SuiteScript

An introduction to NetSuite customization

What Is SuiteScript?

- Enhanced Automation
- Custom Business Logic
- Scripting language based on JavaScript

Enhanced Automation

- Automates business processes such as order processing, inventory management, and financial reporting, reducing manual effort and increasing efficiency.

- Example: Inventory Reorder Automation
  - Use a **Scheduled Script** to run daily checks on inventory levels.
  - When an item falls below the predefined threshold, the script automatically generates a purchase order to the preferred vendor.
  - The script can also send notifications to the procurement team for tracking purposes.

Custom Business Logic

- Allows businesses to implement custom validations, calculations, and complex logic that are not available in standard NetSuite features.

- Example: Custom Discount Calculation on Sales Orders
  - A **User Event Script** runs when a sales order is created or edited.
  - The script checks the customer type (e.g., VIP, regular) and order amount to apply a tiered discount rate.
  - It updates the sales order's discount field with the calculated value and adjusts the total accordingly.

Basics of JavaScript

Why JavaScript?

- SuiteScript is JavaScript with NetSuite - specific functionality.

There are near-endless resources online for learning JavaScript.
We will touch briefly on some of the core concepts in this tutorial.

Core JavaScript Concepts:

1. Data Types
2. Variables
3. Operators
4. Control Structures
5. Functions

Variables

Variables are used to store data. The basic building blocks of your code

- let (Can be changed later in code)

```
let total = 0; // total can be updated later
total = 100; // reassigned value
```

- const (Can’t be changed later in code)

```
const TAX_RATE = 0.07; // This value should not change
TAX_RATE = 0.10 // Attempting to reassign TAX_RATE would cause an error
```

Additional Information

Data Types

Types of data supported by JavaScript. These are typically stored in variables

- Number
- String (enclosed in parenthesis)
- Boolean (true/false)
- Null
- Undefined
- Objects
- Arrays

```
let quantity = 10; // Number
let itemName = 'Laptop'; // String
let isAvailable = true; // Boolean
let lineItem = null; // Null (used often when a record does not exist)
let willUseLater; // Undefined (Create the empty variable to be assigned later)
```

Additional Information

Collections of Data in JavaScript

- Objects
  - An object in JavaScript is a collection of related data and functionalities, represented as key-value pairs.
- Arrays
  - Arrays are lists of data
  - Can be any data type

[Object additional info]
[Arrays additional info]

```
// Creating a simple customer object (this data could come from a Saved Search)
let customer = {
    firstName: 'John',              // Customer's first name (Key = firstnName, Value = John)
    lastName: 'Doe',                // Customer's last name (Key = lastName, Value = Doe)
    email: 'john.doe@example.com',  // Customer's email
    phone: '555-1234'               // Customer's phone number
};

/*
Set the value of a custom field in Netsuite using
the customer object
*/
record.setValue({
    fieldId: 'custbody_firstName',
    value: customer.firstName
})

// Arrays are lists enclosed in brackets []
let listOfCustomers = [customer, customer2, customer3]
```

Operators

Operators perform operations on variables and values.

- Arithmetic operators (+, -, *, /, %.)
- Comparison Operators (==, !=, ===, !==, >, <.)
- Logical Operators (&&, ||, !)

Additional Information

Arithmetic Operators

Arithmetic operators are used to perform basic mathematical operations on numbers.

+ (Addition)

- (Subtraction)

* (Multiplication)

/ (Division)

% (Modulus)

```
// + (Addition): Adds two numbers.
let subtotal = 200;
let tax = 20;
let total = subtotal + tax; // 220
log.debug('Total Amount', total);

// - (Subtraction): Subtracts one number from another.
let price = 100;
let discount = 10;
let finalPrice = price - discount; // 90
log.debug('Final Price', finalPrice);

// * (Multiplication): Multiplies two numbers.
let quantity = 5;
let unitPrice = 20;
let totalCost = quantity * unitPrice; // 100
log.debug('Total Cost', totalCost);

// / (Division): Divides one number by another.
let totalAmount = 200;
let numberOfItems = 4;
let averagePrice = totalAmount / numberOfItems; // 50
log.debug('Average Price', averagePrice);


// % (Modulus): Returns the remainder of a division.
let totalItems = 11;
let itemsPerBox = 4;
let remainingItems = totalItems % itemsPerBox; // 3
log.debug('Remaining Items', remainingItems);
```

Comparison Operators

Comparison operators are used to compare values and return a boolean (true or false).

== (Equality)

=== (Strict Equality)

!= (Inequality)

> (Greater than)

< (Less Than)

>= (Greater than or equal to)

<= (Less than or equal to)

```
// == (Equality): Checks if two values are equal (loose comparison).
let orderStatus = 'active';
if (orderStatus == 'active') {
    log.debug("Orders status is Active")
}

// === (Strict Equality): Checks if two values are equal and of the same data type.
let itemPrice = 25
if (itemPrice === "25"){//Since item price is a NUMBER data type and "25" is a STRING data type return WILL NOT be reached
    return
}
if (itemPrice == 25){ // since item price and 25 are both NUMBER data types return WILL be reached
    return
}

// != (Inequality) also (!== Strict Inequality) : Checks if two values are not equal.
const REQUIRED_RECORD_TYPE = 'SalesOrder';
let editedRecordType = 'CashSale'
if (REQUIRED_RECORD_TYPE != editedRecordType){
    throw error; // Since the required record type doesn't match the record being edited we don't want the script to continue
}

// > (Greater Than): Checks if the left value is greater than the right value. (Same with < <= and >=)
let stockLevel = 5;
let reorderThreshold = 10;
let needsReorder = stockLevel < reorderThreshold; // true
log.debug('Needs Reorder', needsReorder);
```

Logical Operators

Logical operators are used to combine or invert boolean values.

&& (Logical AND)

|| (Logical OR)

! (Logical NOT)

```
// && (Logical AND): Returns true if both conditions are true.
let isInStock = true;
let canSell = isAvailable && isInStock; // true


//|| (Logical OR): Returns true if at least one condition is true.
let hasPermission = false;
let isAdmin = true;
let canAccess = hasPermission || isAdmin; // true


// ! (Logical NOT): Inverts the boolean value; true becomes false and vice versa.
let isComplete = false;
let isPending = !isComplete; // true
```

Control Structures

Additional Information

Control structures manage the flow of code execution.

Common Control Structures:

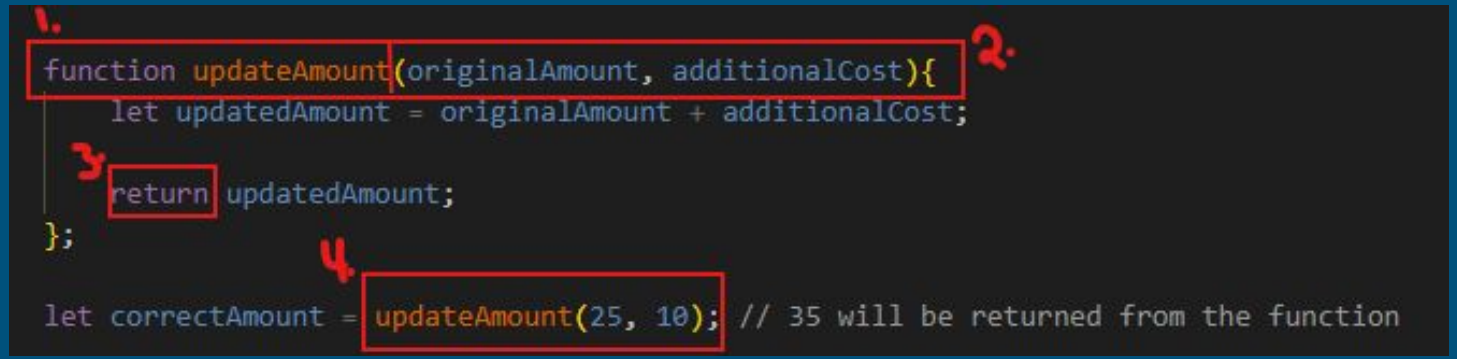
- If - Else Statements
  - Perform logic if a statement is true
- For Loops
  - Runs a predetermined number of times

```
// If / Else Statement, can be chained with mulitple else's
if (customerName === 'John Doe') {
    log.debug('Customer Found', 'Customer is John Doe');
} else {
    log.debug('Customer Not Found', 'Different Customer');
}

//                  index 0     index 1    index 2
let customerList = [customer1, customer2, customer3]
// For loop, iterate (loop) over data
for (let i = 0; i < 3; i++) {
    log.debug(customerList[i]); // This will log the value of customerList at the stated index
}

// let i = 0 initiallizes the counter variable,
// i < 3 means this loop will continue while the value of i is less than 3
// i++ at the end of every loop the value of i will increase by 1.
// The loop will end and the code will continue on once the "i < 3" value is no longer true
```

Functions

Functions are reusable blocks of code that are often given parameters (input) and produce a result (output) when called.

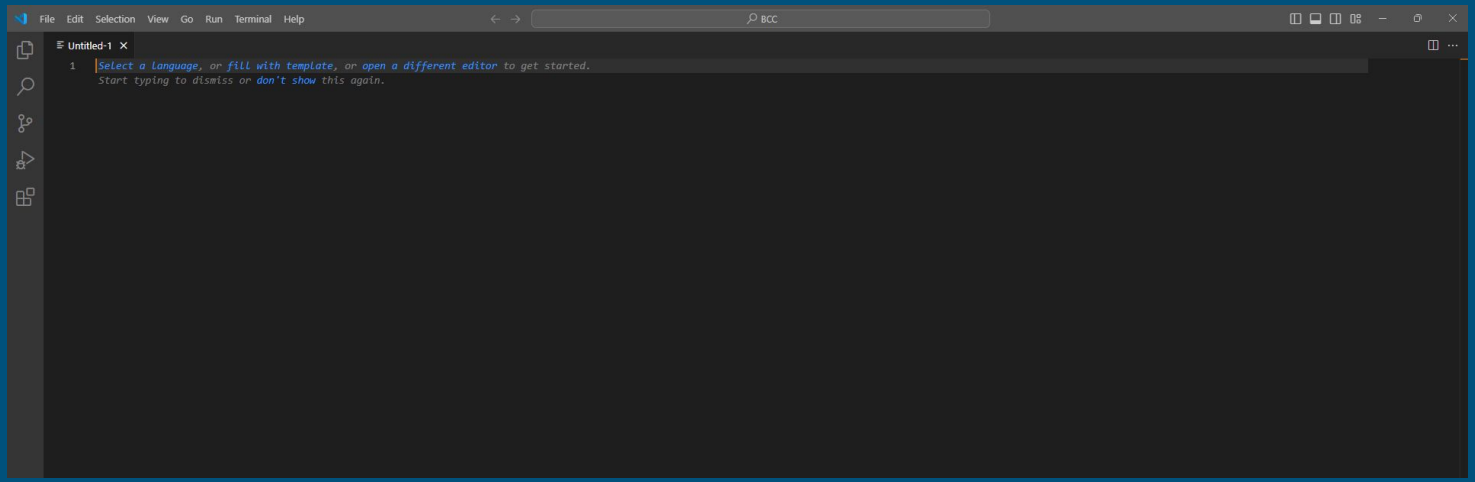
1. Declare the function
2. Declare parameters
3. Return a value
4. Call the function



Download a Text Editor

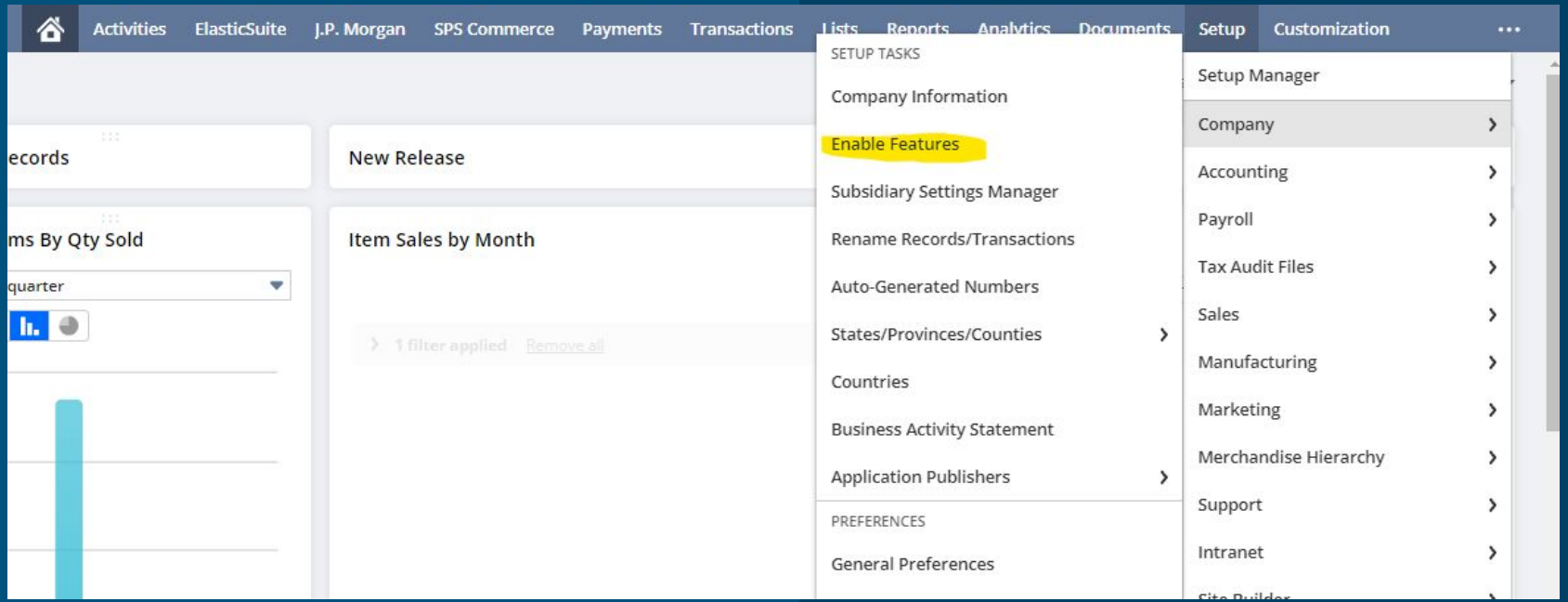
This is where you will write your code.

- Visual Studio Code

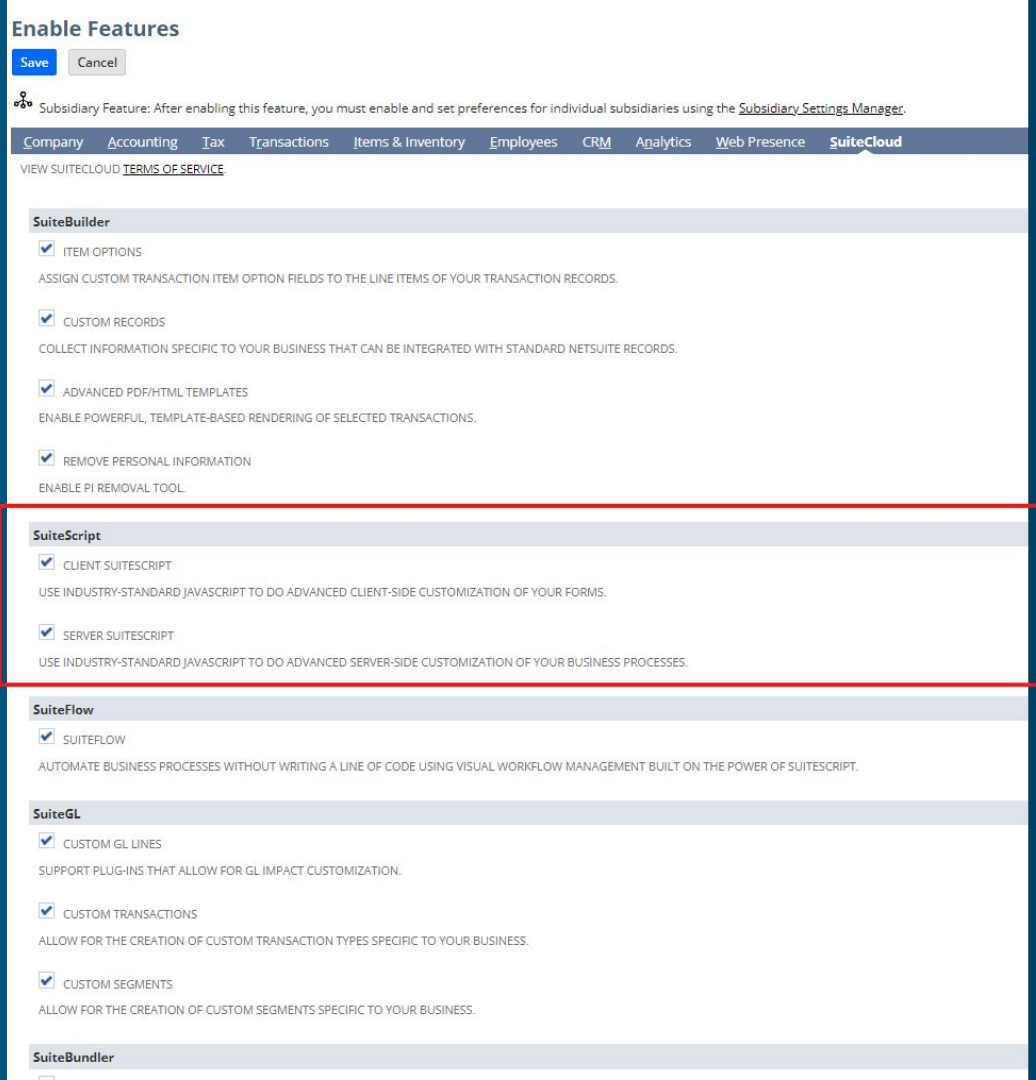


Setting up Netsuite Account to Use SuiteScript

Setup -> Company -> Enable Features



Enable Client and Server SuiteScript



Roles + Permission

- NetSuite Administrator role has full SuiteScript access
- Setting up Custom Roles + Permissions

Accessing SuiteScript API Documentation

NetSuite provides rich documentation about how to use SuiteScript

1. Access the SuiteScript API documentation
2. Use the search bar to find specific help
3. Use the side menu to look through all available documentation

What did we learn?

1. What SuiteScript is
2. Foundations of JavaScript
3. Enabling SuiteScript in your NetSuite account