

# 🌟 FULL REACT COURSE NOTES

## HTML – The Structure of Every Webpage

### What is HTML?

HTML (Hyper Text Markup Language) gives **structure** to a webpage.

Basic Example:

```
<h1>Hello World</h1>
<p>This is a paragraph.</p>
<button>Click Me</button>
```

### Important HTML Elements

#### Headings

```
<h1>Main Title</h1>
<h2>Sub Title</h2>
```

#### Paragraph

```
<p>This is a paragraph showing details.</p>
```

#### Image

```

```

#### Link

```
<a href="https://google.com">Visit Google</a>
```

## Input

```
<input type="text" placeholder="Your Name" />
```

## List

```
<ul>
  <li>HTML</li>
  <li>CSS</li>
</ul>
```

HTML gives **structure**, CSS will give **style**, and JavaScript will give **life (interactivity)**.

---

---



# CSS – Styling the Web

---

---

## What is CSS?

CSS (**Cascading Style Sheets**) is used to **style** HTML elements.

Example:

```
h1 {
  color: blue;
  font-size: 40px;
}
```

## Most Important CSS Concepts

### Colors

```
color: red;  
background-color: black;
```

## Margin & Padding

```
margin: 10px;  
padding: 15px;
```

## Borders

```
border: 1px solid black;
```

## Flexbox (must know for React UI)

```
display: flex;  
justify-content: center;  
align-items: center;
```

## CSS Class

```
<div class="box">Hello</div>  
  
<style>  
.box {  
  background: yellow;  
  padding: 10px;  
}  
</style>
```

CSS improves layout and design — important when building modern React UIs.

---

---

# JavaScript – The Programming Language of the Web

---

---

JavaScript is the brain of a web app. React uses JavaScript heavily, so basics are important.

---

## Variables

```
let student = "Munees";
const age = 22;
```

---

## Data Types

- String
  - Number
  - Boolean
  - Array
  - Object
  - Null
  - Undefined
- 

## Arrays

```
const fruits = ["Apple", "Banana", "Orange"];
```

---

## Objects

```
const user = {
  name: "Munees",
  age: 22,
};
```

---

## Functions

```
function greet() {
  console.log("Hello");
}
```

## Arrow Function

```
const greet = () => console.log("Hello");
```

## Conditions

```
if (age > 18) {  
    console.log("Adult");  
} else {  
    console.log("Minor");  
}
```

## Loops

```
for (let i = 0; i < 5; i++) console.log(i);
```

## Array functions (VERY important for React)

### .map()

```
[1,2,3].map(n => n * 2); // [2,4,6]
```

### .filter()

```
[1,2,3].filter(n => n > 1); // [2,3]
```

### .reduce()

```
[1,2,3].reduce((sum, n) => sum + n, 0); // 6
```

These three functions are the foundation of React list rendering and data processing.

# Introduction to React

---

---

React is a **JavaScript library** used to build **fast, modern, interactive websites**.

---

## Why React?

- ✓ Super fast updates ✓ Component-based ✓ Reusable code ✓ Handles UI changes efficiently ✓ Easy to scale
- 

## What is JSX?

JSX = JavaScript + HTML

Example:

```
<h1>Hello React</h1>
```

---

## React Component

A small reusable block of UI.

```
function Welcome() {  
  return <h1>Hello!</h1>;  
}
```

Use inside App:

```
<Welcome />
```

---

## Props – Sending Data to Components

---

---

Props allow parent components to send data to children.

Example:

```
<Welcome name="Munees" />
```

Child:

```
function Welcome({ name }) {
  return <h2>Hello {name}</h2>;
}
```

Props are **read-only**.

---

---

## State – Dynamic Data in React

---

---

State allows values to change and update UI automatically.

Example:

```
const [count, setCount] = useState(0);
```

Increment:

```
setCount(count + 1);
```

## Handling Forms in React (Controlled Inputs)

---

---

A controlled input is an input controlled by React state.

```
const [name, setName] = useState("");  
  
<input  
  value={name}  
  onChange={(e) => setName(e.target.value)}  
/>
```

## Submit Form Example

```
const handleSubmit = (e) => {  
  e.preventDefault();  
  console.log(name);  
};
```

# Creating a To-Do Application

States:

```
const [task, setTask] = useState("");  
const [tasks, setTasks] = useState([]);
```

Add:

```
setTasks([...tasks, task]);
```

Delete:

```
setTasks(tasks.filter((_, i) => i !== index));
```

List:

```
tasks.map((t, i) => <p key={i}>{t}</p>)
```

---

---

## Fetching APIs in React

---

---

What is an API?

An API gives data from the internet.

---

### Fetch Example

```
const res = await fetch("https://api.example.com");
const data = await res.json();
```

---

### Loading and Error Handling

```
const [loading, setLoading] = useState(true);
const [error, setError] = useState(null);
```

Display:

```
{loading && <p>Loading...</p>}
{error && <p>Error occurred</p>}
```

---

### Example: Thirukkural API

```
const res = await fetch("https://api-thirukkural.vercel.app/api?num=1");
```

---

### Example: Pokedex Fetch

```
const res = await fetch("https://pokeapi.co/api/v2/pokemon/pikachu");
```

---

---

## Conditional Rendering in React

---

---

```
{isLoggedIn ? <Dashboard /> : <Login />}
```

---

---

## Building a Game (Tic Tac Toe)

---

---

Board State:

```
const [board, setBoard] = useState(Array(9).fill(null));
const [isXTurn, setIsXTurn] = useState(true);
```

Winner Logic:

```
const patterns = [
  [0,1,2], [3,4,5], [6,7,8],
  [0,3,6], [1,4,7], [2,5,8],
  [0,4,8], [2,4,6]
];
```

This teaches: ✓ State ✓ Arrays ✓ Conditions ✓ UI updates

---

---

# Expense Tracker Application

---

---

This app teaches:

- Forms
  - State
  - List rendering
  - Filters
  - reduce()
  - Category selection
  - Deleting items
  - Calculating totals
- 

## States:

```
const [title, setTitle] = useState("");
const [amount, setAmount] = useState("");
const [category, setCategory] = useState("Food");
const [expenses, setExpenses] = useState([]);
```

## Add Expense

```
setExpenses([...expenses, {
  id: Date.now(),
  title,
  amount: Number(amount),
  category
}]);
```

## Total Calculation

```
const total = expenses.reduce((sum, e) => sum + e.amount, 0);
```

## Category Filter

```
filtered = expenses.filter(e => e.category === filter);
```

---

---

## Hosting a React Application

---

---

React apps need to be built before hosting:

Create Build:

```
npm run build
```

Folder generated:

```
build/
```

---

## Hosting on Netlify (Recommended for beginners)

---

Method 1 – Drag & Drop

1. Go to Netlify
2. Click **Deploy manually**
3. Upload **build folder**
4. Live website is ready

---

Method 2 – GitHub Auto Deployment

1. Push project to GitHub
2. Netlify → Import from GitHub
3. Set:

```
build command: npm run build  
publish directory: build
```

#### 4. Deploy

Every Git push auto-updates website.

---

## Other Hosting Options

---

### Vercel

- Best for React & Next.js
- Auto Git deployment

### GitHub Pages

- Free forever
- Simple static hosting

### Firebase Hosting

- Best when using Firebase backend

### Render

- Free
  - Good for fullstack hosting
-