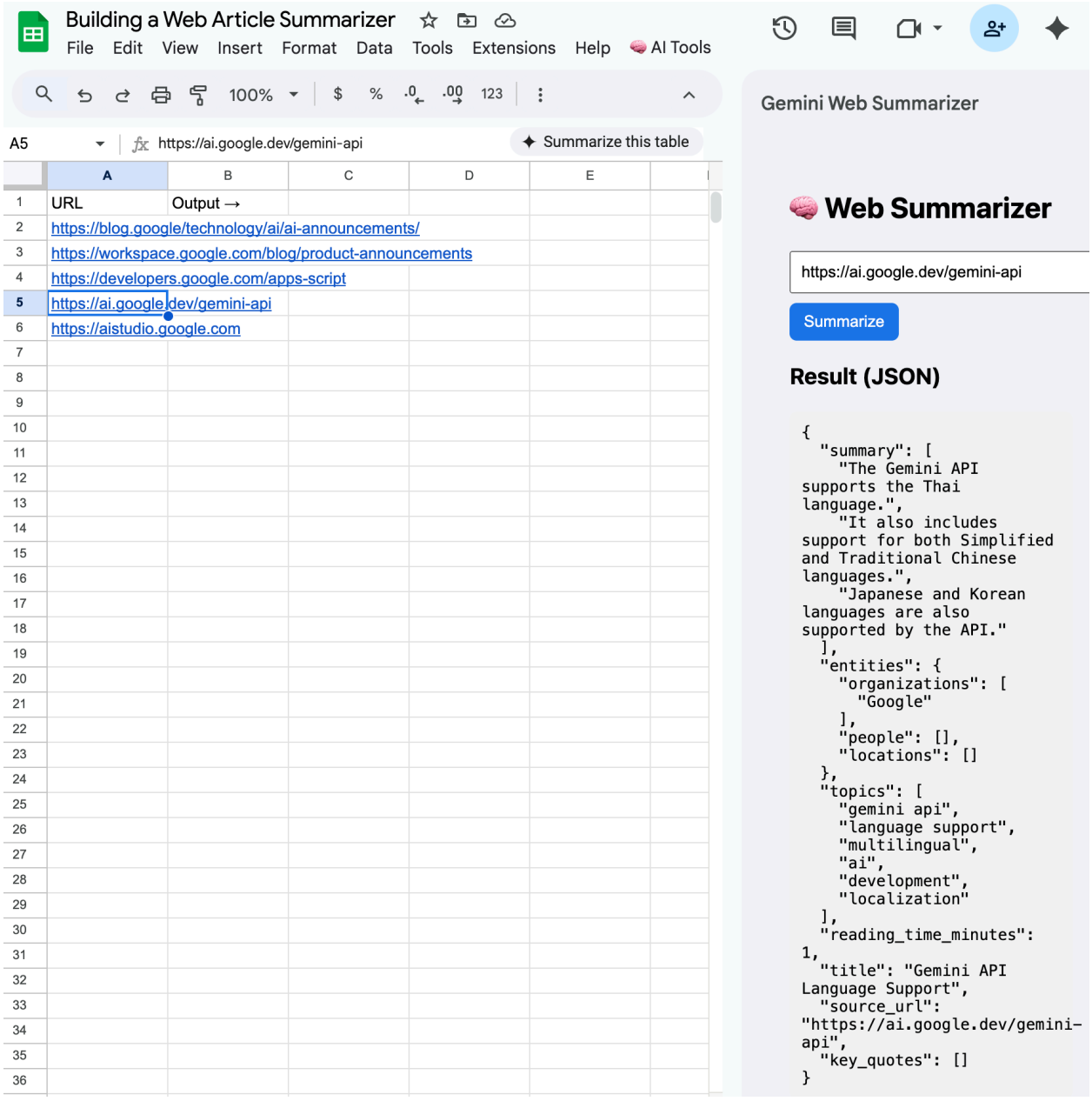
🌐 Building a Web Article Summarizer & Citation Extractor with Google Sheets + Gemini AI



Have you ever wished you could drop a list of article URLs into Google Sheets and instantly get structured summaries — complete with titles, key quotes, named entities, and estimated reading time?
In this exercise, we'll create exactly that using **Google Apps Script** and **Gemini AI**.

🧠 Project Overview

This project transforms your spreadsheet into a lightweight research assistant.[Select random word python]

You'll feed it article URLs, and the script will:

- Fetch and clean each webpage
- Summarize its main points in bullet form
- Extract **people**, **organizations**, and **locations**
- Highlight **key quotes**
- Suggest topic tags and reading time
- Output everything into the next columns automatically

It's perfect for anyone who wants quick, AI-powered overviews of online content — students, journalists, analysts, or content creators.

⚙️ How It Works

The magic happens inside **Google Apps Script**, which acts as a bridge between Google Sheets and Gemini's REST API.

Step 1 — Fetch the Web Page

Each URL is retrieved with `UrlFetchApp.fetch()`.
Scripts, styles, and HTML tags are stripped away, leaving clean readable text.

```
body = body
```

```
  .replace(/<script[\s\S]*?<\/script>/gi, ' ')
  .replace(/<style[\s\S]*?<\/style>/gi, ' ')
  .replace(/<[^>]+>/g, ' ')
  .trim();
```

The result is cached for six hours to avoid hitting rate limits.

Step 2 — Chunk and Summarize

Long pages are split into ~6000-character chunks.
Each chunk is sent to Gemini with a simple instruction:

> "Summarize this web page chunk in 4–6 concise bullet points focusing on facts."

This ensures we don't exceed token limits and still get accurate partial summaries.

Step 3 — Merge and Structure

Once all chunk summaries return, we send them again to Gemini — this time asking for a **strict JSON object**:

```
{
  "title": "string",
  "summary": ["string"],
  "key_quotes": ["string"],
  "entities": {
    "people": ["string"],
    "organizations": ["string"],
    "locations": ["string"]
  },
  "topics": ["string"],
  "reading_time_minutes": 3,
  "source_url": "string"
}
```

The AI consolidates everything into a clean data structure ready for Sheets.

Step 4 — Display Results in Sheets

Each row of URLs expands into seven columns:

Title	Summary (bullets)	Entities (People | Orgs | Locs)	Key Quotes	Topics	Reading Minutes	Source URL
AI Updates from Google	• Gemini 2.5 launch …	Sundar Pichai	Google	“AI for everyone”	ai, research	6	https://blog.google/...

🧩 Code Highlights

- **`fetchUrlText(url)`** — cleans HTML into readable text
- **`summarizeUrl(url)`** — handles chunking, API calls, and JSON consolidation
- **`AI_URL_SUMMARY(url)`** — custom sheet function returning the full JSON
- **`summarizeUrlSelectionToRight()`** — bulk action for selected URLs
- **Sidebar UI** — allows quick one-off summarization inside a small HTML panel

---

🧪 Try It Yourself

1. Open a new Google Sheet → **Extensions** → **Apps Script**
2. Create these files:
    - `Code.gs`
    - `Sidebar.html`
    - `appsscript.json`

3. Paste in the code from the Exercise 7 GitHub package
4. In **Script Properties**, add your Gemini API Key (`GEMINI_API_KEY`).
5. Run `onOpen()` once and reload the Sheet.

Then, use the menu:

> 🧠 **AI Tools → Insert Sample URLs**
> Select column A → **Summarize URL Selection → next columns**

or try a single-cell formula:

```
=AI_URL_SUMMARY(A2)
```

🧩 Tips & Troubleshooting

- If you see ❌ `Set GEMINI_API_KEY…` → add your key in Script Properties.
- Some news sites block fetch requests — try public blogs first.
- Long pages? The script summarizes the first four chunks (≈ 24 K chars).
- Need finer control? Adjust `CHUNK_CHAR_LIMIT` or `MERGE_TOKENS`.

🎯 Why This Matters

This exercise demonstrates **multi-stage AI orchestration** inside a low-code environment. You're combining:

1. Data fetching (UrlFetchApp)
2. Pre-processing and caching
3. Multi-prompt reasoning (chunk → merge)

4. Structured JSON outputs
5. Instant visualization in Sheets

In short — a real-world example of **AI-assisted knowledge extraction** without any external server.

Code.gs

```
/**
* Exercise 7 – Web Article Summarizer & Citation Extractor (Sheets + Gemini REST)
* (Patched to avoid MAX_TOKENS truncation and add resilient merge w/ retry)
*
* What this file gives you:
* - Menu items (onOpen): Open Sidebar, Summarize URL Selection → next columns, Insert Sample URLs
* - Custom function: =AI_URL_SUMMARY(A2)
* - Robust pipeline:
*     fetchUrlText → chunk (short) → per-chunk 3-bullet summaries → merge w/ big token budget
*     → retry with minimal JSON if needed → normalized result to sheet
*
* Tip: Store your API key in Script Properties as GEMINI_API_KEY. (Editor → Project Settings → Script properties)
*/

// ===== CONFIG =====
const GEMINI_API_KEY = '';
const GEMINI_MODEL   = 'gemini-2.5-flash';
const BASE_URL       = 'https://generativelanguage.googleapis.com/v1beta';

const DEFAULT_TEMP     = 0.2;           // low creativity for factual accuracy
```

```
const DEFAULT_TOKENS   = 640;            // ↓ slightly smaller per-chunk
budget
const MERGE_TOKENS     = 1536;           // ↑ much larger budget for the merge
step
const COOLDOWN_MS      = 150;

const CHUNK_CHAR_LIMIT = 5500;           // ↓ smaller chunks → less to merge
overall
const CHUNK_JOINER     = '\n\n';

const CACHE_SEC        = 6 * 3600;       // 6 hours for fetched pages cache

// ===== MENU & UI =====
function onOpen() {
 SpreadsheetApp.getUi()
   .createMenu('🧠 AI Tools')
   .addItem('Open Web Summarizer Sidebar', 'showWebSummarizerSidebar')
   .addItem('Summarize URL Selection → next columns',
'summarizeUrlSelectionToRight')
   .addSeparator()
   .addItem('Insert Sample URLs', 'insertSampleUrls')
   .addToUi();
}

function showWebSummarizerSidebar() {
 const html = HtmlService.createHtmlOutputFromFile('Sidebar')
   .setTitle('Gemini Web Summarizer');
 SpreadsheetApp.getUi().showSidebar(html);
}

// ===== API KEY =====
function getApiKey() {
```

```
 const p =
PropertiesService.getScriptProperties().getProperty('GEMINI_API_KEY');
 return (p && p.trim()) ? p.trim() : (GEMINI_API_KEY || '').trim();
}

// ===== GEMINI HELPERS =====
function extractGeminiText(json) {
 try {
   if (json?.error?.message) return `❌ API error: ${json.error.message}`;

   const block = json?.promptFeedback?.blockReason;
   if (block) {
     const details = (json?.promptFeedback?.safetyRatings || [])
       .map(r => `${r.category}:${r.probability}`).join(', ');
     return `❌ Blocked by safety (${block}${details ? ' — ' + details :
''}).`;
   }

   const cand = json?.candidates?.[0];
   if (!cand) return '';

   if (cand.finishReason === 'MAX_TOKENS') {
     const parts = cand?.content?.parts || [];
     const t = parts.map(p => p?.text || '').join('').trim();
     return t ? `${t}\n\n⚠️ Truncated (MAX_TOKENS).` : '⚠️ Truncated
(MAX_TOKENS) and no visible text.';
   }

   const parts = cand?.content?.parts || [];
   const text = parts.map(p => p?.text || '').join('').trim();
   if (text) return text;

   const cText = cand?.text;
```

```
   if (cText && String(cText).trim()) return String(cText).trim();

   return '';
 } catch (e) { return ''; }
}

function safeParseJson(maybeJson) {
 try {
   if (!maybeJson) return null;
   return JSON.parse(maybeJson);
 } catch (e) {
   // Salvage the first {...} block if extra prose leaks in
   const m = String(maybeJson).match(/\{[\s\S]*\}/);
   if (!m) return null;
   try { return JSON.parse(m[0]); } catch (e2) { return null; }
 }
}

function geminiGenerateText(prompt, maxTokens) {
 const apiKey = getApiKey();
 if (!apiKey) return '❌ Set GEMINI_API_KEY in Script Properties or
Code.gs.';

 const payload = {
   contents: [{ role: "user", parts: [{ text: prompt }] }],
   generationConfig: {
     temperature: DEFAULT_TEMP,
     maxOutputTokens: maxTokens || DEFAULT_TOKENS,
     topP: 0.95,
     topK: 40
   }
 };
```

```
 const url =
`${BASE_URL}/models/${encodeURIComponent(GEMINI_MODEL)}:generateContent?key=${
encodeURIComponent(apiKey)}`;
 const options = { method: 'post', contentType: 'application/json', payload:
JSON.stringify(payload), muteHttpExceptions: true };

 try {
   const res  = UrlFetchApp.fetch(url, options);
   const code = res.getResponseCode();
   const body = res.getContentText();

   if (code < 200 || code >= 300) return `❌ Error: Gemini HTTP ${code}.
${body}`;

   const text = extractGeminiText(JSON.parse(body));
   return text || '⚠️ No output (see Logs).';
 } catch (e) {
   return '❌ Error: ' + e.message;
 }
}

// ===== FETCH & CLEAN HTML =====
function fetchUrlText(url) {
 const cache = CacheService.getScriptCache();
 const key = 'fetch:' + url;
 const cached = cache.get(key);
 if (cached) return cached;

 let res, code, body;
 try {
   res = UrlFetchApp.fetch(url, {
     method: 'get',
     muteHttpExceptions: true,
```

```
    followRedirects: true,
    validateHttpsCertificates: true,
    headers: { 'User-Agent': 'AppsScript-GeminiSummarizer/1.0' }
  });
  code = res.getResponseCode();
  body = res.getContentText();
} catch (e) {
  return '';
}

if (code < 200 || code >= 400) return '';

// Remove scripts/styles/noscript/comments
body = body.replace(/<script[\s\S]*?<\/script>/gi, ' ')
           .replace(/<style[\s\S]*?<\/style>/gi, ' ')
           .replace(/<noscript[\s\S]*?<\/noscript>/gi, ' ')
           .replace(/<!--[\s\S]*?-->/g, ' ');

// Attempt to capture title
const titleMatch = body.match(/<title[^>]*>([\s\S]*?)<\/title>/i);
const t = titleMatch ? 'TITLE: ' + titleMatch[1].trim() + '\n\n' : '';

// Strip remaining tags and normalize whitespace
const text = t + body.replace(/<[^>]+>/g, ' ')
                     .replace(/\s+/g, ' ')
                     .replace(/ /g, ' ')
                     .trim();

cache.put(key, text, CACHE_SEC);
return text;
}

// ===== CHUNKING =====
```

```
function splitTextIntoChunks(input, limit = CHUNK_CHAR_LIMIT) {
 const text = String(input || '');
 if (text.length <= limit) return [text];

 const parts = [];
 for (let i = 0; i < text.length; i += limit) {
   parts.push(text.slice(i, i + limit));
 }
 return parts;
}

// ===== MERGE HELPERS (PATCHED) =====
function looksTruncated_(text) {
 return /Truncated \(MAX_TOKENS\)/i.test(String(text || ''));
}

/**
* Try to merge chunk bullet summaries into strict JSON.
* 1st attempt: full schema (title, summary, key_quotes, entities, topics,
reading_time_minutes, source_url)
* Retry (if truncated/invalid): minimal schema (title, summary, topics,
reading_time_minutes, source_url)
*/
function mergeChunkSummaries_(chunkSummaries, url) {
 // Attempt 1 — full schema
 const mergePrompt = [
   'Consolidate these bullet summaries from one web page into STRICT JSON:',
   '{ "title": string, "summary": string[], "key_quotes": string[],
"entities": { "people": string[], "organizations": string[], "locations":
string[] }, "topics": string[], "reading_time_minutes": number, "source_url":
string }',
   'Rules: 3–7 bullets; up to 3 short verbatim quotes; 3–6 lowercase topics;
integer minutes.',
```

```
  `Set "source_url" to: ${url}`,
  '',
  'Chunk bullets:',
  chunkSummaries.join('\n\n')
].join('\n');

let jsonText = geminiGenerateText(mergePrompt, MERGE_TOKENS);
let obj = safeParseJson(jsonText);
if (obj && !looksTruncated_(jsonText)) return obj;

// Attempt 2 — minimal schema (shorter to avoid truncation)
const retryPrompt = [
  'Return STRICT JSON with keys exactly:',
  '{ "title": string, "summary": string[], "topics": string[],
"reading_time_minutes": number, "source_url": string }',
  'Rules: summary 3–5 bullets, topics 3–5 lowercase tags, integer minutes.',
  `Set "source_url" to: ${url}`,
  '',
  'Chunk bullets:',
  chunkSummaries.join('\n\n')
].join('\n');

jsonText = geminiGenerateText(retryPrompt, MERGE_TOKENS);
obj = safeParseJson(jsonText);
if (!obj) return null;

// Normalize to include entities & key_quotes fields even if minimal response
obj.summary = Array.isArray(obj.summary) ? obj.summary : [];
obj.topics  = Array.isArray(obj.topics)  ? obj.topics  : [];
if (!obj.entities) obj.entities = { people: [], organizations: [], locations:
[] };
if (!Array.isArray(obj.entities.people))        obj.entities.people = [];
```

```
 if (!Array.isArray(obj.entities.organizations)) obj.entities.organizations =
[];
 if (!Array.isArray(obj.entities.locations))     obj.entities.locations = [];
 if (!Array.isArray(obj.key_quotes))             obj.key_quotes = [];

 return obj;
}

// ===== SUMMARIZATION PIPELINE =====
function summarizeUrl(url) {
 const raw = fetchUrlText(url);
 if (!raw) return { error: '❌ Could not fetch URL or empty content.',
source_url: url };

 // 1) Per-chunk summarization — ask for *short* bullets to reduce merge size
 const chunks = splitTextIntoChunks(raw);
 const chunkSummaries = [];
 for (let i = 0; i < chunks.length; i++) {
   const p = [
     'Summarize this chunk in 3 concise fact-only bullets. No preface; bullets
only.',
     '',
     chunks[i]
   ].join('\n');

   const out = geminiGenerateText(p, DEFAULT_TOKENS);
   chunkSummaries.push(out);
   Utilities.sleep(COOLDOWN_MS);

   // keep bounded—summarize first 4 chunks max
   if (i >= 3) break;
 }
```

```
// 2) Merge all bullet summaries into structured JSON (with retry)
const obj = mergeChunkSummaries_(chunkSummaries, url);
if (!obj) {
  return {
    error: '⚠️ Could not parse consolidated JSON after retry.',
    raw: '(truncated or invalid)',
    source_url: url
  };
}

// Normalize fields
const title = String(obj.title || '').trim();
const summary = Array.isArray(obj.summary) ? obj.summary : [];
const key_quotes = Array.isArray(obj.key_quotes) ? obj.key_quotes : [];
const entities = obj.entities || {};
const people = Array.isArray(entities.people) ? entities.people : [];
const orgs   = Array.isArray(entities.organizations) ? entities.organizations
: [];
const locs   = Array.isArray(entities.locations) ? entities.locations : [];
const topics = Array.isArray(obj.topics) ? obj.topics : [];

// Better minutes fallback based on words in bullet summaries (~200 wpm)
const minutes = Number(obj.reading_time_minutes) ||
                Math.max(1, Math.round((chunkSummaries.join('
').split(/\s+/).length) / 200));

return {
  title,
  summary,
  key_quotes,
  entities: { people, organizations: orgs, locations: locs },
  topics,
  reading_time_minutes: minutes,
```

```
   source_url: url
 };
}

// ===== CUSTOM FUNCTION =====
// Use in a cell: =AI_URL_SUMMARY(A2)
function AI_URL_SUMMARY(url) {
 const r = summarizeUrl(String(url || ''));
 return JSON.stringify(r);
}

// ===== SHEETS ACTION =====
function summarizeUrlSelectionToRight() {
 const range = SpreadsheetApp.getActiveRange();
 if (!range) { SpreadsheetApp.getUi().alert('Please select one or more URL
cells first.'); return; }

 const values = range.getValues();
 const out = []; // [Title, SummaryJoined, Entities, Quotes, Topics, Minutes,
Source]

 for (let i = 0; i < values.length; i++) {
   const url = String(values[i][0] || '').trim();
   if (!/^https?:\/\//i.test(url)) {
     out.push(['Invalid URL', '', '', '', '', '', url]);
     continue;
   }

   const r = summarizeUrl(url);
   if (r.error) {
     out.push([r.error, '', '', '', '', '', url]);
   } else {
     const entitiesJoined = [
```

```
      (r.entities.people || []).join(', '),
      (r.entities.organizations || []).join(', '),
      (r.entities.locations || []).join(', ')
    ].filter(Boolean).join(' | ');

    out.push([
      r.title || '(untitled)',
      (r.summary || []).join(' • '),
      entitiesJoined,
      (r.key_quotes || []).map(q => `“${q}”`).join(' '),
      (r.topics || []).join(', '),
      r.reading_time_minutes || '',
      r.source_url || url
    ]);
  }

  Utilities.sleep(COOLDOWN_MS);
}

const sheet = range.getSheet();
const startRow = range.getRow();
const nextCol  = range.getColumn() + range.getNumColumns();
sheet.getRange(startRow, nextCol, out.length, 7).setValues(out);

if (startRow === 1) {
  sheet.getRange(1, nextCol, 1, 7).setValues([[
    'Title', 'Summary (bullets)', 'Entities (people | orgs | locs)', 'Key
Quotes', 'Topics', 'Reading Minutes', 'Source URL'
  ]]);
}

SpreadsheetApp.getActive().toast(`Summarized ${out.length} URL(s) →
${colToLetter(nextCol)}:${colToLetter(nextCol+6)}`);
```

```
}

// ===== SAMPLE DATA =====
function insertSampleUrls() {
 const sheet = SpreadsheetApp.getActiveSheet();
 sheet.clearContents();
 sheet.getRange(1, 1, 6, 2).setValues([
   ['URL', 'Output →'],
   ['https://blog.google/technology/ai/ai-announcements/', ''],
   ['https://workspace.google.com/blog/product-announcements', ''],
   ['https://developers.google.com/apps-script', ''],
   ['https://ai.google.dev/gemini-api', ''],
   ['https://aistudio.google.com', '']
 ]);
 SpreadsheetApp.getActive().toast('✅ Sample URLs inserted (A–B).');
}

// ===== UTIL =====
function colToLetter(col) {
 let temp = '', letter = '';
 while (col > 0) {
   temp = (col - 1) % 26;
   letter = String.fromCharCode(temp + 65) + letter;
   col = (col - temp - 1) / 26;
 }
 return letter;
}
```

Sidebar.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <style>
      body { font-family: system-ui, -apple-system, Segoe UI, Roboto,
Arial, sans-serif; padding: 16px; }
      input { width: 100%; padding: 8px; margin-bottom: 8px; }
      button { background: #1a73e8; color: #fff; border: 0; padding:
8px 12px; border-radius: 6px; cursor: pointer; }
      pre { background: #f1f3f4; padding: 10px; border-radius: 6px;
white-space: pre-wrap; }
      .small { color:#555; font-size:12px; }
      code { background:#eef; padding:2px 4px; border-radius:4px; }
    </style>
  </head>
  <body>
    <h2>🧠 Web Summarizer</h2>
    <input id="url" placeholder="https://example.com/article" />
    <button id="run" onclick="run()">Summarize</button>

    <h3>Result (JSON)</h3>
    <pre id="out">Waiting…</pre>
    <p class="small">Tip: Put URLs in column A and use the menu:
<b>Summarize URL Selection → next columns</b>,
      or try the custom function <code>=AI_URL_SUMMARY(A2)</code>.</p>

    <script>
      function run(){
        const btn = document.getElementById('run');
        const url = document.getElementById('url').value || '';
        document.getElementById('out').textContent = '⏳ Fetching &
summarizing…';
        btn.disabled = true;

        google.script.run
          .withSuccessHandler(r => {
```

```
            document.getElementById('out').textContent =
JSON.stringify(r, null, 2);
            btn.disabled = false;
          })
          .withFailureHandler(e => {
            document.getElementById('out').textContent = 'Error: ' +
e.message;
            btn.disabled = false;
          })
          .summarizeUrl(url);
      }
    </script>
  </body>
</html>
```

---

appsscript.json

```
{
  "timeZone": "America/Toronto",
  "exceptionLogging": "STACKDRIVER",
  "runtimeVersion": "V8",
  "oauthScopes": [
    "https://www.googleapis.com/auth/spreadsheets",
    "https://www.googleapis.com/auth/script.container.ui",
    "https://www.googleapis.com/auth/script.external_request"
  ]
}
```

---

2) Add your Gemini API key

- Apps Script editor → **Project Settings** (gear icon) → **Script properties** → **Add property**
    - **Name**: `GEMINI_API_KEY`

- **Value**: your API key from https://aistudio.google.com/app/apikey

(You can hardcode it in `Code.gs`, but Script Properties is safer.)

---

3) Authorize once

- In the editor, run **onOpen()** → allow the requested scopes.
- Back in the Sheet, reload. You'll see the 🧠 **AI Tools** menu.

---

4) Test it quickly

Option A — **Use sample data**

- Menu → 🧠 **AI Tools** → **Insert Sample URLs**
- Select the URLs in column A
- Menu → 🧠 **AI Tools** → **Summarize URL Selection** → **next columns**

Option B — **Try the sidebar**

- Menu → 🧠 **AI Tools** → **Open Web Summarizer Sidebar**
- Paste a URL → **Summarize**

Option C — **Custom function**

- In B2:
  `=AI_URL_SUMMARY(A2)`