day-7-LIA

Introduction to Astro: A Beginner's Guide

Self-paced learners

In our Online Meetings, we skipped the Step of installing `node.js` on local system by going instead with cloud dev environments with node preinstalled like Github CodeSpaces instead, if you're self-learning either try installing node.js on your System, or try a cloud dev environment like Codespaces or Stackblitz

Overview of Today

- [ ] Understanding Astro and its core concepts
- [ ] Use cases and benefits of Astro
- [ ] Setting up your first Astro project
- [ ] Project structure and configuration
- [ ] Basic Astro template Syntax

Resources that'll help you

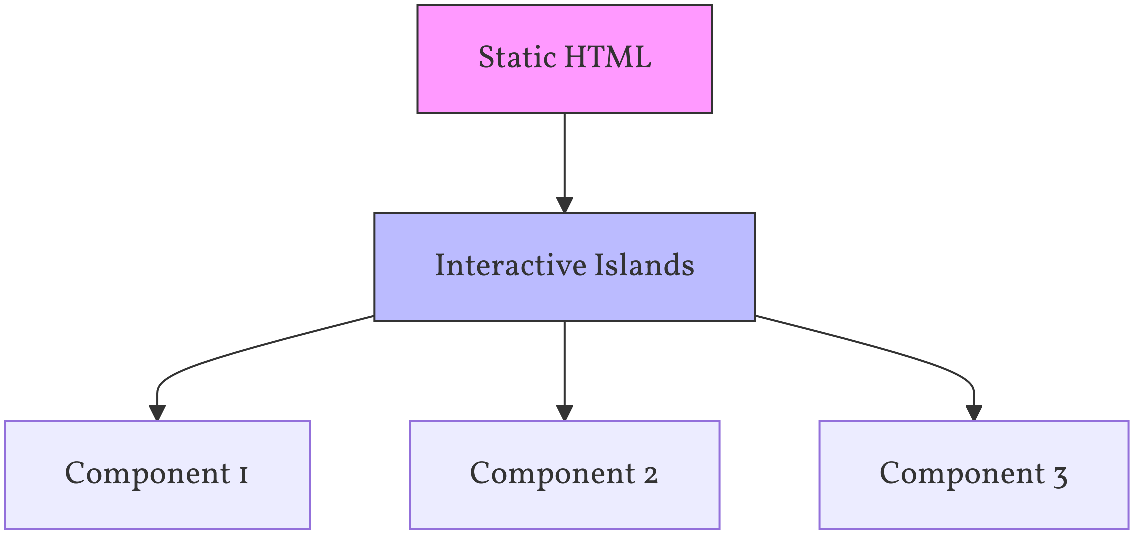
- Astro in 100 Seconds - Quick overview by Fireship
- Official Astro Documentation - Astro Docs - *Highly Recommended*, Seriously the best Documentation I've ever seen.*
- Why Astro? - Understanding Astro's benefits
- Tutorial: Build a blog - An amazing tutorial from Astro itself, *highly recommended
- Our Recordings from the online session

Understanding Astro

Astro is a modern web framework that focuses on delivering content-driven websites with optimal performance. It introduces several key concepts that make it unique in the web development ecosystem. In addition to being one of the most loved web frameworks by devs in 2024 (source: Stackoverflow.

Core Concepts

1. **Content-First**:
   - Optimized for content-rich websites
   - Built-in support for Markdown and MDX
   - Excellent for blogs, documentation, and marketing sites
   - more:
2. **Island Architecture**:



1. **Zero JS by Default**:
   - Pages load as pure HTML and CSS
   - JavaScript loads only where needed
   - Better performance and SEO

Use Cases for Astro

Astro is particularly well-suited for:

1. ✅ Content-driven websites
2. ✅ Documentation sites
3. ✅ Blogs and portfolios
4. ✅ Marketing websites
5. ✅ E-commerce product pages

Our site is actually built with Astro

When to Choose Astro

```
+ Content-heavy websites
+ Static site generation needs
+ SEO-critical (Search-Engine-Optimization) projects
+ Multi-framework components
+ Performance-focused sites

- Complex web applications
- Heavy client-side interactions
- Real-time applications
```

Setting Up Astro Projects

Prerequisites

- Node.js (version 16.12.0 or higher) ~ *preinstalled on dev environments like Codespaces*
- Text editor (VS Code recommended)
- Terminal/Command Line interface - *e.g. powershell or bash*

Creating a New Project

1. Using npm (node package manager):

```
# Create a new project with npm
npm create astro@latest

# Or with pnpm
pnpm create astro
```

Project Structure

A typical Astro project structure:

```
my-astro-site/
├── src/
│   ├── components/
│   ├── layouts/
│   └── pages/
```

```
├── public/
├── astro.config.mjs
└── package.json
```

Other necessary details on Astro Docs here

Others

Stuck on something you can't understand? You'll likely find more details on Astro docs (I highly recommend getting used to using docs)

- Official Astro Documentation

If its too difficult use our online recordings a source.

Astro template Syntax

Astro Component Syntax is a superset of HTML, those who are familiar with HTML & JSX will find really easy, like look at the following Code block:

```
---
// Frontmatter section
const name = "Abbas";
---

<!-- Template Section -->
<div>
  <h1>Hello {name}!</h1>  <!-- Outputs <h1>Hello Abbas!</h1> -->
</div>
```

- **Frontmatter**:
  - Written in JavaScript
  - Runs on build-time on the Server (won't run on client-side)
  - Used to define page-specific data and properties
- **Template**:
  - The HTML-like structure below the frontmatter
  - Can include dynamic content using JavaScript expressions in curly braces `{}`
  - Can use variables defined in the frontmatter

Together, frontmatter and templates create a powerful way to build dynamic pages where you can separate your **data** (frontmatter) from your **presentation** (template).

There a lot to cover so refer to this page as it already summarized all what we need.

Quiz

Why Option *A* doesn't work while option *B* works?

- *A*

```
---
function handleClick () {
    console.log("button clicked!");
}
---
<!-- ❌ This doesn't work! ❌ -->
<button onClick={handleClick}>Nothing will happen!</button>
```

- *B*

```
---
---
<button id="button">Click Me</button>
<script>
  function handleClick () {
    console.log("button clicked!");
  }
  document.getElementById("button").addEventListener("click", handleClick);
</script>
```

- **Answer:** because in *A*, the `handleClick()` will be run only once during build-time (aka deploying the site) since it is in the *frontmatter* section, while in *B*, It is a client-side script since it is in the *template* section (aka looks like regular html), meaning it will be run every time the user uses that button.

Note

Remember to run `npm run dev` to start your development server and view your site at `http://localhost:4321` or click the link you see in the Terminal

Tip

Use the Astro VS Code extension for better development experience

Tasks

- [ ] Create your first Astro project
- [ ] Add a simple page with some content
- [ ] Create and use an Astro component