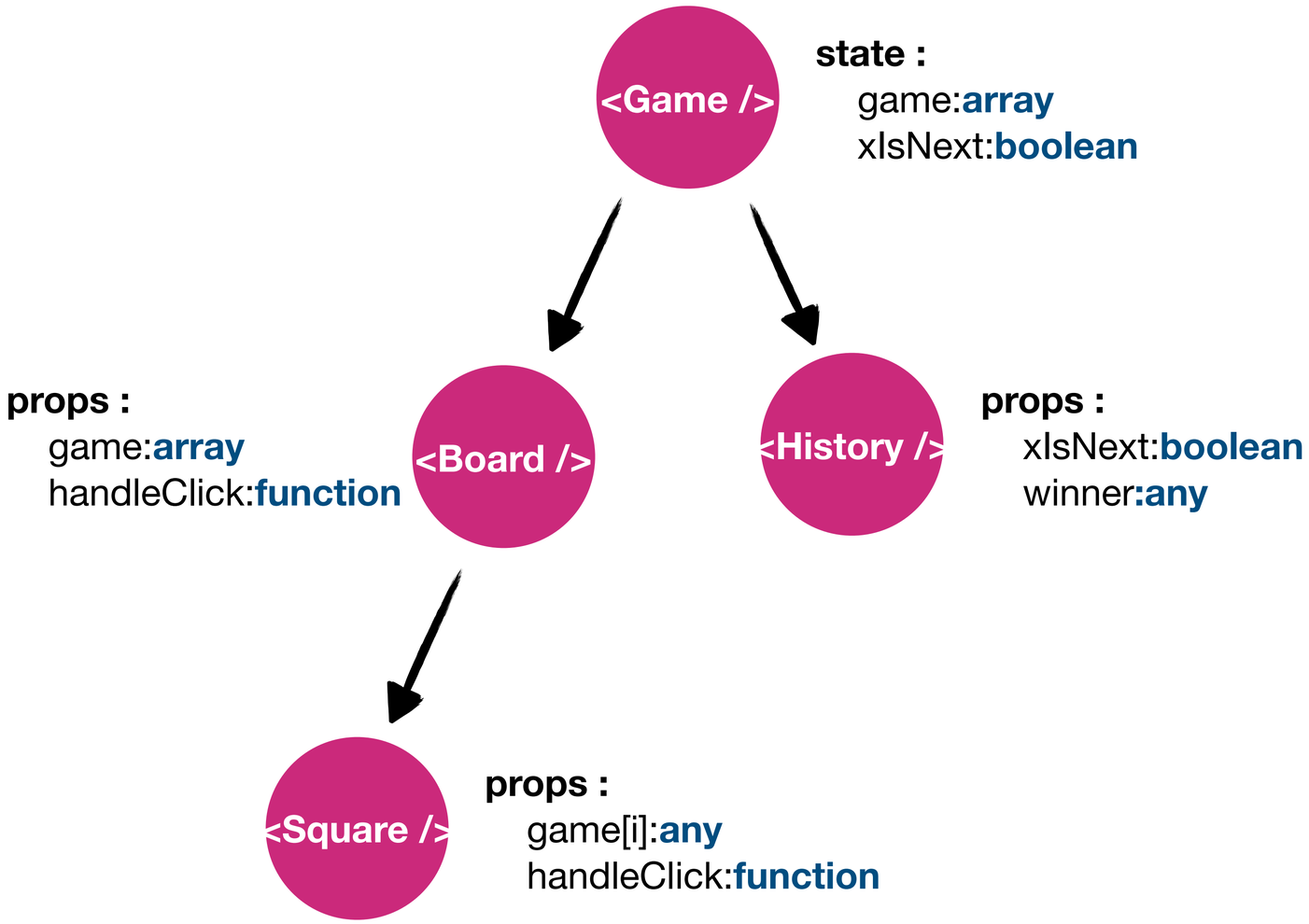


发现问题：

History组件也需要xIsNext状态

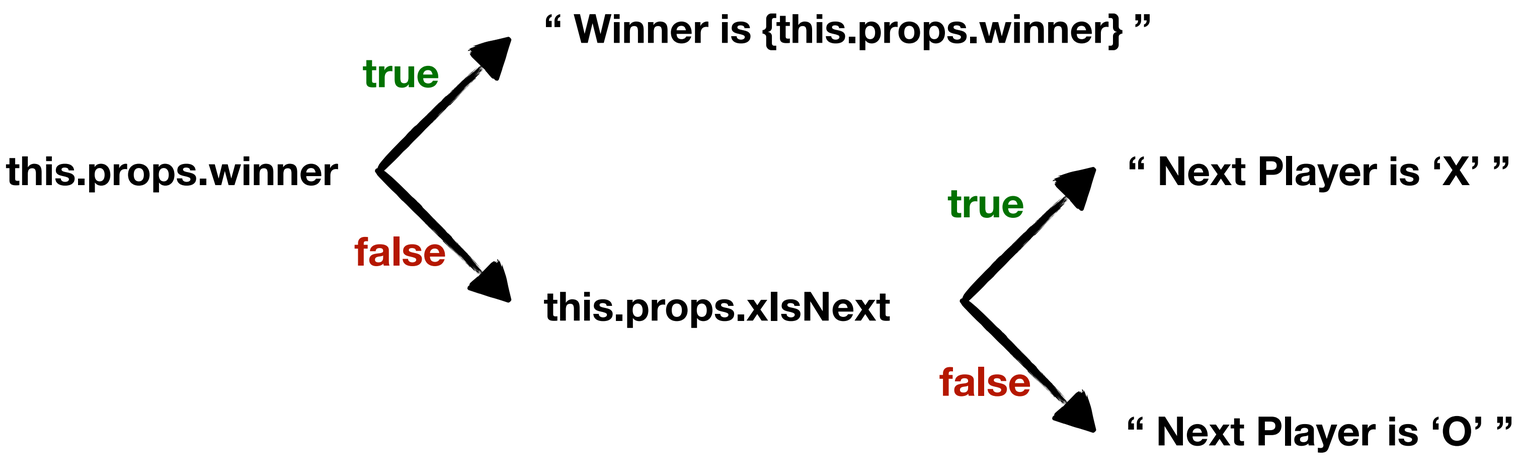


状态提升至根组件<Game />



渲染游戏步骤

<History />

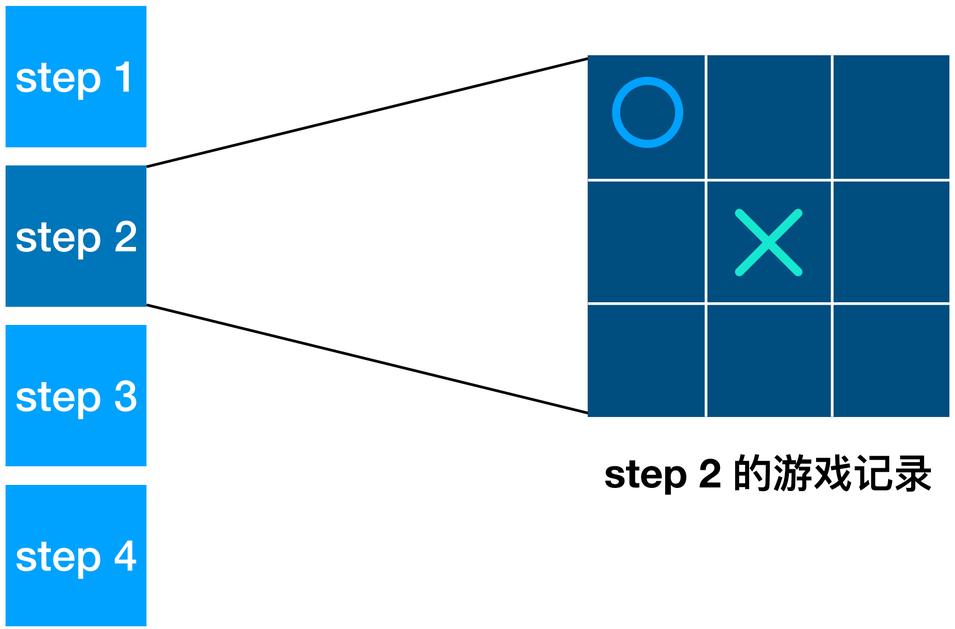


winner状态无需专门维护，传递时计算即可

Step4 - 历史记录数据和跳转

历史记录数据结构说明

history : Array

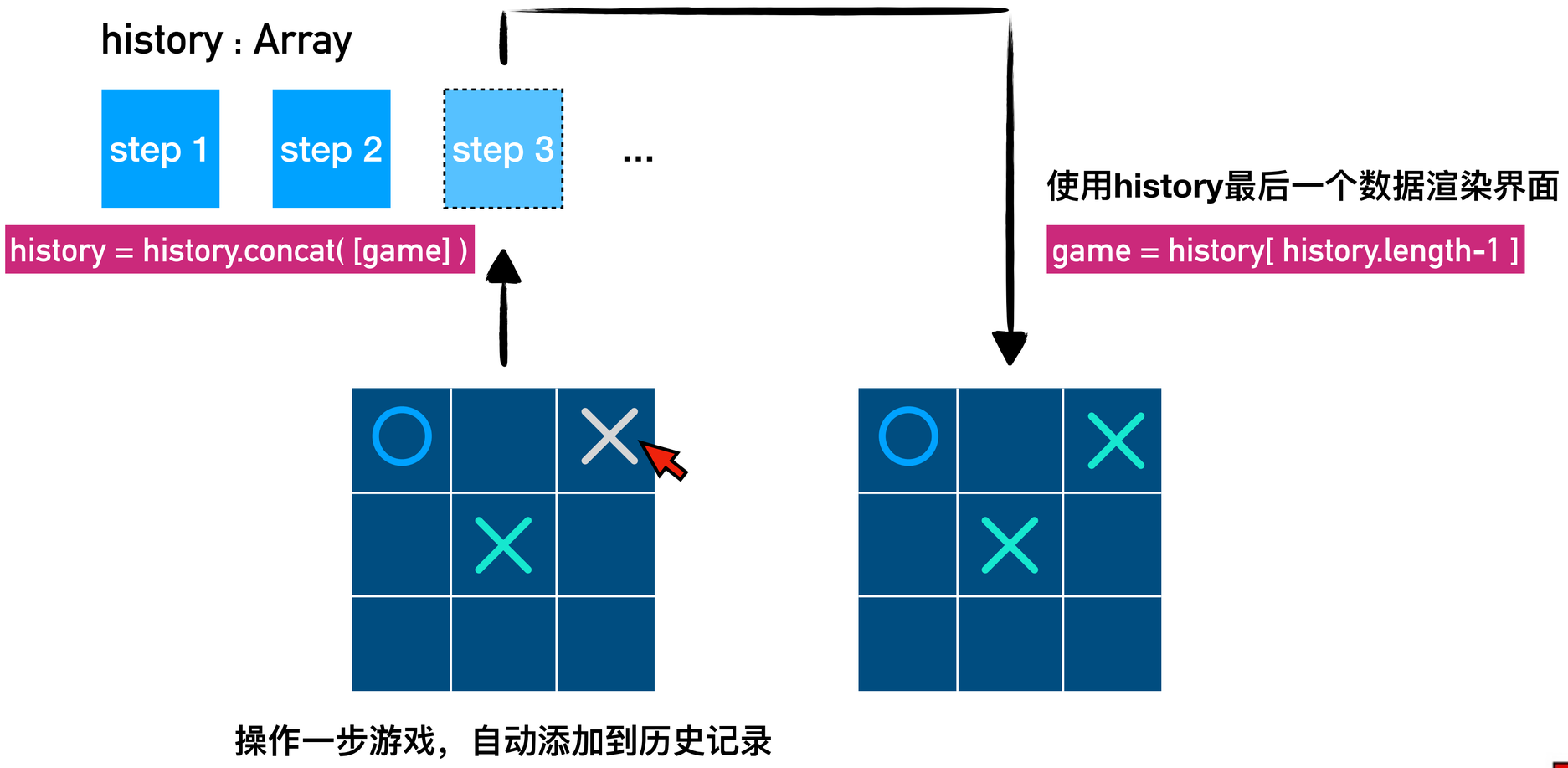


step 2 的游戏记录

使用一个数组history来存储游戏记录，
数组里面的每一项为一步游戏记录

history = [Array(9).fill(null)]

历史记录操作



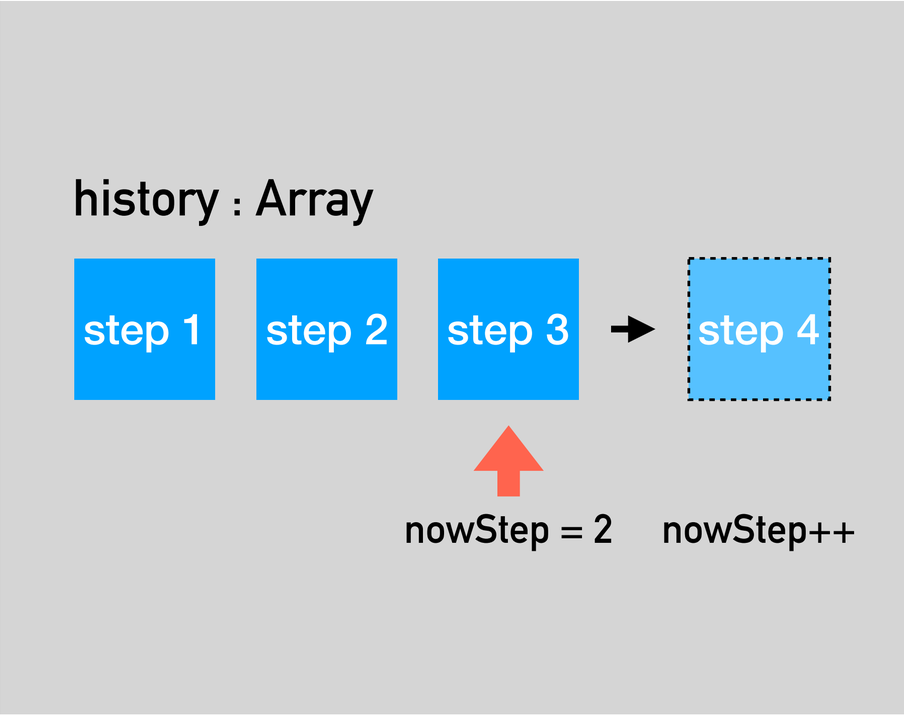
操作一步游戏，自动添加到历史记录

历史记录渲染

需要注意：history状态是在Game组件维护的，需要传递给History组件渲染

历史记录跳转

为了实现跳转，维护nowStep状态，表示当前处于哪一步



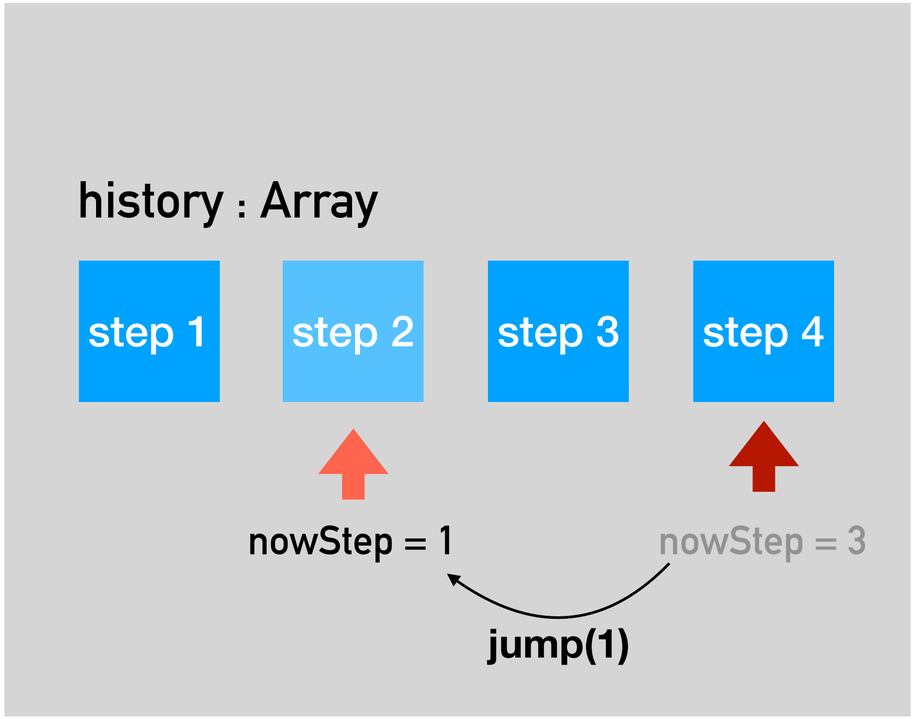
在**正常状态**中，nowStep为历史记录最后一个数据

```
nowStep = history.length - 1
```

别忘了，实时渲染不再应该使用history最后一项了，

而是应该根据nowStep确定

```
game = history[nowStep]
```



在**跳转状态**中，nowStep为跳转赋予的值，

我们要解决几个问题。

1. **跳转后，下一步（xIsNext）是“X”还是“O”?**
2. **跳转后再下下一步棋，历史记录应该如何处理?**
3. **执行后，nowStep应该改变吗?**

Q1：跳转后下一步是“X”还是“O”?

我们是根据xIsNext来判断下一步进行的，

如果我们初始化xIsNext为true，则第一步为“X”

如果初始化为false，则第一步为“O”

以第一步为X为例：

history : Array

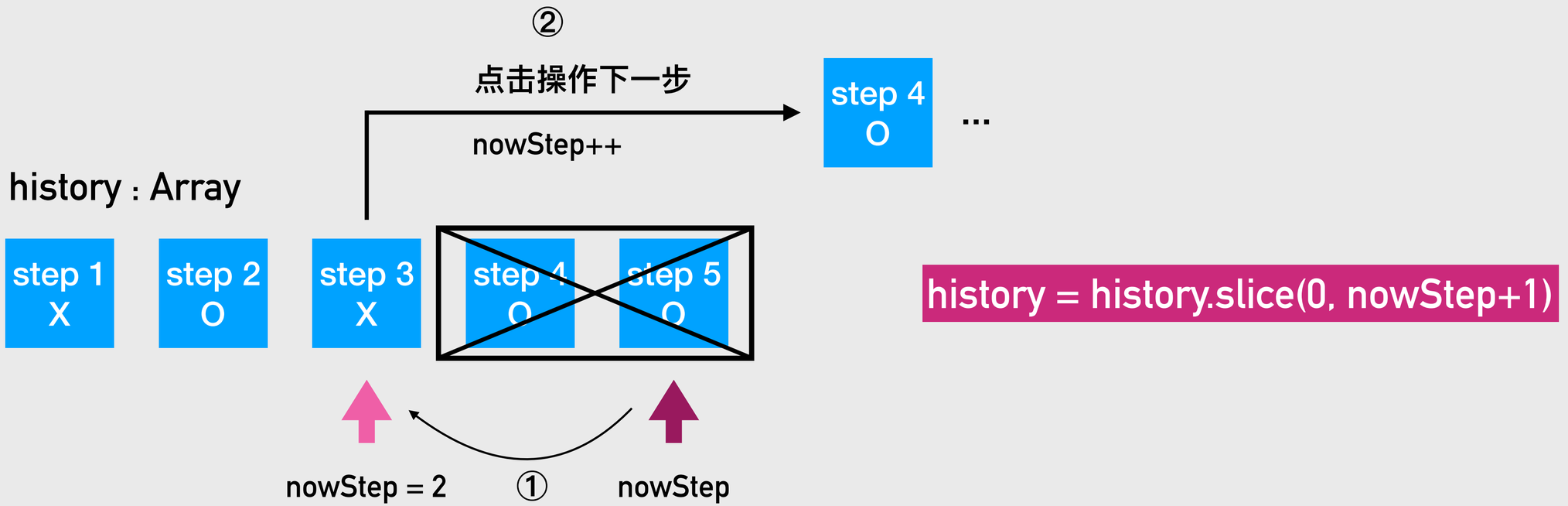
step 1	step 2	step 3	step 4
null	X	O	X

如果nowStep跳转到偶数则下一步为X

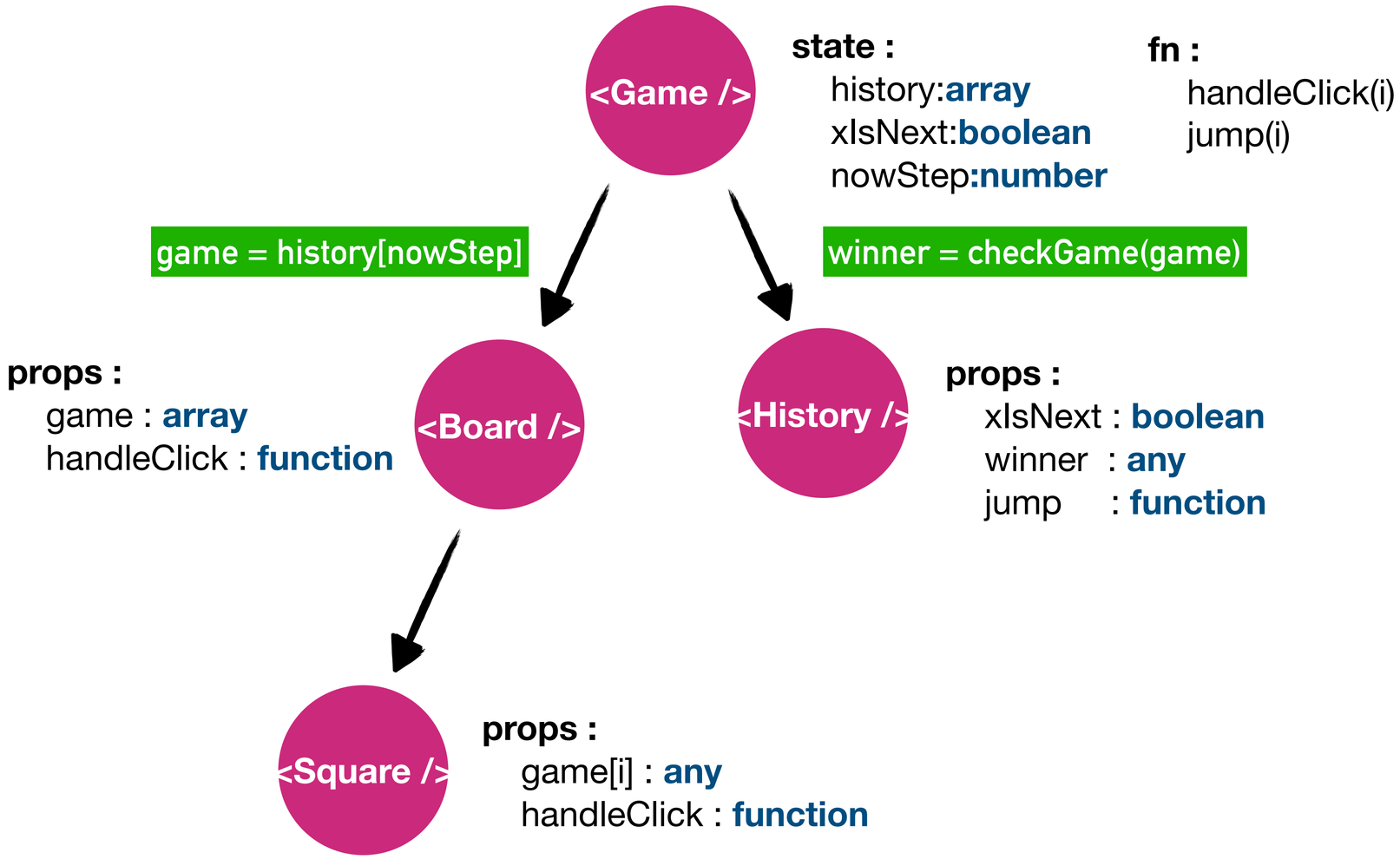
```
if( nowStep % 2 === 0 ){
    xIsNext = false
}
```

Q2：跳转后再操作，历史记录如何处理?

应当清除nowStep后的历史记录，添加新记录



最后组件树整理



Step 5 - 官方推荐练习

练习
Exercise

1. 以二维坐标表示位置，而不是一维。
2. 在history列表里高亮显示当前选中的一步。
3. 在Board组件用两个循环渲染出9个Square。
4. 添加一个正序/倒序按钮改变历史记录排序。
5. 当一方获胜的时候，高亮展示连城一线的3颗棋子。

一些解决方案

问题 2	在history列表里高亮显示当前选中的一步。	添加一个渲染判断条件
问题 4	添加一个正序/倒序按钮改变历史记录排序。	在History组件维护状态 isPositiveOrder 根据状态渲染历史记录
问题 5	当一方获胜的时候，高亮展示连城一线的3颗棋子。	修改checkGame函数，获胜则返回获胜的下标 再根据下标渲染高亮棋子