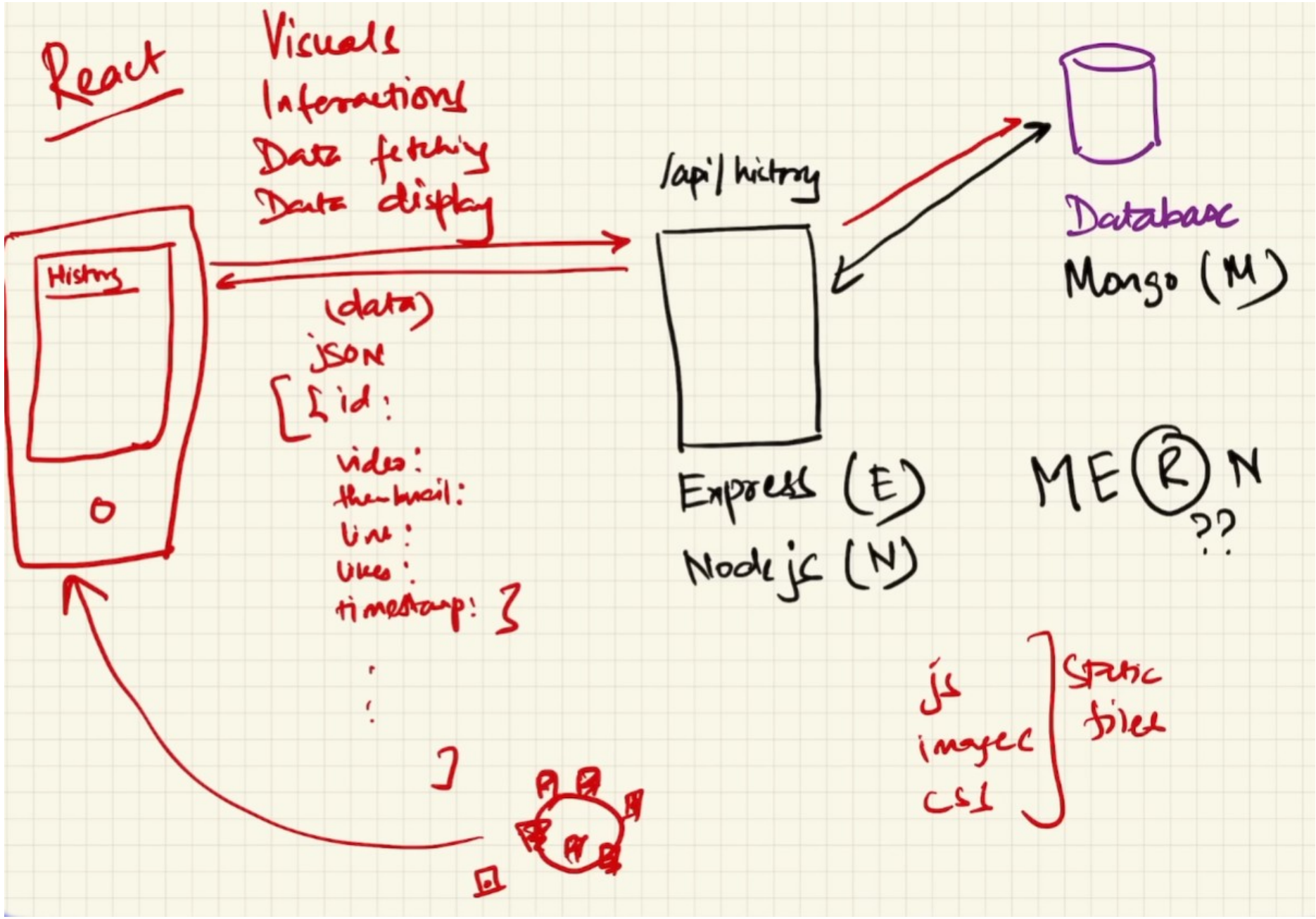
MERN STACK


React
Visuals
Interactions
Data fetching
Data display
History
(data)
JSON
[{ id:
video:
thumbnail:
url:
likes:
timestamp: }
]
/api/history
Database
Mongo (M)
Express (E)
Node.js (N)
MERN
??
js
images
css
Static files

React Tutorial

- React is a JavaScript library for building user interfaces.
- React is used to build single-page applications.
- React allows us to create reusable UI components.
- React, sometimes referred to as a frontend JavaScript framework, is a JavaScript library created by Facebook.
- React only changes what needs to be changed!

React Installation

- First check node version through cmd. Node -v
- Install node.js if not installed
- Npm create vite@latest
- Give project name
- Then go to project folder and install third party deficiencies npm i or(install)
- Npm run dev.

Destructuring:
Destructuring makes it easy to extract only what is needed.

Old way

```
const vehicles = ['mustang', 'f-150', 'expedition'];

// old way
const car = vehicles[0];
const truck = vehicles[1];
const suv = vehicles[2];
```

New way

```
const vehicles = ['mustang', 'f-150', 'expedition'];

const [car, truck, suv] = vehicles;
```

Example

```
function calculate(a, b) {
  const add = a + b;
  const subtract = a - b;
  const multiply = a * b;
  const divide = a / b;

  return [add, subtract, multiply, divide];
}

const [add, subtract, multiply, divide] = calculate(4, 7);
```

Expressions in JSX

- With JSX you can write expressions inside curly braces { }.
- The expression can be a React variable, or property, or any other valid JavaScript expression.

```
<h1>React is {5 + 5} times better with JSX</h1>;
```

One Top Level Element

- The HTML code must be wrapped in ONE top level element.
- For Example:

```
<div>
  <p>I am a paragraph.</p>
  <p>I am a paragraph too.</p>
</div>
```

- This approach is not a good practice because we adding one **extra tag div tag** to the **DOM**.

```
<>
  <p>I am a paragraph.</p>
  <p>I am a paragraph too.</p>
</>
```

tra nodes to the **DOM. Use fragment.**

Attribute class = className and Conditions - if statements

- The **class** keyword is a reserved word in JavaScript, Therefore not allowed to use it in JSX.
- Use attribute **className.**
- **React** supports **if** statements, but not inside JSX.
- To be able to use conditional statements in JSX, put the **if** statements outside of the JSX, or use a **ternary expression** instead.

```
const x = 5;
let text = "Goodbye";
if (x < 10) {
  text = "Hello";
}

const myElement = <h1>{text}</h1>;
```

```
const x = 5;

const myElement = <h1>{(x) < 10 ? "Hello" : "Goodbye"}</h1>;
```

React Components

- Components are like functions that return HTML elements.
- Components are independent and reusable.
- Components come in two types.
  - Class components
  - Function components

```
function Car() {
  return <h2>Hi, I am a Car!</h2>;
}
```

```
class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }
}
```

A class component must include the extends React.Component statement.
The component also requires a render() method, this method returns HTML.

Install Bootstrap

- Npm i bootstrap@latest

Now how to import

```
import "bootstrap/dist/css/bootstrap.css";
```

React Props

- Components can be passed as props, which stands for properties.
- Props are like function arguments, and we send them into the component as attributes

```
function Car(props) {
  return <h2>I am a {props.color} Car!</h2>;
}
```

```
<Car color="red"/>
```

React Props

```
function Car(props) {
  return <h2>I am a { props.brand }!</h2>;
}

function Garage() {
  return (
    <>
      <h1>Who lives in my garage?</h1>
      <Car brand="Ford" />
    </>
  );
}
```

```
function Garage() {
  const carName = "Ford";
  return (
    <>
      <h1>Who lives in my garage?</h1>
      <Car brand={ carName } />
    </>
  );
}
```

Components in Components

- We can refer to components inside other components.
- React is all about re-using code.

```
function Car() {
  return <h2>I am a Car!</h2>;
}

function Garage() {
  return (
    <>
      <h1>Who lives in my Garage?</h1>
      <Car />
    </>
  );
}
```

Map function

```
const items = ["New York", "San Francisco", "Tokyo", "London",
```

```
<ul className="list-group">
  {items.map((item) => (
    <li key={item}>{item}</li>
  ))}
```

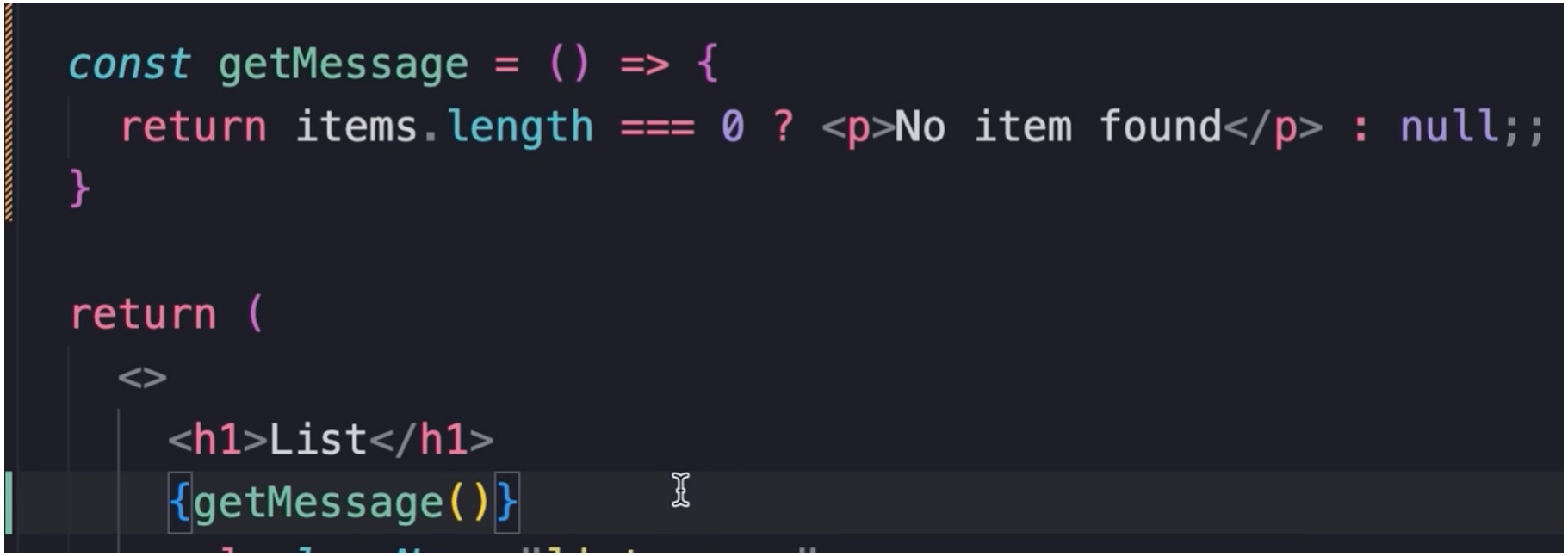
Conditional rendering

```
if (items.length === 0)
  return <><h1>List</h1><p>No item found</p></>;

return (
```

```
return (
  <>
    <h1>List</h1>
    { items.length === 0 ? <p>No item found</p> : null}
    <ul className="list-group">
```

```
const message = items.length === 0 ? <p>No item found</p> : null

return (
  <>
    <h1>List</h1>
    {message}
```

Conditional rendering



More concise and better way of Conditional rendering

```
{items.length === 0 && <p>No item found</p> }
<ul className="list-group">
  {items.map((item) => (
    <li key={item}>{item}</li>
  ))}
</ul>
```

```
> true && 1
< 1
>
```

Handling events

```
{items.map((item) => (
  <li
    className="list-group-item"
    key={item}
    onClick={() => console.log("Clicked")}
  >
    {item}
  </li>
))}
```

```
onClick={() => console.log(item)}
```

Handling events

```
{items.map((item, index) => (
  <li
    className="list-group-item"
    key={item}
    onClick={() => console.log(item, index)}
    {item}
  </li>
```

```
onClick={(event) => console.log(event)}
```

React Events

- Just like HTML DOM events, React can perform actions based on user events.
- React events are written in camelCase syntax:
  - onClick instead of onclick.
  - **React event** handlers are written **inside curly braces**:

`onClick={shoot}` instead of `onclick="shoot()"`.

```
const shoot = () => {
  alert("You clicked the shoot button");
};
```

```
<button onClick={shoot}> Click on Button </button>
```

Passing Arguments to React Events

```
function ButtonComp(probs) {
  const shoot = (name) => {
    alert("My name is " + name);
  };
```

```
<button onClick={() => shoot("Hamza")}> Click on Button </button>
```

React List

```
function StudentNames(nameof) {
  return <li> {nameof.name}</li>;
}
export default StudentNames;
```

```
function ButtonComp(probs) {
  const studens = [
    { reg: 1, name: "Ali" },
    { reg: 2, name: "Ahmad" },
    { reg: 3, name: "Hassan" },
  ];
```

```
<>
  <h1>Name of Students</h1>
  <ul>
    {studens.map((item) => (
      <StudentNames key={item.reg} name={item.name} />
    ))}
  </ul>
</>
```

React Event Object

```
const shoot = (name, e) => {
  alert("My name is " + name + "I clicked the event " + e.type);
};
```

```
<button onClick={(e) => shoot("Hamza", e)}> Click on Button </button>
```

React List

```
function StudentNames(nameof) {
  return <li> {nameof.name}</li>;
}
export default StudentNames;
```

```
function ButtonComp(probs) {
  const studens = [
    { reg: 1, name: "Ali" },
    { reg: 2, name: "Ahmad" },
    { reg: 3, name: "Hassan" },
  ];
```

```
<>
  <h1>Name of Students</h1>
  <ul>
    {studens.map((item) => (
      <StudentNames key={item.reg} name={item.name} />
    ))}
  </ul>
</>
```

Adding Forms in React

- In HTML, form data is usually handled by the DOM.
- In React, form data is usually handled by the components.
- When the data is handled by the components, all the data is stored in the component state.
- We can control changes by adding event handlers in the onChange attribute.
- We can use the useState Hook to keep track of each inputs value.

React Hooks

- Hooks generally replace class components, there are no plans to remove classes from React.
- Hooks allow function components to have access to state and other React features.
- We must import Hooks from react.
- **Hook Rules**
  - Hooks can only be called inside React function components.
  - Hooks can only be called at the top level of a component.
  - Hooks cannot be conditional.

React useState Hooks

- React useState Hook allows us to track state in a function components.
- State generally refers to data or properties that need to be tracking in an application.

Import `useState`

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

Initialize `useState`

- useState accepts an initial state and returns two values:
- The current state.
  - A function that updates the state. a function components.
  - A variable that generally refers to data or properties that need to be tracking in an application.

```
const [getCount, setCount] = useState(0);
Comment Code
export default function UseStateComp() {
  return (
```

React useState Hooks

```
const [getCount, setCount] = useState(0);
Comment Code
export default function UseStateComp() {
  return (
```

- we are destructuring the returned values from useState.
- The first value, **getcount**, is our current state.
- The second value, **setCount**, is the function that is used to update our state.
- We can now include our state anywhere in our component.

```
const [color, setColor] = useState("red");

return <h1>My favorite color is {color}!</h1>
```

Update State

```
<h1>My favorite color is {color}!</h1>
<button
  type="button"
  onClick={() => setColor("blue")}
>Blue</button>
```

```
<button
  onClick={() => {
    setCount(getCount + 1);
  }}
```

```
<button disabled={getCount == 0}
onClick={() => setCount(getCount - 1)}
>-</button>
```

React Router

- Create React App doesn't include page routing.
- React Router is a library for React that enables navigation and routing.
- It allows us to define routes, map them to different components, and handle navigation between them without reloading the page.
- First, install React router in project

```
npm install react-router-dom
```