hover image previews), or *Erik Carter's* site which is very typographic. Look at design portfolios that prioritize text and see how they handle hierarchy and transitions. You can also find templates meant for designers – many are open source (e.g., **Eleventy Duo** is a minimal theme for portfolios/blogs [18]). Picking apart these examples will inform how to achieve an elegant, high-end tone with simple elements.

3. Performance Optimizations & Mobile-First Implementation Tips

A key goal is **speed**, especially on mobile. A custom static site will already drop a lot of bloat that a platform might include, but you should further optimize for a snappy experience:

- **Optimize and Lazy-Load Images:** Visual content is often the heaviest part of a portfolio. Start by **compressing images** (and videos, if any) before including them. Large files dramatically slow pages; export images at appropriate dimensions and use tools or plugins to compress them (e.g. TinyPNG or Squoosh for images) [19]. Serve modern formats like **WebP** or **AVIF** which are much smaller than JPEG/PNG (you can fall back to JPEG for older browsers). Implement native lazy-loading (`<img loading="lazy">`) so below-the-fold images load only as the user scrolls [17] – this is especially crucial for the "Grid view" of thumbnails and any image-heavy case study pages. This way, the initial page load is lightweight, and images pop in as needed, preserving both speed and a progressive loading effect.

- **Minimize CSS and JavaScript:** Aim for a lean site with minimal files. Combine and minify your CSS and JS assets so that the browser makes as few requests as possible [20]. For example, instead of multiple CSS files, have one minified CSS file (your SSG pipeline or build tools like esbuild/webpack can do this). Similarly, any custom JS (perhaps for the view toggle or analytics) should be in one small file. Often, a simple portfolio might not need any large JS libraries – vanilla JS for a toggle or using CSS for interactive effects is sufficient. Avoid heavy frameworks or unnecessary plugins; every kilobyte counts for mobile.

- **Reduce External Dependencies:** Each external resource (fonts, scripts, iframes) can slow down the site. Evaluate if you need them:

- **Webfonts:** Custom fonts can enhance branding, but they add load time. Limit the number of font families/weights, and use `font-display: swap` in CSS so text isn't invisible while fonts load. Consider serving the fonts yourself (download from Google Fonts and host locally) to eliminate extra DNS requests [21]. If performance is paramount and your design can work with system fonts, you could even use a font stack (e.g. sans-serif stack that uses Helvetica/Arial on Mac/PC) for zero font overhead, but most portfolios benefit from a distinctive typeface – just use wisely.
- **Analytics or embeds:** If you use Google Analytics or other scripts, consider lightweight alternatives or none at all, to keep JS minimal. If you have video embeds (Vimeo/YouTube), use preview images or load them on demand (as these can be heavy).

- **Hosting/CDN:** GitHub Pages will host your static files on a global CDN, which is good. If you use Netlify or Cloudflare Pages, they similarly distribute content. Using a CDN ensures faster delivery on mobile networks globally, so you're covered on that front by default.