http://localhost/

99 Performance

78 Accessibility

93 Best Practices

▲ 0-49 50-89 90-100

99

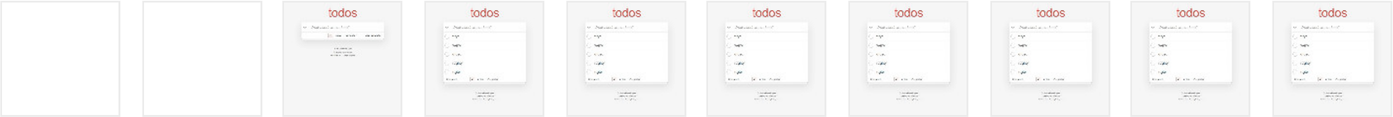
Performance

Metrics

First Contentful Paint	0.6 s	Time to Interactive	0.7 s
Speed Index	0.9 s	Total Blocking Time	50 ms
Largest Contentful Paint	0.6 s	Cumulative Layout Shift	0.086

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

View Original Trace



Diagnostics — More information about the performance of your application. These numbers don't directly affect the Performance score.

Serve static assets with an efficient cache policy — 10 resources found

A long cache lifetime can speed up repeat visits to your page. Learn more.

□ ~~Show 3rd-party resources~~ (0)

URL	Cache TTL	Transfer Size
/js/controller.js (localhost)	None	8 KiB
…todomvc-common/base.js (localhost)	None	7 KiB

URL	Cache TTL	Transfer Size
…todomvc-app-css/index.css (localhost)	None	7 KiB
/js/view.js (localhost)	None	6 KiB
/js/store.js (localhost)	None	4 KiB
/js/model.js (localhost)	None	3 KiB
/js/template.js (localhost)	None	3 KiB
…todomvc-common/base.css (localhost)	None	2 KiB
/js/helpers.js (localhost)	None	2 KiB
/js/app.js (localhost)	None	1 KiB

Avoid chaining critical requests — 10 chains found

The Critical Request Chains below show you what resources are loaded with a high priority. Consider reducing the length of chains, reducing the download size of resources, or deferring the download of unnecessary resources to improve page load. Learn more.

Maximum critical path latency: **620 ms**

Initial Navigation

- http://localhost
  - …todomvc-common/base.css (localhost) **- 10 ms, 2.08 KiB**
  - …todomvc-app-css/index.css (localhost) **- 50 ms, 7.39 KiB**
  - …todomvc-common/base.js (localhost) **- 50 ms, 7.40 KiB**
  - /js/helpers.js (localhost) **- 160 ms, 1.88 KiB**
  - /js/store.js (localhost) **- 160 ms, 4.44 KiB**
  - /js/model.js (localhost) **- 200 ms, 3.40 KiB**
  - /js/template.js (localhost) **- 210 ms, 3.19 KiB**
  - /js/view.js (localhost) **- 60 ms, 5.86 KiB**
  - /js/controller.js (localhost) **- 90 ms, 7.69 KiB**
  - /js/app.js (localhost) **- 90 ms, 0.91 KiB**

Keep request counts low and transfer sizes small — 12 requests • 47 KiB

To set budgets for the quantity and size of page resources, add a budget.json file. Learn more.

Resource Type	Requests	Transfer Size
Total	12	46.5 KiB
Script	8	34.8 KiB
Stylesheet	2	9.5 KiB
Document	1	1.8 KiB

Resource Type	Requests	Transfer Size
Other	1	0.5 KiB
Image	0	0.0 KiB
Media	0	0.0 KiB
Font	0	0.0 KiB
Third-party	0	0.0 KiB

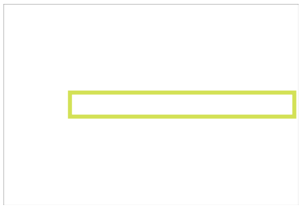
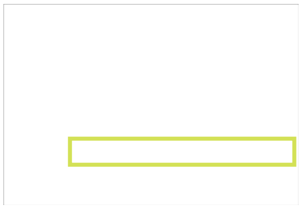
Largest Contentful Paint element — 1 element found

This is the largest contentful element painted within the viewport. Learn More

Element	
	h1

Avoid large layout shifts — 2 elements found

These DOM elements contribute most to the CLS of the page.

Element		CLS Contribution
	::before `<::before>`	0.046
	footer.info	0.04

Avoid long main-thread tasks — 2 long tasks found

Lists the longest tasks on the main thread, useful for identifying worst contributors to input delay. Learn more

□ ~~Show 3rd-party resources~~ (0)

URL	Start Time	Duration
http://localhost	283 ms	392 ms
http://localhost	177 ms	106 ms

Passed audits (31)

Eliminate render-blocking resources — Potential savings of 0 ms

Resources are blocking the first paint of your page. Consider delivering critical JS/CSS inline and deferring all non-critical JS/styles. Learn more.

□ ~~Show 3rd-party resources~~ (0)

URL	Transfer Size	Potential Savings
…todomvc-common/base.css (localhost)	2.1 KiB	60 ms
…todomvc-app-css/index.css (localhost)	7.4 KiB	100 ms

Properly size images

Serve images that are appropriately-sized to save cellular data and improve load time. Learn more.

Defer offscreen images

Consider lazy-loading offscreen and hidden images after all critical resources have finished loading to lower time to interactive. Learn more.

Minify CSS

Minifying CSS files can reduce network payload sizes. Learn more.

Minify JavaScript

Minifying JavaScript files can reduce payload sizes and script parse time. Learn more.

Remove unused CSS

Remove dead rules from stylesheets and defer the loading of CSS not used for above-the-fold content to reduce unnecessary bytes consumed by network activity. Learn more.

Remove unused JavaScript

Remove unused JavaScript to reduce bytes consumed by network activity. Learn more.

Efficiently encode images

Optimized images load faster and consume less cellular data. Learn more.

Serve images in next-gen formats

Image formats like JPEG 2000, JPEG XR, and WebP often provide better compression than PNG or JPEG, which means faster downloads and less data consumption. Learn more.

Enable text compression

Text-based resources should be served with compression (gzip, deflate or brotli) to minimize total network bytes. Learn more.

Preconnect to required origins

Consider adding `preconnect` or `dns-prefetch` resource hints to establish early connections to important third-party origins. Learn more.

Initial server response time was short — Root document took 20 ms

Keep the server response time for the main document short because all other requests depend on it. Learn more.

□ ~~Show 3rd-party resources~~ (0)

URL	Time Spent
http://localhost	20 ms

Avoid multiple page redirects

Redirects introduce additional delays before the page can be loaded. Learn more.

Preload key requests

Consider using `<link rel=preload>` to prioritize fetching resources that are currently requested later in page load. Learn more.

Use HTTP/2

HTTP/2 offers many benefits over HTTP/1.1, including binary headers and multiplexing. Learn more.

Use video formats for animated content

Large GIFs are inefficient for delivering animated content. Consider using MPEG4/WebM videos for animations and PNG/WebP for static images instead of GIF to save network bytes. Learn more

Remove duplicate modules in JavaScript bundles

Remove large, duplicate JavaScript modules from bundles to reduce unnecessary bytes consumed by network activity.

Avoid serving legacy JavaScript to modern browsers

Polyfills and transforms enable legacy browsers to use new JavaScript features. However, many aren't necessary for modern browsers. For your bundled JavaScript, adopt a modern script deployment strategy using module/nomodule feature detection to reduce the amount of code shipped to modern browsers, while retaining support for legacy browsers. Learn More

Preload Largest Contentful Paint image

Preload the image used by the LCP element in order to improve your LCP time. Learn more.

Avoids enormous network payloads — Total size was 47 KiB

Large network payloads cost users real money and are highly correlated with long load times. Learn more.

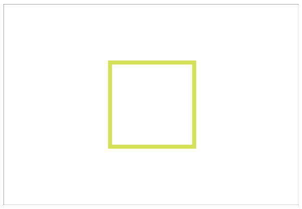
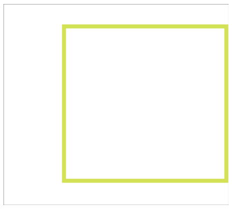
□ ~~Show 3rd-party resources~~ (0)

URL	Transfer Size
/js/controller.js (localhost)	7.7 KiB
…todomvc-common/base.js (localhost)	7.4 KiB

URL	Transfer Size
…todomvc-app-css/index.css (localhost)	7.4 KiB
/js/view.js (localhost)	5.9 KiB
/js/store.js (localhost)	4.4 KiB
/js/model.js (localhost)	3.4 KiB
/js/template.js (localhost)	3.2 KiB
…todomvc-common/base.css (localhost)	2.1 KiB
/js/helpers.js (localhost)	1.9 KiB
http://localhost	1.8 KiB

Avoids an excessive DOM size — 56 elements

A large DOM will increase memory usage, cause longer style calculations, and produce costly layout reflows. Learn more.

Statistic	Element	Value
Total DOM Elements		56
Maximum DOM Depth	 input.toggle	7
Maximum Child Elements	 body	10

User Timing marks and measures

Consider instrumenting your app with the User Timing API to measure your app's real-world performance during key user experiences. Learn more.

JavaScript execution time — 0.2 s

Consider reducing the time spent parsing, compiling, and executing JS. You may find delivering smaller JS payloads helps with this. Learn more.

□ ~~Show 3rd-party resources~~ (0)

URL	Total CPU Time	Script Evaluation	Script Parse
http://localhost	639 ms	25 ms	134 ms
`Unattributable`	279 ms	40 ms	0 ms

Minimizes main-thread work — 1.0 s

Consider reducing the time spent parsing, compiling and executing JS. You may find delivering smaller JS payloads helps with this. Learn more

Category	Time Spent
Other	360 ms
Parse HTML & CSS	258 ms
Script Parsing & Compilation	167 ms
Rendering	101 ms
Script Evaluation	86 ms
Style & Layout	14 ms

All text remains visible during webfont loads

Leverage the font-display CSS feature to ensure text is user-visible while webfonts are loading. Learn more.

Minimize third-party usage

Third-party code can significantly impact load performance. Limit the number of redundant third-party providers and try to load third-party code after your page has primarily finished loading. Learn more.

Lazy load third-party resources with facades

Some third-party embeds can be lazy loaded. Consider replacing them with a facade until they are required. Learn more.

Uses passive listeners to improve scrolling performance

Consider marking your touch and wheel event listeners as `passive` to improve your page's scroll performance. Learn more.

Avoids `document.write()`

For users on slow connections, external scripts dynamically injected via `document.write()` can delay page load by tens of seconds. Learn more.

Avoid non-composited animations

Animations which are not composited can be janky and increase CLS. Learn more

Image elements have explicit `width` and `height`

Set an explicit width and height on image elements to reduce layout shifts and improve CLS. Learn more

78

Accessibility

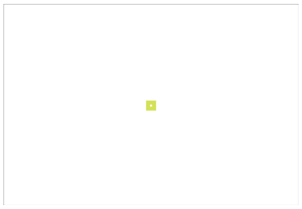
These checks highlight opportunities to improve the accessibility of your web app. Only a subset of accessibility issues can be automatically detected so manual testing is also encouraged.

Names and labels — These are opportunities to improve the semantics of the controls in your application. This may enhance the experience for users of assistive technology, like a screen reader.

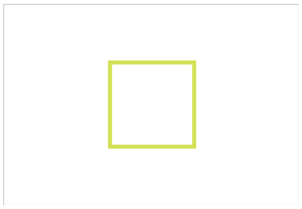
▲ Form elements do not have associated labels

Labels ensure that form controls are announced properly by assistive technologies, like screen readers. Learn more.

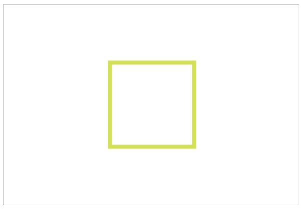
Failing Elements



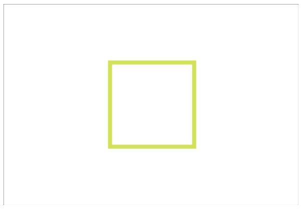
input.toggle-all



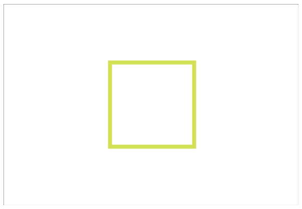
input.toggle



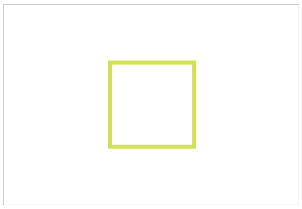
input.toggle



input.toggle



input.toggle



input.toggle

Additional items to manually check (10) — These items address areas which an automated testing tool cannot cover. Learn more in our guide on conducting an accessibility review.

The page has a logical tab order

Tabbing through the page follows the visual layout. Users cannot focus elements that are offscreen. Learn more.

Interactive controls are keyboard focusable

Custom interactive controls are keyboard focusable and display a focus indicator. Learn more.

Interactive elements indicate their purpose and state

Interactive elements, such as links and buttons, should indicate their state and be distinguishable from non-interactive elements. Learn more.

The user's focus is directed to new content added to the page

If new content, such as a dialog, is added to the page, the user's focus is directed to it. Learn more.

User focus is not accidentally trapped in a region

A user can tab into and out of any control or region without accidentally trapping their focus. Learn more.

Custom controls have associated labels

Custom interactive controls have associated labels, provided by aria-label or aria-labelledby. Learn more.

Custom controls have ARIA roles

Custom interactive controls have appropriate ARIA roles. Learn more.

Visual order on the page follows DOM order

DOM order matches the visual order, improving navigation for assistive technology. Learn more.

Offscreen content is hidden from assistive technology

Offscreen content is hidden with display: none or aria-hidden=true. Learn more.

HTML5 landmark elements are used to improve navigation

Landmark elements (<main>, <nav>, etc.) are used to improve the keyboard navigation of the page for assistive technology. Learn more.

Passed audits (10)

`[aria-hidden="true"]` is not present on the document `<body>`

Assistive technologies, like screen readers, work inconsistently when `aria-hidden="true"` is set on the document `<body>`. Learn more.

The page contains a heading, skip link, or landmark region

Adding ways to bypass repetitive content lets keyboard users navigate the page more efficiently. Learn more.

Background and foreground colors have a sufficient contrast ratio

Low-contrast text is difficult or impossible for many users to read. Learn more.

Document has a `<title>` element

The title gives screen reader users an overview of the page, and search engine users rely on it heavily to determine if a page is relevant to their search. Learn more.

Heading elements appear in a sequentially-descending order

Properly ordered headings that do not skip levels convey the semantic structure of the page, making it easier to navigate and understand when using assistive technologies. Learn more.

`<html>` element has a `[lang]` attribute

If a page doesn't specify a lang attribute, a screen reader assumes that the page is in the default language that the user chose when setting up the screen reader. If the page isn't actually in the default language, then the screen reader might not announce the page's text correctly. Learn more.

`<html>` element has a valid value for its `[lang]` attribute

Specifying a valid BCP 47 language helps screen readers announce text properly. Learn more.

Links have a discernible name

Link text (and alternate text for images, when used as links) that is discernible, unique, and focusable improves the navigation experience for screen reader users. Learn more.

Lists contain only `<li>` elements and script supporting elements (`<script>` and `<template>`).

Screen readers have a specific way of announcing lists. Ensuring proper list structure aids screen reader output. Learn more.

List items (`<li>`) are contained within `<ul>` or `<ol>` parent elements

Screen readers require list items (`<li>`) to be contained within a parent `<ul>` or `<ol>` to be announced properly. Learn more.

Not applicable (33)

`[accesskey]` values are unique

Access keys let users quickly focus a part of the page. For proper navigation, each access key must be unique. Learn more.

`[aria-*]` attributes match their roles

Each ARIA `role` supports a specific subset of `aria-*` attributes. Mismatching these invalidates the `aria-*` attributes. Learn more.

`button`, `link`, and `menuitem` elements have accessible names

When an element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more.

`[aria-hidden="true"]` elements do not contain focusable descendents

Focusable descendents within an `[aria-hidden="true"]` element prevent those interactive elements from being available to users of assistive technologies like screen readers. Learn more.

ARIA input fields have accessible names

When an input field doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more.

ARIA `meter` elements have accessible names

When an element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more.

ARIA `progressbar` elements have accessible names

When an element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more.

`[role]`s have all required `[aria-*]` attributes

Some ARIA roles have required attributes that describe the state of the element to screen readers. Learn more.

Elements with an ARIA `[role]` that require children to contain a specific `[role]` have all required children.

Some ARIA parent roles must contain specific child roles to perform their intended accessibility functions. Learn more.

`[role]`s are contained by their required parent element

Some ARIA child roles must be contained by specific parent roles to properly perform their intended accessibility functions. Learn more.

`[role]` values are valid

ARIA roles must have valid values in order to perform their intended accessibility functions. Learn more.

ARIA toggle fields have accessible names

When a toggle field doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more.

ARIA `tooltip` elements have accessible names

When an element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more.

ARIA `treeitem` elements have accessible names

When an element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more.

`[aria-*]` attributes have valid values

Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid values. Learn more.

`[aria-*]` attributes are valid and not misspelled

Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid names. Learn more.

Buttons have an accessible name

When a button doesn't have an accessible name, screen readers announce it as "button", making it unusable for users who rely on screen readers. Learn more.

`<dl>`'s contain only properly-ordered `<dt>` and `<dd>` groups, `<script>`, `<template>` or `<div>` elements.

When definition lists are not properly marked up, screen readers may produce confusing or inaccurate output. Learn more.

Definition list items are wrapped in `<dl>` elements

Definition list items (`<dt>` and `<dd>`) must be wrapped in a parent `<dl>` element to ensure that screen readers can properly announce them. Learn more.

`[id]` attributes on active, focusable elements are unique

All focusable elements must have a unique `id` to ensure that they're visible to assistive technologies. Learn more.

ARIA IDs are unique

The value of an ARIA ID must be unique to prevent other instances from being overlooked by assistive technologies. Learn more.

No form fields have multiple labels

Form fields with multiple labels can be confusingly announced by assistive technologies like screen readers which use either the first, the last, or all of the labels. Learn more.

`<frame>` or `<iframe>` elements have a title

Screen reader users rely on frame titles to describe the contents of frames. Learn more.

Image elements have `[alt]` attributes

Informative elements should aim for short, descriptive alternate text. Decorative elements can be ignored with an empty alt attribute. Learn more.

`<input type="image">` elements have `[alt]` text

When an image is being used as an `<input>` button, providing alternative text can help screen reader users understand the purpose of the button. Learn more.

The document does not use `<meta http-equiv="refresh">`

Users do not expect a page to refresh automatically, and doing so will move focus back to the top of the page. This may create a frustrating or confusing experience. Learn more.

`[user-scalable="no"]` is not used in the `<meta name="viewport">` element and the `[maximum-scale]` attribute is not less than 5.

Disabling zooming is problematic for users with low vision who rely on screen magnification to properly see the contents of a web page. Learn more.

`<object>` elements have `[alt]` text

Screen readers cannot translate non-text content. Adding alt text to `<object>` elements helps screen readers convey meaning to users. Learn more.

No element has a `[tabindex]` value greater than 0

A value greater than 0 implies an explicit navigation ordering. Although technically valid, this often creates frustrating experiences for users who rely on assistive technologies. Learn more.

Cells in a `<table>` element that use the `[headers]` attribute refer to table cells within the same table.

Screen readers have features to make navigating tables easier. Ensuring `<td>` cells using the `[headers]` attribute only refer to other cells in the same table may improve the experience for screen reader users. Learn more.

`<th>` elements and elements with `[role="columnheader"/"rowheader"]` have data cells they describe.

Screen readers have features to make navigating tables easier. Ensuring table headers always refer to some set of cells may improve the experience for screen reader users. Learn more.

`[lang]` attributes have a valid value

Specifying a valid BCP 47 language on elements helps ensure that text is pronounced correctly by a screen reader. Learn more.

`<video>` elements contain a `<track>` element with `[kind="captions"]`

When a video provides a caption it is easier for deaf and hearing impaired users to access its information. Learn more.

Best Practices

General

▲ Browser errors were logged to the console

Errors logged to the console indicate unresolved problems. They can come from network request failures and other browser concerns. Learn more

□ ~~Show 3rd-party resources~~ (0)

Source	Description
/learn.json:1	`Failed to load resource: the server responded with a status of 404 (Not Found)`

Passed audits (16)

Uses HTTPS

All sites should be protected with HTTPS, even ones that don't handle sensitive data. This includes avoiding mixed content, where some resources are loaded over HTTP despite the initial request being served over HTTPS. HTTPS prevents intruders from tampering with or passively listening in on the communications between your app and your users, and is a prerequisite for HTTP/2 and many new web platform APIs. Learn more.

Links to cross-origin destinations are safe

Add `rel="noopener"` or `rel="noreferrer"` to any external links to improve performance and prevent security vulnerabilities. Learn more.

Avoids requesting the geolocation permission on page load

Users are mistrustful of or confused by sites that request their location without context. Consider tying the request to a user action instead. Learn more.

Avoids requesting the notification permission on page load

Users are mistrustful of or confused by sites that request to send notifications without context. Consider tying the request to user gestures instead. Learn more.

Avoids front-end JavaScript libraries with known security vulnerabilities

Some third-party scripts may contain known security vulnerabilities that are easily identified and exploited by attackers. Learn more.

Allows users to paste into password fields

Preventing password pasting undermines good security policy. Learn more.

Displays images with correct aspect ratio

Image display dimensions should match natural aspect ratio. Learn more.

Serves images with appropriate resolution

Image natural dimensions should be proportional to the display size and the pixel ratio to maximize image clarity. Learn more.

Page has the HTML doctype

Specifying a doctype prevents the browser from switching to quirks-mode. Learn more.

Properly defines charset

A character encoding declaration is required. It can be done with a `<meta>` tag in the first 1024 bytes of the HTML or in the Content-Type HTTP response header. Learn more.

Avoids `unload` event listeners

The `unload` event does not fire reliably and listening for it can prevent browser optimizations like the Back-Forward Cache. Consider using the `pagehide` or `visibilitychange` events instead. Learn more

Avoids Application Cache

Application Cache is deprecated. Learn more.

Detected JavaScript libraries

All front-end JavaScript libraries detected on the page. Learn more.

Avoids deprecated APIs

Deprecated APIs will eventually be removed from the browser. Learn more.

Page has valid source maps

Source maps translate minified code to the original source code. This helps developers debug in production. In addition, Lighthouse is able to provide further insights. Consider deploying source maps to take advantage of these benefits. Learn more.

No issues in the `Issues` panel in Chrome Devtools

Issues logged to the `Issues` panel in Chrome Devtools indicate unresolved problems. They can come from network request failures, insufficient security controls, and other browser concerns. Open up the Issues panel in Chrome DevTools for more details on each issue.

Not applicable (1)

Fonts with `font-display: optional` are preloaded

Preload `optional` fonts so first-time visitors may use them. Learn more

Runtime Settings

URL	http://localhost/
Fetch Time	Jul 2, 2021, 4:36 PM CDT
Device	Emulated Desktop
Network throttling	40 ms TCP RTT, 10,240 Kbps throughput (Simulated)
CPU throttling	1x slowdown (Simulated)
Channel	devtools
User agent (host)	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36
User agent (network)	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4420.0 Safari/537.36 Chrome-Lighthouse
CPU/Memory Power	331
Axe version	4.1.2

Generated by **Lighthouse** 7.2.0 | File an issue