```
//9.**********************************promise*****************************
*******

  // Example 1:
  // var data = new Promise(function(res,rej){
  //     setTimeout(function(){
  //         res("data");
  //     },2000)
  // })
  // console.log(data);
  // data.then(function(val){
  //     console.log("Val:" + val);
  // })

  // Example 2:
  // let promise = new Promise(function(resolve, reject) {
  //     setTimeout(() => resolve("done!"), 1000);
  //   });

  //   // resolve runs the first function in .then
  //   promise.then(
  //     result => console.log("Hello" + result), // shows "done!" after 1
second
  //     error => console.log(error) // doesn't run
  //   );
  //***********************************************


  //
**********************************10.async/await************************
**********
  // async function functionName(){
  //     console.log("INSIDE...");
  //     const response = await
fetch('https://fakestoreapi.com/products/');
  //     console.log("Before Response...");
  //     const data = await response.json();
  //     console.log("Data Resolved...");
  //     return data;
  // }
  // console.log("Before Calling the Function");
  // let a = functionName();
  // console.log("After Calling the Function");
  // console.log(a);
  // a.then(data => console.log(data));
  // console.log("End of the Program!!!");
```

```
  //************************************************



  //
**********************************11.axios*******************************
****
  // axios.get('https://fakestoreapi.com/products/').then(res=>{
  //     console.log(res)
  // })

  // async function doPostRequest() {
  //     let payload = { name: 'John Doe', occupation: 'gardener' };
  //     let res = await axios.post('http://httpbin.org/post', payload);
  //     let data = res.data;
  //     console.log(data);
  // }
  // doPostRequest();
  //************************************************
```

React (Events + Props + Hooks)

App.js

```
import "./App.css"
// import CountClassComp from "./Components/CountClassComp";
// import FuncComponent from "./Components/FuncComponent";
import Navbar from './Components/Navbar';
import TextForm from "./Components/TextForm";
import Counter from "./Components/Counter";

function App() {
  return (
  <>
    <Navbar/>
    <TextForm heading="Word Count"></TextForm>
    {/* <Counter></Counter> */}
    {/* <FuncComponent></FuncComponent> */}
    {/* <CountClassComp></CountClassComp> */}
  </>

  )
}

export default App;
```

Counter.js

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

export default function Counter() {

  const [count, setCount] = useState(0);
  const [state, setState] = useState({num:4, theme:'blue'});
  const num = state.num
  const theme = state.theme

  function decrementCount() {
    // setCount(count - 1);
    // setCount(count - 1);

    //setCount(preCount => preCount - 1);
    setCount(preCount => preCount - 1);
    setState(preState => {
      return {count: preState.num - 1}
    })
  }

  function incrementCount() {
    setCount(preCount => preCount + 1);
  }

  return (
    <>
      <button onClick={decrementCount}>-</button>
      <span>{count}</span>
      <span>{num}</span>
      <span>{theme}</span>
      <button onClick={incrementCount}>+</button>
    </>
  );
}
```

Navbar.js

```
import React from 'react'
import propTypes from 'prop-types';

export default function Navbar(props) {
  return (
      <>
   <nav className="navbar navbar-expand-lg navbar-dark bg-dark">
  <div className="container-fluid">
    <a className="navbar-brand" href="/">{props.title}</a>
```

```
    <button className="navbar-toggler" type="button"
data-bs-toggle="collapse" data-bs-target="/navbarSupportedContent"
aria-controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation">
      <span className="navbar-toggler-icon"></span>
    </button>
    <div className="collapse navbar-collapse" id="navbarSupportedContent">
      <ul className="navbar-nav me-auto mb-2 mb-lg-0">
        <li className="nav-item">
          <a className="nav-link active" aria-current="page"
href="/">{props.home}</a>
        </li>
        <li className="nav-item">
          <a className="nav-link" href="/">Link</a>
        </li>
        <li className="nav-item dropdown">
          <a className="nav-link dropdown-toggle" href="/"
id="navbarDropdown" role="button" data-bs-toggle="dropdown"
aria-expanded="false">
            Dropdown
          </a>
          <ul className="dropdown-menu" aria-labelledby="navbarDropdown">
            <li><a className="dropdown-item" href="/">Action</a></li>
            <li><a className="dropdown-item" href="/">Another
action</a></li>
            <li><hr className="dropdown-divider"/></li>
            <li><a className="dropdown-item" href="/">Something else
here</a></li>
          </ul>
        </li>
        <li className="nav-item">
          <a className="nav-link disabled" href="/" tabindex="-1"
aria-disabled="true">Disabled</a>
        </li>
      </ul>
      <form className="d-flex">
        <input className="form-control me-2" type="search"
placeholder="Search" aria-label="Search"/>
        <button className="btn btn-outline-success"
type="submit">Search</button>
      </form>
    </div>
  </div>
</nav>
   </>

  )
```

```
}

Navbar.propTypes = {title: propTypes.string.isRequired}

Navbar.defaultProps = {
    title: 'Set title here'
};
```

TextForm.js

```
import React , {useState} from 'react'

export default function TextForm(props) {
  const [text, setText] = useState('Text Input from useState');
  const handleClick = ()=>{
    let newText = text.toUpperCase();
    setText(newText);
    console.log("Ok button clicked");
  }

  const clearText = ()=>{
    let newText = " ";
    setText(newText);
  }
  const handleOnChange = (e)=>{
    setText(e.target.value);
    console.log("On Change");
  }
  //text = "xyz" wrong way to change the state
  //setText("xyz") correct way to change the state
  return (
    <div className='container'>
        <h1>{props.heading}</h1>
        <div className="mb-3">
        <label htmlfor="mybox" className="form-label"></label>
        <textarea className="form-control" value={text}
onChange={handleOnChange} id="mybox" rows="8"></textarea>
        </div>
        <button className='btn-primary mx-1'
onClick={handleClick}>OK</button>
        <button className='btn-primary mx-1'
onClick={clearText}>Clear</button>
        <div className='container my-3'>
            <h1>Your Text Summary</h1>
            <p>{text.split(" ").length} words and {text.length}
characters</p>
```

```
        </div>
      </div>
    );
}
```

React (Typescript + Class Component + Function Component)

AppClass.tsx

```
import React, { Component } from 'react';

interface AppClassProps {
  emotion: string;
}

interface AppClassState {
  name: string;
  age: number;
  isMale: boolean;
}

class AppClass extends Component<AppClassProps, AppClassState> {
  constructor(props: AppClassProps) {
    super(props);
    this.state = {
      name: " ",
      age: 100,
      isMale: true,
    };
  }

  componentDidMount() {
    this.setState({
      name: "Saif"
    });
  }

  render() {
    const { name, age, isMale } = this.state;
    const { emotion } = this.props;

    return (
```

```
      <div>
        <h1>Hello World</h1>
        <h1>My name is {name}</h1>
        <h1>I am {age} years old</h1>
        <h3>I am {isMale ? "Male" : "Female"}</h3>
        <h4>I am feeling {emotion}</h4>
      </div>
    );
  }
}

export default AppClass;
```

AppFunc.tsx

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

interface AppFuncProps {
  emotion: string;
}

function AppFunc(props: AppFuncProps) {
  const [name, setName] = useState("");
  const [age, setAge] = useState(25);
  const [isMale, setIsMale] = useState(true);

  return (
    <div>
      <h1>Hello World</h1>
      <h1>My name is {name}</h1>
      <h1>I am {age} years old</h1>
      <h3>I am {isMale ? "Male" : "Female"}</h3>
      <h4>I am feeling {props.emotion}</h4>
    </div>
  );
}

export default AppFunc;
```

App.tsx

```
import './App.css'
import AppClass from './AppClass'
import AppFunc from './AppFunc'
```

```
// Functional Component
function App() {
  return (
    <div>
    <AppClass emotion={"Happy"}/>
    <AppFunc emotion={"Happy"}/>
    </div>
  )
}

export default App
```

AppClassJS.js

```
import React from 'react';
class AppClass extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      name: " ",
      age: 27,
      isMale: true,
    };
  }

  render() {

    this.setState({
        name: "Saif"
    })

    const { name, age, isMale } = this.state;
    const { emotion } = this.props;
    return (
        <div>
        <h1>Hello World</h1>
        <h1>My name is {name}</h1>
        <h1>Iam {age} years old</h1>
        <h3>Iam {isMale ? "Male" : "Female"}</h3>
        <h4>Iam feeling {emotion}</h4>
    </div>
    )
  }
}
```

```
export default AppClass;
```

AppFuncJS.js

```
// Two Fundamentals of React:
// 1. props
// 2. State

//Functional Component

import './App.css'
import { useState } from 'react'
function AppFunc(props) {
  const [name, SetName] = useState("");
  const [age, SetAge] = useState(25);
  const [isMale, setIsMale] = useState(true);
  return (
    <div>
    <h1>Hello World</h1>
        <h1>My name is {name}</h1>
        <h1>Iam {age} years old</h1>
        <h3>Iam {isMale ? "Male" : "Female"}</h3>
        <h4>Iam feeling {props.emotion}</h4>
    </div>
  )
}
export default AppFunc
```

Node JS - Routers

Index.js

```
const express = require('express');
const app = express();

const userRoute = require('./Routes/User');
app.use('/user', userRoute);


app.listen(3005,() => {
    console.log("Server Listening on port 3005")
});
```

UserController.js

```
module.exports = {
    GET : (req, res) => {
        res.send("Get ");
    },
    GETone : (req, res) => {
        res.send("Get Clothes");
    },
    GETtwo : (req, res) => {
        res.send("Get Books");
    },
    POST : (req, res) => {
        //User Add Functionality
    },
    PUT : (req, res) => {
        //User Update Functionality
    },
    DELETE : (req, res) => {
        //User Delete Functionality
    }
}
```

User.js

```
const express = require("express");
const router = express.Router();
const controller = require("../Controllers/UserController")

router.get("/get", controller.GET);
router.get("/getOne", controller.GETone);
router.get("/getTwo", controller.GETtwo);

router.post("/addUser", controller.POST);
router.put("/updateUser", controller.PUT);
router.delete("/deleteUser", controller.DELETE);


module.exports = router;
```

Node JS - Cookies and Session

App.js

```
const express = require('express');
const cookieParser = require('cookie-parser');
```

```
const session = require('express-session')

const app = express();

// Cookie
app.use(cookieParser());
app.get('/', (req,res) => {
    res.cookie('Cookie_token_name : Testing Cookie', 'encrypted cookie
strong value', {
        maxAge: 5000,
        expires: new Date(),
        secure: true,
        httponly: true,
        sameSite: 'Lax'
    });
    res.send('Welcome to simple HTTP cookie and cookie send
successfully');
    console.log(req.cookies);
});
app.get('/deleteCookie', (req, res) => {
    res.clearCookie();
    res.send("cookies has been deleted successfully");
});


// Session
// app.use(session({
//     secret: "Your_Secret_Key",
//     resave: true,
//     saveUninitialized: true,
// }));
// app.get("/", function(req,res){
//     req.session.name = "WEB7B"
//     return res.send("Session Set")
// })
// app.get("/session", function(req,res){
//     var sessionName = req.session.name;
//     return res.send(sessionName)
// })


app.listen(8080, function(error)
{   if(error) throw error
    console.log('The server is running on port 8080...');
});
```

BOOK SYSTEM

books.js:

```
const mongoose = require('mongoose');

const BooksSchema = new mongoose.Schema({
   title:
   {
      type: String,
      required: true
   },

   genre:
   {
      type: String,
      required: true
   },

   author:
   {
      type: String,
      required: true
   }
})

const Books = mongoose.model('books', BooksSchema);
module.exports = Books;
```

---

db.js:

```
const mongoose = require("mongoose");
const mongoURI = "mongodb://localhost:27017/books";

const connectToMongo = async () => {
  try {
    mongoose.set("strictQuery", false);
    mongoose.connect(mongoURI);
    console.log("Connected to Mongo Successfully!");
  } catch (error) {
    console.log(error);
  }
};
module.exports = connectToMongo;
```

---

index.js

```
require("dotenv").config();
const express = require("express");
```

```
const bodyParser = require("body-parser");
const cors = require("cors");
const { body, validationResult } = require("express-validator");
const app = express();
const port = process.env.PORT;
const Books = require("./books");
const connectToMongo = require("./db");
connectToMongo();

app.use(cors());
app.use(bodyParser.json({ limit: "10mb" }));
app.use(bodyParser.urlencoded({ extended: true, limit: "10mb" }));
app.use(express.json());

app.post("/addBook", async (req, res) => {
  const { title, genre, author } = req.body;

  const book = await Books.create({
    title,
    genre,
    author,
  });
  return res.json(book);
});

app.get("/getBooks", async (req, res) => {
  try {
    const books = await Books.find();
    return res.json({ books });
  } catch (err) {
    return res.status(500).json({ error: "Unexpected error occured." });
  }
});

app.get("/getBook/:id", async (req, res) => {
  try {
    const { id } = req.params;
    const book = await Books.findById(id);
    return book
      ? res.json(book)
      : res.json({ error: "The book does not exist." });
  } catch (err) {
    return res.status(500).json({ error: "Unexpected error occured." });
  }
});
```

```
app.delete("/deleteBook/:id", async (req, res) => {
  try {
    const { id } = req.params;
    const book = await Books.findById(id);
    if (!book) {
      return res.status(404).json({ error: "Book not found" });
    }
    await book.delete();

    res.json({ message: "Book deleted successfully" });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: "Internal server error" });
  }
});

app.put("/updateBook/:id", async (req, res) => {
  try {
    const { id } = req.params;
    const { title, author, genre } = req.body;

    const book = await Books.findById(id);

    if (!book) {
      return res.status(404).json({ error: "Book not found" });
    }
    book.title = title;
    book.author = author;
    book.genre = genre;
    await book.save();

    res.json({ message: "Book updated successfully" });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: "Internal server error" });
  }
});

app.get("/", (req, res) => {
  res.send("Hello World");
});

app.listen(process.env.PORT || port, () => {
  console.log(`Example app listening on port ${port}`);
});
```