Netgain Accounting SUMMIT 25

Vibe coding for accountants

AI-powered SuiteScript in 50 minutes

Agenda

Vibe coding for accountants

Introduction

AI-development fundamentals

Live demos (rapid-fire)

Deep-dive & best practices

Q&A & next steps

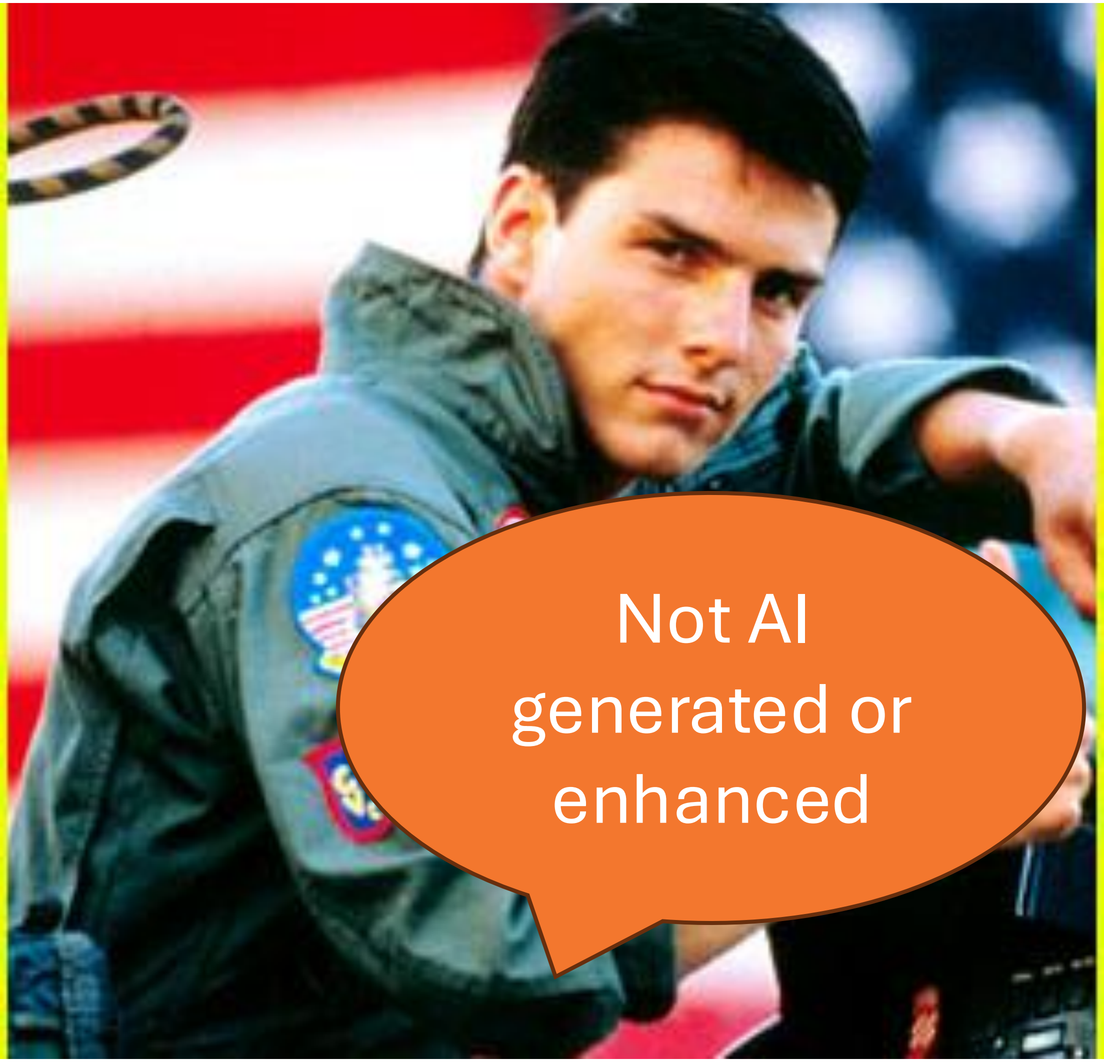
Chris Miller

SVP Product Strategy

Netgain

cmiller@netgain.tech

linkedin.com/in/christopherlmiller



Not AI generated or enhanced

Why vibe coding?

Traditional SuiteScript = specialized developer skill

AI makes scripting accessible to admins & analysts

Rapid prototyping → faster ROI on customizations

Democratizes automation & drives user adoption

What's on your mind today?

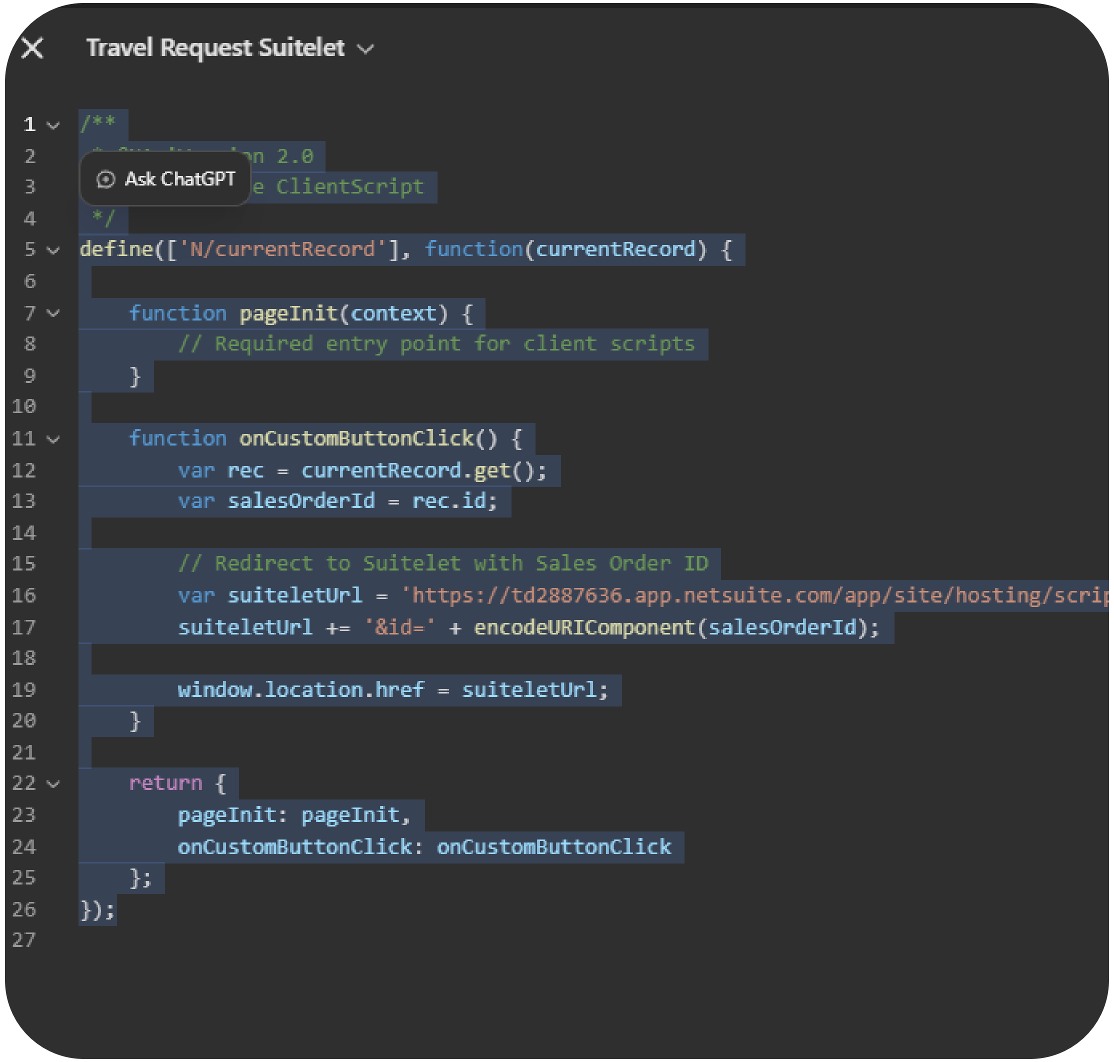
Canvas I am building a chat application using apex 24.2 and oracle autonomous database 23ai.

I want to chunk the data from my letters table into the table letter_chunks table. After it is chunked I want to call the open AI embedding model. However I want to build each piece sequentially so let's get chunking working awesome and then embed. The last piece will be writing this into a trigger so what when new data comes in it will be automatically processed.

Purpose: I'm building an application that holds confidential letters to the CEO. These letters focus on accomplishments, insights, recommendations, and suggestions. It comes in the form of an email each week from each employee. The CEO wants to use RAG and AI to get much better insights into the health and workings of the company using the data. It will be built in oracle apex using autonomous database 23ai.

Learning objectives

- Describe the AI-driven “vibe-coding” lifecycle and how it fits into NetSuite development.
- Craft prompts that generate production-ready SuiteScript 2.x code for common use cases.
- Deploy and test Client, Scheduled, and User Event scripts in NetSuite.
- Apply core vibe-coding best practices—providing strong documentation, clear context, well-scoped instructions, and iterative prompt refinement.
- Identify next steps and resources to accelerate your own AI-customization projects.

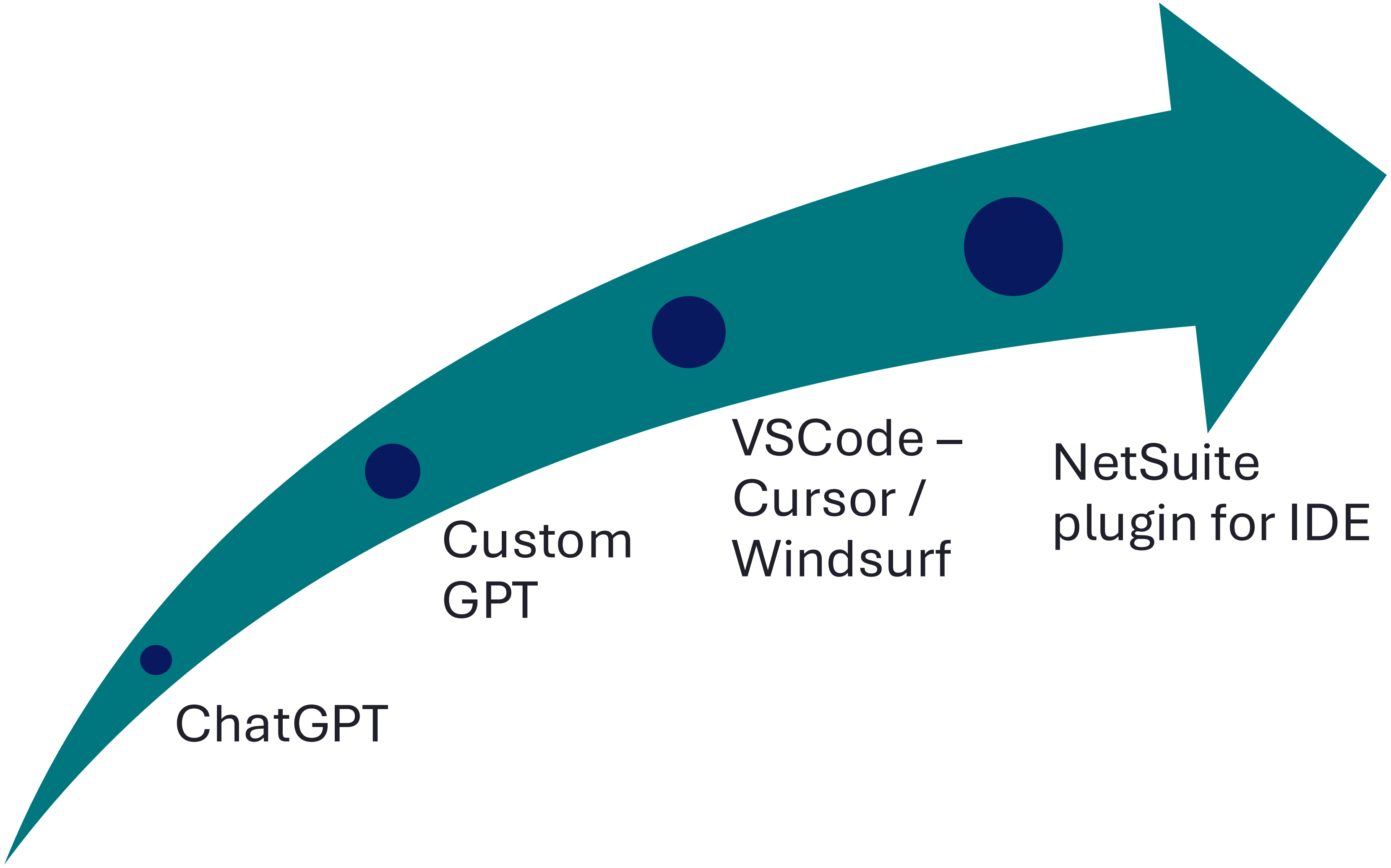


```
/**
 * @NApiVersion 2.0
             e ClientScript
 */
define(['N/currentRecord'], function(currentRecord) {

    function pageInit(context) {
        // Required entry point for client scripts
    }

    function onCustomButtonClick() {
        var rec = currentRecord.get();
        var salesOrderId = rec.id;

        // Redirect to Suitelet with Sales Order ID
        var suiteletUrl = 'https://td2887636.app.netsuite.com/app/site/hosting/scrip
        suiteletUrl += '&id=' + encodeURIComponent(salesOrderId);

        window.location.href = suiteletUrl;
    }

    return {
        pageInit: pageInit,
        onCustomButtonClick: onCustomButtonClick
    };
});
```

DEV fundamentals overview

Environments, mindset, AI-side process

Coding environment options



ChatGPT

Custom GPT

VSCode – Cursor / Windsurf

NetSuite plugin for IDE

Understanding your environments

Stage	Characteristics	When to use
Cursor / Windsurf IDE	Full linting, autocomplete, source control Integrated testing	Production-grade scripts
Script File Editor	Copy/paste into .js file Basic syntax highlighting	Small teams; repeated tweaks
ChatGPT UI Only	Rapid one-off scripts Minimal setup	Quick prototypes, demos

The human side: Guiding principles

Context is king

- Provide schema, field IDs, sample records

Clear and scoped goals

- One script type per prompt; specify entry point & record

Error-handling and governance

- Ask AI to include try/catch, usage logging, limits

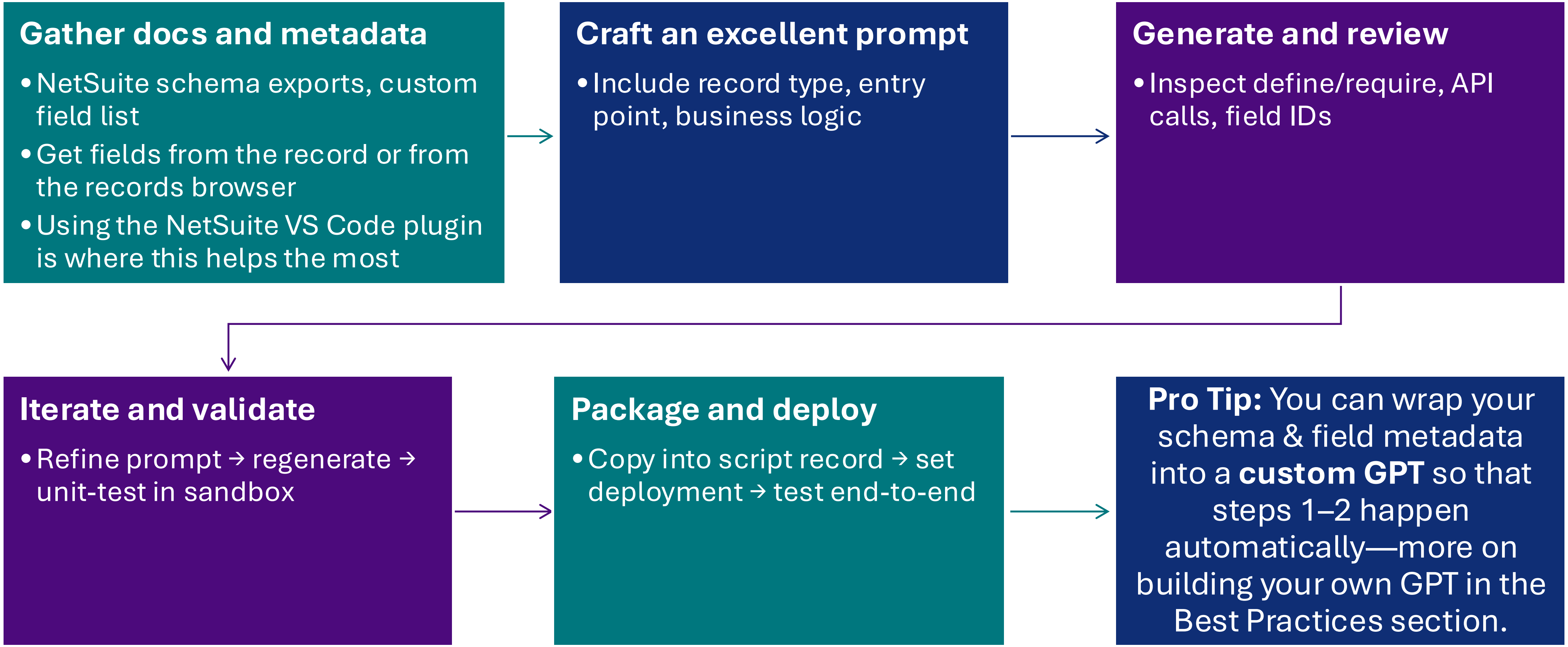
Iterate with intention

- Review output, test, then refine your prompt

Modular and maintainable

- Break large tasks into focused prompts; reuse prompts as templates

AI-side workflow: From docs to deployment



Gather docs and metadata

- NetSuite schema exports, custom field list
- Get fields from the record or from the records browser
- Using the NetSuite VS Code plugin is where this helps the most

Craft an excellent prompt

- Include record type, entry point, business logic

Generate and review

- Inspect define/require, API calls, field IDs

Iterate and validate

- Refine prompt → regenerate → unit-test in sandbox

Package and deploy

- Copy into script record → set deployment → test end-to-end

Pro Tip: You can wrap your schema & field metadata into a **custom GPT** so that steps 1–2 happen automatically—more on building your own GPT in the Best Practices section.

Craft an excellent prompt

Define scope and entry point

- Record type (e.g., journalentry), event (pageInit, fieldChanged, beforeLoad)

Embed context snippets

- Paste field-list excerpt or sample JSON
- Link to business rules (e.g. “If memo blank → alert”)

Specify output requirements

- Include define([…], fn) scaffold
- Ask for error handling & governance logging

Frame with examples

- “Here’s a working snippet for date logic—fit into this pattern”

Set constraints

- Max 100 lines, single function, no external libs

```
# Context
-Record Type: `salesorder`
-    Fields:
-    - `custbody_est_ship_date` (Date)
-    - `custbody_special_order` (Checkbox)
# Business Rules
1.   On **pageInit** and **fieldChanged**, default
     `custbody_est_ship_date` to **today + 5 days**.
2.   If `custbody_special_order` is **checked**, set
     `custbody_est_ship_date` to **today + 21 days** instead.

# Requirements
-    SuiteScript 2.x Client Script
-    Events: `pageInit`, `fieldChanged`-
-    Include AMD `define([...], function(...) { ... })` scaffold-
-    Add governance logging via `N/log`-
-    Prevent manual overrides after initial calculation-
-    Use internal field IDs- Include inline comments explaining each
     section
-    Write the script text to satisfy these requirements.
```

Client Script

Write a SuiteScript 2.x Client Script for the "journalentry" record that on pageInit sets the journal date field ("trandate") to the last day of the current month (instead of today). Then, in fieldChanged, if any line’s "memo" field is blank, display an alert and prevent the user from leaving that line. Include N/log governance logging.

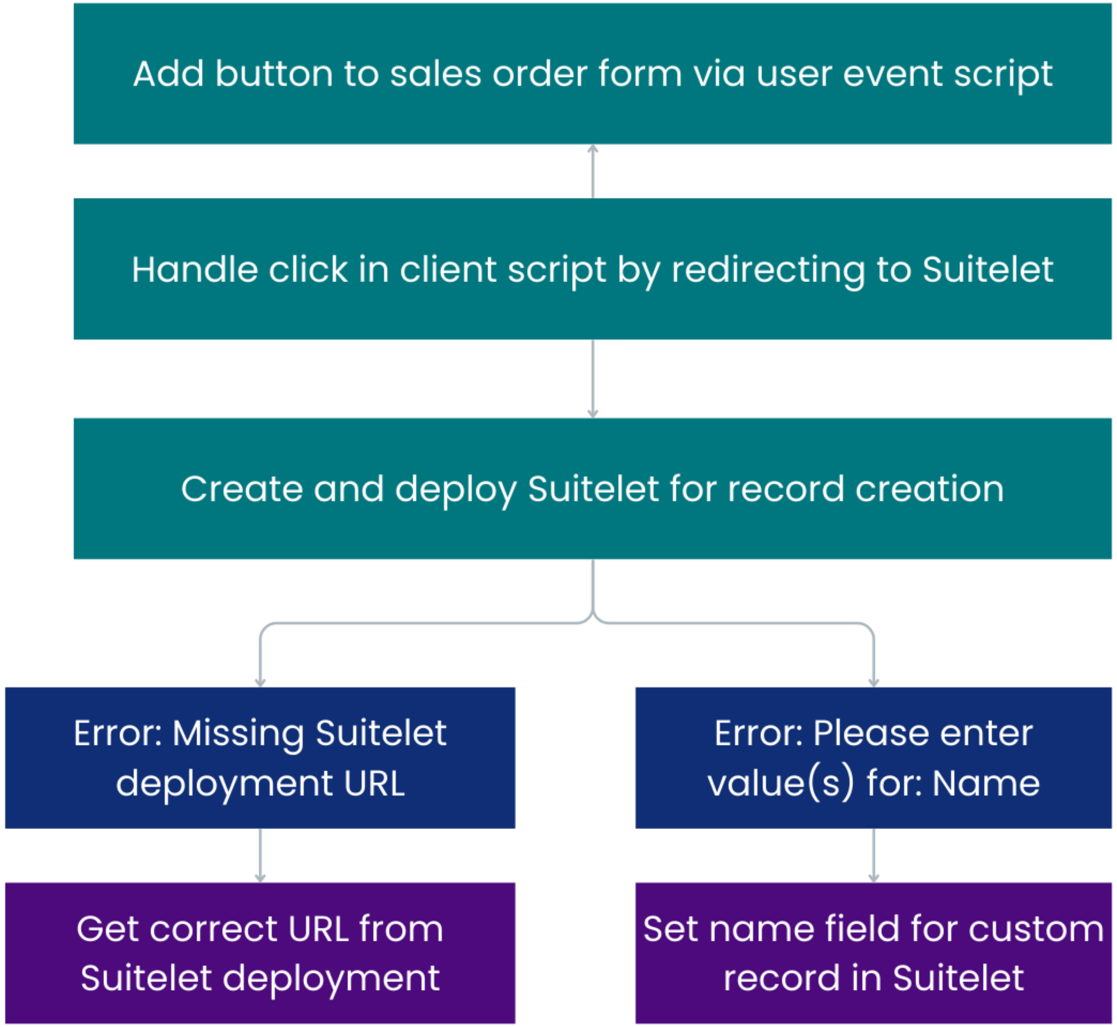
UE button and Suitelet

Create a SuiteScript 2.x User Event script on "journalentry" that, in beforeLoad for VIEW, adds a button labeled "Create Audit Report". The button’s function should call a client redirect to a Suitelet with param "jeId" = the journal entry’s internalId.

Write a Suitelet in SuiteScript 2.x that accepts "jeId" as a parameter, creates a "note" record titled "Audit Report for JE <jeId>", opens that note in edit mode (redirect to record URL), and logs success.

Additional scripting scenario

```
# Context- Record Type: `salesorder` - Goal: Add a custom button on the form
that triggers client-side logic.# Business Rules1. **User Event Script**
(`beforeLoad`, mode=VIEW):   - Add a button labeled `Trigger Action` via
`form.addButton`.   - Button function name: `handleAction`.2. **Client
Script**:   - Define `handleAction(context)` that displays an alert or UI
message:     > "Enter any action you would like here."   - Use `N/ui/message`
or `window.alert` for simplicity.# Requirements- SuiteScript 2.x- Two files:  1.
`SalesOrderButton_UE.js` (User Event script)  2. `SalesOrderButton_CL.js`
(Client script)- AMD `define([...], function(...) { ... })` scaffold in each- Load
`N/ui/serverWidget`, `N/log` in UE; `N/currentRecord`, `N/ui/message`,
`N/log` in CL- Include inline comments explaining each block- Add
governance/logging calls where appropriate# Deliverables- Complete
JavaScript code for both scripts- Deployment instructions: upload paths,
script record setup, apply to Sales Order
```



Add button to sales order form via user event script

Handle click in client script by redirecting to Suitelet

Create and deploy Suitelet for record creation

Error: Missing Suitelet deployment URL

Error: Please enter value(s) for: Name

Get correct URL from Suitelet deployment

Set name field for custom record in Suitelet

DEMO

Client script for validations

User Event & Client Script

I want a button for that!

Top 5 do's and don'ts

Do Provide rich context (field lists, sample JSON).
Don't Assume AI already knows your schema.

Do Scope one script/event per prompt.
Don't Jam multiple features into a single request.

Do Review & test generated code in your sandbox.
Don't Blindly copy-paste without validation.

Do Ask for try/catch blocks and governance logging.
Don't Neglect edge-case or error handling.

Do Document your prompt versions and code changes.
Don't Leave AI output undocumented or untracked.

Resources

Creating a Custom GPT

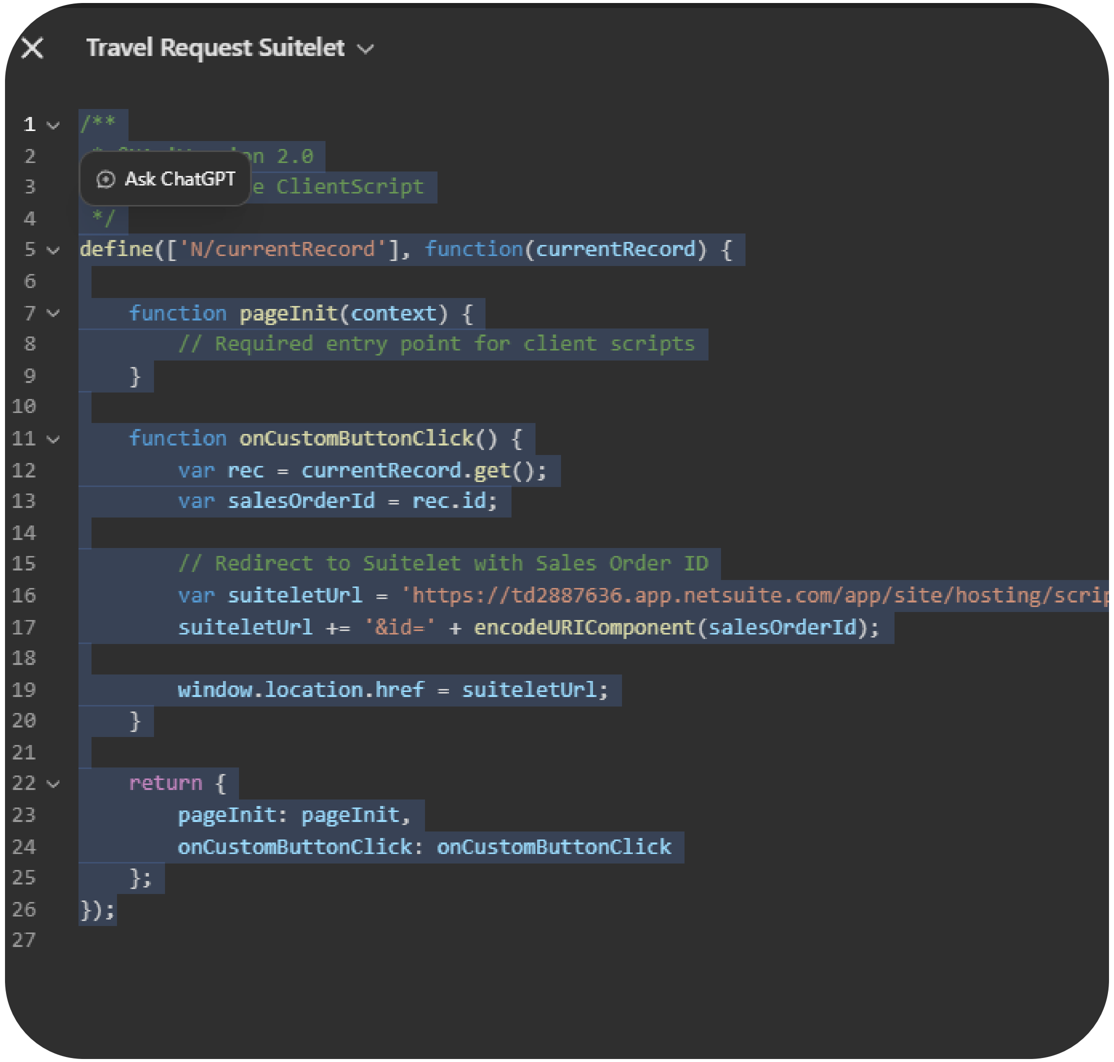
- **Define Your Domain**
  - List record types, custom fields, naming conventions
- **Gather Reference Docs**
  - CSV of field metadata, existing script samples, style guides
- **Use the GPT Builder**
  - Upload docs, configure system instructions & example prompts
- **Test & Iterate**
  - Run sample prompts, refine instructions, add missing context
- **Maintain & Update**
  - Periodically refresh metadata as your schema evolves

Helpful resources

- **NetSuite Help Center**
  - SuiteScript 2.x API reference, code examples, governance guidelines
- **SuiteAnswers**
  - Real-world articles, sample scripts, troubleshooting tips
- **OpenAI Cookbook**
  - Prompt patterns, best practices, reusable templates
- **Custom GPT Builder**
  - Embed your field metadata and style guides into a private assistant
- **Youtube: vibecoding-examples**
  - Starter repo with Client, UE, and Scheduled script samples
- **SuiteCloud Developer Tools**
  - CLI, VS Code extension, local testing frameworks

Learning objectives

- Describe the AI-driven “vibe-coding” lifecycle and how it fits into NetSuite development.
- Craft prompts that generate production-ready SuiteScript 2.x code for common use cases.
- Deploy and test Client, Scheduled, and User Event scripts in NetSuite.
- Apply core vibe-coding best practices—providing strong documentation, clear context, well-scoped instructions, and iterative prompt refinement.
- Identify next steps and resources to accelerate your own AI-customization projects.



```
/**
             on 2.0
             e ClientScript
 */
define(['N/currentRecord'], function(currentRecord) {

    function pageInit(context) {
        // Required entry point for client scripts
    }

    function onCustomButtonClick() {
        var rec = currentRecord.get();
        var salesOrderId = rec.id;

        // Redirect to Suitelet with Sales Order ID
        var suiteletUrl = 'https://td2887636.app.netsuite.com/app/site/hosting/scrip
        suiteletUrl += '&id=' + encodeURIComponent(salesOrderId);

        window.location.href = suiteletUrl;
    }

    return {
        pageInit: pageInit,
        onCustomButtonClick: onCustomButtonClick
    };
});
```

Q&A

Netgain Accounting
SUMMIT 25

Netgain Accounting
SUMMIT 25

Thank you

Scan the QR code to share your thoughts on the session