# Android WebView 中 `Element.computedRole` / `Element.computedName` 可用性调研（Deep Research）

## Executive Summary

综合规范、Chrome Platform Status 与 Chromium 源码证据来看：`Element.computedRole` / `Element.computedName` 仍属于 AOM（Accessibility Object Model）相关的试验性能力，Chromium 侧目前以 `ComputedAccessibilityInfo` 运行时特性门控，默认并不会在稳定版/Android WebView 里对网页脚本“可靠开放”。因此，用它们在 Android WebView 中构建生产级 snapshot 的结论是“**默认不能用 / 不可靠**”，应按“**必须有降级方案**”来设计。

## Key Findings

- **AOM 方向在 Chrome 侧已不再推进**：Chrome Platform Status 的 “Accessible Object Model (AOM)” 记录显示 Chrome 状态为 “No longer pursuing”。[1]
- **`computedRole` / `computedName` 在 Chromium 中仍被特性开关门控**：Blink 的 `Element.idl` 把它们标注为 `RuntimeEnabled=ComputedAccessibilityInfo`；该特性在 `runtime_enabled_features.json5` 中为 `status: "experimental"`，意味着默认不可依赖。 [2][3]
- **Chrome 相关实践明确提示需要启用特定 flag**：Chrome DevRel 的 “Puppetaria” 项目说明里提到 `computedName` / `computedRole` 需要 `ComputedAccessibilityInfo` “launch flag”。[4]
- **“正确做法”更多走自动化/调试协议，而非网页 JS API**：W3C WebDriver 标准提供 “Get Computed Role/Label” 命令；Playwright 在 Chromium 端的可访问性快照通过 CDP 调用 `Accessibility.getFullAXTree` 来拿 AX Tree。 [5][6]
- **生产可行的降级路径是“在 JS 里实现（或复用库实现）可访问名称与角色推导”**：可参考 W3C AccName（Accessible Name and Description）算法与 ARIA in HTML 的隐式角色映射；社区里有 `dom-accessibility-api` 等实现。 [7][8][9]

# Detailed Analysis

## 1) API/规范状态：它们到底“属于哪个规范”，是否有被替换风险？

`computedRole` / `computedName` 在行业语境里常被归入 AOM（Accessibility Object Model）方向的“向网页暴露计算后的无障碍信息”的尝试。Chrome Platform Status 对 AOM 的条目明确标注 Chrome 状态为“No longer pursuing”，这对“长期稳定可用”的风险判断非常关键：即便 Chromium 代码里仍保留了相关接口，Chrome 也未将 AOM 作为要稳定推进并面向 Web 开发者长期维护的方向。[1]

与此同时，WICG 的 AOM 说明文档把这类能力描述为阶段性探索（修改/探索可访问性树），整体处于社区草案/探索性质而非成熟标准。[10]

**结论（规范层面）**：把 `computedRole` / `computedName` 当作“标准化、长期稳定的 Web 平台能力”风险高；更像“实验性/工具化接口”。[1][10]

## 2) Chromium 实现状态：是否默认开启？是否需要 feature flag？

从 Chromium 源码可直接看到两点：

1. `Element.computedName` / `Element.computedRole` 在 `Element.idl` 中存在，但被标注为 `RuntimeEnabled=ComputedAccessibilityInfo`。[2]
2. `ComputedAccessibilityInfo` 在 Blink runtime 特性清单 `runtime_enabled_features.json5` 中标记为 `status: "experimental"`。[3]

根据 Chromium runtime feature 的文档，实验性特性通常需要通过“启用实验性 Web 平台特性”等开关/命令行方式启用，而不应假设在稳定环境默认可用。[11]

此外，Chrome DevRel 的 Puppetaria 项目在介绍中也明确：`computedName` / `computedRole` 受 `ComputedAccessibilityInfo` launch flag 控制。[4]

**结论（Chromium 层面）**：至少到当前源码状态，这两个属性并非“默认稳定开放”。在 Android WebView 场景中，应用很难（或不应该）在生产环境要求用户/系统打开实验性 blink 特性，因此“可靠可用”的概率极低。[2][3][4][11]

## 3) Android WebView 的对应关系：Chromium 版本、Android 版本、更新机制

Android WebView 是可替换 provider 的体系：设备可能使用不同的 WebView provider（如 `com.google.android.webview`、`com.android.webview`，或 OEM 方案），并且在部分系统上可通过“开发者选项/设置”选择 provider；Chromium 的 WebView 文档对 provider 选择与切换有专门说明。[12]

对“版本 ↔ 系统”的关系，需要特别强调两点：

- **Android 7+ 大多数设备上 WebView 可通过应用商店独立更新**（Google Play 体系设备尤甚），因此“Android 系统大版本 ≠ WebView/Chromium 固定版本”。[12][13]
- **中国大陆环境下 Google Play 生态可能缺失**，导致 WebView/Chrome 的更新链路与全球版本节奏不同（依赖 OEM/第三方分发与系统更新）。这会显著增加“实际 WebView 版本分布”的不确定性。[14]

因此，若你的目标是“最低版本门槛”，最佳实践不是按 OS 推断，而是**运行时获取 WebView 版本**（例如通过 Android API 查询当前 WebView provider 包信息），并在服务端/埋点里统计真实分布后再做门槛决策。[12]

**但注意**：即便 WebView 的 Chromium 版本足够新，只要 `ComputedAccessibilityInfo` 默认不开放，`computedRole` / `computedName` 依然可能 `undefined`/空值；“版本新”并不等同于“属性可用”。[2][3][11]

# 4) 兼容性与实测信号：MDN / caniuse / 社区报告

- **MDN**：`https://developer.mozilla.org/en-US/docs/Web/API/Element/computedRole` 与 `.../computedName` 当前返回 404（页面不存在），意味着它们并非 MDN 认可的稳定 Web API 文档条目，也就没有标准的 BCD（browser-compat-data）表可引用做覆盖率判断。[15]
- **caniuse**：未检索到对应独立特性条目（以站内检索/索引结果为准），因此难以通过 caniuse 的方式直接做兼容性矩阵。[16]
- **社区实测**：Stack Overflow 等社区存在“`computedRole` 总是空字符串/不工作”的提问，这与“实验性特性门控、默认不可用”的判断一致，但社区信息需要谨慎作为旁证而非结论来源。[17]

**结论（兼容性证据）**：公开兼容性数据库缺位 + 需要实验性开关 + 社区疑问较多 → 不应把它们当作 WebView 中可依赖的生产 API。[2][3][15][17]

# 5) Playwright 的 `ariaSnapshot()`：它走的是 CDP 还是 JS API?

Playwright 在 Chromium 端获取可访问性树的实现路径是 CDP：源码中可见它调用 `Accessibility.getFullAXTree` 并进行后续处理。[6]
W3C WebDriver 也在协议层提供 “Get Computed Role/Label” 命令，用于自动化环境中取计算后的 role/label。[5]

**推论**：Playwright/自动化生态倾向于通过浏览器提供的调试/自动化协议获取“浏览器真实计算结果”，而不是依赖网页脚本可直接访问的 `computedRole` / `computedName` 属性。[5][6]

## 6) 降级方案：如果 `computedRole` / `computedName` 不可用，怎么做？

可选路径按“准确性/工程成本/可部署性”从高到低建议如下：

1. **协议/调试侧拿 AX Tree（最高准确性，但不适合生产端内嵌）**
   - 自动化/测试环境：WebDriver 的 computed role/label，或 Chromium CDP 的 AX Tree（类似 Playwright）。[5][6]
   - Android WebView：理论上可通过远程调试（开发/测试）连接 WebView 的 DevTools，但这通常不适用于生产（权限、稳定性、用户体验、合规）。[13]
2. **JS 侧实现/复用库计算 accessible name + role（可部署，准确性取决于实现覆盖面）**
   - Accessible name：遵循 W3C AccName 1.2 算法（非常复杂，建议复用成熟实现）。[7]
   - Role：参考 ARIA in HTML 的隐式角色映射（HTML 元素/属性 → role），并处理显式 `role` 与 aria 规则优先级。[8]
   - 实现库：`dom-accessibility-api` 等项目提供了与 Testing Library 生态相匹配的可访问名称/角色推导实现，可作为起点，但要评估性能与与浏览器一致性。[9]
3. **业务可接受的“弱一致”启发式（最低成本，但偏差最大）**
   - 只支持常见控件（按钮、链接、输入框、checkbox、radio、select、img 等）的 role/name 计算；
   - 适合做“可观测性/粗粒度 snapshot”，不适合用于严格对齐浏览器/AT 的无障碍语义。

# Areas of Consensus

- 这类“向 JS 暴露浏览器计算后的无障碍语义”的能力长期存在隐私/指纹、实现成本、互操作性风险，因此常以实验性或工具协议形式出现。[1][3][11]
- 在工程实践上，获取“浏览器真实 AX Tree 计算结果”更常见的路线是 WebDriver/CDP，而不是网页 JS 直接 API。[5][6]

# Areas of Debate

- **是否要在产品里“复刻” AccName/Role 计算**：规范复杂度高，复刻容易偏差；但在 WebView 端内嵌场景，协议侧方案往往不可行，只能权衡“足够好”的 JS 实现。[7][8][9]
- **国内 WebView 生态的版本与更新链路**：即便能拿到“Android 系统版本”信息，也无法准确推断 WebView 版本；在缺少 Google Play 生态的环境下，OEM 更新节奏差异很大，覆盖率估算需要真实数据采样。[12][14]

# Sources


[1] Chrome Platform Status API — Feature “Accessible Object Model (AOM)”（Chrome 状态：No longer pursuing；created 2017-01-12，updated 2022-07-26）(高可信：官方) https://chromestatus.com/api/v0/features/6643371200217088

[2] Chromium (googlesource) — `Element.idl`（`computedName` / `computedRole` 且 `RuntimeEnabled=ComputedAccessibilityInfo`）(高可信：一手源码) https://chromium.googlesource.com/chromium/src/+/refs/heads/main/third_party/blink/renderer/core/dom/Element.idl

[3] Chromium (googlesource) — `runtime_enabled_features.json5`（`ComputedAccessibilityInfo` 为 `experimental`）(高可信：一手源码) https://chromium.googlesource.com/chromium/src/+/refs/heads/main/third_party/blink/renderer/platform/runtime_enabled_features.json5

[4] Google Chrome Developers Blog — “Puppetaria: the accessibility-first puppeteer”（提到 `computedName` / `computedRole` 需 `ComputedAccessibilityInfo` flag）(高可信：官方博客) https://developer.chrome.com/blog/puppetaria

[5] W3C WebDriver Living Standard — “Get Computed Role / Get Computed Label” (高可信：标准) https://w3c.github.io/webdriver/

[6] Microsoft Playwright (GitHub) — Chromium accessibility 实现中调用 `Accessibility.getFullAXTree` (高可信：一手源码) https://github.com/microsoft/playwright/blob/main/packages/playwright-core/src/server/chromium/crAccessibility.ts

[7] W3C — Accessible Name and Description Computation 1.2（2026-01-15）(高可信：标准) https://www.w3.org/TR/accname-1.2/

[8] W3C — ARIA in HTML（隐式 role 映射等）(高可信：标准) https://www.w3.org/TR/html-aria/

[9] eps1lon/dom-accessibility-api (GitHub) — 基于规范实现 accessible name/role 推导 (中-高可信：社区主流实现，需与浏览器差异评估) https://github.com/eps1lon/dom-accessibility-api

[10] WICG AOM Explainer (GitHub) — AOM 背景与阶段性目标 (中可信：社区草案/说明文档) https://github.com/WICG/aom/blob/gh-pages/explainer.md

[11] Chromium Docs — Runtime-enabled features（解释 experimental 特性启用方式/语义）(高可信：官方文档) https://chromium.googlesource.com/chromium/src/+/main/docs/blink/runtime_enabled_features.md

[12] Chromium Docs — WebView implementation selection / providers（说明可替换 provider 与选择机制）(高可信：官方文档) https://chromium.googlesource.com/chromium/src/+/main/android_webview/docs/webview-providers.md

[13] Android Developers — WebView DevTools（介绍 WebView flags/调试能力）(高可信：官方) https://developer.android.com/develop/ui/views/layout/webapps/webview-devtools

[14] Android Authority — “Does Android work in China?”（中国大陆 Google 生态缺失的背景说明）(中可信：主流媒体) https://www.androidauthority.com/does-android-work-in-china-3232645/
[15] MDN — `Element.computedRole` / `Element.computedName` 页面不存在（404）(中可信：文档侧“缺位”证据) https://developer.mozilla.org/en-US/docs/Web/API/Element/computedRole
[16] caniuse 索引检索未发现 `computedRole` / `computedName` 独立条目（旁证）(中可信：站点索引侧证据) https://caniuse.com/
[17] Stack Overflow — “computedRole always empty string” (低-中可信：社区旁证) https://stackoverflow.com/questions/75965090/computedrole-always-empty-string

# Gaps and Further Research

- **必须补一组“真机 WebView 实测”**：在 Android 10/11/12/13/14/15 多机型上，验证默认 WebView 环境中 `('computedRole' in Element.prototype)` 是否为真、返回值是否稳定；同时记录 WebView provider 包名与版本号（用于覆盖率建模）。
- **如果要走 JS 降级实现**：需要定义“与浏览器一致”的接受标准（例如与 Chrome DevTools/AXTree 对齐到何种程度），并针对中文/混合内容、shadow DOM、可见性、`aria-labelledby` 链、`::before/::after` 等边界做专项测试。
- **国内生态覆盖率估算**：需要结合你们产品真实用户分布（机型/ROM/是否有 GMS）与运行时采样数据，而不是仅靠公开市场报告推断。