

```

import Anthropic from "@anthropic-ai/sdk";
import html2canvas from "html2canvas";

// Never ship a real API key to the browser. Point baseURL at your own
// backend proxy that adds the key and forwards to api.anthropic.com.
const client = new Anthropic({
  apiKey: "proxy-handles-this",
  baseURL: "/api/claude-proxy",
  dangerouslyAllowBrowser: true,
  maxRetries: 2,
  timeout: 60_000,
});

// ---- Client-side tools (these run in the user's browser) ----

function compactDOM(el: Element, depth = 0, maxDepth = 6): string {
  if (depth > maxDepth) return "";
  const id = el.id ? `#${el.id}` : "";
  const cls = el.classList.length ? `.${el.classList.slice(0, 2).join(".")}` : ``
  const ownText = [...el.childNodes]
    .filter((n) => n.nodeType === Node.TEXT_NODE)
    .map((n) => n.textContent?.trim())
    .filter(Boolean)
    .join(" ")
    .slice(0, 80);
  const kids = [...el.children]
    .map((c) => compactDOM(c, depth + 1, maxDepth))
    .filter(Boolean)
    .join("");
  return `${"  ".repeat(depth)}<${el.tagName.toLowerCase()}${id}${cls}>${ownText}
  `
}

async function screenshot(el: HTMLElement): Promise<string> {
  const canvas = await html2canvas(el);
  return canvas.toDataURL("image/png").split(",")[1]; // base64 only
}

const tools: Anthropic.Tool[] = [
  {
    name: "compact_dom",
    description: "Get a compact text tree of the user's selected element.",
    input_schema: {
      type: "object",
      properties: { max_depth: { type: "number" } },
    },
  },

```

```

    },
  },
  {
    name: "screenshot",
    description: "Take a screenshot of the user's selected element.",
    input_schema: { type: "object", properties: {} },
  },
  // Optional server-side sandbox, no extra params needed:
  // { type: "code_execution_20250825", name: "code_execution" },
];

// ---- Agent loop ----

export async function runAgent(prompt: string, selected: HTMLElement) {
  const messages: Anthropic.MessageParam[] = [{ role: "user", content: prompt }];

  for (let turn = 0; turn < 10; turn++) {
    const res = await client.messages.create({
      model: "claude-sonnet-5-5",
      max_tokens: 1000,
      tools,
      messages,
    });

    messages.push({ role: "assistant", content: res.content });
    if (res.stop_reason !== "tool_use") return res; // done

    const results: Anthropic.ToolResultBlockParam[] = [];
    for (const block of res.content) {
      if (block.type !== "tool_use") continue;
      try {
        let content: Anthropic.ToolResultBlockParam["content"];
        if (block.name === "compact_dom") {
          const depth = (block.input as any).max_depth ?? 6;
          content = compactDOM(selected, 0, depth);
        } else if (block.name === "screenshot") {
          content = [
            {
              type: "image",
              source: { type: "base64", media_type: "image/png", data: await scre
            },
          ];
        } else {
          content = `Unknown tool: ${block.name}`;
        }
        results.push({ type: "tool_result", tool_use_id: block.id, content });
      } catch (e) {

```

```
        results.push({
            type: "tool_result",
            tool_use_id: block.id,
            content: String(e),
            is_error: true,
        });
    }
}
messages.push({ role: "user", content: results });
}
throw new Error("Agent hit turn limit");
}
```