**Tutorial: Tic-Tac-Toe**

You will build a small tic-tac-toe game during this tutorial. This tutorial does not assume any existing React knowledge. The techniques you’ll learn in the tutorial are fundamental to building any React app, and fully understanding it will give you a deep understanding of React.

=>

Khi bạn muốn build 1 game tic-tac-toe trong hướng dẫn này. Nó ko assume bất cứ kiến thức React. Các kĩ thuật bạn sẽ học trong hướng dẫn này nền móng build mọi app React và hiểu đầy đủ về nó khi đó nó giúp bạn sẽ hiểu sâu hơn vể React

Note : hướng dẫn này đc designerd ( thiết kế ) cho những người thích học từ thực hành và muốn làm nhanh 1 thứ gì đó . Nếu bạn thích học từng bước từng bước thì hãy học Describing the UI

Cái hướng dẫn được chia thành 1 số phần (seveval sections )

* **Setup for the tutorial** ( cài đặt theo hướng dẫn ) sẽ giúp bạn điểm khởi đầu ( a starting point) để làm theo hướng dẫn ( to follow the tutorial.
* Overview ( tổng quan ) sẽ dạy bạn những cái cơ bản về react: components , props and state
* Completing the game ( Hoàn thành game ) sẽ dạy bạn 1 số kĩ thuật phổ biến nhất (the most common tech) trong phát triển phần mềm
* Thời gian cuối hành trình sẽ giúp bạn **có cái nhìn** ( a deeper insight) về những điểm mạnh của react

### What are you building?

Trong tutorial này,. Bạn sẽ xây dựng game tictactoe interactive ( tương tác ) with react

Nếu code chưa có ý nghĩa với bạn, hoặc nếu bạn chưa quen với syntac, đừng lo. Mục đích của hướng dẫn này là giúp bạn hiểu rect và syntax

Chúng tôi gợi ý rằng bạn hãy check out game trc khi đi tục với tutotial này. 1 trong những features( đặc trưng) bạn chú ý là list number ở bên phải board game (bảng game) . list này giúp bạn có lịch sử của tất cả di chuyển các sự kiện xảy ra trong game. Và nó update khi game xử lí

Sau khi người chơi hoàn thành game , hãy tiếp tục cuộn. Bạn sẽ bắt đầu với simpler template (mẫu đơn giản ) trong hướng dẫn này. Bước tiếp theo của chúng tôi là thiết lập cho bạn để bắt đầu building game

## Setup for the tutorial

Trong trình live code bên dưới, click vào Fork trong top-right corner để mở newtab và sử dụng codeSandbox… nó cho phép bạn viết code trong trình duyệt và preview và xem app bạn created. New tabn hiển thi empty squate và bắt đầu code hướng dẫn

## Overview

Now that you’re set up, let’s get an overview of React!

### Inspecting the starter code

1. File section có list file . app.js , index.js, style.css và folder called public
2. The code editor nơi bạn nhìn thấy mã code của seclected file
3. The browser section nơi bạn nhìn thấy code bạn viết hiển thi như nào

export default function Square() {

return <button className="square">X</button>;

}

R

Đ

The browser section hiển thị square with a X trong nó

Ngay bây h hay nhìn vào filse và bắt đầu code

Tạo component, trong react component là 1 phần của mã tái sử dụng ( reusable code) đại diện 1 phần giao diện người dung. Components được sử dụng để render, manage, update the UI element trong app. Hãy nhìn vào component từng dòng từng dopngf để thấy chuyện gì đang xảy ra.

export default function Square() {

return <button className="square">X</button>;

}

Dòng đầu tiên định nghĩa 1 function called Square . The export JS keyword làm cho nó có thể sử dụng ngoài file này. The default keyword đê tells với cái file other khác sử dụng code của bạ, nó là chức năng chính của function trong file.

Dòng thứ 2 return 1 button, the return là 1 keyword có nghĩa là nó trả về giá trình value khi call function này. Button là 1 JSX element. 1 JSX element là 1 combination ( sự kết hợp ) giáu JS và HTML tags dể mô tả những gì bạn muốn hiển thị. ClassName square là 1 buttuon property hoặc prop để css tạo style như nào cho nút.

Style.css

Clicj vào file style.css trong file section . nó defines the style cho react app, Đầu tiền tow CSS selector (\* and body ) definse the style phần lớn của app. Trong khi .square selector defines the style cho chỉ 1 component nơi mà classNAme property đc set square. in code. Nó sẽ would match với button square component .

Index.js

Click vào file labeled index.js in the Files section of codeSandbox. Bạn sẽ không chỉnh sửa file này trong suốt quá tình hướng dẫn nhưng nó là 1 cầu nối giữa component bạn tạo app và web browser

Lines 1-5 brings all the necessary pieces together:

* React
* React’s library to talk to web browsers (React DOM)
* the styles for your components
* the component you created in App.js.

The remainder of the file brings all the pieces together and injects the final product into index.html in the public folder

Phần con lại của file tập hợp tất cả các piếc với nhau vào injects vào sản phẩm cuối cùng into index.html trong 1 folder public

### Building the board

Hãy get back( trở lại ) nơi mà bạn sẽ dùng nhiều trong hướng dẫn

Hiện tại trong bảng chỉ có 1 hình vuông , nhưng bạn cần 9, nếu bạn thử copy paste square thành two square :

export default function Square() {

return <button className="square">X</button><button className="square">X</button>;

}

Nó sẽ lỗi :

React components need to return a single JSX element and not multiple adjacent JSX elements like two buttons. To fix this you can use fragments (<> and </>) to wrap multiple adjacent JSX elements like this:  
nó return single JSX chứ ko multiple adjacent JSX element. Bạn them fragments (<> and </>) nó thì sẽ đc

export default function Square() {

return (

<>

<button className="square">X</button>

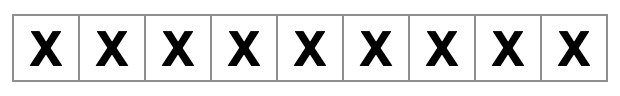
<button className="square">X</button>

</>

);

}

Great! Now you just need to copy-paste a few times to add nine squares and…



Oh no! The squares are all in a single line, not in a grid like you need for our board. To fix this you’ll need to group your squares into rows with divs and add some CSS classes. While you’re at it, you’ll give each square a number to make sure you know where each square is displayed.

Thật tuyệt , bây h bạn cần copy paste để có 9 hình vuông

Các hình vuông all in a singke line, nó ko không phải dạng grid( lưới ) như bạn cần. Để fix nó sẽ cần group your square into rows với thẻ div và add thêm 1 số CSS classes . Khi đó bạn sẽ give nó có 1 số number để nó biết bnos ở đâu khi hiển thị

export default function Square() {

return (

<>

<div className="board-row">

<button className="square">1</button>

<button className="square">2</button>

<button className="square">3</button>

</div>

<div className="board-row">

<button className="square">4</button>

<button className="square">5</button>

<button className="square">6</button>

</div>

<div className="board-row">

<button className="square">7</button>

<button className="square">8</button>

<button className="square">9</button>

</div>

</>

);

}

CSS defined trong styles.css the divs with the className of board-row. Ngay bây giờ bạn có các grouped các components into rows với the styled div bạn có bảng tic-tac-toe.

Nhưng bạn có 1 vấn dề, component square của bạn sẽ ko còn là square nữa. hay đổi tên thành board

### Passing data through props

Truyền dữ liệu cho props.

Tiếp theo bạn sẽ muốn thay đôi value của square từ empty thành X khi người dùng click vao square. bạn xây dựng board chỉ cần copy – paste code. Bạn cần updates 9 lần, mỗi lần cho 1 hình vuông bạn có. React cho phép tái sử dụng code đẻ cho nó ko lộn xộn và lặp code.

Ko giống của các browser thì nó đc viết bắt đầu bằng chữ in hoa ( Square, Board)

Oh no! You lost the numbered squares you had before. Now each square says “1”. To fix this, you will use props to pass the value each square should have from the parent component (Board) to its child (Square).

Update the Square component to read the value prop that you’ll pass from the Board:

No, nó đã mất các number trước đó, ngay abya h mõi square says 1, để fix nó bạn cần use props chuyển giá trình mỗi ô vuông thành từ parent component ( board ) tới child (Square)

Update the Square component to read the value prop that you’ll pass from the Board:

Nâng cấp square component đc value prop bạn sẽ pass from the Board

function Square({ value }) {

return <button className="square">1</button>;

}

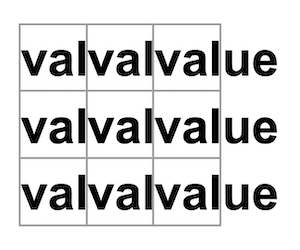
Function Square( {value}) chỉ ra các Square component cần passed a prop called value ( được thông qua giá trị). Now bạn muốn hiên thị value thay vì 1 trong mọi square .

function Square({ value }) {

return <button className="square">value</button>;

}

Oops, this is not what you wanted:



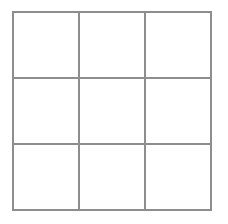
Bạn muốn render JS variable đc gọi value từ component của mình , nó ko phải là “value “ . Bạn cần suwer dụng dấu ngược nhọm để đc.

function Square({ value }) {

return <button className="square">{value}</button>;

}

For now, you should see an empty board:



H nó đăng là e empty board .

Nó là vậy vì board component ko thể tự passed the value prop to echt Square component it renders yet. Để fix nó bạn hay add the value prop to ecth Square component rendered by the board component.

export default function Board() {

return (

<>

<div className="board-row">

<Square value="1" />

<Square value="2" />

<Square value="3" />

</div>

<div className="board-row">

<Square value="4" />

<Square value="5" />

<Square value="6" />

</div>

<div className="board-row">

<Square value="7" />

<Square value="8" />

<Square value="9" />

</div>

</>

);

}

H bạn thấy 1 lưới number

### Making an interactive component

Hãy điền vào square component với X khi bạn click vào nó, Khai bảo 1 chức năng là hanleClick trong Square, Khi đó add thêm onclick vào props chon nuts button JSX element returned from the Square

function Square({ value }) {

function handleClick() {

console.log('clicked!');

}

return (

<button

className="square"

onClick={handleClick}

>

{value}

</button>

);

}

Nếu bạn clickvaof 1 square bây h thì bạn sẽ nhận đc chữ clicked ở console tab trong browser section . Click the square more than once will log “clicked!” again.

Bước tiếp theo, bạn muốn Square component to “remember” thạt it got clicked fill it with X mark ( ghi nhớ click và phủ đầy nó bỏi X mark ). Bạn hãy dùng use state

React cung cấp cho bạn function đặc biết là useState , bạn cần call nó từ component nó sẽ ghi nhớ mọi thứ. Hãy hãy lưu giá trị hiện tại của Square vào state và change nó khi bạn click square .

Import useState vào đầu file , xóa bỏ giá trị prop từ square component. Thay add thêm 1 dòng mới khi bắt đàu square khi call useState , nó trả lại state variable called value.

Value : Giá trị lưu trữ và setValue là 1 function có thể thay đổi giá trị của value. The null passed to useState là là 1 giá trị khởi tạo của biến state. So value here start off equal to null.

Kể từ khi square component no longer accepts props anymore (Không chấp nhận props nên ),( bạn sẽ xóa value prop từ tất cả 9 Square component created by the board component ( được tạo bởi board component )

Bằng cách gọi set function từ onclick handler, bạn telling React to re-render Square bất cứ khi nào button đc click. Sau khi update, The Square value sẽ là x nên bạn sẽ nhìn thấy x on the game board. Click vào Square bất kì X sẽ đc show up.

Mỗi ô vuông có trạng thái riêng của mình: the value stored ( giá trị lưu trữ ) trong mỗi Square là 1 completely independent( hoàn toàn độc lập ) với những cái khác. Khi bạn call set function trong component, React automatically update các thành phần con bên trong.

Sau đó bạn có thể thực hiện những thay đổi tên, code của bạn sẽ trông như này.

### React Developer Tools

## Completing the game

Thời điểm này , bạn có tất cả các block building cơ bản của game tictactoe. Để hoàn thành game bạn cần luân phiên đặt x và o vào board và bạn cần xác định người cách chiến thắng.

### Lifting state up

Hiện tại, mỗi Square component duy trì( maintains) 1 phần trạng thái của game. Để tìm ra người chiến thắng, the Board cần bằng cái nào đó biết được state của mỗi 9 Square component

Vậy làm như nào bạn làm đc nó. ( tiếp cận – approach) . Đầu tiên bạn có thể đoán rằng the Board cần “ask” mỗi Square từ Square’s state. ( trạng thái của mỗi state). Mặc dù tiếp cận bằng cách này có thể làm đc bằng react , chúng tôi không khuyến kích nó bởi vì code sẽ trở lên khó hiểu hơn, dễ bug, và khó refactor ( làm lại ) . thay vào đó cách tốt để có thể tiếp cận là lưu trữ state game trong parent board component thay vì trong Square. The Board component nói cho Square cái gì đang hiển thị bơi passing a prop, giống khi bạn passed number each Squase

Để collect data từ multiple children hoặc để 2 child components giao tiếp với nhau. Thay vào đó hãy khai báo trạng thái trong component cha của chúng. Parent component có thể truyền lại trạng thái đó thông qua props. Điều này giúp đồng bộ các component child đồng bọ với parent.

Việc nâng trạng thái parent component khi react component là tái cấu trúc.

Hãy tận dụng cơ hội này, chỉnh sử component Board nên declares của state variable name square thành 1 mảng mặc định 9 null tương ứng vơi 9 ô.

Sẽ tạo sao 9 element và mỗi sets là null. The useState() call sẽ khai báo 1 square stte ban đầu được đặt là 1 array. Mỗi mục của array tương ứng với 1 value của square. Khi bạn fill vào bảng , squares sẽ trông như này.

Bây h component board cần điền the value prop xuống mỗi ô square

Tiếp theo bạn sẽ chỉnh lại Square component nhận value prop từ Board component. Nó sẽ xóa trạng thái riêng giá trị và button của onclick prop

Mỗi square sẽ nhận đc value của prop hoặc sẽ là X, O, null từ empty squares.

Tiếp theo bạn cần thay dổi những gì xảy ra khi click vào square . Component board sẽ duy trì square đc lấp đầy . bạn sẽ cần tạo 1 way cho Square để update the board’s state. Vì state là private đối với component defines nó . Bạn không thế update state trực tiếp tuwf Square.

Thay vì đó, bạn sẽ truyền xuống function từ the Board component thới Square component và bạn sẽ có Square call function khi 1 square được click. Bạn sẽ bắt đầu với function mà Square component được click. Bạn sẽ call function OnSquareClick.

Bây h bạn sẽ connect the onSquareClick prop với function trong board component mà bạn sẽ đặt tên là handleClick.

To connect onSquareClick to handleClick, bạn cần truyền 1 function tới onSquare prop của the first Square component.

The HandleClick function tao ra 1 bản copy của Square array ( nextSquare ) với JS.slice() method. Sau đó handleClick update the nextSquares array add thêm x vào first square.

Call the setsquare function để React biết state của component đc change. Nó sẽ kích hoạt re-render component và sử dung squares stast board cũng như các child component của nó ( các Square tạo nên board )

H bạn add đc ‘X vào bảng và bạn cần add X cho bất cứ Square nào , bạn chỉ cần thêm I vào là đc

Tiếp theo bạn cần triền I vào handleClick , bạn có thể thứ thiết lập onSquareClick prop của square từ handleClick(0) như thế này nhưng nó ko hoạt động.

<Square value={squares[0]} onSquareClick={handleClick(0)} />

Bạn sẽ biết tại sao nó không work. Do handleLClick (0) đc call thì nó sẽ rendering the board component . bới gì handleClick 0 sau khi thay đổi state nó sẽ đc calling lại bới setsquare, toàn bộ component board sẽ đc render . nhưng handleClick đc call => lặp vô hạn.

Bạn có thế fix nó bằng cách handleFirstSquare Click và call handleClick(0) sau đó. Bạn sẽ chuyển chứ ko gọi call . Bạn sẽ tránh được infinite loop.

export default function Board() {

// ...

return (

<>

<div className="board-row">

<Square value={squares[0]} onSquareClick={() => handleClick(0)} />

// ...

);

}

Lưu ý cú pháp mới () => syntax ( Es6) () => handleClick(0) là 1 arrow function, đó là 1 cách ngắn hơn để định nghĩa chức năng. . Khi square đc clicked , code sau khi => arrow trun sẽ call handClick(0).

Bây h bạn cần update 8 square còn lại call handleClick bằng arrow functiuon bạn truyền. đảm bảo rằng mỗi đối số mỗi lần call tương úng với index chính xác. của Square .

export default function Board() {

// ...

return (

<>

<div className="board-row">

<Square value={squares[0]} onSquareClick={() => handleClick(0)} />

<Square value={squares[1]} onSquareClick={() => handleClick(1)} />

<Square value={squares[2]} onSquareClick={() => handleClick(2)} />

</div>

<div className="board-row">

<Square value={squares[3]} onSquareClick={() => handleClick(3)} />

<Square value={squares[4]} onSquareClick={() => handleClick(4)} />

<Square value={squares[5]} onSquareClick={() => handleClick(5)} />

</div>

<div className="board-row">

<Square value={squares[6]} onSquareClick={() => handleClick(6)} />

<Square value={squares[7]} onSquareClick={() => handleClick(7)} />

<Square value={squares[8]} onSquareClick={() => handleClick(8)} />

</div>

</>

);

};

Bây h bạn cần add lại x vào bất cứ Square nào trong bảng bởi clicking.

Nhưng thời gian này all the state managerment là handled bởi the board component.

Bây h việc xử lí state của bạn nằm trong component board. The parent Board component truyền vào props child Square component để chúng hiển thị chính xác. Khi click vào a square , the child square component sẽ asks parent board component update state của board. Trong khi the board state changes , cả the board component và mọi child Square sẽ rende tự động. Giữ state của all square trong component board sẽ cho phép nó xác định người chiến thắng trong tương lai

Tóm tắt lại những gì xảy ra khi người dung clicj vào góc trên trái Squara trên board và add X cho nó.

1. Click vào square góc trên bên trái => runs function rồi button nhân đc onClick prop từ Square. The Square component đã nhận dc function đó dưới dạng prop onSquareClick của nó từ bảng. The Board component định nghĩ chức năng trực tiếp từ JSX, nó call handleClick cùng argument (0)
2. hanldeClock sử dụng argument (0) update element đầu tiên của square array từ null đến X
3. the squares state của board component được update, vì the board và all các children đc re – render. Điều này khiến giá trị prop của Square thanh đổi từ null thành X

cuối cùng người dùng thấy nó thay đổi từ null sang X khi click vào nó

### Why immutability is important

Lưu ý how in handleClick , bạn call .slice() sẽ tạo ra 1 bản sao của Square thay vi sửa đổi bảng hiện có. Để giải thích tại sao, chúng tôi cần thảo luận về tính bất biến và tại sao tính bất biến quan trong khi học.

Nói chung có 2 cách tiếp cận để thay đổi dữ liệu. cách tiếp cận 1 là mutate the data trực tiếp data của value. Cách tiếp cận 2 là thay thế data bời 1 bản sao có những dữ diệu thay đổi mong muốn. Đây là những gì bạn thấy khi bạn thay đổi array square

const squares = [null, null, null, null, null, null, null, null, null];

squares[0] = 'X';

// Now `squares` is ["X", null, null, null, null, null, null, null, null];

And here is what it would look like if you changed data without mutating the squares array: đây là những gì bạn thấy khi bạn thay đổi data ,mà không thay đổi Square array

const squares = [null, null, null, null, null, null, null, null, null];

const nextSquares = ['X', null, null, null, null, null, null, null, null];

// Now `squares` is unchanged, but `nextSquares` first element is 'X' rather than `null`

Kết quả là giống nhau nhưng bởi ko thay đổi trực tiếp( chỉ thay đổi data cơ bản) bạn sẽ nhận được 1 số lợi ích

Tính bất biến làm cho các tính năng phức tạp dễ triển khai hơn nhiều. Ở phần sau hướng dẫn này bạn sẽ triển khai tính này time travel cho phép bạn xem lại lịch sử của trò chơ và quay lại các nước đi cũ. Chức năng này không dành rieng cho trò chơi, khả năng hoàn tác làm lại một số hành động nhất định là yêu cầu chung đối với ưng dụng.

Việc tránh thay đổi dữ liệu trực tiếp cho phép bạn giữ nguyên các phiên bản trước của dữ liệu và sử dụng lại chúng sau này.

Ngoài ra có 1 số lợi ích khác của tính bất biến . theo mặc định , tất cả child component re-render tự động. khi state of parent component changes. Điều này bao gồm cả child component không bị ảnh hưởng bởi sự thay đổi. Mặc dù re-render không gây chú ý đến user ( bạn không nên chủ động cố gắng tránh nó)

Bạn có thế muốn bỏ qua việc render lại 1 phần của cây không bị ảnh hưởng bởi nó vì lí do hiệu suất. tính bất biến giúp các thành phần so sánh dữ liệu của chúng có thay đổi hay không , bạn có thể tìm hiểu thêm về react chọn thời điểm kết xuất lại 1 thành phần trong tham chiếu API Reference.

### Taking turns

Nó bây giwof cần fixx khuyết điểm lớn trong tictactoe game là the ) ko thể đc đánh trong board. Bạn sẽ cần set nước đi đầu mặc định là X . hãy theo dỗi điều này bằng cách thêm 1 phần trạng thái khác vào phần board.

function Board() {

const [xIsNext, setXIsNext] = useState(true);

const [squares, setSquares] = useState(Array(9).fill(null));

// ...

}

Mỗi khi người chơi di chuyển, xIsNext sẽ Đảo để xác định khi nào người chơi đi tiếp và trạng thái game sẽ lưu lại, Bạn sẽ update the board handleClick function để đảo value xIsNext:

Bây h bạn click vào hình vuông khác nhau, chúng sẽ Luân phiên giữa X và O . Nhưng lại xảy ra vấn đề, hãy thử nhập vào hình vuông nhiều lần

X sẽ đc ghi đề lên bơi o , điều này tạo nên 1 điều thú vị của trò chơi nhưng bây h chúng tôi sẽ tuân theo các quy tác ban đầu.

Khi bạn đánh dấu square với X hoặc O trước tiên bạn không kiểm tra xem square đó đã có X hay O chưa,. Bạn có thể khắc phục nó bằng cách quay lại sớm . Bạn sẽ check nếu Square đó đã có X hoặc O. Nếu Square đó đã có được điền, bạn hãy return trong handleCLick function sớm, trước khi nó update board state.

Bây h bạn chỉ add X hoặc O vào Square rổpngx

### Declaring a winner

Bây h người chơi có thế thay phiên nhau dánh , bạn sẽ muốn show khi nào game là chiến thăng và khi nào khi nào ko còn lượt nào đê thực hiện. để làm điêu này,m bạn sẽ cần add 1 function hỗ trọ là calculateWinner rỗi kiểm tra triến thắng, trả về X hoặc hoặc null nếu thích hợp. Đưng lo quá niếu về hàm tính toán, nó ko not specific to react.

Bạn cần gọi calculateWinner( Square) trong board component handleClick function để check nếu 1 player chiến thắng. bạn có thể thực hiện kiểm tra này luc bạn kiểm tra nếu người chơi đã clicj vào Square đã có X hoặc ) . chúng sẽ return sớm trong 2 trường hợp.

Để player biết khi nào game là kết thúc, bạn cần hiển thị dòng text. Winne X hoặc Winner 0; để làm việc đó bạn cần add thêm 1 status section trong board component. The status sẽ hiển thị the winner nếu game kết thúc, và nếu game còn tiếp tục bạn sẽ hiển thị đến lượt chơi của ai.

## Adding time travel

As a final exercise, let’s make chúng ta tạo khả này go back in time , trả về các nước đi trước đo trong trò chơi, lưu trữ lịch sử di chuyển.

### Storing a history of moves

Bạn dùng slice() mỗi lần di chuyển bạn đã có sự bất biến và bạn có thể lưu lại lịch sử đó bằng history , và mỗi khi bạn muốn quay lại chỉ cần call nó thôi

### Lifting state up, again

Bạn sẽ viết new top-level component called Game để hiển thị list past moves. Đó là nơi bạn đặt toàn bộ lịch sử của bạn trong trò chơi. Đặt trạng thái lịch sử vào component game sẽ cho phép bạn loại bỏ Square state từ child board component . giống như bạn nâng cấp state của mình, từ Square component thành board component. Bây h bạn sẽ nâng nó từ board thành top-level Game component . Điều này mang đến component Game full control over ( toàn quyền kiểm soát ) data và và và cho phép nó hướng dẫn hiển thị các lịch sử từ các lượt trước đó.

Đầu tiên add a Game component with export default, Render board component và markup ( đánh dấu)

function Board() {

// ...

}

export default function Game() {

return (

<div className="game">

<div className="game-board">

<Board />

</div>

<div className="game-info">

<ol>{/\*TODO\*/}</ol>

</div>

</div>

);

}

Tiếp theo tạo handplay function inside the game component được thành phần board call update , Truyền xIsNext , curretSquares và handlePlay as props the board component.

export default function Game() {

const [xIsNext, setXIsNext] = useState(true);

const [history, setHistory] = useState([Array(9).fill(null)]);

const currentSquares = history[history.length - 1];

function handlePlay(nextSquares) {

// TODO

}

return (

<div className="game">

<div className="game-board">

<Board xIsNext={xIsNext} squares={currentSquares} onPlay={handlePlay} />

//...

)

}

Tiếp theo change board component three props xIsNext, square , và new onPlay function.

Và call useState