Professional Certificate in Google Apps Script Fundamentals

Setting Up Your Development Environment

Setting up a development environment is an essential step in learning Google Apps Script (GAS) as it provides the tools and resources necessary to write, test, and debug your code. In this explanation, we will cover the key terms and vocabulary related to setting up a development environment for the Professional Certificate in Google Apps Script Fundamentals.

1. Google Account: A Google Account is required to access GAS and other Google services. You can create a new account or use an existing one.
2. Google Drive: Google Drive is a cloud storage service that allows you to store, share, and access files from any device. GAS code is stored in Google Drive as scripts bound to Google Sheets, Docs, or Forms.
3. Google Sheets, Docs, and Forms: These are Google's productivity apps that allow you to create and edit spreadsheets, documents, and forms, respectively. GAS can be used to automate tasks and add functionality to these apps.
4. Google Apps Script (GAS): GAS is a JavaScript-based scripting language developed by Google for automating tasks and extending functionality in Google Workspace apps.
5. Google Apps Script Editor: The GAS Editor is a web-based Integrated Development Environment (IDE) where you can write, test, and debug your code.
6. Code Editor: The Code Editor is where you write your GAS code. It includes features such as syntax highlighting, code completion, and error checking.
7. Apps Script Dashboard: The Apps Script Dashboard is where you can view and manage your scripts, projects, and triggers.
8. Project: A project is a collection of related GAS files, including code, libraries, and resources.
9. Bound Script: A bound script is a script associated with a specific Google Sheets, Docs, or Forms file.
10. Standalone Script: A standalone script is a script that is not associated with any specific Google Sheets, Docs, or Forms file.
11. Libraries: Libraries are reusable code that can be shared across multiple projects.
12. Triggers: Triggers are automated events that run your GAS code in response to specific actions or time-based events.
13. Deployment: Deployment is the process of publishing your GAS code as a web app or add-on.
14. Web App: A web app is a GAS-powered application that can be accessed from any web browser.
15. Add-on: An add-on is a GAS-powered extension that adds functionality to Google Sheets, Docs, or Forms.
16. Manifest File: The manifest file is a JSON file that defines the properties and settings of your GAS project.
17. Code Execution: Code execution is the process of running your GAS code.
18. Debugging: Debugging is the process of identifying and fixing errors in your GAS code.
19. Stack Trace: A stack trace is a list of function calls leading up to an error in your GAS code.
20. Logging: Logging is the process of recording messages and errors in your GAS code for debugging and

monitoring purposes.

Setting up your development environment for GAS involves the following steps:

1. Create a new Google Account or log in to an existing one.
2. Open Google Drive and create a new Google Sheets, Docs, or Forms file.
3. Open the file and click on "Extensions" > "Apps Script."
4. Welcome to the GAS Editor! Here, you can write, test, and debug your code.
5. To create a new project, click on "File" > "New" > "Project."
6. To create a bound script, open a Google Sheets, Docs, or Forms file and click on "Extensions" > "Apps Script."
7. To create a standalone script, click on "File" > "New" > "Standalone script."
8. To add a library, click on "Resources" > "Libraries" and search for the library you want to add.
9. To create a trigger, click on "Edit" > "Current project's triggers" and click on the "Add Trigger" button.
10. To deploy your code as a web app or add-on, click on "Publish" > "Deploy as web app" or "Publish" > "Publish add-on."

Here are some practical applications and challenges to help you get started with your GAS development environment:

1. Create a new Google Sheets file and bound script. Write a function that adds a new row to the sheet with a timestamp and your name. Test and debug the function.
2. Create a new standalone script. Write a function that takes a string as input and returns the string in reverse order. Test and debug the function.
3. Add the "Ugly Editors" library to your standalone script. Write a function that uses the library to format a string with rainbow colors. Test and debug the function.
4. Create a new trigger that runs your reverse string function every hour. Monitor the trigger logs to ensure it runs correctly.
5. Deploy your standalone script as a web app. Share the web app URL with a friend and ask them to test it. Debug any issues that arise.
6. Create a new add-on for Google Sheets that adds a custom menu with a button to reverse the contents of a selected range. Test and debug the add-on.
7. Use the GAS Logger to record messages and errors in your code. Monitor the logs to ensure your code runs correctly.
8. Debug a stack trace in your GAS code. Identify the cause of the error and fix it.
9. Create a new Google Forms file and bound script. Write a function that sends a notification email when a new response is submitted. Test and debug the function.
10. Use the GAS Manifest file to define the properties and settings of your project. Test and debug the manifest file.

In conclusion, setting up a development environment for GAS involves understanding key terms and vocabulary, creating projects, and using the GAS Editor to write, test, and debug your code. By following the steps outlined in this explanation and completing the practical applications and challenges, you will be well on your way to becoming a proficient GAS developer.