React Basics

🖱️ 1. Handling Events in React

React uses camelCase for events (like onClick, onChange, etc), and you pass a **function**, not a string.

Example:

```
const ClickMe = () => {
  const handleClick = () => {
    alert("You clicked the Bat Button! 🦇");
  };

  return <button onClick={handleClick}>Click Me</button>;
};
```

No onclick="..." like HTML. It’s all JS baby.

🔥 You can also use inline functions:

```
<button onClick={() => console.log("Inline click!")}>Click</button>
```

🔀 2. Conditional Rendering

React lets you show/hide stuff based on conditions. Options:

✅ Ternary operator:

```
const isHero = true;
return <h1>{isHero ? "I'm Batman" : "I'm Bruce Wayne"}</h1>;
```

✅ && operator (show if true):

```
{isHero && <p>This is the Batcave</p>}
```

✅ if statements (outside JSX):

```
let message;
if (isHero) {
```

```
  message = "Justice.";
} else {
  message = "Normal life.";
}
return <h1>{message}</h1>;
```

🔁 3. List Rendering with .map()

Let's say you got a list of names:

```
const heroes = ["Batman", "Superman", "Wonder Woman"];

return (
  <ul>
    {heroes.map((hero, index) => (
      <li key={index}>{hero}</li>
    ))}
  </ul>
);
```

⚠️ Why the key prop?

React needs it to **track** each item uniquely when updating the DOM.

Better than using index? Yup — if you have unique IDs, use those instead.

📝 4. Forms & Controlled Components

In React, form inputs are **controlled** by state. That means the input value is tied to useState.

Example: Controlled input

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

const Form = () => {
  const [name, setName] = useState("");

  const handleSubmit = (e) => {
    e.preventDefault(); // stop page reload
    alert(`Hello, ${name}!`);
  };
```

```
  return (
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        value={name}
        onChange={(e) => setName(e.target.value)}
        placeholder="Enter your name"
      />
      <button type="submit">Submit</button>
    </form>
  );
};
```

🧠 The input is **"controlled"** because React is in charge of its value via state.

🧠 5. useState Hook (Refresher)

You've already touched it, but let's flex a bit more:

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

const Counter = () => {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>🧮 Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Add</button>
      <button onClick={() => setCount(count - 1)}>Subtract</button>
      <button onClick={() => setCount(0)}>Reset</button>
    </div>
  );
};
```

So you just hit:

- Events ✅
- Conditions ✅
- Loops ✅
- Forms ✅

- State ✅

React foundations: **solid af.**

Time to Quizz

Yooo 🔥

No pressure, just vibes and brain gains 🧠💪

⚡ Quiz Time: React Edition (Level: Sidekick 🦇➡️ Hero)

Q1. What's the correct way to handle a button click in React?

A. <button onclick="handleClick()">Click</button>

B. <button onClick={handleClick}>Click</button>

C. <button click="handleClick()">Click</button>

D. <button onclick={handleClick()}>Click</button>

Q2. You want to show the text **"Welcome back!"** *only if* isLoggedIn is true. Which one works?

A. return isLoggedIn ? Welcome back! : null;

B. if (isLoggedIn) return Welcome back! ;

C. return isLoggedIn && Welcome back! ;

D. All of the above

Q3. You have a state like this:

const [text, setText] = useState("");

What happens when you type into this input?

<input value={text} onChange={(e) => setText(e.target.value)} />

A. Nothing

B. It crashes

C. The state updates with your input

D. The input becomes read-only

Q4. You're rendering a list with .map(). What's the main reason for using a key prop?

A. To display index numbers

B. For styling

C. For React to track changes efficiently

D. To sort the list

Q5. In React, which hook is used to **store and update** component data?

A. useEffect()

B. useContext()

C. useMemo()

D. useState()