Objective: BUILD AN APP SCRIPT TO SEND OUT A ‘THANK YOU EMAIL’ AS A RESPONSE WHEN SOMEONE FILLS UP A GOOGLE FORM

Theory:

Google Apps Script is a powerful cloud-based scripting language and development platform provided by Google. It allows users to extend and automate the functionality of various Google services, such as Google Sheets, Google Docs, Gmail, Google Drive, and more. Here are some key theories related to Google Apps Script:

Google Apps Script is built on JavaScript, a widely used programming language. Understanding the basics of JavaScript, including variables, data types, loops, and conditional statements, is crucial for working with Apps Script effectively. Apps Script follows an event-driven programming model. It enables developers to respond to specific events or triggers, such as opening a document, submitting a form, or a time-based trigger. By attaching functions to these triggers, developers can automate tasks and perform actions based on user interactions.

Apps Script provides built-in integration with various G Suite services, allowing developers to access and manipulate data within these services programmatically. For example, you can read data from Google Sheets, send emails using Gmail, create calendar events, access Google Drive files, and interact with Google Docs, among other capabilities. Google Sheets in particular offers the ability to create custom functions using Apps Script. Custom functions allow you to extend the formula capabilities of Google Sheets by implementing your own formulas and calculations. These functions can be used in the same way as built-in functions in Google Sheets.

Apps Script also supports advanced Google APIs, enabling developers to integrate with external services and extend functionality beyond the core Google services. This includes accessing Google Analytics data, interacting with Google Calendar, using the Google Drive API, and more. The ability to leverage external APIs greatly expands the capabilities of Apps Script. Apps Script offers various deployment options, allowing you to run your scripts in different contexts. You can deploy scripts as web apps, add-ons, or as API executables. Web apps can be accessed by users via a URL, add-ons enhance the functionality of Google products, and API executables enable script execution via API calls.App Script enables collaboration and sharing of scripts with others. You can share your scripts with specific users or groups, and multiple users can work together on the same script simultaneously. This promotes teamwork, code reuse, and knowledge sharing within organizations.

Source Code:

We need to follow these steps before writing any code in the app script editor
1. Create a google form inside your google drive.
2. Setup the form so that it collects email addresses by default:
   2.1 Go to Settings -> responses -> collect email address and select 'Responder input'.
   2.2 Do the same in the Settings -> default -> Form defaults section.
3. Go to Responses tab and click on 'view in sheets', a google sheet with the responses opens.
4. In the google sheet, click on extensions -> App Scripts, an app script editor opens,
5. Write the onSubmit() function as shown below and run the script; allow the required permissions.
6. Click on 'triggers' menu on the left sidebar and create a new trigger for the onSubmit function.
   6.1 Put the source as the google sheet.
   6.2 Put the event as 'on form submit' and click on submit.

```
function onSubmit(e){
   const myemail = Session.getActiveUser().getEmail();
   const email = e.namedValues['Email Address'][0];

   MailApp.sendEmail({
     to: email,
     subject: "Ice cream flavor survey",
     htmlBody: "Thank you very much for participating in our ice cream flavor survey."
   });
}
```

Conclusion:

In this lab experiment we were familiarized with Google's App Script Service. We learned how to work with different Google services like mail, forms and sheets programmatically using the app scripts.