Chrome DevTools & Debugging Tips for Beginners

Subtitle: Master Chrome DevTools to debug JavaScript, inspect HTML/CSS, and optimize web projects efficiently.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Debugging is a **critical skill for developers**. Chrome DevTools lets you **inspect, test, and debug** your web pages in real-time. This guide teaches beginners how to use DevTools effectively for faster problem-solving.

Step 1: Open Chrome DevTools

- Shortcut: `F12` or `Ctrl + Shift + I` (Windows) / `Cmd + Option + I` (Mac)
- Right-click → Inspect on any element

Step 2: Elements Panel

- Inspect HTML structure
- Edit elements directly in the browser
- Change styles live using the **Styles** tab
- Add/remove classes to test design changes

Mini tip: Use `Ctrl + Shift + C` to select elements on the page

Step 3: Console Panel

- View JavaScript errors and warnings
- Run JS code directly in the console

```
console.log("Hello DevTools");
console.error("Error example");
console.warn("Warning example");
```

- Test DOM manipulation live

Step 4: Sources Panel

- View all scripts and debug code
- Set **breakpoints** to pause execution
- Step over, step into, or resume code
- Inspect variable values during runtime

Mini project idea: Debug a small To-Do app, step through `addTask` function

---

Step 5: Network Panel

- Monitor API requests and responses
- Check **load time, status codes, payloads**
- Identify slow requests and optimize performance

Example: Inspect a fetch request to `https://api.example.com/data`

Step 6: Performance Panel

- Record page load and runtime performance
- Identify bottlenecks in rendering or JavaScript execution
- Optimize heavy functions or large DOM updates

Step 7: Application Panel

- Inspect Local Storage, Session Storage, and Cookies
- Check service workers and cache
- Useful for debugging saved app state or offline functionality

---

Step 8: Mini Project Idea

- Open a sample project (e.g., To-Do App)
- Use **Console** to log tasks being added

- Inspect **Network** panel to debug API requests
- Set **breakpoints** in Sources to debug function logic
- Check Local Storage for persisted data

Step 9: Best Practices

- Always check **Console for errors** before debugging
- Use **breakpoints** instead of `alert()` for cleaner debugging
- Monitor **Network** for API issues
- Inspect **Application storage** to validate persistence

Step 10: Key Takeaways

- Chrome DevTools is essential for web development
- Inspect HTML/CSS, debug JS, monitor network and storage
- Practice with small projects to master all panels
- Effective debugging saves time and improves code quality

Visit **haas.dev** for detailed DevTools tutorials, debugging exercises, and beginner-friendly web development guides.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app
