Apps Script with Gemini
Quick Start Guide

1. What you're building

By the end of this guide you'll know how to:

- Call **Gemini** from **Apps Script** using the REST API
- Keep your **API key** out of your code
- Build:

    - A simple **logger demo**
    - A **Sheets custom function**: `=GEMINI_COMPLETE("Write a haiku")`
    - A **Sheets sidebar helper**
    - A **Docs summarizer**
    - A basic **Gmail reply drafter**

All examples use the Gemini REST API (`generativelanguage.googleapis.com`) with an API key from **Google AI Studio**.

---

2. One-time setup

2.1 Get a Gemini API key

1. Go to **Google AI Studio** (Gemini API).
2. Create or choose a project.
3. Open **API Keys** and click **Create API key** (or reuse an existing one).
4. Copy the key somewhere safe (it looks like a long random string).

You'll use this key in Apps Script to authenticate calls to the Gemini API.

2.2 Create an Apps Script project

You can start in any Workspace app (Sheets, Docs, etc.). I'll assume **Sheets** first:

1. Open a new or existing **Google Sheet**.
2. Go to **Extensions** → **Apps Script**.
3. Delete any code in `Code.gs`.

2.3 Store your API key securely (Script Properties)

1. In Apps Script, go to **Project Settings (⚙ icon)**.
2. Next to **Script properties**, click **Open in new tab** (or **Add script property**).
3. Add:

    - **Key**: `GEMINI_API_KEY`
    - **Value**: `YOUR_ACTUAL_API_KEY_HERE`
4. Save.

Inside code, you'll read it like this:

```
function getGeminiApiKey_() {
  const scriptProps = PropertiesService.getScriptProperties();
  const key = scriptProps.getProperty('GEMINI_API_KEY');
  if (!key) {
    throw new Error('GEMINI_API_KEY is not set in Script
Properties.');
  }
  return key;
}
```

Using Script Properties is the recommended way to keep secrets out of your source code in Apps Script.

---

3. Core helper: call Gemini from Apps Script

Let's build a simple wrapper around the REST endpoint:

We'll use:

- Endpoint: `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent`

Add this to `Code.gs`:

```
/**
 * Core helper: call Gemini with a text prompt and get back a string.
 *
 * @param {string} prompt The user prompt to send to Gemini.
 * @param {string} [modelId] Optional model ID, defaults to
gemini-2.5-flash.
 * @returns {string} The plain text response from Gemini.
```

```
 */
function callGemini_(prompt, modelId) {
  modelId = modelId || 'gemini-2.5-flash';

  const url =
`https://generativelanguage.googleapis.com/v1beta/models/${modelId}:ge
nerateContent`;

  const apiKey = getGeminiApiKey_();

  const payload = {
    contents: [
      {
        parts: [
          { text: prompt }
        ]
      }
    ]
  };

  const options = {
    method: 'post',
    contentType: 'application/json',
    headers: {
      'x-goog-api-key': apiKey
    },
    muteHttpExceptions: true,
    payload: JSON.stringify(payload)
  };

  const response = UrlFetchApp.fetch(url, options);
  const code = response.getResponseCode();
  const text = response.getContentText();

  if (code !== 200) {
    // Log for debugging and throw an error
    console.error('Gemini error', code, text);
    throw new Error('Gemini API error: ' + code + ' ' + text);
```

```
  }

  const data = JSON.parse(text);

  // Basic extraction of first candidate text
  const candidates = data.candidates || [];
  if (!candidates.length) {
    return '';
  }
  const parts = candidates[0].content?.parts || [];
  const resultText = parts.map(p => p.text || '').join('');
  return resultText;
}
```

Now you have a reusable function `callGemini_(prompt)` you can use everywhere.

---

4. First tiny test: log a response

Add this:

```
function testGeminiSimple() {
  const prompt = 'Write a two-line motivational quote about learning
to code.';
  const reply = callGemini_(prompt);
  Logger.log(reply);
}
```

Run **testGeminiSimple**:

1. Click the ▶️ (Run) button.
2. The first time, you'll need to **authorize** the script (click through the prompts).
3. Open **View** → **Logs** to see Gemini's response.

> If you see an error like "API key not found" or HTTP 401/403, double-check the key, Script Properties, and that the key is allowed to call the Generative Language API.

---

5. Example: Sheets custom function =GEMINI_COMPLETE()

Let's make a beginner-friendly custom function.

Add this to `Code.gs`:

```
/**
 * Custom function: =GEMINI_COMPLETE("write a haiku about
spreadsheets")
 *
 * @param {string} prompt
 * @return {string}
 * @customfunction
 */
function GEMINI_COMPLETE(prompt) {
  if (!prompt) {
    return 'Please provide a prompt, e.g. =GEMINI_COMPLETE("Explain
VLOOKUP in simple terms")';
  }

  try {
    const result = callGemini_(String(prompt));
    return result;
  } catch (err) {
    // Show a short error in the cell (full details in logs)
    console.error(err);
    return 'Error: ' + err.message;
  }
}
```

Usage in your Sheet:

- In any cell, type:

```
=GEMINI_COMPLETE("List 3 creative warm-up questions for an online
class.")
```

or reference a cell:

```
=GEMINI_COMPLETE(A2)
```

> Tip: Keep prompts fairly short in cells; for long prompts, store them in a separate cell and reference them.

---

6. Example: Sheets sidebar “Prompt Pad”

Let’s build a small UI inside Sheets so beginners can type a prompt and paste the result into a cell.

6.1 Create the sidebar HTML

In Apps Script:

1. Click **+** → **HTML**.
2. Name it `Sidebar.html`.
3. Paste:

```
<!DOCTYPE html>
<html>
  <head>
    <base target="_top">
    <style>
      body { font-family: Arial, sans-serif; padding: 10px; }
      textarea { width: 100%; height: 100px; }
      button { margin-top: 8px; padding: 6px 12px; }
      #output { margin-top: 10px; white-space: pre-wrap; border: 1px
solid #ccc; padding: 6px; min-height: 60px; }
      label { font-weight: bold; }
    </style>
  </head>
  <body>
    <h2>Gemini Prompt Pad</h2>
    <label for="prompt">Prompt:</label><br>
    <textarea id="prompt" placeholder="Ask Gemini
something..."></textarea><br>
```

```
    <button onclick="callGemini()">Ask Gemini</button>
    <button onclick="insertIntoSheet()">Insert into Sheet</button>
    <div id="output"></div>

    <script>
      let lastResult = '';

      function callGemini() {
        const prompt = document.getElementById('prompt').value;
        if (!prompt) {
          alert('Please enter a prompt.');
          return;
        }
        document.getElementById('output').textContent = 'Thinking...';
        google.script.run
          .withSuccessHandler(function(result) {
            lastResult = result;
            document.getElementById('output').textContent = result;
          })
          .withFailureHandler(function(err) {
            document.getElementById('output').textContent = 'Error: '
+ err.message;
          })
          .sidebarAskGemini(prompt);
      }

      function insertIntoSheet() {
        if (!lastResult) {
          alert('No result to insert yet.');
          return;
        }
        google.script.run.insertResultIntoActiveCell(lastResult);
      }
    </script>
  </body>
</html>
```

6.2 Backend functions

Back in `Code.gs`, add:

```
/**
 * Open the Gemini Prompt Pad sidebar.
 */
function showGeminiSidebar() {
  const html = HtmlService.createHtmlOutputFromFile('Sidebar')
    .setTitle('Gemini Prompt Pad');
  SpreadsheetApp.getUi().showSidebar(html);
}

/**
 * Called from the sidebar to ask Gemini.
 */
function sidebarAskGemini(prompt) {
  return callGemini_(prompt);
}

/**
 * Insert text into the currently active cell.
 */
function insertResultIntoActiveCell(text) {
  const sheet = SpreadsheetApp.getActiveSheet();
  const cell = sheet.getActiveCell();
  cell.setValue(text);
}
```

6.3 Add a custom menu to open the sidebar

Still in `Code.gs`:

```
function onOpen() {
  const ui = SpreadsheetApp.getUi();
  ui.createMenu('AI Tools')
    .addItem('Open Gemini Prompt Pad', 'showGeminiSidebar')
    .addToUi();
}
```

Reload the Sheet to see **AI Tools → Open Gemini Prompt Pad**.

---

7. Example: Summarize selected text in a Doc

Now let's jump to **Google Docs**.

7.1 Create an Apps Script bound to a Doc

1. Open a Google Doc.
2. **Extensions → Apps Script**.
3. Paste the **helper functions** from earlier (`getGeminiApiKey_`, `callGemini_`).
4. Then add:

```
function onOpen() {
  DocumentApp.getUi()
    .createMenu('AI Tools')
    .addItem('Summarize selection with Gemini',
'summarizeSelectionWithGemini')
    .addToUi();
}

/**
 * Summarize the current selection in a Google Doc.
 */
function summarizeSelectionWithGemini() {
  const doc = DocumentApp.getActiveDocument();
  const selection = doc.getSelection();
  if (!selection) {
    DocumentApp.getUi().alert('Please select some text first.');
    return;
  }

  // Collect selected text
  let text = '';
  const elements = selection.getRangeElements();
  elements.forEach(function(el) {
    if (el.getElement().editAsText) {
```

```
      const t = el.getElement().editAsText()
        .getText()
        .substring(el.getStartOffset(), el.getEndOffsetInclusive() +
1);
      text += t + '\n';
    }
  });

  const prompt =
    'Summarize the following text for a beginner in 3–5 bullet
points:\n\n' +
    text;

  const summary = callGemini_(prompt);

  // Append summary at end of doc
  const body = doc.getBody();
  body.appendParagraph('---');
  body.appendParagraph('Gemini summary:');
  body.appendParagraph(summary);
}
```

Usage:

1. Select some paragraphs in the Doc.
2. Use **AI Tools → Summarize selection with Gemini**.
3. A summary is appended to the end of the document.

---

8. Example: Gmail “Draft reply with Gemini”

This uses the GmailApp service to get the current message and draft a reply.

> For simplicity, this example runs from the script editor. More advanced flows use Gmail add-ons, which require extra config.

Create a **standalone** Apps Script project (script.google.com → New Project), add the helper functions, then:

```
/**
 * Draft a reply to the most recent email in your inbox using Gemini.
 * This is a simple demo for beginners.
 */
function draftReplyWithGemini() {
  const threads = GmailApp.getInboxThreads(0, 1); // newest thread
  if (!threads.length) {
    Logger.log('No threads in inbox.');
    return;
  }

  const thread = threads[0];
  const messages = thread.getMessages();
  const last = messages[messages.length - 1];

  const sender = last.getFrom();
  const subject = last.getSubject();
  const body = last.getPlainBody().slice(0, 2000); // keep it short
for demo
  const prompt =
    'You are a polite and concise assistant. Read the email below and
write a friendly, professional reply.\n\n' +
    'Subject: ' + subject + '\n' +
    'From: ' + sender + '\n\n' +
    body;

  const replyText = callGemini_(prompt);

  // Create a draft in the same thread
  const draftSubject = 'Re: ' + subject;
  GmailApp.createDraft(sender, draftSubject, replyText, { inReplyTo:
last.getId() });

  Logger.log('Draft created. Check Gmail drafts.');
}
```

When you run it, check your **Drafts** folder for the generated reply. Always **review and edit** before sending.

---

9. Mini examples for practice (beginners)

Here are some tiny scripts to help learners “get the feel” of prompting and code:

9.1 Turn a list into bullet points

```
function testGeminiBullets() {
  const prompt = `
Turn this list into clear bullet points for a beginner:

- HTML elements
- CSS styling
- JavaScript interactivity
`;
  Logger.log(callGemini_(prompt));
}
```

9.2 Explain code for a beginner

```
function explainCode() {
  const code = `
function add(a, b) {
  return a + b;
}
`;
  const prompt = 'Explain this JavaScript code step-by-step for a
beginner:\n\n' + code;
  Logger.log(callGemini_(prompt));
}
```

9.3 Generate quiz questions

```
function generateQuizQuestions() {
  const topic = 'variables and data types in JavaScript';
  const prompt =
    'Create 5 multiple-choice questions (with answers) for a beginner
about ' +
    topic +
```

```
    '. Use plain text only.';
  Logger.log(callGemini_(prompt));
}
```

---

10. Basic error handling & limits

When calling external APIs from Apps Script, keep in mind:

- **Quotas**:
    - Gemini API may limit requests per minute/day. Check usage in AI Studio / Cloud console.
    - Apps Script has UrlFetch timeouts and daily quotas.
- **Common errors**:
    - `401` / `403`: invalid or unauthorized API key.
    - `429`: too many requests; slow down.
    - `400`: malformed request (check JSON).
- **Tips**:
    - Start small, log responses.
    - Wrap `callGemini_` calls in `try/catch`.
    - Avoid calling Gemini in tight loops (e.g., 10,000 rows at once). Batch or limit.

Example with catch:

```
function safeGeminiExample() {
  try {
    const result = callGemini_('Say hello in one short sentence.');
    Logger.log(result);
  } catch (e) {
    Logger.log('Something went wrong: ' + e.message);
  }
}
```