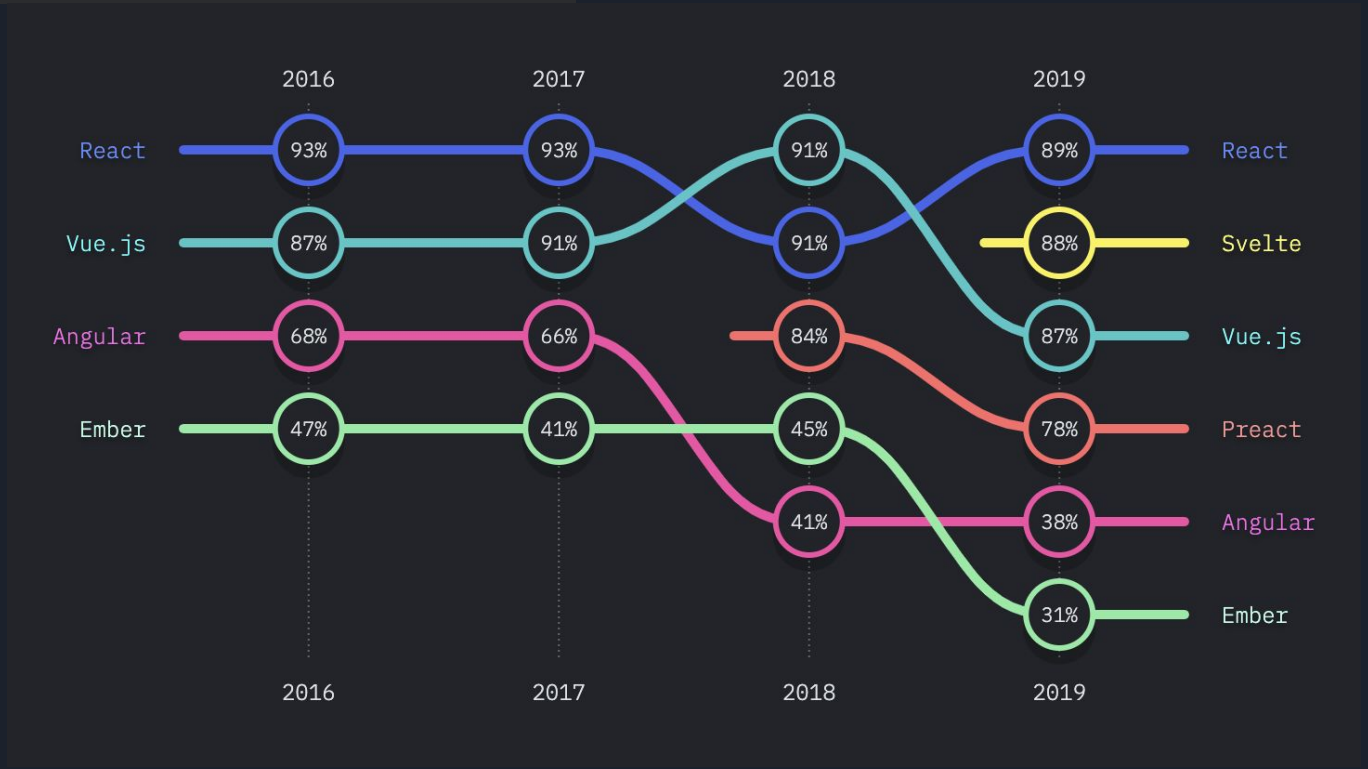
# Tic Tac Toe Game

Jisoo Park, Sneha Yendluri, Samriddhi Lamichhane

# React


2016
2017
2018
2019
React
Vue.js
Angular
Ember
93%
87%
68%
47%
93%
91%
66%
41%
91%
91%
84%
45%
41%
89%
88%
87%
78%
38%
31%
React
Svelte
Vue.js
Preact
Angular
Ember
2016
2017
2018
2019

# React vs Scala


# Callbacks

```
    return (
        <div className="App">
            {/* Shrinks the popup when there is no winner */}
            <div className={`winner ${winner !== '' ? '' : 'shrink'}`}>
                {/* Display the current winner */}
                <div className='winner-text'>{winner}</div>
                {/* Button used to reset the board */}
                <button onClick={() => resetBoard()}>
                    Reset Board
                </button>
            </div>
            {/* Custom made board component comprising of
            the tic-tac-toe board  */}
            <Board reset={reset} setReset={setReset} winner={winner}
                setWinner={setWinner} />
            <Info />
        </div>
    );
}
```

```
return (
    <div ref={boardRef} className="board">
        <div className="input input-1"
            onClick={(e) => draw(e, 1)}></div>
        <div className="input input-2"
            onClick={(e) => draw(e, 2)}></div>
        <div className="input input-3"
            onClick={(e) => draw(e, 3)}></div>
        <div className="input input-4"
            onClick={(e) => draw(e, 4)}></div>
        <div className="input input-5"
            onClick={(e) => draw(e, 5)}></div>
        <div className="input input-6"
            onClick={(e) => draw(e, 6)}></div>
        <div className="input input-7"
            onClick={(e) => draw(e, 7)}></div>
        <div className="input input-8"
            onClick={(e) => draw(e, 8)}></div>
        <div className="input input-9"
            onClick={(e) => draw(e, 9)}></div>
    </div>
)
}
```

# Higher-order functions

- useRef

```
// Creating a reference for the board
const boardRef = useRef(null);
```

```
function TextInputWithFocusButton() {
  const inputEl = useRef(null);
  const onButtonClick = () => {
    // `current` points to the mounted text input element
    inputEl.current.focus();
  };
  return (
    <>
      <input ref={inputEl} type="text" />
      <button onClick={onButtonClick}>Focus the input</button>
    </>
  );
}
```

# Higher-order functions

- useState

```
import { useState } from 'react';

function App() {

    // Creating a reset state, which indicates whether
    // the game should be reset or not
    const [reset, setReset] = useState(false);

    // Creating a winner state, which indicates
    // the current winner
    const [winner, setWinner] = useState('');

    // Sets the reset property to true
    // which starts the chain
    // reaction of resetting the board
    const resetBoard = () => {
        setReset(true);
    }
```

```
// Importing the useState hook, useEffect hook and useRef hook
import { useState, useEffect, useRef } from "react";

const Board = ({ reset, setReset, winner, setWinner }) => {

    // Creating a turn state, which indicates the current turn
    const [turn, setTurn] = useState(0);

    // Creating a data state, which contains the
    // current picture of the board
    const [data, setData] = useState(['', '', '', '', '',
        '', '', '', '']);
```

# Higher-order functions

- useEffect

```
// A winner is decided
useEffect(() => {

    // Clearing the data state
    setData(['', '', '', '', '', '', '', '', '']);

    // Getting all the children(cells) of the board
    const cells = boardRef.current.children

    // Clearing out the board
    for (let i = 0; i < 9; i++) {
        cells[i].innerText = '';
    }

    // Resetting the turn to player 0
    setTurn(0);

    // Resetting the winner
    setWinner('');
    setReset(false);
}, [reset, setReset, setWinner])
```

```
// A winner is decided
useEffect(() => {

    // Checks for the win condition in rows
    const checkRow = () => {
        let ans = false;
        for (let i = 0; i < 9; i += 3) {
            ans |= (data[i] === data[i + 1] &&
            data[i] === data[i + 2] &&
            data[i] !== '')
        }
        return ans;
    }

    // Checks for the win condition in cols
    const checkCol = () => {
        let ans = false;
        for (let i = 0; i < 3; i++) {
            ans |= (data[i] === data[i + 3] &&
            data[i] === data[i + 6] &&
            data[i] !== '')
        }
        return ans;
    }

    // Checks for the win condition in diagonals
    const checkDiagonal = () => {
        return ((data[0] === data[4] &&
        data[0] === data[8] && data[0] !== '') ||
        (data[2] === data[4] && data[2] === data[6] &&
        data[2] !== ''));
    }
```

# Abstract Data Type

- Render function

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './css/index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

# Demo

# References


- https://reactjs.org/docs/hooks-reference.html
- https://www.javiercasas.com/articles/typescript-adts
- https://www.codecademy.com/article/goku-kun/introduction-to-adts-in-javascript
- https://reactjs.org/docs/faq-functions.html
- https://reactjs.org/docs/higher-order-components.html#:~:text=A%20higher%2Dorder%20component%20(HOC,and%20returns%20a%20new%20component.

# Thank you :)