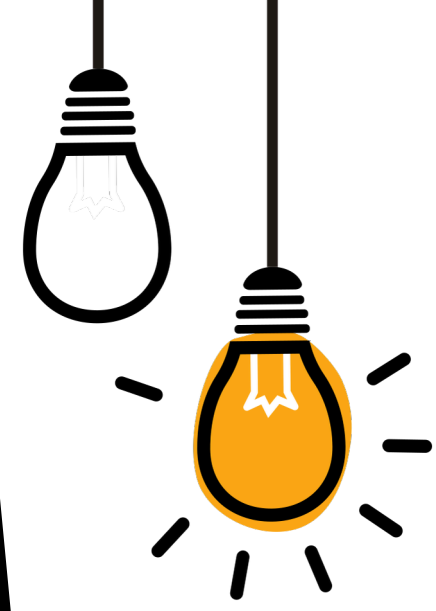
TRAILHEAD IQ



# HOW TO CREATE A PROGRESS INDICATOR IN SALESFORCE LWC


@TrailheadIQ

Swipe next ⟶

# What is a Progress Indicator?

A Progress Indicator (also known as a Progress Bar) provides feedback to users about the current stage in a process. It's an essential UI element in forms, wizards, and workflows.

**Common Use Cases:**

- Step-based forms
- Multi-stage approval workflows
- Data upload tracking

Swipe next →

**@TrailheadIQ**

# Why Use SLDS for Progress Indicators in Salesforce?

**Benefits of SLDS (Salesforce Lightning Design System):**

- Provides a consistent look and feel across all Salesforce components.
- Ensures that your UI is mobile-friendly and responsive.
- Built-in accessibility features, so your apps are inclusive.

Swipe next ⟶

**@TrailheadIQ**

# Components of a Progress Indicator

Before jumping into code, let's break down the key elements:

- **Progress Bar:** A horizontal bar that visually indicates the completion percentage.
- **Steps:** Optional labels to show stages in a process (e.g., Step 1, Step 2, etc.)
- **Dynamic Progress Updates:** Real-time updates as users complete each step.

Swipe next ⟶

**@TrailheadIQ**

# Example Code - LWC for Progress Indicator

## 📂 Step 1: Create the LWC Component

Html

```
<template>
    <lightning-progress-indicator current-step={currentStep} type="path" variant="base">
        <template for:each={steps} for:item="step">
            <lightning-progress-step
                label={step.label}
                value={step.value}
                key={step.label}
                onclick={handleStepClick}
            ></lightning-progress-step>
        </template>
    </lightning-progress-indicator>
</template>
```

Swipe next ⟶

@TrailheadIQ

TRAILHEAD IQ

# JavaScript Controller

JS

```
import { LightningElement,track } from 'lwc';

export default class TestLwc extends LightningElement {
    @track currentStep = 'step-1';

    steps = [
        { label: 'Contacted', value: 'step-1' },
        { label: 'Open', value: 'step-2' },
        { label: 'Unqualified', value: 'step-3' },
        { label: 'Nurturing', value: 'step-4' },
        { label: 'Closed', value: 'step-5' },
    ];

    handleStepClick(event) {
        const clickedStep = event.target.value;
        this.updateStep(clickedStep);
    }

    updateStep(newStep) {
        const currentIndex = this.steps.findIndex(step => step.value === this.currentStep);
        const newIndex = this.steps.findIndex(step => step.value === newStep);

        if (newIndex > currentIndex) {
            // Moving forward: update all steps in between
            for (let i = currentIndex + 1; i <= newIndex; i++) {
                this.markStepComplete(this.steps[i].value);
            }
        } else if (newIndex < currentIndex) {
            // Moving backward: update all steps in between
            for (let i = currentIndex; i > newIndex; i--) {
                this.markStepIncomplete(this.steps[i].value);
            }
        }

        this.currentStep = newStep;
        this.dispatchStepChangeEvent();
    }

    markStepComplete(stepValue) {
        const step = this.template.querySelector(`lightning-progress-step[value="${stepValue}"]`);
        if (step) {
            step.classList.add('slds-is-completed');
        }
    }

    markStepIncomplete(stepValue) {
        const step = this.template.querySelector(`lightning-progress-step[value="${stepValue}"]`);
        if (step) {
            step.classList.remove('slds-is-completed');
        }
    }

    dispatchStepChangeEvent() {
        const stepChangeEvent = new CustomEvent('stepchange', {
            detail: { step: this.currentStep }
        });
        this.dispatchEvent(stepChangeEvent);
    }
}
```

@TrailheadIQ

Swipe next ⟶

TRAILHEAD IQ

# Dynamic Progress Updates

- To make the progress dynamic, you can use JavaScript to update the completion percentage or stage based on user interactions, such as clicking a button to move to the next step.

- Each step click updates both the CSS class and the progressValue.

- The visual progress bar is adjusted in real-time using SLDS classes.

Swipe next ⟶

**@TrailheadIQ**

# Testing and Debugging Tips

When creating your Progress Indicator, keep these in mind:

- **Responsiveness:** Test on different screen sizes. SLDS ensures good mobile support, but always check.

- **Accessibility:** Use aria-* attributes to ensure screen reader compatibility.

- **Error Handling:** If the process has failure states, update the UI to reflect partial completion or errors.

Swipe next ⟶

**@TrailheadIQ**

# Conclusion

- Creating a Progress Indicator in Salesforce LWC using SLDS is a straightforward way to improve user experience in step-based workflows.

- SLDS provides the styling, while LWC handles the logic.

- With a few lines of HTML, JavaScript, and SLDS utility classes, you can add a powerful progress indicator to your Salesforce apps!

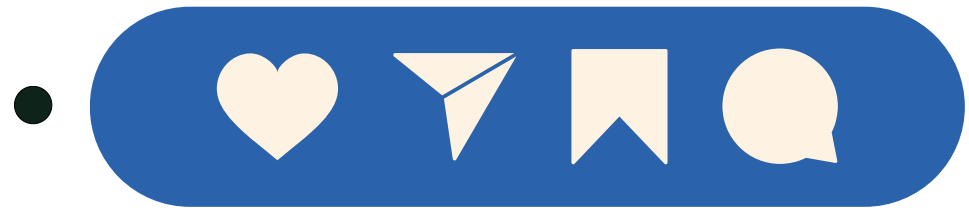
Happy coding! 😊

Swipe next ⟶

**@TrailheadIQ**

# THANK YOU

Remember, every like, share, and comment helps us reach more people and make a bigger impact. Together, we can make the Salesforce community stronger and more informed.#TrailheadIQ#SalesforceCommunity



www.trailheadiq.com

TRAILHEAD IQ