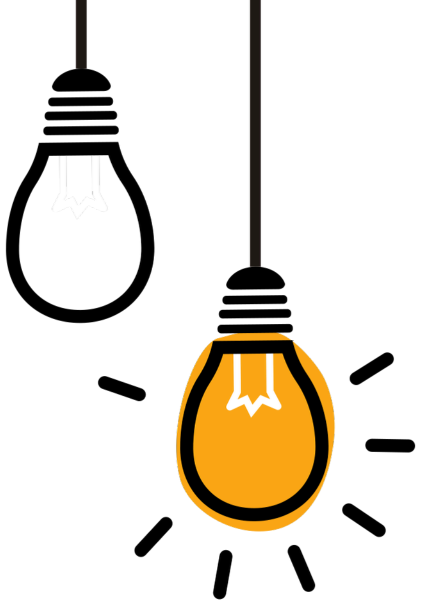
TRAILHEAD IQ



GENERATING PDF IN LIGHTNING WEB COMPONENT SALESFORCE

Swipe next →

@TrailheadIQ

Generating PDF in LWC

- In Salesforce, the ability to generate PDFs is a powerful feature used for creating printable documents such as invoices, reports, and contracts.

- With the introduction of Lightning Web Components (LWC), it's essential to understand how to generate PDFs efficiently.

Swipe next ⟶

@TrailheadIQ

TRAILHEAD IQ

Understanding the Core Concept

Generating PDFs in LWC requires a combination of backend logic (Apex) and frontend integration (LWC). The process typically involves:

- LWC to display the data.
- Apex for data handling and converting the HTML content into a PDF format.
- Leveraging the jsPDF or PDF.js library for formatting and rendering PDFs.

Swipe next ⟶

Setup PDF Libraries

For LWC, we commonly use jsPDF to generate PDFs. First, we must install and configure the library.

- Download jsPDF from the official GitHub repo.
- Upload the jsPDF library as a static resource.

Steps:

- Go to Setup > Static Resources > Upload jsPDF file.
- Ensure the name is accessible by the component.

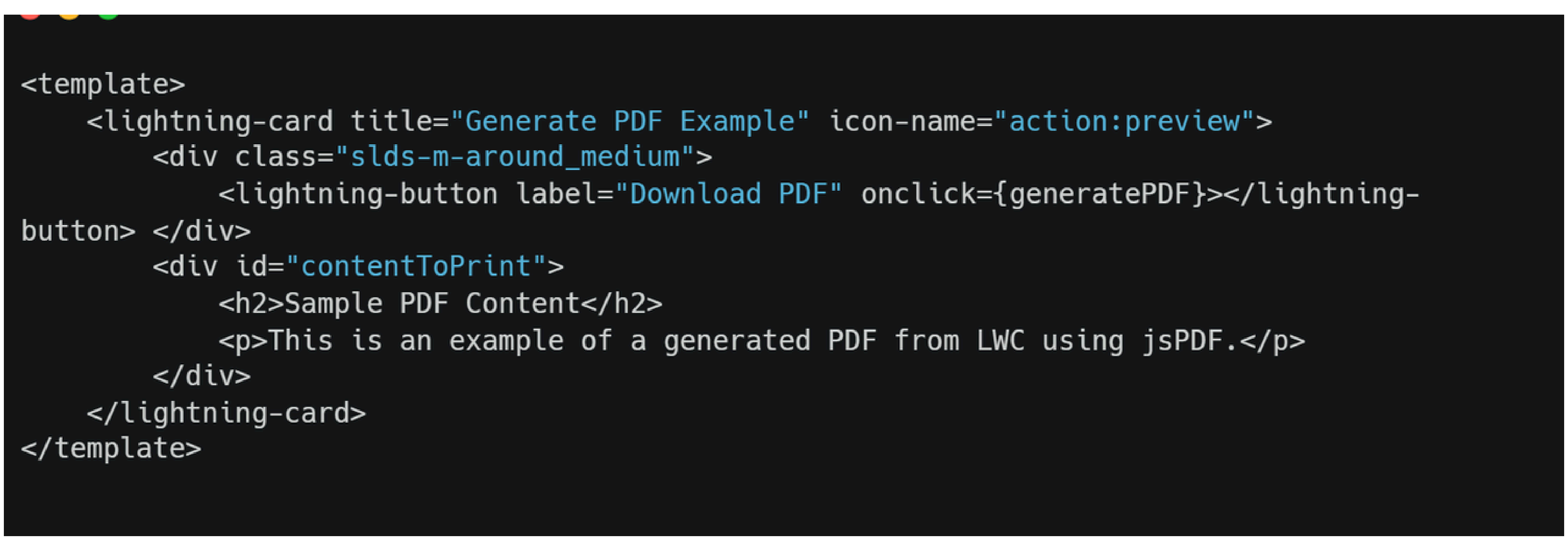
Swipe next ⟶

@TrailheadIQ

TRAILHEAD IQ

Create Lightning Web Component (LWC)

HTML File



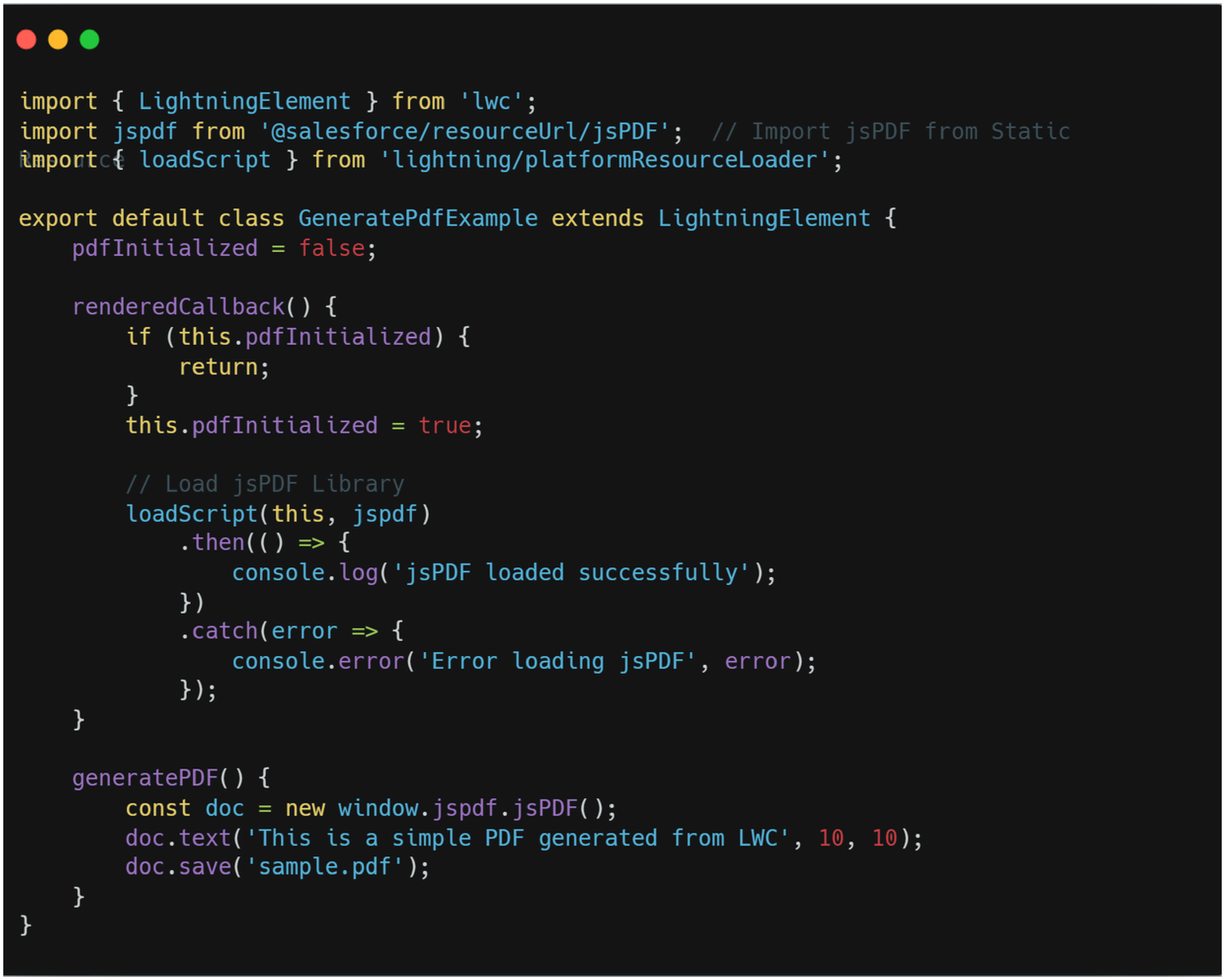
```
<template>
    <lightning-card title="Generate PDF Example" icon-name="action:preview">
        <div class="slds-m-around_medium">
            <lightning-button label="Download PDF" onclick={generatePDF}></lightning-
button> </div>
        <div id="contentToPrint">
            <h2>Sample PDF Content</h2>
            <p>This is an example of a generated PDF from LWC using jsPDF.</p>
        </div>
    </lightning-card>
</template>
```

Swipe next →

@TrailheadIQ

TRAILHEAD IQ

Js File



```
import { LightningElement } from 'lwc';
import jspdf from '@salesforce/resourceUrl/jsPDF';  // Import jsPDF from Static
import { loadScript } from 'lightning/platformResourceLoader';

export default class GeneratePdfExample extends LightningElement {
    pdfInitialized = false;

    renderedCallback() {
        if (this.pdfInitialized) {
            return;
        }
        this.pdfInitialized = true;

        // Load jsPDF Library
        loadScript(this, jspdf)
            .then(() => {
                console.log('jsPDF loaded successfully');
            })
            .catch(error => {
                console.error('Error loading jsPDF', error);
            });
    }

    generatePDF() {
        const doc = new window.jspdf.jsPDF();
        doc.text('This is a simple PDF generated from LWC', 10, 10);
        doc.save('sample.pdf');
    }
}
```

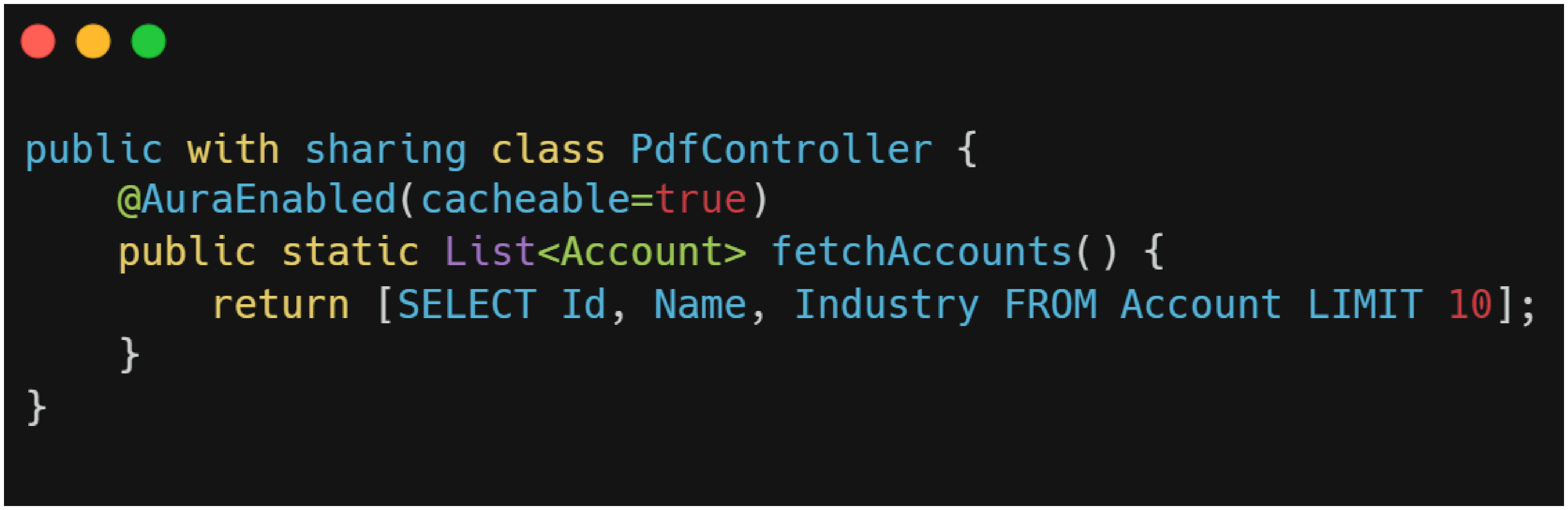
@TrailheadIQ

Swipe next ⟶

TRAILHEAD IQ

Using Apex for Complex Data

For scenarios where we need to fetch or manipulate data server-side before generating the PDF, we can leverage Apex.

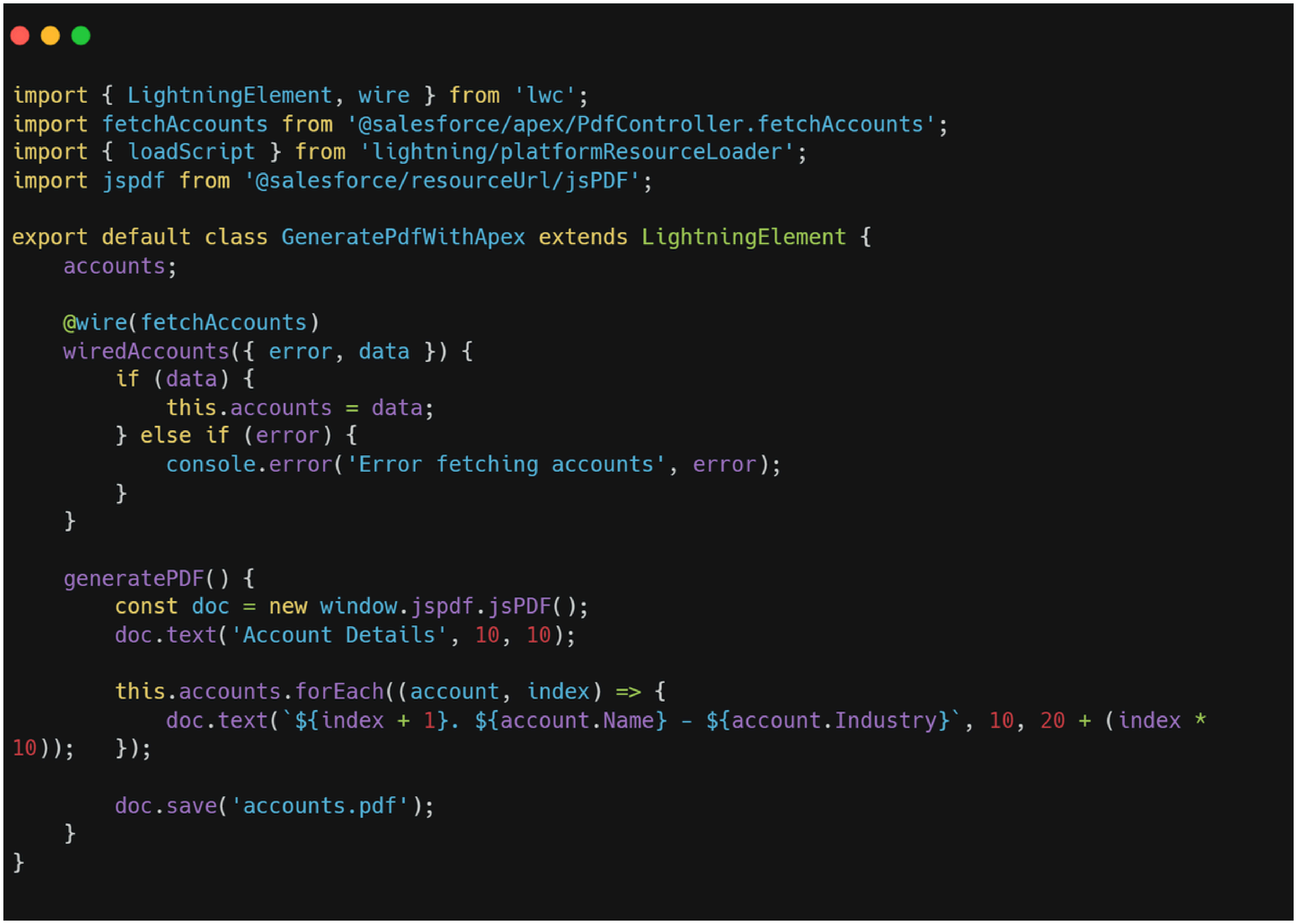


```
public with sharing class PdfController {
    @AuraEnabled(cacheable=true)
    public static List<Account> fetchAccounts() {
        return [SELECT Id, Name, Industry FROM Account LIMIT 10];
    }
}
```

Swipe next ⟶

@TrailheadIQ

LWC JavaScript with Apex



```
import { LightningElement, wire } from 'lwc';
import fetchAccounts from '@salesforce/apex/PdfController.fetchAccounts';
import { loadScript } from 'lightning/platformResourceLoader';
import jspdf from '@salesforce/resourceUrl/jsPDF';

export default class GeneratePdfWithApex extends LightningElement {
    accounts;

    @wire(fetchAccounts)
    wiredAccounts({ error, data }) {
        if (data) {
            this.accounts = data;
        } else if (error) {
            console.error('Error fetching accounts', error);
        }
    }

    generatePDF() {
        const doc = new window.jspdf.jsPDF();
        doc.text('Account Details', 10, 10);

        this.accounts.forEach((account, index) => {
            doc.text(`${index + 1}. ${account.Name} - ${account.Industry}`, 10, 20 + (index *
10));   });

        doc.save('accounts.pdf');
    }
}
```

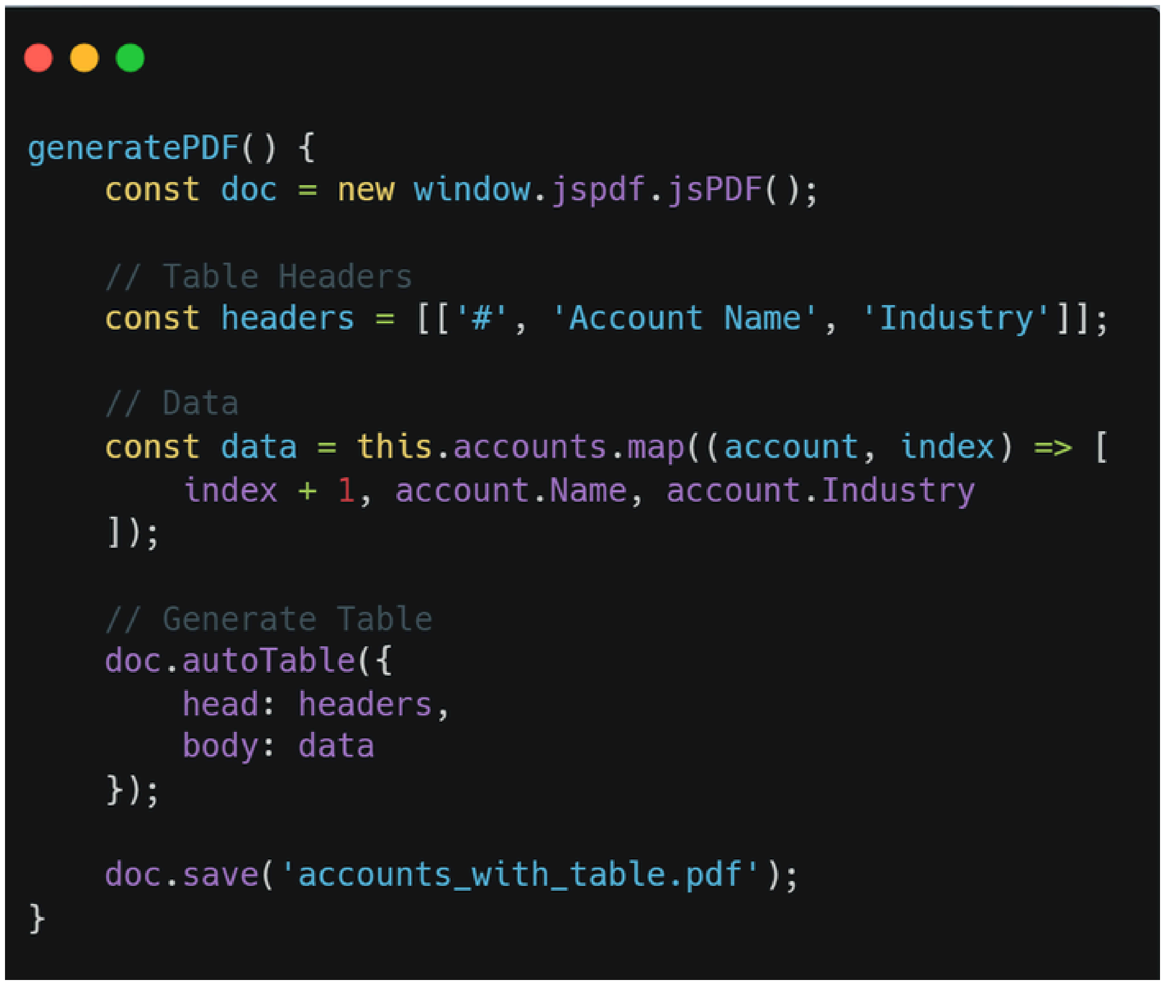
@TrailheadIQ

Swipe next

Styling and Formatting the PDF

When generating PDFs, you can format the content (like adding tables, headers, and images). Here's how to add some basic styling:

Adding a Table Example:



```
generatePDF() {
    const doc = new window.jspdf.jsPDF();

    // Table Headers
    const headers = [['#', 'Account Name', 'Industry']];

    // Data
    const data = this.accounts.map((account, index) => [
        index + 1, account.Name, account.Industry
    ]);

    // Generate Table
    doc.autoTable({
        head: headers,
        body: data
    });

    doc.save('accounts_with_table.pdf');
}
```

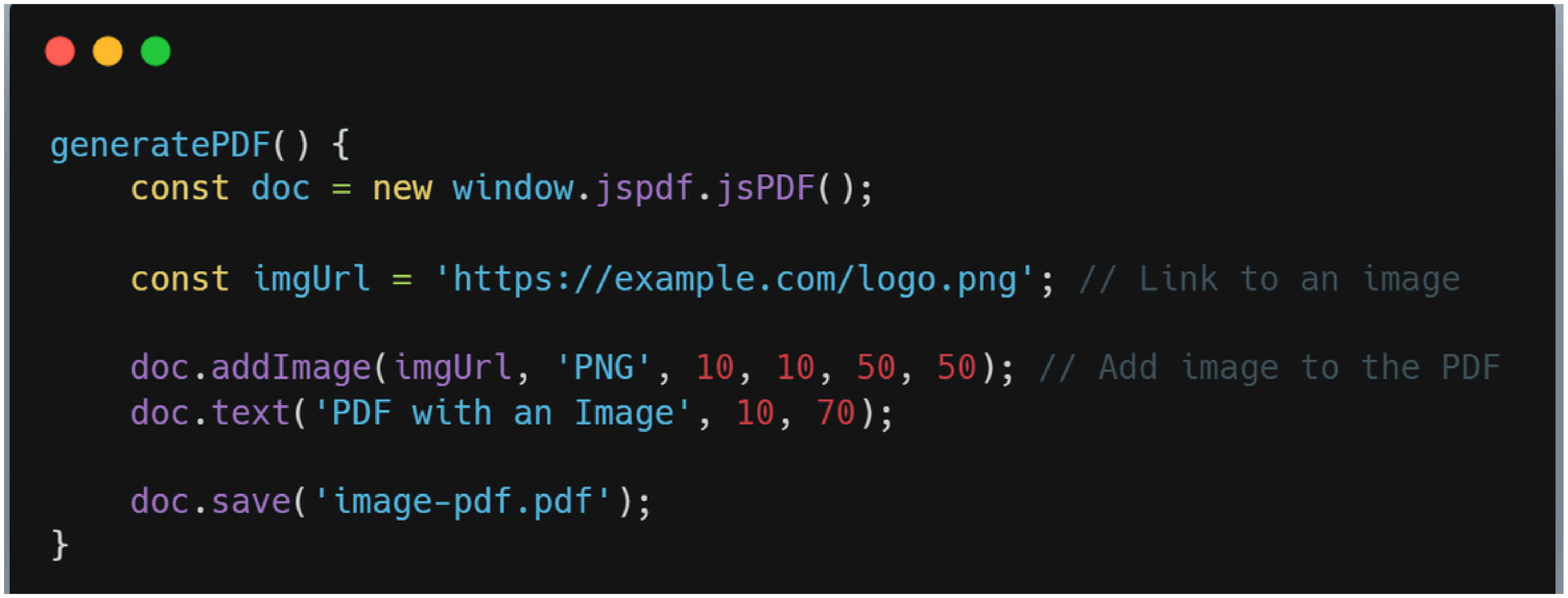
@TrailheadIQ

Swipe next

TRAILHEAD IQ

Generating PDF with Images and Custom Fonts

You can also include images and custom fonts in your PDFs for a professional appearance.
Adding an Image:



```
generatePDF() {
    const doc = new window.jspdf.jsPDF();

    const imgUrl = 'https://example.com/logo.png'; // Link to an image

    doc.addImage(imgUrl, 'PNG', 10, 10, 50, 50); // Add image to the PDF
    doc.text('PDF with an Image', 10, 70);

    doc.save('image-pdf.pdf');
}
```

Custom Fonts: You'll need to load the custom fonts as a static resource and embed them similarly to jsPDF.

@TrailheadIQ

Swipe next ⟶

TRAILHEAD IQ

Key Considerations and Limitations

- **Security Restrictions:** Salesforce imposes certain limits, such as the size of PDFs and file handling in client-side libraries.

- **Governor Limits:** Be cautious when using Apex to retrieve large datasets for PDF generation to avoid hitting governor limits.

- **Browser Compatibility:** Some PDF libraries may have browser compatibility issues—ensure proper testing.

- **Performance:** Depending on the complexity of the PDF, large documents may impact performance.

@TrailheadIQ

Swipe next

🚀 Conclusion

- Using LWC to generate PDFs in Salesforce provides a flexible and scalable approach to creating dynamic, professional documents.
- By combining Apex for server-side logic and jsPDF (or similar libraries) for PDF generation, you can handle a variety of use cases like reports, invoices, or customized forms.
- Feel free to explore different use cases and enrich your PDF with tables, images, and custom fonts!

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