Astro

Content-driven sites that ship zero JavaScript by default.

01 Create a project

```
npm create astro@latest
npm create astro@latest -- --template blog
npm run dev    # http://localhost:4321
```

The wizard sets up TypeScript and git. Routes live in src/pages/; files in public/ are served as-is.

02 Component script & template

```
---
import Header from './Header.astro';
const res = await fetch(API_URL);
const data = await res.json();
---
<Header />
<h1>{data.title}</h1>
<style>h1 { color: blue; }</style>
```

The --- script runs only on the server, at build or per request. fetch() needs a full URL. <style> is scoped.

03 Props & expressions

```
---
interface Props {
  title: string;
  items?: string[];
}
const { title, items = [] } = Astro.props;
---
<h2>{title}</h2>
<ul>{items.map((i) => <li>{i}</li>)}</ul>
```

interface Props types Astro.props; destructure with defaults for optional props.

04 Slots & layouts

```
<!-- src/layouts/Layout.astro -->
<slot name="header" />
<main><slot /></main>
<slot name="footer">Default footer</slot>
<Layout>   <!-- in a page -->
  <h1 slot="header">My Page</h1>
  <p>Goes to the default slot</p>
</Layout>
```

Slot content is the fallback when nothing is passed.

05 File-based routing

index.astro	/
blog/[slug].astro	/blog/:slug
[...path].astro	Any depth: /a, /a/b/c
_draft.astro	Not routed (_ prefix)

Priority: more segments, then static, [param], [...rest].

06 Define a collection

```
// src/content.config.ts
const blog = defineCollection({
  loader: glob({ base: './src/data/blog',
    pattern: '**/*.md' }),
  schema: z.object({ title: z.string(),
    pubDate: z.date(),
    draft: z.boolean().optional() }),
});
export const collections = { blog };
```

z comes from astro/zod, glob from astro/loaders; defineCollection, getCollection, getEntry and render from astro:content.

07 Query collections

```
---
const posts = await getCollection('blog',
  ({ data }) => data.draft !== true);
---
{posts.map(({ id, data }) => (
  <a href={`/blog/${id}`}>{data.title}</a>
))}
```

getEntry('blog', id) gets one entry. Sort results yourself.

08 A page per entry

```
// src/pages/blog/[id].astro frontmatter
export async function getStaticPaths() {
  const posts = await getCollection('blog');
  return posts.map((post) => ({
    params: { id: post.id },
    props: { post },
  }));
}
const { post } = Astro.props;
const { Content } = await render(post);
```

Show it with <Content />; render() also returns headings.

09 Client directives (islands)

(none)	Static HTML, no JavaScript
client:load	Hydrate on page load
client:idle	When the browser is idle
client:visible	When scrolled into view
client:media="..."	When a media query matches
client:only="react"	Client only, skips SSR

Astro

Islands, transitions and server features.

10 Framework components

```
---
import Counter from './Counter.jsx';
import Button from './Button.vue';
---
<Counter count={5} client:load />
<Button title="Hi" client:visible />
```

Props are plain attributes in every framework (no Vue :prop). Hydrated props must serialize: no functions.

11 Images

```
---
import { Image } from 'astro:assets';
import photo from '../assets/photo.jpg';
---
<Image src={photo} alt="Hero" />
<Image src="/img/a.jpg" alt="From public/"
  width={600} height={400} />
```

Imported images are optimized and sized to avoid layout shift. public/ and remote images need width and height.

12 View transitions

```
---
import { ClientRouter }
  from 'astro:transitions';
---
<head><ClientRouter /></head>
<h1 transition:animate="slide">Hi</h1>
<img transition:name="logo" src="/logo.png">
<a href="/" data-astro-reload>Reload</a>
```

Built-in animations: fade (default), slide, none, initial. Re-run scripts on astro:page-load.

13 Endpoints

```
// src/pages/api/posts.json.ts
import type { APIRoute } from 'astro';
export const GET: APIRoute = () =>
  new Response(JSON.stringify([{ id: 1 }]));
export const POST: APIRoute = async (ctx) =>
  new Response(await ctx.request.text(),
    { status: 201 });
```

Export GET, POST, PUT, DELETE or ALL. Static output prerenders GET; others need prerender = false.

14 Server islands

```
---
import Cart from '../components/Cart.astro';
---
<Cart server:defer>
  <p slot="fallback">Loading cart</p>
</Cart>
```

The page ships with the fallback; a small script then loads the island from its own endpoint. Needs an adapter.

15 Define actions

```
import { z } from 'astro/zod';
export const server = {
  like: defineAction({
    accept: 'form',
    input: z.object({ id: z.string() }),
    handler: async ({ id }) =>
      ({ likes: await addLike(id) }),
  }),
};
```

In src/actions/index.ts; defineAction is from astro:actions.

16 Call actions

```
---
import { actions } from 'astro:actions';
const result =
  Astro.getActionResult(actions.like);
---
<form method="POST" action={actions.like}>
  <button name="id" value="1">Like</button>
</form>
{result?.data && <p>{result.data.likes}</p>}
```

The page must render on demand (prerender = false). Client code passes FormData: actions.like(formData).

17 Middleware & sessions

```
// src/middleware.ts
export const onRequest = defineMiddleware(
  async ({ locals, session }, next) => {
    const user = await session?.get('user');
    locals.user = user ?? null;
    return next();
  },
);
```

sequence() chains several. Type App.Locals in src/env.d.ts.

18 Config & adapters

```
// astro.config.mjs
export default defineConfig({
  adapter: vercel(),
  session: { driver: sessionDrivers.redis({
    url: process.env.REDIS_URL }) },
});
```

An adapter enables on-demand pages, islands and sessions.