

React JS:

It is an open source library developed by Facebook. It's used for building interactive user interfaces and web applications quickly and easily without code complications. The ReactJS framework combines the speed and efficiency of JavaScript with a more efficient method of manipulating the DOM to render web pages faster and create highly dynamic and responsive web applications. It is not a framework. There are various libraries like this. Example: `node.js`. React is a component-based architecture. While HTML and CSS felt bored with dynamic web pages, JavaScript is developed by Brendan Eich. These webpages and its elements run on a browser and the first browser applied in Mozilla Firefox where Brendan Eich was working. Later it is applied in all the browsers. Every browser has separate JavaScript engines. Not only in browsers, JS can be used in all displays. After this revolution, ECMAScript was created in 1997. ECMAScript, also known as JavaScript, is a programming language adopted by the European Computer Manufacturer's Association as a standard for performing computations in Web applications.

Installation:

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

File structure :

- Node (node manager)
- Public
 - Image
 - Index.html
- Src
 - Components
 - Card.js

■ Cards.js

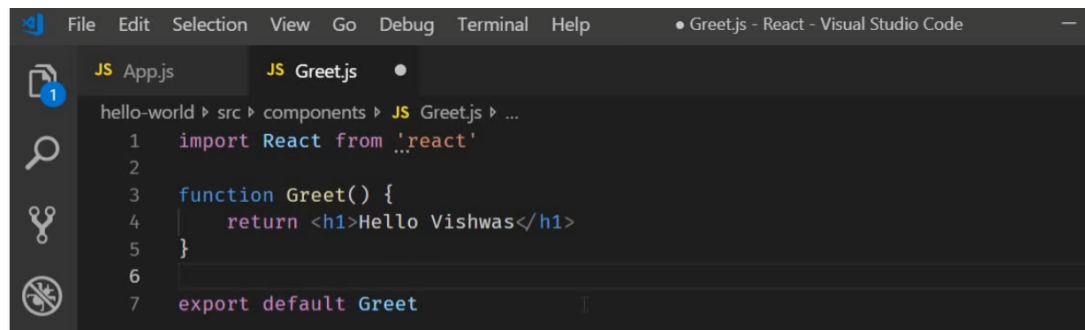
- App.js(responsible for HTML display)
- Index.js (root component)
- Index.css (style for body tag)
- Style.css (css for app.js and components)

➤ package. Json (scripts and dependencies required)

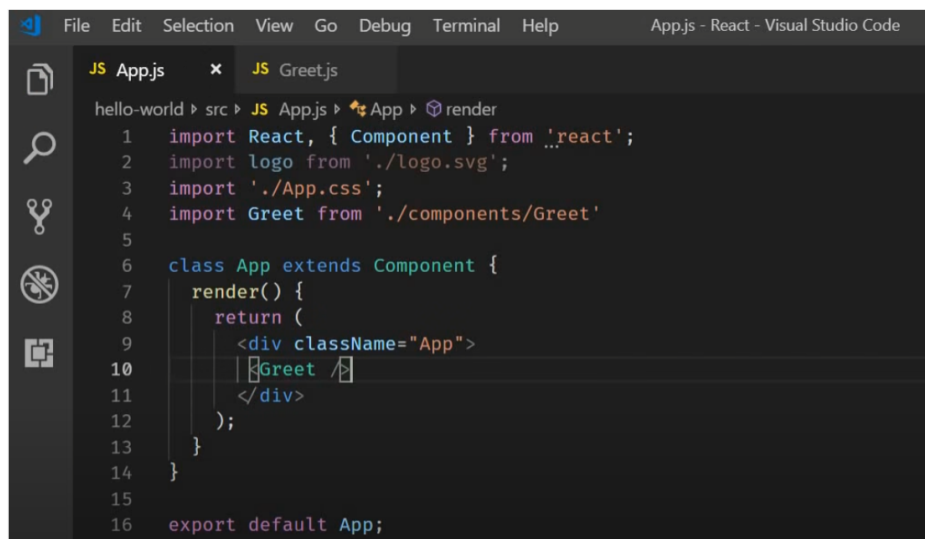
Components:

They are the parts of the interface. They can either js or jsx files. They are reusable and can be nested. There are two types of components namely

- Functional components
 - Returns javascript functions
 - Gets properties as 'props' and returns jsx files (HTML)



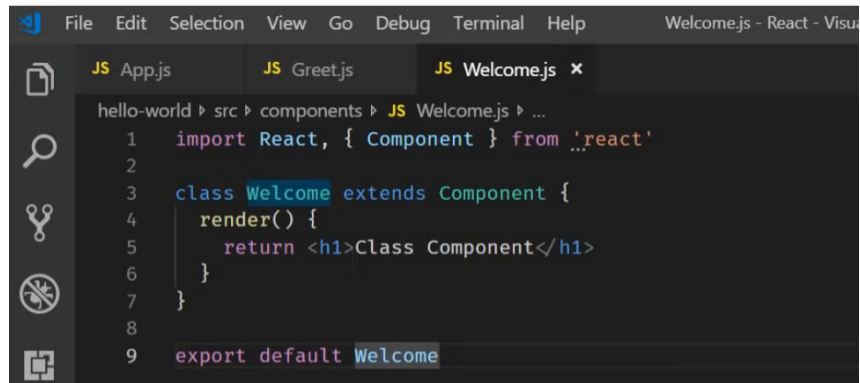
```
1 import React from 'react'
2
3 function Greet() {
4   return <h1>Hello Vishwas</h1>
5 }
6
7 export default Greet
```



```
1 import React, { Component } from 'react';
2 import logo from './logo.svg';
3 import './App.css';
4 import Greet from './components/Greet'
5
6 class App extends Component {
7   render() {
8     return (
9       <div className="App">
10         <Greet />
11       </div>
12     );
13   }
14 }
15
16 export default App;
```

- If we create as function we can call it with any name in the app.js but if create function using => function then exact name must be called in app.js with curly braces in import statement ,
- Preferable ,but little difficult with this key word

- Initially complex but easy
Mainly easy preparation on UI logic
- Stateful class components
 - Regular Es6 class extends the Javascript where render methods returns html elements
 - Optionally gets properties as 'props' and returns jsx files (HTML)
 - Must contains 'extends' and 'render'



```

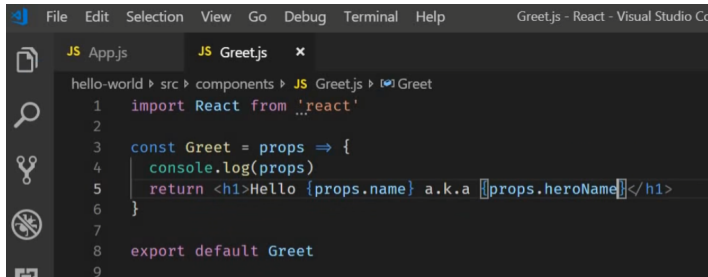
1  import React, { Component } from 'react'
2
3  class Welcome extends Component {
4    render() {
5      return <h1>Class Component</h1>
6    }
7  }
8
9  export default Welcome

```

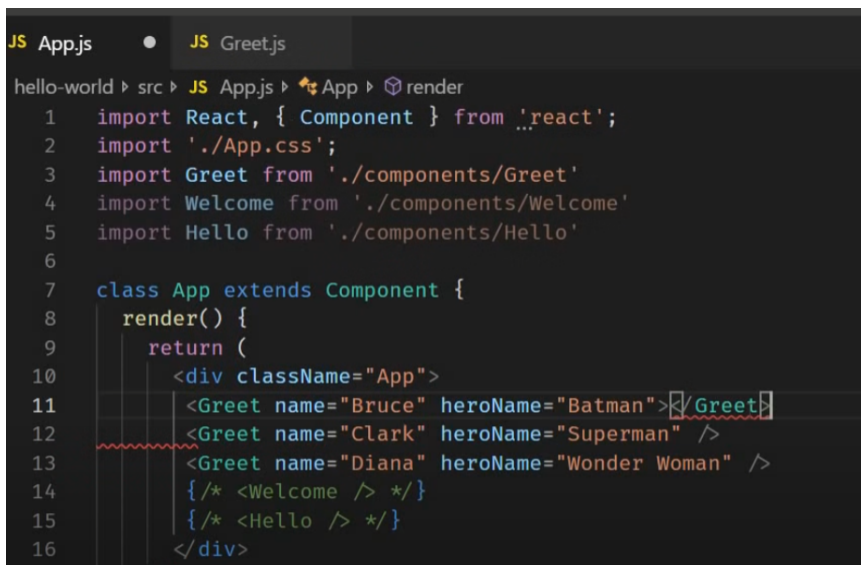
-
- Provide life cycle hocks
- Complex Ui logic

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 parameters in the component. This keyword must be used while used inside a class. Props are immutable they get passed through componen



```
File Edit Selection View Go Debug Terminal Help Greetjs - React - Visual Studio Co
JS App.js JS Greetjs x
hello-world ▸ src ▸ components ▸ JS Greetjs ▸ Greet
1 import React from 'react'
2
3 const Greet = props => {
4   console.log(props)
5   return <h1>Hello {props.name} a.k.a {props.heroName}</h1>
6 }
7
8 export default Greet
9
```



```
JS App.js JS Greetjs
hello-world ▸ src ▸ JS App.js ▸ App ▸ render
1 import React, { Component } from 'react';
2 import './App.css';
3 import Greet from './components/Greet'
4 import Welcome from './components/Welcome'
5 import Hello from './components/Hello'
6
7 class App extends Component {
8   render() {
9     return (
10       <div className="App">
11         <Greet name="Bruce" heroName="Batman"></Greet>
12         <Greet name="Clark" heroName="Superman" />
13         <Greet name="Diana" heroName="Wonder Woman" />
14         { /* <Welcome /> */ }
15         { /* <Hello /> */ }
16       </div>

```

State:

Managed inside the components. Variables declared 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 to edit the value setState method is used

setState:

Each component can have a state. `setState()` enqueues change to the component state and tell React that this component and its children need to be re-rendered with the updated state. This is the primary method you should use to update the UI. A function with the signature: `(state, props) => stateChange`. `state` is the current, up-to-date component state, and the function returns the updated state.

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 event handlers. There are three kinds of event handlers: event handlers, inline event handlers and callback event handlers.

Conditionals in React:

- **if/else**
- **Element variables**
- **Ternary conditional operator**
- **Short circuit operator**

List rendering :

A list is created as an array and operations performed using map() function. It can also be done by props. Mostly map function is preferred .

Hooks:

which lets you use state and code without creating functions. React Hooks are in-built functions that allow React developers to use state and lifecycle methods inside functional components, they also work together with existing code, so they can easily be adopted into a codebase. The way Hooks were pitched to the public was that they allow developers to use state in functional components but under the hood, Hooks are much more powerful than that. They allow React Developers

- **useState()**
 - update, handle and manipulate state inside functional components without needing to convert it to a class component.
- **useEffect()**
 - initial state as an argument and then returns, by making use of array destructuring in JavaScript
- **useContext()**

- hook accepts a context object, i.e the value that is returned from `React.createContext`, and then it returns the current context value for that context.
- `useReducer()`
 - It is used for handling complex states and transitions in state.

Tic toc :

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:

X		
	O	X

Next player: O

ascending

1. **Go to move #3 (1, 1)**
2. Go to move #2 (2, 2)
3. Go to move #1 (3, 2)
4. Go to game start