

Performance stellari per siti web orientati ai **contenuti.**

Davide Milan - Full Stack developer @Wavelop

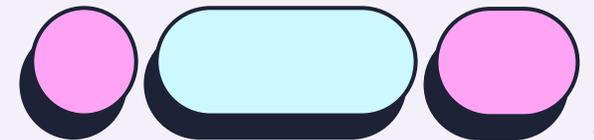


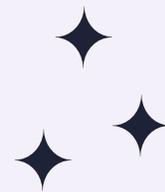
Hello! I'm...

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

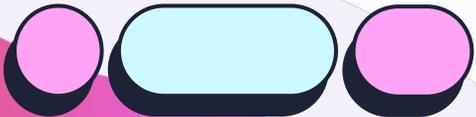


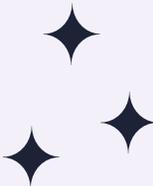


“

*The web framework for
content-driven websites*

– astro.build





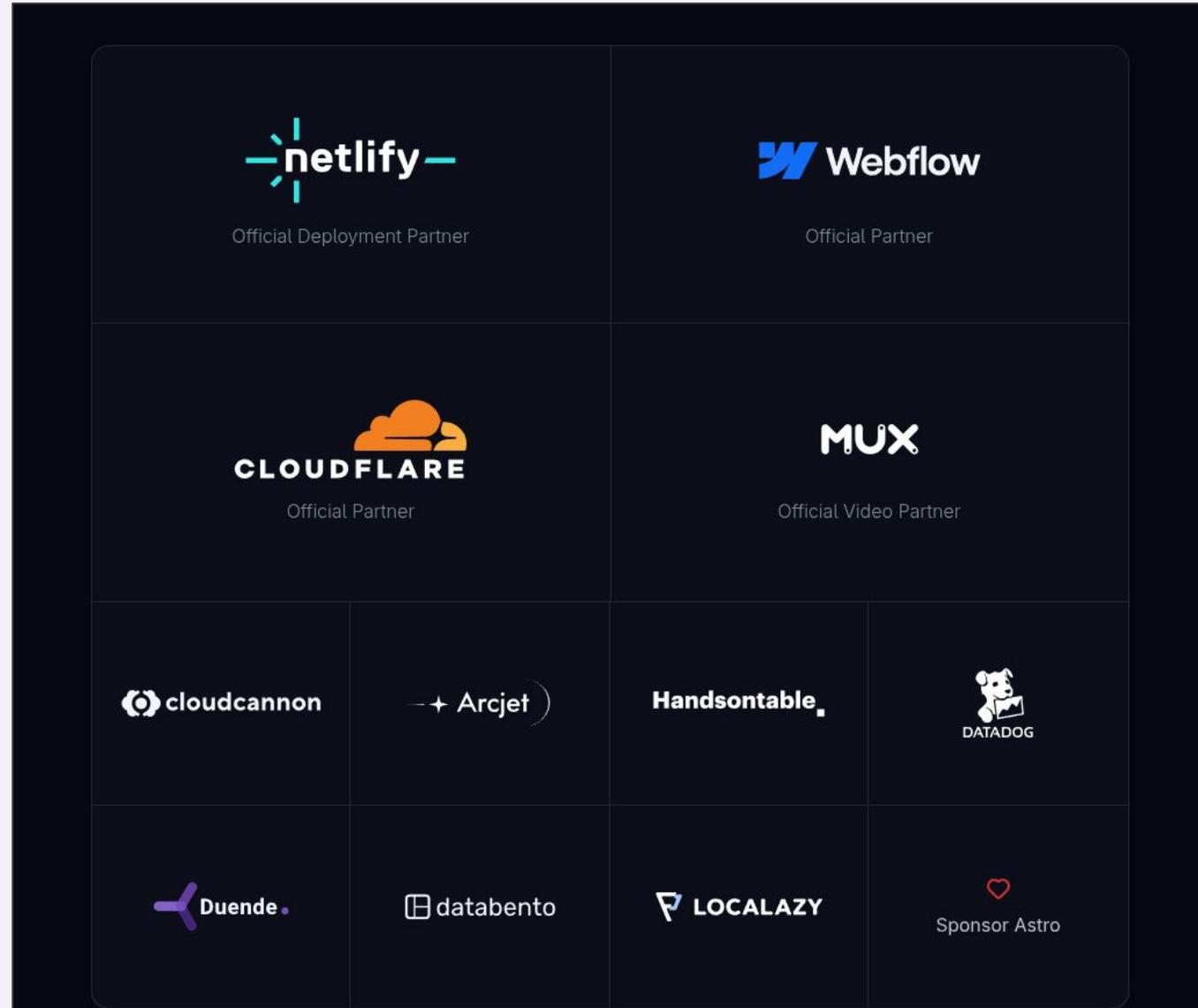
Cos'è Astro?

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,...

O
R
I
G
I
N
I

Sponsorizzato da



Sponsorizzato da

January 16, 2026

The Astro Technology Company joins Cloudflare

By



Fred Schott

astro.build/blog/joining-cloudflare/



 databento

 LOCALAZY


Sponsor Astro

Da chi è **utilizzato**



duckycoding.dev



Da chi è **utilizzato**

Google



VISA

Microsoft

PORSCHE

OpenAI



NordVPN®



CLOUDFLARE®

NBC NEWS

The Guardian



MICHELIN

Proton

duckycoding.dev

Webflow

netlify Platform Solutions Integrations Start Building Docs Pricing Contact Log in Sign up

Connect everything. Build anything.

Netlify is the modern development platform for Enterprises to realize the speed, agility and performance of a scalable, composable web architecture.

Deploy to Netlify Request demo

CLOUDFLARE Sales: +1 (888) 99 FLARE Support

Solutions Products Pricing Resources Partners Why Cloudflare Sign up Show attack? Log in

Connect, protect, and build everywhere

We make websites, apps, and networks faster and more secure. Our developer platform is the best place to build modern apps and deliver AI initiatives.

Start for free

Firebase Studio Docs Support Get Started

The full stack AI workspace

Firebase Studio accelerates your entire development lifecycle with AI agents. Build backends, front ends, and mobile apps, all in one place.

Try Firebase Studio

Menu PORSCHE

Fully-jaw-dropping-electric.

The Taycan. From \$90,900

The Guardian Developers

Home Open People Open source Diversity Progression Jobs

We're shaping the future of digital journalism.

Join Us.

We build in pursuit of a cause that is greater than...

Empowering journalists We build products that make it easier for journalists to...

Coding in the open We do much of our coding in public. Check out our [GitHub](#)

MICHELIN LE GROUPE ACTEUR TOUT DURABLE EXPERTISE ET TECHNOLOGIES INVESTISSEURS MEDIA Vous êtes

Michelin, pionnier de la science des matériaux

Découvrir le film

astro.build/showcase/

Punti chiave dell'approccio Astro

01 Zero JS

0KB di JS inviati al browser, di default

04 UI-agnostico

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

02 Server-first

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

05 Veloce

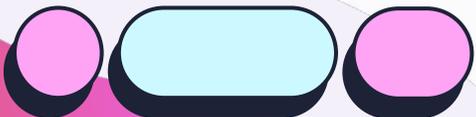
Se il tuo sito non è veloce, nessuno si ferma a guardarne i contenuti

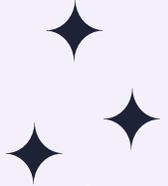
03 Isole

Piccole "isole" di interattività in un mare di HTML

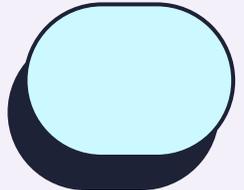
06 Facile uso

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

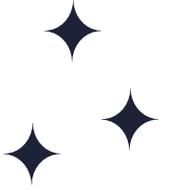




Vediamo le **basi!**



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

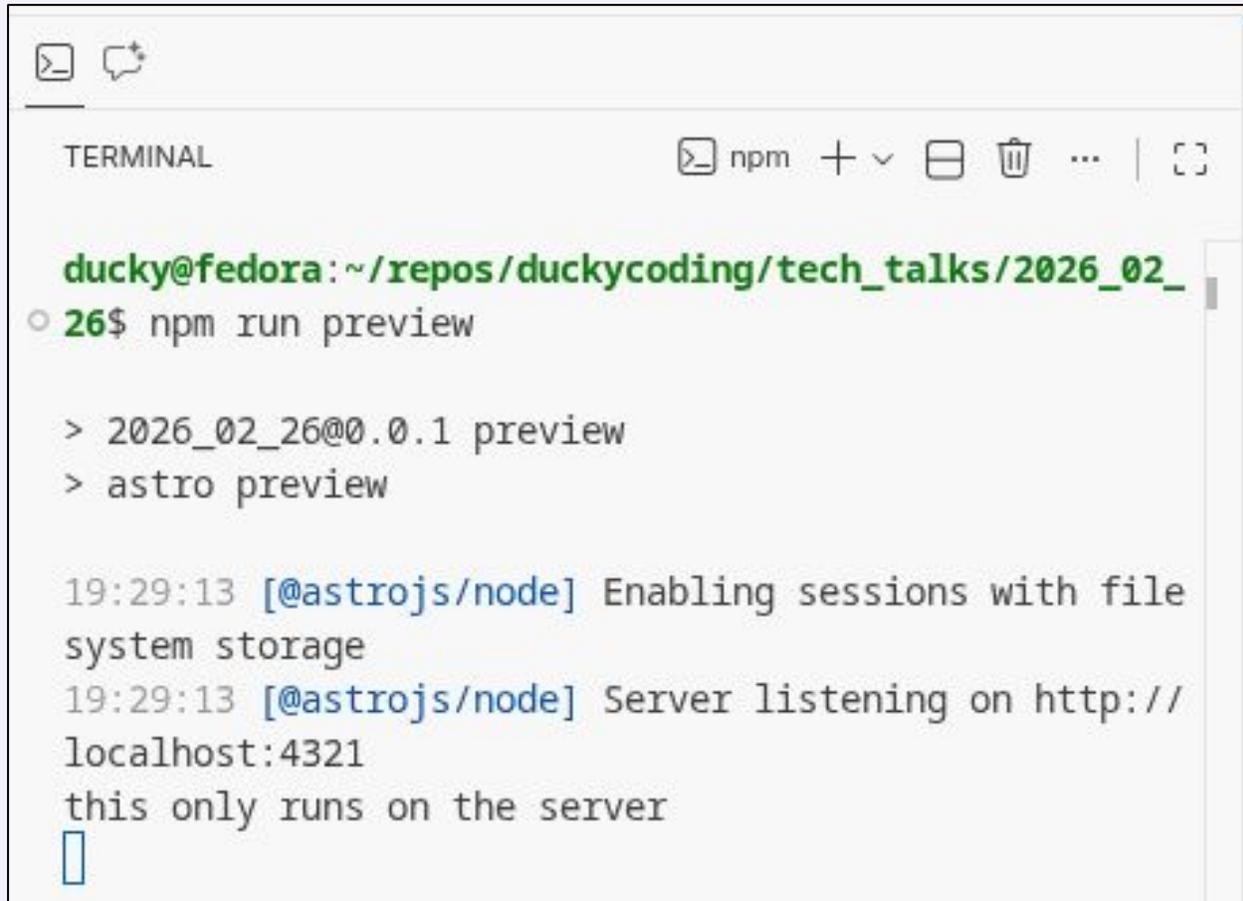
Struttura progetto



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

File `.astro`

```
1 ---
2 // server side code and imports go here
3 import '@/styles/global.css';
4 import RootLayout from '../layouts/RootLayout.astro';
5
6 // top level await
7 const result = await fetch('...');
8
9 console.log('this only runs on the server');
10 const greetings = 'Hello MUG!';
11 ---
12
13 <RootLayout title="Astro Tech Talk - Ducky Coding">
14   <section>
15     /* comment omitted from HTML */
16     <!-- comment embedded in html -->
17     <h1>{greetings}</h1>
18   </section>
19 </RootLayout>
20
21 <script>
22   // client side code goes here
23   console.log('this only runs on the client');
24 </script>
25
26 <style>
27   /* Automatically scoped CSS thanks to "data-astro-cid-" generated classnames */
28
29   h1 {
30     color: fuchsia;
31   }
32 </style>
33
```



```
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
```

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 + ☀ 🌙 📄

Layout Computed Changes Compatibility

```
element {}
h1[data-astro-cid-giil2vyp] {}
  color: #f0f;
1 {}
  font-size: 3rem;
h1, h2, h3 {}
  color: var(--color-heading);
  line-height: 1.1;
  margin-bottom: 1rem;
```

Flexbox: Select a Flex container or item to continue.

Grid: CSS Grid is not in use on this page.

Box Model: margin: 0, border: 0, padding: 0, 655x52.8

Filter Output Errors Warnings Info Logs Debug CSS XHR Requests

```
this only runs on the client
hello:5:2478
```



Componenti .astro

```
1 ---
2 // example-component.astro
3 import type { HTMLAttributes } from 'astro/types';
4
5 interface Props extends HTMLAttributes<'div'> {
6   // define your props here
7   titleColor: string;
8 }
9 const { titleColor } = Astro.props; // automatically typed as Props
10
11 // automatically exported as default export
12 ---
13
14 <div>
15   <h2 class="title">This is an example component</h2>
16   <slot /><!-- this is where the children passed to the component will be rendered -->
17 </div>
18
19 <style define:vars={{titleColor}}>
20   .title {
21     /* Automatically scoped */
22     color: var(--titleColor);
23   }
24   /* ... */
25 </style>
26
```

```
1 ---
2 import ExampleComponent from '../components/example-component.astro';
3 ---
4
5 <ExampleComponent titleColor="yellow">
6   <p>
7     Lorem ipsum dolor sit amet consectetur,
8     adipiscing elit. Porro, quis!
9   </p>
10 </ExampleComponent>
```

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
Q 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>
        <div data-astro-cid-j5azywkh="" style="--titleColor: yellow;">
          <h2 class="title" data-astro-cid-j5azywkh="" style="--titleColor: yellow;">This is an example component</h2>
          <p data-astro-cid-giil2vyp="">
            Lorem ipsum dolor sit amet consectetur, adipisicing elit. Porro, quis!
          </p>
          <!--this is where the children passed to the component will be rendered-->
        </div>
      </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 + [Layout] Computed Changes Compatibility

```

element :: {
  --titleColor: yellow;
}
.title[data-astro-cid-j5azywkh] :: {
  color: var(--titleColor);
}
h2 :: {
  font-size: 2rem;
  margin-top: 2rem;
}
h1, h2, h3 :: {
  color: var(--color-heading);
  line-height: 1.1;
}

```

Layout: Flexbox, Grid, Box Model

Filter Output Errors Warnings Info Logs Debug CSS XHR Requests

```

[astro] Initializing prefetch script
[vite] connecting...
this only runs on the client
[vite] connected.

```

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/`:

```
1 <!-- relative path to script at `src/scripts/local.js` -->
2 <script src="../../scripts/local.js"></script>
3
4 <!-- also works for local TypeScript files -->
5 <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:

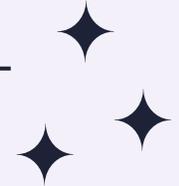
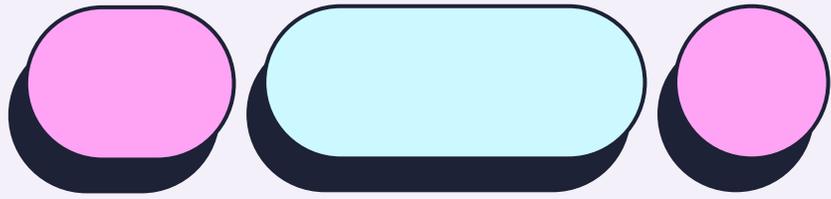
```
1 <!-- absolute path to a script at `public/my-script.js` -->
2 <script is:inline src="/my-script.js"></script>
3
4 <!-- full URL to a script on a remote server -->
5 <script is:inline src="https://my-analytics.com/script.js"></script>
```

Modalità di rendering

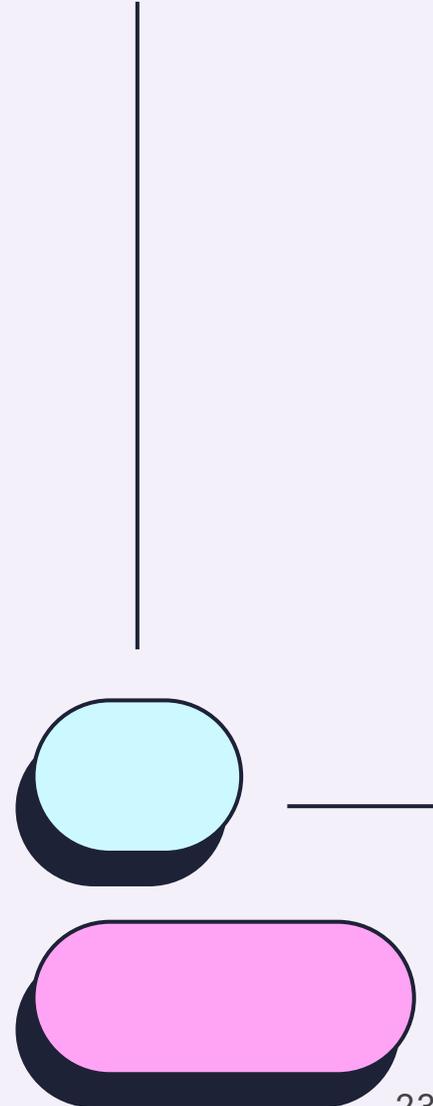
- Di default tutte le pagine sono generate staticamente
- Modificabile in `astro.config.mjs` -> `output: 'server'`
- Override per pagina tramite export:

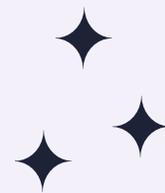
```
1 ---
2 // with astro.config.mjs `output: 'static' (default)
3 export const prerender = false
4 ---
5 <!-- server-rendered content -->
6 <!-- the rest of my site is static -->
```

```
1 ---
2 // with astro.config.mjs `output: 'server'
3 export const prerender = true
4 ---
5 <!-- statically-rendered content -->
6 <!-- 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 `...`
- 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

Front-end frameworks



@astrojs/alpinejs



@astrojs/preact



@astrojs/solid-js



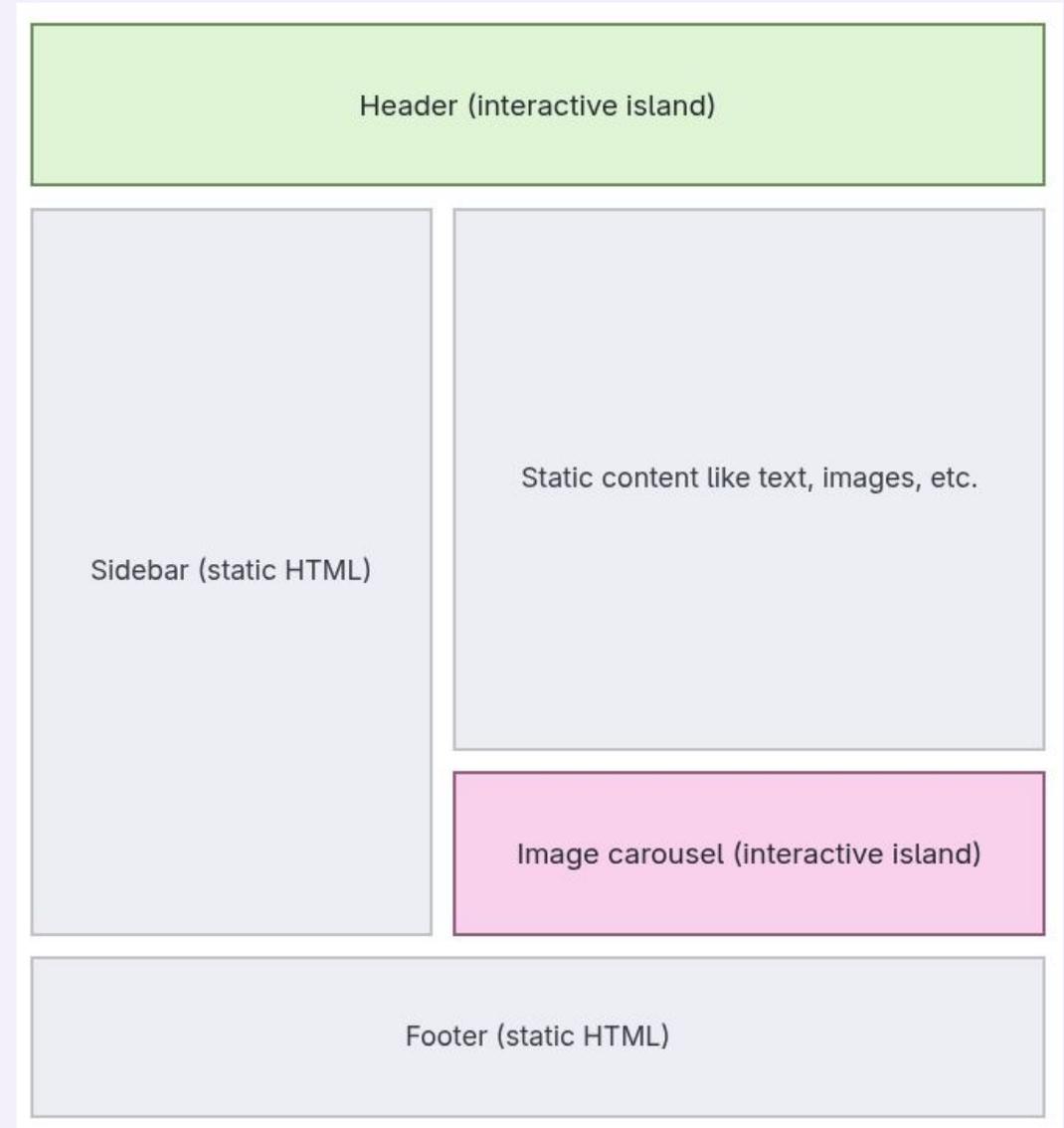
@astrojs/react



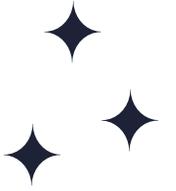
@astrojs/svelte



@astrojs/vue



Tipi di isole



Client Island

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

```
1 <!-- This component is now interactive on the page!
2   The rest of your website remains static. -->
3 <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`

```
1 ---
2 import Avatar from "../components/Avatar.astro";
3 ---
4 <Avatar server:defer />
```

Direttive client:*

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

`:load`

Carica e idrata il JS appena la pagina si è caricata

`:idle`

Carica e idrata il JS quando avviene l'evento `requestIdleCallback`

`timeout` custom

Fallback a `:load` per browser che non supportano `requestIdleCallback` (es. Safari)

`:visible`

Carica e idrata il JS quando il componente entra nella viewport

`rootMargin` per anticipare il rendering (es: `200px`)

`:media`

Carica e idrata il JS quando una CSS media query viene triggerata

es:
`client:media="(min-width: 1000px)"`

`:only`

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 e cacheate 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

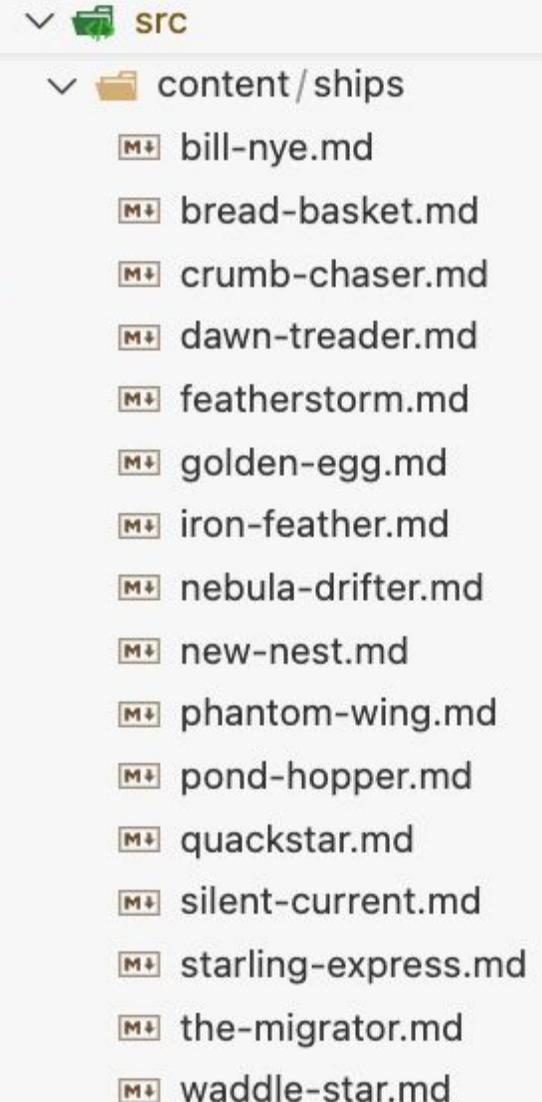
Content collections (2)

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

Content collections (2)

- APT essenziali da `astro:content`.

```
1 // src/content.config.ts
2 import { defineCollection, z } from 'astro:content';
3 import { glob } from 'astro/loaders';
4
5 const ships = defineCollection({
6   // loader automatically generates id from filename
7   loader: glob({ pattern: '**/*.md', base: './src/content/ships' }),
8   schema: z.object({
9     name: z.string(),
10    // ... other fields as needed
11  }),
12 });
13
14 // collection name is the key in this export
15 export const collections = { ships };
16
```



The screenshot shows a file explorer view of a directory structure. The root directory is 'src', which contains a subdirectory 'content/ships'. Inside 'content/ships', there are 17 markdown files, each with a small icon to its left. The files are: bill-nye.md, bread-basket.md, crumb-chaser.md, dawn-treader.md, featherstorm.md, golden-egg.md, iron-feather.md, nebula-drifter.md, new-nest.md, phantom-wing.md, pond-hopper.md, quackstar.md, silent-current.md, starling-express.md, the-migrator.md, and waddle-star.md.

Content collections (2)

```
1 ---
2 name: "QSS Quackstar"
3 class: "Explorer"
4 status: "Active"
5 crew: 142
6 commissioned: 2024-03-15
7 topSpeed: "Warp 7.5"
8 armament: ["Photon Torpedoes",
9 homePort: "Pond Zero"
10 captain: "Commander Mallard"
11 ---
12
13
14 # Quackstar title from .md \#
15 The QSS Quackstar is the pride
16 Launched from Pond Zero's orb
17 Her crew discovered three habi
18
19 The Quackstar's advanced senso
20
```

```
1 ---
2 import RootLayout from '@/layouts/RootLayout.astro';
3 import { getEntry, render } from 'astro:content';
4
5 const { id } = Astro.params;
6 if (!id) {
7   return Astro.redirect('/demos/5-content');
8 }
9
10 const ship = await getEntry('ships', id);
11
12 if (!ship) {
13   return Astro.redirect('/demos/5-content');
14 }
15
16 const { Content } = await render(ship);
17 ---
18
19 <RootLayout title={ship.data.name}>
20   {/* renders HTML generated by md content */}
21   <Content />
22 </RootLayout>
23
```

```
src
content/ships
bill-nve.md
```

le for deep space scouting missions.

```
the-migrator.md
waddle-star.md
```

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

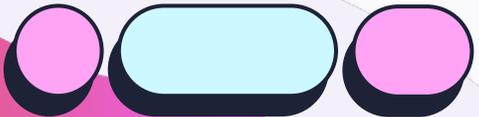
Content Layer API

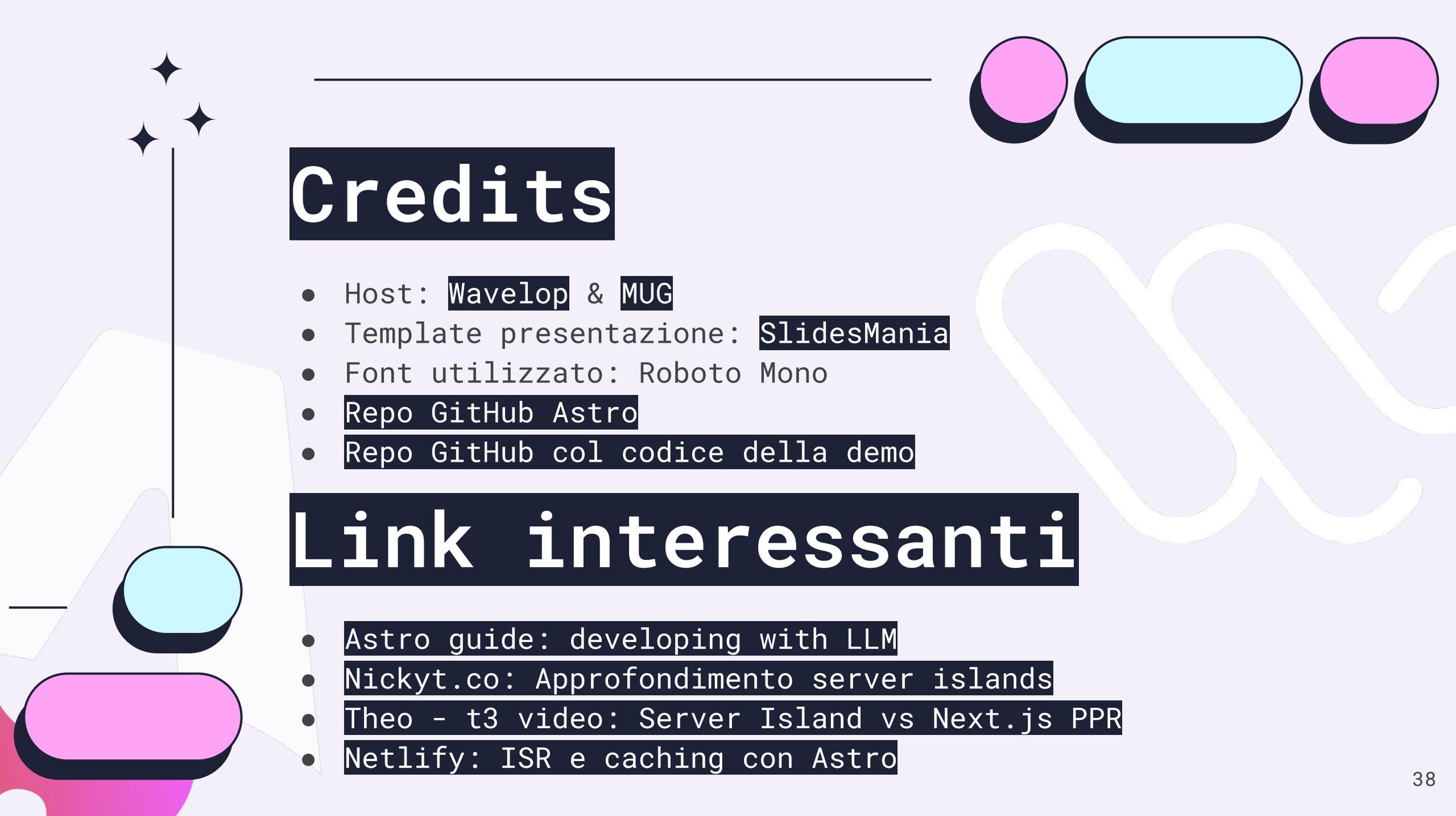
...



“

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!



Per rimanere in contatto...



Se e quando scriverò qualcosa...

THANK YOU