.html

Responsive Images & Srcset: A Practical Checklist

Use this checklist to audit existing projects and as a guide for new ones to ensure you are following the best practices for responsive images.

📋 Phase 1: Strategy & Preparation

Before you write any code, make sure you have a clear plan.

- ☐ **Audit Images:** Identify the heaviest and most important images on the page (e.g., the Largest Contentful Paint element).
- ☐ **Identify Layout Breakpoints:** Determine at what viewport widths your design changes and how the image container's size is affected.
- ☐ **Create a Set of Image Sizes:** Generate several versions of each image (e.g., 400w, 800w, 1200w, 1600w, 2000w) from a high-resolution source.
- ☐ **Choose the Right Format:**
  - **JPEG/WebP:** For photographs.
  - **PNG/WebP:** For graphics with transparency.
  - **SVG:** For logos and icons.
- ☐ **Determine the Need for Art Direction:** Decide if you need to show different crops of an image on different screens. If yes, plan to use the `<picture>` element.

💻 Phase 2: HTML Implementation

Write your code correctly for maximum efficiency and compatibility.

- ☐ **Use `<img>` with `srcset` and `sizes` for Resolution Switching:** This is the primary method for displaying the same image at different sizes.
- ☐ **Specify the `w` Descriptor:** Ensure each source in `srcset` has its actual width specified (e.g., `image-800w.jpg 800w`).
- ☐ **Write an Accurate `sizes` Attribute:** This attribute is **MANDATORY** when using the `w` descriptor. It must accurately describe how wide the image will be under different viewport conditions.
- ☐ **Use `<picture>` for Art Direction:** For showing different cropped versions of an image.
- ☐ **Use `<picture>` for WebP Support:** Provide the WebP format with a JPEG/PNG fallback for older browsers.
- ☐ **Include a Standard `src`:** Always include a `src` attribute as a fallback for the oldest browsers that don't support `srcset`.
- ☐ **Write a Descriptive `alt` Attribute:** Always fill in the `alt` attribute for accessibility.

⚙️ Phase 3: Optimization & Automation

Streamline your workflow and achieve maximum performance.

- ☐ **Compress All Images:** Use tools to compress images without a visible loss in quality.
- ☐ **Consider Using an Image CDN:** Services like Cloudinary or Imgix can automate size creation, WebP conversion, and image delivery.
- ☐ **Use CMS Plugins:** If you're on WordPress, use plugins (like Smush, ShortPixel) to automatically generate `srcset`.
- ☐ **Implement Lazy Loading:** For images below the fold, use `loading="lazy"` to speed up the initial page load.

✅ Phase 4: Testing & Verification

Make sure everything works as intended.

- ☐ **Use Browser Developer Tools:** Open the "Network" tab and, while resizing the window, confirm that the correct (smaller) image versions are being loaded for smaller screens.
- ☐ **Test on Real Devices:** Test on actual smartphones and tablets, not just in an emulator.
- ☐ **Check Load Speed:** Use PageSpeed Insights or WebPageTest to assess the impact of your images on overall performance.
- ☐ **Validate Your HTML:** Ensure there are no syntax errors in your markup.