

# CHROME DEVTOLS QUICK REFERENCE

Elements, Console, Network, Performance, debugging

## Elements

### Inspecting & Editing

<b>Right-click -&gt; Inspect</b>	Open Elements panel for element
<b>Double-click tag/attribute</b>	Edit HTML inline
<b>Delete key</b>	Delete selected node
<b>Ctrl+Z</b>	Undo DOM change
<b>H key</b>	Toggle visibility of selected element
<b>Drag node</b>	Move element in DOM tree

### Styles Panel

<b>element.style {}</b>	Add inline styles to element
<b>Click property value</b>	Edit CSS value live
<b>Checkbox next to rule</b>	Toggle CSS property on/off
<b>hov button</b>	Force element pseudo-states (hover, focus)
<b>.cls button</b>	Add/remove CSS classes
<b>Color swatch</b>	Open color picker
<b>Computed tab</b>	View final computed CSS values

## Console

### Console API

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

### Console Utilities

<b>\$0</b>	Currently selected element in Elements
<b>\$('.sel') / \$\$('sel')</b>	querySelector / querySelectorAll shorthand
<b>copy(obj)</b>	Copy object as string to clipboard
<b>clear()</b>	Clear console output
<b>monitor(fn)</b>	Log calls to function fn
<b>monitorEvents(el, 'click')</b>	Log events on element
<b>keys(obj) / values(obj)</b>	Object keys / values
<b>\$_</b>	Result of last evaluated expression

### Filtering

<b>Log levels dropdown</b>	Filter by verbose/info/warn/error
<b>Filter text box</b>	Search console output
<b>-url:extension</b>	Exclude messages by source URL
<b>context: dropdown</b>	Filter by iframe/worker context

## Network

### Network Panel Features

<b>Filter bar</b>	Filter by type: XHR, JS, CSS, Img, Doc, WS
<b>Search (Ctrl+F)</b>	Search across all request/response bodies
<b>Preserve log</b>	Keep log across page navigation
<b>Disable cache</b>	Bypass browser cache while DevTools open
<b>Throttling dropdown</b>	Simulate Slow 3G, Fast 3G, Offline
<b>Block request URL</b>	Right-click request -> Block URL

### Request Details Tabs

<b>Headers</b>	Request/response headers, status code
<b>Payload</b>	POST body, query parameters
<b>Preview</b>	Formatted response (JSON, HTML, image)
<b>Response</b>	Raw response body
<b>Timing</b>	DNS, connect, TLS, TTFB, download
<b>Initiator</b>	Stack trace that triggered request
<b>Cookies</b>	Cookies sent/received

### Copy & Export

<b>Right-click -&gt; Copy as cURL</b>	Copy request as cURL command
<b>Copy as fetch</b>	Copy as fetch() JavaScript
<b>Export HAR</b>	Export all requests as HAR file
<b>Copy response</b>	Copy response body to clipboard

## Sources

### Breakpoints

<b>Click line number</b>	Toggle line breakpoint
<b>Right-click -&gt; Conditional</b>	Break only when expression is true
<b>Right-click -&gt; Logpoint</b>	Log without pausing execution
<b>DOM breakpoint</b>	Break on subtree/attribute/removal
<b>XHR/Fetch breakpoint</b>	Break when URL contains string
<b>Event listener breakpoint</b>	Break on specific event types

### Debugger Controls

<b>F8 / Ctrl+\</b>	Resume / Pause execution
<b>F10</b>	Step over next function call
<b>F11</b>	Step into function call
<b>Shift+F11</b>	Step out of current function
<b>Ctrl+Shift+P -&gt; "never pause"</b>	Disable all breakpoints

### Debug Panels

<b>Watch</b>	Monitor expression values
<b>Scope</b>	Local, closure, global variables
<b>Call Stack</b>	Function call chain
<b>Snippets</b>	Save and run reusable JS code

## Performance

### Recording

<b>Record button (Ctrl+E)</b>	Start/stop recording performance
<b>Reload button</b>	Record page load performance
<b>Screenshots checkbox</b>	Capture visual timeline
<b>CPU throttle dropdown</b>	Simulate 4x/6x CPU slowdown
<b>Network throttle</b>	Simulate slow network during recording

## Reading the Flame Chart

<b>Main track</b>	JavaScript execution (flame chart)
<b>Network track</b>	Network requests timeline
<b>Frames track</b>	FPS and frame durations
<b>Timings track</b>	FCP, LCP, DCL, Load markers
<b>Yellow bars</b>	JavaScript (scripting)
<b>Purple bars</b>	Layout / rendering
<b>Green bars</b>	Painting / compositing

### Bottom-Up & Summary

<b>Summary tab</b>	Time breakdown: scripting, rendering, etc.
<b>Bottom-Up tab</b>	Most expensive functions first
<b>Call Tree tab</b>	Root-to-leaf call hierarchy
<b>Event Log tab</b>	Chronological event list

## Application

### Storage

<b>Local Storage</b>	View/edit key-value pairs per origin
<b>Session Storage</b>	View/edit session-scoped storage
<b>IndexedDB</b>	Inspect object stores and records
<b>Cookies</b>	View/edit/delete cookies per domain
<b>Cache Storage</b>	Inspect Service Worker caches
<b>Clear storage</b>	Bulk clear selected storage types

### Service Workers & Manifest

<b>Service Workers</b>	View registration, status, push/sync
<b>Update on reload</b>	Force SW update on each reload
<b>Bypass for network</b>	Skip SW and go to network
<b>Manifest</b>	View web app manifest details
<b>Offline checkbox</b>	Simulate offline mode

## Lighthouse

### Running Audits

<b>Mode: Navigation</b>	Full page load audit
<b>Mode: Timespan</b>	Audit user interactions over time
<b>Mode: Snapshot</b>	Audit current page state
<b>Categories</b>	Performance, Accessibility, Best Practices, SEO
<b>Device</b>	Mobile or Desktop simulation

### Key Metrics

<b>FCP (First Contentful Paint)</b>	Time to first visible content
<b>LCP (Largest Contentful Paint)</b>	Time to largest visible element
<b>TBT (Total Blocking Time)</b>	Sum of long task blocking time
<b>CLS (Cumulative Layout Shift)</b>	Visual stability score
<b>SI (Speed Index)</b>	How quickly content is visually populated
<b>INP (Interaction to Next Paint)</b>	Responsiveness to user input

## Shortcuts

### Opening DevTools

<b>F12 / Ctrl+Shift+I</b>	Toggle DevTools open/close
<b>Ctrl+Shift+J</b>	Open Console panel
<b>Ctrl+Shift+C</b>	Open Elements + inspect mode
<b>Ctrl+Shift+M</b>	Toggle device toolbar (responsive)

### Navigation & Search

<b>Ctrl+Shift+P</b>	Command Menu (run any action)
<b>Ctrl+P</b>	Open file (Go to File)
<b>Ctrl+Shift+F</b>	Search across all sources
<b>Ctrl+G</b>	Go to line number in Sources
<b>Ctrl+[ / Ctrl+] </b>	Switch between panels

### Editing & Console

<b>Ctrl+Shift+D</b>	Toggle DevTools dock position
<b>Ctrl+L (Console)</b>	Clear console output
<b>Shift+Enter (Console)</b>	Multi-line input
<b>Esc</b>	Toggle console drawer
<b>Ctrl+K (Console)</b>	Clear console

## Mobile Debug

### Device Toolbar

<b>Ctrl+Shift+M</b>	Toggle device toolbar
<b>Device dropdown</b>	Preset phone/tablet dimensions
<b>Responsive mode</b>	Freely resize viewport
<b>DPR dropdown</b>	Change device pixel ratio
<b>Throttling</b>	Simulate mobile CPU + network
<b>Show media queries</b>	Visualize CSS breakpoints

### Remote Debugging (Android)

<b>1. Enable USB debugging</b>	Settings -> Developer Options on device
<b>2. Connect USB</b>	Connect Android device to computer
<b>3. chrome://inspect</b>	Open in desktop Chrome
<b>4. Click Inspect</b>	Open DevTools for mobile page

Requires Chrome on both desktop and Android device

### Sensors Override

<b>Geolocation</b>	Override GPS coordinates
<b>Orientation</b>	Simulate device orientation
<b>Touch</b>	Simulate touch events
<b>Idle detection</b>	Override idle detection API

## Common Patterns

### Debug Network Issues

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

## Performance Workflow

<b>1. Lighthouse audit</b>	Identify top performance issues
<b>2. Performance recording</b>	Find long tasks in flame chart
<b>3. Coverage tab</b>	Find unused CSS/JS (Ctrl+Shift+P -> Coverage)
<b>4. Network waterfall</b>	Identify blocking resources
<b>5. Rendering tab</b>	Visualize paint/layout (Ctrl+Shift+P -> Rendering)

## Useful Console Snippets

```
// 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 });
```