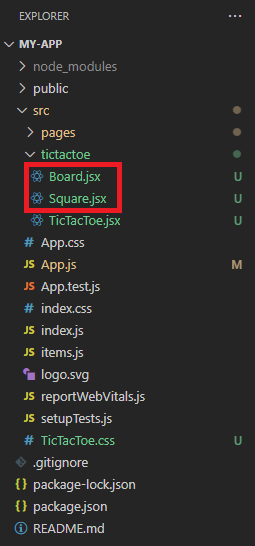
Tic-Tac-Toe Step 2: Square component and base for the Board component

# Introduction:

In this exercise we will build a **base for our game board**. We will start by creating the Square component and move to the board after that. We will build the board by **rendering the squares inside** of it, the squares are the Tic-Tac-Toe game boxes **where you make the moves**. In this exercise we will only make the board visible, we will start adding functionality to it in the later steps.

Start by creating two files inside the tictactoe folder called **Square.jsx** and **Board.jsx**.



# Square.jsx:

Create base component for the Square.jsx file. Component name will simply be **Square**. This component will be the individual box of the board where the players make their turns (**adding X or O in turns**). The game will not be playable for a while, the goal of this exercise is only to make the board visible, it will not have any function yet. The Square will be very short and simple component.

When we eventually add some functionality on the game, we’ll use the onClick event handler to make the turns, but for now we’re going to **focus on the visual side**.

Let’s start by **adding a header** inside the components div element, and name it **X** or something that we will see something in our browser after we render the board:

import React from "react";

export const Square = () => {

    return (

        <div>

            <h1>X</h1>

        </div>

    )

}

At this point we are going to add className as a prop to the Square component, so we can set classNames for the individual squares from the **parent component**, this will be **critical** for the stylesheet to work properly.

export const Square = (props) => {

    const classes = props.className?`${props.className} square`:'square';

    return (

        <div className={classes}>

            <h1>X</h1>

        </div>

    )

}

# ternary operator:

We're going to use the JavaScript [**ternary operator**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Conditional_Operator) to swich classNames.

The ternary operator works with a **boolean** value. If the condition is **true** it will return the first choice, if it is **false**, it will give the latter one:

\*\*\*EXAMPLE\*\*\*

(isThisTrue ? 'yes' : 'no');

We will use the operator to check if the className prop is set and **if it is**, we're adding it **and** square as the className for the square, otherwise we'll set the className to be **just** square.

# Board.jsx:

For now, we will leave the Square component alone and move to the Board.jsx file. We will build a **rough structure** for the board and add the styling at this point.

Start with adding a base for the component named Board, **and importing the Square component at the top of the page:**

import React from "react";

import { Square } from './Square';

export const Board = () => {

    return (

        <div>

        </div>

    )

}

As you know, the Tic-Tac-Toe board consists of 9 boxes, 3 boxes per row, on 3 rows(**3x3**). We will create a div element for every row, every row will contain **three** of our Square components.

import React from "react";

import { Square } from './Square';

export const Board = () => {

    return (

        <div>

            <div>

                <Square />

                <Square />

                <Square />

            </div>

            <div>

                <Square />

                <Square />

                <Square />

            </div>

            <div>

                <Square />

                <Square />

                <Square />

            </div>

        </div>

    )

}

Now we have a **very** rough structure of the board. Here we will add functionality in the future.

# TicTacToe.jsx:

Now that we have created the board, let’s move to the base component (TicTacToe.jsx) and render our board there, once again **remember to import the board at the top of the file**. Let’s also add additional div element for the game:

import React from "react";

import './TicTacToe.css';

import { Board } from "./Board";

export const TicTacToe = () => {

    return (

        <div>

            <div>

                <h1>Tic-Tac-Toe</h1>

                <Board />

            </div>

        </div>

    );

}

Styling:

We should now see a bunch of X’s on the browser, but they are rendered in a list on 9 rows:

A picture containing graphical user interface

Description automatically generated

To fix that we will **copy the class names** for our components to make the stylesheet kick in:

Board.jsx:

export const Board = () => {

    return (

        <div>

            <div className="row">

                <Square className="b-bottom-r"/>

                <Square className="b-bottom-r"/>

                <Square className="b-bottom"/>

            </div>

            <div className="row">

                <Square className="b-bottom-r"/>

                <Square className="b-bottom-r"/>

                <Square className="b-bottom"/>

            </div>

            <div className="row">

                <Square className="b-right"/>

                <Square className="b-right"/>

                <Square />

            </div>

        </div>

    )

}

TicTacToe.jsx

export const TicTacToe = () => {

    return (

        <div>

            <div className="game">

                <h1>Tic-Tac-Toe</h1>

                <Board />

            </div>

        </div>

    );

}

You can go ahead and **copy the code above** in the files (we copied and linked the stylesheet to the board in the previous exercise, so you don’t have to worry about that).

Now our board should be showing in the browser:

Icon

Description automatically generated

The X’s are **hard coded** and work as **placeholders** at the moment, they will be removed when we start writing on the board, nothing is clickable yet. But now that we have our board visible and aligned correctly. We will add useState to our game in the next exercise **See you there!** 😊