

DevTools in QA Testing

How a Junior Tester can find UI, JavaScript and API issues using browser DevTools

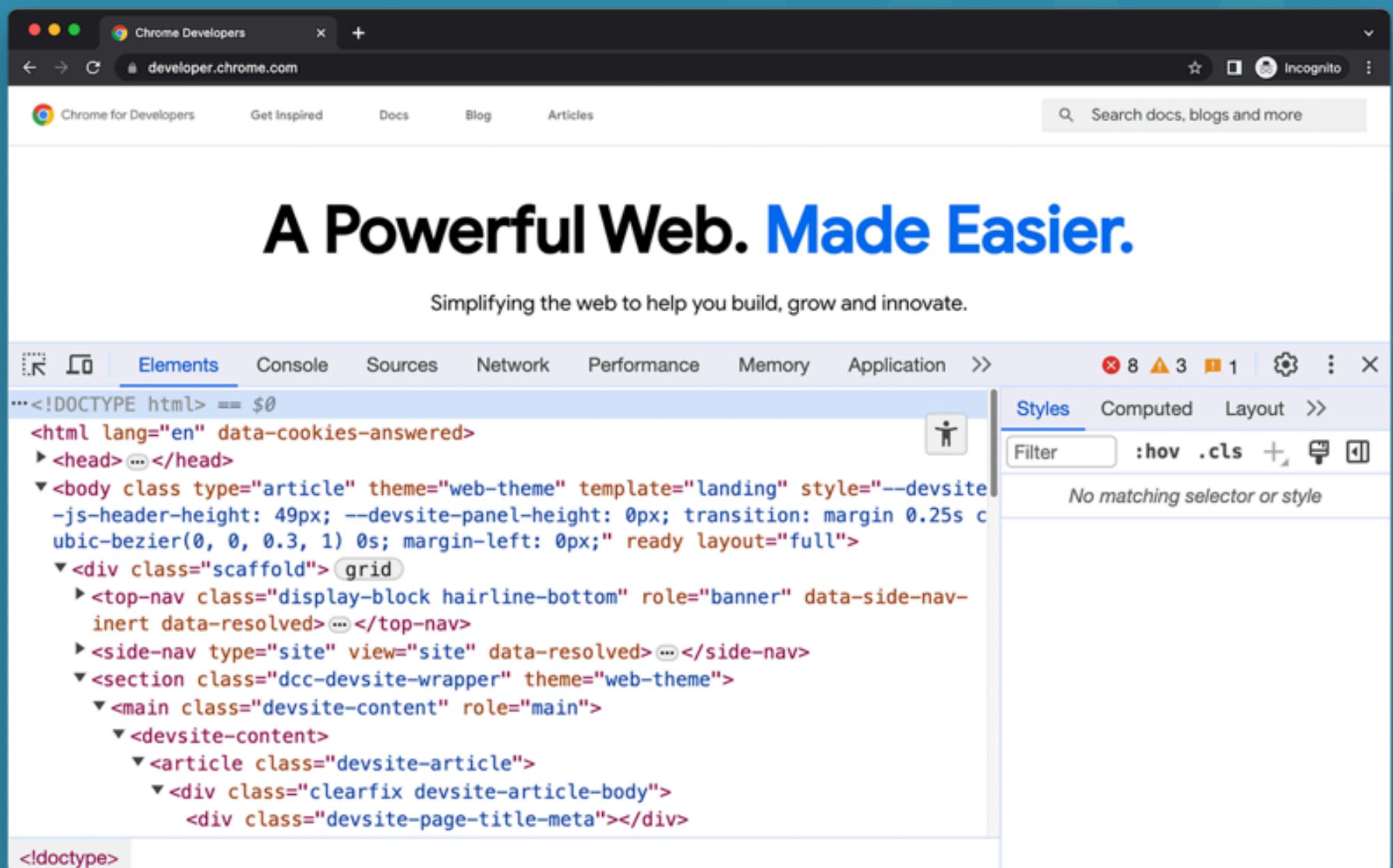


DevTools

Developer Tools (DevTools) are built-in tools available in web browsers that allow inspection and analysis of web applications in real time.

They provide access to:

- page structure (HTML)
- styles (CSS)
- browser logs
- network activity



How to open DevTools



- Right click on page → Inspect
- WINDOWS/LINUX: Keyboard shortcut F12 or Ctrl+Shift+I.
- MAC: Command+Option+I
- Available in all modern browsers

HOW TESTERS USE DEVTOOLS

Testers use DevTools to:

- identify UI and layout issues
- detect JavaScript errors
- analyze API requests and responses
- verify data stored in the browser
- simulate slow network or offline mode

💡 For testers: DevTools is the first place to check when something does not work as expected.

TABS



🔍 **Testers usually focus on Elements, Console, Network and Application tabs.**

- ELEMENTS
- CONSOLE
- SOURCES
- NETWORK
- PERFORMANCE
- MEMORY
- APPLICATION
- SECURITY
- LIGHTHOUSE

ELEMENTS TAB (UI TESTING)

Elements – UI and CSS issues

What testers check:

- missing or hidden elements
- incorrect styles (display, color, size)
- layout issues on mobile devices

How:

- inspect HTML elements
- modify CSS values temporarily
- use device toolbar (responsive mode)

 Example bug:

Button visible on desktop, hidden on mobile.



CONSOLE TAB (JS ERRORS)

Console – JavaScript errors

What testers check:

- red errors (Uncaught TypeError)
- warnings
- error messages after user actions

Test case:

1. Perform an action (e.g. submit a form)
2. Check Console
3. No errors = test passed

🐞 Example bug:

Uncaught TypeError after submitting a form.

JS errors often break functionality.

The screenshot shows the Google Chrome Developer Tools interface with the "Console" tab selected. The title bar indicates the URL is <https://googlechrome.github.io/devtools-samples/debug-js/get-started>. The left sidebar shows the file structure: top, googlechrome.github.io, devtools-samples, get-started, and get-started.js. The "get-started.js" file is open in the main editor area, showing the following code:

```
* YOU MAY OBTAIN A COPY OF THE LICENSE AT
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* UNLESS REQUIRED BY APPLICABLE LAW OR AGREED
* DISTRIBUTED UNDER THE LICENSE IS DISTRIBUTED
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND
* SEE THE LICENSE FOR THE SPECIFIC LANGUAGE GO
* LIMITATIONS UNDER THE LICENSE. */
function onClick() {
  if (!inputsAreEmpty()) {
    label.textContent = 'Error: one or both inputs are empty';
    return;
  }
  updateLabel();
}
function inputsAreEmpty() {
  if (getNumber1() === '' || getNumber2() === '') {
    return true;
  } else {
    return false;
  }
}
function updateLabel() {
  var addend1 = getNumber1();
  var addend2 = getNumber2();
  var sum = addend1 + addend2;
  label.textContent = addend1 + ' + ' + addend2;
}
```

The line `if (!inputsAreEmpty()) {` is highlighted in blue and has a yellow arrow pointing to it from the left margin, indicating it is a breakpoint. The status bar at the bottom right of the developer tools window says "Paused on breakpoint". The right sidebar contains sections for Paused on breakpoint, Watch, Call Stack, onClick, Scope, Local, Global, Breakpoints, XHR/fetch Breakpoints, DOM Breakpoints, Global Listeners, and Event Listener Breakpoints. Under Breakpoints, there is a checked checkbox for "get-started.js:15".

NETWORK TAB (API TESTING)

Network – API testing without Postman

What testers check:

- request type (GET / POST)
- HTTP status codes (200, 400, 500)
- request payload
- response body

Test case:

- Form submitted
- POST request returns 500
- Bug reported

 Example bug:

POST request returns 500 Internal Server Error.



NETWORK THROTTLING

Testing slow network and offline mode

Why it matters:

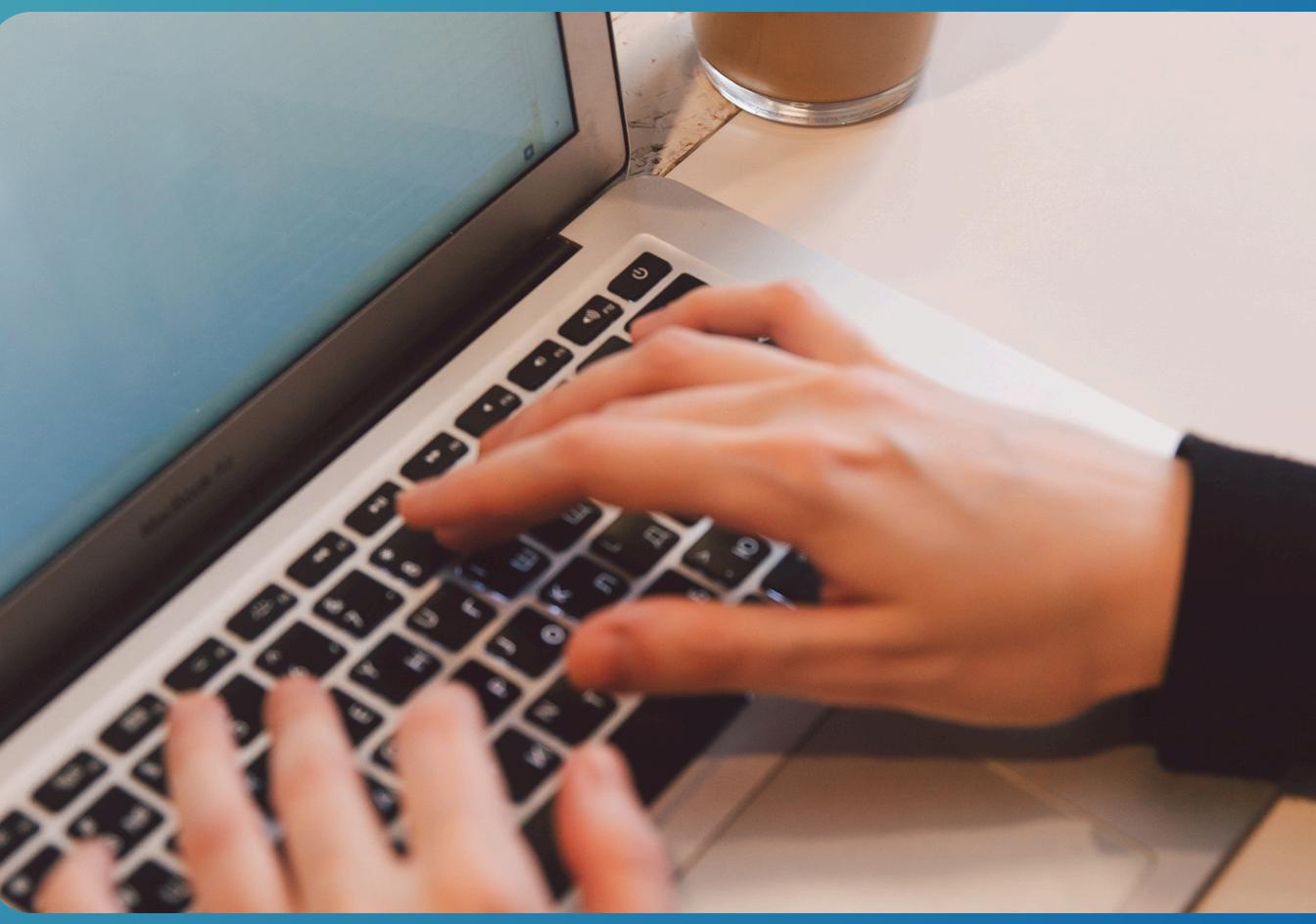
- users may have slow internet
- application should handle delays

How:

- Network → Throttling → Slow 3G
- Offline mode

🐞 Example bug:

Application crashes when internet connection is lost.



PERFORMANCE TAB (BASICS)

Performance – basic speed analysis

What testers check:

- page loading time
- long tasks
- UI freezes

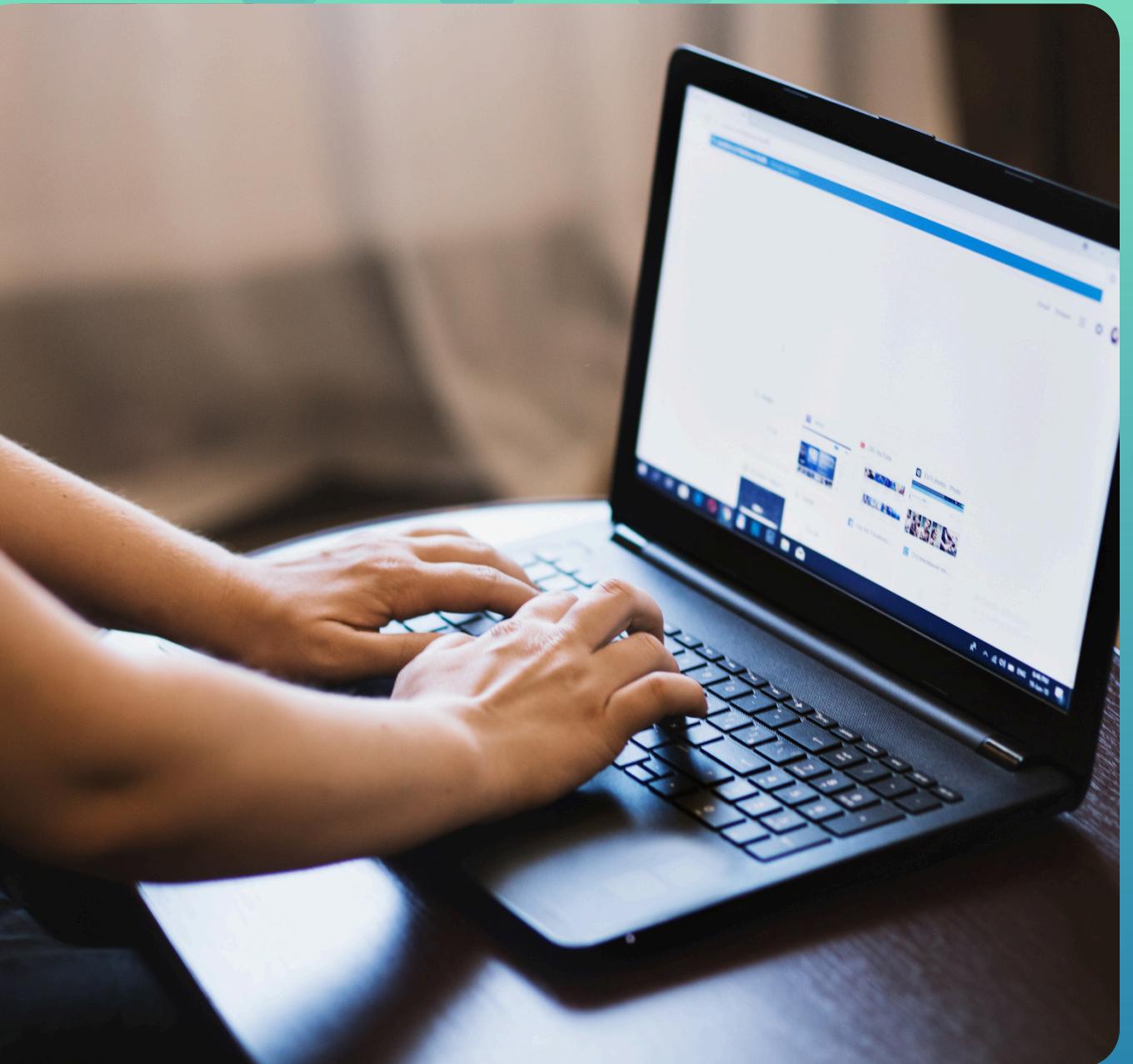
How:

Record page load in Performance tab.

Used mainly for basic performance awareness.

🐞 Example bug:

Page takes longer than 5 seconds to load.





APPLICATION TAB (STORAGE)

Application – user data verification

What testers check:

- Local Storage
- Session Storage
- Cookies

Test case:

- User logs in
- Data should be saved in storage
- Missing or incorrect value = bug

🔑 Important for login and user sessions.

🐞 Example bug:

User session data is missing after page refresh.

LIGHTHOUSE – AUTOMATED AUDITS



Lighthouse – Your Assistant for Quality & Accessibility

An automated tool for improving the quality of web pages.

Key Metrics:

- Performance: Analyzes loading speed and interactivity
- Accessibility: Detects issues for users with disabilities (e.g., low color contrast)
- Best Practices: Checks for modern web standards and security

Tester Use Case: Generate a full report after a release to ensure no performance regressions were introduced



SUMMARY (TESTER FLOW)

When should a tester use DevTools?

- UI looks wrong → Elements
- Feature does not work → Console
- Data not saved / API issue → Network
- User session problems → Application
- Page is slow → Performance

✓ DevTools help testers find bugs faster and report them with evidence.