Guidance to build modern web experiences that work on any browser.

## Docs

- (https://web.dev/articles.md.txt): Guidance to build modern web experiences that work on any browser.
- [3D and CSS](https://web.dev/articles/3d-css.md.txt): Guidance to build modern web experiences that work on any browser.
- [5 CSS snippets every front-end developer should know in 2024](https://web.dev/articles/5-css-snippets-every-front-end-developer-should-know-in-2024.md.txt): Toolbelt worthy, powerful, and stable CSS you can use today.
- [6 CSS snippets every front-end developer should know in 2023](https://web.dev/articles/6-css-snippets-every-front-end-developer-should-know-in-2023.md.txt): Toolbelt worthy, powerful, and stable CSS you can use today.
- [Accessibility for teams](https://web.dev/articles/a11y-for-teams.md.txt): How to incorporate accessibility into your team&#39;s process.
- [Accessibility for web developers](https://web.dev/articles/a11y-tips-for-web-dev.md.txt): It&#39;s important to build sites that are inclusive and accessible to everyone. There are at least six key areas of disability we can optimize for: visual, hearing, mobility, cognition, speech and neural.
- [Profiling your WebGL Game with the about:tracing flag](https://web.dev/articles/abouttracing.md.txt): Guidance to build modern web experiences that work on any browser.
- [CSS accent-color](https://web.dev/articles/accent-color.md.txt): Bring your brand color to built-in HTML form inputs with one line of code.
- [Accessibility](https://web.dev/articles/accessibility.md.txt): Improving accessibility for web pages
- [Accessibility audit with react-axe and eslint-plugin-jsx-a11y](https://web.dev/articles/accessibility-auditing-react.md.txt): react-axe is a library that audits a React application and logs any accessibility issues to the Chrome DevTools console. eslint-plugin-jsx-a11y is an ESLint plugin that identifies and enforces a number of accessibility rules directly in your JSX. Using them in combination can provide a comprehensive auditing approach to find and fix any accessibility concerns in your application.
- [Audit your Angular app&#39;s accessibility with codelyzer](https://web.dev/articles/accessible-angular-with-codelyzer.md.txt): Learn how to make your Angular application accessible using codelyzer.
- [Accessible responsive design](https://web.dev/articles/accessible-responsive-design.md.txt): We know that it&#39;s a good idea to design responsively to provide the best multi-device experience, but responsive design also yields a win for accessibility.
- [Accessible tap targets](https://web.dev/articles/accessible-tap-targets.md.txt): It&#39;s important that interactive elements have enough space around them, when used on a mobile or touchscreen device. This helps everyone, and especially those with motor impairments.
- [Adapting typography to user preferences with CSS](https://web.dev/articles/adapting-typography-to-user-preferences-with-css.md.txt): A method to adapt a font to your users&#39; preferences, so they&#39;re maximally comfortable reading your content.
- [Adaptive loading: improving web performance on slow devices](https://web.dev/articles/adaptive-loading-cds-2019.md.txt): Learn about adaptive loading pattern, how to implement it, and how Facebook, Tinder, eBay, and other companies use adaptive loading in production.
- [Adaptive serving based on network quality](https://web.dev/articles/adaptive-serving-based-on-network-quality.md.txt): Use Network Information API to adapt the assets served to users based on the quality of their connection.
- [Add a web app manifest](https://web.dev/articles/add-manifest.md.txt): The web app manifest is a simple JSON file that tells the browser how your web app should behave.
- [Add a web app manifest with Create React App](https://web.dev/articles/add-manifest-react.md.txt): Create React App includes a web app manifest by default. Modifying this file will allow you to change how your application is displayed when installed on the user&#39;s device.
- [Add media to a web page](https://web.dev/articles/add-media.md.txt): Embed a media file in a web page using the and tags.
- [Add touch to your site](https://web.dev/articles/add-touch-to-your-site.md.txt): Touchscreens are available on more and more devices, from phones up to desktop screens. Your app should

respond to their touch in intuitive and beautiful ways.
- [Additional HTML elements](https://web.dev/articles/additional-html-elements.md.txt): Essential metrics for a healthy site
- [Advancing the web framework ecosystem](https://web.dev/articles/advancing-framework-ecosystem-cds-2019.md.txt): Learn about how Chrome is investing efforts in a number of open-source tools to advance the JavaScript ecosystem
- [Build a local and offline-capable chatbot with the Prompt API](https://web.dev/articles/ai-chatbot-promptapi.md.txt): Guidance to build modern web experiences that work on any browser.
- [Build a local and offline-capable chatbot with WebLLM](https://web.dev/articles/ai-chatbot-webllm.md.txt): Guidance to build modern web experiences that work on any browser.
- [Part 2: Build client-side AI toxicity detection](https://web.dev/articles/ai-detect-toxicity-build.md.txt): Toxicity detection protects your users and creates a safer online environment. In part two, we learn how to build a client-side AI tool to detect and mitigate toxicity at its source.
- [Part 1: Client-side AI to combat online toxicity](https://web.dev/articles/ai-detect-toxicity-context.md.txt): &#34;Protect your users and create a safer online environment with toxicity detection. In part one, we share the context you need to deploy AI to mitigate toxicity at its source: users&#39; keyboards.&#34;
- [Ethics and AI](https://web.dev/articles/ai-ethics.md.txt): There are numerous ethical considerations when using AI tools and generating new content.
- [Benefits and limits of large language models](https://web.dev/articles/ai-llms-benefits.md.txt): Learn about the benefits and drawbacks of on-device and client-side large language models.
- [What is artificial intelligence?](https://web.dev/articles/ai-overview.md.txt): Understand the difference between client-side AI and server-side AI, machine learning and deep learning, and so much more.
- [Upgrade your site search: Contextual answers with generative AI](https://web.dev/articles/ai-rag-search.md.txt): There are numerous ethical considerations when using AI tools and generating new content.
- [Meet the Web.dev AI Team](https://web.dev/articles/ai-team.md.txt): Meet the tech writers and developer relations team.
- [Providing shipping and contact information from an Android payment app](https://web.dev/articles/android-payment-apps-delegation.md.txt): Learn how to modify your Android payment app to provide the user&#39;s selected shipping address as well as contact information when the merchant has requested them using the Payment Request API.
- [Android payment app developers guide](https://web.dev/articles/android-payment-apps-developers-guide.md.txt): Learn how to adapt your Android payment app to work with Web Payments and provide a better user experience for customers.
- [Writing an AngularJS App with Socket.IO](https://web.dev/articles/angular-websockets.md.txt): Guidance to build modern web experiences that work on any browser.
- [Animating Between Views](https://web.dev/articles/animating-between-views.md.txt): Learn how to animate between two views in your apps.
- [Animating Modal Views](https://web.dev/articles/animating-modal-views.md.txt): Learn how to animate modal views in your apps.
- [Animations and performance](https://web.dev/articles/animations-and-performance.md.txt): Animations must perform well, otherwise they will negatively impact the user experience.
- [Examples of high-performance CSS animations](https://web.dev/articles/animations-examples.md.txt): Demonstrating how high performance techniques can create complex and beautiful animations.
- [How to create high-performance CSS animations](https://web.dev/articles/animations-guide.md.txt): To improve the performance of your CSS animations, use the transform and opacity CSS properties as much as possible, and avoid anything that triggers layout or painting.
- [Why are some animations slow?](https://web.dev/articles/animations-overview.md.txt): Animating well is core to a great web experience. This post explains why certain types of animation perform better than others.
- [Antialiasing 101](https://web.dev/articles/antialiasing-101.md.txt): Guidance to build modern web experiences that work on any browser.
- [An API for fast, beautiful web fonts](https://web.dev/articles/api-for-fast-beautiful-web-fonts.md.txt): An update on the Google Fonts CSS API—how it works, how to use it, and how it can efficiently deliver your web fonts.
- [Make your PWA feel more like an app](https://web.dev/articles/app-like-pwas.md.txt): Learn how to make your Progressive Web App feel like a &#34;real&#34; app by understanding how to

implement platform-specific app patterns with web technologies.
- [Get things done quickly with app shortcuts](https://web.dev/articles/app-shortcuts.md.txt): App shortcuts give quick access to a handful of common actions that users need frequently.
- [A beginner&#39;s guide to using the application cache](https://web.dev/articles/appcache-beginner.md.txt): Guidance to build modern web experiences that work on any browser.
- [Preparing for AppCache removal](https://web.dev/articles/appcache-removal.md.txt): Details of Chrome&#39;s and other browsers&#39; plans to remove AppCache.
- [Apply instant loading with the PRPL pattern](https://web.dev/articles/apply-instant-loading-with-prpl.md.txt): PRPL is an acronym that describes a pattern used to make web pages load and become interactive, faster. In this guide, learn how each of these techniques fit together but still can be used independently to achieve performance results.
- [Positioning virtual objects in real-world views](https://web.dev/articles/ar-hit-test.md.txt): The WebXR Hit Test API is an enhancement to the web&#39;s augmented reality that lets you place virtual objects in a real-world view.
- [ARIA labels and relationships](https://web.dev/articles/aria-labels-and-relationships.md.txt): Using ARIA labels to create accessible element descriptions
- [ARIA: poison or antidote?](https://web.dev/articles/aria-poison-or-antidote.md.txt): How lying to screen readers cures accessibility, when it doesn&#39;t rub salt in it!
- [The CSS aspect-ratio property](https://web.dev/articles/aspect-ratio.md.txt): Maintaining aspect ratio within images and elements is now easier to achieve with the new aspect-ratio CSS property.
- [Assess spam damage](https://web.dev/articles/assess-spam-damage.md.txt): Find out what to do if your site has been hacked to host spam.
- [Simple asset management for HTML5 games](https://web.dev/articles/assetmanager.md.txt): Guidance to build modern web experiences that work on any browser.
- [Asymmetric animation timing](https://web.dev/articles/asymmetric-animation-timing.md.txt): Breaking symmetry provides contrast and appeal to your projects. Learn when and how to apply this to your projects.
- [Unblocking clipboard access](https://web.dev/articles/async-clipboard.md.txt): Async Clipboard API simplifies permissions-friendly copy and paste.
- [Asynch JS - The power of $.Deferred](https://web.dev/articles/async-deferred.md.txt): Guidance to build modern web experiences that work on any browser.
- [Async functions: making promises friendly](https://web.dev/articles/async-functions.md.txt): Async functions allow you to write promise-based code as if it were synchronous.
- [Using asynchronous web APIs from WebAssembly](https://web.dev/articles/asyncify.md.txt): Learn how to invoke asynchronous web APIs when compiling traditionally synchronous languages to WebAssembly.
- [@property: giving superpowers to CSS variables](https://web.dev/articles/at-property.md.txt): Learn how to implement CSS custom properties with semantic typing, a fallback value, and more, directly in your CSS file.
- [Synchronize audio and video playback on the web](https://web.dev/articles/audio-output-latency.md.txt): The Web Audio API makes it possible to properly achieve AV synchronization.
- [Quick guide to implementing the HTML5 audio tag](https://web.dev/articles/audio-quick.md.txt): Guidance to build modern web experiences that work on any browser.
- [A tale of two clocks](https://web.dev/articles/audio-scheduling.md.txt): Guidance to build modern web experiences that work on any browser.
- [Measure browser autofill on your forms](https://web.dev/articles/autofill-measure.md.txt): To optimize user experience, it&#39;s essential to understand how users interact with your forms. Browser autofill plays a significant role in this process. Learn how to collect and analyze data on how users use autofill in your forms.
- [Automating audits with AutoWebPerf](https://web.dev/articles/autowebperf.md.txt): A new modular tool that enables automatic gathering of performance data from multiple sources.
- [Deploying AVIF for more responsive websites](https://web.dev/articles/avif-updates-2023.md.txt): An overview of how AVIF is adopted in the ecosystem, and what kind of performance and quality benefits developers can expect from AVIF for still images and animations.
- [Avoid large, complex layouts and layout thrashing](https://web.dev/articles/avoid-large-complex-layouts-and-layout-thrashing.md.txt): Layout is where the browser figures out the geometric information for elements - their size and location in the page. Each element will have explicit or implicit sizing information based on the CSS that was used, the contents of the element, or a parent element. The process is called Layout in Chrome.
- [Create OS-style backgrounds with backdrop-filter](https://web.dev/articles/backdrop-filter.md.txt): Learn how to add background effects like blurring and transparency to UI

elements on the web using the CSS backdrop-filter property.
- [Download AI models with the Background Fetch API](https://web.dev/articles/background-fetch-ai.md.txt): Guidance to build modern web experiences that work on any browser.
- [The nuances of base64 encoding strings in JavaScript](https://web.dev/articles/base64-encoding.md.txt): Understand and avoid common problems when applying base64 encoding and decoding to strings.
- [How to think about Baseline and polyfills](https://web.dev/articles/baseline-and-polyfills.md.txt): Knowing when to reach for a polyfill can be a hard decision, but using Baseline features can help you answer this question. Learn more in this guide!
- [Create Baseline tools with the web-features package](https://web.dev/articles/baseline-tools-web-features.md.txt): Learn how to use the data in the web-features npm package to create your own Baseline tools!
- [Feedback wanted: The road to a better layout shift metric for long-lived pages](https://web.dev/articles/better-layout-shift-metric.md.txt): Learn about our plans for improving the Cumulative Layout Shift metric and give us feedback.
- [Back/forward cache](https://web.dev/articles/bfcache.md.txt): Learn to optimize your pages for instant loads when using the browser&#39;s back and forward buttons.
- [Compiling to and optimizing Wasm with Binaryen](https://web.dev/articles/binaryen.md.txt): Using the example of a synthetic toy language called ExampleScript, learn how to write and optimize WebAssembly modules in JavaScript using the Binaryen.js API.
- [Bitrate](https://web.dev/articles/bitrate.md.txt): Bitrate is the maximum number of bits used to encode one second of a stream. The more bits used to encode a second of stream, the higher the fidelity.
- [Bridging the gap](https://web.dev/articles/bridging-the-gap.md.txt): Making it easier to build for the web.
- [Broadcast updates to pages with service workers](https://web.dev/articles/broadcast-updates-guide.md.txt): How service workers can proactively communicate with the page to inform about certain events.
- [Browser-level image lazy loading for the web](https://web.dev/articles/browser-level-image-lazy-loading.md.txt): This post covers the loading attribute and how it can be used to control the loading of images.
- [Browser-level lazy loading for CMSs](https://web.dev/articles/browser-level-lazy-loading-for-cmss.md.txt): This post provides guidance on how to adopt the loading attribute in content management systems.
- [Browser sandbox](https://web.dev/articles/browser-sandbox.md.txt): To defend against attacks, a developer needs to mitigate vulnerabilities and add security features to an application. Luckily, on the web, the browser provides many security features, including the idea of a &#34;sandbox&#34;.
- [Build a support team](https://web.dev/articles/build-a-support-team.md.txt): Work with your site hoster to deal with spam damage, and find online communities to help.
- [Building a PWA at Google, part 1](https://web.dev/articles/building-a-pwa-at-google-part-1.md.txt): What the Bulletin team learned about service workers while developing a PWA.
- [Building Chrometober!](https://web.dev/articles/building-chrometober.md.txt): How the scrolling book came to life for sharing fun and frightening tips and tricks this Chrometober.
- [Building multiple Progressive Web Apps on the same domain](https://web.dev/articles/building-multiple-pwas-on-the-same-domain.md.txt): Explore the recommended and not recommended ways of building multiple PWAs reusing the same domain with their pros and cons.
- [Building a 3D game menu component](https://web.dev/articles/building/a-3d-game-menu-component.md.txt): A foundational overview of how to build a responsive, adaptive, and accessible 3D game menu.
- [Building a breadcrumbs component](https://web.dev/articles/building/a-breadcrumbs-component.md.txt): A foundational overview of how to build a responsive and accessible breadcrumbs component for users to navigate your site.
- [Building a button component](https://web.dev/articles/building/a-button-component.md.txt): A foundational overview of how to build color-adaptive, responsive, and accessible components.
- [Building a color scheme](https://web.dev/articles/building/a-color-scheme.md.txt): A foundational overview of how to establish a dynamic and configurable color scheme
- [Building a crooked grid illusion](https://web.dev/articles/building/a-crooked-grid-illusion.md.txt): A fun exploration of ways to recreate an optical illusion with CSS.
- [Building a dialog component](https://web.dev/articles/building/a-dialog-component.md.txt): A foundational overview of how to build color-adaptive, responsive, and accessible mini and mega modals with the `` element.

- [Building a floating action button (FAB) component](https://web.dev/articles/building/a-fab-component.md.txt): A foundational overview of how to build color-adaptive, responsive, and accessible FAB components.
- [Building a loading bar component](https://web.dev/articles/building/a-loading-bar-component.md.txt): A foundational overview of how to build a color adaptive and accessible loading bar with the `` element.
- [Building a media scroller component](https://web.dev/articles/building/a-media-scroller-component.md.txt): A foundational overview of how to build a responsive horizontal scrollview for TVs, phones, desktops, etc.
- [Building a multi-select component](https://web.dev/articles/building/a-multi-select-component.md.txt): A foundational overview of how to build a responsive, adaptive, and accessible, multiselect component for sort and filter user experiences.
- [Building a Settings component](https://web.dev/articles/building/a-settings-component.md.txt): A foundational overview of how to build a settings component of sliders and checkboxes.
- [Building a sidenav component](https://web.dev/articles/building/a-sidenav-component.md.txt): A foundational overview of how to build a responsive slide out sidenav
- [Building a split-button component](https://web.dev/articles/building/a-split-button-component.md.txt): A foundational overview of how to build an accessible split-button component.
- [Building a Stories component](https://web.dev/articles/building/a-stories-component.md.txt): A foundational overview of how to build an experience similar to Instagram Stories on the web.
- [Building a switch component](https://web.dev/articles/building/a-switch-component.md.txt): A foundational overview of how to build a responsive and accessible switch component.
- [Building a Tabs component](https://web.dev/articles/building/a-tabs-component.md.txt): A foundational overview of how to build a tabs component similar to those found in iOS and Android apps.
- [Building a theme switch component](https://web.dev/articles/building/a-theme-switch-component.md.txt): A foundational overview of how to build an adaptive and accessible theme switch component.
- [Building a toast component](https://web.dev/articles/building/a-toast-component.md.txt): A foundational overview of how to build an adaptive and accessible toast component.
- [Building a tooltip component](https://web.dev/articles/building/a-tooltip-component.md.txt): A foundational overview of how to build a color-adaptive and accessible tooltip custom element.
- [Building an adaptive favicon](https://web.dev/articles/building/an-adaptive-favicon.md.txt): A foundational overview of how to build an adaptive favicon.
- [Building split text animations](https://web.dev/articles/building/split-text-animations.md.txt): A foundational overview of how to build split letter and word animations.
- [Bundling non-JavaScript resources](https://web.dev/articles/bundling-non-js-resources.md.txt): Learn how to import and bundle various types of assets from JavaScript in a way that works both in browsers and bundlers.
- [The Cache API: A quick guide](https://web.dev/articles/cache-api-quick-guide.md.txt): Learn how to use the Cache API to make your application data available offline.
- [Control camera pan, tilt, and zoom](https://web.dev/articles/camera-pan-tilt-zoom.md.txt): Pan, tilt, and zoom features on cameras are finally controllable on the web.
- [High DPI Canvas](https://web.dev/articles/canvas-hidipi.md.txt): Guidance to build modern web experiences that work on any browser.
- [Image filters with canvas](https://web.dev/articles/canvas-imagefilters.md.txt): Guidance to build modern web experiences that work on any browser.
- [Canvas inspection using Chrome DevTools](https://web.dev/articles/canvas-inspection.md.txt): Guidance to build modern web experiences that work on any browser.
- [Integrating Canvas into your Web App](https://web.dev/articles/canvas-integrating.md.txt): Guidance to build modern web experiences that work on any browser.
- [No tears guide to HTML5 games](https://web.dev/articles/canvas-notearsgame.md.txt): Guidance to build modern web experiences that work on any browser.
- [Improving HTML5 Canvas performance](https://web.dev/articles/canvas-performance.md.txt): Guidance to build modern web experiences that work on any browser.
- [Typographic effects in canvas](https://web.dev/articles/canvas-texteffects.md.txt): Guidance to build modern web experiences that work on any browser.
- [Best practices for carousels](https://web.dev/articles/carousel-best-practices.md.txt): Learn how to optimize carousels for performance and usability.
- [Everything announced at Chrome Dev Summit 2021](https://web.dev/articles/cds2021-updates.md.txt): A roundup of all the key announcements from the 2021 Chrome Dev Summit, with

the links you need to find out more.
- [Centering in CSS](https://web.dev/articles/centering-in-css.md.txt): Follow 5 centering techniques as they go through a series of tests to see which one is the most resilient to change.
- [Help users change passwords easily by adding a well-known URL for changing passwords](https://web.dev/articles/change-password-url.md.txt): By redirecting requests to /.well-known/change-password to the change password URL, you can let users update their passwords easier than before.
- [Choose a JavaScript library or framework](https://web.dev/articles/choose-js-library-or-framework.md.txt): Understand the decisions around using a JavaScript library or framework.
- [Choose the right image format](https://web.dev/articles/choose-the-right-image-format.md.txt): Selecting the right image format is the first step in delivering optimized images on your website. This post helps you to make the right choice.
- [Choosing the right easing](https://web.dev/articles/choosing-the-right-easing.md.txt): Choose the appropriate easing for your project, whether that&#39;s easing in, out, or both. Maybe even use bounces for extra fun!
- [Chrome and Firefox soon to reach major version 100](https://web.dev/articles/chrome-firefox-100.md.txt): User-Agent string changes, the strategies that Chrome and Firefox are taking to mitigate the impact, and how you can help.
- [Clean and maintain your site](https://web.dev/articles/clean-and-maintain-your-site.md.txt): This post explains how to clean up your site after it has been hacked.
- [Click to Call](https://web.dev/articles/click-to-call.md.txt): On devices with phone capabilities, make it easy for users to directly connect with you by simply tapping a phone number, more commonly known as click to call.
- [Improve performance and UX for client-side AI](https://web.dev/articles/client-side-ai-performance.md.txt): Discover the benefits of client-side AI, including low latency, reduced server-side costs, no API key requirements, increased user privacy, and offline access.
- [Client-side rendering of HTML and interactivity](https://web.dev/articles/client-side-rendering-of-html-and-interactivity.md.txt): Rendering HTML with JavaScript is different than rendering HTML that&#39;s sent by the server—and that can affect performance. Learn the difference in this guide, and what you can do to preserve your website&#39;s rendering performance—especially where interactions are concerned.
- [Cumulative Layout Shift (CLS)](https://web.dev/articles/cls.md.txt): This post introduces the Cumulative Layout Shift (CLS) metric and explains how to measure it.
- [Code splitting with React.lazy and Suspense](https://web.dev/articles/code-splitting-suspense.md.txt): The React.lazy method makes it easy to code-split a React application on a component level using dynamic imports. Use it along with Suspense to show appropriate loading states to your users.
- [Code splitting with dynamic imports in Next.js](https://web.dev/articles/code-splitting-with-dynamic-imports-in-nextjs.md.txt): Guidance to build modern web experiences that work on any browser.
- [Adapt video to image serving based on network quality](https://web.dev/articles/codelab-adapt-video-to-image-serving-based-on-network-quality.md.txt): Learn how to use the Network Information API to adapt your content based on the network quality.
- [Art direction](https://web.dev/articles/codelab-art-direction.md.txt): In this codelab, learn how to load completely different images based on device display size.
- [Codelab: Building a Sidenav component](https://web.dev/articles/codelab-building-a-sidenav-component.md.txt): Learn how to build a responsive slide out side navigation layout component.
- [Building resilient search experiences with Workbox](https://web.dev/articles/codelab-building-resilient-search-experiences.md.txt): How to implement a resilient search experience with Workbox, using Background Sync and the Push API.
- [Codelab: Centering in CSS](https://web.dev/articles/codelab-centering-in-css.md.txt): Learn 5 different centering techniques with CSS.
- [Reduce JavaScript payloads with code splitting](https://web.dev/articles/codelab-code-splitting.md.txt): In this codelab, learn how to improve the performance of a simple application through code splitting.
- [Use density descriptors](https://web.dev/articles/codelab-density-descriptors.md.txt): In this codelab, learn how to use density descriptors and srcset to load images with the right pixel density for the user&#39;s device.
- [Explore DevTools Network panel](https://web.dev/articles/codelab-explore-network-panel.md.txt): In this codelab, learn how to inerpret network traffic using Chrome&#39;s DevTools.

- [Extract and inline critical CSS with Critical](https://web.dev/articles/codelab-extract-and-inline-critical-css.md.txt): Learn how use Critical to extract and inline critical CSS, and improve render times.
- [Fix sneaky 404s](https://web.dev/articles/codelab-fix-sneaky-404.md.txt): In this codelab, learn how to track down a sneaky 404 that may prevent your page from being properly indexed.
- [Configuring HTTP caching behavior](https://web.dev/articles/codelab-http-cache.md.txt): In this codelab, learn how to control resource caching behavior using HTTP headers.
- [Using Imagemin with webpack](https://web.dev/articles/codelab-imagemin-webpack.md.txt): In this codelab, learn how to use imagemin with webpack to optimize JPEG and PNG images for faster download.
- [Make it installable](https://web.dev/articles/codelab-make-installable.md.txt): In this codelab, learn how to make a site installable using the beforeinstallprompt event.
- [Get started with the Notifications API](https://web.dev/articles/codelab-notifications-get-started.md.txt): In this codelab, learn how to request user permission and send notifications.
- [Build a push notifications server](https://web.dev/articles/codelab-notifications-push-server.md.txt): In this codelab, learn how to build a push notifications server.
- [Use a Service Worker to manage notifications](https://web.dev/articles/codelab-notifications-service-worker.md.txt): In this codelab, learn how to manage notifications with a service worker.
- [Optimize third-party JavaScript](https://web.dev/articles/codelab-optimize-third-party-javascript.md.txt): Learn about techniques for optimizing slow third-party resources with some help from Lighthouse.
- [Payment form best practices codelab](https://web.dev/articles/codelab-payment-form-best-practices.md.txt): Learn best practices for payment forms.
- [Codelab: Preload critical assets to improve loading speed](https://web.dev/articles/codelab-preload-critical-assets.md.txt): In this codelab, learn how to improve the performance of a page by preloading and prefetching resources.
- [Preload web fonts to improve loading speed](https://web.dev/articles/codelab-preload-web-fonts.md.txt): In this codelab, learn how to improve the performance of a page by preloading web fonts.
- [Remove unused code](https://web.dev/articles/codelab-remove-unused-code.md.txt): In this codelab, learn how to improve the performance of an application by removing any unused and unneeded dependencies.
- [Replace GIFs with video](https://web.dev/articles/codelab-replace-gifs-with-video.md.txt): In this codelab, learn how to improve performance by replacing an animated GIF with a video.
- [Same Origin Policy &amp; Fetch requests](https://web.dev/articles/codelab-same-origin-fetch.md.txt): In this codelab, learn how the same-origin policy works when fetching resources.
- [Same Origin Policy &amp; iframe](https://web.dev/articles/codelab-same-origin-iframe.md.txt): In this codelab, learn how the same-origin policy works when accessing data inside an iframe.
- [Serve images with correct dimensions](https://web.dev/articles/codelab-serve-images-correct-dimensions.md.txt): In this codelab, learn how to serve images with the correct dimensions to improve network performance.
- [Creating WebP Images with the Command Line](https://web.dev/articles/codelab-serve-images-webp.md.txt): In this codelab, learn how to serve optimized images using WebP.
- [Serve modern code to modern browsers for faster page loads](https://web.dev/articles/codelab-serve-modern-code.md.txt): In this codelab, learn how to improve the performance of an application by minizming how much code is transpiled.
- [Working with service workers](https://web.dev/articles/codelab-service-workers.md.txt): In this codelab, learn how to make an application reliable by registering a service worker.
- [Setting performance budgets with webpack](https://web.dev/articles/codelab-setting-performance-budgets-with-webpack.md.txt): Learn how to set performance budgets and keep your bundlesize in check with webpack.
- [Use cross-platform browser features to build a sign-in form](https://web.dev/articles/codelab-sign-in-form-best-practices.md.txt): Use cross-platform browser features to build a simple sign-in form that&#39;s secure, accessible, and easy to use.
- [Sign-up form best practices codelab](https://web.dev/articles/codelab-sign-up-form-best-practices.md.txt): Use cross-platform browser features to build a simple sign-up form that&#39;s secure, accessible, and easy to use.
- [Specifying multiple slot widths](https://web.dev/articles/codelab-specifying-multiple-slot-widths.md.txt): In this codelab, learn how to use the sizes attribute to size images correctly depending on the user&#39;s viewport.
- [Minify and compress network payloads with gzip](https://web.dev/articles/codelab-text-

compression.md.txt): In this codelab, learn how both minifying and compressing the JavaScript bundle for an application improves page performance by reducing the app&#39;s request size.
- [Minify and compress network payloads with brotli](https://web.dev/articles/codelab-text-compression-brotli.md.txt): In this codelab, learn how Brotli compression can further reduce compression ratios and your app&#39;s overall size.
- [Two ways to prefetch: &lt;link&gt; tags and HTTP headers](https://web.dev/articles/codelab-two-ways-to-prefetch.md.txt): Learn how to speed up future navigations by prefetching resources.
- [Color and contrast accessibility](https://web.dev/articles/color-and-contrast-accessibility.md.txt): Learn to make your pages more accessible by improving your color and contrast usage.
- [How CommonJS is making your bundles larger](https://web.dev/articles/commonjs-larger-bundles.md.txt): CommonJS modules are very dynamic, which prevents JavaScript optimizers and bundles perform advanced optimizations over them.
- [Google Community Guidelines and Anti-Harassment Policy for In-Person and Virtual Events](https://web.dev/articles/community-guidelines.md.txt): Community Guidelines
- [Compiling mkbitmap to WebAssembly](https://web.dev/articles/compiling-mkbitmap-to-webassembly.md.txt): The mkbitmap C program reads an image and applies one or more of the following operations to it, in this order: inversion, highpass filtering, scaling, and thresholding. Each operation can be individually controlled and turned on or off. This article shows how to compile mkbitmap to WebAssembly.
- [HowTo Components – Overview](https://web.dev/articles/components-examples-overview.md.txt): HowTo Components
- [HowTo Components – howto-checkbox](https://web.dev/articles/components-howto-checkbox.md.txt): Guidance to build modern web experiences that work on any browser.
- [HowTo Components – howto-tabs](https://web.dev/articles/components-howto-tabs.md.txt): Guidance to build modern web experiences that work on any browser.
- [HowTo Components – howto-tooltip](https://web.dev/articles/components-howto-tooltip.md.txt): Guidance to build modern web experiences that work on any browser.
- [Choose the correct level of compression](https://web.dev/articles/compress-images.md.txt): Many images can be heavily compressed, giving excellent performance improvements. This post helps you to choose the right level of compression to maintain the look of images while getting the best performance.
- [Using AVIF to compress images on your site](https://web.dev/articles/compress-images-avif.md.txt): Serving desktop-sized images to mobile devices can use 2–4x more data than needed. Instead of a &#34;one-size-fits-all&#34; approach to images, serve different image sizes to different devices.
- [Use conic gradients to create a cool border](https://web.dev/articles/conic-gradient-border.md.txt): Guidance to build modern web experiences that work on any browser.
- (https://web.dev/articles/constraintvalidation.md.txt): Guidance to build modern web experiences that work on any browser.
- [Constructable Stylesheets](https://web.dev/articles/constructable-stylesheets.md.txt): Constructable Stylesheets provide a seamless way to create and distribute styles to documents or shadow roots without worrying about FOUC.
- [Containers and codecs](https://web.dev/articles/containers-and-codecs.md.txt): Media files are a bit like onions. The file that you see in your operating system shell is only a container with multiple data streams and different allowable types of encodings.
- [Content delivery networks (CDNs)](https://web.dev/articles/content-delivery-networks.md.txt): This article provides a comprehensive overview of content delivery networks (CDNs). In addition, it explains how to choose, configure, and optimize a CDN setup.
- [Content reordering](https://web.dev/articles/content-reordering.md.txt): When creating a layout using CSS, ensure you give the same experience for keyboard users as other users.
- [content-visibility: the new CSS property that boosts your rendering performance](https://web.dev/articles/content-visibility.md.txt): The CSS content-visibility property enables web content rendering performance benefits by skipping rendering of off-screen content. This article shows you how to use this new CSS property for faster initial load times, using the auto keyword. You will also learn about the CSS Containment Spec and other values for content-visibility that give you more control over how your content renders in the browser.
- [Control focus with tabindex](https://web.dev/articles/control-focus-with-tabindex.md.txt): Standard HTML elements have keyboard accessibility built-in. When building custom interactive components, use tabindex to ensure that they&#39;re keyboard accessible.
- [Keeping third-party scripts under control](https://web.dev/articles/controlling-third-party-scripts.md.txt): Third-party scripts, or &#34;tags&#34; can be a source of performance problems

on your site, and therefore a target for optimization. However, before you start optimizing the tags you have added, make sure that you are not optimizing tags you don&#39;t even need. This article shows you how to assess requests for new tags, and manage and review existing ones.
- [Best practices for cookie notices](https://web.dev/articles/cookie-notice-best-practices.md.txt): Learn about how cookie notices affect performance, performance measurement, and UX.
- [Making your website &quot;cross-origin isolated&quot; using COOP and COEP](https://web.dev/articles/coop-coep.md.txt): Some web APIs increase the risk of side-channel attacks like Spectre. To mitigate that risk, browsers offer an opt-in-based isolated environment called cross-origin isolated. Use COOP and COEP to set up such an environment and enable powerful features like `SharedArrayBuffer`, `performance.measureUserAgentSpecificMemory()` or high resolution timer with better precision.
- [Ensure your website is available and usable for everyone during COVID-19](https://web.dev/articles/covid19.md.txt): How to ensure that the core functionality of your website is always available, accessible, secure, usable, discoverable, and fast.
- [Create a Progressive Web App with the Angular CLI](https://web.dev/articles/creating-pwa-with-angular-cli.md.txt): Learn how to create installable progressive Angular applications.
- [Creative list styling](https://web.dev/articles/creative-list-styling.md.txt): A look at some useful and creative ways to style a list.
- [Critical Rendering Path](https://web.dev/articles/critical-rendering-path.md.txt): Optimizing the critical rendering path refers to prioritizing the display of content that relates to the current user action.
- [Adding Interactivity with JavaScript](https://web.dev/articles/critical-rendering-path/adding-interactivity-with-javascript.md.txt): JavaScript allows us to modify just about every aspect of the page: content, styling, and its response to user interaction. However, JavaScript can also block DOM construction and delay when the page is rendered. To deliver optimal performance, make your JavaScript async and eliminate any unnecessary JavaScript from the critical rendering path.
- [Analyzing Critical Rendering Path Performance](https://web.dev/articles/critical-rendering-path/analyzing-crp.md.txt): Learn to identify and resolve critical rendering path performance bottlenecks.
- [Constructing the Object Model](https://web.dev/articles/critical-rendering-path/constructing-the-object-model.md.txt): Learn how the browser constructs the DOM and CSSOM trees.
- [Measure the Critical Rendering Path](https://web.dev/articles/critical-rendering-path/measure-crp.md.txt): Learn to measure the critical rendering path.
- [Optimizing the Critical Rendering Path](https://web.dev/articles/critical-rendering-path/optimizing-critical-rendering-path.md.txt): Learn the key factors in optimizing the critical rendering path.
- [PageSpeed Rules and Recommendations](https://web.dev/articles/critical-rendering-path/page-speed-rules-and-recommendations.md.txt): This guide examines PageSpeed Insights rules in context: what to pay attention to when optimizing the critical rendering path, and why.
- [Render Blocking CSS](https://web.dev/articles/critical-rendering-path/render-blocking-css.md.txt): By default CSS is treated as a render blocking resource. Learn how to prevent it from blocking rendering.
- [Render-tree Construction, Layout, and Paint](https://web.dev/articles/critical-rendering-path/render-tree-construction.md.txt): One critical step in the critical rendering path involves the construction of the render tree, performing layout operations to create a page from it, as well as painting the pixels of the resulting page to the screen.
- [Cross-Origin Resource Sharing (CORS)](https://web.dev/articles/cross-origin-resource-sharing.md.txt): The browser&#39;s same-origin policy blocks reading a resource from a different origin for security purposes. Enabling CORS lets the server tell the browser it can use an additional origin.
- [Why is CrUX data different from my RUM data?](https://web.dev/articles/crux-and-rum-differences.md.txt): Learn about reasons why RUM data can show different Core Web Vitals numbers from CrUX.
- [Content security policy](https://web.dev/articles/csp.md.txt): Content Security Policy can significantly reduce the risk and impact of cross-site scripting attacks in modern browsers.
- [CSS animated grid layouts](https://web.dev/articles/css-animated-grid-layouts.md.txt): In CSS Grid, the `grid-template-columns` and `grid-template-rows` properties allow you to define line names and track sizing of grid columns and rows, respectively. Supporting interpolation for these properties allows grid layouts to smoothly transition between states, instead of snapping at the halfway point of an animation or transition.

- [CSS border animations](https://web.dev/articles/css-border-animations.md.txt): Looking at several ways to animate a border in CSS
- [Create interesting image shapes with CSS&#39;s clip-path property](https://web.dev/articles/css-clipping.md.txt): Using clipping in CSS can help us move away from everything in our designs looking like a box. By using various basic shapes, or an SVG, you can create a clip path. Then cut away the parts of an element you don&#39;t want to show.
- [Quickly create nice CSS gradients with the CSS Gradient Creator](https://web.dev/articles/css-gradient-generator.md.txt): This tool by Josh W Comeau makes it super simple to create nice looking gradients.
- [Finer grained control over CSS transforms with individual transform properties](https://web.dev/articles/css-individual-transform-properties.md.txt): Learn how you can use the individual translate, rotate, and scale CSS properties to approach transforms in an intuitive way.
- [New CSS functional pseudo-class selectors :is() and :where()](https://web.dev/articles/css-is-and-where.md.txt): These seemingly small additions to CSS selector syntax are going to have a big impact.
- [Custom bullets with CSS ::marker](https://web.dev/articles/css-marker-pseudo-element.md.txt): Use CSS to to customize the color, size or type of numbers or bullets in `` or `` elements.
- [Apply effects to images with the CSS mask-image property](https://web.dev/articles/css-masking.md.txt): CSS masking gives you the option of using an image as a mask layer. This means that you can use an image, an SVG, or a gradient as your mask, to create interesting effects without an image editor.
- [Using CSS Module Scripts to import stylesheets](https://web.dev/articles/css-module-scripts.md.txt): Learn how to use CSS module scripts to import CSS stylesheets using the same syntax as JavaScript modules.
- [CSS paint times and page render weight](https://web.dev/articles/css-paint-times.md.txt): Guidance to build modern web experiences that work on any browser.
- [Smarter custom properties with Houdini’s new API](https://web.dev/articles/css-props-and-vals.md.txt): Though useful, CSS variables are hard to work with because they can take any value and be overridden and you can’t use transitions with them. CSS Properties and Values API Level 1 overcomes these issues.
- [Magazine-like layout for the web with CSS regions and exclusions](https://web.dev/articles/css-regions-exclusions.md.txt): Guidance to build modern web experiences that work on any browser.
- [Well-controlled scrolling with CSS Scroll Snap](https://web.dev/articles/css-scroll-snap.md.txt): CSS Scroll Snap allows web developers to create well-controlled scroll experiences by declaring scroll snapping positions. This enables common UX scroll patterns without the need for JavaScript.
- [CSS size-adjust for @font-face](https://web.dev/articles/css-size-adjust.md.txt): As a web font loads, you can now adjust its scale to normalize the document font sizes and prevent layout shift when switching between fonts
- [Trigonometric functions in CSS](https://web.dev/articles/css-trig-functions.md.txt): Calculate the sine, cosine, tangent, and more in CSS.
- [CSS versus JavaScript animations](https://web.dev/articles/css-vs-javascript.md.txt): You can animate with CSS or JavaScript. Which should you use, and why?
- [CSS for Web Vitals](https://web.dev/articles/css-web-vitals.md.txt): This article covers CSS-related techniques for optimizing Web Vitals.
- [Custom easing](https://web.dev/articles/custom-easing.md.txt): Go offroad and create totally custom animations for your projects.
- [Custom Element Best Practices](https://web.dev/articles/custom-elements-best-practices.md.txt): Custom elements let you construct your own HTML tags. This checklist covers best practices to help you build high quality elements.
- [Custom Elements v1 - Reusable Web Components](https://web.dev/articles/custom-elements-v1.md.txt): Custom elements allow web developers to define new HTML tags, extend existing ones, and create reusable web components.
- [Custom metrics](https://web.dev/articles/custom-metrics.md.txt): Custom metrics let you measure and optimize aspects of your site&#39;s experience that are unique to your site.
- [How Nordhealth uses Custom Properties in Web Components](https://web.dev/articles/custom-properties-web-components.md.txt): The benefits of using Custom Properties in design systems and component libraries.
- [Working with Custom Elements](https://web.dev/articles/customelements.md.txt): Guidance to build modern web experiences that work on any browser.

- [Correlating Core Web Vitals and ad revenue with Google tools](https://web.dev/articles/cwv-impact-ad-revenue.md.txt): Learn how you can use Google tools to help correlate your Core Web Vitals with ad revenue.
- [Debounce your input handlers](https://web.dev/articles/debounce-your-input-handlers.md.txt): Input handlers are a potential cause of performance problems in your apps, as they can block frames from completing, and can cause additional and unnecessary layout work.
- [Debug layout shifts](https://web.dev/articles/debug-layout-shifts.md.txt): Learn how to identify and fix layout shifts.
- [Debug performance in the field](https://web.dev/articles/debug-performance-in-the-field.md.txt): Learn how to attribute your performance data with debug information to help you identify and fix real-user issues with analytics
- [Debug media playback errors on the web](https://web.dev/articles/debug-playback-errors.md.txt): Learn how to debug media playback errors on the web.
- [Declarative Shadow DOM](https://web.dev/articles/declarative-shadow-dom.md.txt): Declarative Shadow DOM is a new way to implement and use Shadow DOM directly in HTML.
- [Decrease front-end size](https://web.dev/articles/decrease-frontend-size.md.txt): How to use webpack to make your app as small as possible
- [Deep dive into top web developer pain points](https://web.dev/articles/deep-dive-into-developer-pain-points.md.txt): A collection of insights on the top pain points, collected from a number of one to one conversations with web developers.
- [Defer non-critical CSS](https://web.dev/articles/defer-non-critical-css.md.txt): Learn how to defer non-critical CSS with the goal of optimizing the Critical Rendering Path, and improving First Contentful Paint (FCP).
- [How to define your install strategy](https://web.dev/articles/define-install-strategy.md.txt): Best practices for combining different installation offerings to increase installation rates and avoid platform competition and cannibalization.
- [How the Core Web Vitals metrics thresholds were defined](https://web.dev/articles/defining-core-web-vitals-thresholds.md.txt): The research and methodology behind Core Web Vitals thresholds
- [Designcember Calculator](https://web.dev/articles/designcember-calculator.md.txt): A skeuomorphic attempt at recreating a solar calculator on the web that makes use of the ambient light sensor and the window controls overlay feature.
- [Detached window memory leaks](https://web.dev/articles/detached-window-memory-leaks.md.txt): Detached windows are a common type of memory leak that is particularly difficult to find and fix.
- [Web Developer Satisfaction](https://web.dev/articles/developer-satisfaction.md.txt): Web Developer Satisfaction is a Google project to gather information about the needs of web developers. The goal is a more reliable, predictable and interoperable web platform, enabling developers to invest in and trust it, and adopt and use new features to grow the platform and their business.
- [Device Orientation &amp; Motion](https://web.dev/articles/device-orientation.md.txt): Device motion and orientation events provide access to the built-in accelerometer, gyroscope, and compass in mobile devices.
- [Pixel-perfect rendering with devicePixelContentBox](https://web.dev/articles/device-pixel-content-box.md.txt): Since Chrome 84, ResizeObserver supports a new box measurement called device-pixel-content-box, that measures the element&#39;s dimension in physical pixels. This is crucial for rendering pixel-perfect graphics, especially in the context of high-density screens.
- [Accessing hardware devices on the web](https://web.dev/articles/devices-introduction.md.txt): This article helps Web developers pick the right hardware API based on a given device.
- [Disable mouse acceleration to provide a better FPS gaming experience](https://web.dev/articles/disable-mouse-acceleration.md.txt): Web apps can now disable mouse acceleration when capturing pointer events.
- [Improving page dismissal in synchronous XMLHttpRequest()](https://web.dev/articles/disallow-synchronous-xhr.md.txt): It&#39;s common for a page or app to have unsubmitted analytics or other data at the time a user closes it. Sites use a synchronous call to XMLHttpRequest() to keep the page or app open until its data is passed to the server. It hurts the user experience and ignores better ways to save data. Chrome 80 implements a recent spec change to address this.
- [Discover performance opportunities with Lighthouse](https://web.dev/articles/discover-performance-opportunities-with-lighthouse.md.txt): Lighthouse is a tool that helps you measure and find ways to improve a page&#39;s performance. Lighthouse gives you a report on how the page did. The report provides a score for each metric and a list of opportunities which, if you implement them, should make the page load faster.

- [DOM Order Matters](https://web.dev/articles/dom-order-matters.md.txt): The importance of the default DOM order
- [How large DOM sizes affect interactivity, and what you can do about it](https://web.dev/articles/dom-size-and-interactivity.md.txt): Large DOM sizes can be a factor in whether interactions are fast or not. Learn more about the relationship between DOM size and INP, and what you can do to reduce DOM size and other ways to limit rendering work when your page has lots of DOM elements.
- [Jumping the hurdles with the Gamepad API](https://web.dev/articles/doodles-gamepad.md.txt): Guidance to build modern web experiences that work on any browser.
- [Downloading resources in HTML5 - a[download]](https://web.dev/articles/downloading-resources-in-html5-a-download.md.txt): Chrome now supports the HTML spec&#39;s new `download` attribute to `a` elements.
- [The HTML5 Drag and Drop API](https://web.dev/articles/drag-and-drop.md.txt): The HTML5 Drag and Drop API lets developers make almost any element on a page draggable.
- [Drawing to canvas in Emscripten](https://web.dev/articles/drawing-to-canvas-in-emscripten.md.txt): Learn how to render 2D graphics to a canvas on the web from WebAssembly with Emscripten.
- [How Progressive Web Apps can drive business success](https://web.dev/articles/drive-business-success.md.txt): Build a solid business case for your PWA. Learn when you should invest, and how you can measure its success.
- [Easy high DPI images](https://web.dev/articles/easy-high-dpi-images.md.txt): Guidance to build modern web experiences that work on any browser.
- [Effectively managing memory at Gmail scale](https://web.dev/articles/effectivemanagement.md.txt): Guidance to build modern web experiences that work on any browser.
- [Efficiently load third-party JavaScript](https://web.dev/articles/efficiently-load-third-party-javascript.md.txt): Learn how to improve load times and user experience by avoiding the common pitfalls of using third-party scripts.
- [Best practices for using third-party embeds](https://web.dev/articles/embed-best-practices.md.txt): This document discusses performance best practices that you can use when loading third-party embeds, efficient loading techniques and the Layout Shift Terminator tool that helps reduce layout shifts for popular embeds.
- [Emscripten’s embind](https://web.dev/articles/embind.md.txt): Emscripten’s embind
- [Introduction to Encrypted Media Extensions](https://web.dev/articles/eme-basics.md.txt): Guidance to build modern web experiences that work on any browser.
- [Empowering payment apps with Web Payments](https://web.dev/articles/empowering-payment-apps-with-web-payments.md.txt): Web Payments aims to provide frictionless payment experience on the web. Learn how it works, its benefits, and get ready to integrate your payment app with Web Payments.
- [Embedding JavaScript snippets in C++ with Emscripten](https://web.dev/articles/emscripten-embedding-js-snippets.md.txt): Learn how to embed JavaScript code in your WebAssembly library to communicate with the outside world.
- [Emscripten and npm](https://web.dev/articles/emscripten-npm.md.txt): How do you integrate WebAssembly into this setup? In this article we are going to work this out with C/C&#43;&#43; and Emscripten as an example.
- [Emscripting a C library to Wasm](https://web.dev/articles/emscripting-a-c-library.md.txt): Emscripting a C library to Wasm
- [Enable HTTPS on your servers](https://web.dev/articles/enable-https.md.txt): Enabling HTTPS on your servers is critical to securing your webpages.
- [ES modules in service workers](https://web.dev/articles/es-modules-in-sw.md.txt): Service workers can use static imports of ES modules to bring in extra code, as an alternative to importScripts().
- [Data-binding revolutions with Object.observe()](https://web.dev/articles/es7-observe.md.txt): Guidance to build modern web experiences that work on any browser.
- [JavaScript eventing deep dive](https://web.dev/articles/eventing-deepdive.md.txt): Guidance to build modern web experiences that work on any browser.
- [Stream updates with server-sent events](https://web.dev/articles/eventsource-basics.md.txt): Guidance to build modern web experiences that work on any browser.
- [Excalidraw and Fugu: Improving Core User Journeys](https://web.dev/articles/excalidraw-and-fugu.md.txt): A write-up of Thomas Steiner&#39;s Google I/O 2021 talk titled Excalidraw and Fugu: Improving Core User Journeys
- [Extract critical CSS](https://web.dev/articles/extract-critical-css.md.txt): Learn how to

improve render times with critical CSS technique and how to choose the best tool for your project.
- [FAQ for hacked sites](https://web.dev/articles/faq-for-hacked-sites.md.txt): This page brings together answers to the questions about hacking we at Google hear most often.
- [Farewell to HTML5Rocks](https://web.dev/articles/farewell-html5rocks.md.txt): So long HTML5Rocks, it&#39;s been nice knowing you.
- [Fast ads matter](https://web.dev/articles/fast-ads-matter.md.txt): Understand the value of fast ads and how to think about ad speed.
- [Fast playback with audio and video preload](https://web.dev/articles/fast-playback-with-preload.md.txt): Faster playback start means more people watching your video or listening to your audio. That&#39;s a known fact. In this article I&#39;ll explore techniques you can use to accelerate your media playback by actively preloading resources depending on your use case.
- [Optimize Angular&#39;s change detection](https://web.dev/articles/faster-angular-change-detection.md.txt): Learn how to optimize your Angular app&#39;s change detection to make it more responsive.
- [First Contentful Paint (FCP)](https://web.dev/articles/fcp.md.txt): This post introduces the First Contentful Paint (FCP) metric and explains how to measure it
- [Implement error handling when using the Fetch API](https://web.dev/articles/fetch-api-error-handling.md.txt): Catching errors when working with the Fetch API.
- [Protect your resources from web attacks with Fetch Metadata](https://web.dev/articles/fetch-metadata.md.txt): Fetch Metadata is a new web platform feature designed to allow servers to protect themselves from cross-origin attacks.
- [Optimize resource loading with the Fetch Priority API](https://web.dev/articles/fetch-priority.md.txt): The Fetch Priority API indicates the relative priority of resources to the browser. It can enable optimal loading and improve Core Web Vitals.
- [First Input Delay (FID)](https://web.dev/articles/fid.md.txt): This post introduces the First Input Delay (FID) metric and explains how to measure it
- [The synchronous FileSystem API for workers](https://web.dev/articles/filesystem-sync.md.txt): Guidance to build modern web experiences that work on any browser.
- [Find slow interactions in the field](https://web.dev/articles/find-slow-interactions-in-the-field.md.txt): Before you can reproduce slow interactions in the lab to optimize your website&#39;s Interaction to Next Paint, you&#39;ll need to lean on field data to find them. Learn how to do just that in this guide.
- [First-party cookie recipes](https://web.dev/articles/first-party-cookie-recipes.md.txt): Learn how to set first-party cookies to ensure security, cross-browser compatibility, and minimize chances of breakage once third-party cookies are phased out.
- [Fix the gibberish hack](https://web.dev/articles/fix-the-gibberish-hack.md.txt): This guide is created specifically for a type of hack that adds keyword-heavy gibberish pages to your site.
- [Fix the Japanese keyword hack](https://web.dev/articles/fix-the-japanese-keyword-hack.md.txt): This guide is created specifically for a type of hack that creates autogenerated Japanese text on your site.
- [Fixing layout instability](https://web.dev/articles/fixing-layout-instability.md.txt): A walkthrough of using WebPageTest to identify and fix layout instability issues.
- [Fixing mixed content](https://web.dev/articles/fixing-mixed-content.md.txt): Find out how to fix mixed content errors on your website, in order to protect users and ensure that all of your content loads.
- [Fixing website speed cross-functionally](https://web.dev/articles/fixing-website-speed-cross-functionally.md.txt): How other departments can help make your website speed optimization project a bigger success.
- [Best practices for fonts](https://web.dev/articles/font-best-practices.md.txt): Learn about how to optimize web fonts for Core Web Vitals.
- [Making Fullscreen Experiences](https://web.dev/articles/fullscreen.md.txt): Going fullscreen.
- [Play the Chrome dino game with your gamepad](https://web.dev/articles/gamepad.md.txt): Learn to control web games with the Gamepad API.
- [GDE community highlight: Alba Silvente Fuentes](https://web.dev/articles/gde-focus/alba-silvente-fuentes.md.txt): One of a series of interviews with members of the Google Developers Experts (GDE) program.
- [A win-win situation](https://web.dev/articles/gde-focus/helpdev.md.txt): GDE Enrique Fernandez Guerra on open sourcing his NGO HelpDev.
- [GDE community highlight: Lars Knudsen](https://web.dev/articles/gde-focus/lars-knudsen.md.txt): One of a series of interviews with members of the Google Developers Experts (GDE) program.

- [GDE community highlight: Nishu Goel](https://web.dev/articles/gde-focus/nishu-goel.md.txt): One of a series of interviews with members of the Google Developers Experts (GDE) program.
- [Finding courage and inspiration in the developer community](https://web.dev/articles/gde-mentoring.md.txt): Web Google Developers Experts on how mentoring programs empowered them to become leaders.
- [Get started: optimize an Angular application](https://web.dev/articles/get-started-optimize-angular.md.txt): Learn how to make your Angular application faster, more reliable, discoverable, installable, and accessible!
- [Get started: optimize your React app](https://web.dev/articles/get-started-optimize-react.md.txt): React is an open-source library that makes building UIs easier. This learning path will cover different APIs and tools within the ecosystem that you should consider using to improve the performance and usability of your application.
- [Capture audio and video in HTML5](https://web.dev/articles/getusermedia-intro.md.txt): Guidance to build modern web experiences that work on any browser.
- [Global and local variable scope](https://web.dev/articles/global-and-local-scope.md.txt): Learn about scope and how it works in JavaScript.
- [Glossary for hacked sites](https://web.dev/articles/glossary-for-hacked-sites.md.txt): The glossary covers a collection of technical terms that are referenced throughout our security documentation.
- [Rendering HTML5 in older browsers with Google Chrome Frame](https://web.dev/articles/google-chrome-frame.md.txt): Guidance to build modern web experiences that work on any browser.
- [Web developer tools for debugging JavaScript issues in Google Search](https://web.dev/articles/google-search-tools.md.txt): How to debug SEO issues on individual pages or across an entire site.
- [GOV.UK drops jQuery from their front end.](https://web.dev/articles/gov-uk-drops-jquery.md.txt): GOV.UK dropped their jQuery dependency from their front end. You&#39;ll never guess what happened. (Yes, you will.)
- [Improved Next.js and Gatsby page load performance with granular chunking](https://web.dev/articles/granular-chunking-nextjs.md.txt): Learn how both Next.js and Gatsby have improved their build output to minimize duplicate code and improve page load performance
- [Help, I think I&#39;ve been hacked](https://web.dev/articles/hacked.md.txt): Learn how and why websites are hacked.
- [Hacked with malware](https://web.dev/articles/hacked-with-malware.md.txt): This guide explains how to deal with a site that has been hacked with malware.
- [Handling navigation requests](https://web.dev/articles/handling-navigation-requests.md.txt): Navigation requests are requests for HTML made whenever you enter a new URL in the navigation bar, or follow a link on a page. This is where service workers make their biggest impact on performance: by using a service worker to respond without waiting for the network, you can ensure that navigations are reliably fast and resilient.
- [Handling optional payment information with a service worker](https://web.dev/articles/handling-optional-payment-information.md.txt): Once a web-based payment app is registered, it&#39;s ready to accept payment requests from merchants. This article teaches you how to orchestrate a payment transaction from a service worker during runtime.
- [Hands-on with Portals: seamless navigation on the web](https://web.dev/articles/hands-on-portals.md.txt): The newly proposed Portals API helps keep your front-end simple while allowing seamless navigations with custom transitions. In this article, get hands-on experience using Portals to improve user experience across your site.
- [Headings and landmarks](https://web.dev/articles/headings-and-landmarks.md.txt): By using the correct elements for headings and landmarks, you can dramatically improve the navigation experience for users of assitive technology.
- [Hiding and updating content](https://web.dev/articles/hiding-and-updating-content.md.txt): Hiding content from assistive technology
- [High DPI images for variable pixel densities](https://web.dev/articles/high-dpi.md.txt): Understand how to serve the best quality images as efficiently as possible.
- [Cross-browser paint worklets and Houdini.how](https://web.dev/articles/houdini-how.md.txt): Learn how to implement cross-browser Houdini Paint API&#39;s and explore a world of Houdini worklets with Houdini.how.
- [How AMP can guarantee fastness in your Next.js app](https://web.dev/articles/how-amp-can-guarantee-fastness-in-your-nextjs-app.md.txt): Guidance to build modern web experiences that work on any browser.
- [How can performance improve conversion?](https://web.dev/articles/how-can-performance-

improve-conversion.md.txt): Learn what impact website performance has on different parts of the e-commerce funnel
- [How do I know if my site was hacked?](https://web.dev/articles/how-do-i-know-if-my-site-was-hacked.md.txt): Steps to follow if you think that your site may be hacked.
- [How the payment ecosystem works](https://web.dev/articles/how-payment-ecosystem-works.md.txt): Learn more about who is involved in the Web Payments ecosystem, how they interact with each other, and how you can participate.
- [How Payment Request API works](https://web.dev/articles/how-payment-request-api-works.md.txt): Learn how the Payment Request API works at a high level.
- [How search works](https://web.dev/articles/how-search-works.md.txt): Search engines are the digital version of a librarian. They use a comprehensive index to help find the right information for a query. Understanding the basics of search prepares you to make your content discoverable for users.
- [How to choose your Baseline target](https://web.dev/articles/how-to-choose-your-baseline-target.md.txt): Learn what Baseline targets are, how to choose one, and how it can help your development experience.
- [How to distribute Signed HTTP Exchanges (SXG) using nginx](https://web.dev/articles/how-to-distribute-signed-http-exchanges.md.txt): How to get and serve SXG files using nginx, and the challenges of subresource prefetching.
- [How to file a good browser bug](https://web.dev/articles/how-to-file-a-good-bug.md.txt): Telling browser vendors about issues you find in their browser, on a specific device or platform is an integral part of making the web platform better!
- [How to measure speed?](https://web.dev/articles/how-to-measure-speed.md.txt): Real-world performance is highly variable due to differences in users&#39; devices, network connections, and other factors. In this post we explore tools that can help you collect lab or field data to assess page performance.
- [How to review for accessibility](https://web.dev/articles/how-to-review.md.txt): How to review your site for accessibility issues.
- [How to set up Signed HTTP Exchanges (SXG) using nginx](https://web.dev/articles/how-to-set-up-signed-http-exchanges.md.txt): How to generate a TLS certificate with SXG extensions, install tools for generating SXG files, and configure nginx to serve SXG files.
- [How to stay fast?](https://web.dev/articles/how-to-stay-fast.md.txt): Brands that optimize speed will often find they regress quickly. In this post we explore how to make sure your fast experience stays fast.
- [Use HTTPS for local development](https://web.dev/articles/how-to-use-local-https.md.txt): Guidance to build modern web experiences that work on any browser.
- [web.dev engineering blog #1: How we build the site and use Web Components](https://web.dev/articles/how-we-build-webdev-and-use-web-components.md.txt): In this first post from the web.dev engineering team, learn about how we build the site—including our use of Eleventy and Web Components.
- [Building Designcember](https://web.dev/articles/how-we-built-designcember.md.txt): A peek into the process and tools used to build the holiday-calendar-style experience of Designcember.
- [How browsers work](https://web.dev/articles/howbrowserswork.md.txt): Guidance to build modern web experiences that work on any browser.
- [Prevent unnecessary network requests with the HTTP Cache](https://web.dev/articles/http-cache.md.txt): The browser&#39;s HTTP Cache is your first line of defense against unnecessary network requests.
- [Improve security and privacy by updating HTTP Cache](https://web.dev/articles/http-cache-security.md.txt): Forgetting or misusing the Cache-Control header might negatively impact the security of your website and your users&#39; privacy. Get recommendations for high-value websites.
- [Icons and browser colors](https://web.dev/articles/icons-and-browser-colors.md.txt): Modern browsers make it easy to customize certain components, like icons, the address bar color, and even add things like custom tiles. These simple tweaks can increase engagement and bring users back to your site.
- [Identify resources loaded from the network](https://web.dev/articles/identify-resources-via-network-panel.md.txt): Coming up with the right caching strategies for your web application requires getting a handle on what exactly you&#39;re loading. When building a reliable web application, the network can be subject to all kinds of dark forces. You need to understand the network&#39;s vulnerabilities if you hope to cope with them in your app.
- [Identify slow third-party JavaScript](https://web.dev/articles/identify-slow-third-party-javascript.md.txt): Learn how to use Lighthouse and Chrome DevTools to identify slow third-party

resources.
- [Identify the vulnerability](https://web.dev/articles/identify-the-vulnerability.md.txt): How to search for vulnerabilities on your site.
- [It&#39;s time to lazy-load offscreen iframes!](https://web.dev/articles/iframe-lazy-loading.md.txt): This post covers the loading attribute and how to use it to control the loading of iframes.
- [Use image CDNs to optimize images](https://web.dev/articles/image-cdns.md.txt): Consider using an image CDN to optimize your site&#39;s images and reduce its data cost for your users.
- [Feedback from the summer 2019 image optimization survey](https://web.dev/articles/image-optimization-survey-2019.md.txt): Comments from survey respondents about various image optimization techniques.
- [Image policies for fast load times and more](https://web.dev/articles/image-policies.md.txt): Images take up a significant amount of visual space and make up the majority of the downloaded bytes on a website. Use the new feature policies to identify oversized images.
- [Make your site picture perfect with images.tooling.report](https://web.dev/articles/images-tooling-report.md.txt): Check out the state of image tooling.
- [Imperative caching guide](https://web.dev/articles/imperative-caching-guide.md.txt): How to communicate window and service worker to perform tasks related to performance, caching and offline.
- [HTML Imports](https://web.dev/articles/imports.md.txt): Guidance to build modern web experiences that work on any browser.
- [Explore product review suggestions with client-side AI](https://web.dev/articles/improve-reviews-ai.md.txt): Online stores can see a 270% increase in conversions by displaying product reviews. Support better reviews with client-side AI.
- [Incorporate performance budgets into your build process](https://web.dev/articles/incorporate-performance-budgets-into-your-build-tools.md.txt): The best way to keep an eye on your performance budget is to automate it. Find out how to choose a tool that best fits your needs and current setup.
- [Work with IndexedDB](https://web.dev/articles/indexeddb.md.txt): A guide to the basics of IndexedDB.
- [Best Practices for Persisting Application State with IndexedDB](https://web.dev/articles/indexeddb-best-practices-app-state.md.txt): Learn best practices for syncing application state between IndexedDB and popular state management libraries.
- [Databinding UI elements with IndexedDB](https://web.dev/articles/indexeddb-uidatabinding.md.txt): Guidance to build modern web experiences that work on any browser.
- [The inert attribute](https://web.dev/articles/inert.md.txt): The inert property is a global HTML attribute that simplifies how to remove and restore user input events for an element, including focus events and events from assistive technologies.
- [Interaction to Next Paint (INP)](https://web.dev/articles/inp.md.txt): This post introduces the Interaction to Next Paint (INP) metric and explains how it works, how to measure it, and offers suggestions on how to improve it.
- [What does it take to be installable?](https://web.dev/articles/install-criteria.md.txt): Progressive Web App installability criteria.
- [IntersectionObserver&#39;s coming into view](https://web.dev/articles/intersectionobserver.md.txt): IntersectionObservers let you know when an observed element enters or exits the browser&#39;s viewport.
- [Trust is good, observation is better: Intersection Observer v2](https://web.dev/articles/intersectionobserver-v2.md.txt): Intersection Observer v2 adds the capability to not only observe intersections per se, but to also detect if the intersecting element was visible at the time of intersection.
- [Important Security Terminology](https://web.dev/articles/intro-to-security-terminology.md.txt): Two of the hurdles developers face when migrating to HTTPS are concepts and terminology. This guide provides a brief overview of both.
- [Introduction to fetch()](https://web.dev/articles/introduction-to-fetch.md.txt): The `fetch()` API is landing in the window object and can replace XHRs.
- [Is it :modal?](https://web.dev/articles/is-it-modal.md.txt): This handy CSS pseudo-selector gives you a way to select elements that are modal.
- [JavaScript: What is the meaning of this?](https://web.dev/articles/javascript-this.md.txt): Figuring out the value of `this` can be tricky in JavaScript, here&#39;s how to do it…
- [The difference between JavaScript libraries and frameworks](https://web.dev/articles/js-libraries-vs-frameworks.md.txt): Understand the differences between frameworks and libraries in the context of a client-side JavaScript environment.

- [Keyboard access fundamentals](https://web.dev/articles/keyboard-access.md.txt): Many different users rely on the keyboard to navigate applications—from users with temporary and permanent motor impairments to users who use keyboard shortcuts to be more efficient and productive. Having a good keyboard navigation strategy for your application creates a better experience for everyone.
- [Why lab and field data can be different (and what to do about it)](https://web.dev/articles/lab-and-field-data-differences.md.txt): Learn why tools that monitor Core Web Vitals metrics may report different numbers, and how to interpret those differences.
- [Labels and text alternatives](https://web.dev/articles/labels-and-text-alternatives.md.txt): In order for a screen reader to present a spoken UI to the user, meaningful elements must have proper labels or text alternatives. A label or text alternative gives an element its accessible name, one of the key properties for expressing element semantics in the accessibility tree.
- [Lazy loading video](https://web.dev/articles/lazy-loading-video.md.txt): This post explains lazy loading and the options available to you to lazy loading video.
- [Largest Contentful Paint (LCP)](https://web.dev/articles/lcp.md.txt): This post introduces the Largest Contentful Paint (LCP) metric and explains how to measure it
- [The performance effects of too much lazy loading](https://web.dev/articles/lcp-lazy-loading.md.txt): Eagerly loading images within the initial viewport—while liberally lazy loading the rest—can improve Web Vitals while loading fewer bytes.
- [Life of a payment transaction](https://web.dev/articles/life-of-a-payment-transaction.md.txt): Learn how merchants integrate payment apps, how payment transactions work with the Payment Request API, and what&#39;s possible in Web Payments.
- [CSS color-scheme-dependent colors with light-dark()](https://web.dev/articles/light-dark.md.txt): description: Define colors that react to the used color-scheme with the light-dark() function.
- [Performance monitoring with Lighthouse CI](https://web.dev/articles/lighthouse-ci.md.txt): Learn how to setup Lighthouse CI and integrate it into developer workflows.
- [Lighthouse user flows](https://web.dev/articles/lighthouse-user-flows.md.txt): Try out a new Lighthouse API to measure performance and best practices throughout a user flow.
- [Prefetch resources to speed up future navigations](https://web.dev/articles/link-prefetch.md.txt): Learn about rel=prefetch resource hint and how to use it.
- [web.dev LIVE wrap-up](https://web.dev/articles/live-wrap-up.md.txt): A summary of the major news and updates that were announced during our 3-day online community event, and a reminder about upcoming regional events.
- [Understand LLM sizes](https://web.dev/articles/llm-sizes.md.txt): Take a look at a few real-world LLMs and the practical implications of different model sizes.
- [Techniques to make a web app load fast, even on a feature phone](https://web.dev/articles/load-faster-like-proxx.md.txt): Feature phones are making a resurgence and are popular, especially in emerging markets where 2G is the norm. Here are our learnings from making PROXX, a mobile Minesweeper clone, load fast on feature phones on 2G.
- [Effectively loading ads without impacting page speed](https://web.dev/articles/loading-ads-page-speed.md.txt): In today&#39;s digital world, online advertising is a critical part of the free web we all enjoy. However, poorly implemented ads can lead to a slower browsing experience, frustrating users and diminishing engagement. Learn how to effectively load ads without impacting your page speed, ensuring a seamless user experience, and maximizing revenue opportunities for website owners.
- [Loading WebAssembly modules efficiently](https://web.dev/articles/loading-wasm.md.txt): When working with WebAssembly, you often want to download a module, compile it, instantiate it, and then use whatever it exports in JavaScript. This post explains our recommended approach for optimal efficiency.
- [Logical layout enhancements with flow-relative shorthands](https://web.dev/articles/logical-property-shorthands.md.txt): New logical property shorthands and new inset properties for Chromium.
- [Are long JavaScript tasks delaying your Time to Interactive?](https://web.dev/articles/long-tasks-devtools.md.txt): Learn to diagnose costly, work-preventing user interaction.
- [Love your cache ❤️](https://web.dev/articles/love-your-cache.md.txt): First load performance is important, but it&#39;s not everything. Users who load your site a second time will use their cache to get access to your content—so it&#39;s key to make sure it works well too, both for speed and correctness.
- [How Chrome handles updates to the web app manifest](https://web.dev/articles/manifest-updates.md.txt): What it takes to change icons, shortcuts, colors, and other metadata in your web app manifest for your PWA.

- [Real-Time Effects For Images and Video](https://web.dev/articles/manipulating-live-effects.md.txt): Many of today&#39;s most popular apps let you apply filters and effects to images or video. This article shows how to implement these features on the open web.
- [Manually diagnose slow interactions in the lab](https://web.dev/articles/manually-diagnose-slow-interactions-in-the-lab.md.txt): You&#39;ve looked through your field data, and it turns out that you have some slow interactions. The next step is to learn more about how to manually test those interactions, and identify the causes behind them.
- [Adaptive icon support in PWAs with maskable icons](https://web.dev/articles/maskable-icon.md.txt): Maskable icons are a format that give you more control and let your Progressive Web App use adaptive icons. A maskable icon can look great on all Android devices.
- [Measuring offline usage](https://web.dev/articles/measuring-offline-usage.md.txt): How to track offline usage of your site so that you can make a case as to why your site needs a better offline experience.
- [Media application basics](https://web.dev/articles/media-application-basics.md.txt): Working with media often requires changing the characteristics of media files. On this page, you&#39;ll learn about the tools used and how to install them quickly.
- [Capturing an image from the user](https://web.dev/articles/media-capturing-images.md.txt): Most browsers can get access to the user&#39;s camera.
- [Media conversion](https://web.dev/articles/media-conversion.md.txt): Commands needed to convert a raw mov file to media assets packaged for DASH or HLS.
- [What is EME?](https://web.dev/articles/media-eme.md.txt): Enabling HTTPS on your servers is critical to securing your webpages.
- [Media encryption](https://web.dev/articles/media-encryption.md.txt): Learn digital rights management concepts, and the commands needed to get from a raw mov file to encrypted media packaged for MPEG-DASH or HLS using both Clear Key or Widevine encryption.
- [What is a media experience?](https://web.dev/articles/media-experience.md.txt): Videos can be fun and informative. For a good user experience, videos need to meet a number of technical requirements.
- [Media file basics](https://web.dev/articles/media-file-basics.md.txt): You might think that you can take a video from a camera and upload it to the web as is. Preparing a video for serving from your own site is a bit more complicated.
- [Media frameworks](https://web.dev/articles/media-frameworks.md.txt): Media frameworks come in both the proprietary and open-source variety, and at their core are a set of APIs that support audio and/or video playback for various container formats and transmission protocols.
- [Mobile Web Video Playback](https://web.dev/articles/media-mobile-web-video-playback.md.txt): Create the best mobile media experience on the Web by following these best practises.
- [Media Source Extensions](https://web.dev/articles/media-mse-basics.md.txt): Media Source Extensions (MSE) is a JavaScript API that lets you build streams for playback from segments of audio or video.
- [New syntax for range media queries](https://web.dev/articles/media-query-range-syntax.md.txt): Find out how this new syntax can streamline media queries.
- [Recording Audio from the User](https://web.dev/articles/media-recording-audio.md.txt): Most browsers can get access to the user&#39;s microphone.
- [Recording Video from the User](https://web.dev/articles/media-recording-video.md.txt): Most browsers can get access to the user&#39;s camera.
- [Customize media notifications and playback controls with the Media Session API](https://web.dev/articles/media-session.md.txt): Web developers can customize media notifications and respond to media related events such as seeking or track changing with the Media Session API.
- [Media streaming basics](https://web.dev/articles/media-streaming-basics.md.txt): Media streaming is the method for continuously delivering multimedia content from a server where the content has been split into individual chunks of data that can be joined back together during playback in a specific order through a range request using protocols such as DASH and HLS.
- [Migrate to User-Agent Client Hints](https://web.dev/articles/migrate-to-ua-ch.md.txt): Strategies to migrate your site from relying on the user-agent string to User-Agent Client Hints.
- [CSS min(), max(), and clamp()](https://web.dev/articles/min-max-clamp.md.txt): Min, max, and clamp provide powerful CSS capabilities that enable more responsive styling with fewer liens of code. This post goes over how to control element sizing, maintain proper spacing, and implement fluid typography using these well-supported CSS math functions.
- [What are mini apps?](https://web.dev/articles/mini-apps/mini-app-about.md.txt): This chapter introduces the concept of mini apps and provides examples of their look and feel.

- [Other mini app runtime environments](https://web.dev/articles/mini-apps/mini-app-alternative-runtime-environments.md.txt): This chapter presents a number of runtime environments for mini apps that are not mobile devices.
- [Mini app components](https://web.dev/articles/mini-apps/mini-app-components.md.txt): This chapter provides details on the components that all mini app platforms make available.
- [Concluding thoughts about mini apps from a web developer](https://web.dev/articles/mini-apps/mini-app-conclusion.md.txt): This chapter concludes the mini apps collection with the observation that thinking outside of the box and taking input and inspiration from outside of one&#39;s own bubble can definitely help when building a better future on the web..
- [Mini app DevTools](https://web.dev/articles/mini-apps/mini-app-devtools.md.txt): This chapter provides details on the DevTools experience of various mini apps platforms.
- [Applying the mini app programming principles to an example project](https://web.dev/articles/mini-apps/mini-app-example-project.md.txt): This chapter shows an example project that follows the &#34;programming the mini app way&#34; approach.
- [Mini app markup, styling, scripting, and updating](https://web.dev/articles/mini-apps/mini-app-markup-styling-and-scripting.md.txt): This chapter looks at the mark-up, styling, scripting, and updating options of various mini apps platforms.
- [Mini app open source projects](https://web.dev/articles/mini-apps/mini-app-open-source-projects.md.txt): This chapter presents a selection of interesting mini app open source projects.
- [Programming the mini app way](https://web.dev/articles/mini-apps/mini-app-programming-way.md.txt): This chapter introduces the way of programming the mini app way.
- [Project structure, lifecycle, and bundling](https://web.dev/articles/mini-apps/mini-app-project-structure-lifecycle-and-bundling.md.txt): This chapter covers the project structure, the lifecycle, and the bundling of mini apps.
- [Mini app standardization](https://web.dev/articles/mini-apps/mini-app-standardization.md.txt): This chapter introduces the standardization effort that has been started for mini apps.
- [Mini apps and super apps](https://web.dev/articles/mini-apps/mini-app-super-apps.md.txt): This chapter introduces the concept of super apps and presents the major super app providers.
- [What are H5 and QuickApp?](https://web.dev/articles/mini-apps/mini-app-what-are-h5-and-quickapp.md.txt): This chapter provides background on H5 apps and QuickApp, which are both distinct from mini apps.
- [Mobifying your HTML5 site](https://web.dev/articles/mobifying.md.txt): Guidance to build modern web experiences that work on any browser.
- [A non-responsive approach to building cross-device webapps](https://web.dev/articles/mobile-cross-device.md.txt): Guidance to build modern web experiences that work on any browser.
- [HTML5 techniques for optimizing mobile performance](https://web.dev/articles/mobile-optimization-and-performance.md.txt): Guidance to build modern web experiences that work on any browser.
- [Profiling mobile HTML5 apps with Chrome DevTools](https://web.dev/articles/mobile-profiling.md.txt): Guidance to build modern web experiences that work on any browser.
- [Multi-touch web development](https://web.dev/articles/mobile-touch.md.txt): Guidance to build modern web experiences that work on any browser.
- [Touch and mouse](https://web.dev/articles/mobile-touchandmouse.md.txt): Guidance to build modern web experiences that work on any browser.
- [The &lt;model-viewer&gt; web component](https://web.dev/articles/model-viewer.md.txt): The &lt;model-viewer&gt; web component lets you use 3D models on a web page declaratively.
- [Threading the web with module workers](https://web.dev/articles/module-workers.md.txt): Module workers make it easy to unblock the main thread by moving expensive code to a background thread while keeping the ergonomic and performance benefits of standard JavaScript modules.
- [Preload modules](https://web.dev/articles/modulepreload.md.txt): Module preload offers a way of declaratively loading JavaScript modules ahead of time.
- [Monitor and analyze the app](https://web.dev/articles/monitor-and-analyze.md.txt): What tools to use to keep track of and analyze the webpack bundle
- [Monitor your web page&#39;s total memory usage with measureUserAgentSpecificMemory()](https://web.dev/articles/monitor-total-page-memory-usage.md.txt): Learn how to measure memory usage of your web page in production to detect regressions.
- [More capable form controls](https://web.dev/articles/more-capable-form-controls.md.txt): New web platform features make it easier to build custom elements that work like built-in form controls.
- [Media Source Extensions for Audio](https://web.dev/articles/mse-seamless-playback.md.txt):

Media Source Extensions (MSE) provide extended buffering and playback control for the HTML5 audio and video elements. While originally developed to facilitate Dynamic Adaptive Streaming over HTTP (DASH) based video players, MSE can be used for audio; specifically for gapless playback.
- [Multi-device content](https://web.dev/articles/multi-device-content.md.txt): Consider content as well as layout and graphic design when building for a range of users and devices.
- [Progressive Web Apps in multi-origin sites](https://web.dev/articles/multi-origin-pwas.md.txt): Multi-origin architectures presents many challenges when building PWAs. Explore the good and bad uses of multiple origins, and some workarounds to build PWAs in multi-origin sites.
- [Multiplayer audio fun](https://web.dev/articles/multiplayer-audio-fun.md.txt): A demo using the Web Audio API.
- [HTML5 vs Native](https://web.dev/articles/nativedebate.md.txt): Guidance to build modern web experiences that work on any browser.
- [Semantics and navigating content](https://web.dev/articles/navigating-content.md.txt): The role of semantics in page navigation
- [How to assess loading performance in the field with Navigation Timing and Resource Timing](https://web.dev/articles/navigation-and-resource-timing.md.txt): Learn the basics of using the Navigation and Resource Timing APIs to assess loading performance in the field.
- [What is network reliability and how do you measure it?](https://web.dev/articles/network-connections-unreliable.md.txt): The modern web is enjoyed by a wide swath of people, using a range of different devices and types of network connections. Your creations can reach users all across the world, but delivering a reliable experience on the web for all of your users can be challenging. It can be a challenge just to understand what reliability means.
- [Network Error Logging (NEL)](https://web.dev/articles/network-error-logging.md.txt): Use Network Error Logging (NEL) to collect client-side network errors.
- [New patterns for media apps](https://web.dev/articles/new-patterns-for-media-apps.md.txt): This blog post announces a new collection of patterns for media apps.
- [New patterns](https://web.dev/articles/new-patterns-july-2022.md.txt): Animations, theming, components, and more layout patterns are live and ready to help kick start or inspire your UI and UX.
- [New Progressive Web App training now available](https://web.dev/articles/new-pwa-training.md.txt): A new, six part Progressive Web App training is now available, complete with a new series of codelabs to teach you how to build reliable, installable, and capable PWAs.
- [The new responsive: Web design in a component-driven world](https://web.dev/articles/new-responsive.md.txt): User-preference based media features, container queries, and media queries for new screen types, such as foldable screens, will enable us to usehr in a new era of responsive web design.
- [Using the Notifications API](https://web.dev/articles/notifications-quick.md.txt): Guidance to build modern web experiences that work on any browser.
- [Use web workers to run JavaScript off the browser&#39;s main thread](https://web.dev/articles/off-main-thread.md.txt): The browser&#39;s main thread is incredibly overworked. By using web workers to shift code off the main thread, you can significantly improve your app&#39;s reliability and user experience.
- [The Offline Cookbook](https://web.dev/articles/offline-cookbook.md.txt): Some common recipes for making your app work offline.
- [Create an offline fallback page](https://web.dev/articles/offline-fallback-page.md.txt): Learn how to create a simple offline experience for your app.
- [Offline UX design guidelines](https://web.dev/articles/offline-ux-design-guidelines.md.txt): Guidance to build modern web experiences that work on any browser.
- [OffscreenCanvas—speed up your canvas operations with a web worker](https://web.dev/articles/offscreen-canvas.md.txt): This document explains how you can use the OffscreenCanvas API to achieve performance improvements when rendering graphics in your web app.
- [Ten modern layouts in one line of CSS](https://web.dev/articles/one-line-layouts.md.txt): This post highlights a few powerful lines of CSS that do some serious heavy lifting and help you build robust modern layouts.
- [Optimize Cumulative Layout Shift](https://web.dev/articles/optimize-cls.md.txt): Cumulative Layout Shift (CLS) is a metric that quantifies how often users experience sudden shifts in page content. In this guide, we&#39;ll cover optimizing common causes of CLS such as images and iframes without dimensions or dynamic content.
- [Optimize CSS background images with media queries](https://web.dev/articles/optimize-css-background-images-with-media-queries.md.txt): Use media queries to send images that are only as

large as they need to be, a technique commonly known as responsive images.
- [Optimize Core Web Vitals for business decision makers](https://web.dev/articles/optimize-cwv-business.md.txt): Learn how business decision makers and non-developers can improve Core Web Vitals.
- [Optimize Interaction to Next Paint](https://web.dev/articles/optimize-inp.md.txt): Learn how to optimize your website&#39;s Interaction to Next Paint.
- [Optimize input delay](https://web.dev/articles/optimize-input-delay.md.txt): Input delay can contribute significantly to total interaction latency and negatively affect your page&#39;s INP. In this guide, learn what input delay is, and how you can reduce it for faster interactivity.
- [Optimize JavaScript execution](https://web.dev/articles/optimize-javascript-execution.md.txt): JavaScript often triggers visual changes. Sometimes that&#39;s directly through style manipulations, and sometimes it&#39;s calculations that result in visual changes, like searching or sorting data. Badly-timed or long-running JavaScript is a common cause of performance issues. You should look to minimize its impact where you can.
- [Optimize Largest Contentful Paint](https://web.dev/articles/optimize-lcp.md.txt): A step-by-step guide on how to break down LCP and identify key areas to improve.
- [Optimize long tasks](https://web.dev/articles/optimize-long-tasks.md.txt): You&#39;ve been told &#34;don&#39;t block the main thread&#34; and &#34;break up your long tasks&#34;, but what does it mean to do those things?
- [Optimize Time to First Byte](https://web.dev/articles/optimize-ttfb.md.txt): Learn how to optimize for the Time to First Byte metric.
- [Optimizing Web Vitals using Lighthouse](https://web.dev/articles/optimize-vitals-lighthouse.md.txt): Today, we will cover new tooling features in Lighthouse, PageSpeed and DevTools to help identify how your site can improve on the Web Vitals.
- [Optimize WebFont loading and rendering](https://web.dev/articles/optimize-webfont-loading.md.txt): This post explains how to load WebFonts to prevent layout shifts and blank pages when WebFonts are not available when the page loads.
- [Eliminating Unnecessary Downloads](https://web.dev/articles/optimizing-content-efficiency-eliminate-downloads.md.txt): You should audit your resources periodically to ensure that each resource is helping deliver a better user experience.
- [JavaScript Start-up Optimization](https://web.dev/articles/optimizing-content-efficiency-javascript-startup-optimization.md.txt): Keep your network transmission and parse/compile cost for JavaScript low to ensure pages get interactive quickly.
- [Load Third-Party JavaScript](https://web.dev/articles/optimizing-content-efficiency-loading-third-party-javascript.md.txt): Third-party scripts provide a wide range of useful features, making the web more dynamic. Learn how to optimize the loading of third-party scripts to reduce their impact on performance.
- [Optimize the encoding and transfer size of text-based assets](https://web.dev/articles/optimizing-content-efficiency-optimize-encoding-and-transfer.md.txt): Next to eliminating unnecessary resource downloads, the best thing you can do to improve page load speed is to minimize the overall download size by optimizing and compressing the remaining resources.
- [Delivering Fast and Light Applications with Save-Data](https://web.dev/articles/optimizing-content-efficiency-save-data.md.txt): The Save-Data client hint request header available in Chrome, Opera, and Yandex browsers enables developers to deliver fast and light applications to users who have opted-in to &#39;data saving&#39; mode in the browser.
- [Orchestrating payment transactions with a service worker](https://web.dev/articles/orchestrating-payment-transactions.md.txt): Once a web-based payment app is registered, it&#39;s ready to accept payment requests from merchants. This article teaches you how to orchestrate a payment transaction from a service worker during runtime.
- [Requesting performance isolation with the Origin-Agent-Cluster header](https://web.dev/articles/origin-agent-cluster.md.txt): The Origin-Agent-Cluster header cuts off synchronous access to other origins on the same domain, and hints to the browser to give your origin dedicated resources.
- [The origin private file system](https://web.dev/articles/origin-private-file-system.md.txt): Guidance to build modern web experiences that work on any browser.
- [Fix an overloaded server](https://web.dev/articles/overloaded-server.md.txt): How to determine a server&#39;s bottleneck, quickly fix the bottleneck, improve server performance, and prevent regressions.
- [Using the Page Visibility API](https://web.dev/articles/pagevisibility-intro.md.txt): Guidance to build modern web experiences that work on any browser.
- [How to measure search engine optimization with Lighthouse](https://web.dev/articles/pass-

lighthouse-seo-audit.md.txt): The Lighthouse SEO audit scans your page, tests for things that matter to search engines, and gives you a score so you can see specific areas for improvement. SEO matters because it&#39;s how you get more relevant users viewing your content.
- [Secure and seamless passkeys: A deployment checklist](https://web.dev/articles/passkey-checklist.md.txt): A checklist for developers to make sure their passkey implementations are following all the best practices.
- [Sign in with a passkey through form autofill](https://web.dev/articles/passkey-form-autofill.md.txt): Create a sign in experience that leverages passkeys while still accommodating existing password users.
- [Designing the user experience of passkeys on Google accounts](https://web.dev/articles/passkey-google-ux.md.txt): Bringing better security and a better user experience to Google accounts.
- [Help users manage passkeys effectively](https://web.dev/articles/passkey-management.md.txt): These passkey management guidelines help you accomplish an intuitive, secure, and robust experience for your users.
- [Create a passkey for passwordless logins](https://web.dev/articles/passkey-registration.md.txt): Passkeys make a website&#39;s user accounts safer, simpler, easier to use and passwordless. This document discusses how to allow users to create passkeys for your website.
- [Payment and address form best practices](https://web.dev/articles/payment-and-address-form-best-practices.md.txt): Maximize conversions by helping your users complete address and payment forms as quickly and easily as possible.
- [Visual searching with the Web Perception Toolkit](https://web.dev/articles/perception-toolkit.md.txt): Wouldn&#39;t it be great if users could search your site using their camera?
- [Performance as a default with Next.js](https://web.dev/articles/performance-as-a-default-with-nextjs.md.txt): Guidance to build modern web experiences that work on any browser.
- [Auditing Performance](https://web.dev/articles/performance-audit.md.txt): Auditing your site or app will help you build a resilient, performant experience — and highlight quick wins that can be implemented with minimal sign-off. An audit also gives you a baseline for data-driven development. Did a change make things better? How does your site compare with competitors? You get metrics to prioritize effort, and concrete evidence to brag about once you&#39;ve made improvements.
- [Next steps](https://web.dev/articles/performance-audit-next.md.txt): Having completed a site audit, you should have accurate review data in a form that makes it easy for developers and other stakeholders to prioritize and justify changes.
- [Prework](https://web.dev/articles/performance-audit-prework.md.txt): Before gathering performance metrics for a site audit, there are several checks you can do to identify easy fixes and areas for focus. Some of these checks are relatively subjective, but can identify problems that affect perceived performance.
- [Check site security](https://web.dev/articles/performance-audit-security.md.txt): You won&#39;t be able to build a PWA without HTTPS. Serving your site over HTTPS is fundamental for security, and many APIs won&#39;t work without it. If you need to justify implementation costs, find out why HTTPS matters.
- [Share the results](https://web.dev/articles/performance-audit-share.md.txt): Once you&#39;ve audited a site, make sure to package the results in a digestible form. Be sensitive to the people on the receiving end of your audit, structure your report carefully and present your data in terms of opportunities and solutions.
- [Use tools to measure performance](https://web.dev/articles/performance-audit-tools.md.txt): There are several core objectives for building a performant, resilient site with low data cost. For each objective, you need an audit.
- [Performance budgets 101](https://web.dev/articles/performance-budgets-101.md.txt): Good performance is rarely a side effect. Learn about performance budgets and how they can set you on track for success.
- [Performance budgets with the Angular CLI](https://web.dev/articles/performance-budgets-with-the-angular-cli.md.txt): Learn how to use performance budgets directly in the Angular CLI!
- [Use forensics and detective work to solve JavaScript performance mysteries](https://web.dev/articles/performance-mystery.md.txt): Guidance to build modern web experiences that work on any browser.
- [Optimizing Content Efficiency](https://web.dev/articles/performance-optimizing-content-efficiency.md.txt): The amount of data downloaded by apps continues to increase over time. To deliver great performance you must optimize data delivery as much as possible.
- [Adapting to Users with Client Hints](https://web.dev/articles/performance-optimizing-content-

efficiency-client-hints.md.txt): Client hints are a set of HTTP request headers we can use to change how we deliver page resources based on characteristics of a user&#39;s device and network connection. In this article, you&#39;ll learn all about client hints, how they work, and a few ideas on how you can use them to make your site faster for users.
- [Understanding Low Bandwidth and High Latency](https://web.dev/articles/performance-poor-connectivity.md.txt): It&#39;s important to understand what using your app or site feels like when connectivity is poor or unreliable, and build accordingly. A range of tools can help you.
- [Web permissions best practices](https://web.dev/articles/permissions-best-practices.md.txt): This guide outlines best practices for websites to follow when asking users for permission to access sensitive capabilities (such as camera, microphone, and location) to minimize unnecessary prompts and blocked access.
- [Persistent storage](https://web.dev/articles/persistent-storage.md.txt): Persistent storage can help protect critical data from eviction, and reduce the chance of data loss.
- [Google Photography Prize Gallery](https://web.dev/articles/photographyprize.md.txt): Guidance to build modern web experiences that work on any browser.
- [Pointer lock and first person shooter controls](https://web.dev/articles/pointerlock-intro.md.txt): Guidance to build modern web experiences that work on any browser.
- [Porting USB applications to the web. Part 2: gPhoto2](https://web.dev/articles/porting-gphoto2-to-the-web.md.txt): Learn how gPhoto2 was ported to WebAssembly to control external cameras over USB from a web app.
- [Porting USB applications to the web. Part 1: libusb](https://web.dev/articles/porting-libusb-to-webusb.md.txt): Learn how code that interacts with external devices can be ported to the web with WebAssembly and Fugu APIs.
- [Practical prompt engineering for smaller LLMs](https://web.dev/articles/practical-prompt-engineering.md.txt): Learn how to adjust your prompts to achieve your preferred results across different LLMs, models, and model sizes.
- [Precaching in Create React App with Workbox](https://web.dev/articles/precache-with-workbox-react.md.txt): Workbox is built into Create React App with a default configuration that precaches all static assets in your application with every build.
- [Precaching with the Angular service worker](https://web.dev/articles/precaching-with-the-angular-service-worker.md.txt): Learn how to use the Angular service worker to precache the static assets in your app.
- [Establish network connections early to improve perceived page speed](https://web.dev/articles/preconnect-and-dns-prefetch.md.txt): Learn about rel=preconnect and rel=dns-prefetch resource hints and how to use them.
- [Faster web navigation with predictive prefetching](https://web.dev/articles/predictive-prefetching.md.txt): Code splitting allows you to speed up your applications, but it may slow down subsequent navigation. Predictive prefetching is an efficient way to use data analytics to smartly prefetch what the user is likely to use next, optimizing network utilization.
- [prefers-color-scheme: Hello darkness, my old friend](https://web.dev/articles/prefers-color-scheme.md.txt): Many devices now support an operating system wide dark mode or dark theme experience. This post explains how dark mode can be supported on web pages, lists best practices, and introduces a custom element named dark-mode-toggle that allows web developers to offer users a way to override their operating system level preference on specific web pages.
- [prefers-reduced-motion: Sometimes less movement is more](https://web.dev/articles/prefers-reduced-motion.md.txt): The prefers-reduced-motion media query detects whether the user has requested that the system minimize the amount of animation or motion it uses. This is for users who either require or prefer minimized animations; for example people with vestibular disorders often prefer animations to be kept to a minimum.
- [Preload critical assets to improve loading speed](https://web.dev/articles/preload-critical-assets.md.txt): As soon as you open any web page, the browser requests an HTML document from a server, parses the contents of the HTML file, and submits separate requests for any other external references. The critical request chain represents the order of resources that are prioritized and fetched by the browser.
- [Prevent layout shifting and flashes of invisible text (FOIT) by preloading optional fonts](https://web.dev/articles/preload-optional-fonts.md.txt): By optimizing rendering cycles, Chrome 83 eliminates layout shifting when preloading optional fonts. Combining with font-display: optional is the most effective way to guarantee jank-free rendering of custom fonts.
- [Preload responsive images](https://web.dev/articles/preload-responsive-images.md.txt): Learn about new and exciting possibilities for preloading responsive images to ensure great user experience.
- [Don&#39;t fight the browser preload scanner](https://web.dev/articles/preload-

scanner.md.txt): Find out what the browser preload scanner is, how it helps performance, and how you can stay out of its way.
- [Prepare media files for the web](https://web.dev/articles/prepare-media.md.txt): In this section you&#39;ll learn how to format video for mobile web playback, and how to create multiple files to cover a range of browsers, plus how to encrypt them.
- [Pre-render routes with react-snap](https://web.dev/articles/prerender-with-react-snap.md.txt): react-snap is a third-party library that pre-renders pages on your site into static HTML files. This can improve First Paint times in your application.
- [Profiling Web Audio apps in Chrome](https://web.dev/articles/profiling-web-audio-apps-in-chrome.md.txt): Learn how to profile the performance of Web Audio apps in Chrome using `about://tracing` and the **WebAudio** extension in Chrome DevTools.
- [Progressively enhance your Progressive Web App](https://web.dev/articles/progressively-enhance-your-pwa.md.txt): Learn how to progressively enhance your Progressive Web App so that it remains useful on all modern browsers, but delivers an advanced experience on browsers that support new web capabilities like file system access, system clipboard access, contacts retrieval, periodic background sync, screen wake lock, web sharing features, and many more.
- [JavaScript Promises: an introduction](https://web.dev/articles/promises.md.txt): Promises simplify deferred and asynchronous computations. A promise represents an operation that hasn&#39;t completed yet.
- [Patterns for promoting PWA installation](https://web.dev/articles/promote-install.md.txt): How to promote installation of Progressive Web Apps and best practices.
- [Photoshop&#39;s journey to the web](https://web.dev/articles/ps-on-the-web.md.txt): Over the last three years, Chrome has been working to empower web applications that want to push the boundaries of what&#39;s possible in the browser. One such web application has been Photoshop. The idea of running software as complex as Photoshop directly in the browser would have been hard to imagine just a few years ago. However, by using various new web technologies, Adobe has now brought a public beta of Photoshop to the web.
- (https://web.dev/articles/publish-modern-javascript.md.txt): Sorry, but this content is no longer available
- [Codelab: Build a push notification client](https://web.dev/articles/push-notifications-client-codelab.md.txt): A step-by-step interactive tutorial that shows you how to build a client that subscribes the user to push notifications, displays push messages as notifications, and unsubscribes the user from push notifications.
- [Common Issues and Reporting Bugs](https://web.dev/articles/push-notifications-common-issues-and-reporting-bugs.md.txt): There are right ways of using notifications, and ways of using them better. Learn what makes a good notification. We won&#39;t just show you what to do. We&#39;ll show you how to do it.
- [Common notification patterns](https://web.dev/articles/push-notifications-common-notification-patterns.md.txt): Guidance to build modern web experiences that work on any browser.
- [Displaying a Notification](https://web.dev/articles/push-notifications-display-a-notification.md.txt): Guidance to build modern web experiences that work on any browser.
- [Push notifications FAQ](https://web.dev/articles/push-notifications-faq.md.txt): Guidance to build modern web experiences that work on any browser.
- [Push events](https://web.dev/articles/push-notifications-handling-messages.md.txt): Guidance to build modern web experiences that work on any browser.
- [How push works](https://web.dev/articles/push-notifications-how-push-works.md.txt): Guidance to build modern web experiences that work on any browser.
- [Notification behavior](https://web.dev/articles/push-notifications-notification-behaviour.md.txt): Guidance to build modern web experiences that work on any browser.
- [Push notifications overview](https://web.dev/articles/push-notifications-overview.md.txt): An overview of what push notifications are, why you might use them, and how they work.
- [Permission UX](https://web.dev/articles/push-notifications-permissions-ux.md.txt): Guidance to build modern web experiences that work on any browser.
- [Codelab: Build a push notification server](https://web.dev/articles/push-notifications-server-codelab.md.txt): A step-by-step interactive tutorial that shows you how to build a server that manages push notification subscriptions and sends web push protocol requests to a push service.
- [Subscribing a User](https://web.dev/articles/push-notifications-subscribing-a-user.md.txt): Guidance to build modern web experiences that work on any browser.
- [Basics of UX](https://web.dev/articles/push-notifications-video.md.txt): Guidance to build modern web experiences that work on any browser.

- [The Web Push Protocol](https://web.dev/articles/push-notifications-web-push-protocol.md.txt): A step-by-step interactive tutorial that shows you how to build a server that manages push notification subscriptions and sends web push protocol requests to a push service.
- [What makes a good Progressive Web App?](https://web.dev/articles/pwa-checklist.md.txt): What makes a good, or great Progressive Web App?
- [Address Bar Install for Progressive Web Apps on the Desktop](https://web.dev/articles/pwa-install-addressbar.md.txt): Progressive Web Apps are easy to install with a new install button in Chrome’s address bar (omnibox).
- [How do I notify users that my PWA is installable?](https://web.dev/articles/pwa-install-patterns.md.txt): How to promote installation of Progressive Web Apps and best practices.
- [PWA with offline streaming](https://web.dev/articles/pwa-with-offline-streaming.md.txt): Building a PWA with offline streaming has its challenges. In this article you will learn about the APIs and techniques that provide users with a high-quality offline media experience.
- [PWAs in app stores](https://web.dev/articles/pwas-in-app-stores.md.txt): Progressive Web Apps can be submitted to app stores like the Android Play Store or the Microsoft Store and more.
- [PWAs on Oculus Quest 2](https://web.dev/articles/pwas-on-oculus-2.md.txt): The Oculus Quest 2 is a virtual reality (VR) headset created by Oculus, a division of Meta. Developers can now build and distribute 2D and 3D Progressive Web Apps (PWA) that take advantage of Oculus Quest 2&#39;s multitasking feature. This article describes the experience and how to build, sideload, and test your PWA on the Oculus Quest 2.
- [Quarantine your site](https://web.dev/articles/quarantine-your-site.md.txt): Take your site offline while you fix the problems.
- [Speed up navigations in React with Quicklink](https://web.dev/articles/quicklink.md.txt): quicklink is a library to achieve faster subsequent page-loads by prefetching in-viewport links during idle time.
- [Measure performance with the RAIL model](https://web.dev/articles/rail.md.txt): RAIL model enables designers and developers to reliably target the performance optimization work that has the highest impact on user experience. Learn what goals and guidelines the RAIL model sets out and which tools you can use to achieve them.
- [Introduction to Raphaël.js](https://web.dev/articles/raphael.md.txt): Guidance to build modern web experiences that work on any browser.
- [Read files in JavaScript](https://web.dev/articles/read-files.md.txt): How to select files, read file metadata and content, and monitor read progress.
- [Ready Player Web](https://web.dev/articles/ready-player-web.md.txt): The Web platform is very mature for game development nowadays. The key to building a modern web game is embracing the best practices of game design and monetization. This post provides provides guidance towards this goal.
- [Reduce JavaScript payloads with code splitting](https://web.dev/articles/reduce-javascript-payloads-with-code-splitting.md.txt): Sending large JavaScript payloads impacts the speed of your site significantly. Instead of shipping all the JavaScript to your user as soon as the first page of your application is loaded, split your bundle into multiple pieces and only send what&#39;s necessary at the very beginning.
- [Reduce JavaScript payloads with tree shaking](https://web.dev/articles/reduce-javascript-payloads-with-tree-shaking.md.txt): Knowing where to begin optimizing your application&#39;s JavaScript can be daunting. If you&#39;re taking advantage of modern tooling such as webpack, however, tree shaking might be a good place to start!
- [Reduce the scope and complexity of style calculations](https://web.dev/articles/reduce-the-scope-and-complexity-of-style-calculations.md.txt): Badly-timed or long-running JavaScript that triggers visual changes on your site can be a common cause of performance issues. Minimize its impact where you can.
- [Reduce web font size](https://web.dev/articles/reduce-webfont-size.md.txt): This post explains how to reduce the size of the WebFonts that you use on your site, so that good typography doesn&#39;t mean a slow site.
- [Referer and Referrer-Policy best practices](https://web.dev/articles/referrer-best-practices.md.txt): Consider setting a referrer policy of `strict-origin-when-cross-origin`. It retains much of the referrer&#39;s usefulness, while mitigating the risk of leaking data cross-origins.
- [Registering a custom protocol handler](https://web.dev/articles/registering-a-custom-protocol-handler.md.txt): Guidance to build modern web experiences that work on any browser.
- [Registering a web-based payment app](https://web.dev/articles/registering-a-web-based-payment-app.md.txt): Learn how to register a web-based payment app to a customers&#39; browser. You&#39;ll also learn how to debug them.

- [Remove unused code](https://web.dev/articles/remove-unused-code.md.txt): Analyze your JavaScript bundle to detect and remove unused code.
- [Rendering on the Web](https://web.dev/articles/rendering-on-the-web.md.txt): Recommendations for implementing logic and rendering in apps.
- [Rendering performance](https://web.dev/articles/rendering-performance.md.txt): Users notice if sites and apps don&#39;t run well, so optimizing rendering performance is crucial!
- [Replace animated GIFs with video for faster page loads](https://web.dev/articles/replace-gifs-with-videos.md.txt): Have you ever seen an animated GIF on a service like Imgur or Gfycat, inspected it in your dev tools, only to find out that GIF was really a video? There&#39;s a good reason for that. Animated GIFs can be downright huge! By converting large GIFs to videos, you can save big on users&#39; bandwidth.
- [Request a review](https://web.dev/articles/request-a-review.md.txt): You must request a review from Google to have your page or site unflagged as dangerous or possibly deceptive to users.
- [requestAutocomplete](https://web.dev/articles/requestautocomplete.md.txt): Guidance to build modern web experiences that work on any browser.
- [Perform efficient per-video-frame operations on video with requestVideoFrameCallback()](https://web.dev/articles/requestvideoframecallback-rvfc.md.txt): The requestVideoFrameCallback() method allows web authors to register a callback that runs in the rendering steps when a new video frame is sent to the compositor.
- [ResizeObserver: it&#39;s like document.onresize for elements](https://web.dev/articles/resize-observer.md.txt): `ResizeObserver` notifies you when an element&#39;s content rectangle changes size so that you can react accordingly.
- [Responsive images](https://web.dev/articles/responsive-images.md.txt): A picture is worth 1000 words, and images play an integral part of every page. But they also often account for most of the downloaded bytes. With responsive web design not only can our layouts change based on device characteristics, but images as well.
- [Responsive web design basics](https://web.dev/articles/responsive-web-design-basics.md.txt): Create sites that respond to the needs and capabilities of the device they&#39;re viewed on.
- [Route-level code splitting in Angular](https://web.dev/articles/route-level-code-splitting-in-angular.md.txt): Learn how to make your initial app bundle smaller by using route-level code splitting.
- [Route prefetching in Next.js](https://web.dev/articles/route-prefetching-in-nextjs.md.txt): Guidance to build modern web experiences that work on any browser.
- [Route preloading strategies in Angular](https://web.dev/articles/route-preloading-in-angular.md.txt): Learn how to use Angular&#39;s preloading strategies for faster apps.
- [Same-origin policy](https://web.dev/articles/same-origin-policy.md.txt): A browser can load and display resources from multiple sites. If there is no restriction on interactions between those resources, and if a script is compromised by an attacker, the script could expose everything on a user&#39;s browser.
- [&quot;Same-site&quot; and &quot;same-origin&quot;](https://web.dev/articles/same-site-same-origin.md.txt): &#34;same-site&#34; and &#34;same-origin&#34; are frequently cited but often misunderstood terms. This page explains what they are and how they are different.
- [SameSite cookie recipes](https://web.dev/articles/samesite-cookie-recipes.md.txt): Sites can now explicitly mark their cookies for cross-site usage. Learn how to mark up your cookies to ensure that your first-party and third-party cookies continue to work after this change is implemented.
- [SameSite cookies explained](https://web.dev/articles/samesite-cookies-explained.md.txt): Learn to mark your cookies for first-party and third-party usage with the SameSite attribute. You can enhance your site&#39;s security by using SameSite&#39;s Lax and Strict values to improve protection against CSRF attacks. Specifying the new None attribute lets you explicitly mark your cookies for cross-site usage.
- [Play safely in sandboxed IFrames](https://web.dev/articles/sandboxed-iframes.md.txt): Guidance to build modern web experiences that work on any browser.
- [Safe DOM manipulation with the Sanitizer API](https://web.dev/articles/sanitizer.md.txt): The new Sanitizer API aims to build a robust processor for arbitrary strings to be safely inserted into a page. This article introduces the API, and explains its usage.
- [Scaling multithreaded WebAssembly applications with mimalloc and WasmFS](https://web.dev/articles/scaling-multithreaded-webassembly-applications.md.txt): WasmFS and the mimalloc feature in Emscripten can help a lot with allocation and I/O performance. This guide shows how these features can lead to speed improvements of 10 times or more in some cases.
- [Securely hosting user data in modern web applications](https://web.dev/articles/securely-

hosting-user-data.md.txt): How to securely display user-controlled content on web applications.
- [What are security attacks?](https://web.dev/articles/security-attacks.md.txt): An insecure application could expose users and systems to various types of damage. When a malicious party uses vulnerabilities or lack of security features to their advantage to cause damage, it is called an attack. We&#39;ll take a look at different types of attacks in this guide so you know what to look for when securing your application.
- [The Credential Management API](https://web.dev/articles/security-credential-management.md.txt): Guidance to build modern web experiences that work on any browser.
- [Sign in Users](https://web.dev/articles/security-credential-management-retrieve-credentials.md.txt): Guidance to build modern web experiences that work on any browser.
- [Save Credentials from Forms](https://web.dev/articles/security-credential-management-save-forms.md.txt): Guidance to build modern web experiences that work on any browser.
- [Security headers quick reference](https://web.dev/articles/security-headers.md.txt): This article lists the most important security headers you can use to protect your website. Use it to understand web-based security features, learn how to implement them on your website, and as a reference for when you need a reminder.
- [Security should not be so scary!](https://web.dev/articles/security-not-scary.md.txt): When the word &#34;security&#34; comes to mind, it&#39;s usually in the context of bad news. But security is something to be taken as a positive and necessary part of web development just like &#34;user experience&#34; or &#34;accessibility&#34;.
- [Semantics and screen readers](https://web.dev/articles/semantics-and-screen-readers.md.txt): Have you ever stopped to wonder how assistive technology, such as a screen reader, knows what to announce to users? The answer is that these technologies rely on developers marking up their pages with semantic HTML. But what are semantics, and how do screen readers use them?
- [Introduction to ARIA](https://web.dev/articles/semantics-aria.md.txt): Introduction to ARIA and non-native HTML semantics
- [Introduction to semantics](https://web.dev/articles/semantics-builtin.md.txt): Introduction to semantics and assistive technology
- [Use WebP images](https://web.dev/articles/serve-images-webp.md.txt): WebP images are smaller than their JPEG and PNG counterparts—usually on the magnitude of a 25–35% reduction in filesize. This decreases page sizes and improves performance.
- [Serve images with correct dimensions](https://web.dev/articles/serve-images-with-correct-dimensions.md.txt): We&#39;ve all been there—you forgot to scale down an image before adding it to the page. The image looks fine, but it is wasting users&#39; data and hurting page performance.
- [Serve modern code to modern browsers for faster page loads](https://web.dev/articles/serve-modern-code-to-modern-browsers.md.txt): Building websites that work well on all major browsers is a core tenet of an open web ecosystem. However, this means additional work of ensuring that all of the code you write is supported in each browser that you plan to target. If you want to use new JavaScript language features, you need to transpile these features to backwards-compatible formats.
- [Serve responsive images](https://web.dev/articles/serve-responsive-images.md.txt): Serving desktop-sized images to mobile devices can use 2–4x more data than needed. Instead of a &#34;one-size-fits-all&#34; approach to images, serve different image sizes to different devices.
- [Service worker caching and HTTP caching](https://web.dev/articles/service-worker-caching-and-http-caching.md.txt): The pros and cons of using consistent or different expiry logic across the service worker cache and HTTP cache layers.
- [The service worker lifecycle](https://web.dev/articles/service-worker-lifecycle.md.txt): A deep-dive into the service worker lifecycle.
- [Service worker mindset](https://web.dev/articles/service-worker-mindset.md.txt): Working with service workers is new and unfamiliar for many web devs. This post provides some tips for wrapping your mind around them.
- [Service workers and the Cache Storage API](https://web.dev/articles/service-workers-cache-storage.md.txt): The browser&#39;s HTTP cache is your first line of defense. It&#39;s not necessarily the most powerful or flexible approach, and you have limited control over the lifetime of cached responses. But there are several rules of thumb that give you a sensible caching implementation without much work, so you should always try to follow them.
- [Service worker registration](https://web.dev/articles/service-workers-registration.md.txt): Best practices for timing your service worker registration.
- [Setting up a payment method](https://web.dev/articles/setting-up-a-payment-method.md.txt): A payment transaction using Web Payments starts with the discovery of your payment app. Learn how

to set up a payment method and get your payment app ready for merchants and customers to make payments.
- [Shadow DOM 101](https://web.dev/articles/shadowdom.md.txt): Guidance to build modern web experiences that work on any browser.
- [Shadow DOM 201](https://web.dev/articles/shadowdom-201.md.txt): Guidance to build modern web experiences that work on any browser.
- [Shadow DOM 301](https://web.dev/articles/shadowdom-301.md.txt): Guidance to build modern web experiences that work on any browser.
- [Shadow DOM v1 - Self-Contained Web Components](https://web.dev/articles/shadowdom-v1.md.txt): Shadow DOM allows web developers to create compartmentalized DOM and CSS for web components
- [Getting Started with CSS Shapes](https://web.dev/articles/shapes-getting-started.md.txt): Guidance to build modern web experiences that work on any browser.
- [Sign-in form best practices](https://web.dev/articles/sign-in-form-best-practices.md.txt): Use cross-platform browser features to build sign-in forms that are secure, accessible and easy to use.
- [What makes for a good sign-out experience?](https://web.dev/articles/sign-out-best-practices.md.txt): Practical developer guidance about what to do when a user logs out of the website.
- [Sign-up form best practices](https://web.dev/articles/sign-up-form-best-practices.md.txt): Help your users sign up, sign in and manage their account details with a minimum of fuss.
- [Signed Exchanges (SXGs)](https://web.dev/articles/signed-exchanges.md.txt): An SXG is a delivery mechanism that makes it possible to authenticate the origin of a resource independently of how it was delivered.
- [How to set up Signed Exchanges using Web Packager](https://web.dev/articles/signed-exchanges-webpackager.md.txt): Learn how to serve signed exchanges (SXGs) using Web Packager.
- [Simplify paint complexity and reduce paint areas](https://web.dev/articles/simplify-paint-complexity-and-reduce-paint-areas.md.txt): Paint is the process of filling in pixels that eventually get composited to the users&#39; screens. It is often the longest-running of all tasks in the pipeline, and one to avoid if at all possible.
- [Relating site speed and business metrics](https://web.dev/articles/site-speed-and-business-metrics.md.txt): Leverage A/B testing to evaluate the impact of site speed on your business metrics.
- [Towards an animation smoothness metric](https://web.dev/articles/smoothness.md.txt): Learn about measuring animations, how to think about animation frames, and overall page smoothness.
- [SMS OTP form best practices](https://web.dev/articles/sms-otp-form.md.txt): Asking a user to provide an OTP (one-time password) delivered via SMS is a common way to confirm a user&#39;s phone number. This post provides you with the best practices to build an SMS OTP form with great user experience.
- [Scroll snapping after layout changes](https://web.dev/articles/snap-after-layout.md.txt): Starting in Chrome 81, scrollers remain snapped when the page layout changes. In other words, you no longer need to add event listeners to force resnapping.
- [What are source maps?](https://web.dev/articles/source-maps.md.txt): Improve web debugging experience with source maps.
- [Avoiding Unnecessary Paints - Animated GIF Edition](https://web.dev/articles/speed-animated-gifs.md.txt): Guidance to build modern web experiences that work on any browser.
- [Speed at scale: what&#39;s new in web performance?](https://web.dev/articles/speed-at-scale.md.txt): For Google I/O 2019, we introduced three new Web Performance initiatives that we hope will lead to better user experiences for everyone.
- [Improving the performance of your HTML5 App](https://web.dev/articles/speed-html5.md.txt): Guidance to build modern web experiences that work on any browser.
- [Accelerated Rendering in Chrome](https://web.dev/articles/speed-layers.md.txt): Guidance to build modern web experiences that work on any browser.
- [Parallaxin&#39;](https://web.dev/articles/speed-parallax.md.txt): Guidance to build modern web experiences that work on any browser.
- [Best practices for a faster web app with HTML5](https://web.dev/articles/speed-quick.md.txt): Guidance to build modern web experiences that work on any browser.
- [Jank busting for better rendering performance](https://web.dev/articles/speed-rendering.md.txt): Guidance to build modern web experiences that work on any browser.
- [Deep dive into the murky waters of script loading](https://web.dev/articles/speed-script-loading.md.txt): Guidance to build modern web experiences that work on any browser.
- [Static memory javascript with Object Pools](https://web.dev/articles/speed-static-mem-pools.md.txt): Guidance to build modern web experiences that work on any browser.

- [Speed tooling evolutions: highlights from Chrome Developer Summit 2019](https://web.dev/articles/speed-tooling-evolutions-cds-2019.md.txt): Read about the latest developments in speed tooling including new performance metrics, updates to PageSpeed Insights and Chrome User Experience Report (CrUX), and insights from Web Almanac analysis of the web ecosystem.
- [How To Think About Speed Tools](https://web.dev/articles/speed-tools.md.txt): How To Think About Speed Tools
- [Avoiding unnecessary paints](https://web.dev/articles/speed-unnecessary-paints.md.txt): Guidance to build modern web experiences that work on any browser.
- [Performance tips for JavaScript in V8](https://web.dev/articles/speed-v8.md.txt): Guidance to build modern web experiences that work on any browser.
- [Speedy CSS Tip! Animated Gradient Text](https://web.dev/articles/speedy-css-tip-animated-gradient-text.md.txt): Guidance to build modern web experiences that work on any browser.
- [Speedy CSS Tip! Animated Loader](https://web.dev/articles/speedy-css-tip-animated-loader.md.txt): Guidance to build modern web experiences that work on any browser.
- [Keeping things fresh with stale-while-revalidate](https://web.dev/articles/stale-while-revalidate.md.txt): stale-while-revalidate helps developers balance between immediacy—loading cached content right away—and freshness—ensuring updates to the cached content are used in the future.
- [Stick to Compositor-Only Properties and Manage Layer Count](https://web.dev/articles/stick-to-compositor-only-properties-and-manage-layer-count.md.txt): Compositing is where the painted parts of the page are put together for displaying on screen.
- [Storage for the web](https://web.dev/articles/storage-for-the-web.md.txt): There are many different options for storing data in the browser. Which one is best for your needs?
- [Screensharing a browser tab in HTML5?](https://web.dev/articles/streaming-screenshare.md.txt): Guidance to build modern web experiences that work on any browser.
- [Streams—The definitive guide](https://web.dev/articles/streams.md.txt): The Streams API allows JavaScript to programmatically access streams of data received over the network and process them as desired.
- [Mitigate cross-site scripting (XSS) with a strict Content Security Policy (CSP)](https://web.dev/articles/strict-csp.md.txt): Learn how to deploy a CSP based on script nonces or hashes as a defense-in-depth against cross-site scripting.
- [Deep-copying in JavaScript using structuredClone](https://web.dev/articles/structured-clone.md.txt): For the longest time, you had to resort to workarounds and libraries to create a deep copy of a JavaScript value. The Platform now ships with `structuredClone()`, a built-in function for deep-copying.
- [Style focus](https://web.dev/articles/style-focus.md.txt): The focus indicator (often signified by a &#34;focus ring&#34;) identifies the currently focused element. For users who are unable to use a mouse, this indicator is extremely important, as it acts as a stand-in for their mouse-pointer.
- [Splash vector graphics on your responsive site](https://web.dev/articles/svg-mobile-fundamentals.md.txt): Guidance to build modern web experiences that work on any browser.
- [Handling range requests in a service worker](https://web.dev/articles/sw-range-requests.md.txt): Make sure your service worker knows what to do when a partial response is requested.
- [Synchronized cross-device mobile testing](https://web.dev/articles/synchronized-cross-device-testing.md.txt): Guidance to build modern web experiences that work on any browser.
- [Four common types of code coverage](https://web.dev/articles/ta-code-coverage.md.txt): Learn what code coverage is and discover four common ways to measure it.
- [Pyramid or Crab? Find a testing strategy that fits](https://web.dev/articles/ta-strategies.md.txt): Discover how to combine different testing types into a reasonable strategy that matches your project.
- [Defining test cases and priorities](https://web.dev/articles/ta-test-cases.md.txt): Determine what to test, define your test cases, and prioritize.
- [Three common types of test automation](https://web.dev/articles/ta-types.md.txt): Let&#39;s start with the basics! Exploring the two general testing modes and three common types of test automation.
- [To test or not to test, a technical perspective](https://web.dev/articles/ta-what-to-test.md.txt): Determine what you need to test and what you can rule out.
- [Best practices for tags and tag managers](https://web.dev/articles/tag-best-practices.md.txt): Optimize tags and tag managers for Core Web Vitals.
- [Total Blocking Time (TBT)](https://web.dev/articles/tbt.md.txt): This post describes the

Total Blocking Time (TBT) metric and explains how to measure it
- [The Front-End Developer&#39;s Guide to the Terminal](https://web.dev/articles/terminal-for-js-devs.md.txt): This resource can help you quickly find your way around the terminal.
- [Compare LLM capability with summarization](https://web.dev/articles/test-llm-capabilities.md.txt): Evaluate the results of different models and prompts with the LLM as a judge technique. Instead of relying on human judgment, model validation is delegated to another LLM.
- [Testing Web Design Color Contrast](https://web.dev/articles/testing-web-design-color-contrast.md.txt): An overview of three tools and techniques for testing and verifying accessible color contrast of your design.
- [Text Alternatives for Images](https://web.dev/articles/text-alternatives-for-images.md.txt): Using the alt attribute to provide text alternatives for images
- [Boldly link where no one has linked before: Text Fragments](https://web.dev/articles/text-fragments.md.txt): Text Fragments let you specify a text snippet in the URL fragment. When navigating to a URL with such a text fragment, the browser can emphasize and/or bring it to the user&#39;s attention.
- [The Accessibility Tree](https://web.dev/articles/the-accessibility-tree.md.txt): Introduction to the Accessibility Tree
- [The amazing powers of CSS](https://web.dev/articles/the-amazing-powers-of-css.md.txt): Guidance to build modern web experiences that work on any browser.
- [The Basics of easing](https://web.dev/articles/the-basics-of-easing.md.txt): Learn how to soften and give weighting to your animations.
- [The end of Internet Explorer](https://web.dev/articles/the-end-of-ie.md.txt): What ending support for Internet Explorer meant for the customers and developers at Maersk.com.
- [Third-party JavaScript performance](https://web.dev/articles/third-party-javascript.md.txt): This post describes the common kinds of third-party JavaScript and the performance issues they can cause. It also provides general guidance about how to optimize third-party scripts.
- [What are third-party origin trials?](https://web.dev/articles/third-party-origin-trials.md.txt): Guidance to build modern web experiences that work on any browser.
- [Getting started with Three.js](https://web.dev/articles/three-intro.md.txt): Guidance to build modern web experiences that work on any browser.
- [The most effective ways to improve Core Web Vitals](https://web.dev/articles/top-cwv.md.txt): A collection of best practices that Chrome has identified as the biggest opportunities to optimize web performance and improve Core Web Vitals
- [Top ways sites get hacked by spammers](https://web.dev/articles/top-ways-sites-get-hacked-by-spammers.md.txt): Find out the ways that spammers can compromise your site.
- [Confound malicious middlemen with HTTPS and HTTP Strict transport security](https://web.dev/articles/transport-layer-security.md.txt): Guidance to build modern web experiences that work on any browser.
- [A Simple Trip Meter using the Geolocation API](https://web.dev/articles/trip-meter.md.txt): Guidance to build modern web experiences that work on any browser.
- [Prevent DOM-based cross-site scripting vulnerabilities with Trusted Types](https://web.dev/articles/trusted-types.md.txt): Introducing Trusted Types: a browser API to prevent DOM-based cross-site scripting in modern web applications.
- [Time to First Byte (TTFB)](https://web.dev/articles/ttfb.md.txt): This post introduces the Time to First Byte (TTFB) metric and explains how to measure it.
- [Time to Interactive (TTI)](https://web.dev/articles/tti.md.txt): This post introduces the Time to Interactive (TTI) metric and explains how to measure it
- [Two-way communication with service workers](https://web.dev/articles/two-way-communication-guide.md.txt): How establish a two-way communication channel between the page and the service worker.
- [The UI fund](https://web.dev/articles/ui-fund.md.txt): Announcing the UI fund from Chrome, designed to provide grants for people who work on design tools, CSS, and HTML.
- [Understanding cookies](https://web.dev/articles/understanding-cookies.md.txt): Learn about how cookies work and what are first-party and third-party cookies.
- [Understanding CSS filter effects](https://web.dev/articles/understanding-css.md.txt): Guidance to build modern web experiences that work on any browser.
- [What are the parts of a URL?](https://web.dev/articles/url-parts.md.txt): What&#39;s the difference between a host, site and origin? What is an eTLD&#43;1? This article explains.
- [Use Baseline with Browserslist](https://web.dev/articles/use-baseline-with-browserslist.md.txt): Add Baseline to your development linting and packaging tools with browserslist-config-baseline.

- [Use Imagemin to compress images](https://web.dev/articles/use-imagemin-to-compress-images.md.txt): Uncompressed images bloat your pages with unnecessary bytes. Run Lighthouse to check for opportunities to improve page load by compressing images.
- [Use Lighthouse for performance budgets](https://web.dev/articles/use-lighthouse-for-performance-budgets.md.txt): Lighthouse now supports performance budgets. This feature, LightWallet, can be set up in under five minutes and provides feedvack on the size and quantity of page resources.
- [Make use of long-term caching](https://web.dev/articles/use-long-term-caching.md.txt): How webpack helps with asset caching
- [Use push notifications to engage and re-engage users](https://web.dev/articles/use-push-notifications-to-engage-users.md.txt): Use push notifications to engage users with targeted, timely updates.
- [Use Search Console](https://web.dev/articles/use-search-console.md.txt): Verify ownership then check your site using Search Console.
- [Use semantic HTML for easy keyboard wins](https://web.dev/articles/use-semantic-html.md.txt): By using the correct semantic HTML elements you may be able to meet most or all of your keyboard access needs. That means less time fiddling with tabindex, and more happy users!
- [Top tips for web performance](https://web.dev/articles/use-srcset-to-automatically-choose-the-right-image.md.txt): Use srcset to automatically choose the right image size.
- [User-centric performance metrics](https://web.dev/articles/user-centric-performance-metrics.md.txt): User-centric performance metrics are a critical tool in understanding and improving the experience of your site in a way that benefits real users.
- [User Location](https://web.dev/articles/user-location.md.txt): Most browsers and devices have access to the user&#39;s geographic location. Learn how to work with the user&#39;s location in your site and apps.
- [User preference media features client hints headers](https://web.dev/articles/user-preference-media-features-headers.md.txt): Guidance to build modern web experiences that work on any browser.
- [The :user-valid and :user-invalid pseudo-classes](https://web.dev/articles/user-valid-and-user-invalid-pseudo-classes.md.txt): About the :user-valid and :user-invalid pseudo-classes and how to use them to improve the user experience of input validation.
- [User timing API](https://web.dev/articles/usertiming.md.txt): Guidance to build modern web experiences that work on any browser.
- [Using a PWA in your Android app](https://web.dev/articles/using-a-pwa-in-your-android-app.md.txt): How to open a Progressive Web App in an Android app.
- [Using bundlesize with Travis CI](https://web.dev/articles/using-bundlesize-with-travis-ci.md.txt): Define performance budgets with minimal setup and enforce them as part of your development workflow using bundlesize with Travis CI.
- [Using Lighthouse Bot to set a performance budget](https://web.dev/articles/using-lighthouse-bot-to-set-a-performance-budget.md.txt): You&#39;ve done hard work to get fast, now make sure you stay fast by automating performance testing in Travis CI with Lighthouse Bot.
- [Using tabindex](https://web.dev/articles/using-tabindex.md.txt): Use the tabindex attribute to explicitly set an element&#39;s tab position.
- [Basics of UX](https://web.dev/articles/ux-basics.md.txt): A step-by-step guide to the basics of UX design.
- [The value of speed](https://web.dev/articles/value-of-speed.md.txt): Demonstrate the revenue generated by site improvements while excluding external factors such as marketing campaigns.
- [Introduction to variable fonts on the web](https://web.dev/articles/variable-fonts.md.txt): How variable fonts work, how typographers implement variable fonts, and how to work with variable fonts in CSS.
- [Variable fonts in real life](https://web.dev/articles/variable-fonts-in-real-life.md.txt): Sharing a practical guide to variable fonts, with lots of examples.
- [The &lt;video&gt; and &lt;source&gt; tags](https://web.dev/articles/video-and-source-tags.md.txt): You&#39;ve properly prepared a video file for the web. You&#39;ve given it correct dimensions and the correct resolution. You&#39;ve even created separate WebM and MP4 files for different browsers. For anyone to see it, you still need to add it to a web page.
- [Going beyond images with basic video for the web](https://web.dev/articles/video-basics.md.txt): Research shows that web video lead to higher engagement and sales. Even if you haven&#39;t added video to your sites yet, it&#39;s just a matter of time until you do.
- [Virtualize large lists with the Angular CDK](https://web.dev/articles/virtualize-lists-with-angular-cdk.md.txt): Learn how to make large lists more responsive by implementing virtual scrolling with the Angular Component Dev Kit.

- [Virtualize large lists with react-window](https://web.dev/articles/virtualize-long-lists-react-window.md.txt): react-window is a library that allows large lists to be rendered efficiently.
- [Web Vitals](https://web.dev/articles/vitals.md.txt): Essential metrics for a healthy site
- [Measure and debug performance with Google Analytics 4 and BigQuery](https://web.dev/articles/vitals-ga4.md.txt): Learn how to send Web Vitals data to Google Analytics 4 properties and export the data for analysis in BigQuery and Looker Studio.
- [Getting started with measuring Web Vitals](https://web.dev/articles/vitals-measurement-getting-started.md.txt): Learn how to measure your site&#39;s Web Vitals in both real-world and lab environments.
- [How SPA architectures affect Core Web Vitals](https://web.dev/articles/vitals-spa-faq.md.txt): Answers to common questions about SPAs, Core Web Vitals, and Google&#39;s plan to address current measurement limitations.
- [Core Web Vitals workflows with Google tools](https://web.dev/articles/vitals-tools.md.txt): With the growing importance of Core Web Vitals, site owners and developers increasingly focus on performance and key user experiences. Google provides many tools to help evaluate, optimize, and monitor pages, but users are often confused by the different sources of data and how to use them effectively. This guide proposes a workflow combining several tools and clarifies where and how they make sense along the development process.
- [Virtual reality comes to the web](https://web.dev/articles/vr-comes-to-the-web.md.txt): Virtual reality came to the web in Chrome 79. Based on the WebXR Devicer API, this launch is the foundation for both augmented and virtual reality. This article is the first in a series, exploring basic concepts and describing how to enter an XR session. Other browsers will soon be supporting the WebXR Device API, including Firefox Reality, Oculus Browser, Edge and Magic Leap&#39;s Helio browser, among others.
- [Virtual reality comes to the web, part II](https://web.dev/articles/vr-comes-to-the-web-pt-ii.md.txt): Virtual reality came to the web in Chrome 79. Based on the WebXR Device API, this launch is the foundation for both augmented and virtual reality. This article is the second in a series, focusing on the frame loop, the part of an XR session where images are shown to a viewer. Other browsers will soon be supporting the WebXR Device API, including Firefox Reality, Oculus Browser, Edge and Magic Leap&#39;s Helio browser, among others.
- [Extending the browser with WebAssembly](https://web.dev/articles/wasm-av1.md.txt): WebAssembly lets us extend the browser with new features. This article shows how to port the AV1 video decoder and play AV1 video in any modern browser.
- [New functionality for developers—brought to you by WebAssembly](https://web.dev/articles/wasm-libraries.md.txt): A showcase of tools now available on the web thanks to WebAssembly.
- [WebAssembly Threads ready to try in Chrome 70](https://web.dev/articles/wasm-threads.md.txt): WebAssembly thread support has shipped in Chrome 70 under an origin-trial.
- [Augmented reality: You may already know it](https://web.dev/articles/web-ar.md.txt): If you&#39;ve already used the WebXR Device API, you&#39;ll be happy to know there&#39;s very little new to learn. Entering a WebXR session is largely the same. Running a frame loop is largely the same. The differences lie in configurations that allow content to be shown appropriately for augmented reality.
- [Web-based payment apps overview](https://web.dev/articles/web-based-payment-apps-overview.md.txt): Learn how to adapt your web-based payment app to work with Web Payments and provide a better user experience for customers.
- [Building components](https://web.dev/articles/web-components.md.txt): Components are the building blocks of modern web applications. What best practices should you follow when building your own components so they can stand the test of time?
- [Web components: the secret ingredient helping power the web](https://web.dev/articles/web-components-io-2019.md.txt): This post sums up a talk on the state of web components in 2019, given by Kevin Schaaf of the Polymer Project and Caridy Patiño of Salesforce.
- [web.dev at I/O 2019](https://web.dev/articles/web-dev-io-2019.md.txt): For Google I/O 2019, the folks on the web.dev team have shipped a number of updates including a refreshed design, more Lighthouse docs, and a brand new blog.
- [Web on Android](https://web.dev/articles/web-on-android.md.txt): Learn how different components can be used to render web content inside Android apps.
- [Fill OTP forms within cross-origin iframes with WebOTP API](https://web.dev/articles/web-otp-iframe.md.txt): WebOTP API can now receive an OTP from within an iframe.
- [Web Payments overview](https://web.dev/articles/web-payments-overview.md.txt): Learn more about Web Payments and how they work.

- [Updates to the Web Payments APIs](https://web.dev/articles/web-payments-updates.md.txt): Since the launch of the Payment Request API in Chrome 53 and the Payment Handler API in Chrome 68, there have been quite a few changes made to their respective specifications. This post summarizes those updates and will continue accumulating those API changes.
- [Web Performance Made Easy - Google I/O 2018 edition](https://web.dev/articles/web-performance-made-easy.md.txt): At Google IO 2018, we presented a roundup of tools, libraries and optimization techniques that make improving web performance easier. Here we explain them using The Oodles Theater app. We also talk about our experiments with predictive loading and the new Guess.js initiative.
- [How to query the Web Platform Dashboard for Baseline tooling](https://web.dev/articles/web-platform-dashboard-baseline.md.txt): Learn about the Web Platform Dashboard and how you can query its HTTP API to get data on features that have reached Baseline to build tools for your development workflow.
- [Integrate with the OS sharing UI with the Web Share API](https://web.dev/articles/web-share.md.txt): With the Web Share API, web apps are able to use the same system-provided share capabilities as platform-specific apps. The Web Share API makes it possible for web apps to share links, text, and files to other apps installed on the device in the same way as platform-specific apps.
- [Web storage overview](https://web.dev/articles/web-storage.md.txt): Guidance to build modern web experiences that work on any browser.
- [WebAPKs on Android](https://web.dev/articles/webapks.md.txt): When the user adds your Progressive Web App to their home screen on Android, Chrome automatically generates an APK for you, which we sometimes call a WebAPK. Being installed via an APK makes it possible for your app to show up in the app launcher, in Android&#39;s app settings and to register a set of intent filters.
- [WebAssembly feature detection](https://web.dev/articles/webassembly-feature-detection.md.txt): Learn how to use the newest WebAssembly features while supporting users across all browsers.
- [Debugging memory leaks in WebAssembly using Emscripten](https://web.dev/articles/webassembly-memory-debugging.md.txt): Learn how to use WebAssembly to bring libraries, written in other languages, to the Web in a safe and idiomatic manner.
- [WebAssembly performance patterns for web apps](https://web.dev/articles/webassembly-performance-patterns-for-web-apps.md.txt): In this guide, aimed at web developers who want to benefit from WebAssembly, you&#39;ll learn how to make use of Wasm to outsource CPU-intensive tasks with the help of a running example.
- [Using WebAssembly threads from C, C++ and Rust](https://web.dev/articles/webassembly-threads.md.txt): Learn how to bring multithreaded applications written in other languages to WebAssembly.
- [Case Study - A Tale of an HTML5 Game with Web Audio](https://web.dev/articles/webaudio-fieldrunners.md.txt): Guidance to build modern web experiences that work on any browser.
- [Developing game audio with the Web Audio API](https://web.dev/articles/webaudio-games.md.txt): Guidance to build modern web experiences that work on any browser.
- [Getting started with Web Audio API](https://web.dev/articles/webaudio-intro.md.txt): Guidance to build modern web experiences that work on any browser.
- [Mixing positional audio and WebGL](https://web.dev/articles/webaudio-positional-audio.md.txt): Guidance to build modern web experiences that work on any browser.
- [Determine the passkey provider with AAGUID](https://web.dev/articles/webauthn-aaguid.md.txt): Relying parties can determine where the passkey comes from by examining AAGUID. Find out how it works.
- [Simpler WebAuthn feature detection with getClientCapabilities()](https://web.dev/articles/webauthn-client-capabilities.md.txt): Short description
- [Credential Management API Feature Detection Check-up](https://web.dev/articles/webauthn-credential-management.md.txt): Credential Management API Feature Detection Check-up
- [Discoverable credentials deep dive](https://web.dev/articles/webauthn-discoverable-credentials.md.txt): Learn what are discoverable credentials and how to build user experiences that suit your use case.
- [Prevent creation of a new passkey if one already exists](https://web.dev/articles/webauthn-exclude-credentials.md.txt): Learn how to prevent creating a new passkey if one already exists in the user&#39;s password manager.
- [Allow passkey reuse across your sites with Related Origin Requests](https://web.dev/articles/webauthn-related-origin-requests.md.txt): Learn how to use Related Origin Requests to allow passkey reuse across your sites.

- [userVerification deep dive](https://web.dev/articles/webauthn-user-verification.md.txt): Learn how to use `userVerification` in WebAuthn
- [Web Components v1 - the next generation](https://web.dev/articles/webcomponents-org.md.txt): Web Components are gaining cross-browser support, the community is growing in leaps and bounds, and there’s a brand-new Web Component catalog to find exactly the component you need.
- [A simple TODO list using HTML5 WebDatabases](https://web.dev/articles/webdatabase-todo.md.txt): Guidance to build modern web experiences that work on any browser.
- [Quick guide to webfonts via @font-face](https://web.dev/articles/webfonts-quick.md.txt): Guidance to build modern web experiences that work on any browser.
- [WebGL Fundamentals](https://web.dev/articles/webgl-fundamentals.md.txt): Guidance to build modern web experiences that work on any browser.
- [Making of the World Wonders 3D Globe](https://web.dev/articles/webgl-globe.md.txt): Guidance to build modern web experiences that work on any browser.
- [Writing augmented reality applications using JSARToolKit](https://web.dev/articles/webgl-jsartoolkit-webrtc.md.txt): Guidance to build modern web experiences that work on any browser.
- [Animating a million letters using Three.js](https://web.dev/articles/webgl-million-letters.md.txt): Guidance to build modern web experiences that work on any browser.
- [WebGL orthographic 3D](https://web.dev/articles/webgl-orthographic-3d.md.txt): Guidance to build modern web experiences that work on any browser.
- [An introduction to shaders](https://web.dev/articles/webgl-shaders.md.txt): Guidance to build modern web experiences that work on any browser.
- [WebGL transforms](https://web.dev/articles/webgl-transforms.md.txt): Guidance to build modern web experiences that work on any browser.
- [Typed arrays - Binary data in the browser](https://web.dev/articles/webgl-typed-arrays.md.txt): Guidance to build modern web experiences that work on any browser.
- [Webpack](https://web.dev/articles/webpack.md.txt): Bundling for modern web applications
- [Webpack conclusion](https://web.dev/articles/webpack-conclusion.md.txt): Summing up Webpack
- [Get started with WebRTC](https://web.dev/articles/webrtc-basics.md.txt): Guidance to build modern web experiences that work on any browser.
- [Send data between browsers with WebRTC data channels](https://web.dev/articles/webrtc-datachannels.md.txt): Guidance to build modern web experiences that work on any browser.
- [Build the backend services needed for a WebRTC app](https://web.dev/articles/webrtc-infrastructure.md.txt): Guidance to build modern web experiences that work on any browser.
- [WebRTC is now a W3C and IETF standard](https://web.dev/articles/webrtc-standard-announcement.md.txt): A brief overview of the history, architecture, use cases, and future of WebRTC.
- [Building the main navigation for a website](https://web.dev/articles/website-navigation.md.txt): This tutorial describes how to build an accessible main navigation of a website. You learn about semantic HTML, accessibility, and how using ARIA attributes can sometimes do more harm than good.
- [Introducing WebSockets - Bringing Sockets to the Web](https://web.dev/articles/websockets-basics.md.txt): Guidance to build modern web experiences that work on any browser.
- [Welcome to the immersive web](https://web.dev/articles/welcome-to-immersive.md.txt): The immersive web means virtual world experiences hosted through the browser. This entire virtual reality experiences surfaced in the browser or in VR enabled headsets.
- [What are Progressive Web Apps?](https://web.dev/articles/what-are-pwas.md.txt): An introduction to Progressive Web Apps (PWAs) and the three pillars that separate them from other web apps.
- [What is accessibility?](https://web.dev/articles/what-is-accessibility.md.txt): An accessible site is one whose content can be accessed regardless of any user&#39;s impairments, and whose functionality can also be operated by the most diverse range of users possible.
- [What is mixed content?](https://web.dev/articles/what-is-mixed-content.md.txt): Mixed content occurs when initial HTML is loaded over a secure HTTPS connection, but other resources are loaded over an insecure HTTP connection.
- [What is speed?](https://web.dev/articles/what-is-speed.md.txt): Speed matters, but what exactly do we mean by it? What does it mean to have a fast site?
- [What&#39;s the CSS :scope pseudo-class for?](https://web.dev/articles/what-s-the-CSS-scope-pseudo-class-for.md.txt): Guidance to build modern web experiences that work on any browser.
- [What&#39;s new in PageSpeed Insights](https://web.dev/articles/whats-new-pagespeed-insights.md.txt): Learn about the latest in PageSpeed Insights to help you better measure and optimize your page and site quality.
- [When to use HTTPS for local development](https://web.dev/articles/when-to-use-local-

https.md.txt): Guidance to build modern web experiences that work on any browser.
- [Why you need &quot;cross-origin isolated&quot; for powerful features](https://web.dev/articles/why-coop-coep.md.txt): Some web APIs increase the risk of side-channel attacks like Spectre. To mitigate that risk, browsers offer an opt-in-based isolated environment called cross-origin isolated. Learn why cross-origin isolation is needed to use powerful features such as `SharedArrayBuffer`, `performance.measureUserAgentSpecificMemory()` and high resolution timer with better precision.
- [Why HTTPS matters](https://web.dev/articles/why-https-matters.md.txt): HTTPS protects the integrity of your website, protects the privacy and security of your users, and is a prerequisite for new and powerful web platform APIs.
- [Customize the window controls overlay of your PWA&#39;s title bar](https://web.dev/articles/window-controls-overlay.md.txt): With the Window Controls Overlay feature, developers can customize the title bar of installed PWAs so that their PWAs feel more like apps.
- [Build in-browser WordPress experiences with WordPress Playground and WebAssembly](https://web.dev/articles/wordpress-playground.md.txt): The full WordPress powered by PHP running solely in the browser with WebAssembly
- [Workbox: your high-level service worker toolkit](https://web.dev/articles/workbox.md.txt): Workbox is a high-level service worker toolkit built on top of the Service Worker and Cache Storage APIs. It provides a production-ready set of libraries for adding offline support to web apps.
- [Integrate PWAs into built-in sharing UIs with Workbox](https://web.dev/articles/workbox-share-targets.md.txt): How to register routes in Workbox so that your Progressive Web App shows up in system-level sharing UIs, alongside platform-specific apps.
- [Workers overview](https://web.dev/articles/workers-overview.md.txt): How web workers and service workers can improve the performance of your website, and when to use a web worker versus a service worker.
- [New tricks in XMLHttpRequest2](https://web.dev/articles/xhr2.md.txt): Guidance to build modern web experiences that work on any browser.
- [Building web apps with Yeoman and Polymer](https://web.dev/articles/yeoman.md.txt): Guidance to build modern web experiences that work on any browser.
- [Your first performance budget](https://web.dev/articles/your-first-performance-budget.md.txt): Ensure your site loads fast with a step-by-step guide to defining thresholds for performance metrics that are meaningful for your site.