CHROME DEVTOOLS 快速参考

元素、控制台、网络、性能、调试

元素面板

检查与编辑

`Right-click -> Inspect`	打开元素面板定位到该元素
`Double-click tag/attribute`	内联编辑 HTML
`Delete key`	删除选中节点
`Ctrl+Z`	撤销 DOM 修改
`H key`	切换选中元素的可见性
`Drag node`	在 DOM 树中移动元素

样式面板

`element.style {}`	给元素添加内联样式
`Click property value`	实时编辑 CSS 值
`Checkbox next to rule`	切换 CSS 属性开关
`:hov button`	强制伪状态（:hover、:focus）
`.cls button`	添加/移除 CSS 类
`Color swatch`	打开颜色选择器
`Computed tab`	查看最终计算后的 CSS 值

控制台

Console API

```
console.log("info");          console.warn("warning");
console.error("error");       console.table(arrayOrObj);
console.group("label");       console.groupEnd();
console.time("t"); /*...*/ console.timeEnd("t");
```

控制台工具

`$0`	元素面板当前选中的元素
`$('sel') / $$('sel')`	querySelector / querySelectorAll 简写
`copy(obj)`	将对象复制为字符串到剪贴板
`clear()`	清空控制台输出
`monitor(fn)`	记录函数 fn 的每次调用
`monitorEvents(el, 'click')`	记录元素上的事件
`keys(obj) / values(obj)`	对象的键 / 值
`$_`	上一次表达式的结果

过滤

`Log levels dropdown`	按 verbose/info/warn/error 过滤
`Filter text box`	搜索控制台输出
`-url:extension`	按来源 URL 排除消息
`context: dropdown`	按 iframe/worker 上下文过滤

网络面板

网络面板功能

`Filter bar`	按类型过滤：XHR、JS、CSS、Img、Doc、WS
`Search (Ctrl+F)`	搜索所有请求/响应内容
`Preserve log`	页面导航后保留日志
`Disable cache`	DevTools 打开时绕过浏览器缓存
`Throttling dropdown`	模拟慢速 3G、快速 3G、离线
`Block request URL`	右键请求 → Block URL

请求详情标签

`Headers`	请求/响应头、状态码
`Payload`	POST body、查询参数
`Preview`	格式化响应（JSON、HTML、图片）
`Response`	原始响应体
`Timing`	DNS、连接、TLS、TTFB、下载
`Initiator`	触发请求的调用栈
`Cookies`	发送/接收的 Cookie

复制与导出

`Right-click -> Copy as cURL`	将请求复制为 cURL 命令
`Copy as fetch`	复制为 fetch() JavaScript
`Export HAR`	将所有请求导出为 HAR 文件
`Copy response`	将响应体复制到剪贴板

源码面板

断点

`Click line number`	切换行断点
`Right-click -> Conditional`	仅满足条件时中断
`Right-click -> Logpoint`	记录日志而不暂停执行
`DOM breakpoint`	子树/属性/移除时中断
`XHR/Fetch breakpoint`	URL 包含特定字符串时中断
`Event listener breakpoint`	特定事件类型时中断

调试控制

`F8 / Ctrl+\`	恢复 / 暂停执行
`F10`	步过下一个函数调用
`F11`	步入函数调用
`Shift+F11`	跳出当前函数
`Ctrl+Shift+P -> "never pause"`	禁用所有断点

调试面板

`Watch`	监视表达式值
`Scope`	本地、闭包、全局变量
`Call Stack`	函数调用链
`Snippets`	保存并运行可复用的 JS 代码

性能面板

录制

`Record button (Ctrl+E)`	开始/停止录制性能
`Reload button`	录制页面加载性能
`Screenshots checkbox`	捕获视觉时间线截图
`CPU throttle dropdown`	模拟 4x/6x CPU 降速
`Network throttle`	录制期间模拟慢速网络

火焰图解读

`Main track`	JavaScript 执行（火焰图）
`Network track`	网络请求时间线
`Frames track`	FPS 与帧时长
`Timings track`	FCP、LCP、DCL、Load 标记
`Yellow bars`	JavaScript（脚本）
`Purple bars`	布局 / 渲染
`Green bars`	绘制 / 合成

自下而上与摘要

`Summary tab`	时间分布：脚本、渲染等
`Bottom-Up tab`	耗时最多的函数排前
`Call Tree tab`	从根到叶的调用层级
`Event Log tab`	按时间顺序的事件列表

应用面板

存储

`Local Storage`	按来源查看/编辑键值对
`Session Storage`	查看/编辑会话存储
`IndexedDB`	检查对象存储和记录
`Cookies`	按域查看/编辑/删除 Cookie
`Cache Storage`	检查 Service Worker 缓存
`Clear storage`	批量清除选定的存储类型

Service Worker 与 Manifest

`Service Workers`	查看注册状态、push/sync
`Update on reload`	每次重载强制更新 SW
`Bypass for network`	跳过 SW 直接走网络
`Manifest`	查看 Web App Manifest 详情
`Offline checkbox`	模拟离线模式

Lighthouse

运行审计

`Mode: Navigation`	完整页面加载审计
`Mode: Timespan`	审计一段时间内的用户交互
`Mode: Snapshot`	审计当前页面状态
`Categories`	性能、可访问性、最佳实践、SEO
`Device`	移动端或桌面端模拟

核心指标

`FCP (First Contentful Paint)`	首次内容绘制时间
`LCP (Largest Contentful Paint)`	最大内容绘制时间
`TBT (Total Blocking Time)`	长任务阻塞时间之和
`CLS (Cumulative Layout Shift)`	视觉稳定性分数
`SI (Speed Index)`	内容视觉填充速度
`INP (Interaction to Next Paint)`	用户输入响应速度

快捷键

打开 DevTools

`F12 / Ctrl+Shift+I`	切换 DevTools 开/关
`Ctrl+Shift+J`	打开控制台面板
`Ctrl+Shift+C`	打开元素面板 + 检查模式
`Ctrl+Shift+M`	切换设备工具栏（响应式）

导航与搜索

`Ctrl+Shift+P`	命令菜单（执行任意操作）
`Ctrl+P`	打开文件（跳转到文件）
`Ctrl+Shift+F`	跨所有源码搜索
`Ctrl+G`	跳转到源码行号
`Ctrl+[ / Ctrl+]`	切换面板

编辑与控制台

`Ctrl+Shift+D`	切换 DevTools 停靠位置
`Ctrl+L (Console)`	清空控制台输出
`Shift+Enter (Console)`	多行输入
`Esc`	切换控制台抽屉
`Ctrl+K (Console)`	清空控制台

移动端调试

设备工具栏

`Ctrl+Shift+M`	切换设备工具栏
`Device dropdown`	预设手机/平板尺寸
`Responsive mode`	自由调整视口大小
`DPR dropdown`	更改设备像素比
`Throttling`	模拟移动端 CPU + 网络
`Show media queries`	可视化 CSS 断点

远程调试（Android）

`1. Enable USB debugging`	设备 Settings → Developer Options
`2. Connect USB`	将 Android 设备连接到电脑
`3. chrome://inspect`	在桌面 Chrome 中打开
`4. Click Inspect`	为移动端页面打开 DevTools

需要桌面端和 Android 设备均安装 Chrome

传感器模拟

`Geolocation`	覆盖 GPS 坐标
`Orientation`	模拟设备方向
`Touch`	模拟触摸事件
`Idle detection`	覆盖空闲检测 API

常用模式

调试网络问题

```
// In Console: monitor all fetch requests
const origFetch = window.fetch;
window.fetch = (...args) => {
  console.log('fetch:', args);
  return origFetch(...args);
};
```

性能调优流程

`1. Lighthouse audit`	定位主要性能问题
`2. Performance recording`	在火焰图中找长任务
`3. Coverage tab`	找未使用的 CSS/JS（Ctrl+Shift+P → Coverage）
`4. Network waterfall`	识别阻塞资源
`5. Rendering tab`	可视化绘制/布局（Ctrl+Shift+P → Rendering）

常用控制台代码

```
// List all event listeners on element
getEventListeners($0);

// Monitor layout shifts
new PerformanceObserver(l => l.getEntries().forEach(
  e => console.log('CLS:', e)
)).observe({ type: 'layout-shift', buffered: true });
```