**Component**

React-এর component হলো React এর সবচেয়ে গুরুত্বপূর্ণ অংশগুলোর একটি। Component-কে তুমি একটি পুনর্ব্যবহারযোগ্য (reusable) কোড ব্লক বলতে পারো যা ইউজার ইন্টারফেসের (UI) নির্দিষ্ট একটি অংশ তৈরি করে।

নিচে বাংলায় সহজভাবে React Component এর বিষয়গুলো ব্যাখ্যা করছি:

**🔹 Component কি?**

Component হলো একটি ফাংশন বা ক্লাস যা HTML-এর মত JSX কোড রিটার্ন করে এবং সেটি ব্রাউজারে দেখায়। প্রতিটি Component আলাদা আলাদা কাজের জন্য বানানো হয়, যাতে কোডটা পরিষ্কার ও মেইনটেইনযোগ্য হয়।

**🔹 Component এর ধরন**

React-এ প্রধানত দুই ধরণের Component আছে:

1. **Functional Component (সবচেয়ে বেশি ব্যবহৃত)**  
   এটা একটা সিম্পল JavaScript function যা JSX রিটার্ন করে।

✅ উদাহরণ:



1. **Class Component (পুরোনো পদ্ধতি)**  
   এটি একটি ক্লাস যা React.Component থেকে extend করে এবং render() মেথডে JSX রিটার্ন করে।

✅ উদাহরণ:



**🔹 JSX কি?**

JSX হলো JavaScript এর ভিতরে HTML টাইপ কোড লেখার একটা উপায়। React Component JSX ব্যবহার করে UI তৈরি করে।

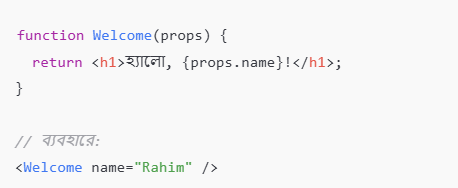
✅ উদাহরণ:



**🔹 Props (Properties)**

Props হলো Component-এর মধ্যে ডেটা পাঠানোর উপায়। এক Component থেকে আরেকটিতে ডেটা পাঠাতে Props ব্যবহার করা হয়।

✅ উদাহরণ:



**🔹 State**

State হলো Component-এর নিজস্ব ডেটা। এটি পরিবর্তনযোগ্য এবং Component-এর UI তে প্রভাব ফেলে।

✅ উদাহরণ (Functional Component এ):



**🔹 Component ব্যবহার কিভাবে করে?**

Component গুলোকে অন্য Component-এর ভিতরে কল করে ইউজার ইন্টারফেস তৈরি করা হয়।

✅ উদাহরণ:



**The rules of JSX**

JSX (JavaScript XML) হলো React-এর একটি বিশেষ syntax যা HTML এর মতো দেখতে হলেও এটি JavaScript এর ভিতরে লেখা হয়। JSX ব্যবহার করলে UI component তৈরি করা অনেক সহজ ও বুঝতে সুবিধা হয়। তবে JSX লিখার কিছু নির্দিষ্ট নিয়ম (rules) আছে যা জানা খুব জরুরি।

নিচে JSX-এর প্রধান নিয়মগুলো বাংলায় ব্যাখ্যা করলাম:

## ✅ JSX-এর নিয়মাবলি (Rules of JSX in Bangla):

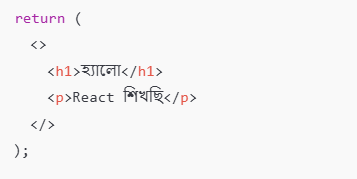
### 🔹 ১. একটাই Parent Element থাকতে হবে

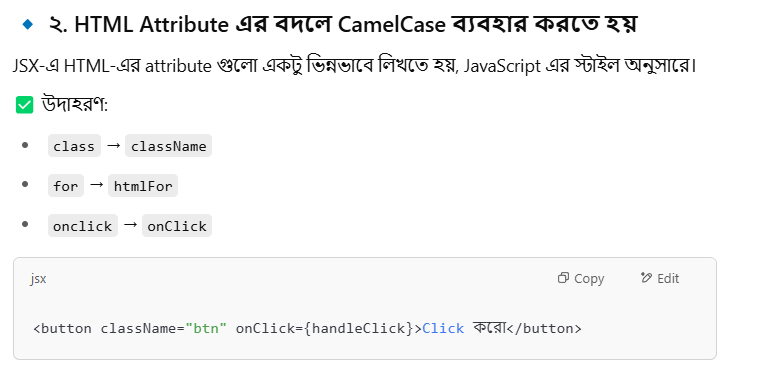
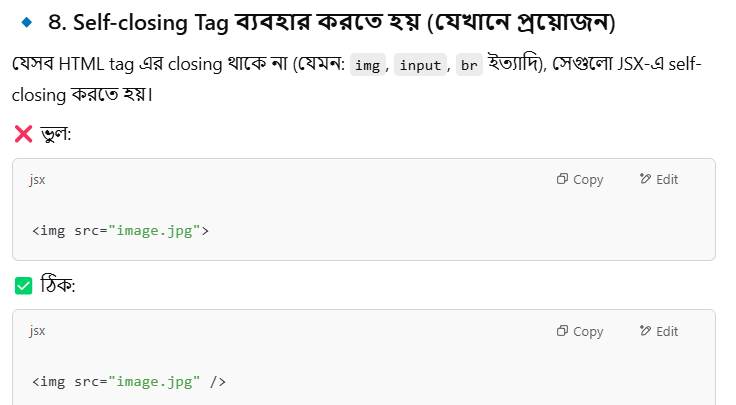
JSX-এ সবকিছু একটাই parent tag এর ভিতরে রাখতে হয়।

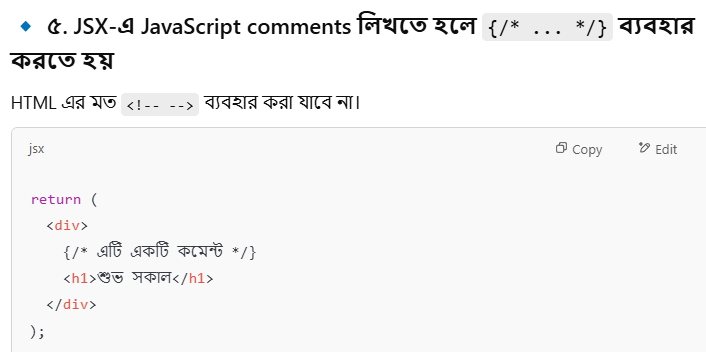
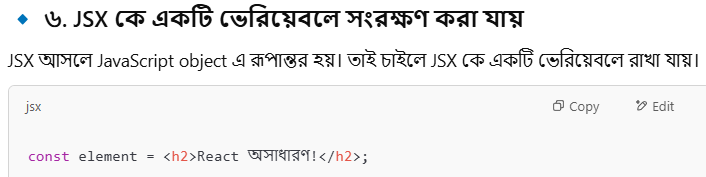
❌ ভুল:

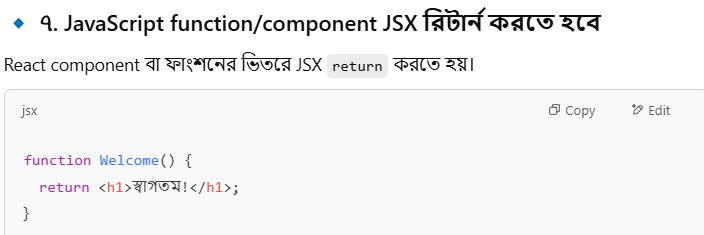


বা, React Fragment ব্যবহার করেও করা যায়:





**Best Practice**

অবশ্যই! JSX-এ কাজ করার সময় কিছু **Best Practices** (সেরা অভ্যাস) অনুসরণ করলে কোডটা হবে আরও **পরিষ্কার, সুন্দর, এবং maintainable**। নিচে React JSX লেখার সময় যেসব best practices মানা উচিত, সেগুলো বাংলায় দিলাম:

## ✅ JSX Best Practices (JSX লেখার সেরা অভ্যাস) – বাংলায়

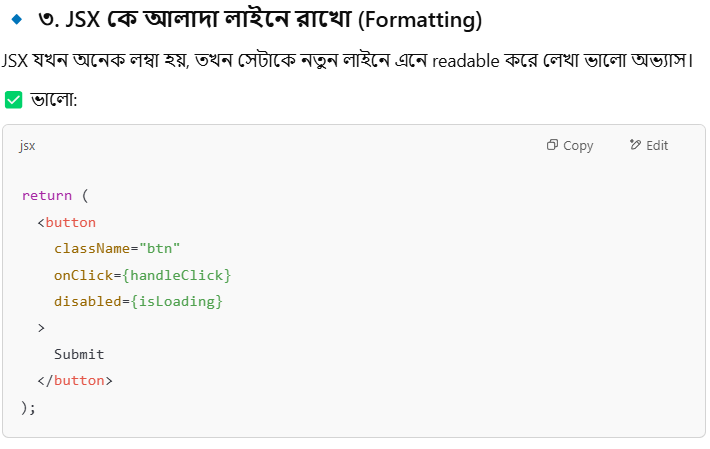
### 🔹 ১. Meaningful Component Name ব্যবহার করো (PascalCase)

React Component এর নাম সবসময় **Capital Letter** দিয়ে শুরু হওয়া উচিত এবং **PascalCase** ফরম্যাটে লেখা উচিত।



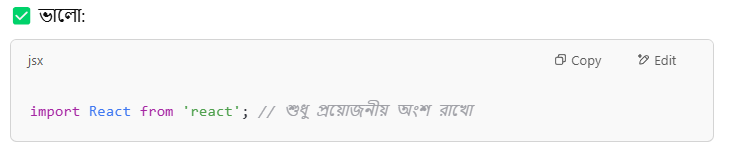
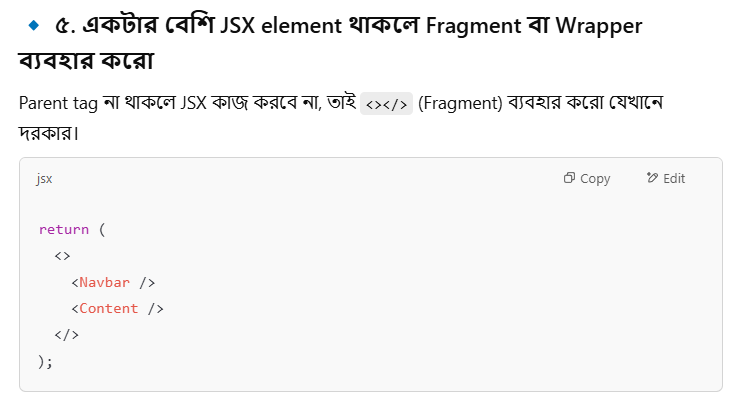
### 🔹 ২. ছোট Component বানাও (Reusability & Clean Code)

যদি Component অনেক বড় হয়ে যায়, তাহলে সেটাকে ছোট ছোট reusable Component-এ ভাগ করে ফেলো।

### 🔹 ৪. ব্যবহৃত না হওয়া কোড বাদ দাও (Remove Unused)

অপ্রয়োজনীয় import, variable, comment ইত্যাদি রেখে কোড নোংরা কোরো না।

### 🔹 ৬. Meaningful & Accessible HTML Tag ব্যবহার করো

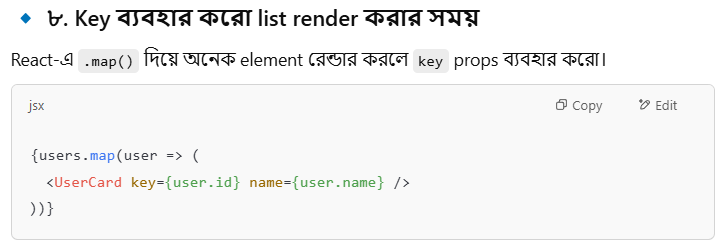
div দিয়ে পুরো app না বানিয়ে, context অনুযায়ী semantic HTML tag ব্যবহার করো (যেমন: header, main, footer, section ইত্যাদি)।

### 🔹 ৭. Inline styling কম ব্যবহার করো

Inline CSS না দিয়ে CSS ক্লাস অথবা styled-component / Tailwind CSS / CSS Module ব্যবহার করো।







### 🔹 ৯. Single Responsibility Principle মানো

প্রতিটি Component যেন একটি নির্দিষ্ট কাজেই ফোকাস করে। Component যদি অনেক কিছু করে, সেটা ভেঙে ছোট ছোট অংশে ভাগ করো।

### 🔹 ১০. PropTypes বা TypeScript ব্যবহার করে Props Validate করো

JSX এ যেসব props পাস করো, সেগুলোর ধরন নির্দিষ্ট করে দিলে future bug কমে।



**Props**

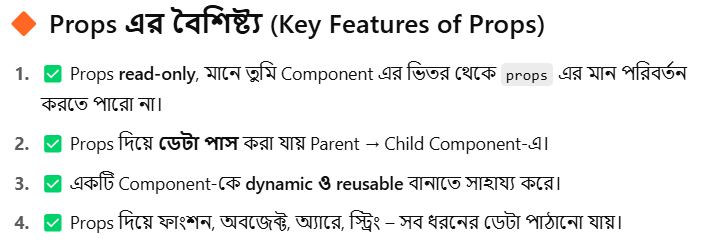
React-এর অন্যতম গুরুত্বপূর্ণ বিষয় হচ্ছে **Props**। এটা শেখা মানেই React Component-এর মধ্যে **ডেটা আদান-প্রদান** বোঝা, যেটা প্রতিদিনের কাজে অনেক কাজে লাগে।

## 🔷 Props কি? (What are Props in Bangla)

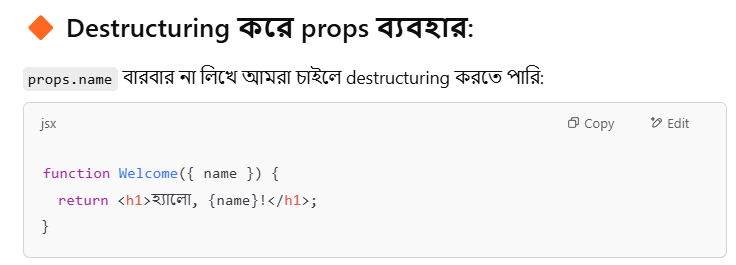
**Props** শব্দটা এসেছে **"Properties"** থেকে।  
React-এ Props ব্যবহার করে **এক Component থেকে অন্য Component-এ ডেটা পাঠানো** যায়। এটি হচ্ছে **React Component-এর ইনপুট**।

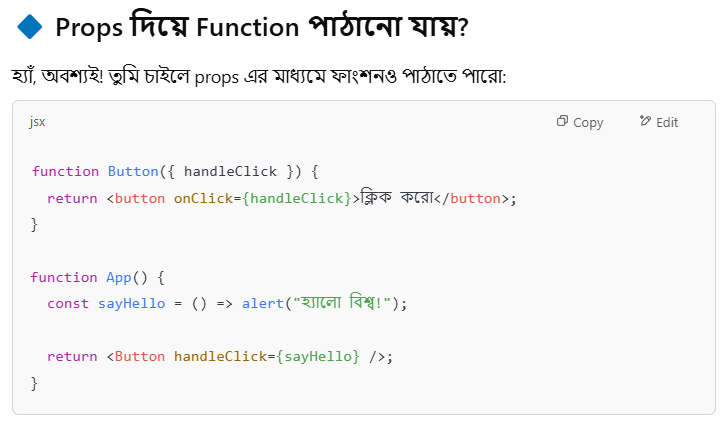
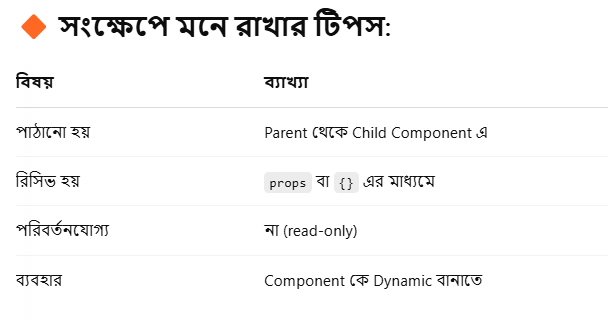
তুমি যেভাবে ফাংশনে প্যারামিটার পাঠাও, ঠিক সেভাবেই Component-এ Props পাঠানো হয়।









**Rendering**

React এর সবচেয়ে গুরুত্বপূর্ণ বিষয়গুলোর মধ্যে একটা হলো **Rendering**। এটা বুঝলে তুমি জানবে **React কিভাবে UI বানায়, আপডেট করে এবং দেখায়**।

## 🔷 Rendering মানে কী? (What is Rendering in Bangla)

**Rendering** মানে হচ্ছে –  
React Component গুলো থেকে তৈরি JSX/HTML কে **Browser-এ দেখা যায় এমন UI তে রূপান্তর করা**।

সহজ করে বললে, তুমি React Component এ যেটা লিখো, সেটা কিভাবে ইউজার দেখবে – সেটাই Rendering।

## 🔸 দুই ধরনের Rendering আছে:

### 1️⃣ ****Initial Rendering****

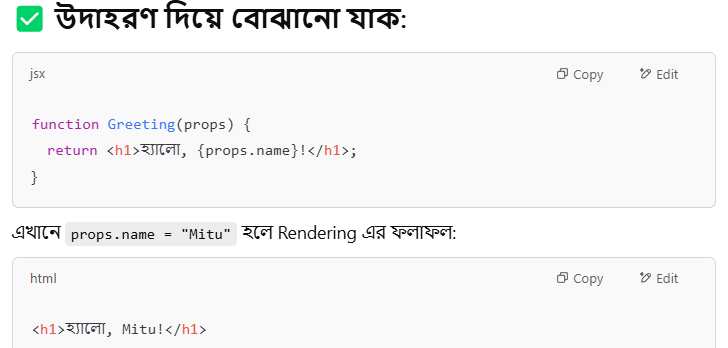
এটা তখন ঘটে, যখন React App প্রথমবার লোড হয়।

🧠 React DOM Component কে HTML এ রূপান্তর করে ব্রাউজারে দেখায়।

### 2️⃣ ****Re-Rendering****

এটা তখন ঘটে, যখন Component এর **state** বা **props** বদলায়।

🌀 তখন React শুধু যেসব জায়গায় পরিবর্তন হয়েছে, শুধু সেগুলোই ব্রাউজারে আপডেট করে।

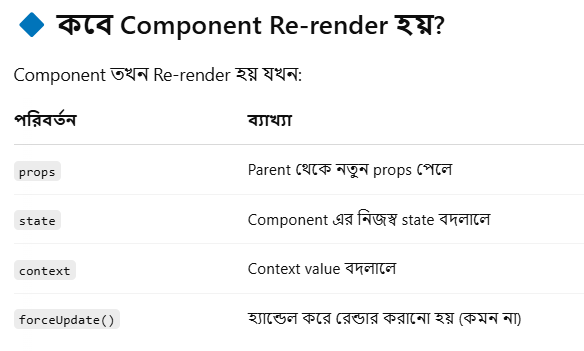


## 🔶 Rendering এর পিছনের মজার প্রযুক্তি – Virtual DOM

React একবারে পুরো ব্রাউজার DOM কে আপডেট করে না।

বরং, এটা **Virtual DOM** নামে একটা lightweight কপি তৈরি করে  
এবং দুইটা DOM এর মধ্যে পার্থক্য বের করে **যেখানে পরিবর্তন হয়েছে শুধু সেখানে update করে**।

🎯 এর ফলে performance অনেক ভালো হয় — fast and efficient UI rendering!



## 🔸 Rendering Flow (সাধারণভাবে):

1. Component তৈরি হয় (Function or Class)
2. JSX → JavaScript object এ রূপান্তর হয়
3. Virtual DOM তৈরি হয়
4. React দেখে কোন অংশ বদলেছে
5. শুধু পরিবর্তিত অংশ Browser DOM এ update হয় (Re-render)

## 🔶 React Rendering Best Practice:

1. ✅ Component গুলো ছোট রাখো
2. ✅ প্রয়োজন ছাড়া State বা Props চেঞ্জ কোরো না
3. ✅ React.memo বা useMemo ব্যবহার করো performance boost করতে
4. ✅ Unique key ব্যবহার করো .map() দিয়ে list render করলে



**HOOKs**

React শেখার পথে **Hooks** হলো একদমই গেম-চেঞ্জার! এটা জানলে তুমি **Function Component দিয়েই সব কিছু করতে পারবে**, আগের মতো আলাদা করে Class Component লেখার দরকার পড়বে না।

চলো, বাংলা ভাষায় সুন্দরভাবে বোঝাই —

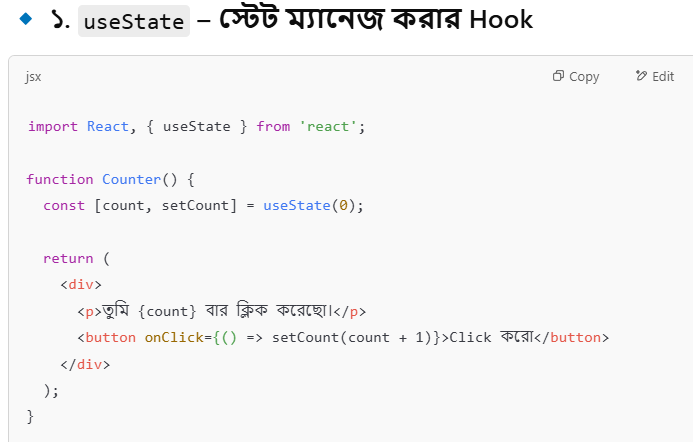
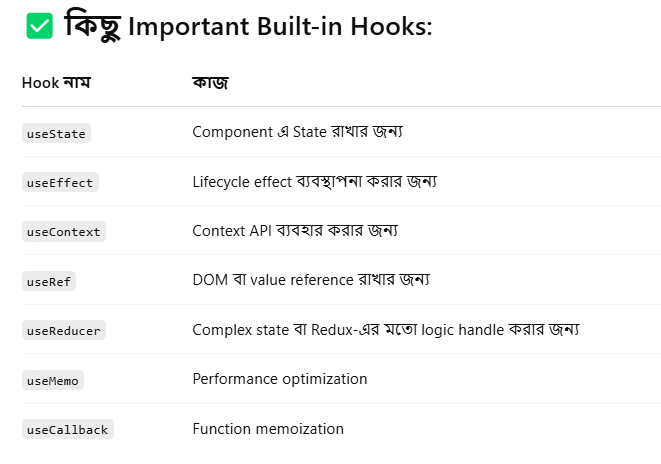
## 🔷 React Hooks কী? (Hooks in Bangla)

**Hooks** হচ্ছে React 16.8 এ আসা একধরনের **special function**, যা দিয়ে তুমি **state** এবং **lifecycle features** ব্যবহার করতে পারো **function component**-এর ভেতরে।

আগে এগুলো শুধু **class component**-এ করা যেতো। কিন্তু এখন শুধু function দিয়েই সব ম্যানেজ করা যায় — কারণ আছে Hooks! 🪝

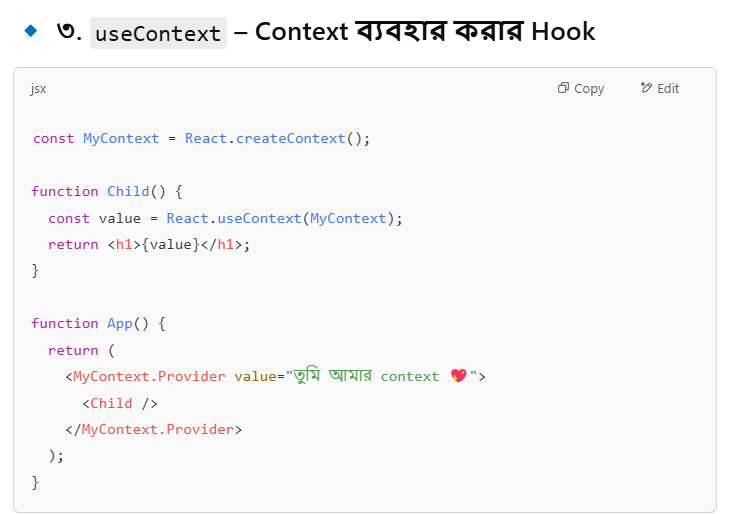
## 🔸 সহজ ভাষায় বলা যায়:

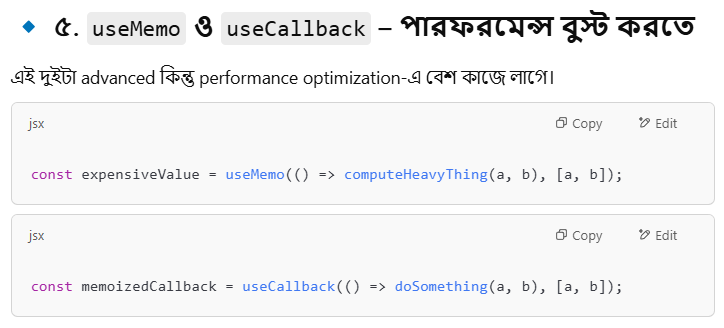
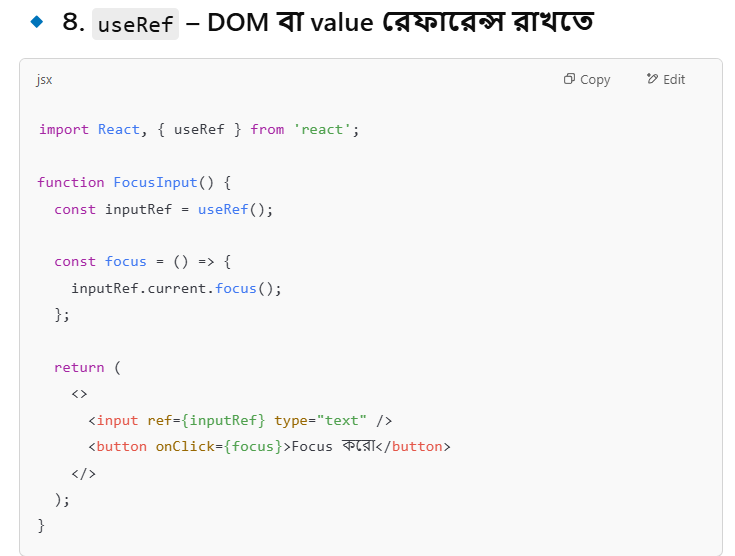
🔁 Hooks মানে হচ্ছে “React Function Component এর ভিতরে **জীবন** আনা।”  
মানে: State, Effect, Context, Reducer, Ref, ইত্যাদি।



🧠 এখানে:

* count হচ্ছে state
* setCount হচ্ছে সেটা আপডেট করার ফাংশন
* useState(0) মানে initial value 0





## 🔸 কিছু Hook ব্যবহারের নিয়ম:

✔️ সবসময় React Component বা Custom Hook এর ভিতরে Call করো  
❌ শর্ত (if-else, loop) এর ভিতরে Call করা যাবে না  
✔️ Custom Hook বানাতে চাইলে নাম use দিয়ে শুরু করতে হবে



**UseState**

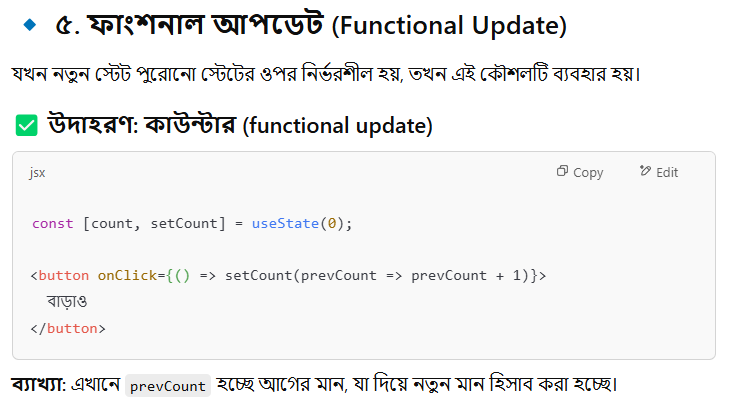
useState React এর সবচেয়ে বেশি ব্যবহৃত Hook — কারণ এর সাহায্যে তুমি তোমার Component-এ **dynamic, changeable data** রাখতে পারো।

এখন আমি useState এর **সবচেয়ে গুরুত্বপূর্ণ ব্যবহারগুলো (categories)** বাংলায় ব্যাখ্যা করব উদাহরণসহ, যাতে তুমি ভালোভাবে বুঝে ফেলো।

## ✅ useState এর সবধরনের ব্যবহার (Categories with Examples in Bangla)

🔹 ৩. অবজেক্ট স্টেট (Object State)







# onChange() এবং onClick() কী?

React-এ onChange() এবং onClick() হলো **ইভেন্ট হ্যান্ডলার**, মানে ইউজার কোনো কাজ করলে (যেমন লেখা, ক্লিক করা), তখন React একটা ফাংশন চালায়।  
এসব ইভেন্ট দিয়ে ইউজারের interaction-এর উপর রেসপন্স করা হয়।

# ১. ➡️ onChange()

* onChange() সাধারণত ইনপুট টাইপের ফিল্ডে ব্যবহার করা হয় (যেমন: <input>, <textarea>, <select> ইত্যাদি)।
* যখন ইউজার ইনপুটের ভ্যালু পরিবর্তন করে, তখন onChange() ফায়ার হয়।



**ব্যাখ্যা:**

* ইউজার ইনপুট ফিল্ডে টাইপ করলে handleChange() ফাংশন চলে।
* event.target.value ইনপুটের বর্তমান ভ্যালু ধরে নেয়।
* সেই ভ্যালু setText() দিয়ে আপডেট করা হয়।
* ফলে স্ক্রিনে রিয়েল-টাইমে লেখা দেখা যায়।

# ২. ➡️ onClick()

* onClick() ব্যবহার হয় যখন ইউজার কোনো বাটন, লিংক, ডিভ ইত্যাদিতে ক্লিক করে।
* ক্লিকের সাথে সাথে নির্দিষ্ট ফাংশন চলে যায়।



**ব্যাখ্যা:**

* যখন বাটনে ক্লিক করা হয়, তখন handleClick() ফাংশন রান করে।
* count এক করে বাড়ে, এবং নতুন ভ্যালু স্ক্রিনে দেখায়।



**Optional Chaining**

React-এ **Optional Chaining (?.)** হচ্ছে JavaScript-এর একটি ফিচার যা React কোডেও খুবই উপকারীভাবে ব্যবহার করা হয়। এটি ব্যবহার করা হয় কোনো অবজেক্ট বা ভেরিয়েবল এর গভীর প্রপার্টি অ্যাক্সেস করার সময় যাতে যদি কোথাও undefined বা null থাকে, তাহলে অ্যাপ্লিকেশন ক্র্যাশ না করে undefined রিটার্ন করে।

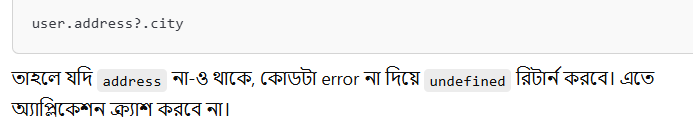
### 🧠 বাংলা ভাষায় সহজভাবে বুঝি:

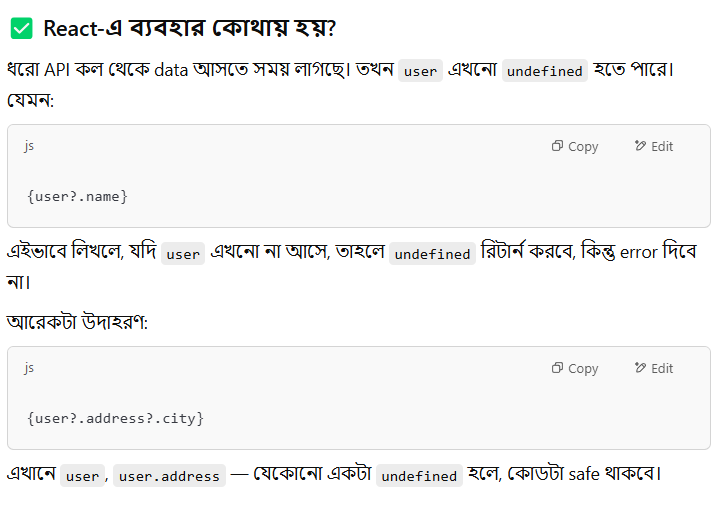
ধরো, তোমার কাছে একটা অবজেক্ট আছে:



তুমি যদি user.address.city অ্যাক্সেস করো, ঠিক আছে। কিন্তু যদি কখনো address না থাকে, মানে undefined হয়, তখন user.address.city লিখলে error দিবে:  
**"Cannot read property 'city' of undefined"**

এখন যদি আমরা optional chaining ব্যবহার করি:





**🔥 উপকারিতা:**

1. কোড ছোট হয়।
2. Error কমে যায়।
3. API ডেটা লোড হওয়া পর্যন্ত অ্যাপ্লিকেশন চলে।

**📝 মনে রাখার নিয়ম:**

🔹 ?. মানে হচ্ছে —  
“এই প্রপার্টি বা ভেরিয়েবলটা আছে তো? থাকলে পরেরটা দেখো, না থাকলে undefined দাও।”