Fast webpages with Astro

By Sam Tracey

Contents

- What is Astro?
- When to use Astro?
- How to use Astro
- Try it out (30 mins)
- SSG vs SSR
- Island Architecture
- Discussion

What is Astro?

Astro is an **all-in-one web framework** for building **fast, content-focused** websites.

- Another Static Site Generator (SSG)
- Component based syntax that will be familiar to users of React and related frameworks.
- Server-first. Zero JS by default. Expensive hydration moved off of your users' devices (Better performance and SEO).
- Typescript first
- Customisable: Tailwind, MDX and 100+ more.
- Framework agnostic (React, Preact, Svelte, Vue, Solid, Lit and more).
- MPA not SPA.
- Edge ready (easy deployment).

When to use Astro?

- Your site is content rich (blog, online store)
- SEO matters
- You don’t have the need for much client-side JS
- You want to support several frameworks
- MPA makes more sense than SPA
- You are just making a view for an existing REST or GraphQL API

How to use Astro

npm create astro@latest

(choose a directory to create it and select blog when asked. Otherwise choose the default answers)

Prerequisites

- **Node.js** - `v18.14.1` or higher.
- **Text editor** - We recommend VS Code with the Official Astro extension.
- **Terminal** - Astro is accessed through its command-line interface (CLI).

If you don't have NodeJs and npm installed you can try things out in the browser here: https://astro.new/ (Choose the Blog one)

Cool features

1. Markdown, mdx, JSON supported out of the box
2. Collections
3. Astro components
4. Fetch on the server
5. Zig schema definitions
6. Compatible with most frameworks (backed by Vite)
7. View transitions

Try it out

1. Install and run the site with NPM
2. Change the blog text
3. Add a new blog entry
4. Experiment with layout

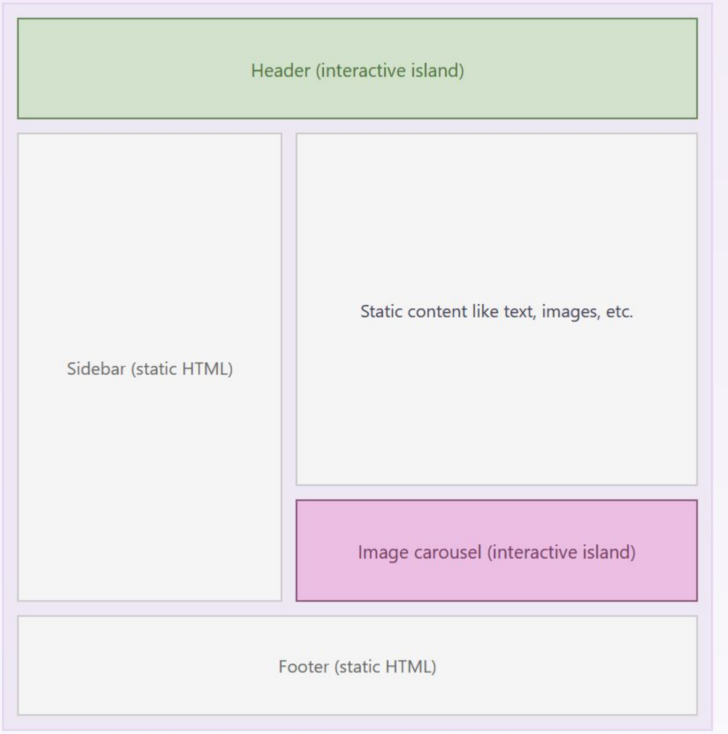
Try one or more of the following:

1. Use JSON collection instead of markdown
2. Create a data page from an existing REST endpoint (https://github.com/public-apis/public-apis)
3. Add an island of activity using your favourite framework

Server Side Rendering (SSR)

1. Implement sessions for login state in your app.
2. Render data from an API called dynamically with `fetch`.
3. Deploy your site to a host using an *adapter*.
4. URL/Query parameters can be handled by routing

Island Architecture


Header (interactive island)
Static content like text, images, etc.
Sidebar (static HTML)
Image carousel (interactive island)
Footer (static HTML)

Source: Islands Architecture: Jason Miller

Links

1. https://astro.build
2. https://docs.astro.build
3. https://astro.new/latest/
4. https://github.com/withastro/astro/tree/main/examples
5. https://github.com/one-aalam/awesome-astro
6. Discord - Astro Lounge
7. Me - @drbloke on Github