Performance stellari per siti web orientati ai contenuti.

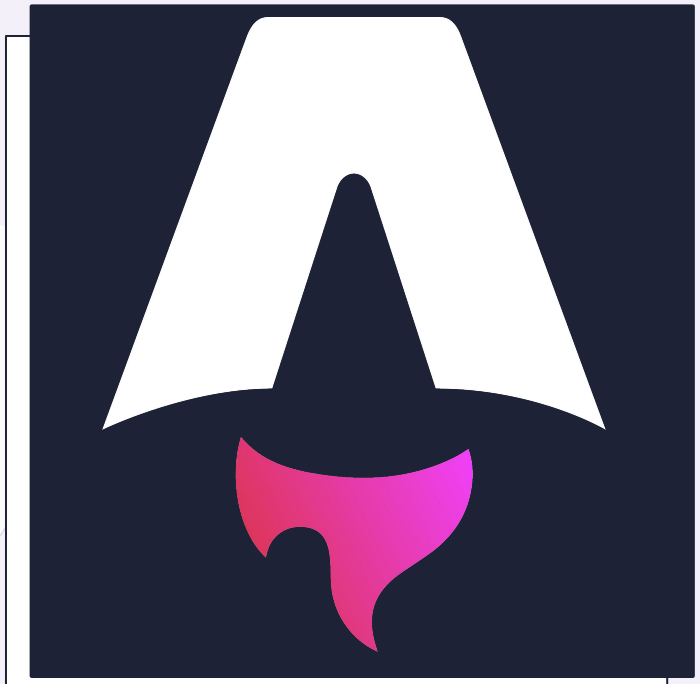
Davide Milan - Full Stack developer @Wavelop

Hello! I'm...

Davide Milan!

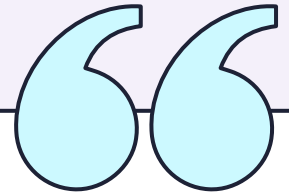
Full stack developer @Wavelop

- Laurea triennale in informatica 🖥️
- Capo Scout 🏕️
- Appassionato di papere 🦆🦆🦆
- 日本語の学生 🇯🇵



Introduzione ad Astro

Scopriamo insieme a chi è rivolto, casi d’uso principali, utilizzo base e alcune feature fondamentali



The web framework for content-driven websites

— astro.build

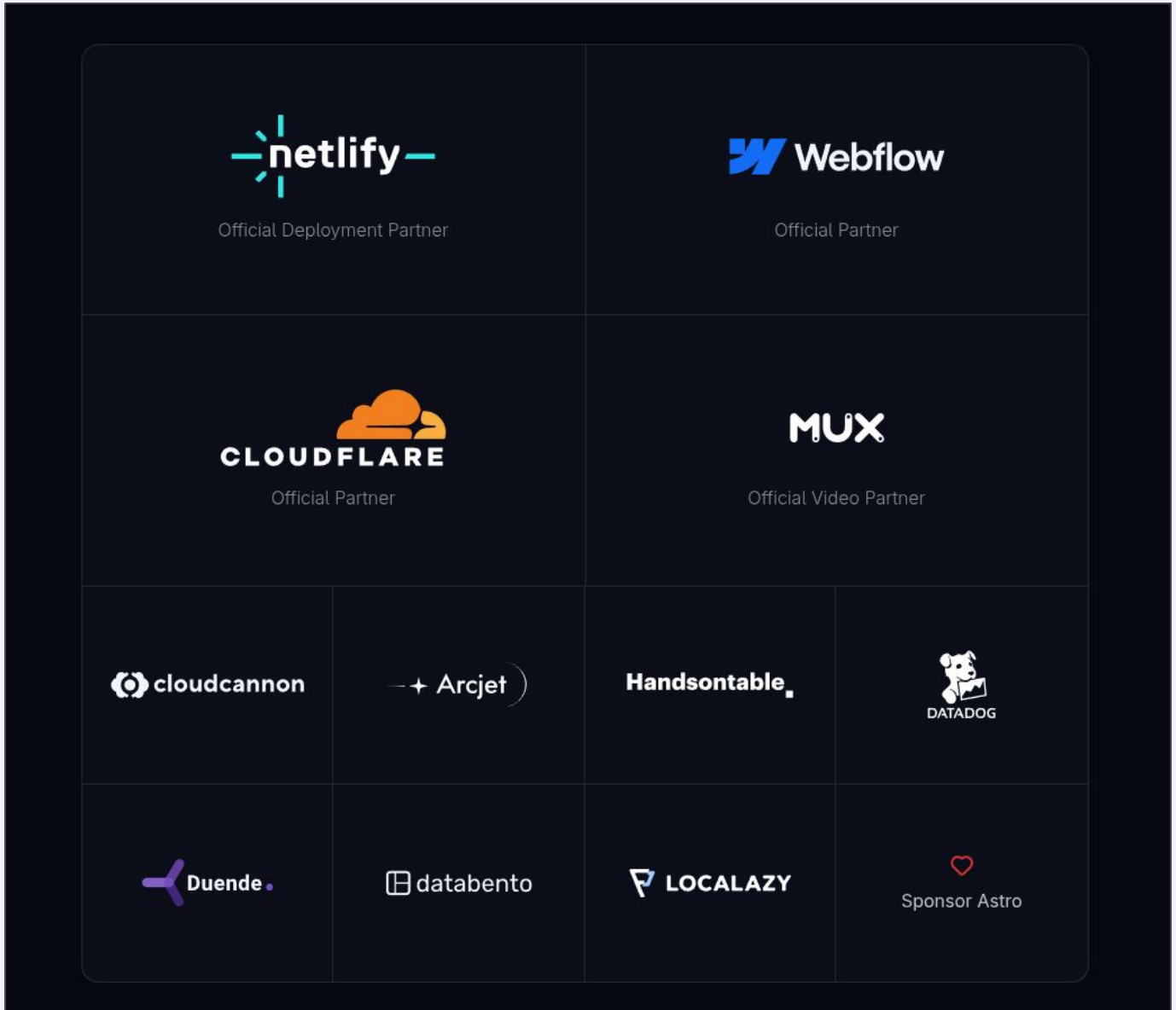
ORIGINI

Cos'è Astro?

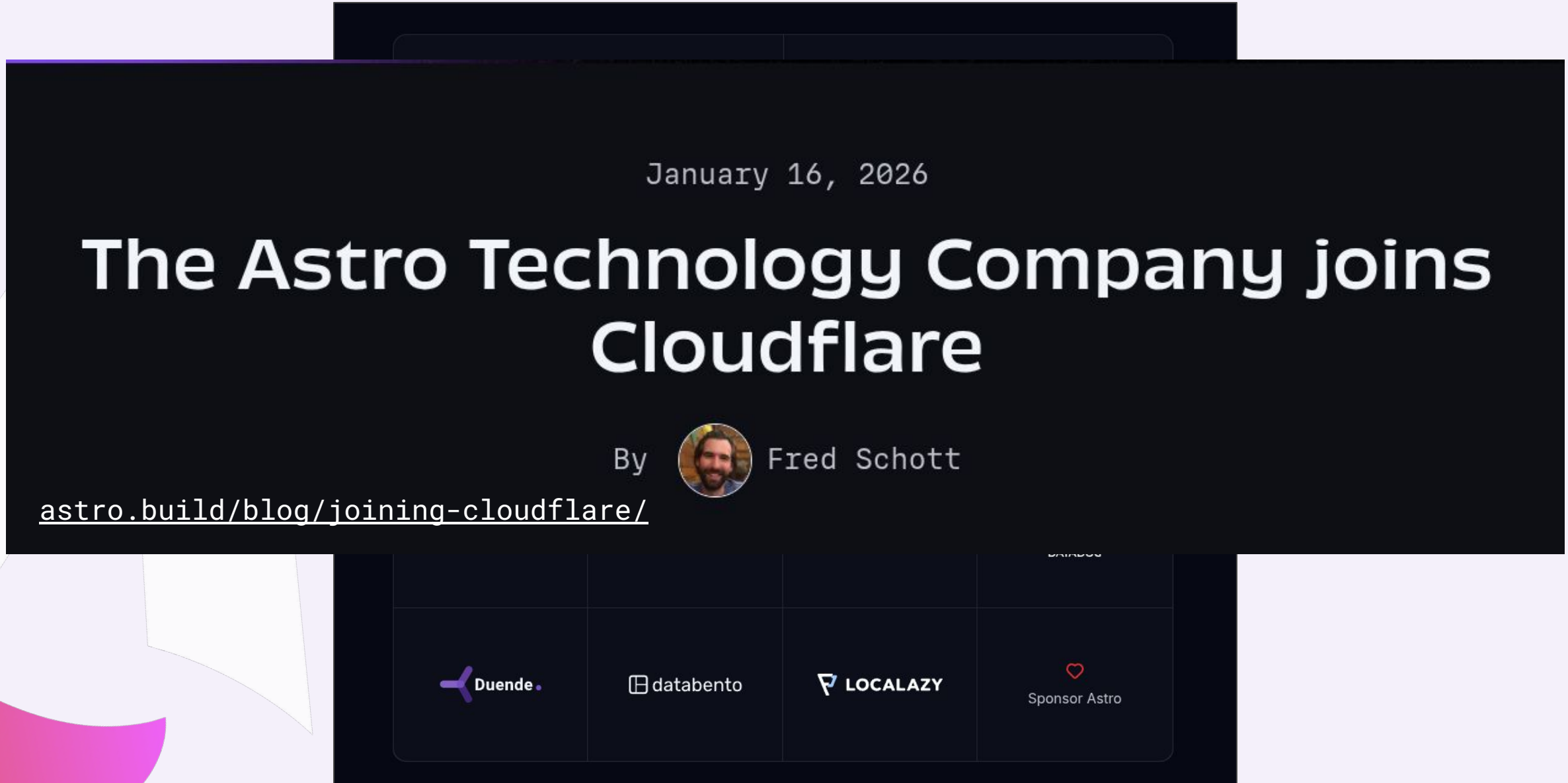
La risposta a un ecosistema colmo di framework per SPA

- Framework open-source – 57K⭐ su GitHub (withastro/astro)
- Creato da Fred Schott e Nate Moore – Giugno 2021
- Costruito sopra a Vite
- Nato come static site generator, evolutosi in un web framework completo
- Ideale per siti che richiedono caricamenti rapidi e ottima SEO: blog, siti di marketing, e-commerce, documentazione, portfolio,...

Sponsorizzato da



Sponsorizzato da

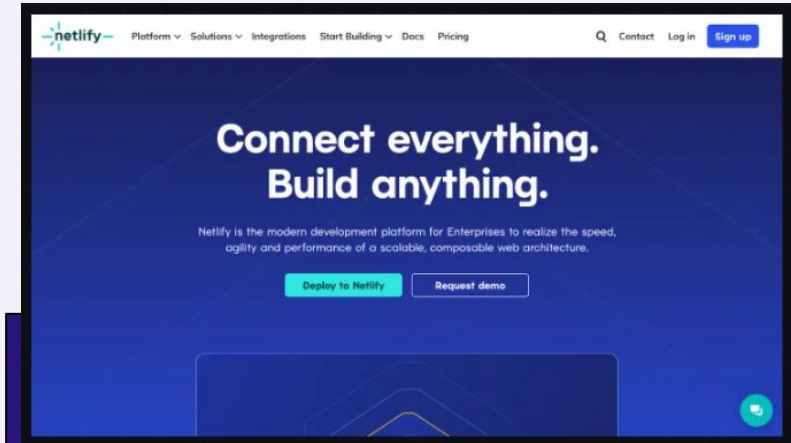


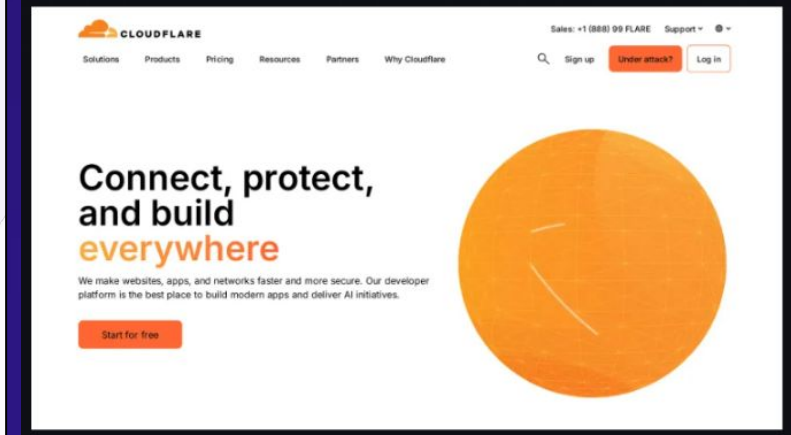
Da chi è **utilizzato**

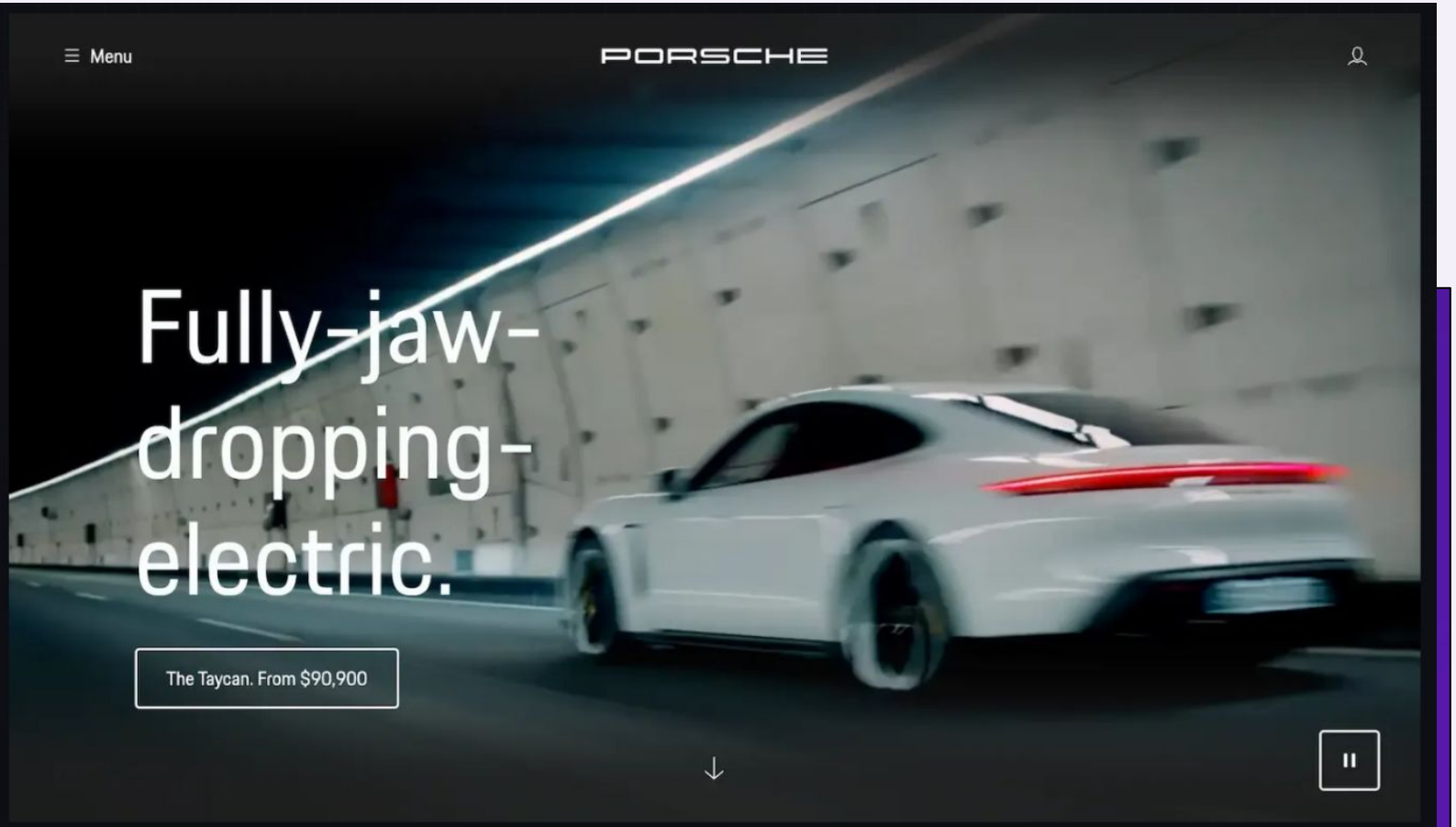
Me

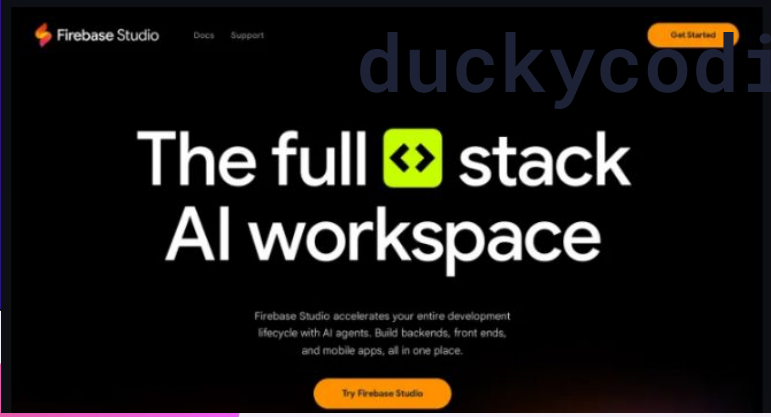
duckycoding.dev

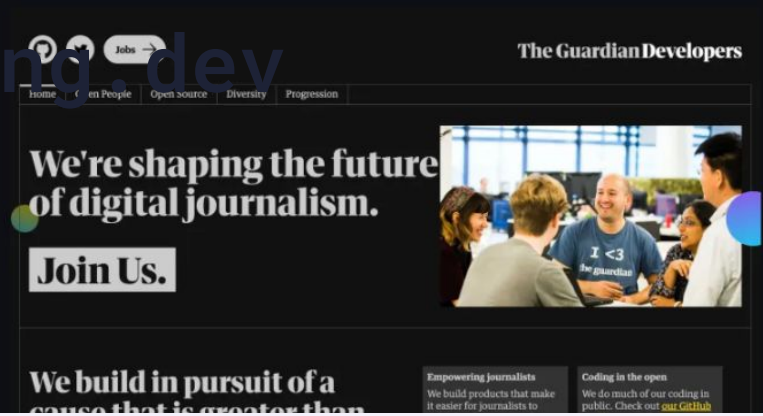
Da chi è utilizzato

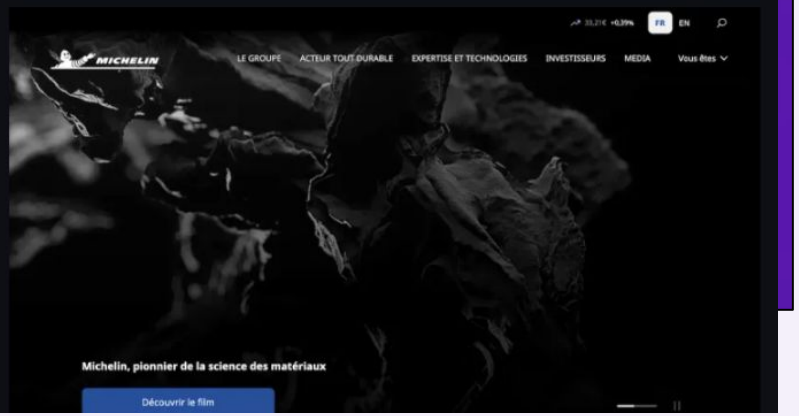












astro.build/showcase/

Punti chiave dell'approccio Astro

01 Zero JS

0KB di JS inviati al browser, di default

02 Server-first

Il server renderizza, il browser mostra HTML e basta... PHP sei tu?

03 Isole

Piccole "isole" di interattività in un mare di HTML

04 UI-agnostico

Supporta React, Preact, Svelte, Vue, Solid, HTMX, web components, e altro

05 Veloce

Se il tuo sito non è veloce, nessuno si ferma a guardarne i contenuti

06 Facile uso

Linguaggio superset di HTML/JSX, scoped CSS, TypeScript,...

Vediamo le basi!

Framework a confronto

Astro 🚀

- zero js di default
- zero hydration
- ideale per siti di contenuti
- SEO score
- possibilità di usare diversi
- SSG e SSR nativi, ISR tramite Cache-Control header
- supporto per file MD/MDX
- open source
- ...

Next.js ⚛️

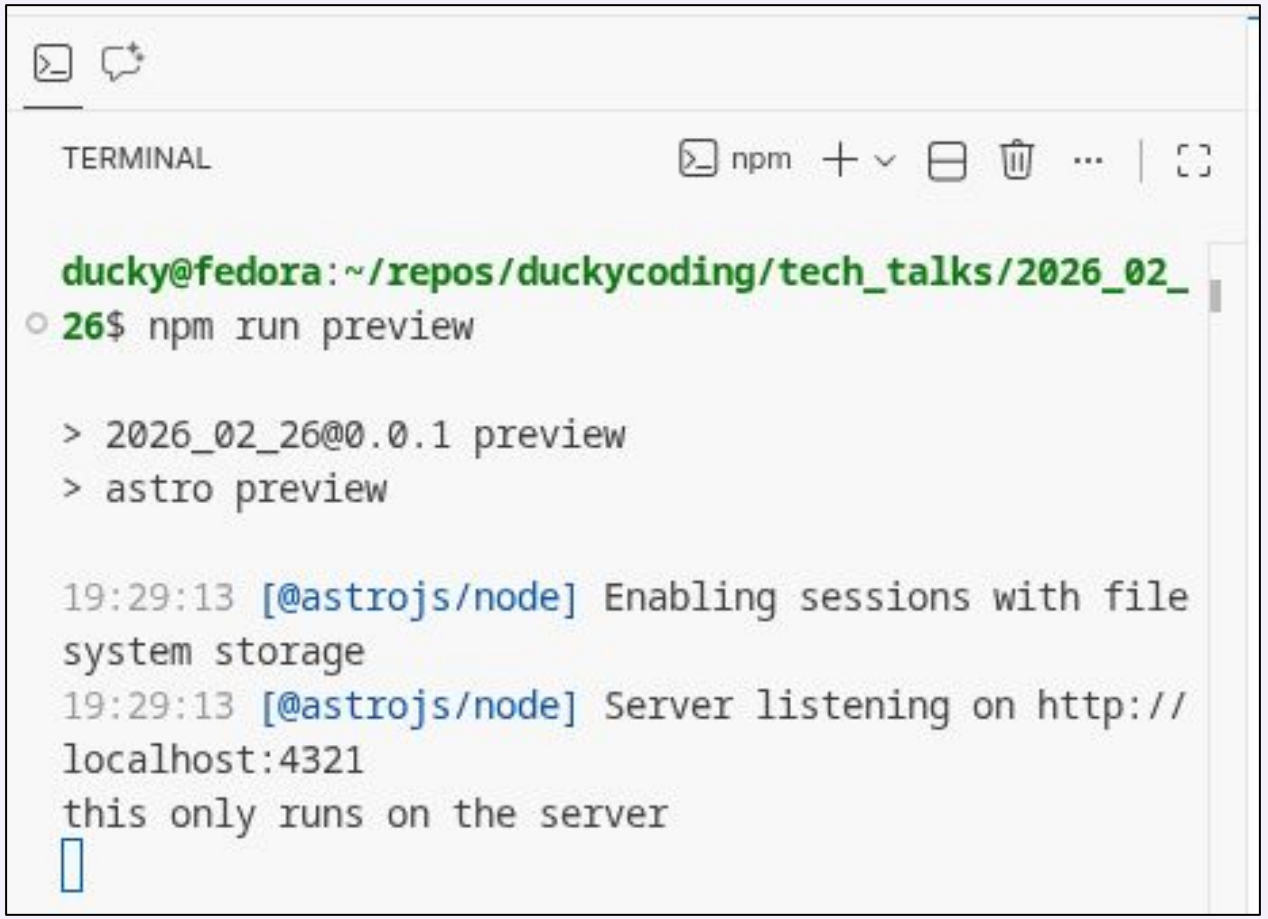
- bundle js importante
- hydration per qualsiasi cosa
- ideale per alta interattività
- solo React e suo ecosistema
- SSG, SSR e ISR built-in
- più vecchio -> user base più ampia, più fonti
- closed source

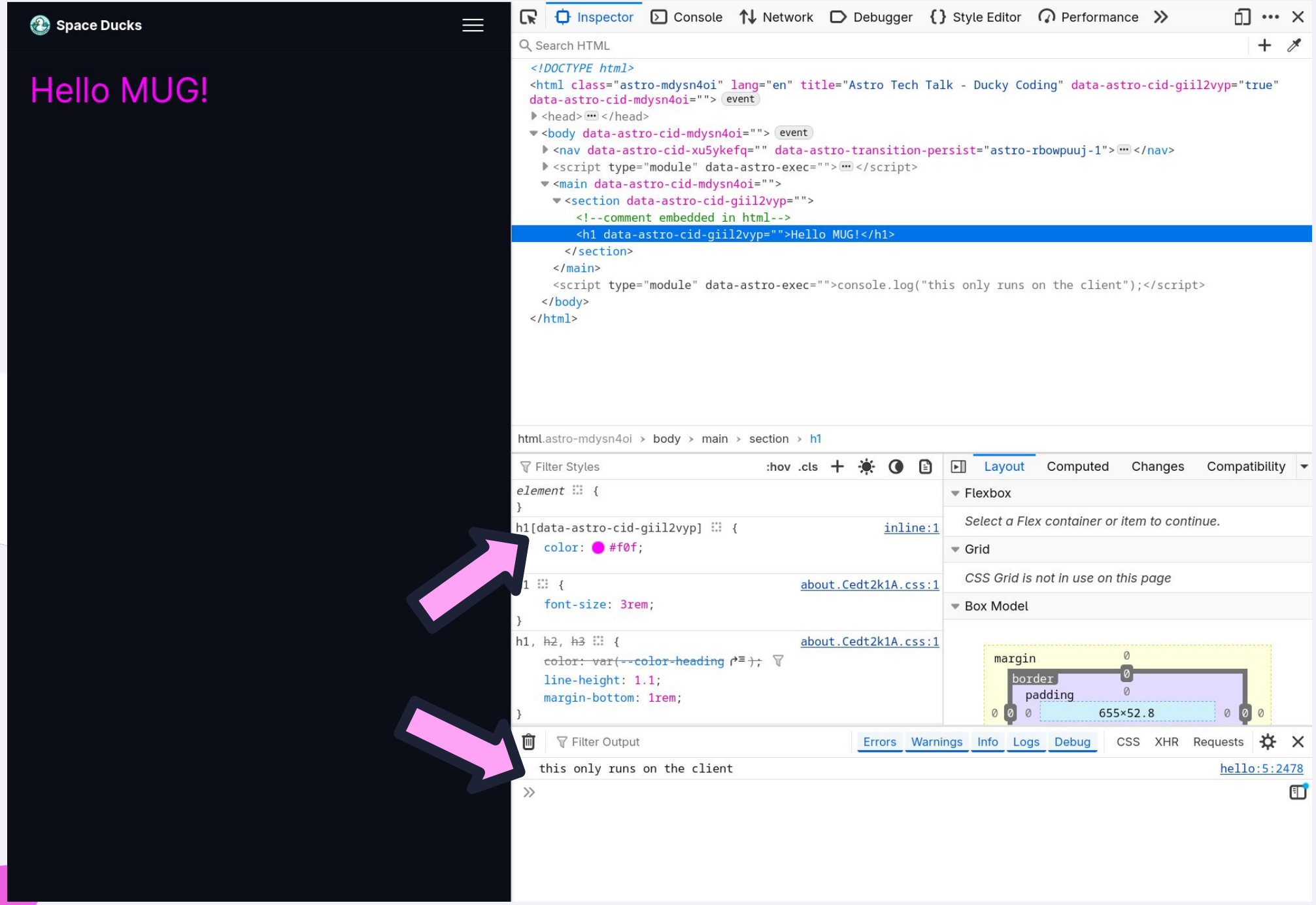
Struttura progetto

- `src/pages/` è l'unica cartella obbligatoria
- `public/` viene copiata così com'è nella build

File `.astro`

```astro
---
// server side code and imports go here
import '@/styles/global.css';
import RootLayout from '../layouts/RootLayout.astro';

// top level await
const result = await fetch('...');

console.log('this only runs on the server');
const greetings = 'Hello MUG!';
---

<RootLayout title="Astro Tech Talk - Ducky Coding">
  <section>
    {/* comment omitted from HTML */}
    <!-- comment embedded in html -->
    <h1>{greetings}</h1>
  </section>
</RootLayout>

<script>
  // client side code goes here
  console.log('this only runs on the client');
</script>

<style>
  /* Automatically scoped CSS thanks to "data-astro-cid-" generated classnames */

  h1 {
    color: fuchsia;
  }
</style>

```

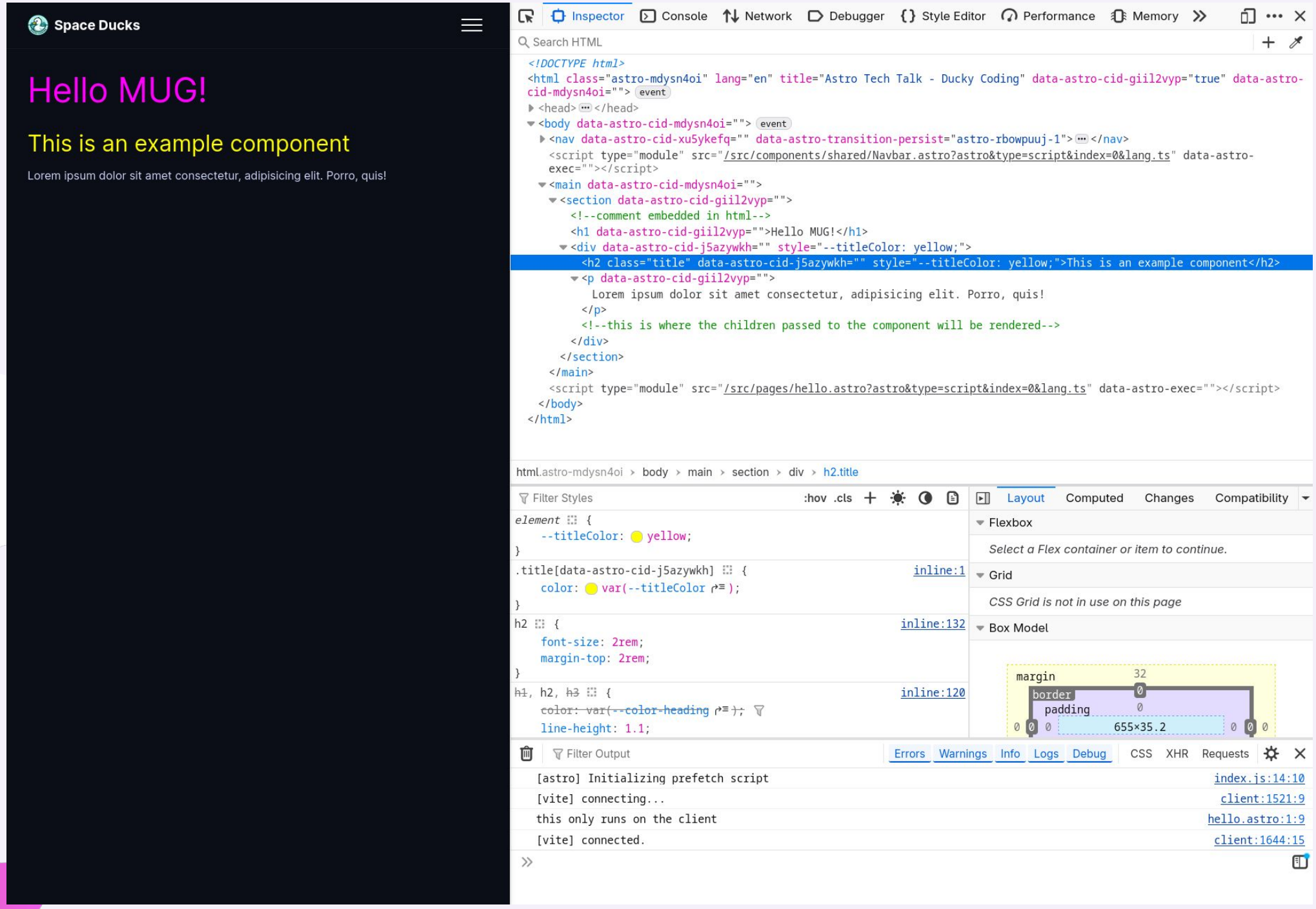

TERMINAL
npm
ducky@fedora:~/repos/duckycoding/tech_talks/2026_02_
26$ npm run preview
> 2026_02_26@0.0.1 preview
> astro preview
19:29:13 [@astrojs/node] Enabling sessions with file
system storage
19:29:13 [@astrojs/node] Server listening on http://
localhost:4321
this only runs on the server


Space Ducks
Hello MUG!
Inspector
Console
Network
Debugger
Style Editor
Performance
Search HTML
<!DOCTYPE html>
<html class="astro-mdysn4oi" lang="en" title="Astro Tech Talk - Ducky Coding" data-astro-cid-giil2vyp="true" data-astro-cid-mdysn4oi="">
<head>…</head>
<body data-astro-cid-mdysn4oi="">
<nav data-astro-cid-xu5ykefq="" data-astro-transition-persist="astro-rbowpuuj-1">…</nav>
<script type="module" data-astro-exec="">…</script>
<main data-astro-cid-mdysn4oi="">
<section data-astro-cid-giil2vyp="">
<!--comment embedded in html-->
<h1 data-astro-cid-giil2vyp="">Hello MUG!</h1>
</section>
</main>
<script type="module" data-astro-exec="">console.log("this only runs on the client");</script>
</body>
</html>
html.astro-mdysn4oi › body › main › section › h1
Filter Styles
:hov .cls
element { }
h1[data-astro-cid-giil2vyp] { color: #f0f; } inline:1
1 { font-size: 3rem; } about.Cedt2k1A.css:1
h1, h2, h3 { color: var(--color-heading); line-height: 1.1; margin-bottom: 1rem; } about.Cedt2k1A.css:1
Layout
Computed
Changes
Compatibility
Flexbox
Select a Flex container or item to continue.
Grid
CSS Grid is not in use on this page
Box Model
margin
border
padding
655×52.8
Filter Output
Errors
Warnings
Info
Logs
Debug
CSS
XHR
Requests
this only runs on the client
hello:5:2478

Componenti .astro

```astro
---
// example-component.astro
import type { HTMLAttributes } from 'astro/types';

interface Props extends HTMLAttributes<'div'> {
  // define your props here
  titleColor: string;
}
const { titleColor } = Astro.props; // automatically typed as Props

// automatically exported as default export
---

<div>
  <h2 class="title">This is an example component</h2>
  <slot /><!-- this is where the children passed to the component will be rendered -->
</div>

<style define:vars={{titleColor}}>
  .title {
    /* Automatically scoped */
    color: var(--titleColor);
  }
  /* ... */
</style>
```

```astro
---
import ExampleComponent from '../components/example-component.astro';
---

<ExampleComponent titleColor="yellow">
  <p>
    Lorem ipsum dolor sit amet consectetur,
    adipisicing elit. Porro, quis!
  </p>
</ExampleComponent>
```


Space Ducks
Hello MUG!
This is an example component
Lorem ipsum dolor sit amet consectetur, adipisicing elit. Porro, quis!
Inspector
Console
Network
Debugger
Style Editor
Performance
Memory
Search HTML
<!DOCTYPE html>
<html class="astro-mdysn4oi" lang="en" title="Astro Tech Talk - Ducky Coding" data-astro-cid-giil2vyp="true" data-astro-cid-mdysn4oi="">
<head>…</head>
<body data-astro-cid-mdysn4oi="">
<nav data-astro-cid-xu5ykefq="" data-astro-transition-persist="astro-rbowpuuj-1">…</nav>
<script type="module" src="/src/components/shared/Navbar.astro?astro&type=script&index=0&lang.ts" data-astro-exec=""></script>
<main data-astro-cid-mdysn4oi="">
<section data-astro-cid-giil2vyp="">
<!--comment embedded in html-->
<h1 data-astro-cid-giil2vyp="">Hello MUG!</h1>

<h2 class="title" data-astro-cid-j5azywkh="" style="--titleColor: yellow;">This is an example component</h2>

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Porro, quis!

<!--this is where the children passed to the component will be rendered-->

</section>
</main>
<script type="module" src="/src/pages/hello.astro?astro&type=script&index=0&lang.ts" data-astro-exec=""></script>
</body>
</html>
html.astro-mdysn4oi › body › main › section › div › h2.title
Filter Styles
:hov .cls
element { --titleColor: yellow; }
.title[data-astro-cid-j5azywkh] { color: var(--titleColor); } inline:1
h2 { font-size: 2rem; margin-top: 2rem; } inline:132
h1, h2, h3 { color: var(--color-heading); line-height: 1.1; } inline:120
Layout
Computed
Changes
Compatibility
Flexbox
Select a Flex container or item to continue.
Grid
CSS Grid is not in use on this page
Box Model
margin 32
border 0
padding 0
655×35.2
Filter Output
Errors
Warnings
Info
Logs
Debug
CSS
XHR
Requests
[astro] Initializing prefetch script index.js:14:10
[vite] connecting... client:1521:9
this only runs on the client hello.astro:1:9
[vite] connected. client:1644:15

Script JavaScript (1)

Di default Astro processa i tag `<script>` che non contengono alcun attributo (a parte `src`) in questo modo:

- supportano TypeScript di default
- file locali e moduli NPM vengono “bundlati” insieme
- `type=”module”` assegnato automaticamente
- se un componente contiene uno script e viene usato più volte in una pagina, lo script viene incluso una volta sola
- se lo script è piccolo viene incluso inline nell’HTML

Script JavaScript (2)

- Se il tag `<script>` contiene degli attributi o se contiene la direttiva `is:inline` non verrà processato
- È possibile caricare script .js o .ts definiti in file locali in `src/`:

```
<!-- relative path to script at `src/scripts/local.js` -->
<script src="../scripts/local.js"></script>

<!-- also works for local TypeScript files -->
<script src="./script-with-types.ts"></script>
```

- O usare script js esterni definiti in `public/` o in remoto usando la direttiva `is:inline` e path assoluti:

```
<!-- absolute path to a script at `public/my-script.js` -->
<script is:inline src="/my-script.js"></script>

<!-- full URL to a script on a remote server -->
<script is:inline src="https://my-analytics.com/script.js"></script>
```

Modalità di rendering

- Di default tutte le pagine sono generate staticamente
- Modificabile in `astro.config.mjs` -> `output: ‘server’`
- Override per pagina tramite export:

```
---
// with astro.config.mjs `output: 'static' (default)
export const prerender = false
---
<!-- server-rendered content -->
<!-- the rest of my site is static -->
```

```
---
// with astro.config.mjs `output: 'server'
export const prerender = true
---
<!-- statically-rendered content -->
<!-- the rest of my site is dynamic -->
```

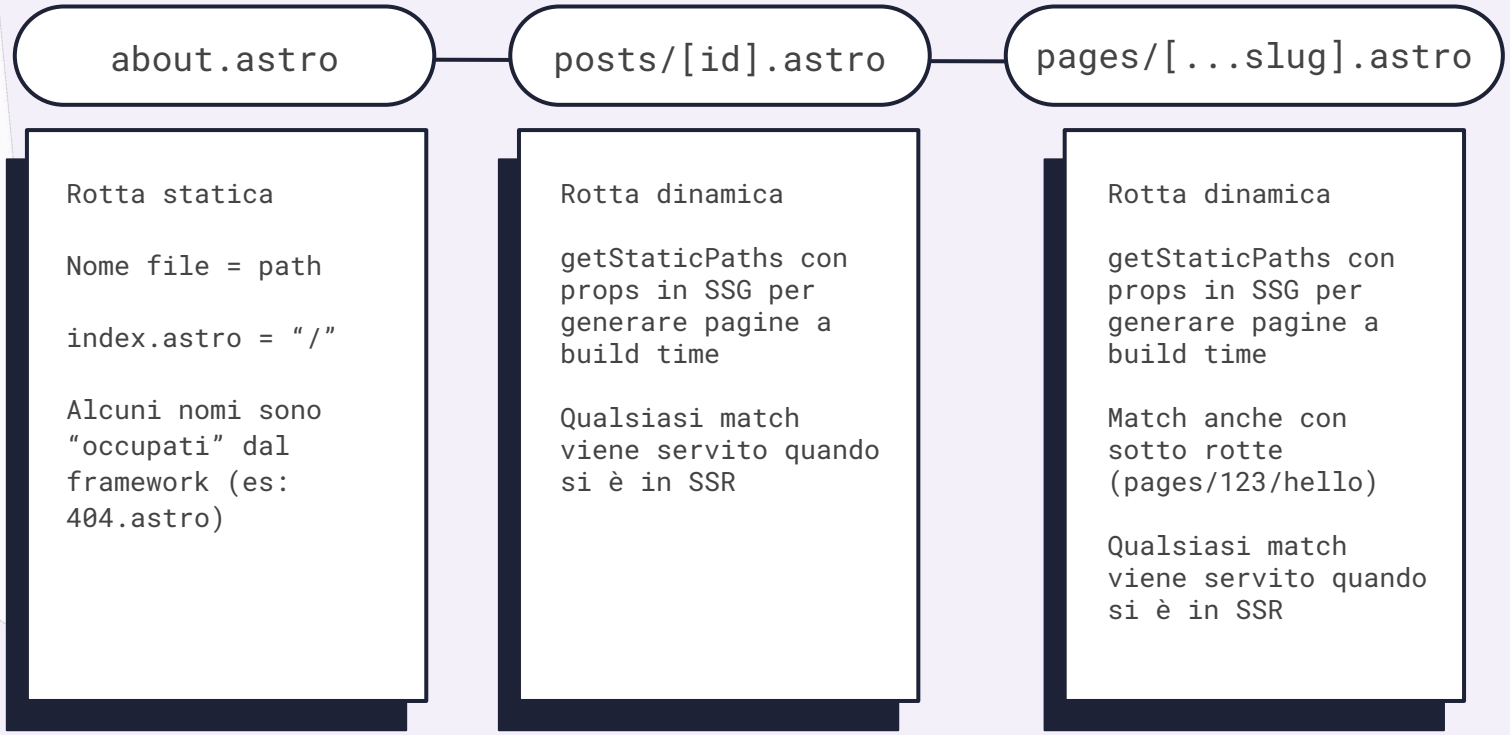
Con questo terminiamo le basi…

…e iniziamo a vedere alcuni elementi chiave del framework.

La documentazione: astro.build/en/getting-started/

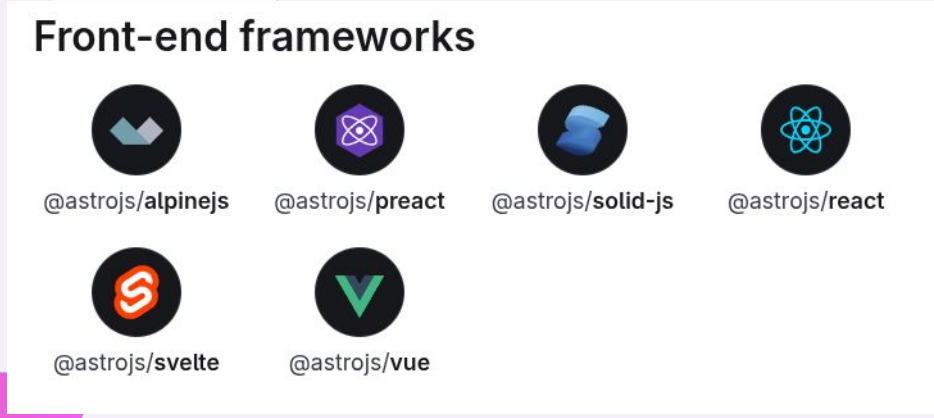
Routing

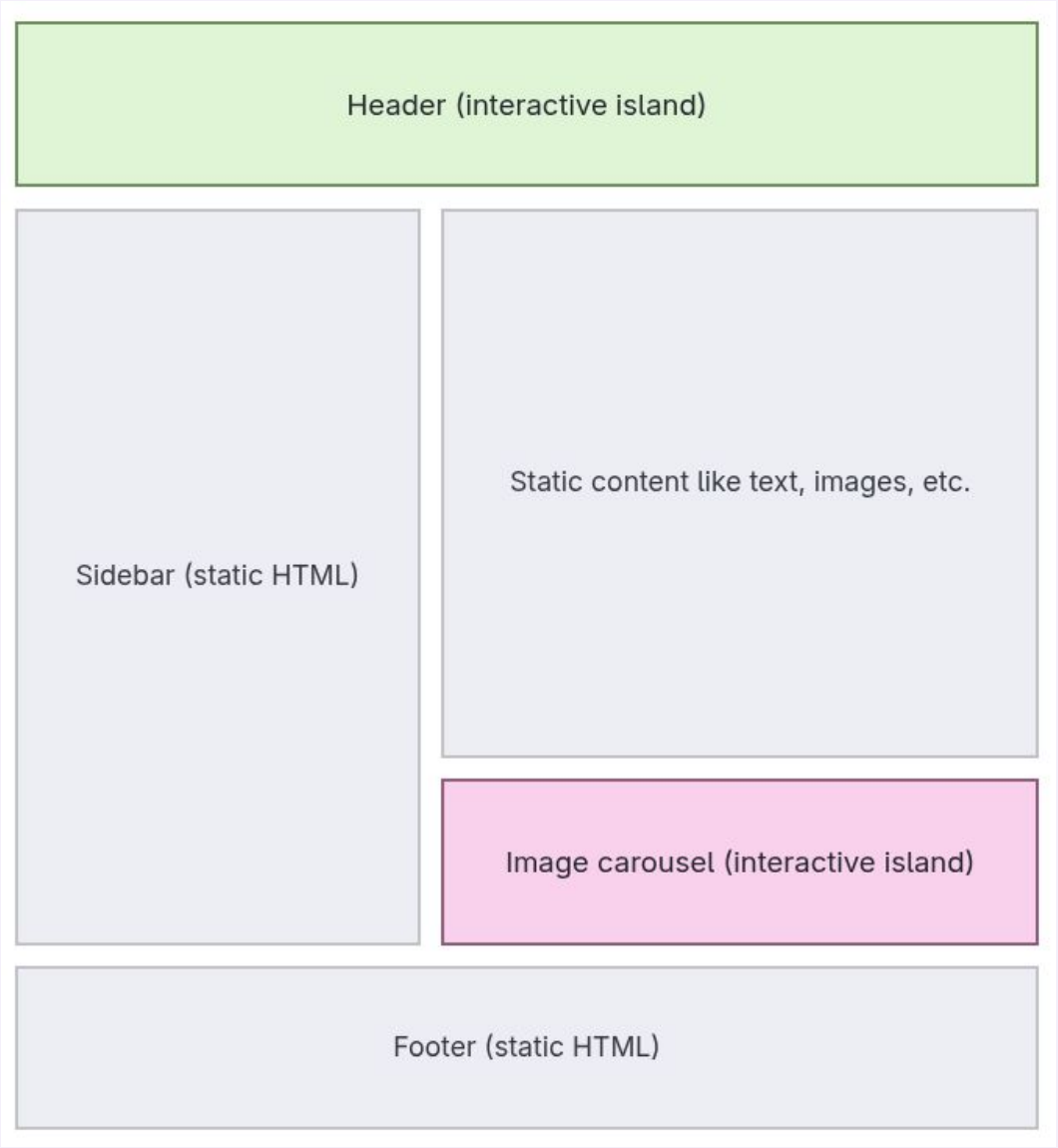
- Navigazione con `<a href="">...</a>`
- prefetch con `data-astro-prefetch="..."` o config
- file-based routing: file all'interno di `src/pages/`
- tipi di file:



Isole

- Contenuto meno importante per SEO
- Idratazione solo dove e quando necessaria
- Elementi “widget” isolati
- Componenti fatti anche con framework diversi
- Richiede avere JS abilitato





Tipi di isole 

Client Island

- Direttive `client:*` (dopo)
- Senza direttiva, puro HTML
- Solo props serializzabili *
- UI framework a scelta
- Indipendenti tra loro
- Accettano children

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

Server Island

- Direttiva `server:defer`
- Solo props serializzabili *
- Render asincrono
- Contenuto di fallback/default
- Endpoint apposito
- Indipendenti tra loro
- Solo componenti `.astro`

```
---
import Avatar from "../components/Avatar.astro";
---
<Avatar server:defer />
```

Direttive client:*

- Controllano come i componenti fatti con framework UI vengono renderizzati e idratati
- Customizzabili + ne esistono di default:

:load	:idle	:visible	:media	:only
Carica e idrata il JS appena la pagina si è caricata	Carica e idrata il JS quando avviene l'evento requestIdleCallback timeout custom Fallback a :load per browser che non supportano requestIdleCallback (es. Safari)	Carica e idrata il JS quando il componente entra nella viewport rootMargin per anticipare il rendering (es: 200px)	Carica e idrata il JS quando una CSS media query viene triggerata es: client:media="(min-width: 1000px)"	Carica e idrata il JS appena la pagina si è caricata e esegue il primo render Fallback content Serve specificare il framework, es: client:only="react"

In tutti i casi, tranne `client:only`, vengono renderizzati lato server e poi solo idratati lato client

Immagini

- SVG direttamente importabili (solo) in file `.astro`
- Componenti `<Image />` e `<Picture />`
  - Ottimizzazione automatica immagini locali
  - Formati multipli (jpeg, png, webp, **avif**)
  - Proprietà responsive (srcset e sizes)
  - Gestione Cumulative Layout Shift (CLS)
- Immagini processate cachate tra build
- API per lavorare con immagini anche lato server
  - es: `getImage()` per usare immagini in posti diversi dall'HTML
- Domini autorizzati per immagini remote

Content collections (1)

- “Set di dati strutturalmente simili”
  - es. Cartella coi post di un blog
  - es. Singoli file con liste di elementi
- File locali o remoti: Markdown, MDX, JSON, YAML,...
- Strutture type safe definendo schemi con Zod
- Loaders: funzioni per recuperare e parsare i dati
  - built-in: `glob()` e `file()`
- Per ogni entry della collezione il loader deve ritornare un campo `id` univoco
- Cache tra build

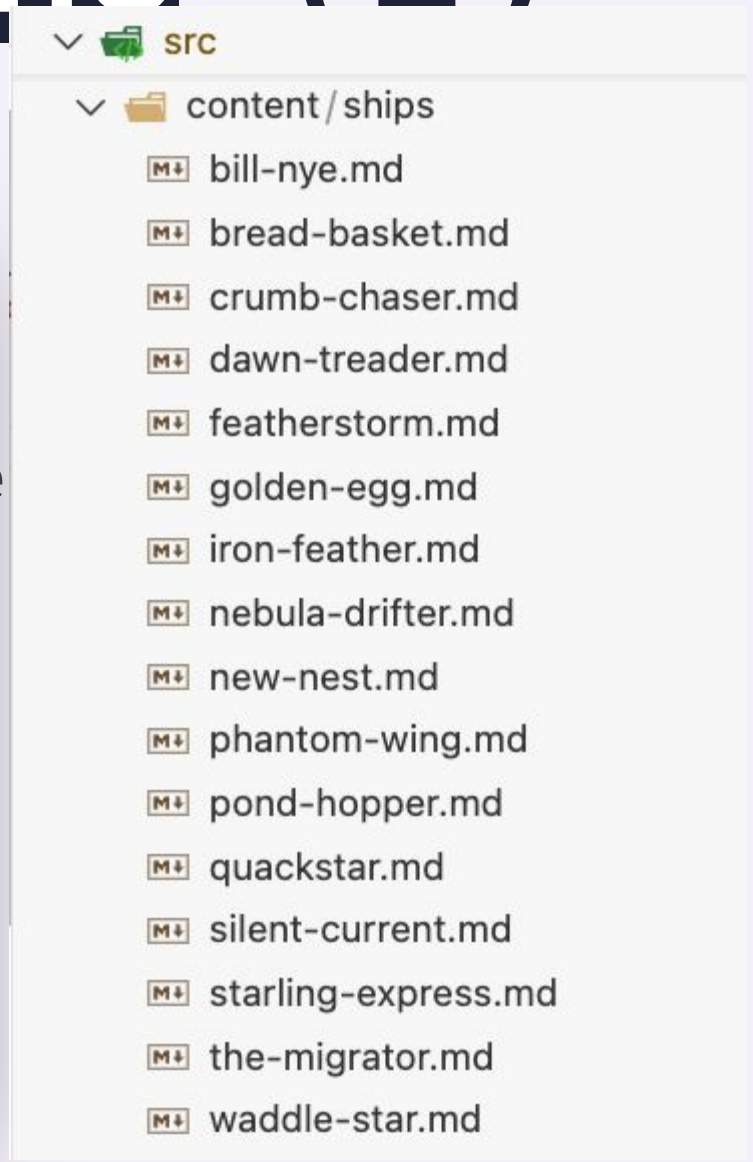
Content collections (2)

- API essenziali da `astro:content`:
  - z() (zod re-export)
  - defineCollection({loader, schema})
  - getCollection(collectionName, filterCallback?)
  - getEntry(collectionName, idEntry)
  - render(entry)

Content collections (2)

- API essenziali da `astro:content`:

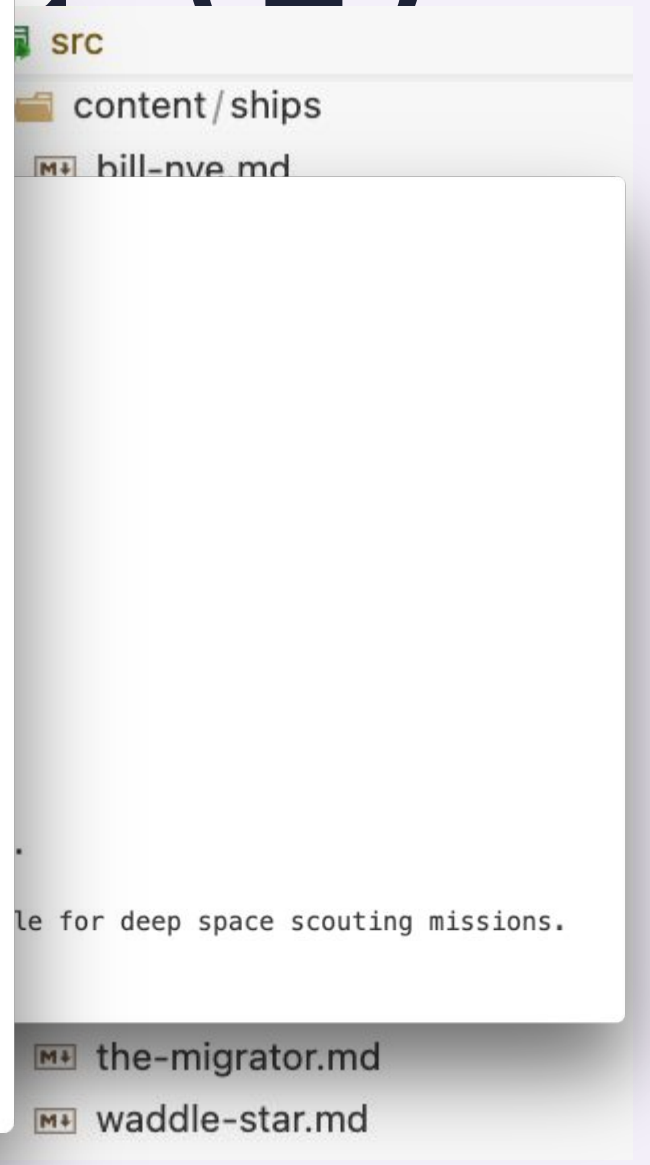
```ts
// src/content.config.ts
import { defineCollection, z } from 'astro:content';
import { glob } from 'astro/loaders';

const ships = defineCollection({
  // loader automatically generates id from filename
  loader: glob({ pattern: '**/*.md', base: './src/content/ships' }),
  schema: z.object({
    name: z.string(),
    // ... other fields as needed
  }),
});

// collection name is the key in this export
export const collections = { ships };

```



Content collections (2)

```yaml
---
name: "QSS Quackstar"
class: "Explorer"
status: "Active"
crew: 142
commissioned: 2024-03-15
topSpeed: "Warp 7.5"
armament: ["Photon Torpedoes",
homePort: "Pond Zero"
captain: "Commander Mallard"
---


# Quackstar title from .md \#
The QSS Quackstar is the pride
Launched from Pond Zero's orbi
Her crew discovered three habi

The Quackstar's advanced sensc

```

```astro
---
import RootLayout from '@/layouts/RootLayout.astro';
import { getEntry, render } from 'astro:content';

const { id } = Astro.params;
if (!id) {
  return Astro.redirect('/demos/5-content');
}

const ship = await getEntry('ships', id);

if (!ship) {
  return Astro.redirect('/demos/5-content');
}

const { Content } = await render(ship);
---

<RootLayout title={ship.data.name}>
  {/* renders HTML generated by md content */}
  <Content />
</RootLayout>

```

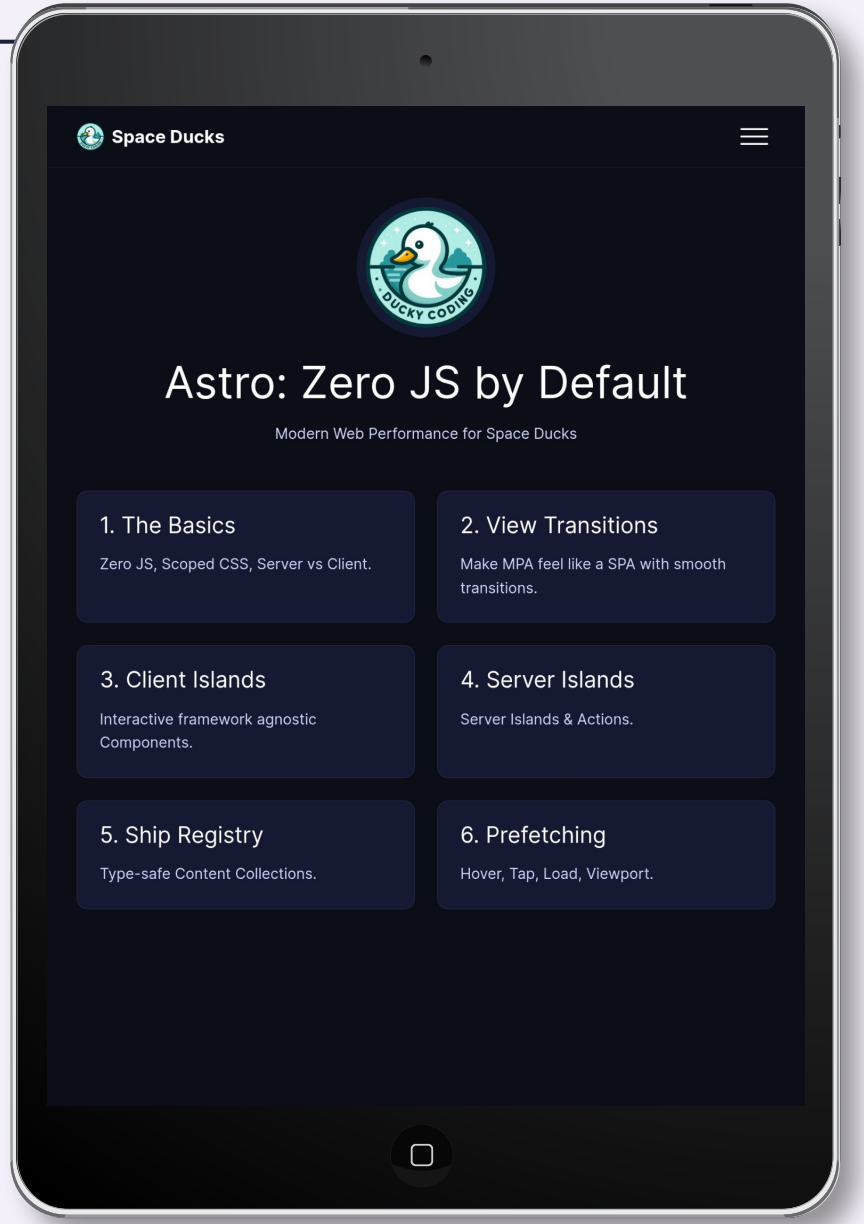


View transitions

- `<ClientRouter>`: permette navigazione stile SPA
- intercetta le navigazioni aggiungendo delle feature
- alcune transizioni automatiche
- elementi condivisi pagine gestiti: `transition:persist`
- animazioni pre-esistenti con `transition:animate`: fade, slide, none
- morphing di elementi tra pagine con `transition:name`
- fallback a navigazione normale su browser non supportati

Vediamo Astro all'opera

Una piccola demo per mostrare le feature principali di cui abbiamo parlato finora



Alcune delle altre feature

Actions

Font API

Env API

Server endpoint

Middleware

i18n

Integrazioni

Servizi terzi

Content Layer API

...

Tempo per le domande

Credits

- Host: Wavelop & MUG
- Template presentazione: SlidesMania
- Font utilizzato: Roboto Mono
- Repo GitHub Astro
- Repo GitHub col codice della demo

Link interessanti

- Astro guide: developing with LLM
- Nickyt.co: Approfondimento server islands
- Theo - t3 video: Server Island vs Next.js PPR
- Netlify: ISR e caching con Astro

Grazie a tutti!

THANK YOU

Per rimanere in contatto…

Se e quando scriverò qualcosa…