WebView Agent WebTools：单工具 vs 多工具，以及是否需要 `open`（Deep Research）

Executive Summary

在 OpenAI/LLM 的工具调用（tool/function calling）里，“把一堆能力拆成多个小工具”通常更可靠、更可控；但工具数量也不宜无限增加。OpenAI 的函数调用指南明确建议把函数数量保持较低（经验值：少于约 20 个），同时保持每个工具职责清晰、参数简单。基于这些最佳实践，你们当前 15 个左右的 `web_*` 工具规模并不“离谱”，反而处在推荐区间；但为了让 Agent 能主动跳转页面，补一个 `web_open(url)`（以及可选的 back/forward/reload）会更符合浏览器自动化直觉与可用性。[1][2]

Key Findings

- **多工具并不一定“坏”**：OpenAI 指南建议把函数数量保持低（例如 <20）以提高可靠性；你们当前工具数量大致在这个范围内，属于合理规模。[1]
- **更推荐“一个工具一个职责”**：OpenAI Agents SDK 文档强调工具应当小而可组合（one responsibility per tool），有利于模型推理、调试和安全控制。[2]
- **“一个工具 + command 字符串”通常最不推荐**：它会把可验证的 JSON Schema 退化成不可控文本协议，难以做参数校验、最小权限控制与结构化输出约束，容易引发注入/跑偏与不可复现实验结果（需要你自己再实现一层解析与沙箱）。[2][4]
- **“一个工具 + action 枚举 + 参数”是折中方案**：可以减少 tool 数量与上下文 token，但会引入条件必填（conditional required）与更复杂的 schema；如果没有启用严格 schema 约束（structured outputs/strict），模型更容易生成无效组合（例如 action=fill 但缺 value）。[3][5]
- **`open/navigate` 在 Web 自动化里通常是基础能力**：如果 Agent 需要从任务描述直接跳转 URL（而不是靠点击现有 link），应提供显式导航工具；否则 Agent 只能在当前页面内“走路”。[1][2]

Detailed Analysis

1) 多工具 vs 单工具：可靠性、可控性与 token 成本

多工具（例如 `web_snapshot` / `web_click` / `web_fill` ...）的优点

1. **更强的可验证性**：每个工具都有单独 schema，参数必填/枚举约束更清晰，容易在 executor 层做严格校验并返回对模型友好的结构化错误。[2]
2. **更容易最小权限**：可以在不同场景只允许一部分工具（例如只读：snapshot/query；可写：click/fill）。这比“一个万能工具”更容易实现最小权限与审计。[2]
3. **更利于评测与回归**：单个工具失败的原因更明确，便于写测试与评估（e.g. overlay/scroll/query kind）。[2]

多工具的代价

- **上下文 token 成本**：函数/工具定义（name/description/parameters）会占用 prompt tokens；当工具很多时，会挤占上下文并影响效果。Microsoft 的函数调用指南也提醒描述/定义会消耗 token，过多时应减少或做聚合/检索。[4]
- **工具选择难度随数量上升**：OpenAI 函数调用指南建议保持函数数量较低以提高准确率（常见经验值是 <20）。[1]

结论（对你们当前规模）

你们的 web 工具目前大约 15 个，仍处在 OpenAI “尽量少、例如 <20” 这个经验范围内；因此“工具太多导致不对劲”的主要风险不是数量本身，而是：

- schema 和 executor 口径漂移（你们 v17 正在做的 schema gate 正对症）；以及
- 工具描述/提示词是否会诱导模型做错误策略（例如频繁 outerHTML、忘记 ref 短生命周期）。[1][2]

2) 单工具的两种变体：`action` 枚举 vs `command` 字符串

2.1 单工具：`web(action, ...)`（结构化折中）

这种方案类似 OpenAI “computer use”一类的动作路由思路：只给一个工具名，但用 `action`（enum）区分 click/fill/scroll 等，再用参数承载 action 所需字段。

优点：

- **工具数量更少**：减少工具定义 token，降低“选择错工具”的空间。[1][4]

代价与风险：

- **schema 复杂度提高**：要表达“action=fill 时必须有 value；action=click 时必须有 ref”等条件必填，往往需要 `oneOf`/分支 schema。若没有强约束（strict/structured outputs），模型更容易生成无效组合或多余字段。[3][5]
- **debug 成本**：所有失败都集中在一个入口，要靠错误码/日志再分流，评测颗粒度更粗。[2]

建议：

- 若你们要走这个方向，建议把 schema 做成**分支严格**（`oneOf` + `additionalProperties: false`），并尽可能启用 structured outputs/strict（同时禁用 parallel tool calls）以提高参数合规率。[3][5]

2.2 单工具：`web(command: "click ...")`（不结构化，通常最不推荐）

把 `agent-browser` CLI 的“文本命令协议”直接暴露给模型，表面上只有一个工具，但实际上把所有校验与安全边界都推给 executor 去解析文本。

主要问题：

- **失去 JSON Schema 的强约束价值**：模型输出是自由文本，参数类型/必填无法在工具层校验与约束；structured outputs 也很难发挥作用。[3][5]
- **安全与可控性更差**：命令字符串更容易被 prompt 注入、拼接多操作、或绕过你预期的限制；你最终不得不实现一套“解析器 + 白名单 + 沙箱 + 逐条审计”。[2][4]

因此，这种方案一般只建议用于“人类 CLI”，不建议作为 LLM 工具调用的主接口。

3) “没有 `open`”到底离谱不离谱：取决于产品定位

你说得对：如果对外宣称“浏览器自动化工具”，没有 `open/navigate` 会让 Agent 能力出现明显断层——Agent 只能在当前页面里点来点去，无法从任务描述直接跳到 URL（除非你在系统侧先 load URL）。这也会导致你无法覆盖很多典型用例（例如“打开某个站点后登录/搜索”）。[1][2]

但从你们目前的离线 E2E 设计看：

- instrumentation 预先 `loadUrlAndWait(file:///android_asset/...)`，所以 agent-run E2E “起跑时”不需要 open；
- 但这只是测试 harness 的便利，不应该限制库/工具对外能力。

结论：建议补一个 `web_open(url)`（和可选的 `web_back/web_forward/web_reload`），并让 `web_snapshot` 返回 `url/title` 作为轻量的“当前页面状态”。[1][2]

4) 给你们仓库的可执行建议（结合 v17）

结合你们正在做的 v17 “schema 单一来源 + 门禁测试”，更推荐这样收敛：

1. **保留多工具形态**，把数量控制在 ~20 以内（你们当前规模 OK）。[1][2]
2. **补齐导航最小集**：

- `web_open(url)`：明确支持 `file:///android_asset/...`（离线）与 `https?://...`（在线可选）；返回 `{ok,url,title}`。
- （可选）`web_back` / `web_forward` / `web_reload`：如果你们 PRD/测试清单确实要对齐浏览器语义。

3. **用 schema gate 防漂移**：`docs/tools/web-tools.openai.json` 只作为“对外契约展示”，代码侧 schema 为单一来源，测试负责对齐（忽略 key 顺序）。[1][2]
4. **如果未来工具数暴涨**：优先考虑“按场景裁剪 allowed tools / tool_choice”而不是退化为 command 字符串协议。[1][2][4]

Areas of Consensus

- 工具 schema/描述对模型表现影响很大；工具越多越需要规范化命名、清晰职责和一致的参数风格。[1][2][4]
- 与其把能力塞进一个自由文本命令，不如保持结构化 schema，让系统能校验、拒绝、审计与回放。[2][3][4]
- 工具数量要控制：少量高质量工具通常比大量相似工具更可靠。[1][4]

Areas of Debate

- **“action 枚举单工具”是否优于多工具**：在工具数量较多且 token 紧张时，单工具可能有优势；但在没有 strict/structured outputs 或 SDK 对复杂 schema 支持不佳时，多工具往往更稳。[1][2][3][5]
- **并行工具调用（parallel tool calls）**：并行能加速，但会降低 schema 严格匹配的可靠性；OpenAI Structured Outputs 也明确指出 strict 与并行工具调用不兼容，需要按场景取舍。[3]

Sources

[1] OpenAI, “Function calling guide” (platform docs). Credibility: official documentation. https://platform.openai.com/docs/guides/function-calling
[2] OpenAI, “Agents SDK — Tools” (tools are small & composable; one responsibility per tool). Credibility: official documentation. https://openai.github.io/openai-agents-python/tools/
[3] OpenAI, “Introducing Structured Outputs in the API” (strict schema via tools; strict not compatible with parallel function calls). Credibility: official announcement + guidance. https://openai.com/index/introducing-structured-outputs-in-the-api/
[4] Microsoft Learn, “Function calling with OpenAI API” (tool definitions consume tokens; minimize/reduce as needed). Credibility: major vendor documentation. https://learn.microsoft.com/en-us/dotnet/ai/conceptual/understanding-openai-functions

[5] OpenAI, “Structured outputs — supported schemas” (JSON mode vs Structured Outputs; guidance on schema/validation). Credibility: official documentation. https://platform.openai.com/docs/guides/structured-outputs/supported-schemas

Gaps and Further Research

- 你们当前 OpenAgentic SDK/provider 是否支持把 `strict: true` 透传到 Responses/Tools，以及是否支持 `parallel_tool_calls=false` 这类开关；若不支持，单工具的复杂分支 schema 风险会更高，需要更谨慎地选择“多工具”路线。[3][5]
- WebView 的导航等待策略（onPageFinished vs networkidle 等）在不同 ROM/厂商上可能不一致；`web_open` 的 DoD 应包含稳定等待条件与可回放证据（截图/录屏）。[2][4]