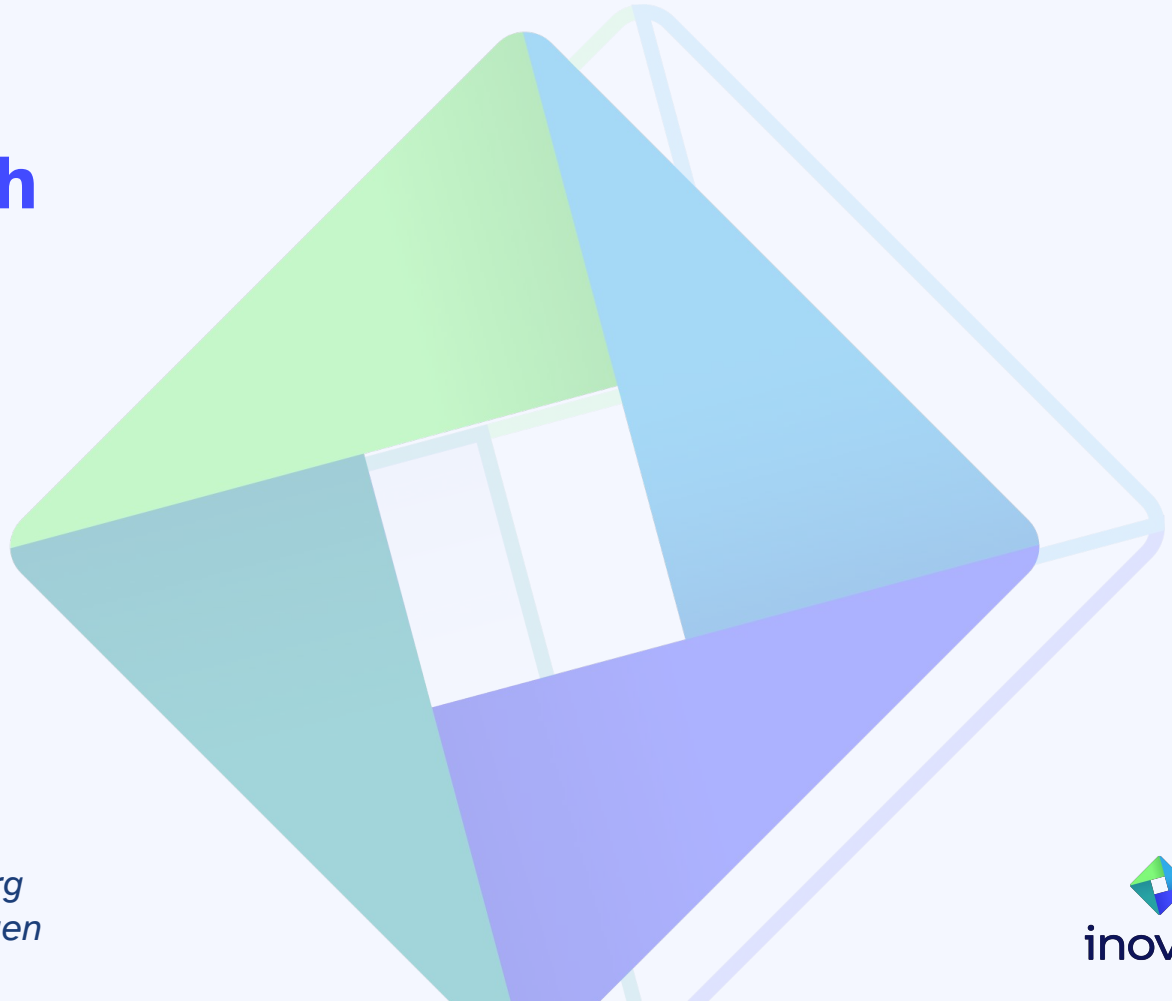
Multi Page Applications with Astro



inovex Meetup
Karlsruhe
2023-07-06
Bernd Kaiser

Team inovex
Karlsruhe · Köln · München · Hamburg
Berlin · Stuttgart · Pforzheim · Erlangen

inovex

Bernd Kaiser

Bernd Kaiser

@meldron

kaiser.land

Developer at inovex Erlangen

Focus:

- Web
- Security

🎸🎢🏥🧾🏦🃏📺

JSCC Organizer

inovex

Astro

“all-in-one web framework for building fast, content-focused websites”

inovex

About Astro

- MIT License
- 31.7k Github ⭐
- 536 Contributors
- 87k weekly downloads

- TypeScript
- Vite
- pnpm mono repo

- V2
- Astro Technology Company
- VC funded

Source: Github

inovex

Why Astro?

- Content-focused
- Server-first
- Fast by default
- Easy to use
- Fully-featured, but flexible

Source: Why Astro

Content Focused

Astro **was designed for content-rich websites** like:

- Marketing Sites
- Publishing Sites
- Documentation
- Blogs
- Portfolios

Image Source: Ula Kuźma

Content Focused

Astro is **not** for applications

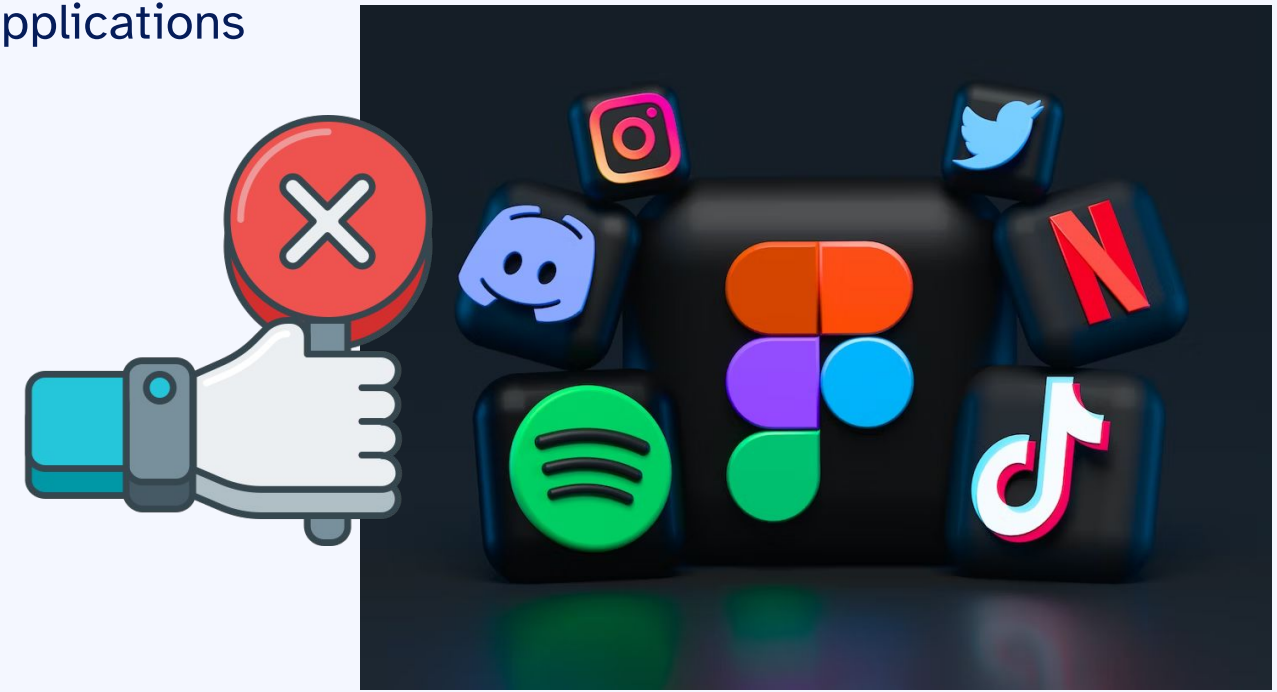


Image Sources: Alexander Shatov & Flaticon.com

inovex

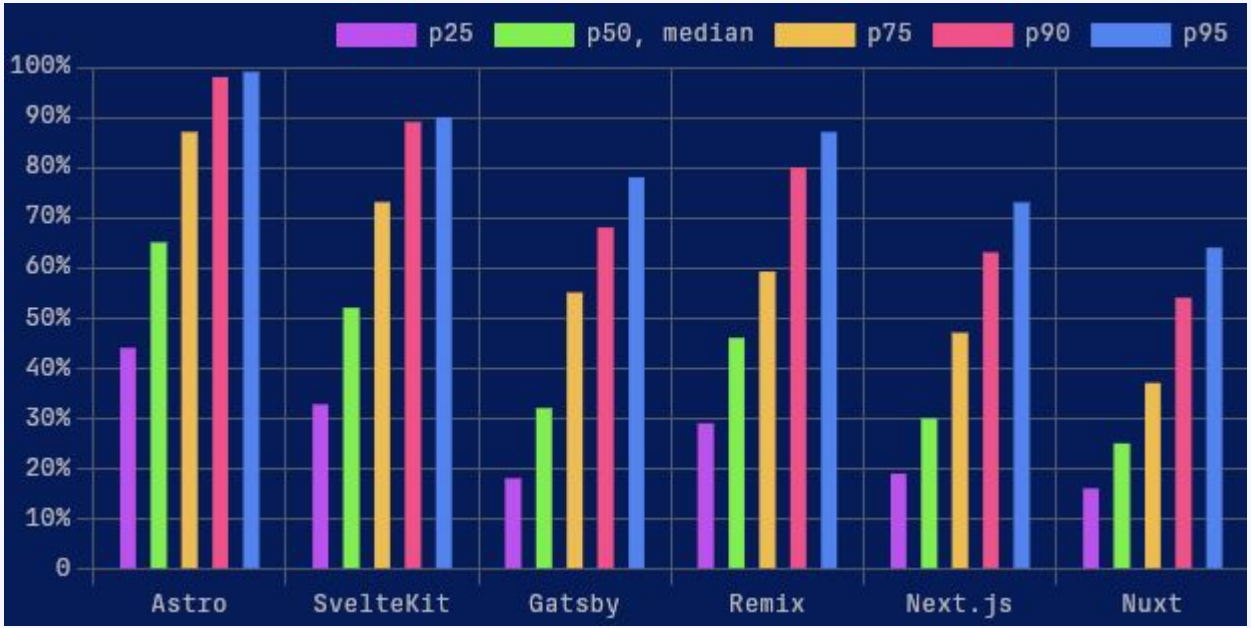
Server-first

- JavaScript as its server language and runtime
- Leverages server-side rendering
- Multi Page Apps (MPA)
- Static Site Generator (SSG)
- Server-Side Rendering (SSR)

inovex

Fast by Default

“It should be nearly impossible to build a slow website with Astro”



Lighthouse Performance Score, Breakdown

inovex

Easy to use

“Astro’s goal is to be accessible to every web developer”

- “simple” .astro UI Language
- opt-in to complexity with Astro Islands 🏝️
  - React
  - Preact
  - Svelte
  - Vue
  - Solid
  - Lit

 Image Source: Pablo Arroyo

inovex

Fully-featured, but flexible

100+ integrations

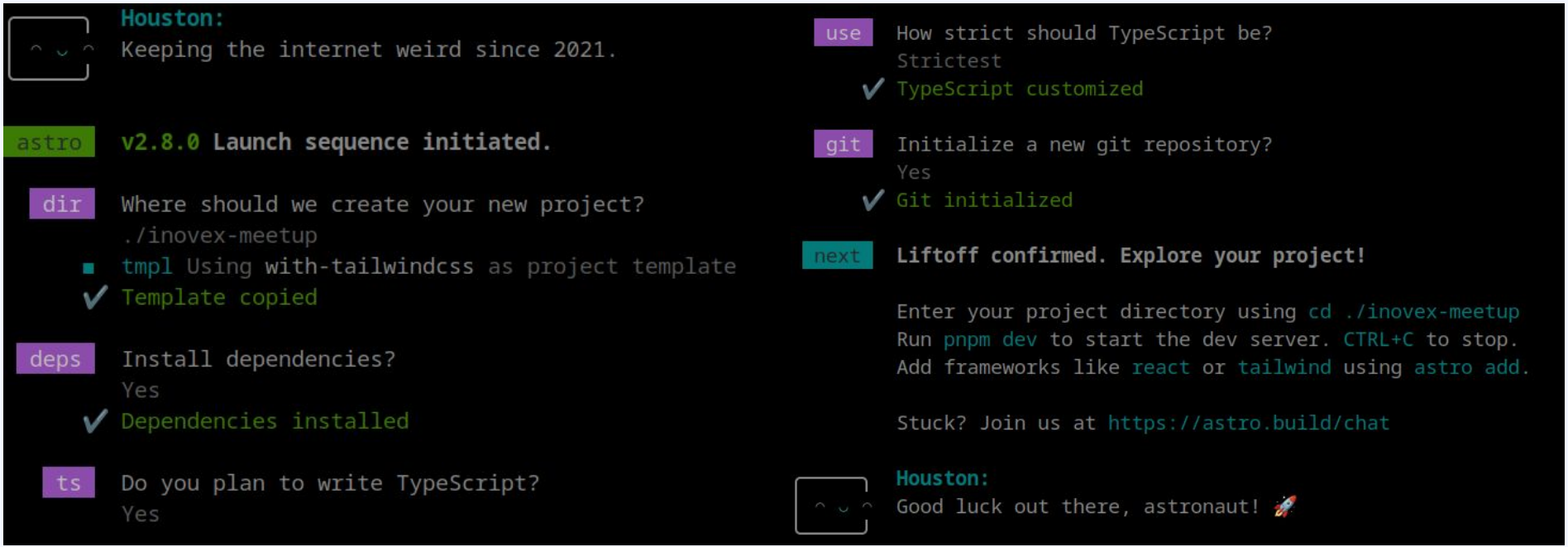
- Deployment Adapters
- CMS Connectors
- Image Optimization
- MDX
- Tailwind
- Sitemaps

Image Source: Clay Banks

inovex

Project Setup

```
pnpm create astro@latest --template with-tailwindcss
```



inovex

Project Structure

inovex

Astro Component Template

- Reusable & Composable
- HTML-only templating with no client-side runtime
- .astro file extension

Image Source: Ross Sneddon

inovex

.astro Component

```
---
import Banner from '../components/Banner.astro';
import SolidJSComponent from '../components/SolidJSComponent.tsx';

const { title } = Astro.props;
const myFavoriteFood = [/* ... */];
---
<style is:global>body { color: white; }</style> // global styles
<style>h1 { color: red; }</style> // Scoped by default

<Banner />
<h1>Hello, world!</h1>

<slot name="before-title"/>
<p>{title}</p>
<slot name="after-title"/>

<SolidJSComponent client:visible />

<ul>
  {myFavoriteFood.map((data) =>
    <li class:list={[mt-5, { font-bold: data.isSuper }]}>{data.name}</li>
  )}
</ul>
```


inovex

Pages

- located in src/pages
- Supported page files
  - .astro
  - .md
  - .mdx
  - .html
  - .ts / .js endpoints
- File-based routing

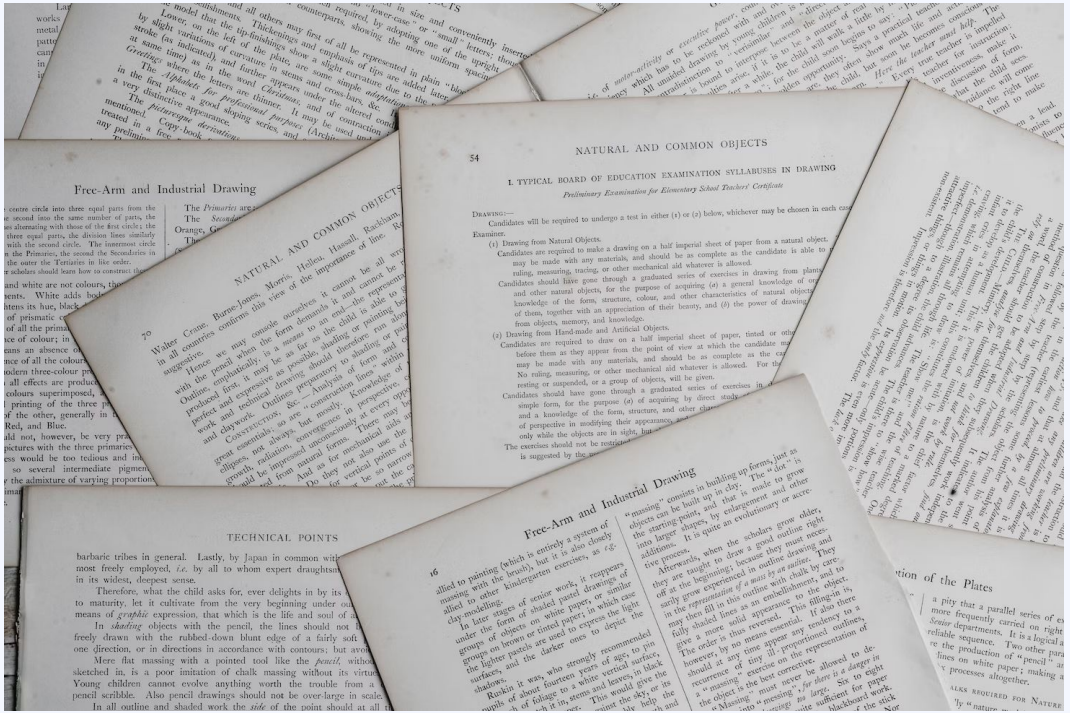


 Image Source: [Annie Spratt]

inovex

Routing

- Navigate with <a> elements
- Static Routes
- Dynamic routes
  - SSG (getStaticPaths())
  - SSR
- Built-in Pagination

Image Source: [Anne Rosly]

inovex

Static Routes

All .astro, .md, ... files in src/pages/ automatically become pages:

```
# 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
```

No separate routing config

Dynamic Routes

[parameters] can be used to generate multiple pages

Example:

```
src/pages/dogs/[dog].astro
```

Different requirements for **SSG** and **SSR** mode

Dynamic Routes - SSG Mode

- All routes must be determined at build time
- getStaticPaths() returns objects with a params property
- Each object will generate a page
- params are encoded in the URL, therefore limited to strings

[...dog].astro:

```
---
export function getStaticPaths() {
  return [
    {params: {dog: 'clifford'}},
    {params: {dog: 'rover'}},
    {params: {dog: 'spot'}},
  ];
}

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

<div>Good dog, {dog}!</div>
```

Generates /dogs/cliffford, /dogs/rover, /dogs/spot

inovex

Dynamic Routes - SSG Mode - Data passing with props

- props can be used to pass additional data
- props are not encoded into the URL, therefore not limited to strings

pages/posts/[id].astro:

```
---
export async function getStaticPaths() {
  const posts = [
    {id: '1', category: "astro", title: "API Ref"},
    {id: '2', category: "react", title: "Counter"}
  ];
  return posts.map((post) => {
    return {
      params: { id: post.id },
      props: { post }
    };
  });
}
const { id } = Astro.params;
const { post } = Astro.props;
---
<body>
  <h1>{id}: {post.title}</h1>
  <h2>Category: {post.category}</h2>
</body>
```

Generates: /posts/1, /posts/2

Dynamic Routes - SSR Mode

- SSR pages cannot use getStaticPaths()
- Pages will be served to any matching route
- Cannot receive props

src/pages/[...slug].astro:

```
---
import getPages from '../helper/dynamic.ts';
// Array<{slug: string, title: string, text: string}>
const pages = await getPages();
const { slug } = Astro.params;
const page = pages.find((page) => page.slug === slug);
if (!page) return Astro.redirect("/404");
const { title, text } = page;
---

<html>
<head>
  <title>{title}</title>
</head>
<body>
  <h1>{title}</h1>
  <p>{text}</p>
</body>
</html>
```


inovex

Output Modes

Mode	Variant	Effect
static	Static Site Generator	Everything created at build time
server	Server Side Rendering	Server-rendered by default; opt-in pre-rendering
hybrid	Server Side Rendering	Pre-rendered by default; opt-in server-rendering

Layouts

- Components to provide reusable UI structure
- Provide common UI elements like headers, navigation bars & footers
- Accept props & compose other components
- Placed in src/layouts

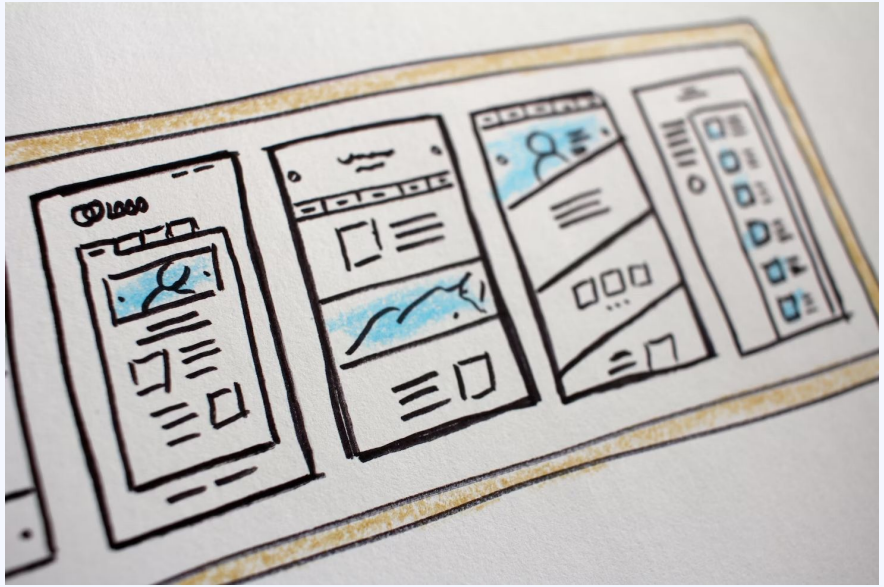


Image Source: Hal Gatewood

inovex

Content Collections

- Help organize documents
- Validate frontmatter
- Automatic TypeScript type-safety
- Each collection is a top level directory in src/content
- src/content/config.ts configures content collections

Image Source: Edgar Chaparro

Content Collections - defineCollection

- defineCollection() uses zod (23k ⭐) schemas
- automatic type generation
- automatic validation during build time

src/content/config.ts:

```ts
import { z, defineCollection }
  from 'astro:content';

const blogPosts = defineCollection({
  type: 'content',
  schema: z.object({
    title: z.string(),
    date: z.date(),
    image: z.string().optional(),
    author: reference('authors')
  }),
});

const authors = defineCollection({
  type: 'data',
  schema: z.object({...}),
});

export const collections = {
  'blogPosts': blogPosts,
  'authors': authors,
};
```

Content Collections - Folder Structure


```
---

import { getCollection } from 'astro:content';
const blogEntries = await getCollection('blogPosts');

---

<ul>
  {blogEntries.map(entry => (
    <li>
      <a href={`/my-blog-url/${entry.slug}`}>
        {entry.data.title}
      </a>
      <time datetime={...}>
        {entry.data.date.toDateString()}
      </time>
    </li>
  ))}
</ul>
```


inovex

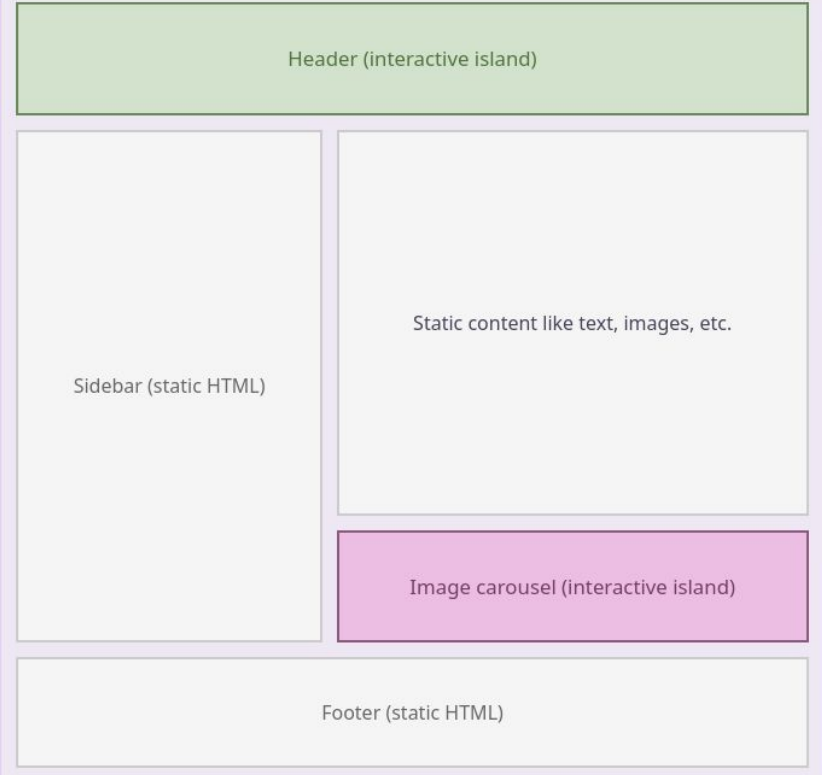
Astro Islands

- Component Islands
- Interactive Islands on a static page of HTML
- Multiple Islands render in isolation

Image Source: Datingscout

inovex

Islands Architecture

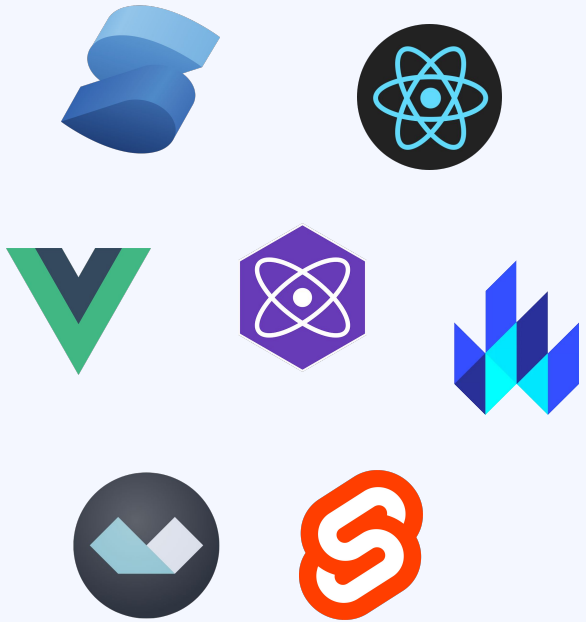


Source: [Jason Miller]

inovex

Astro Islands - Interactivity

- Multiple frontend framework integrations
- Astro will render Components (on the server-side) as HTML and strip out any JavaScript by default
- Client directives allow the creation of interactive Islands (which ship & rehydrate JS)



inovex

Astro Islands - Examples

```
---
import MyReactComponent from '../components/MyReactComponent.tsx';
import MySvelteComponent from '../components/MySvelteComponent.tsx';
import MySolidLocalDateComponent from '../components/MySolidLocalDateComponent.tsx';
---
<!-- 100% HTML, Zero JavaScript loaded on the page! -->
<MyReactComponent />

<!-- Rendered on the server, hydrated on the client  -->
<MySvelteComponent client:load />

<!-- Skips HTML server-rendering -->
<MySolidLocalDateComponent client:only="solid-js" date={date} showTime={false}>
  <time datetime={date.toISOString()}>
    {date.toDateString()} <!-- Mon Jul 03 2023 -->
  </time>
</MySolidLocalDateComponent>
```

Client Directives

Directive	Priority	Usecase
client:load	High	Interactive elements that must load immediately
client:idle	Medium	Lower-priority elements that load after the page's initial load completes
client:visible	Low	low-priority elements that load when they enter the user’s viewport
client:media=”(max-width: 50em)”	Low	Only load when a specific CSS media query is met
client:only=”solid-js”	N/A	Components that skip server-rendering and load immediately on the client side.

inovex

Astro Islands - Challenges

- Sharing state between components
- State management libraries have to be framework-agnostic
- Astro recommends using nanostores
- nanostores should not be used on server-side components

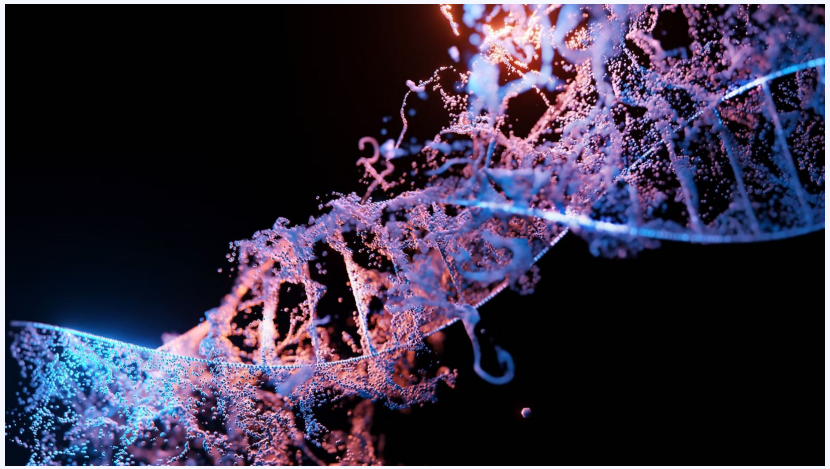
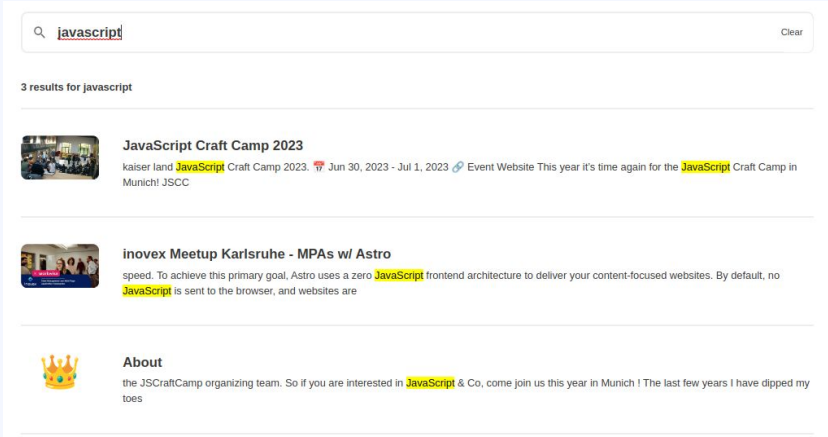


Image Source: Sangharsh Lohakare

Intermission: pagefind Island for a fully static Search

- wasm powered static search
- after SSG built, pagefind creates an search index & fragments
- includes prebuilt UI
- “can run a full-text search on a 10,000 page site with a total network payload under 300KB”[1]



Usage: `astro build && pagefind --source dist`

pagefind

inovex

Deployment Adapters

Netlify
SSR STATIC

Vercel
SSR STATIC

Firebase Hosting
STATIC

Heroku
STATIC

Deno Deploy
SSR

GitHub Pages
STATIC

Microsoft Azure
STATIC

Buddy
STATIC

GitLab Pages
STATIC

Cloudflare Pages
SSR STATIC

Edgio
SSR STATIC

Render
STATIC

AWS
STATIC

AWS Flightcontrol
SSR STATIC

Surge
STATIC

Cleavr
SSR STATIC

AWS via SST
SSR STATIC

Google Cloud
SSR STATIC

Kinsta
SSR STATIC

Deta Space
SSR STATIC

inovex

Summary

Pros

- Simple & fast
- Good typing
- Stable
- Awesome community

Cons

- .astro files tooling
- CSP problems

inovex

Resources

- 🗒️ Website, Docs, Discord
- 🎙️ JSJ 561, PodRocket, Syntax
- ✍️ Building a multi-framework dashboard with Astro
- 🎥 Astro in 100 seconds, Fireship Code Report, Astro 2.0 is Revolutionary

Thank you!

inovex is an IT project center driven by innovation and quality, focusing its services on ‘Digital Transformation’.

- founded in 1999
- 500+ employees
- 8 offices across Germany

www.inovex.de

Bernd Kaiser

Developer

bernd.kaiser@inovex.de

inovex