Glossary

ON THIS PAGE

The core of SvelteKit provides a highly configurable rendering engine. This section describes some of the terms used when discussing rendering. A reference for setting these options is provided in the documentation above.

CSR

Client-side rendering (CSR) is the generation of the page contents in the web browser using JavaScript.

In SvelteKit, client-side rendering will be used by default, but you can turn off JavaScript with the `csr = false` page option.

Hydration

Svelte components store some state and update the DOM when the state is updated. When fetching data during SSR, by default SvelteKit will store this data and transmit it to the client along with the server-rendered HTML. The components can then be initialized on the client with that data without having to call the same API endpoints again. Svelte will then check that the DOM is in the expected state and attach event listeners in a process called hydration. Once the components are fully hydrated, they can react to changes to their properties just like any newly created Svelte component.

In SvelteKit, pages will be hydrated by default, but you can turn off JavaScript with the `csr`

Prerendering means computing the contents of a page at build time and saving the HTML for display. This approach has the same benefits as traditional server-rendered pages, but avoids recomputing the page for each visitor and so scales nearly for free as the number of visitors increases. The tradeoff is that the build process is more expensive and prerendered content can only be updated by building and deploying a new version of the application.

Not all pages can be prerendered. The basic rule is this: for content to be prerenderable, any two users hitting it directly must get the same content from the server, and the page must not contain actions. Note that you can still prerender content that is loaded based on the page's parameters as long as all users will be seeing the same prerendered content.

Pre-rendered pages are not limited to static content. You can build personalized pages if user-specific data is fetched and rendered client-side. This is subject to the caveat that you will experience the downsides of not doing SSR for that content as discussed above.

In SvelteKit, you can control prerendering with the `prerender` page option and `prerender` config in `svelte.config.js`.

Routing

By default, when you navigate to a new page (by clicking on a link or using the browser's forward or back buttons), SvelteKit will intercept the attempted navigation and handle it instead of allowing the browser to send a request to the server for the destination page. SvelteKit will then update the displayed contents on the client by rendering the component for the new page, which in turn can make calls to the necessary API endpoints. This process of updating the page on the client in response to attempted navigation is called client-side routing.

In SvelteKit, client-side routing will be used by default, but you can skip it with `data-sveltekit-reload`.

HTML file which then does client-side rendering of the requested contents based on the requested URL. All navigation is handled on the client-side in a process called client-side routing with per-page contents being updated and common layout elements remaining largely unchanged. SPAs do not provide SSR, which has the shortcoming described above. However, some applications are not greatly impacted by these shortcomings such as a complex business application behind a login where SEO would not be important and it is known that users will be accessing the application from a consistent computing environment.

In SvelteKit, you can build an SPA with `adapter-static`.

SSG

Static Site Generation (SSG) is a term that refers to a site where every page is prerendered. SvelteKit was not built to do only static site generation like some tools and so may not scale as well to efficiently render a very large number of pages as tools built specifically for that purpose. However, in contrast to most purpose-built SSGs, SvelteKit does nicely allow for mixing and matching different rendering types on different pages. One benefit of fully prerendering a site is that you do not need to maintain or pay for servers to perform SSR. Once generated, the site can be served from CDNs, leading to great “time to first byte” performance. This delivery model is often referred to as JAMstack.

In SvelteKit, you can do static site generation by using `adapter-static` or by configuring every page to be prerendered using the `prerender` page option or `prerender` config in `svelte.config.js`.

SSR

Server-side rendering (SSR) is the generation of the page contents on the server. SSR is generally preferred for SEO. While some search engines can index content that is dynamically generated on the client-side it may take longer even in these cases. It also tends or is disabled (which happens more often than you probably think).

In SvelteKit, pages are server-side rendered by default. You can disable SSR with the `ssr` page option.