Astro Islands

Astro pioneered and popularized a frontend architecture called **Islands.** Islands architecture results in better frontend performance by helping you avoid monolithic JavaScript patterns and stripping all non-essential JavaScript from the page automatically. Developers keep using their favorite UI components and frameworks with Astro and still get these benefits.

A brief history

The term "component island" was first coined by Etsy's frontend architect Katie Sylor-Miller in 2019. This idea was then expanded on and documented in this post by Preact creator Jason Miller on August 11, 2020.

> The general idea of an "Islands" architecture is deceptively simple: render HTML pages on the server, and inject placeholders or slots around highly dynamic regions […] that can then be "hydrated" on the client into small self-contained widgets, reusing their server-rendered initial HTML.
> — Jason Miller, Creator of Preact

The technique that this architectural pattern builds on is also known as **partial** or **selective hydration.**

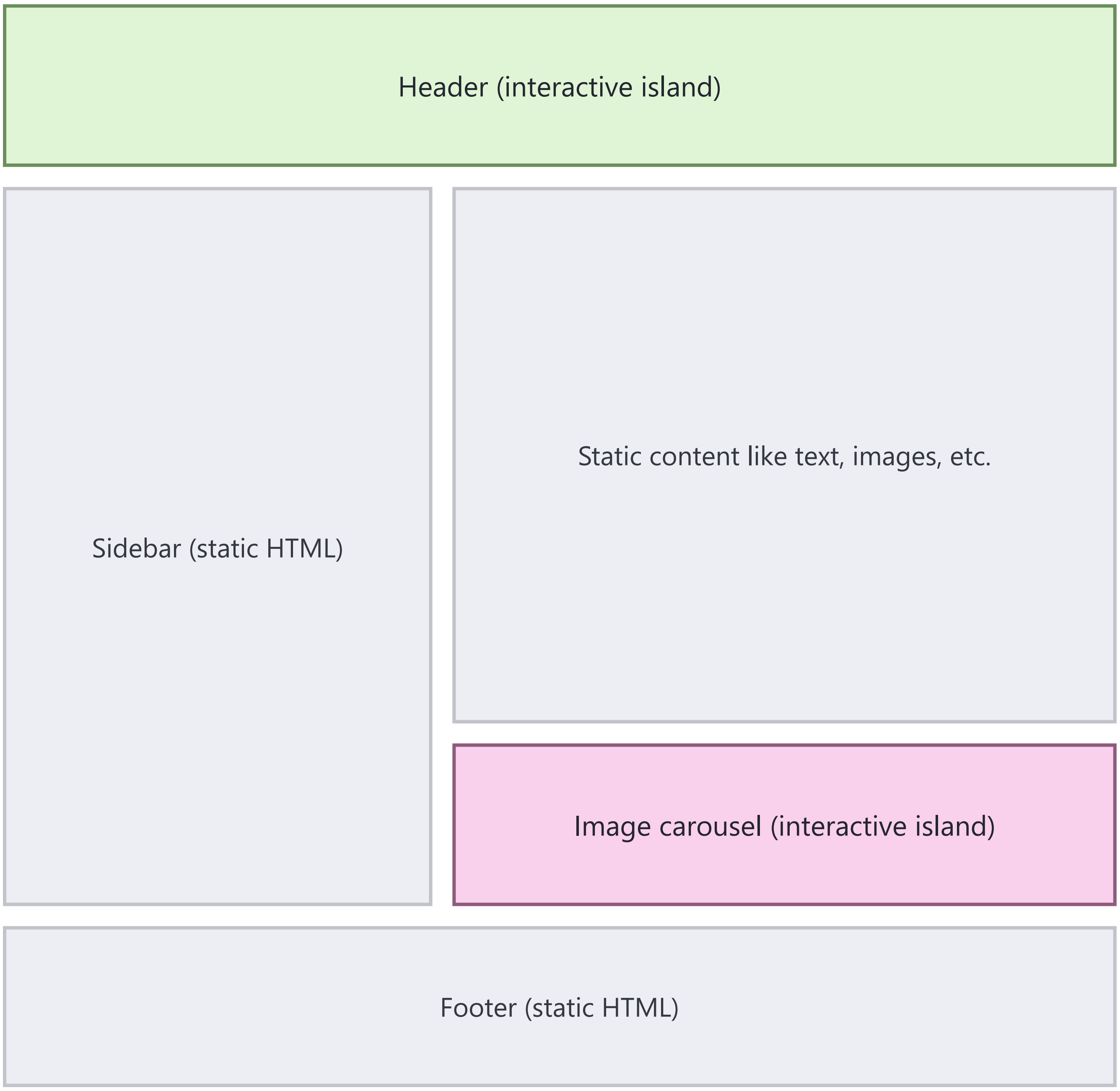
In contrast, most JavaScript-based web frameworks hydrate & render an entire website as one large JavaScript application (also known as a single-page application, or SPA). SPAs provide simplicity and power but suffer from page-load performance problems due to heavy client-side JavaScript usage.

SPAs have their place, even embedded inside an Astro page. But, SPAs lack the native ability to selectively and strategically hydrate, making them a heavy-handed choice for most projects on the web today.

Astro became popular as the first mainstream JavaScript web framework with selective hydration built-in, using that component islands pattern first coined by Sylor-Miller.

What is an island?

In Astro, an "island" refers to any interactive UI component on the page. Think of an island as an interactive widget floating in a sea of otherwise static, lightweight, server-rendered HTML.


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

Source: [Islands Architecture: Jason Miller]

An island always runs in isolation from other islands on the page, and multiple islands can exist on a page. Islands can still share state and communicate with each other, even though they run in different component contexts.

This flexibility allows Astro to support multiple UI frameworks like React, Preact, Svelte, Vue, and SolidJS. Because they are independent, you can even mix several frameworks on each page.

> **Tip**
>
> Although most developers will stick to just one UI framework, Astro supports multiple frameworks in the same project. This allows you to:
>
> - Choose the framework that is best for each component.
> - Learn a new framework without needing to start a new project.
> - Collaborate with others even when working in different frameworks.
> - Incrementally convert an existing site to another framework with no downtime.

Creating an island

By default, Astro will automatically render every UI component to just HTML & CSS, **stripping out all client-side JavaScript automatically.**

src/pages/index.astro

```
<MyReactComponent />
```

This may sound strict, but this behavior is what keeps Astro websites fast by default and protects developers from accidentally sending unnecessary or unwanted JavaScript that might slow down their website.

Turning any static UI component into an interactive island requires only a `client:*` directive. Astro then automatically builds and bundles your client-side JavaScript for optimized performance.

src/pages/index.astro

```
<!-- This component is now interactive on the page!
     The rest of your website remains static. -->
<MyReactComponent client:load />
```

With islands, client-side JavaScript is only loaded for the explicit interactive components that you mark using `client:*` directives.

And because interaction is configured at the component-level, you can handle different loading priorities for each component based on its usage. For example, `client:idle` tells a component to load when the browser becomes idle, and `client:visible` tells a component to load only once it enters the viewport.

What are the benefits of Islands?

The most obvious benefit of building with Astro Islands is performance: the majority of your website is converted to fast, static HTML and JavaScript is only loaded for the individual components that need it. JavaScript is one of the slowest assets that you can load per-byte, so every byte counts.

Another benefit is parallel loading. In the example illustration above, the low-priority “image carousel” island doesn’t need to block the high-priority “header” island. The two load in parallel and hydrate in isolation, meaning that the header becomes interactive immediately without having to wait for the heavier carousel lower down the page.

Even better, you can tell Astro exactly how and when to render each component. If that image carousel is really expensive to load, you can attach a special client directive that tells Astro to only load the carousel when it becomes visible on the page. If the user never sees it, it never loads.

In Astro, it’s up to you as the developer to explicitly tell Astro which components on the page need to also run in the browser. Astro will only hydrate exactly what’s needed on the page and leave the rest of your site as static HTML.

Islands are the secret to Astro’s fast-by-default performance story!

Edit page

Previous
Why Astro?

Next
Project Structure