Lanza tu cohete al espacio web con Astro

@davidrojom

Workshop Agenda

1

What is Astro?

Learn the core principles of Astro

2

Configuration

Integrations, environment variables, adapters

3

Routing and Pages

Static and dynamic pages, SSR and SSG

4

Components

Astro components, client islands and server islands

5

View Transitions

Provide seamless animations to your website

6

Content Layer

Organize and query your content

7

Actions

Call your backend functions with type-safety

8

Lets Build!

Learn the core principles of Astro

1 What is Astro?

Astro is a JavaScript web framework optimized for building fast, content-driven websites

Server First

Improved performance by rendering components on the server

Content-driven

Designed to work with content, load data from file system, external APIs or a CMS

Customizable

Extend Astro with your favorite tools

2

Configuration

Integrations






Output

SSG

SSR

Adapters



Environment variables

Prefetching

Prerendering

3

Routing and Pages

While SSG (or prerendered) pages are rendered within the build process, SSR pages are rendered in the server for each request

Static Routes

books.astro
books.md
books.mdx
books.html

Dynamic Routes

books/[slug].astro

books/[...slug].astro

getStaticPaths for prerendered pages

4

Components

Astro Components

Render in the server to static HTML by default (no JS sent to the client)

Support params, props and slots

Can include Client and Server Islands

Client Islands

Framework components hydrated on the client side

Parallel loading between islands

Control over when to hydrate a component

Server Islands

Rendered on demand independently of page rendering

Placeholders while loading

Allows caching the main content

5

View Transitions

Animated page transitions without full-page loads

<ClientRouter />

transition:name

Name a pair of elements to be animated

transition:animate

Customize the behaviour of transitions

- fade (default)
- initial
- slide
- none
- custom

transition:persist

Persist the state of a component between pages

6

Content Layer

Structure and Validate Your Content

Organized Content

Group related content (e.g., blog posts, docs) in named collections

Schema Validation

Use Zod schemas to enforce consistent properties and catch content errors early

Type Safety and Intellisense

Get full TypeScript support: safer queries, autocomplete, and better DX in your editor

Wide Content Source Support

Works with any type of data source. MD, MDX, APIs...

7

Actions

Call your Backend Functions with Ease

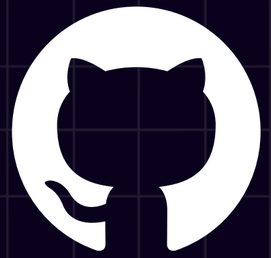
Type-Safe functions with Intellisense to call the backend from the client side.

Automatically validate endpoint input and output via zod schemas.

Standarized backend errors.

8

Lets Build!



github.com/DavidRojoM/astro-workshop