Introducing Astro with a real world use case: Beer Tech Group site

1# Tech and Beer, 12th April 2023

Francesca Motisi

Software Architect @ Cloudmind,
WTM Ambassador

Francesca Motisi

@francesca_mts

mts88

What’s Astro?

Astro is the all-in-one web framework designed for speed. Pull your content from anywhere and deploy everywhere, all powered by your favorite UI components and libraries.

From official Astro Doc

Official key features

Content-focused: designed for content-rich websites.

Server-first: leverages server-side rendering over client-side rendering as much as possible.

Fast by default: reduce Javascript by 90%.

Easy to use: you don’t need to be an expert to build something.

Fully-featured, but flexible: all-in-one web framework with 100+ integrations.

Multi Page Application Framework

A **Multi-Page Application** (MPA) is a website consisting of multiple HTML pages, mostly rendered on a server.

Others MPA frameworks are: Ruby on Rails, Python Django, PHP Laravel, WordPress, Joomla, Drupal.

A **Single-Page Application** (SPA) is a website consisting of a single JavaScript application that loads in the user's browser and then renders HTML locally.

Angular, and Create React App are examples of SPA frameworks.

Which is better?

None of them. It’s depends on use cases.

Astro is great for:

- marketing sites;
- documentation sites;
- blogs and portfolios;
- some ecommerce sites;

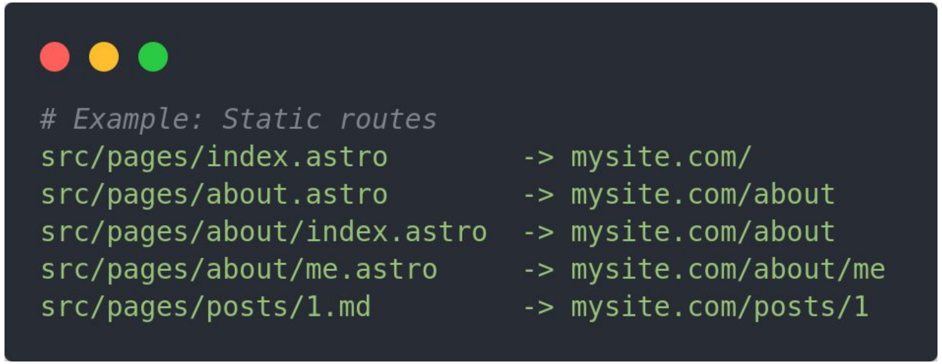
- first-load performance is essential;
- SEO;
- great accessibility;
- static site generator (SSG);

If your project not falls into these use cases, Astro might not be the right choice for your project… and that’s okay! Check out other solutions.

Routing system

Astro provides both **static** and **dynamic** route generator.

`.astro` page components as well as **Markdown** and **MDX** Files (.md, .mdx) within the `src/pages/` directory automatically become pages on your website.



```
# Example: Static routes
src/pages/index.astro        -> mysite.com/
src/pages/about.astro        -> mysite.com/about
src/pages/about/index.astro  -> mysite.com/about
src/pages/about/me.astro     -> mysite.com/about/me
src/pages/posts/1.md         -> mysite.com/posts/1
```

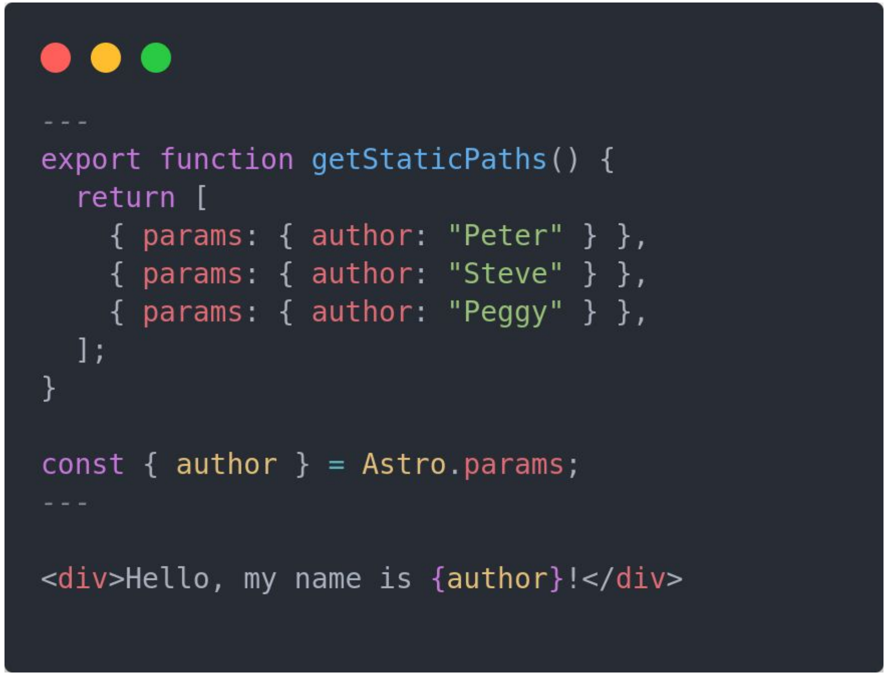
Dynamic routes (SSG)

An Astro page file can specify dynamic route parameters to generate multiple pages.

For example, `src/pages/authors/[author].astro` generates a **bio page for every author** on your blog.

This will generate three static pages:

- `/authors/peter`
- `/authors/steve`
- `/authors/peggy`

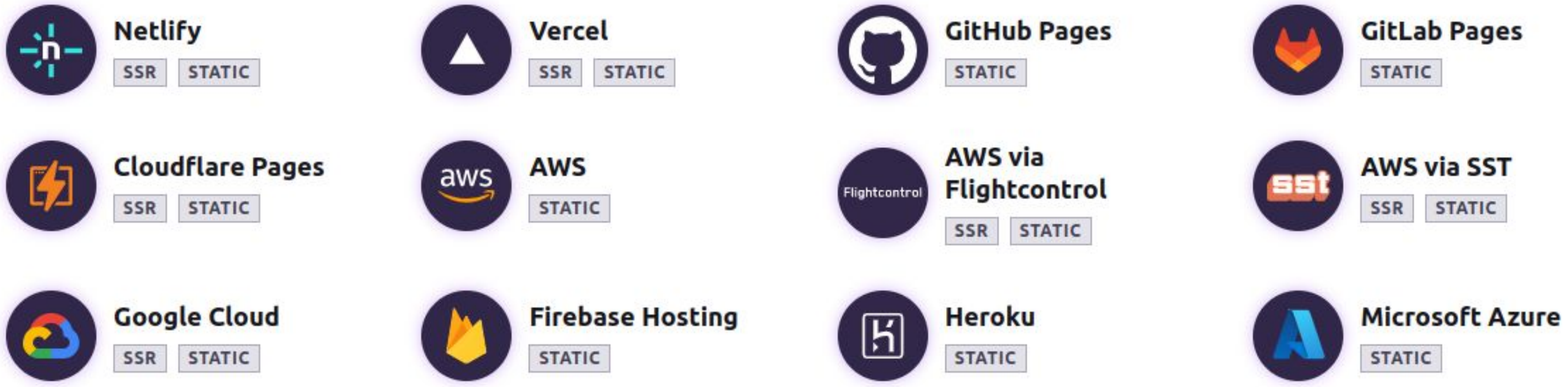


```
---
export function getStaticPaths() {
  return [
    { params: { author: "Peter" } },
    { params: { author: "Steve" } },
    { params: { author: "Peggy" } },
  ];
}

const { author } = Astro.params;
---

<div>Hello, my name is {author}!</div>
```

Content via MD, MDX or your CMS

Storyblok
Contentful
ButterCMS
Ghost
Tina CMS
WordPress
Builder.io
Crystallize
Directus
KeystoneJS
Netlify CMS
Payload CMS
Prismic
Sanity
Spinal

Deploy Astro site



- Netlify — SSR, STATIC
- Vercel — SSR, STATIC
- GitHub Pages — STATIC
- GitLab Pages — STATIC
- Cloudflare Pages — SSR, STATIC
- AWS — STATIC
- AWS via Flightcontrol — SSR, STATIC
- AWS via SST — SSR, STATIC
- Google Cloud — SSR, STATIC
- Firebase Hosting — STATIC
- Heroku — STATIC
- Microsoft Azure — STATIC

UI Frameworks supported

@astrojs/**alpinejs**

@astrojs/**lit**

@astrojs/**preact**

@astrojs/**react**

@astrojs/**solid-js**

@astrojs/**svelte**

@astrojs/**vue**

*you can mix and match different ui frameworks in same Astro project

Astro Island

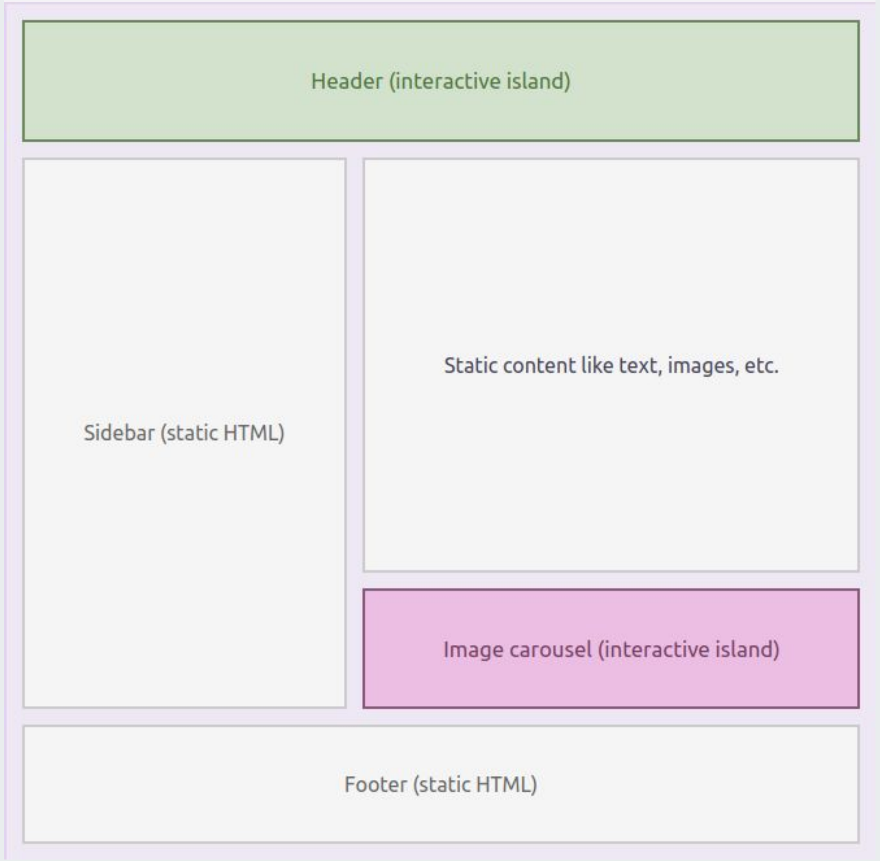
A pattern of web architecture pioneered by Astro.

Coined by Etsy’s frontend architect Katie Sylor-Miller in 2019, and expanded by Preact creator Jason Miller.

Refers to an interactive UI component on an otherwise static page of HTML.

An island always renders in isolation.

Think of them as islands in a sea of static, non-interactive HTML.


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

How islands works?

Static by default, can be hydrated using `client:*` directive.

This keeps every site fast by default by removing all unused JavaScript from the page.

The islands load in parallel and hydrate in isolation.

`client:*` directive tells how to load component. EG: interactive as soon as possible or hydrated when entered the user’s viewport.

Real world use case:
Beer Tech Group site

https://github.com/beer-tech-group/beer-tech-group.github.io

Why Astro?

A static site, as MPA

SEO

Very tiny time to market

Fast learning curve and team with different knowledges

Deploy fast and on budget

Beer Tech Stack

Strongly typed **Typescript**;

Integrations:

- **React** + **Vue**;
- **Tailwind** + Headless UI;
- **SEO**;
- Compressor;
- Image Optimization;
- SASS;

Content with **MDX** and **collections**;

Github Actions;

Firebase Hosting Deploy;

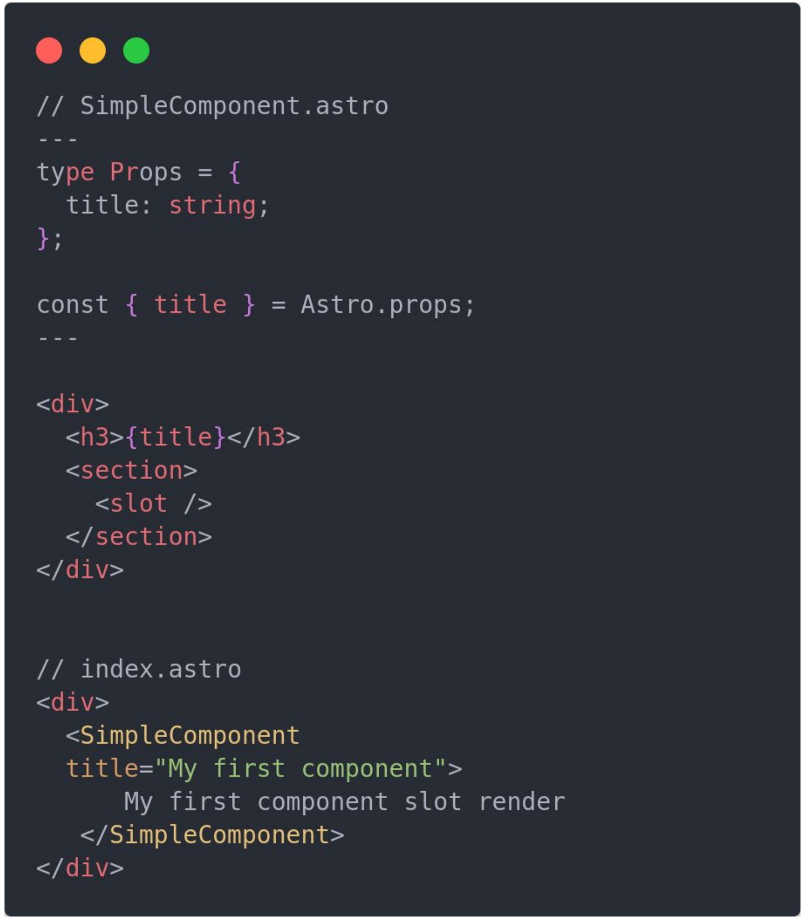
Strongly typed

By **default** Astro supports **Typescript**, even if is not used.

Provides three settings for `tsconfig.json`:

- base;
- strict;
- strictest;

We have a lots of problems so we use strictest (and is not so strictest enough).

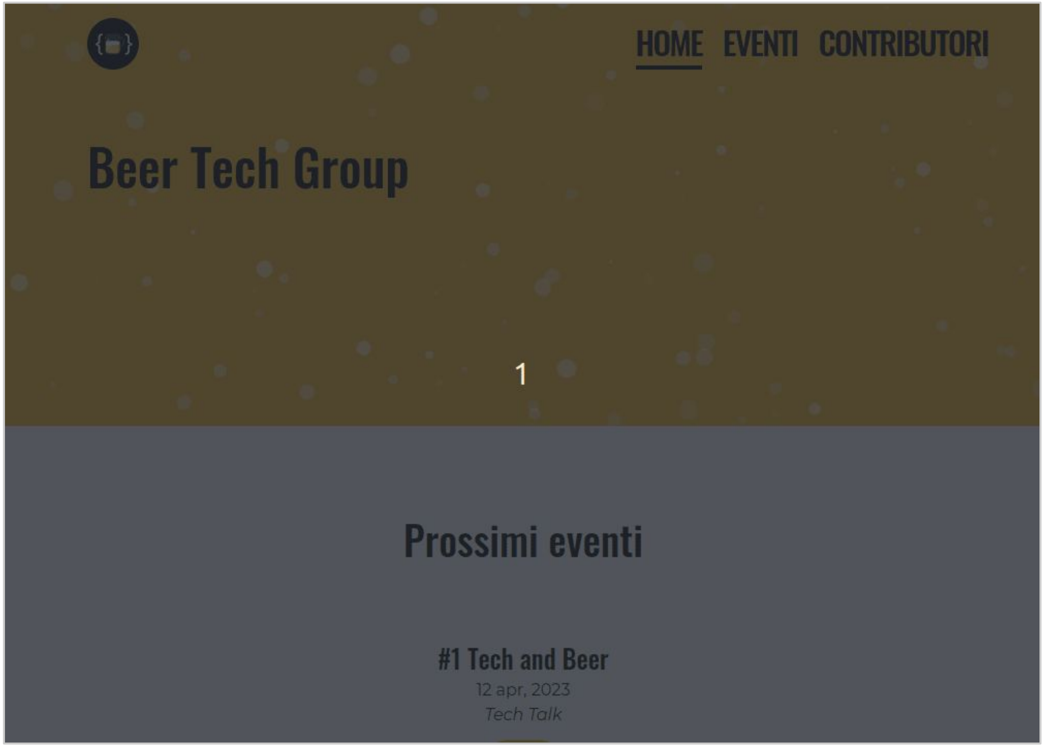


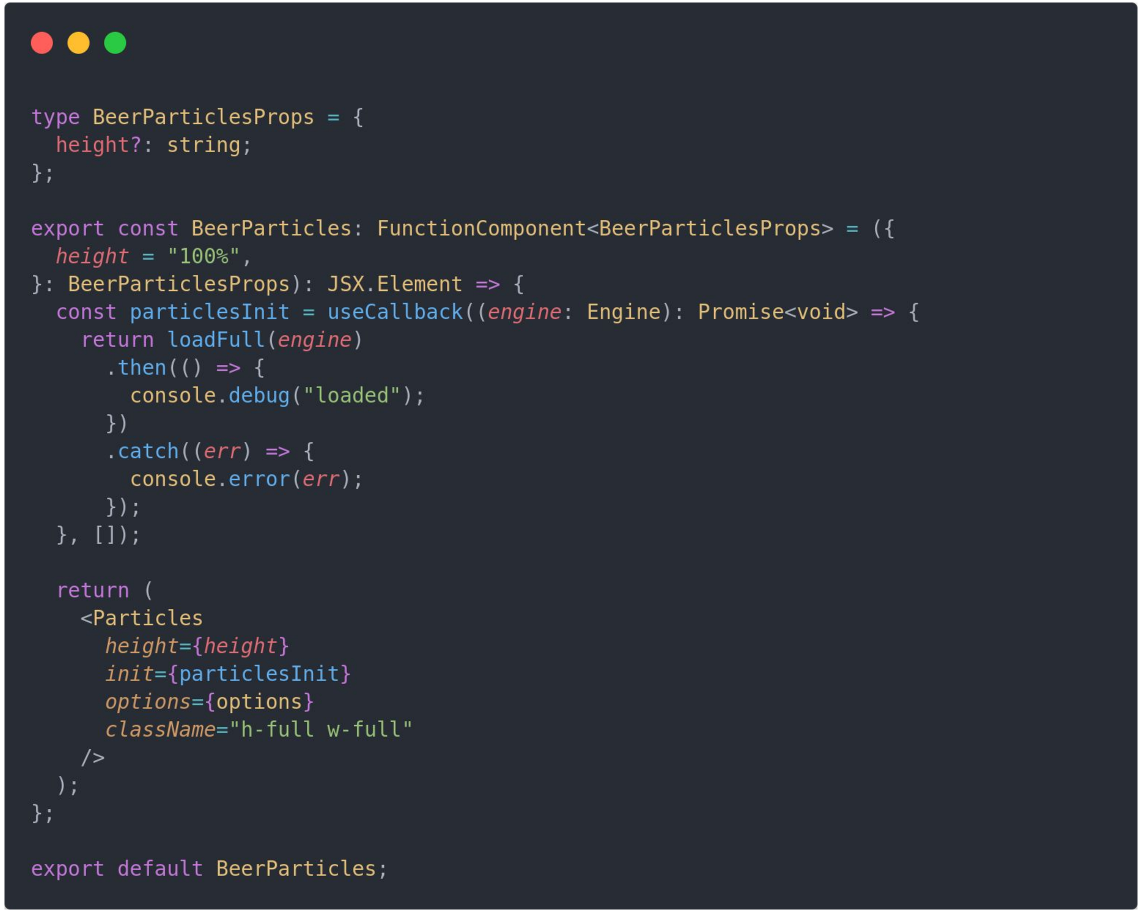
```
// SimpleComponent.astro
---
type Props = {
  title: string;
};

const { title } = Astro.props;
---

<div>
  <h3>{title}</h3>
  <section>
    <slot />
  </section>
</div>


// index.astro
<div>
  <SimpleComponent
  title="My first component">
      My first component slot render
   </SimpleComponent>
</div>
```

React Component

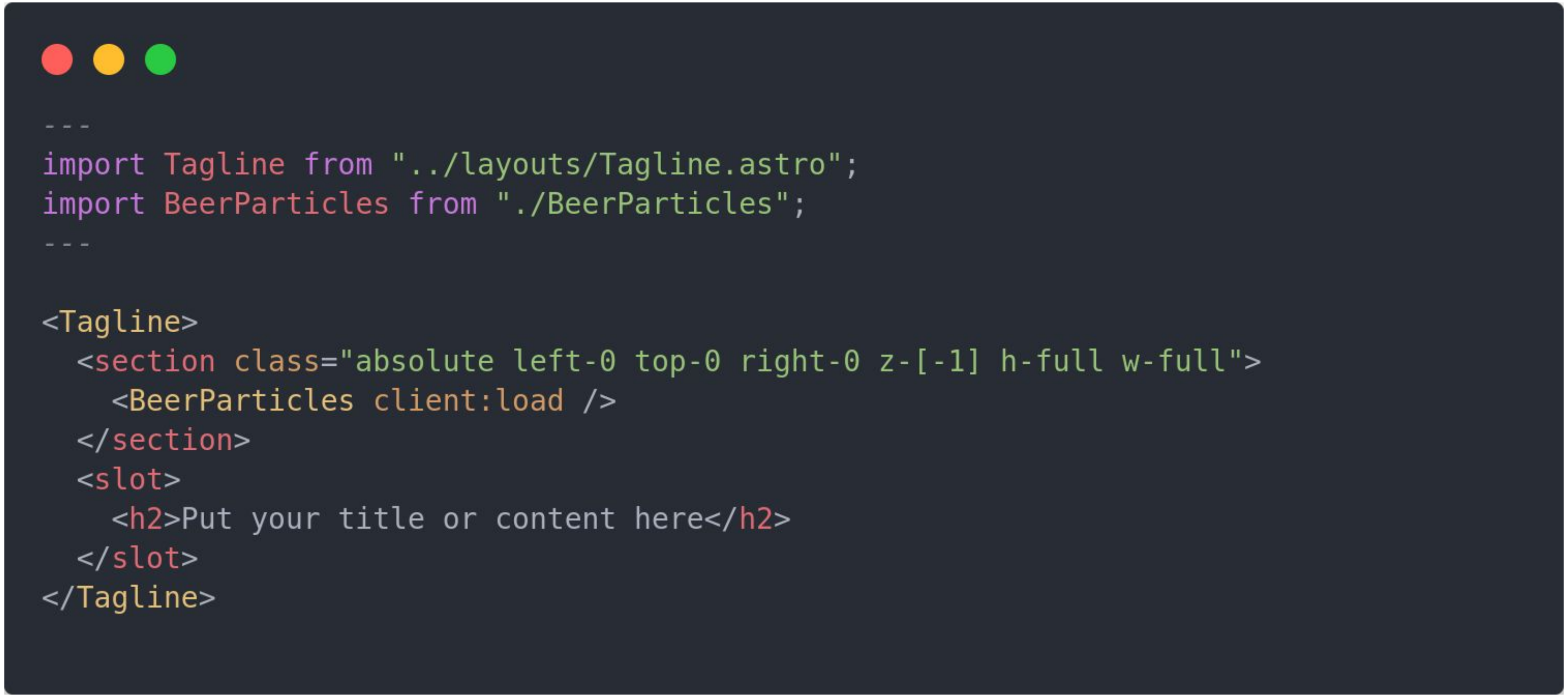
`BeerParticles` is a **react component** and render in a **Astro Island**.

It’s a stateful component.





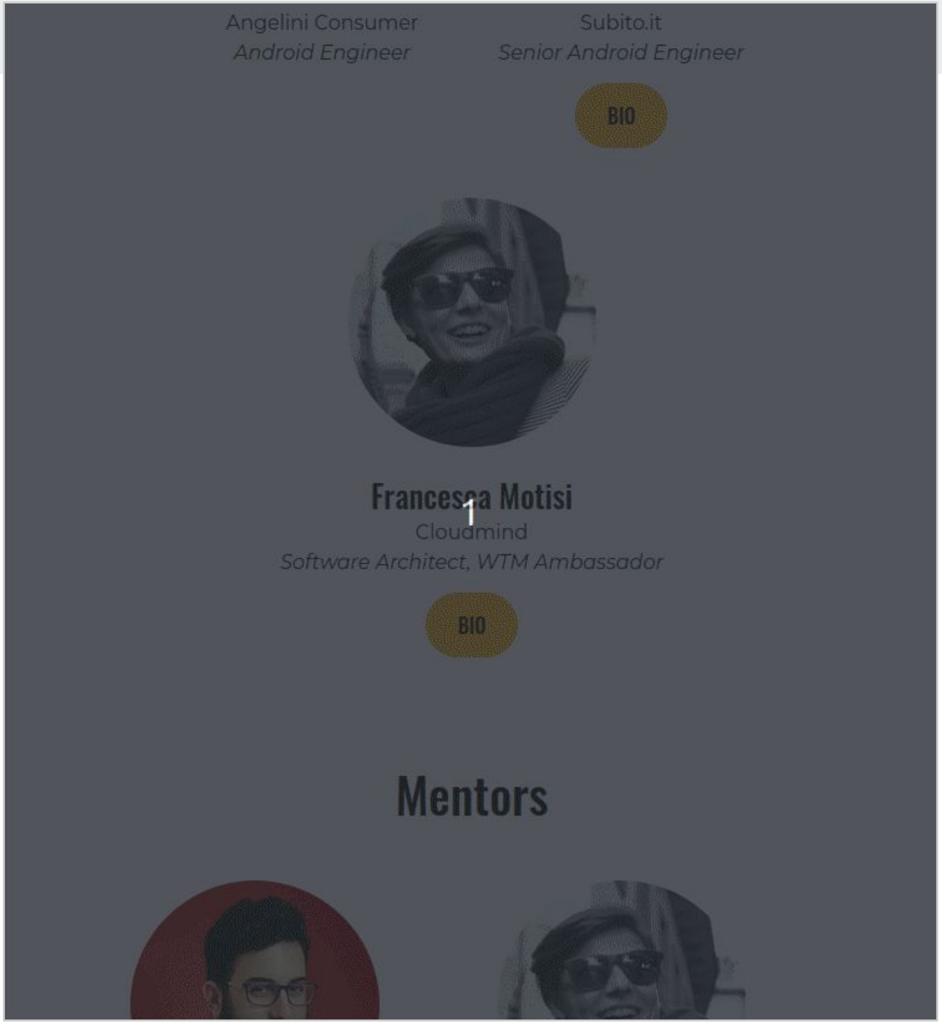
```
type BeerParticlesProps = {
  height?: string;
};

export const BeerParticles: FunctionComponent<BeerParticlesProps> = ({
  height = "100%",
}: BeerParticlesProps): JSX.Element => {
  const particlesInit = useCallback((engine: Engine): Promise<void> => {
    return loadFull(engine)
      .then(() => {
        console.debug("loaded");
      })
      .catch((err) => {
        console.error(err);
      });
  }, []);

  return (
    <Particles
      height={height}
      init={particlesInit}
      options={options}
      className="h-full w-full"
    />
  );
};

export default BeerParticles;
```


```
---
import Tagline from "../layouts/Tagline.astro";
import BeerParticles from "./BeerParticles";
---

<Tagline>
  <section class="absolute left-0 top-0 right-0 z-[-1] h-full w-full">
    <BeerParticles client:load />
  </section>
  <slot>
    <h2>Put your title or content here</h2>
  </slot>
</Tagline>
```

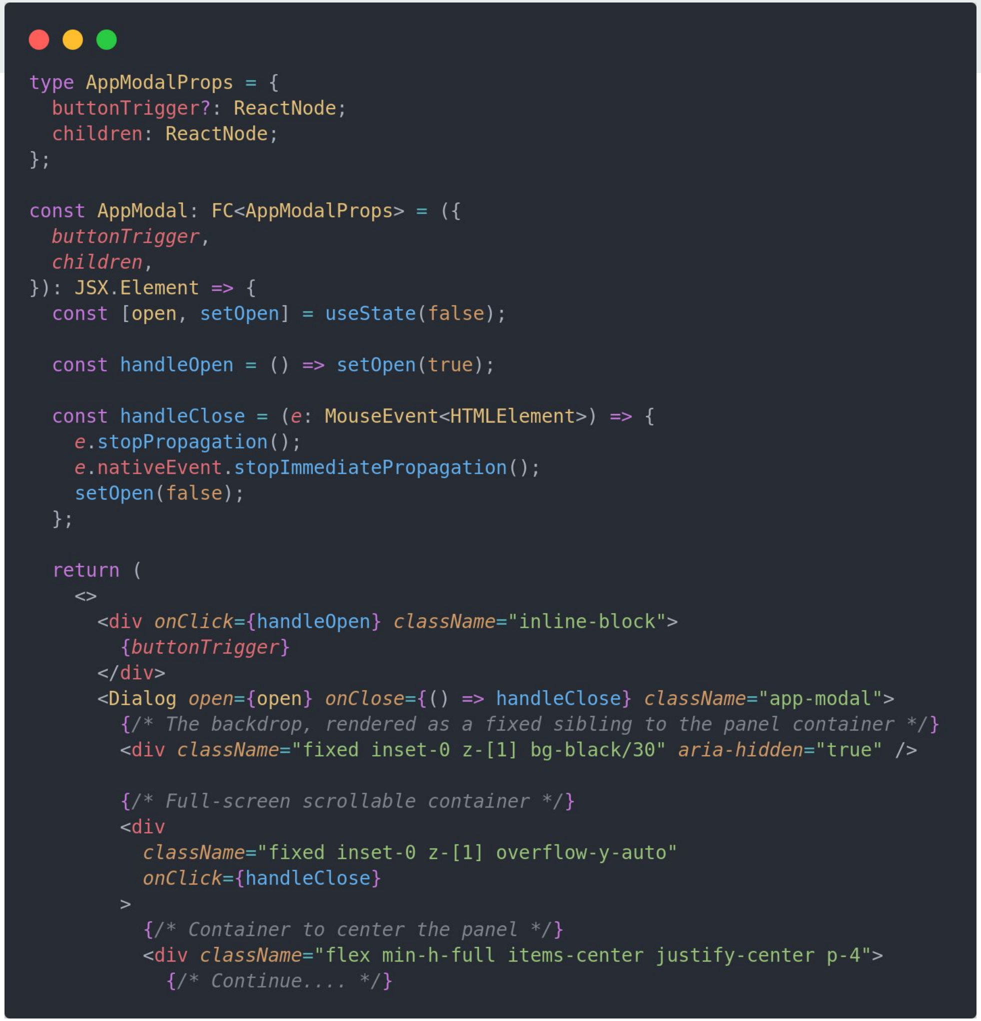
And modals?

Also modals are react component, but not so stateful as BeerParticles.

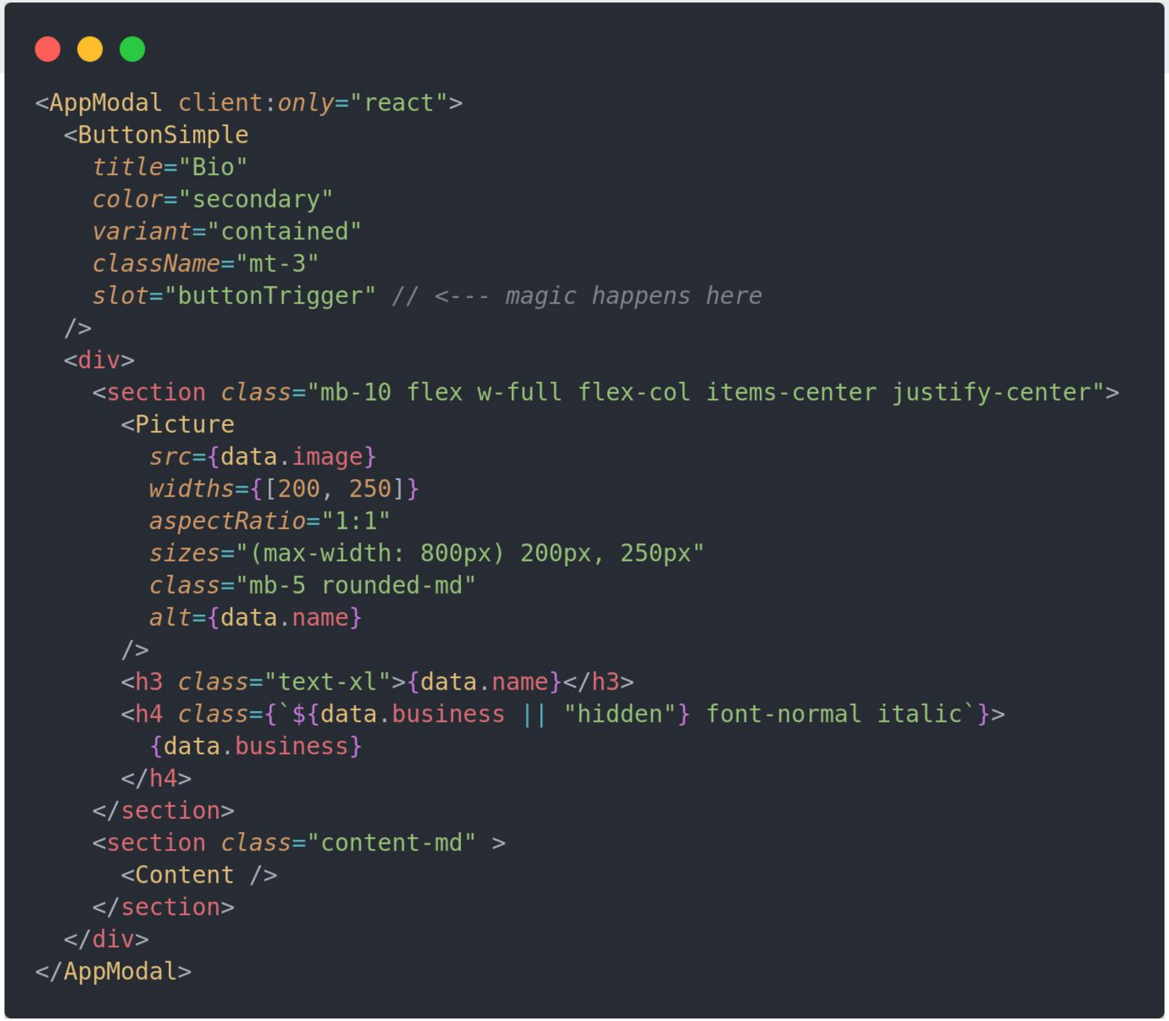
It expects **children** and a **ReactNode component** from Astro or other parents.


Angelini Consumer
Android Engineer
Subito.it
Senior Android Engineer

Francesca Motisi
Cloudmind
Software Architect, WTM Ambassador



```tsx
type AppModalProps = {
  buttonTrigger?: ReactNode;
  children: ReactNode;
};

const AppModal: FC<AppModalProps> = ({
  buttonTrigger,
  children,
}): JSX.Element => {
  const [open, setOpen] = useState(false);

  const handleOpen = () => setOpen(true);

  const handleClose = (e: MouseEvent<HTMLElement>) => {
    e.stopPropagation();
    e.nativeEvent.stopImmediatePropagation();
    setOpen(false);
  };

  return (
    <>
      <div onClick={handleOpen} className="inline-block">
        {buttonTrigger}
      </div>
      <Dialog open={open} onClose={() => handleClose} className="app-modal">
        {/* The backdrop, rendered as a fixed sibling to the panel container */}
        <div className="fixed inset-0 z-[1] bg-black/30" aria-hidden="true" />

        {/* Full-screen scrollable container */}
        <div
          className="fixed inset-0 z-[1] overflow-y-auto"
          onClick={handleClose}
        >
          {/* Container to center the panel */}
          <div className="flex min-h-full items-center justify-center p-4">
            {/* Continue.... */}
```



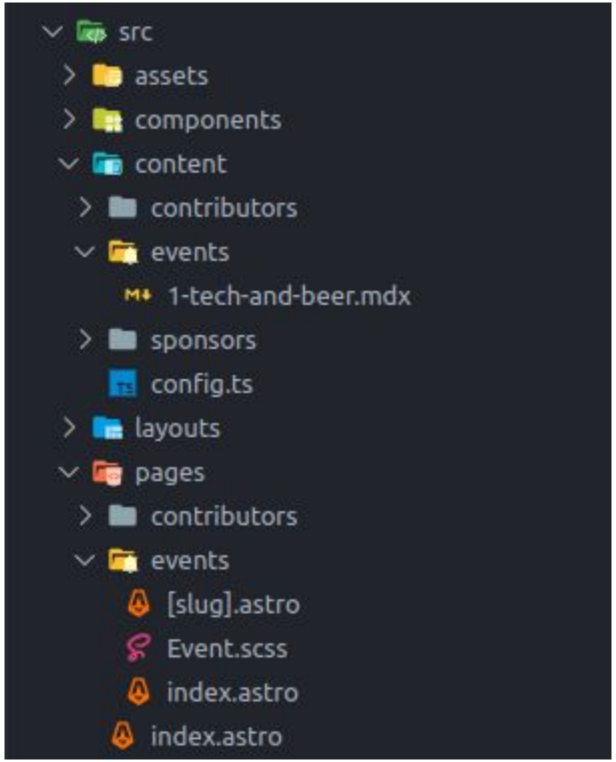
```
<AppModal client:only="react">
  <ButtonSimple
    title="Bio"
    color="secondary"
    variant="contained"
    className="mt-3"
    slot="buttonTrigger" // <--- magic happens here
  />
  <div>
    <section class="mb-10 flex w-full flex-col items-center justify-center">
      <Picture
        src={data.image}
        widths={[200, 250]}
        aspectRatio="1:1"
        sizes="(max-width: 800px) 200px, 250px"
        class="mb-5 rounded-md"
        alt={data.name}
      />
      <h3 class="text-xl">{data.name}</h3>
      <h4 class={`${data.business || "hidden"} font-normal italic`}>
        {data.business}
      </h4>
    </section>
    <section class="content-md" >
      <Content />
    </section>
  </div>
</AppModal>
```

Content using collections

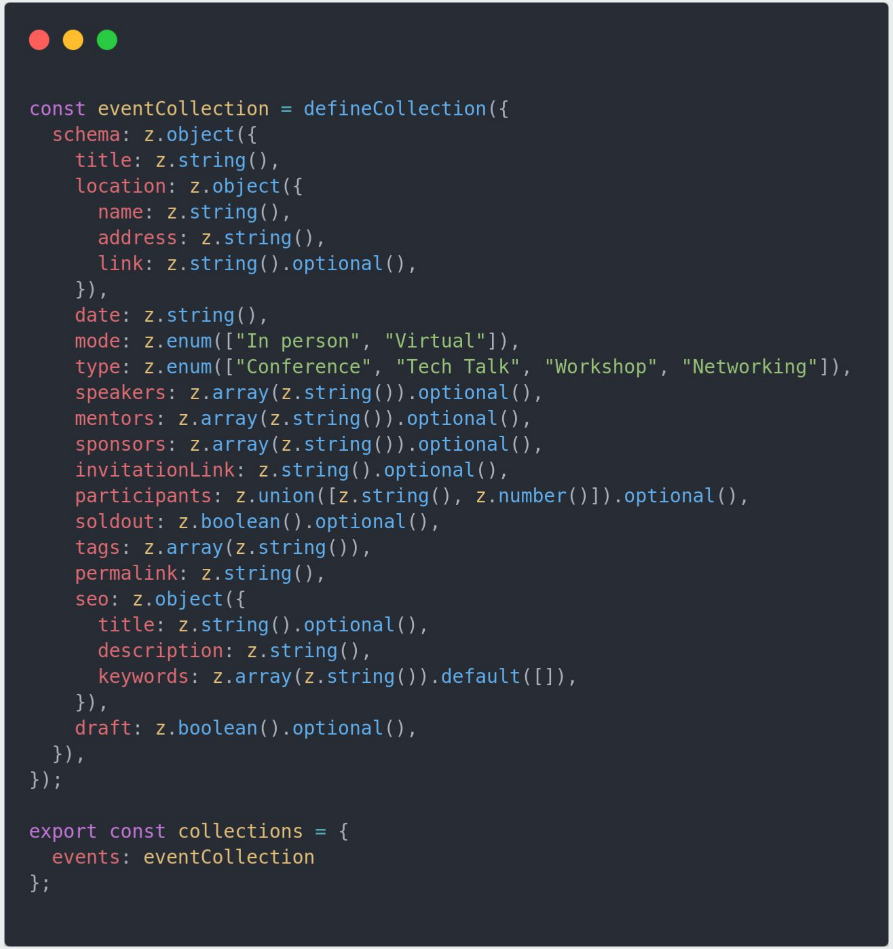
Content collections are a feature of Astro that help manage your content files in a project.

Helps to organize your **content**, **validate** your frontmatter, and **provide automatic TypeScript type-safety** for all of your content.

Uses slug for pretty URL and to query the entry directly from your collection.



```
src
  assets
  components
  content
    contributors
    events
      1-tech-and-beer.mdx
    sponsors
    config.ts
  layouts
  pages
    contributors
    events
      [slug].astro
      Event.scss
      index.astro
    index.astro
```

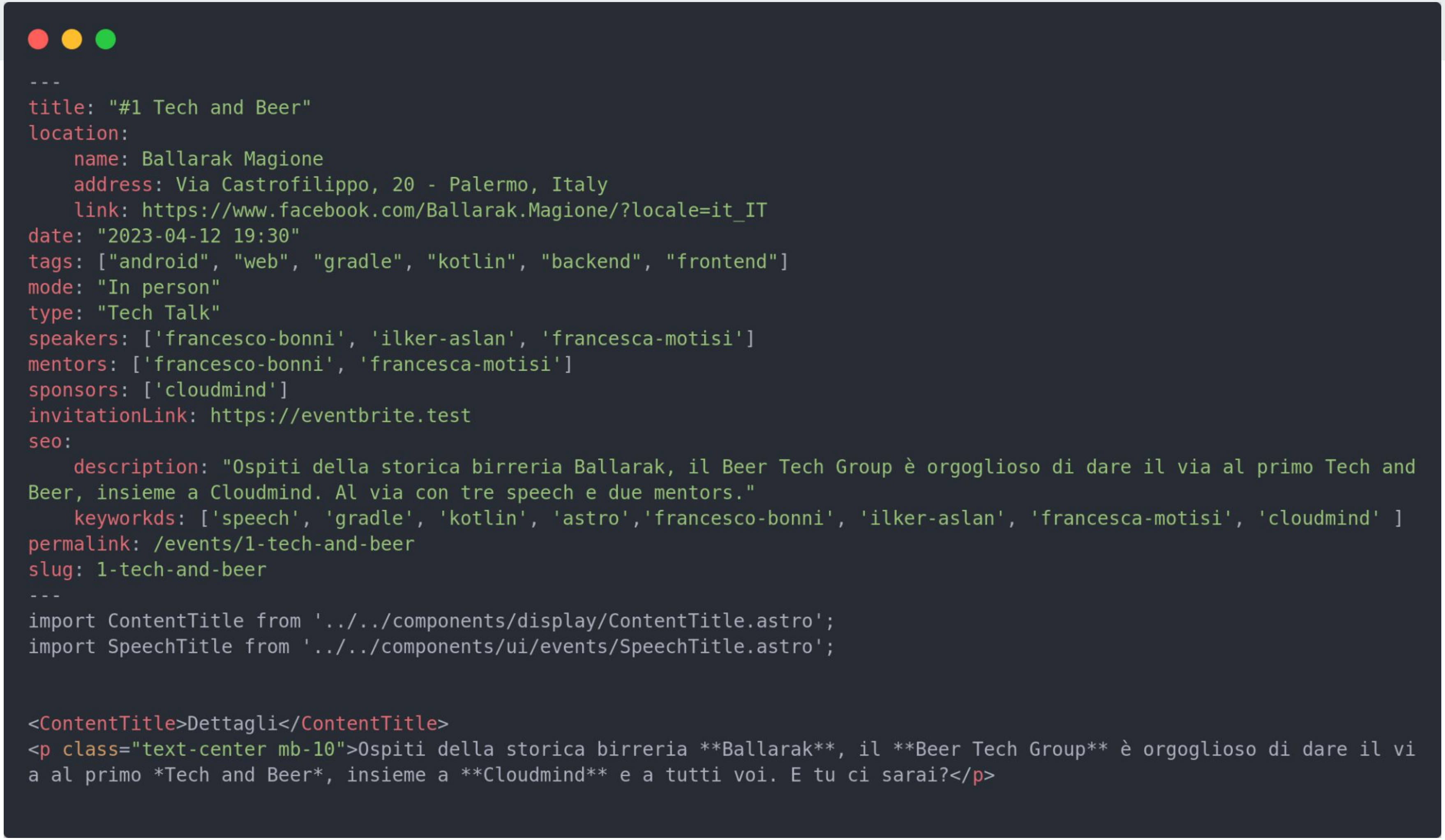


```
const eventCollection = defineCollection({
  schema: z.object({
    title: z.string(),
    location: z.object({
      name: z.string(),
      address: z.string(),
      link: z.string().optional(),
    }),
    date: z.string(),
    mode: z.enum(["In person", "Virtual"]),
    type: z.enum(["Conference", "Tech Talk", "Workshop", "Networking"]),
    speakers: z.array(z.string()).optional(),
    mentors: z.array(z.string()).optional(),
    sponsors: z.array(z.string()).optional(),
    invitationLink: z.string().optional(),
    participants: z.union([z.string(), z.number()]).optional(),
    soldout: z.boolean().optional(),
    tags: z.array(z.string()),
    permalink: z.string(),
    seo: z.object({
      title: z.string().optional(),
      description: z.string(),
      keywords: z.array(z.string()).default([]),
    }),
    draft: z.boolean().optional(),
  }),
});

export const collections = {
  events: eventCollection
};
```

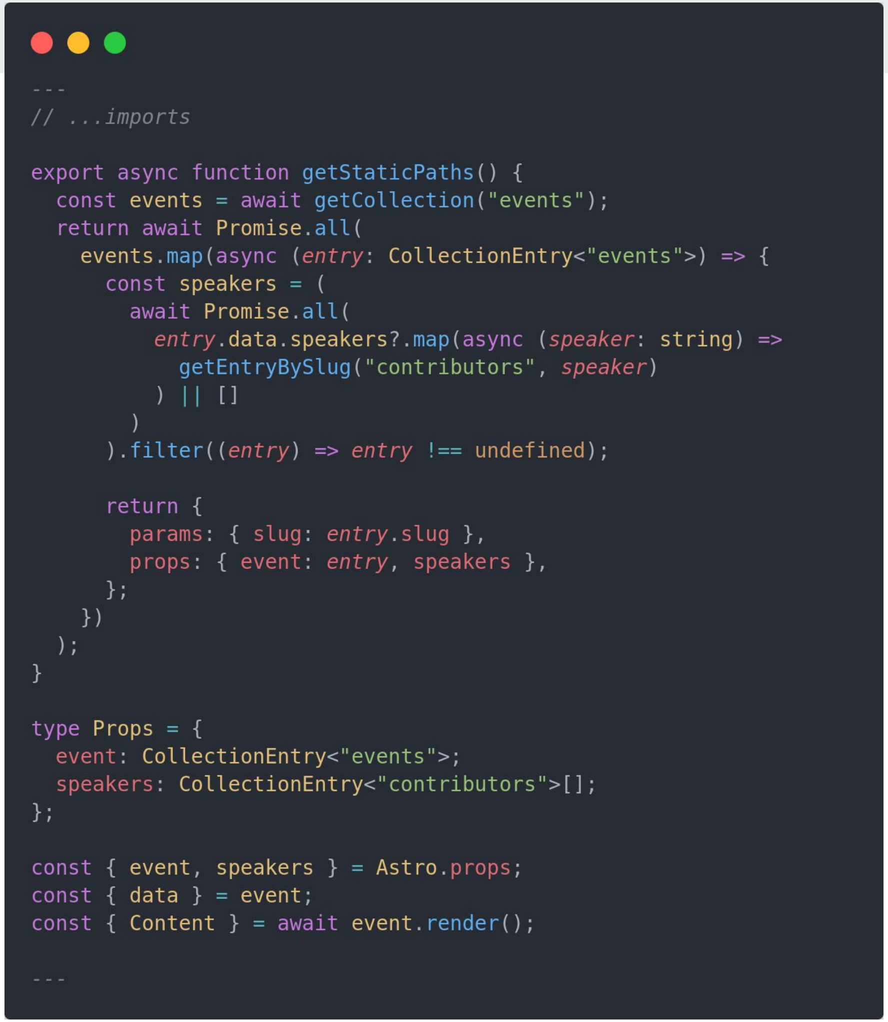
Schema definition

Enforces consistent frontmatter and **guarantees** a predictable form.

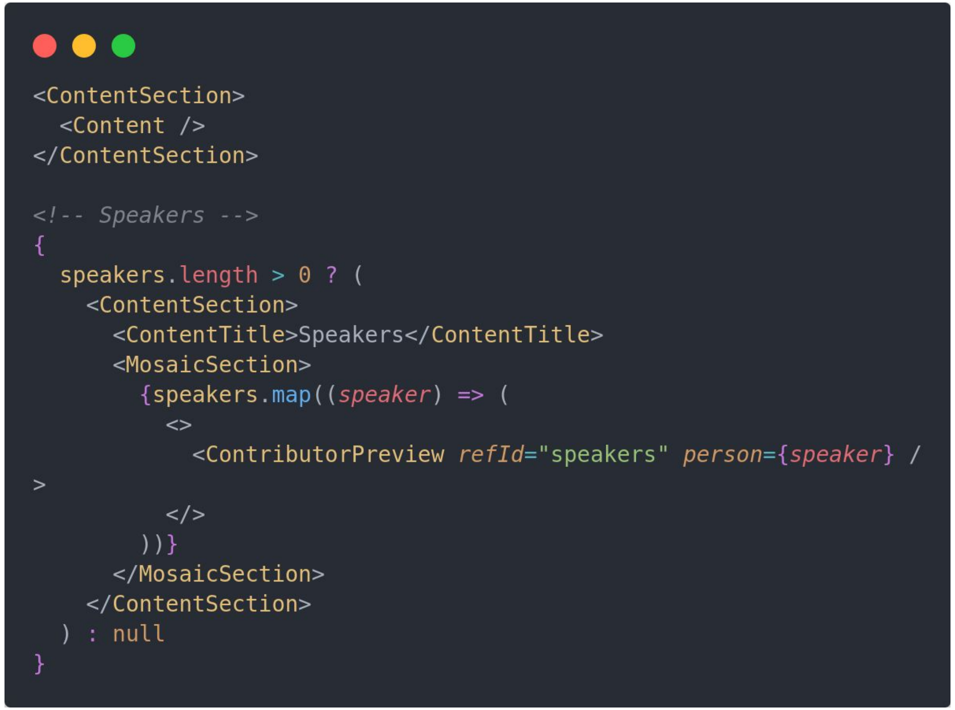
Astro will automatically **generate** and apply a TypeScript **interface** to it.



```
---
title: "#1 Tech and Beer"
location:
    name: Ballarak Magione
    address: Via Castrofilippo, 20 - Palermo, Italy
    link: https://www.facebook.com/Ballarak.Magione/?locale=it_IT
date: "2023-04-12 19:30"
tags: ["android", "web", "gradle", "kotlin", "backend", "frontend"]
mode: "In person"
type: "Tech Talk"
speakers: ['francesco-bonni', 'ilker-aslan', 'francesca-motisi']
mentors: ['francesco-bonni', 'francesca-motisi']
sponsors: ['cloudmind']
invitationLink: https://eventbrite.test
seo:
    description: "Ospiti della storica birreria Ballarak, il Beer Tech Group è orgoglioso di dare il via al primo Tech and
Beer, insieme a Cloudmind. Al via con tre speech e due mentors."
    keyworkds: ['speech', 'gradle', 'kotlin', 'astro','francesco-bonni', 'ilker-aslan', 'francesca-motisi', 'cloudmind' ]
permalink: /events/1-tech-and-beer
slug: 1-tech-and-beer
---
import ContentTitle from '../../components/display/ContentTitle.astro';
import SpeechTitle from '../../components/ui/events/SpeechTitle.astro';


<ContentTitle>Dettagli</ContentTitle>
<p class="text-center mb-10">Ospiti della storica birreria **Ballarak**, il **Beer Tech Group** è orgoglioso di dare il vi
a al primo *Tech and Beer*, insieme a **Cloudmind** e a tutti voi. E tu ci sarai?</p>
```



```
---
// ...imports

export async function getStaticPaths() {
  const events = await getCollection("events");
  return await Promise.all(
    events.map(async (entry: CollectionEntry<"events">) => {
      const speakers = (
        await Promise.all(
          entry.data.speakers?.map(async (speaker: string) =>
            getEntryBySlug("contributors", speaker)
          ) || []
        )
      ).filter((entry) => entry !== undefined);

      return {
        params: { slug: entry.slug },
        props: { event: entry, speakers },
      };
    })
  );
}

type Props = {
  event: CollectionEntry<"events">;
  speakers: CollectionEntry<"contributors">[];
};

const { event, speakers } = Astro.props;
const { data } = event;
const { Content } = await event.render();

---
```

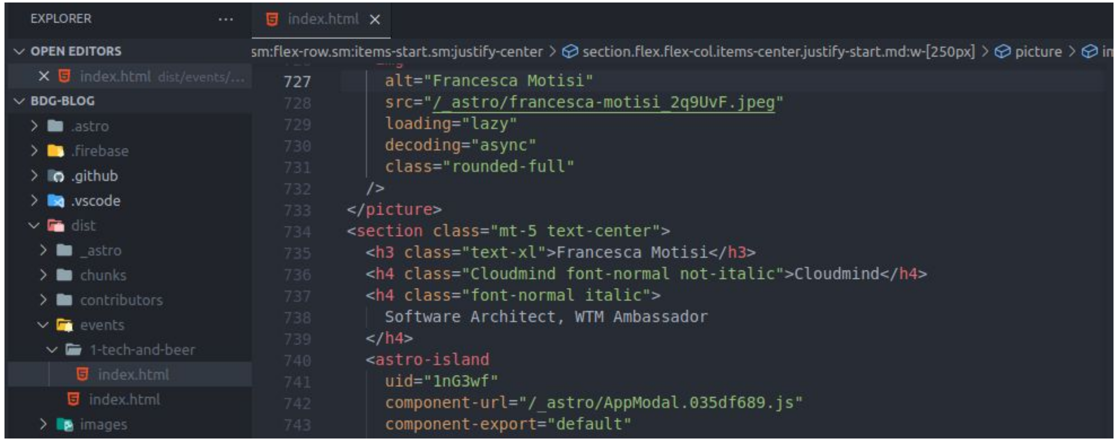


```
<ContentSection>
  <Content />
</ContentSection>

<!-- Speakers -->
{
  speakers.length > 0 ? (
    <ContentSection>
      <ContentTitle>Speakers</ContentTitle>
      <MosaicSection>
        {speakers.map((speaker) => (
          <>
            <ContributorPreview refId="speakers" person={speaker} /
>
          </>
        ))}
      </MosaicSection>
    </ContentSection>
  ) : null
}
```

Generate content with:

`npm run build`

Will create a static HTML page with all data from my collection.


EXPLORER
OPEN EDITORS
index.html dist/events/...
BDG-BLOG
.astro
.firebase
.github
.vscode
dist
_astro
chunks
contributors
events
1-tech-and-beer
index.html
index.html
images
index.html
sm:flex-row.sm:items-start.sm:justify-center > section.flex.flex-col.items-center.justify-start.md:w-[250px] > picture > im

```html
      alt="Francesca Motisi"
      src="/_astro/francesca-motisi_2q9UvF.jpeg"
      loading="lazy"
      decoding="async"
      class="rounded-full"
    />
  </picture>
  <section class="mt-5 text-center">
    <h3 class="text-xl">Francesca Motisi</h3>
    <h4 class="Cloudmind font-normal not-italic">Cloudmind</h4>
    <h4 class="font-normal italic">
      Software Architect, WTM Ambassador
    </h4>
    <astro-island
      uid="1nG3wf"
      component-url="/_astro/AppModal.035df689.js"
      component-export="default"
```

Introducing Astro with a real word use case: Beer Tech Group site

Francesca Motisi

My personal expercience with this new web framework **Astro** with a real world use case: Beer Tech Group site.

Looking deep how this framework works, pros, cons and code by example to see Astro's features and potential.

[Italiano]

Speakers

Ilker Aslan

Angelini Consumer

Android Engineer

Francesco Bonnì

Subito.it

Senior Android Engineer

BIO

Francesca Motisi

Cloudmind

Software Architect, WTM Ambassador

BIO

My two cents

Very fast learning curve

Easy to use

Growing community and frequent updates

Not mature enough for big projects

Very fun. *It's my new crush!*

- **Astro Official Doc**: https://docs.astro.build/en/getting-started
- **Astro Learning Tutorial**: https://docs.astro.build/en/tutorial/0-introduction
- **Beer Tech Group Github:** https://github.com/beer-tech-group/beer-tech-group.github.io
- **Astro Performance Report**: https://astro.build/blog/2023-web-framework-performance-report
- **Fabio Biondi**: https://www.youtube.com/watch?v=DgbBEb_ISH8&ab_channel=TomorrowDevs

Thank you

That's all folks!

[Francesca Motisi]

[@francesca_mts]

[mts88]