HOW SOFTWARE WORKS · BUILDER · 15 MIN READ

React basics: components, props and state

Read and fix the React code AI builders write: components, props, state, lists with keys, events, effects, and the errors you'll see most.

Pass the check (4 of 5) online to earn the “Component Builder” badge:
https://digitalbuilders757.com/university/study/react-basics-components-props-state

Components: the building blocks

A React app is a tree of components. A component is a function that returns what to show, written in JSX: HTML-like tags inside JavaScript. For example, function Greeting() { return <h1>Hello!</h1> } is used on a page as <Greeting />. Component names start with a capital letter.

Lovable, v0 and Bolt all write React, so most AI-built apps you open are folders of components: a Header, a TaskList, a SignUpForm. Reading an app starts with finding the component for the part of the screen you care about.

Props: inputs from the parent

Props pass data into a component, like arguments to a function. <Greeting name="Sam" /> sends a name, and function Greeting({ name }) { return <h1>Hello, {name}!</h1> } shows it. Curly braces put a JavaScript value into the JSX.

Props flow down from parent to child and are read-only: a component shouldn't change its own props. If a child needs to change something, the parent passes it a function to call, like onDelete.

State: the component's memory

A plain variable resets every time the component runs again. State remembers. const [count, setCount] = useState(0) gives you the current value and a setter. Calling setCount(count + 1) stores the new value and tells React to draw the component again, so the screen always matches the data.

Never change state directly. count++ or items.push(newItem) changes the old value without telling React, so the screen doesn't update. Always call the setter with a new value, for example setItems([...items, newItem]).

Lists, events and effects

- Lists: items.map((item) => <li key={item.id}>{item.name}). Each item needs a key that is unique and stable, usually an id from your data, so React can tell items apart. Don't use the array index when items can be added, removed or reordered, and never a random number.
- Events: onClick={() => setOpen(true)} passes a function for React to call later. onClick={setOpen(true)} calls it straight away while drawing the screen, which loops.
- Showing things conditionally: {isLoading ? Loading... : <TaskList />}, or {error && {error} }.
- Effects: useEffect runs code after the screen updates, to keep in sync with things outside React such as a timer or a subscription. Wrong dependencies cause stale data or endless loops, and many things don't need an effect at all: work out values while rendering instead.

Errors you'll meet

- "Each child in a list should have a unique key prop": add key={item.id} to the outer element inside the .map.
- "Too many re-renders": something sets state while drawing, often onClick={setX(...)} instead of onClick={() => setX(...)}.
- "Objects are not valid as a React child": you tried to show a whole object. Show one field, like {user.name}.
- "Cannot read properties of undefined": the data hasn't arrived yet. Show a loading state or read it safely with user?.name.
- "Hydration failed" in Next.js: the server and the browser drew different things, often the time, a random number or something from localStorage.

Key terms

React
The most popular JavaScript library for building interfaces from components.

Component
A reusable piece of interface, in React a function that returns what to show.

Props
The inputs passed into a React component.

State
Data a component or app remembers and updates, like what's in a cart.

JSX
HTML-like tags written inside JavaScript, which React turns into elements on the page.

Hook (React)
A React function whose name starts with use, like useState or useEffect, that adds a feature to a component.

useEffect
A React hook that runs code after the screen updates, to stay in sync with things outside React like timers.

Key (React lists)
A unique, stable id on each item in a list, so React can tell the items apart between renders.

DOM (Document Object Model)
The browser's live tree of a page's elements, which JavaScript can read and change.

Next.js
A React framework for pages, server code and fast loading, made by Vercel.

Try it: Build and break a tiny to-do list

1. Ask your AI builder for a React to-do list component with an input, an Add button and a list, using useState.
2. Find the useState line and say out loud what the state variable and its setter are called.
3. Find the .map and check each item has a stable key from an id, not the index or a random number.
4. Add a 'done' checkbox to each item, then ask the AI to explain how it updates state without changing the old array.
5. Break it on purpose: remove the key, open the browser Console and read the warning. Put it back.

KNOWLEDGE CHECK

React basics: components, props and state

5 questions. Circle one answer for each; 4 or more correct passes. Answers are at the back.

1. What is a React component?

A. A function that returns what to show
B. A database table that stores a page's buttons and text
C. A CSS file that styles one part of the page
D. A plugin you download from the React app store

2. Why use useState instead of a plain variable for a counter?

A. Plain variables can't hold numbers inside React
B. State remembers its value between renders, and setting it redraws the screen
C. useState makes the counter load faster
D. Plain variables only work in TypeScript files

3. React warns: 'Each child in a list should have a unique key prop'. Best fix?

A. Use key={Math.random()} so every key is always unique
B. Ignore it, because warnings never change how the list behaves
C. Give each item a stable key, like key={item.id}
D. Wrap the whole list in a second

4. A button has onClick={setOpen(true)} and React says 'Too many re-renders'. What's wrong?

A. setOpen only works inside useEffect, so move the call there and add open to its dependency list
B. The button needs type="submit" before clicks can register
C. React can't handle clicks, so use addEventListener
D. It calls setOpen while drawing; use onClick={() => setOpen(true)}

5. Showing {user} crashes with 'Objects are not valid as a React child'. The fix?

A. Show one field, like {user.name}
B. Add key={user} so React knows which object to show
C. Wrap it in quotes: {'user'}
D. Convert the page to plain HTML so React isn't involved at all

Official sites

React docs: State, a component's memory: https://react.dev/learn/state-a-components-memory
React docs: Rendering lists (keys): https://react.dev/learn/rendering-lists

Last reviewed 2026-11-25. AI tools change fast; check each tool's own site for current features, plans and prices.

ANSWER KEY

React basics: components, props and state

1. A: A function that returns what to show
Components are functions that return JSX. Apps are trees of them, from a whole page down to a single button.

2. B: State remembers its value between renders, and setting it redraws the screen
A plain variable resets every time the component runs. State persists, and calling the setter tells React to update the screen.

3. C: Give each item a stable key, like key={item.id}
Keys tell React which item is which between renders. Random keys change every render, which resets items and loses their state.

4. D: It calls setOpen while drawing; use onClick={() => setOpen(true)}
With the arrow function, React calls setOpen when the button is clicked. Without it, setOpen runs during every render, which triggers another render.

5. A: Show one field, like {user.name}
React can show text and numbers but not a whole object. Pick the field you mean, or map over it if it's a list.