54. script 标签位置的性能影响：为什么建议写在 body 末尾?

1. 核心概念 (Core Concept)

`script` 标签用于在 HTML 页面中嵌入或引用 JavaScript 代码。其在 HTML 文档中的位置会显著影响页面的加载、解析和渲染过程，进而影响用户体验。将 `script` 标签放置在 `<body>` 标签的末尾是一种优化前端性能的常见最佳实践。

2. 为什么需要它？ (The "Why")

将 `script` 标签放置在 `</body>` 结束标签之前主要有以下几个原因，都与提升页面加载性能和用户体验相关：

- **阻塞渲染 (Render Blocking):** 传统上，浏览器在解析 HTML 时遇到 `script` 标签（不带有 `async` 或 `defer` 属性时），会暂停对后续 HTML 内容的解析和渲染，直到脚本被下载、解析并执行完毕。这会延迟用户看到页面内容的时间（白屏时间）。
- **依赖 DOM (DOM Dependency):** 大多数操作 DOM 的 JavaScript 代码需要等待 DOM 结构被完全解析和构建。如果脚本在 DOM 元素解析之前执行，可能会因为找不到对应的元素而引发错误。
- **优化下载顺序 (Optimize Download Order):** 将脚本放在末尾允许浏览器优先下载和解析 HTML 内容、CSS 文件（CSS 通常放在 `<head>` 中，会阻塞渲染直到下载解析完成，以避免无样式内容闪烁 - FOUC），从而更快地构建出页面的视觉结构，提升感知性能。

3. API 与用法 (API & Usage)

此知识点不涉及具体的 JavaScript API，而是对 HTML `script` 标签的**放置策略**。核心用法是：

将所有**非异步加载**的 `script` 标签放置在 `</body>` 结束标签之前，即 HTML 文件的末尾。

经典代码示例:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>HTML 页面结构</title>
    <link rel="stylesheet" href="style.css">
    <!-- 建议将 CSS 放在 head 中 -->
</head>
<body>
    <header>
        <h1>页面标题</h1>
```

```
    </header>

    <main>
        <p>这是一个页面的主要内容。</p>
        <div>
            <img src="image.jpg" alt="示例图片">
        </div>
    </main>

    <footer>
        <p>&copy; 2023 版权所有</p>
    </footer>

    <!--
        强烈建议将 script 标签放在这里，body 的末尾。
        这将允许 HTML 结构和 CSS 样式先加载并渲染，
        然后再下载和执行 JavaScript 脚本。
    -->
    <script src="main.js"></script>
    <script>
        // 行内脚本也可以放在这里
        console.log("脚本已执行");
    </script>

</body>
</html>
```

注意: 现代前端开发中，引入了 `async` 和 `defer` 属性来更好地控制脚本的加载和执行，这为 `head` 中放置 `script` 提供了可行性，且不阻塞 HTML 解析：

- `<script src="script.js" async></script>`：脚本会与 HTML 解析并行下载，下载完成后立即执行，期间可能阻塞 HTML 解析。适合不依赖其他脚本且不操作 DOM 的独立脚本（如统计）。
- `<script src="script.js" defer></script>`：脚本会与 HTML 解析并行下载，但会等到 HTML 解析完成后（DOM 构建完成）才按顺序执行。适合依赖 DOM 或其他 `defer` 脚本的场景。

4. 关键注意事项 (Key Considerations)

- **优先级:** `script` 标签的下载和执行优先级相对较高，不恰当的位置会延缓页面首屏内容的呈现。
- **DOM依赖:** 如果脚本需要操作特定的 DOM 元素，它必须在这些元素被浏览器构建完成后才能执行。将脚本放在 `body` 末尾自然满足这一条件。
- **现代属性 (`async` / `defer`):** 虽然将脚本放在 `body` 末尾是一种普适的古早优化手段，现代浏览器和框架更倾向于使用 `async` 或 `defer` 属性结合在 `head` 中引用脚本，这在实现非阻塞加载的同时，提供了更灵活的执行时机控制。了解并使用这些属性是更佳的实践。

- **关键脚本:** 非常小的、同步的、且必须在 DOM 解析前执行的脚本（例如，用于检测浏览器特性并据此加载不同 CSS 或 JS 的脚本），有时仍需要放在 `<head>` 的顶部，但这应该谨慎使用，并确保脚本体积极小且执行快速，避免长时间阻塞。

5. 参考资料 (References)

- **MDN Web Docs: `<script>` : The Script element** - https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script
- **Google Developers: Optimize the Critical Rendering Path** - https://developers.google.com/web/fundamentals/performance/critical-rendering-path/blocking-css-js (虽然链接可能更新，核心概念不变，搜索 "critical rendering path script blocking" 可找到最新资料)
- **HTML Standard (whatwg): 4.12.1 The script element** - https://html.spec.whatwg.org/multipage/scripting.html#the-script-element (权威规范，描述了脚本的解析和执行行为)