* npm = **Install and manage packages** (like downloading games).   
  **Download** code packages (like Lego blocks of code).

**Install** them into your project.

**Manage** them (like update, remove, etc.).

* npx = **Run tools temporarily** (like trying a game without installing it). **npx** is like a one-time-use **tool runner**. Instead of **installing** a tool with npm and then using it, npx just **runs it directly** — no need to install it forever.
* a file can only have one “default” export and may have other exports which shud be rendered in app component’s jsx
* React components should start with uppercase ALWAYS!

1)SETUP REACT APP

npx create-react-app mern-todo

cd mern-todo

npm start

2)Understand React Structure

src/App.js is your main component.

Components are reusable UI blocks.

3)Create first component in App.js  
import React from 'react';

function App() {

return (

<div>

<h1>My MERN Todo App</h1>

<p>Let's build something awesome!</p>

</div>

);

}

export default App;

4)REACT BASICS

**a) JSX**

* Looks like HTML, but it’s JavaScript!

Imagine you’re writing a story, but instead of choosing either pictures 🖼️ or words ✍️, you use **both at the same time**. JSX is like that — it mixes HTML (the structure) with JavaScript (the logic).

Eg:

const name = "Alex";

function Hello() {

return <h1>Hello, {name}!</h1>;

}

 <h1> is like HTML.

 {name} is JavaScript.

 Together, it’s JSX!

Eg): const fruits = ["apple", "orange", "kiwi"];

function FruitList() {

return (

<ul>

{fruits.map((fruit, index) => (

<li key={index}>{fruit}</li>

))}

</ul>

);

}

* **const fruits = ["apple", "orange", "kiwi"];**

Just a simple array.

* **<ul> ... </ul>**

This is HTML-style JSX for an **unordered list**.

* **{fruits.map((fruit, index) => ( ... ))}**

=> map is a way to loop through an array and do something with each item.

 It creates a new list (like li items here).

* **=>(fruit, index) => (...) – arrow function**

For every fruit in the array, do this…….(inside bracket stuff)

* =>**<li key={index}>{fruit}</li>**

This creates:

<li>apple</li>

<li>orange</li>

<li>kiwi</li>

* **key={index}**

=>React needs a **unique key** for each item when you loop, to keep track of them.  
Using index is okay for now, since it's just a basic list.

**b) Components**

* Functions that return JSX.

**c) Props**

* Pass data to components.

**d) State**

* Use useState to manage data inside a component.

//React component with state

import React, { useState } from 'react'; //**useState** function lets you make variables that can change

function App() {

const [count, setCount] = useState(0); //initialise count to 0,watch count variable and update the . screen whenever it changes

return (

<div>

<h1>Counter: {count}</h1> //show current no. like counter:0

<button onClick={() => setCount(count + 1)}>Increase</button> When you click the … button, increase the number by 1 using setCount.

</div>

);

}

**🚀 So the full flow is:**

1. You click the button 🔘
2. setCount(count + 1) runs ✅
3. React says "Oh the number changed!" 🔄
4. It re-renders the screen with new number 🔢

**🎮 TL;DR:**

* useState(0) gives you a number that **changes and updates the screen**
* count is the value
* setCount() is how you change it
* React does the rest to show it live!

FINAL APP.JS

//step3

import React,{useState} from 'react';

function App(){

    const[todos,setTodos]=useState([]); //todos holds the list of items u typed //setTodos updates . . it // useState([]) is empty as list is empty (if 0 then 1st item shows as 0)

    const[input,setInput]=useState(''); //input holds what u typing and setInput updates the input . . value //state is set to “” meaning null or empty input

📦 React watches these and **auto-updates** the screen when they change.

    const addTodo=()=>{

      if(input.trim()==='') //ignore if input is empty or just spaces (=== is strictEqualityOperator)

        return;

      setTodos([...todos,input]); //make a **new array** with all old todos + the new one

      setInput(''); //clear the input box after adding

    }; // (…todos) is spread operator that clones the existing obj/string

    return(

      <div>

        <h1>To-do List</h1>

        <input //input box

         value={input} //the input box shows whatever is in the input variable

         onChange={e=>setInput(e.target.value)} //every time you type, update input

         placeholder='add a todo'

         />

        <button onClick={addTodo}>ADD</button>

        <ul>

          {todos.map((todo,index)=>( //Loop through todos array

            <li key={index}>{todo}</li> //For each one, create an <li> to show it

          ))}

        </ul>

      </div>

    );

}

export default App;

//final APP.JS with other functionalities like toggle,delete and clear

import React,{useState} from 'react';

function App(){

    const[todos,setTodos]=useState([]);

    const[input,setInput]=useState('');

    const addTodo=()=>{

      if(input.trim()==='')

        return;

      const newTodo={text:input,done:false}; //list of objects ,make a newTodo

      setTodos([...todos,newTodo]); //clone prev list and add new one

      setInput(''); //set ip box to default empty

    };

    const deleteTodo=(index)=>{

    const newTodos=todos.filter((\_,i)=>i!==index); //filter will remove that index in new objlist

    setTodos(newTodos); //create newlist w/o that todo

    };

    const toggleDone=(index)=>{

      const newTodos=[...todos]; //clone the array

      newTodos[index].done=! newTodos[index].done; //flip the boolean

      setTodos(newTodos); //update state so react re-renders

    };

    const clearAll=()=>{

      setTodos([]); //empty the array

    }

    return(

      <div>

        <h1>To-do List</h1>

        <input //ip box

         value={input} //show whatever is in ip variable

         onChange={e=>setInput(e.target.value)} //everytime u type,update ip

         placeholder='add a todo'

         />

        <button onClick={addTodo}>ADD</button>

        <button onClick={clearAll}>clear</button>

        <ul>

          {todos.map((todo,index)=>( //loop thro array of todos and for every todo create li to..

            <li key={index}> //to show it with index key

                {todo.done?'completed=>  ':'yet to do =>  '}{todo.text} //toggle calling

                <button onClick={()=>toggleDone(index)}>toggle</button>

            <button onClick={()=>deleteTodo(index)}>Delete</button></li> //delete calling

          ))}

        </ul>

      </div>

    );

}

export default App;

//REFER THE NOTEBOOK NOTES also FOR THE REACT FILE

Then we added components folder to put all login inside TodoList.js   
then update the APP.js logic to render <TodoList/>

Then we made a TodoItem.js to include logic of <li> for each component briefly and import in todoList.js and update todoList.js accordingly in map logic

TodoItem includes a checkbox,toggle,delete functs using fav icons.