**Optimize Your Responsive Site**

Apply real-world performance techniques to your project and measure improvements.

**Instructions**

**Part 1: Optimize Images**

* Replace at least one <img> with <picture> and srcset.
* Compress your images using [Squoosh](https://squoosh.app/).
* Add loading="lazy" to all below-the-fold images.

**Part 2: Minify and Combine Assets**

* Minify your CSS and JS using an online tool or build tool.
* Combine multiple CSS files into one if possible.
* Use a bundler (Vite, Webpack) to automate this if applicable.

**Part 3: Test Performance**

Use these tools:

* **[Lighthouse in Chrome DevTools]** → Run audit > Performance
* **WebPageTest.org** or **GTmetrix** → See TTFB, speed index
* **Network tab** → Check load time and request sizes

**How to Use Lighthouse**

1. **Open your website** in **Google Chrome**.
2. **Right-click** anywhere on the page and choose:  
   Inspect  
   (or press Cmd + Option + I on Mac / Ctrl + Shift + I on Windows)
3. In DevTools, click the **“Lighthouse”** tab.  
   (If you don’t see it, click the » menu at the top of DevTools to reveal hidden tabs.)
4. Choose your settings:
   * **Device**: Select Mobile or Desktop
   * **Categories**: Check Performance (you can also include Accessibility, Best Practices, etc.)
   * Optional: Enable Simulated throttling to mimic slow networks
5. Click the **“Analyze page load”** button.
6. Wait a few seconds while Lighthouse audits the page.
7. Review your report! You'll see:
   * **Performance score**
   * **Largest Contentful Paint (LCP)**
   * **Time to Interactive**
   * **Total Blocking Time**
   * **Image size warnings**
   * **Improvement suggestions**

**Tips and Tricks:**

* For more accurate results, **run in Incognito Mode** to disable browser extensions.
* Run the audit **after a hard reload** (Cmd + Shift + R / Ctrl + Shift + R).
* Click “View Original Trace” to dig into load timelines.

**Sample Template for Keeping Track of Testing & Results**

|  |  |  |  |
| --- | --- | --- | --- |
| **Optimization Step** | **Before (ms)** | **After (ms)** | **Notes** |
| Image Optimization | 450ms | 150ms | Used WebP + lazy loading |
| JS/CSS Minification | 700ms | 400ms | Used Vite + Terser |
| Total Page Load | 2.5s | 1.2s | Better mobile scores |

**Review:**

**1. Which change can make the biggest difference?**

Using <picture> with srcset for responsive images and compressing them with Squoosh likely has the biggest impact. It reduced total image size by over 60% and significantly improved load time and speed index.

**2. What performance trade-offs might you face?**

* Combining CSS files speeds up load time but made development harder without source maps.
* Using lazy loading improves *perceived* performance but causes a slight delay in image appearance when scrolling quickly.
* Minifying assets removes comments and spacing, which can be annoying during debugging without proper sourcemaps.

**3. What tools or automation could help in the future?**

* **Vite/Webpack** for automated bundling and minification.
* **Image CDN** for responsive image delivery.
* **Lighthouse CI** for automated audits in CI/CD pipelines.
* **npm scripts** or tools like Gulp to automate asset compression.