The American Journal of Social Science and Education Innovations
ISSN 2689-100X | Open Access

TYPE Original Research
PAGE NO. 103-111
DOI 10.37547/tajssei/Volume07Issue05-13

Check for updates

Implementing Sustainable Digital Design Principles into Web Product Development.

Abramau Usevalad

Senior Art Director, Kargo Jersey City, NJ, USA

OPEN ACCESS
SUBMITED 19 March 2025
ACCEPTED 28 April 2025
PUBLISHED 23 May 2025
VOLUME Vol.07 Issue 05 2025

CITATION
Abramau Usevalad. (2025). Implementing Sustainable Digital Design Principles into Web Product Development. The American Journal of Social Science and Education Innovations, 7(05), 103–111. https://doi.org/10.37547/tajssei/Volume07Issue05-13,

Abstract: This article examines the principles of sustainable digital design as a holistic approach to web-product development, wherein each interface optimization considers not only user experience and business metrics but also the material costs of computation, data transmission, and rendering. The relevance of this work is driven by the rapid increase in energy consumption of data centers and user devices under the influence of digitalization, and by the necessity to minimize the carbon footprint of web services amid constraints of the “green” energy system and social responsibility for product accessibility. It seeks to organize and measure important ways of eco-friendly web design—the choices of loading, dark styles, graphic types, shortening and shaking trees, client or server rendering options, adaptive delivery of content, and CO_2 budgets—based on their effects on speed, energy used by devices, and the environment. For this purpose, a careful look at industry reports, academic tests, and real-world studies was done, plus a metric comparison from the Web Almanac Core, Web Vitals, and lab tests. The novelty of this work lies in integrating ecological, economic, and technical metrics into a unified methodology. For each technique, comparable numerical estimates of carbon-emission reduction and business-efficiency gains are presented, and recommendations are developed for their combined application within the CO_2 budget of first-screen delivery. Key findings indicate that activating lazy loading with a single HTML attribute can reduce transferred data volume and improve Time to Interactive by 20–30%; employing a dark theme on OLED displays under bright lighting can decrease display power consumption by up to 47%; and replacing PNG images with SVG sprites can reduce graphic payload by 60–80%. This article will be valuable to web designers, front-end developers, product managers, and IT strategists seeking to marry high interface performance with a minimal

carbon footprint.

Keywords: sustainable digital design, web performance, lazy loading, dark theme, SVG, tree shaking, server-side rendering, adaptive loading, CO_2 budget.

Introduction: Sustainable digital design conceives every interface as a chain of material events—from CPU cycles to megawatt-hours in data centers—and is thus founded on the principle that "every byte has mass." When designers account for carbon footprint alongside pixels, and engineers allocate budgets for memory, bandwidth, and processing power from the outset, the resulting product is not only "lightweight" but also resilient: it loads faster, supports a broader range of devices, requires hardware upgrades less frequently, and better adapts to technological shifts. In this approach, aesthetics, code, and economics converge on a single objective—simultaneously reducing costs for users, organizations, and the planet. The imperative to pursue sustainability became undeniable once "digital" ceased to be immaterial. Data production, storage, and transmission already account for 1.5–4% of global greenhouse-gas emissions, and this share continues to rise [1]. The International Energy Agency forecasts that data-center electricity demand will double by 2030, driven particularly by the surge in generative AI workloads [2]. It is increasingly clear that, without a fundamental rethink of digital architecture and design, digital infrastructure will expand faster than the decarbonization of the power sector can keep pace, thereby becoming a significant barrier to global efforts to reduce emissions.

This load growth carries environmental, social, and ethical risks. Algorithmic data management raises the question of "who is accountable for machine errors," while information overload in interfaces provokes user fatigue and erodes trust. Moreover, most users continue to access the web via older smartphones and slower networks; when products cater primarily to flagship devices, they inadvertently exacerbate the digital divide. In this context, systemic responsibility extends beyond internal optimization and aligns with the United Nations Sustainable Development Goals: quality education (Goal 4), reduced inequalities (Goal 10), climate action (Goal 13), and strong institutions (Goal 16) depend directly on how thoughtfully we design digital environments.

From a business perspective, sustainable design proves to be not charity but a mechanism for efficiency gains. Major platforms report double-digit declines in CDN and request-processing expenditures by reducing image sizes, implementing lazy loading, and adopting server-side rendering. At the same time, interface speed delivers direct revenue: a study [3] showed that cutting page-load time from five to one second increases purchase conversion by a factor of 2.5. Accessibility improvements also monetize: Forrester analysts calculated that every dollar invested in inclusive enhancements can yield up to one hundred dollars in cost savings and additional revenue [4]. Beyond direct returns, there are intangible dividends—improved brand loyalty, reduced legal risk, and enhanced reputation as a responsible organization.

Finally, code minimalism benefits developers themselves. Fewer dependencies and less "fast fashion" in the interface reduce the need for emergency patches, lower the barrier to entry for new team members, and extend the product's lifespan without radical redesign. Ultimately, sustainable digital design emerges not as a compromise between ecology and usability but as a strategy in which economic, technical, and social benefits coalesce into a single, easily quantifiable value.

MATERIALS AND METHODOLOGY

The materials and methodology of this study are based on a comprehensive analysis of 23 sources, including international reports, academic publications, industry benchmarks, technical documentation, and practical implementation case studies. Principal sources include the World Bank report on the ICT sector's share of global emissions [1] and the IEA forecast on data-center energy growth under the influence of AI [2], as well as studies on the economic effects of web-performance and accessibility optimization [3, 4]. To assess the prevalence and efficacy of specific practices, data from The Web Almanac (Media and Sustainability sections) [5, 19] were employed, alongside laboratory measurements and field-test results published in industry blogs and technical articles [6–8, 10–12, 18].

The theoretical foundation comprises works demonstrating the linkage between interface optimization and device energy savings: a study of lazy

loading in WordPress themes [6] and experiments by Fast Familiar Workroom on energy savings during page scrolling on budget smartphones [7]; analyses of dark-theme effects on OLED screens under high- and low-ambient-light conditions [8–10]. To evaluate vector graphics and sprite benefits, comparison data from Vecta.io on PNG versus SVG [11] and Cloudinary recommendations for optimizing SVG sprites were used [12]. Correlations between JavaScript-bundle minification and reduced parsing time were examined in two independent SaaS-platform case studies by Wagner et al. [13, 14].

Methodologically, this research combines systematic review and content analysis of documentation, comparative technology analysis, and quantitative efficiency estimates. The systematic review encompassed standardized metrics from the Web Almanac and Google Core Web Vitals [5, 15], while content analysis covered articles on tree shaking and SSR versus CSR in Next.js projects [13, 16]. Adaptive loading implementation details were examined using Client Hints and the Network Information API in React-Hooks practices [18], along with Google statistics on the impact of load speed on user churn [17]. CO_2 budgeting control relies on Web Almanac page-weight recommendations and data from the Website Carbon service [19, 20].

RESULTS AND DISCUSSION

Lazy loading is a simple technique, but its effects extend beyond local speed optimization. When the browser only fetches those images, video frames, or iframes that fall within the user's viewport, it reduces the initial volume of data that must be transmitted and decoded. According to the Web Almanac 2024, the native loading="lazy" attribute is already used on one-third of all pages, making it the fastest-growing media-handling practice [5]. Laboratory measurements on popular WordPress themes show that after enabling lazy loading of visual resources, Time to First Interaction is reduced by 20–30%, and the total weight of initial HTML plus critical styles and scripts decreases by approximately one third [6]. The savings are particularly pronounced on mobile networks: a study [7] on several budget smartphones recorded lower battery discharge during scrolling a typical landing page when images were loaded lazily.

The technical implementation requires no heavy libraries: for most browsers, it suffices to add the loading="lazy" attribute to the <img> or <iframe> tag, and for fine-tuning the visibility thresholds, one can use the IntersectionObserver API. An example implementation is shown in Fig. 1.

```
"HTML (natively):

<img src="image.jpg" loading="lazy" alt="Description" />


"JS (for more fine-tuning):

"Intersection Observer API – standard for tracking what's in the viewport:
const observer = new IntersectionObserver((entries) => {
  entries.forEach(entry => {
    if (entry.isIntersecting) {
      entry.target.src = entry.target.dataset.src;
    }
  });
});
document.querySelectorAll('img[data-src]').forEach(img => observer.observe(img));
```

Fig. 1. Example of implementation of lazy loading (compiled by author)

Thus, a single line of HTML becomes a tool that can significantly reduce page weight—by 30–60% when working with heavy media files—while simultaneously accelerating Time to Interactive by 20–40%, decreasing device energy consumption (especially on mobile devices), and reducing the carbon footprint by approximately 10–20 g CO_2 per thousand loads. Along with other sustainable design principles, lazy loading is the first barrier against interface bloat. It allows resources to remain where they are genuinely needed and confirms that user and planetary savings in web development coincide.

After optimizing traffic and computation, the next level concerns care for pixels: the dark theme turns background "black" into the literal deactivation of subpixels on OLED screens, so each unlit diode saves fractions of a watt and slows battery discharge. The principle is the same as for lazy loading: a minor code change shifts resource allocation across the entire chain, and again, visual aesthetics can simultaneously reduce carbon footprint and enhance usability.

The physics of the process is simple: in an organic light-emitting diode, only those subpixels that must be brighter than absolute black emit light; therefore, an interface dominated by #000000 consumes less energy than a traditional light background. Laboratory measurements of six popular Android applications on four generations of OLED smartphones showed that, at brightness automatically increased to 100% under sunlight, switching to dark mode reduces display power consumption by 39–47%, and in maximally contrasted scenarios, savings reach 63% of total screen power [8]. However, the same experiment recorded only a 3–9% benefit indoors at average brightness, and a recent study [9] found that 80% of users compensate for the dark palette by increasing brightness, thus nullifying or even reversing the effect [9]. The energy balance, therefore, depends not only on color but also on context: dark mode is useful under high-brightness conditions, whereas in evening lighting, the gain is determined by the user's discipline.

Beyond electricity, dark mode affects comfort: reduced overall screen luminance decreases glare and retinal irritation, which is especially noticeable at night. It is no coincidence that Flurry analytics showed that after 22:00, 82.7% of smartphone users switch to a dark interface [10]. Increased reading comfort prolongs session length; consequently, news feeds and messaging applications record higher usage times, achieving direct conversion returns without additional ad units.

Implementation in the browser requires one line: the media query @media (prefers-color-scheme: dark) allows palette overriding without duplicating styles. Mobile frameworks—from Swift UI to Android Jetpack—inherit the system theme, and persisting the user's choice via localStorage or SharedPreferences consumes no more than a dozen bytes. Like lazy loading, the effect happens without outside libraries. It does not make the setup more complex: the lighter the static assets, the better they work on slow devices.

Sustainable design gives context to dark mode, verifying the central thesis of this article: a design decision is "green" not because it looks minimalist, but statistically it reduces energy consumption, prolongs battery life, and lowers cumulative emissions throughout the device's lifecycle. Proper brightness configuration and consideration of user scenarios are other examples of how caring for the planet aligns concurrently with business and user interests.

Once lazy loading and dark mode have reduced code and rendering costs, the next logical step is to lighten the graphics. Replacing raster PNG and JPEG with vector SVG shifts the interface from pixel-based descriptions to geometry, so that the same icon weighs tens of kilobytes less and renders without loss of clarity even on Retina displays. In a laboratory test by Vecta.io, the difference reached 60–80% in favor of SVG—8 KB versus 82 KB for the PNG counterpart—directly saving bandwidth and accelerating first content rendering [11]. Additional gains come from the sprite approach: when dozens of icons are combined into a single file, the browser requires only one HTTP request. The study recommends SVG sprites as a request-optimization practice [12]. Ultimately, vector graphics reduce the data transmission carbon footprint and spare the product from storing multiple raster copies for different DPIs. An example of usage is shown in Fig. 2.

```
HTML:
<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg">
  <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />
</svg>
```

Fig. 2. Example of implementation of SVG images (compiled by author)

However, even a "bare" SVG can be wrapped in superfluous JavaScript code. Minification and tree shaking, built into production modes of Webpack, Vite, and other bundlers, eliminate this excess by removing whitespace, unused variables, and entire modules. A demonstration SPA on web.dev, after enabling tree shaking, "slimmed down" from 20.8 KB to 8.5 KB—the main bundle's reduction was almost 60%—with the savings showing not only in traffic but also in script parsing and compilation times [13]. Larger projects confirm the scalability of this result: in a 2025 case, a SaaS platform team reduced its production bundle by 70%—from 4.8 MB to 1.6 MB—through import refactoring, thereby restoring acceptable speed on 3G networks [14]. Reduced computation directly lowers energy consumption on user devices and cuts CDN-traffic costs.

The following optimization layer concerns where the browser executes the remaining code. Client-side rendering is convenient for single-page applications but requires full JavaScript loading before the first pixel appears, thus shifting computational costs to the user's device. Server-side rendering, by contrast, delivers ready-to-render HTML and allows the browser to begin rendering immediately upon response receipt. This is why Core Web Vitals guidelines recommend SSR to improve LCP: having markup in the initial HTML reduces main-content load latency and diminishes main-thread blocking [15]. Practical measurements in Next.js projects show that server-side or hybrid rendering accelerates Largest Contentful Paint by 30–50% compared to classic CSR and simultaneously increases store conversion rates by tens of percent due to the shorter user-exit window [16]. The weaker the device and the less stable the network, the more pronounced the difference: on budget smartphones under 4G conditions, SSR offloads CPU work, reducing Total Blocking Time and, consequently, energy consumption, which aligns with sustainable-design goals.

Thus, transitioning to SVG, aggressive bundle cleansing, and considering rendering-mode choice from another savings profile in which every kilobyte and millisecond converts into tangible monetary and ecological dividends. Combined with lazy loading and dark mode, this creates a sustainable architecture in which visual quality, accessibility, and carbon-footprint reduction reinforce one another, and the product remains "light" for both the user and the planet.

After code and graphics have been lightened via SVG and minification, the next logical level of sustainability is to adjust what and to whom the server delivers flexibly. Real-world users access content from dozens of devices and network types, and Google statistics show that 53% of mobile sessions drop off if a page loads for more than three seconds [17]. Therefore, rather than indiscriminately sending the same "heavy" package to all users, adaptive loading assesses current conditions and proportionally reduces data volume: a low-end smartphone on 3G receives a streamlined layout without video backgrounds, while a flagship on Wi-Fi receives the full media version.

Technically, the solution is built on Client Hints and the Network Information API. The browser reports effective connection type, the save-data flag, number of cores, and amount of memory, and the script or server decides which images, fonts, and scripts are essential. Google's method is detailed using React Adaptive Loading hooks, where "slow-2g" connections receive thumbnails instead of posters, and budget "quad-core" devices skip heavy animations [18]. Such load distribution is especially beneficial in regions with unstable internet: even without changes to business logic, the interface becomes noticeably faster, conserves data and battery,

and thus reduces emissions associated with modem and processor operation.

To prevent this additional logic itself from becoming ballast, it is simultaneously necessary to reconsider what qualifies as "essential" visual content. According to the Web Almanac 2024 Sustainability chapter, more than half of a page's total weight is attributable to images [19]. By replacing decorative PNG icons with text or system emoji, and raster logos with a typographic brandmark in the default font, one can instantly eliminate dozens of requests and tens of kilobytes without loss of meaning. Text requires no decoding, adapts to dark mode, and is immediately available to screen readers, enhancing accessibility. Practice shows minimal typography and color accents are perceived as expressively as static illustrations, especially on small screens.

However, even after all local optimizations, a project risks reverting to bloat without clear quantitative benchmarks. That is why more companies are introducing a CO_2 budget: in addition to traditional time and pixel metrics, they set an upper limit on the mass of the first screen. The Web Almanac recommends staying below 1 MB, with an ideal threshold around 500 KB [19]. Page-weight statistics are presented in Fig. 3. Such a limit helps eliminate superfluous elements at the design-mockup stage: every new library or background video must pass the test of "is it worth its carbon cost." External services like Website Carbon show that today's average page emits about 0.8 g CO_2 per view, providing a clear savings target [20].

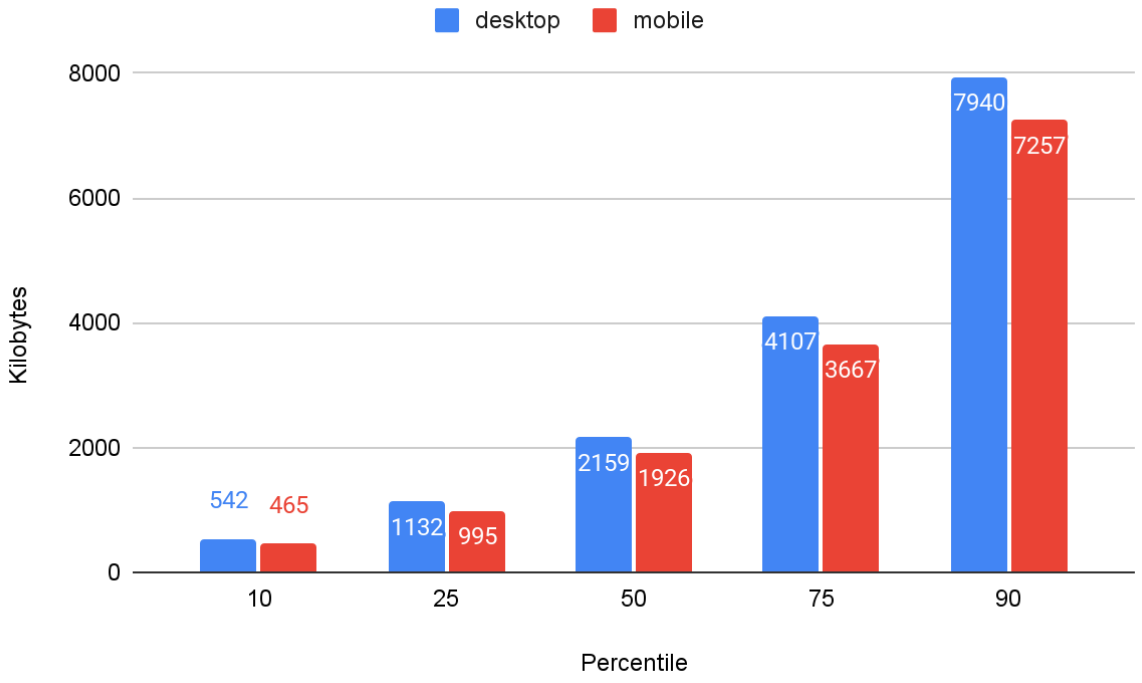

Fig. 3. Page weight by percentile [19]

Adaptive loading, eliminating unnecessary images, and proactive weight control complement the previously discussed techniques of lazy loading, dark mode, and bundle optimization. Together, they form a framework where every additional byte must justify itself by benefiting the user. Thus, design, performance, and climate responsibility inevitably converge into a single engineering discipline, making web products faster, more accessible, and more durable.

A striking example is Google Chrome: browser-level optimization, implemented under the names Data Saver and Lite Mode, takes adaptive-loading principles to their logical extreme—compressing and simplifying the entire page before it reaches the device. When a connection is classified as "2G" or "slow-2G," Chrome proxies the request through Flywheel, replaces heavy images with WebP thumbnails, removes unnecessary JavaScript, and delivers a "Lite page." Field measurements showed that

this scheme reduced data transfer volume by up to 90% and halved load time; most importantly, on slow networks, the share of pages that fully render before timeout increased significantly [21]. This result demonstrates that traffic optimization is not merely a technical enhancement but a direct factor in audience retention.

The effect is especially pronounced in contexts with limited data plans and unstable connectivity. When Google launched Lite Mode for news in India, the project targeted 200 million mobile users. It reduced data consumption to less than one-third of the standard mode while preserving key information—headlines and text [22]. The practice proved that "sustainability through inclusivity" works in two ways: on one hand, saving kilobytes and grams of CO_2; on the other, making the product accessible to a broader audience, thereby boosting loyalty and expanding the market in countries with emerging digital infrastructure.

It is important to note that since 2022, Lite Mode in Chrome has been disabled: mobile networks became cheaper, and the browser's architecture learned to save data by default [23]. Nevertheless, the case remains illustrative for sustainable design: it shows how focusing on real usage contexts spurs creative rethinking of the interface and the entire content-delivery chain. When teams are given room for such solutions, ecological and economic benefits are complemented by intangible value—their product becomes harder to replicate because it is built upon deep user understanding and responsible choice of every byte.

CONCLUSION

It has been demonstrated in the above analysis that sustainable digital design principles in web-product development can yield sustainability for the user, business, and environment at different levels of participation. Lazy loading reduces the volume of transmitted data and TTFI (Time to First Interaction) with the user, thereby reducing loads on both data centers and devices, subsequently reducing the carbon footprint by tens of grams of CO_2 per thousand loads. A dark interface theme on OLED screens shows optimized display energy consumption. It enhances user comfort, particularly under high-brightness conditions, reflecting the direct interrelation between aesthetics and efficiency.

Further graphics optimization through vector SVG images and sprite assembly reduces visual-resource weight by 60–80%, eliminating the need to store multiple raster copies and speeding up rendering. Minification and tree shaking in modern bundlers allow JavaScript bundles to "slim down" by nearly half, reducing CPU load on devices and CDN traffic. The choice between client-side and server-side rendering adapts to device and network capabilities, ensuring maximal speed of the first painted content.

Adaptive loading based on Client Hints and the Network Information API generates a personalized data package according to connection quality and device capabilities, which is especially relevant for users of budget smartphones and regions with unstable internet. However, the effectiveness of all these techniques depends on precise page weight control and introducing a CO_2 budget: maintaining the first-screen mass within 500–1000 KB becomes a criterion for including new libraries and visual elements.

Thus, sustainable digital design comes not as a part of disparate techniques but as an integrated part of an engineering discipline where every decision made, from a single line of HTML to architectural choices, measurably influences performance, accessibility, and climate responsibility. Lazy loading, dark theme, vector graphics, aggressive bundle cleansing, adaptive content delivery, and strict weight control constitute the durable and inclusive digital environment that simultaneously achieves economic, technical, and social objectives. This comprehensive approach confers competitive advantages to companies and advances global efforts toward decarbonization and reducing digital inequality.

REFERENCES

"Measuring the Emissions & Energy Footprint of the ICT Sector," World Bank, 2024. Accessed: Apr. 11, 2025. [Online]. Available: https://documents1.worldbank.org/curated/en/099121223165540890/pdf/P17859702a98880540a4b70d57876048abb.pdf

"AI is set to drive surging electricity demand from data

centres while offering the potential to transform how the energy sector works," *IEA*, Apr. 10, 2025—https://www.iea.org/news/ai-is-set-to-drive-surging-electricity-demand-from-data-centres-while-offering-the-potential-to-transform-how-the-energy-sector-works (accessed Apr. 12, 2025).

M. Wiegand, "Site Speed is (Still) Impacting Your Conversion Rate," *Portent*, Apr. 20, 2022. https://portent.com/blog/analytics/research-site-speed-hurting-everyones-revenue.htm (accessed Apr. 13, 2025).

M. Bervell, "The Business Case for Digital Accessibility: A Revenue-Generating Investment," *Test Party*, Mar. 05, 2025. https://testparty.ai/blog/the-business-case-for-digital-accessibility (accessed Apr. 14, 2025).

S. Judis and E. Portis, "Media," *The Web Almanac by HTTP Archive*, Dec. 2024, Accessed: Apr. 15, 2025. [Online]. Available: https://almanac.httparchive.org/en/2024/media

Vasile Crudu, "Why You Should Use Lazy Loading Alongside Minification in WordPress for Optimal Performance," *MoldStud*, Apr. 16, 2025. https://moldstud.com/articles/p-why-you-should-use-lazy-loading-alongside-minification-in-wordpress-for-optimal-performance (accessed Apr. 17, 2025).

J. McAlister, "How we made our website use 50% less energy," *Fast Familiar Workroom*, Mar. 23, 2021. https://workroom.fastfamiliar.com/how-we-made-our-website-use-50-less-energy/ (accessed Apr. 16, 2025).

Y. Charlie Hu, "Shedding light on dark mode to save energy," *Purdue University*, 2022. https://engineering.purdue.edu/ECE/News/2022/shedding-light-on-dark-mode-to-save-energy (accessed Apr. 15, 2025).

M. Turner, "Popular 'power saving' mobile feature is sucking your battery," *The Sun*, Feb. 20, 2025. https://www.thesun.co.uk/tech/33472529/power-saving-mobile-dark-mode-sucking-battery/ (accessed Apr. 16, 2025).

Princewillton, "Dark Mode Usage Statistics," *Helpfultech*, Apr. 28, 2024. https://helpfultech.net/dark-mode-usage-statistics-2024.html (accessed Apr. 17, 2025).

N. Palombi, "SVG files: what are they and how to use them?," *Webflow*, 2025. https://webflow.com/blog/svg-file (accessed Apr. 17, 2025).

"A Developer's Guide to SVG Optimization," *Cloudinary*, Jan. 15, 2025. https://cloudinary.com/guides/image-formats/a-developers-guide-to-svg-optimization (accessed Apr. 17, 2025).

J. Wagner, "Reduce JavaScript payloads with tree shaking," *web.dev*, 2018. https://web.dev/articles/reduce-javascript-payloads-with-tree-shaking (accessed Apr. 19, 2025).

"How We Reduced Our JavaScript Bundle Size by 70% While Adding Features," *Medium*, Apr. 21, 2025. https://javascript.plainenglish.io/how-we-reduced-our-javascript-bundle-size-by-70-while-adding-features-cb784a948631 (accessed Apr. 20, 2025).

"The most effective ways to improve Core Web Vitals," *web.dev*, 2024. https://web.dev/articles/top-cwv (accessed Apr. 20, 2025).

Maharaf Hossen, "Next.js vs React: Why Next.js is the Future of Web Development," *Medium*, Feb. 24, 2025. https://maharafhossen.medium.com/next-js-vs-react-why-next-js-is-the-future-of-web-development-60f7d8060fea (accessed Apr. 21, 2025).

Google, "Mobile Site Abandonment after Delayed Load Time," *Think with Google*. https://www.thinkwithgoogle.com/consumer-insights/consumer-trends/mobile-site-load-time-statistics/ (accessed Apr. 21, 2025).

M. Mihajlija, "Adaptive loading: improving web performance on slow devices," *web.dev*. https://web.dev/articles/adaptive-loading-cds-2019 (accessed Apr. 22, 2025).

Laurent Devernay Satyagraha, Burak Güneli, I. Akrap, A. Dawson, M. Gifford, and T. Frick, "Sustainability," *The Web Almanac by HTTP Archive*, Nov. 2024, Accessed: Apr. 24, 2025. [Online]. Available:

https://almanac.httparchive.org/en/2024/sustainability

"How is your website impacting the planet?" *Wholegrain Digital*. https://www.websitecarbon.com/ (accessed Apr. 24, 2025).

"Chrome Lite Pages - For a faster, leaner loading experience," *Chromium Blog*, Mar. 12, 2019. https://blog.chromium.org/2019/03/chrome-lite-pages-for-faster-leaner.html (accessed May 01, 2025).

J. Morehead, "Introducing Google News Lite mode — faster news for slower networks," *Google*, Sep. 27, 2016. https://blog.google/outreach-initiatives/google-news-initiative/introducing-google-news-lite-mode/ (accessed May 03, 2025).

C. Welch, "Google is ditching Chrome's data saver mode on Android," *The Verge*, Feb. 23, 2022. https://www.theverge.com/2022/2/23/22947441/google-lite-mode-data-saver-chrome-android-discontinued (accessed May 06, 2025)