# Day -11 Rendering 08.12.2025

## 🚀 Create Phase (JavaScript / React Basics)

### ✔ Components

There are **two types of components**:

1. **Class Components**
2. **Functional Components**

Both are used to create reusable UI blocks in React.

---

## ✔ Create Phase = Dynamic Process

- In JavaScript, variables and values are created dynamically.
- You do not need to specify the data type.
- The engine decides the type at runtime.

Example:

```
let a = 10;       // number
a = "Hello";      // string (type changed)
```

This is called **dynamic typing**.

---

## ✔ Keywords used to declare variables

| Keyword | Scope | Value Change |
|---|---|---|
| var | Function scope | Can change |
| let | Block scope | Can change |
| const | Block scope | Cannot change |

---

## ✔ Data Types in JavaScript

### ◆ Primitive (Basic data types)

| Type | Example |
|---|---|
| string | "Hello" |
| number | 10, 10.5 |

| Type | Example |
|---|---|
| boolean | true, false |
| undefined | Variable declared but no value |
| null | Empty (value is nothing) |

◆ **Non-Primitive (Complex data types)**

| Type | Example |
|---|---|
| Array | [10, "Hello", true] |
| Object | {name: "Jagath", age: 23} |

---

### ✔ Dynamic Meaning

- JavaScript is **dynamic**
- Variable type can change at runtime
- Used many times in program
- No need of type declaration (int, string, etc. like in Java/C++)

Example:

```
let x = 10;
x = "Dynamic";
```

# 📌 1. Screen Rendering (React)

Screen rendering means **displaying data on the UI (screen)**.
We can render different types of data like **String, Number, Boolean, Null, Undefined** inside JSX `{ }`.

---

### ▯ 1. String Rendering

### 📌 When to use?

When you have **text data** and you want to show it in multiple places in your component.

### ▯ Example:

```
const data = "This is a string";
```

### JSX Rendering:

```
<p>{data}</p>
```

---

## 🔢 2. Number Rendering

### 📌 When to use?

When you have a **number value** that should appear multiple times in UI.

### □ Example:

```
const number = 23;
```

### JSX Rendering:

```
<p>{number}</p>
```

✕The example you wrote `"23:09 PM 08/12/2025"` is **not a number**.
It is a **string**, not a number.

---

## ✔ 3. Boolean Rendering

Boolean values give **true** or **false**.

```
const isActive = true;
```

### 📌 Where used?

Boolean is used only in:

- Conditional rendering
- Optional rendering

### Example:

```
{isActive && <p>User is Active</p>}
```

---

## ⊘ 4. Null Rendering

Null means "empty value".

```
const data = null;
```

### 📌 Why use Null?

- To show **nothing**
- To **reset** a variable

### Nullish Values:

`null` and `undefined` are called **Nullish values**

---

## 5. Undefined Rendering

Undefined means **variable declared but no value assigned**.

```
let value;
```

Here `value` is undefined.

### 📌 Rendering Undefined:

By default, undefined will **not render on screen**.

## Conditional Rendering (React) — Complete Explanation

Conditional rendering means **deciding what to show on the screen based on a condition**.
UI changes depending on **true or false**.

---

## ✔ Why We Use Conditional Rendering?

Because UI should change based on:

- Login / Logout
- Loading / Loaded
- Error / Success
- Show / Hide elements

---

## Different Types of Conditional Rendering

---

### 1. IF-ELSE Rendering

### ▶ Syntax:

```
if(condition){
  return <UI for true />;
}
else {
  return <UI for false />;
}
```

### 💡 Example:

```
const isLoggedIn = true;

if(isLoggedIn){
   return <h1>Welcome User</h1>;
}
else{
   return <h1>Please Login</h1>;
}
```

✔ Best for **multiple conditions**

---

## ❓ 2. Ternary Operator Rendering

### ▶ Syntax:

```
condition ? true : false
```

### 💡 Example:

```
const isDarkMode = false;

return (
  <div>
    {isDarkMode ? "Dark Mode Enabled" : "Light Mode Enabled"}
  </div>
);
```

✔ Best for **simple** true/false UI

---

## ❓ 3. AND Operator (&&) — Optional Rendering

### ▶ Syntax:

```
condition && UI
```

### 💡 Example:

```
const isActive = true;

return (
  <div>
    {isActive && <p>Account is Active</p>}
  </div>
);
```

✔ Only shows UI when condition **true**

---

## 🔵 4. OR Operator (||) — Default Rendering

### ▶ Syntax:

```
condition || UI
```

### 💡 Example:

```
const username = "";
return <p>{username || "Guest User"}</p>;
```

✔ Used when no data → show default

---

## 🧠 Real Use Cases in Applications

| Situation | True shows | False shows |
|---|---|---|
| Logged In | Dashboard | Login Page |
| Network Success | Data | Error message |
| Loading | Spinner | Data after load |
| Admin | Admin Panel | User access denied |

## 📌 3. OPTIONAL RENDERING (&&)

Optional Rendering means **display UI only if the condition is true**.
It uses the **logical AND operator (&&)**.

---

### ✔ Key Point:

🔹 `"True only shows"`
🔹 Nothing will show when the value is **false**

---

### 🧠 Basic Syntax:

```
condition && <UI />
```

---

### 💡 Example:

```
const showMessage = true;

{showMessage && <p>Hello User</p>}
```

Output:

```
Hello User
```

---

### ⊘ When condition becomes false:

```
const showMessage = false;

{showMessage && <p>Hello User</p>}
```

Output:

```
(nothing shown)
```

---

## ❓ Why we use Optional Rendering?

### ✔ When UI should show only on true:

- Show / hide message
- Show button only when needed
- Show extra info

## 📌 4. Nullish Coalescing (??)

Nullish Coalescing is an operator used in JavaScript and React.

### 🎯 Purpose:

If the **left side value** is:

- `null` or
- `undefined`

👉 then the **right side value** will be used.

---

### 🔢 Operator:

```
??
```

---

### ❓ Syntax:

```
value ?? message;
```

Meaning:
✔ If `value` is `null` or `undefined`, show `message`.

---

### 🧑‍💻 Example:

```
const username = null;
console.log(username ?? "Guest User");
```

Output:

```
Guest User
```

---

### 💡 Useful for API Data

When API doesn't return data, instead of showing **undefined**, we show a **default message**.

Example:

```
const userName = apiName ?? "No User Found";
```

---

### 🤔 Nullish vs OR (Important difference)

#### OR (||) checks:

- false
- 0
- "" (empty string)
- null
- undefined

#### Nullish (??) checks:

- only `null`
- only `undefined`

So, **nullish is better** when data may be empty but valid.

---

#### Example to understand better:

```
const amount = 0;
console.log(amount || "No Amount");   // "No Amount"
console.log(amount ?? "No Amount");   // 0
```

✔ `??` is correct here.

---

### 🏁 Interview Summary

- Used to handle `null` and `undefined`
- Works like “default value”
- Mostly used in **API response**
- Syntax: `left ?? right`

# Day 11 - Task

```
const Day11Task = () => {

    const box: boolean = true;
    const number: number = 31;

    const value: any = true;

    const color: string = "yellow";

    const heroName = "Jagath";
    const companyName = "Tata"

    const getProject = true;


    return (
        <>
            {/* Task - 1 Hide and Show Box */}

            {box && <div style={{ backgroundColor: "black", color: "white", padding:
"10px", width: "100px", height: "100px", textAlign: "center", marginBottom: "30px"
}}>
                <p>This is Box</p>
            </div>}

            {/* 2. Hide and show text */}

            {box ? "Show" : "Hide"}

            {/* 3. odd or even 2 or 3  */}

            {number % 2 === 0 ? `${number} is Even Number` : `${number} is Odd
Number`}

            {/* 4. truthy and false value button */}
```

```
            {value ? <button style={{ padding: "10px", backgroundColor: "green",
color: "white", border: "none" }}>Truthy Value</button> :
                <button style={{ padding: "10px", backgroundColor: "red", color:
"white", border: "none" }}>Falsy Value</button>
            }

            {/* 5. green - go
                red - stop
                yellow - ready  */}

            {color === "red" && <button style={{ backgroundColor: "red", border:
"none", padding: "20px", fontWeight: "bold", color: "white" }}>Stop</button>}
            {color === "yellow" && <button style={{ backgroundColor: "yellow",
border: "none", padding: "20px", fontWeight: "bold" }}>Ready</button>}
            {color === "green" && <button style={{ backgroundColor: "green", border:
"none", padding: "20px", fontWeight: "bold", color: "white" }}>Go</button>}


            {/* 6. value undefined nullesh value return */}

            {value ?? "The value is undefined or null"}

            {/* 7. Own Story */}

            {getProject ?
                <p>
                    <span style={{ fontWeight: "bold", color: "green"
}}>{heroName}</span> joined <span style={{ backgroundColor: "red", fontWeight:
"bolder" }}>{companyName}</span> as a junior software developer.
                    After a few months, the <span style={{ color: "red", fontWeight:
"bolder" }}>{companyName}</span> announced a big project. Many employees wanted it,
but <span style={{ fontWeight: "bold", color: "green" }}>{heroName}</span> also
applied confidently.

                    IF <span style={{ fontWeight: "bold", color: "green"
}}>{heroName}</span> gets the project…

                    <span style={{ fontWeight: "bold", color: "green"
}}>{heroName}</span> works with full focus at <span style={{ backgroundColor: "red",
fontWeight: "bolder" }}>{companyName}</span>. He spends extra hours understanding
requirements, helping teammates, and improving the design. His coding is clean and
efficient. After successful testing, the product is launched.
                    The client appreciates the work and sends positive feedback. The
manager calls <span style={{ fontWeight: "bold", color: "green" }}>{heroName}</span>
to the conference room and congratulates him in front of everyone.
                    <span style={{ fontWeight: "bold", color: "green"
}}>{heroName}</span> earns a promotion and becomes a mentor for new joiners at
```

```
Phoenix Tech <span style={{ backgroundColor: "red", fontWeight: "bolder"
}}>{companyName}</span></p> :

                <p>
                    <span style={{ fontWeight: "bold", color: "green"
}}>{heroName}</span> doesn’t get the project…

                    He feels disappointed but doesn’t stop learning. <span style={{
fontWeight: "bold", color: "green" }}>{heroName}</span> requests feedback from
seniors, practices new technologies, and supports the team in smaller tasks.
                    One day, another urgent project arrives at . Because <span
style={{ fontWeight: "bold", color: "green" }}>{heroName}</span> has been learning
consistently, he solves problems faster than others.
                    The manager notices his growth and gives him a chance to lead a
mini project. <span style={{ fontWeight: "bold", color: "green" }}>{heroName}</span>
succeeds and wins the respect of everyone.
                    Even though he didn’t get the first project,
<span>{heroName}</span> becomes an important contributor at <span style={{
backgroundColor: "red", fontWeight: "bolder" }}>{companyName}</span>.
                </p>

            }
        </>
    )

};

export default Day11Task;
```