Professional Certificate in Google Apps Script Advanced Techniques

Leveraging Google Apps Script Libraries

Google Apps Script (GAS) is a JavaScript-based scripting language developed by Google for light-weight application development in the G Suite platform. GAS libraries are reusable pieces of code that can be shared and used across different scripts and projects. Leveraging GAS libraries can save development time, promote code consistency, and enable collaboration among developers. This explanation will cover key terms and vocabulary related to using GAS libraries, focusing on practical applications and challenges.

Library: A library in GAS is a container for functions and variables that can be shared and reused across different scripts and projects. Libraries can be created, published, and installed to enable code reuse and collaboration.

Publishing a Library: To publish a library, you need to create a new project in the Google Apps Script editor, write and test your code, then follow these steps:

1. Go to `File > Project properties` and set a **Project key**
2. Go to `Publish > Deploy from manifest...`
3. Select `New deployment`
4. Choose `Web app` as the deployment type
5. Set the access level to `Anyone, even anonymous`
6. Click `Deploy`
7. Copy the **Current web app URL**

Now you can share this URL with others to allow them to install the library.

Installing a Library: To install a library in your project, you need to follow these steps:

1. Go to `Resources > Libraries`
2. Paste the library's project key or web app URL in the **Find a library** field
3. Click `Add`
4. Set the **Version** and **Library alias**
5. Click `Save`

Now you can use the library's functions in your code by calling them with the library alias.

Library Alias: A library alias is a short name assigned to a library when installing it in your project. The alias is used to call the library's functions in your code.

Library Versioning: GAS libraries support versioning to manage changes and ensure compatibility. When publishing a library, a new version is created automatically with each deployment. When installing a library, you can choose the version you want to use.

Library Functions: To call a library's function in your code, you need to use the library alias followed by a dot (.) and the function name. For example, if your library alias is `MyLib` and the function name is `doSomething`, you can call it like this:

`MyLib.doSomething();`

Library Variables: Library variables can be accessed and modified using the library alias, just like functions. To set a variable, you can use the library alias followed by a dot (.) and the variable name. For example:

`MyLib.someVariable = "new value";`

Library Callbacks: Library functions can be used as callbacks in your code. To do this, you need to assign the library function to a variable in your code and use the variable as the callback function. For example:

`var myCallback = MyLib.myCallbackFunction;`
`myButton.addClickHandler(myCallback);`

Library Security: To ensure security and prevent unauthorized access, you can use the `onInstall(e)` function in your library to check the installing user's email address or other properties. If the user is not authorized, you can throw an error or display a warning message.

Library Documentation: Documenting your library functions and variables is crucial for understanding their purpose and usage. You can use JSDoc comments to document your library. JSDoc comments are written in a specific format and can be automatically parsed by documentation generators like JSDoc or Closure Compiler.

Here's an example of JSDoc comments for a library function:
```javascript
/**
 * Adds two numbers and returns the result.
 *
 * @param {number} a The first number.
 * @param {number} b The second number.
 * @return {number} The sum of the two numbers.
 */
function addNumbers(a, b) {
  return a + b;
}
```
By following these best practices and guidelines, you can effectively leverage GAS libraries to improve your G Suite application development productivity and quality.

In summary, GAS libraries provide a powerful way to reuse code and promote collaboration among developers. By understanding key terms such as library, library alias, versioning, and security, you can create,

publish, install, and use libraries to enhance your G Suite app development experience. Following best practices such as documentation and versioning can help ensure a smooth and efficient development process.