ScopeTreasure Courses

REACT COURSES · LESSON 5.2 · CODE SCOPE SERIES

Components & JSX

Pages built from LEGO bricks: function components, the capital-letter rule, JSX and how it differs from HTML, {} expressions, fragments, and reusing one Button three times.

Introduction

A React page is a tree of **components** — small functions that return the piece of screen they own. The code they return looks like HTML but is called **JSX**, and it lives right inside your JavaScript. Once you can write a component and read JSX, every React file in the world starts to make sense. *(Roman Urdu: Component aik chhota function hai jo screen ka apna hissa banata hai, aur JSX woh HTML jaisi zuban hai jo woh wapas bhejta hai.)*

Key points

- **A component is a function** whose name starts with a **capital letter** — `Button`, `Navbar`, `App`. The capital is not style: React reads lowercase tags as plain HTML (`<div>`) and capitalised ones as your components (`<Button />`).
- **JSX is not HTML:** it only looks similar. Differences that bite beginners: `class` becomes **`className`**, every tag must be **closed** (`<img />`, `<br />`), and attributes are camelCase (`onClick`, `tabIndex`).
- **One parent rule:** a component must return **one** wrapper element. When you do not want an extra `<div>` in the page, return a **fragment** `<> ... </>` — a wrapper that leaves no trace in the real DOM.

  (Roman Urdu: Component hamesha aik hi wrapper wapas bhejta hai — fazool div na chahiye to fragment <> </> use karo.)
- **Curly braces are a door:** inside JSX, `{ }` drops you back into JavaScript — `{name}`, `{price * 2}`, `{items.length}`. Anything between the braces is evaluated and printed.
- **Components compose:** a component can use other components, and the same component can appear many times with no extra code — that is the LEGO idea from the video.
- **Files:** usually one component per file, exported with `export default`, imported where needed — the ES6 modules you already know.

Code examples

Example 1 — A Button, reused three times

```
function Button() {
  return <button className="btn">Click me</button>;
}

function App() {
  return (
    <div>
      <Button />
      <Button />
      <Button />
    </div>
  );
}
```

What it does: `Button` is written once and used three times in `App`. Each `<Button />` is an independent copy on the screen — change the component's code once, and all three update together.

What you will see: three identical buttons in a row, all coming from one small function.

Example 2 — JSX is not HTML: className, closed tags, fragments

```
function ProfilePhoto() {
  return (
    <>
      <img src="/ali.jpg" alt="Ali" className="photo" />
      <p className="caption">Ali — React student</p>
    </>
  );
}
```

What it does: Three JSX rules in one small piece: the fragment `<> </>` wraps two elements without adding a real div; the image tag closes itself with `/>`; and styling uses `className`, never `class`. In plain HTML these details are forgiven; in JSX they are syntax.

What you will see: a photo with a caption under it — and if you inspect the page, no extra wrapper div around them.

Example 3 — Curly braces: real JavaScript inside the markup

```
function Bill() {
  const price = 250;
  const quantity = 3;
  return (
    <div className="bill">
      <p>Price: Rs {price}</p>
      <p>Quantity: {quantity}</p>
      <p>Total: Rs {price * quantity}</p>
      <p>{quantity > 1 ? "Bulk order" : "Single item"}</p>
    </div>
  );
}
```

What it does: Every `{ }` is live JavaScript: two variables are printed, a multiplication is calculated inside the markup, and a ternary picks the label. Nothing here is typed twice — change `quantity` and the whole bill corrects itself.

What you will see: Price: Rs 250, Quantity: 3, Total: Rs 750, and the label "Bulk order".

Quick recap

- A **component** is a capitalised function that returns **JSX**; lowercase tags are HTML, capitalised tags are yours.
- JSX rules: `className` not `class`, close every tag, one parent (or a fragment `<> </>`), and `{ }` for live JavaScript inside the markup.
- Write a piece once, reuse it everywhere — that is the whole LEGO promise.

Common mistakes

1. **Lowercase component names** — `function button()` used as `<button />` renders a plain HTML button tag, and your component never runs. The fix: components always start with a capital letter — `Button`, and use `<Button />`.
2. **Writing `class` in JSX** — React warns and the styles silently do not apply the way you expect. The fix: always `className` in JSX (and `htmlFor` instead of `for` on labels).
3. **Returning two parents** — `return <h1>A</h1><p>B</p>` is an error; a component returns one wrapper. The fix: wrap in a `<div>` when a real wrapper is fine, or a fragment `<> </>` when it is not.

Practice

1. Write a `StudentCard` component that shows a name in an `<h3>` and a course in a `<p>`, using two variables and curly braces. Use it three times inside `App` (the values can repeat for now — real per-card

data arrives with props in the next lesson).

2. Take any plain HTML snippet you wrote in the HTML series (a small form or a card). Rewrite it as a JSX component: fix `class` → `className`, close every open tag, and wrap the whole thing in a fragment.

Practice quiz

1. Why must a component name start with a capital letter?

- It is only a style habit
- React reads lowercase tags as plain HTML and capitalised tags as your components
- Lowercase names are syntax errors
- Capital letters make it run faster

Answer: React reads lowercase tags as plain HTML and capitalised tags as your components

2. In JSX, what do you write instead of the HTML attribute class?

- class, exactly as in HTML
- className, and every tag must be closed
- classname in lowercase
- style-class

Answer: className, and every tag must be closed

3. A component must return one wrapper. What do you use when you do not want an extra div in the page?

- Return two parents anyway
- A fragment <> ... </>, a wrapper that leaves no trace in the real DOM
- Return null always
- A span, always

Answer: A fragment <> ... </>, a wrapper that leaves no trace in the real DOM

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/react-components-jsx · More free tools and notes at https://scopetreasure.com