Class 5: Conditional Rendering

- Review: Last Class
- Conditional Rendering
  - Conditional Statements
  - Ternary Operator
  - Logical AND Operator
- Practice Problem Workshops

Review: JSX Expression Placeholders

Use **curly braces** `{}` to *embed* JavaScript *expressions* inside the HTML-like syntax of JSX.

```
export default function ComponentName() {
  const name = "Prof. Harms"

  return <div>Hello, {name}!</div>
}
```

Gotcha: Expression placeholders **only** exists inside the HTML-like portions of a JSX file.

Review: Props

Props (short for *properties*) are a way to pass data into a React component.

```
export default function Citation({source}) {
  return (
    <p className="citation">
      Source: <cite>{source}</cite>
    </p>
  )
}
```

Review: Expression Placeholders & Object Literals

When using an object literal inside a JSX expression placeholder, you need to use **double** curly braces `{{ ... }}`.

```
<div className="warning" style={{ color: 'red', fontSize: '20px' }}>
  Hello
</div>
```

The **outside** pair of curly braces is for the JSX expression placeholder.
The **inside** pair of curly braces is for the JavaScript object literal.

Activity: Dynamic Styling with Props

1. Create a `bgColor` prop on the `Card` component.
2. Set the default value of the `bgColor` prop to `#fff`.
3. Use the bgColor prop to set the background color of the card using inline styles.
4. Test the Card component by passing different bgColor values when rendering it.

   Sample color: `#e3cef2` or `lightgrey`

Conditional Rendering

Conditional Rendering

Render a component **only when** a condition is **true**.

Examples:

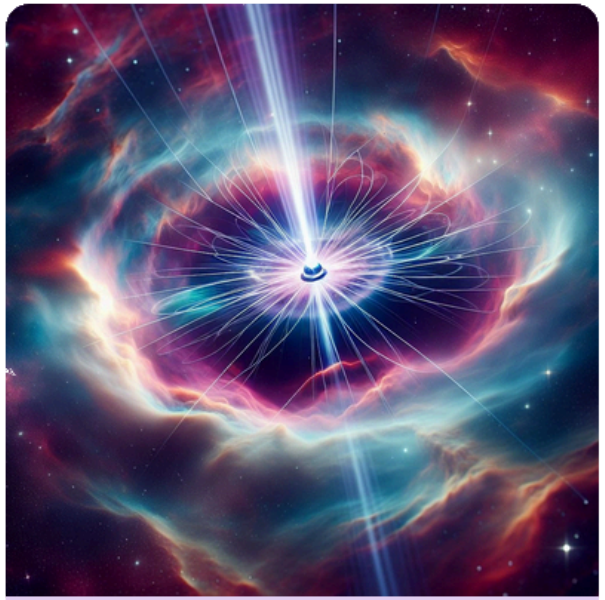
- Render a “hollow” heart icon when an item is not favorited, and a “filled” heart icon when it is favorited.
- Render a “Login” button only when the user is not logged in.
- Render a light mode button when in dark mode.
- Render a warning message when a form field is invalid.

Common Conditional Rendering Cases

- Change user interface in response to user actions.
- Change the appearance of a component based on a prop value.

Discussion: What Needs Conditional Rendering?

Source: *Microsoft Copilot*



A pulsar is a rapidly rotating neutron star that emits beams of electromagnetic radiation.

Source:

Source:

Conditional Statements in JavaScript

Conditional Statements in JavaScript

`if`, `else if`, and `else` statements allow you to execute different blocks of code based on certain conditions.

```
if (conditionalExpression) {
  // code to run when conditionalExpression is true
} else if (anotherConditionalExpression) {
  // code to run when anotherConditionalExpression is true
} else {
  // code to run when neither condition is true
}
```

Conditional Expressions

A **conditional expression** is an expression that evaluates to either `true` or `false`.

Examples:

```
color == 'red'
text != '' || text != null
age >= 18
shoppingCart.length > 0
isDarkMode === true
```

True & False in JavaScript

A conditional expression evaluates to **truthy** or **falsy**.

Falsy

- `null`
- `undefined`
- `false`
- `NaN`
- `0`
- `-0`
- `0n`
- `""` or `''`
- `document.all`

Truthy

Anything that is **not** falsy.

Examples:

- `true`
- `[]`
- `{}`
- `42`
- `"hello"`
- `function() {}`
- `new Date()`

Examples: Conditional Statements

```
const color
if (color === 'red') {
  console.log('The color is red.')
}
```

```
const age = 20
if (age >= 18) {
  console.log('You are an adult.')
} else {
  console.log('You are a minor.')
}
```

Comparison Operators

Operator	Description	Example	Evaluates to
`==`	Equal to	`5 == '5'`	truthy
`===`	Strict equal to	`5 === '5'`	falsy
`!=`	Not equal to	`5 != '5'`	falsy
`!==`	Strict not equal to	`5 !== '5'`	truthy
`>`	Greater than	`5 > 3`	truthy
`<`	Less than	`5 < 3`	falsy
`>=`	Greater than or equal to	`5 >= 5`	truthy
`<=`	Less than or equal to	`5 <= 3`	falsy

Gotcha: Assignment vs. Comparison

The assignment operator `=` is used to assign a value to a variable.

```
let x = 10       // Assignment operator (=)
if (x == 10) {   // Comparison operator (==)
  console.log('x is 10')
}
```

```
if (x = 5) {     // Incorrect: assignment instead of comparison
  console.log('This will always run because x is assigned 5, which is truthy.')
}
```

Logical Operators

Logical operators allow you to combine multiple conditional expressions.

Operator	Description	Example	Evaluates to
`&&`	Logical AND	`true && false`	falsy
`||`	Logical OR	`true || false`	truthy
`!`	Logical NOT	`!true`	falsy

Conditional Rendering

Conditional Statements

Rendering with a Conditional Statement

```
export default function DarkModeSwitch({ isDarkMode }) {
  if (isDarkMode) {
    return <button>
             <img src="light-mode.svg" />
           </button>
  } else {
    return <button>
             <img src="dark-mode.svg" />
           </button>
  }
}
```

Alternative with Conditional Statement

Store the element in a variable and return it.

```
export default function DarkModeSwitch({ isDarkMode }) {
  let img
  if (isDarkMode) {
    img = <img src="light-mode.svg" />
  } else {
    img = <img src="dark-mode.svg" />
  }
  return <button>{img}</button>
}
```

Demo: Rendering with a Conditional Statement

```
export default function Citation({ citation }) {
  if (citation) {
    return (
      <p className="citation">
        Source: <cite>{citation}</cite>
      </p>
    )
  } else {
    return null
  }
}
```

Activity: Conditional Statement Practice

Working with your peers (2-4), complete Part I of the handout.

```
export default function DarkModeSwitch({ isDarkMode }) {
  if (isDarkMode) {
    return <button>
             <img src="light-mode.svg" />
           </button>
  } else {
    return <button>
             <img src="dark-mode.svg" />
           </button>
  }
}
```

Conditional Rendering

Ternary Operator

JavaScript Ternary Operator

The **ternary operator** is a shorthand way to write an `if-else` statement as an **expression**.

```
condition ? expressionIfTrue : expressionIfFalse
```

Example:

```
const ariaLabel = isDarkMode ? 'Dark Mode' : 'Light Mode'
console.log(ariaLabel)
```

Rendering with the Ternary Operator

```
export default function DarkModeSwitch({ isDarkMode }) {
  return isDarkMode ? (
    <button>
      <img src="light-mode.svg" />
    </button>
  ) : (
    <button>
      <img src="dark-mode.svg" />
    </button>
  )
}
```

Alternative with Ternary Operator

Store the element in a variable and return it.

```
export default function DarkModeSwitch({ isDarkMode }) {
  const img = isDarkMode ? (
    <img src="light-mode.svg" />
  ) : (
    <img src="dark-mode.svg" />
  )
  return <button>{img}</button>
}
```

Alternative II with Ternary Operator

```
export default function DarkModeSwitch({ isDarkMode }) {
  return <button>{isDarkMode ? (
    <img src="light-mode.svg" />
  ) : (
    <img src="dark-mode.svg" />
  )}</button>
}
```

Alternative III with Ternary Operator

```
export default function DarkModeSwitch({ isDarkMode }) {
  return (
    <button>
      <img src={isDarkMode ? "light-mode.svg" : "dark-mode.svg"} />
    </button>
  )
}
```

Demo: Rendering with the Ternary Operator

```
export default function Citation({ citation }) {
  return citation ? (
    <p className="citation">
      Source: <cite>{citation}</cite>
    </p>
  ) : null
}
```

Activity: Ternary Operator Practice

Working with your peers (2-4), complete Part II of the handout.

```
export default function DarkModeSwitch({ isDarkMode }) {
  return <button>{isDarkMode ? (
    <img src="light-mode.svg" />
  ) : (
    <img src="dark-mode.svg" />
  )}</button>
}
```

Conditional Rendering

Logical AND Operator

JavaScript Logical AND Operator

The **logical AND operator** `&&` allows you to evaluate two expressions and returns the second expression if the first is truthy.

```
condition && expressionIfTrue
```

Example:

```
isDarkMode && console.log('Dark mode is enabled.')
```

Rendering with the Logical AND Operator

This is often the **preferred** way to do conditional rendering in React.

```
export default function DarkModeSwitch({ isDarkMode}) {
  return (
    <button>
      {isDarkMode && <img src="light-mode.svg" />}
      {!isDarkMode && <img src="dark-mode.svg" />}
    </button>
  )
}
```

Better Practice: `!!` (Boolean Coercion)

`!` returns the opposite boolean value of its operand.
`!!` returns the boolean value of its operand.

```
export default function DarkModeSwitch({ isDarkMode}) {
  return (
    <button>
      {!!isDarkMode && <img src="light-mode.svg" />}
      {!isDarkMode && <img src="dark-mode.svg" />}
    </button>
  )
}
```

Demo: Rendering With the Logical AND Operator

`Card.jsx`:

```
{!!citation && <Citation source={citation} />}
```

Activity: Logical AND Operator Practice

Working with your peers (2-4), complete Part III of the handout.

```
export default function DarkModeSwitch({ isDarkMode}) {
  return (
    <button>
      {!!isDarkMode && <img src="light-mode.svg" />}
      {!isDarkMode && <img src="dark-mode.svg" />}
    </button>
  )
}
```

Activity: Conditionally Rendered Caption

Conditional render the `<Caption>` component only if the `caption` prop is provided.

```
export default function DarkModeSwitch({ isDarkMode}) {
  return (
    <button>
      {!!isDarkMode && <img src="light-mode.svg" />}
      {!isDarkMode && <img src="dark-mode.svg" />}
    </button>
  )
}
```

Practice Problem Workshops

Motivation

Imagine the following scenario:

> Aeneas's homework is due this evening at 11:59PM. They have homework for two other classes also due tonight and a prelim tomorrow. They ask Chat to complete this homework for them. They read over the response and understand it.

Discussion

Did Aeneas learn what they needed to learn from that homework? How will Aeneas perform on the exam on this topic?

Reading something and understanding it is **not** the same as being able to **apply** that knowledge.

This is known as the **“illusion of mastery”**. It feels easy, so we think we’ve learned something. The opposite is often true. If something feels challenging, we are more likely to have learned it.

Retrieval Practice

Retrieval practice is the act of actively recalling information from memory, like quizzing oneself.

Research Findings: This strengthens memory and improves long-term retention of the material.

It isn’t enough to read a response from Chat and understand it. You need to be able to **recall** and **apply** that information on your own.

Practice Problem Workshops

Every Friday section will be dedicated to retrieval practice.

- Part I: I will provide a *few* practice problems related to the material covered that week.
  - You will work on these problems independently first.
  - Then you will work with your peers to finalize your solutions.
- Part II: Next, you will practice retrieval by quizzing each other.
  - As a group, you will create a few practice problems.
  - When finished, you will swap problems with another group and try to complete their questions.
- Submit your workshop packet for completion credit.

Exam Benefit

A *few* of the best problems from your practice problem workshops will be selected for the exams.

Every group's practice problems will be posted publicly for studying.

Summary

- Conditional rendering allows you to render components based on certain conditions.
- You can use conditional statements, the ternary operator, or the logical AND operator.
- The logical AND operator is often the preferred method for conditional rendering in React.
- Retrieval practice is essential for solidifying your understanding and preparing for exams.

What's Next

Released Today: Project 1, Milestone 2

Friday Section: Practice Problem Workshop

Due Sunday: Class 6 Preparation
Due Monday: Project 1, Milestone 2