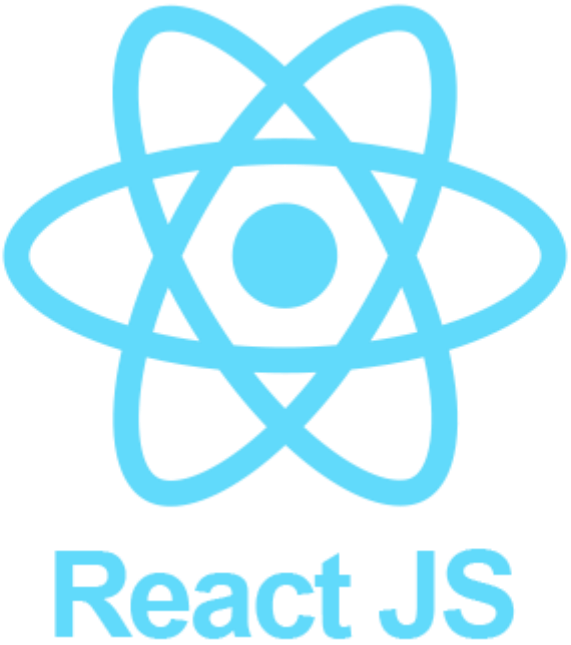


**ASSIGNMENT**

**NAME :** VIGNESHWAR K R

**ROLL NO:** 20I159

## WHAT IS REACT.JS ?

React is a popular JavaScript library used for web development. React.js or ReactJS or React are different ways to represent ReactJS. Today's many large-scale companies (Netflix, Instagram, to name a few) also use React JS. There are many advantages of using this framework over other frameworks, and It's ranking under the top 10 programming languages for the last few years under various language ranking indices. React.js is a front-end JavaScript framework developed by Facebook. To build composable user interfaces predictably and efficiently using declarative code, we use React. It's an open-source and component-based framework responsible for creating the application's view layer.

- ReactJs follows the Model View Controller (MVC) architecture, and the view layer is accountable for handling mobile and web apps.
- React is famous for building single-page applications and mobile apps.

## FEATURES OF REACT:

**1. JSX - JavaScript Syntax Extension:** JSX is a preferable choice for many web developers. It isn't necessary to use JSX in React development, but there is a massive difference between writing react.js documents in JSX and JavaScript. JSX is a syntax extension to JavaScript. By using that, we can write HTML structures in the same file that contains JavaScript code.

**2. Unidirectional Data Flow and Flux**: React.js is designed so that it will only support data that is flowing downstream, in one direction. If the data has to flow in anot

her direction, you will need additional features. React contains a set of immutable values passed to the component renderer as properties in HTML tags. The components cannot modify any properties directly but support a call back function to do modifications.

**3. Virtual Document Object Model (VDOM)**: React contains a lightweight representation of real DOM in the memory called Virtual DOM. Manipulating real DOM is much slower compared to VDOM as nothing gets drawn on the screen. When any object's state changes, VDOM modifies only that object in real DOM instead of updating whole objects. That makes things move fast,

particularly compared with other front-end technologies that have to update each object even if only a single object changes in the web application.

**4. Extensions:** React supports various extensions for application architecture. It supports server-side rendering, Flux, and Redux extensively in web app development. React Native is a popular framework developed from React for creating cross-compatible mobile apps.

## INSTALLATION:

- Download NodeJS from the web and install it.
- Open cmd prompt and create react project using the syntax – 'npx/npm react-create-app appname.
- This node can be either 'npx/npm'.
- Npx is created with minimal basic library from JS where as npm is created with all the library packages.
- Source files and scripts will be created in a folder name of your app name is the directory you created in command prompt.

## FILE STRUCTURES:

➢ Node (node manager)

➢ Public

- Image
- Index.html

➢ Src

- Components
  - Card.js
  - Cards.js
- App.js( responsible for HTML display)
- Index.js (root component)
- Index.css (style for body tag)

○ Style.css (css for app.js and components)

➢ package. Json (scripts and dependencies required )

## PROPS:

Props are attribute passed by HTML to components of JavaScript, which are similar to OOPs concept where we only pass attribute names but no need parameter in the component. This keyword must be used while used inside a class. Props are immutable they get passed through component.

## STATE:

Managed inside the components. Variables inside the function body. They are mutable. Accessed by 'useState' in functional component and 'this.state' in class component. Using state we can create a value setSate method is used

## EVENT HANDLING:

Handling events with react elements is very similar to handling events on DOM elements. Event handlers in react for HTML elements are button and input elements. Onclick event uses different kinds of handlers. There are three kinds of event handlers:

- Event Handler
- Inline Event Handler
- Callback Event Handler

## COMPONENTS:

Components in React basically return a piece of JSX code that tells what should be rendered on the screen. In React, we mainly have two types of components:

1. **Functional Components**: Functional components are simply javascript functions. We can create a functional component in React by writing a javascript function. These functions may or may not receive data as parameters, we will discuss this later in the tutorial.

2. **Class Components**: The class components are a little more complex than the functional components. The functional components are not aware of the other components in your program whereas the class components can work with each other. We can pass data from one class component to other class components. We can use JavaScript ES6 classes to create class-based components in React.

# Tic tac Toe:

1. [X] Display the location for each move in the format (col, row) in the move history list.

2. [X] Bold the currently selected item in the move list.

3. [X] Rewrite Board to use two loops to make the squares instead of hardcoding them.

4. [X] Add a toggle button that lets you sort the moves in either ascending or descending order.

5. [X] When someone wins, highlight the three squares that caused the win.

6. [X] When no one wins, display a message about the result being a draw.

## Code:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
function Square(props) {
const className = 'square' + (props.highlight ? ' highlight' : '');
return (
<button
className={className}
onClick={props.onClick}>
{props.value}
</button>
);
}
class Board extends React.Component {
renderSquare(i) {
const winLine = this.props.winLine;
return (
<Square
key={i}
value={this.props.squares[i]}
onClick={() => this.props.onClick(i)}
highlight={winLine && winLine.includes(i)}
```

```
/>
);
}
render() {
// Use two loops to make the squares
const boardSize = 3;
let squares = [];
for (let i = 0; i < boardSize; ++i) {
let row = [];
for (let j = 0; j < boardSize; ++j) {
row.push(this.renderSquare(i * boardSize + j));
}
squares.push(<div key={i} className="board-row">{row}</div>);
}
return (
<div>{squares}</div>
);
}
}
class Game extends React.Component {
constructor(props) {
super(props);
this.state = {
```

```
history: [
{
squares: Array(9).fill(null)
}
],
stepNumber: 0,
xIsNext: true,
isAscending: true
};
}
handleClick(i) {
const history = this.state.history.slice(0, this.state.stepNumber + 1);
const current = history[history.length - 1];
const squares = current.squares.slice();
if (calculateWinner(squares).winner || squares[i]) {
return;
}
squares[i] = this.state.xIsNext ? "X" : "O";
this.setState({
history: history.concat([
{
squares: squares,
// Store the index of the latest moved square
```

```
latestMoveSquare: i
}
]),
stepNumber: history.length,
xIsNext: !this.state.xIsNext
});
}
jumpTo(step) {
this.setState({
stepNumber: step,
xIsNext: (step % 2) === 0
});
}
handleSortToggle() {
this.setState({
isAscending: !this.state.isAscending
});
}
render() {
const history = this.state.history;
const stepNumber = this.state.stepNumber;
const current = history[stepNumber];
const winInfo = calculateWinner(current.squares);
```

```
const winner = winInfo.winner;
let moves = history.map((step, move) => {
const latestMoveSquare = step.latestMoveSquare;
const col = 1 + latestMoveSquare % 3;
const row = 1 + Math.floor(latestMoveSquare / 3);
const desc = move ?
`Go to move #${move} (${col}, ${row})` :
'Go to game start';
return (
<li key={move}>
{/* Bold the currently selected item */ }
<button
className={move === stepNumber ? 'move-list-item-selected' : ''}
onClick={() => this.jumpTo(move)}>{desc}
</button>
</li>
);
});
let status;
if (winner) {
status = "Winner: " + winner;
} else {
if (winInfo.isDraw) {
```

```
status = "Draw";
} else {
status = "Next player: " + (this.state.xIsNext ? "X" : "O");
}
}
const isAscending = this.state.isAscending;
if (!isAscending) {
moves.reverse();
}
return (
<div className="game">
<div className="game-board">
<Board
squares={current.squares}
onClick={i => this.handleClick(i)}
winLine={winInfo.line}
/>
</div>
<div className="game-info">
<div>{status}</div>
<button onClick={() => this.handleSortToggle()}>
{isAscending ? 'descending' : 'ascending'}
</button>
```

```
<ol>{moves}</ol>
</div>
</div>
);
}
}
// ========================================
ReactDOM.render(<Game />, document.getElementById("root"));
function calculateWinner(squares) {
const lines = [
[0, 1, 2],
[3, 4, 5],
[6, 7, 8],
[0, 3, 6],
[1, 4, 7],
[2, 5, 8],
[0, 4, 8],
[2, 4, 6]
];
for (let i = 0; i < lines.length; i++) {
const [a, b, c] = lines[i];
if (squares[a] && squares[a] === squares[b] && squares[a] === squares[c]) {
return {
```

```
winner: squares[a],
line: lines[i],
isDraw: false,
};
}
}
let isDraw = true;
for (let i = 0; i < squares.length; i++) {
if (squares[i] === null) {
isDraw = false;
break;
}
}
return {
winner: null,
line: null,
isDraw: isDraw,
};
```

**OUTPUT:**