HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

Teil 3: Astro

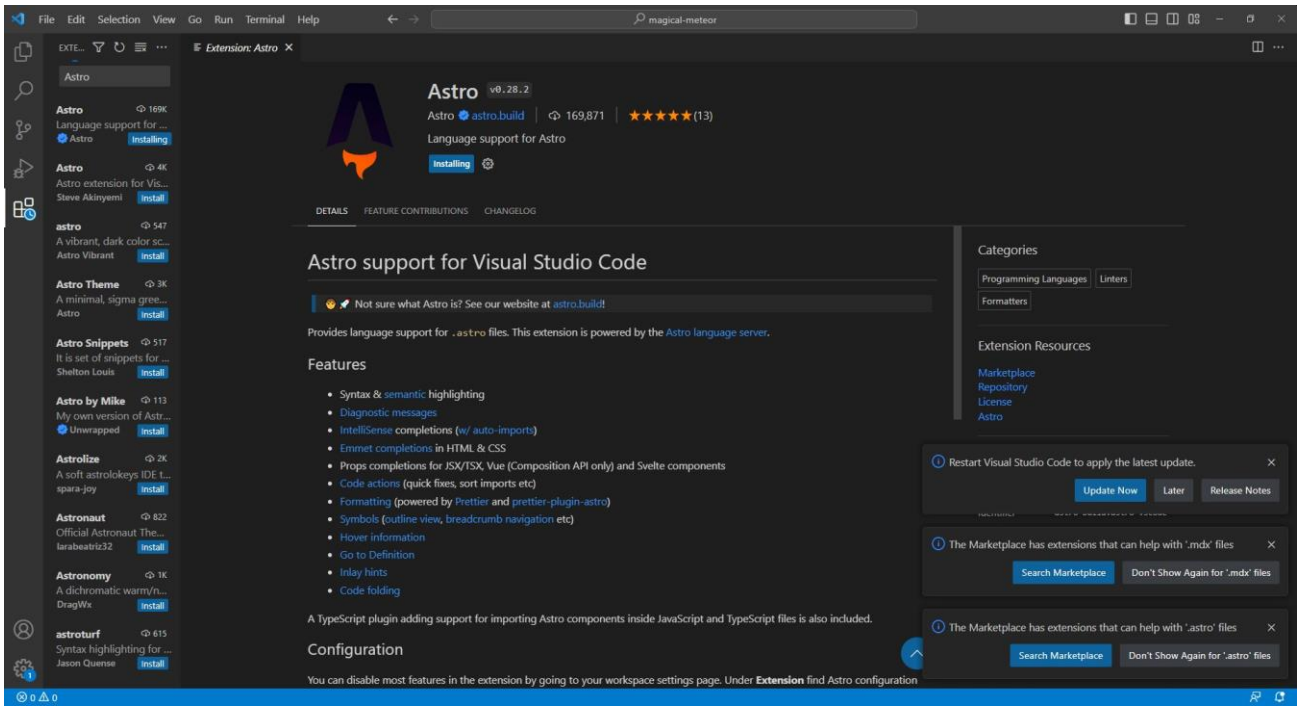
Prof. Dr.-Ing. Thorsten Weiss

Version 1.0

Astro / Umgebung einrichten

- Astro ist ein **all-in-one web framework** für die Erstellung von **schnellen, content-basierten Webseiten.**
- Astro ist vom Grundsatz her ein **Server-Side-Rendering (SSR) System** und eignet somit für Seiten wie Portfolios, Blogs etc. mit wenig UI-Logik.
- Astro erlaubt aber die einfache **Integration** von **intelligenten Komponenten** wichtiger Frameworks wie React, Svelte, Solid und vielen mehr.
- Auch die Integration des **leichtgewichtigen tailwind.css** und weiterer UI-Kits ist möglich.
- Auf diese Weise kann die **optimale SEO-capability mit komplexer UI-Logik** von Portalen oder visuellen Projekten **kombiniert** werden.
- Die **vergleichsweise einfache Installation** und Deploy ermöglicht einen schnelleren Einstieg als in andere Frameworks.
- Astro ist kein Homepage-Baukasten, es ist mehr eine Plattform, um SSR mit intelligenten Komponenten zu kombinieren.
- Astro bietet die Möglichkeit, **Endpoints einfach zu integrieren** und ohne großartige Serverkenntnisse Daten bereitzustellen. Es erinnert an die **serverless Functions** von Firebase oder AWS.
- Durch den SSR Ansatz und die Integration intelligenter Komponten können sowohl die SSR als auch die Ansätze der Single Page Apps gelernt werden.
- Astro bewirbt, dass im Frontend kein JS nötig ist und das ein Geschwindigkeitsvorteil hat. Dies stimmt für rein darstellende Seiten, aber nicht für Komponenten mit aufwendiger UI-Logik.
  → Aber dafür ist die Integration der intelligenten Komponenten ermöglicht.
- So gesehen widerspricht sich die Webseite von Astro selbst. Es funktioniert erfahrungsgemäß aber sehr gut, weswegen wir darüber hinweg sehen können ☺

- Ein Astro Plugin bietet Support für Astro unter Visual Studio Code.
- Klicken sie auf Erweiterungen und geben sie “Astro” in die Suchleiste ein. Das Plugin von astro.build funktioniert schon recht gut.

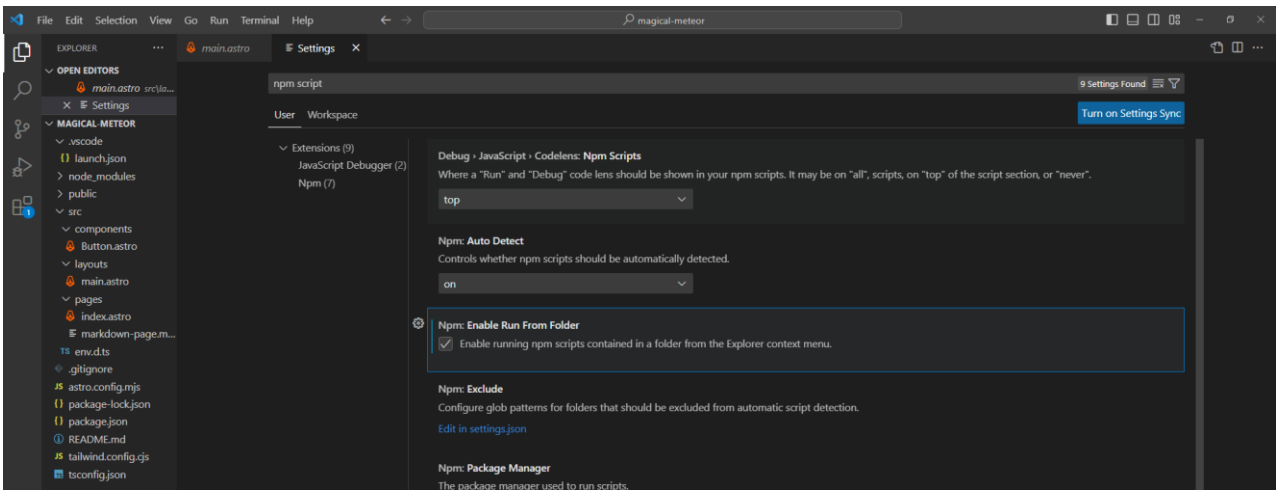


Astro / Umgebung einrichten

- Um Astro mit einem Click starten zu können, klicken sie auf

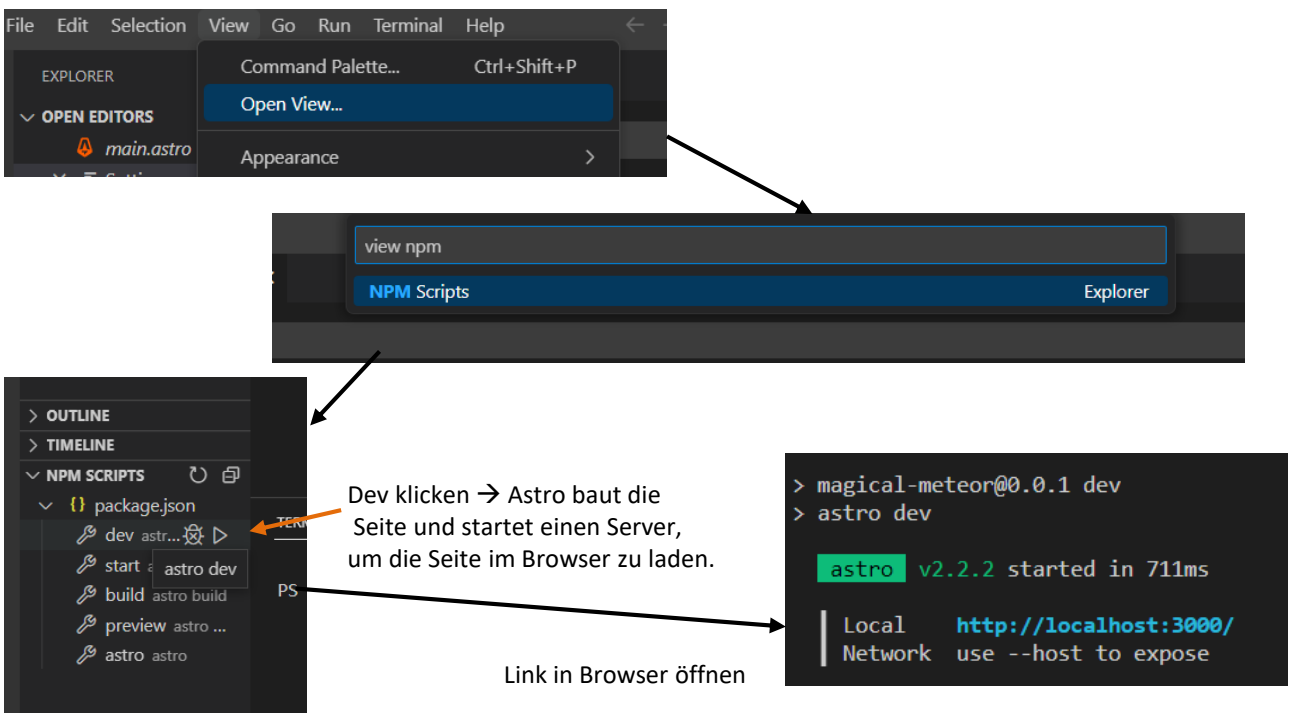
  Strg + , (Also Strg und ,)

- „npm script“ in die Suchleiste eingeben und den Punkt „Enable Run From Folder“ markieren.



- Terminal öffnen und „npm i“ eingeben.

- Dann auf npm Scripts klicken. Falls dieses Feld nicht sichtbar ist: View → Open View → view npm

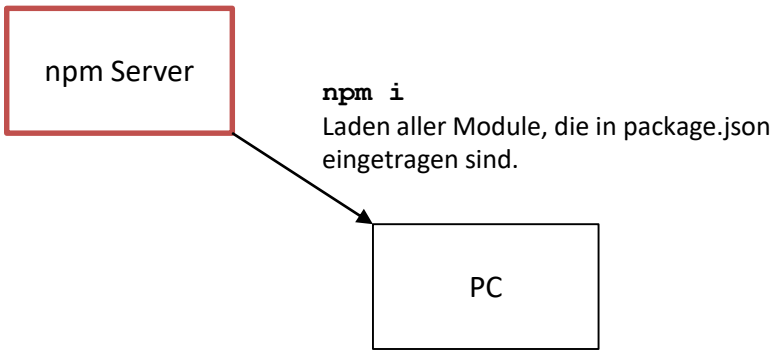


- d

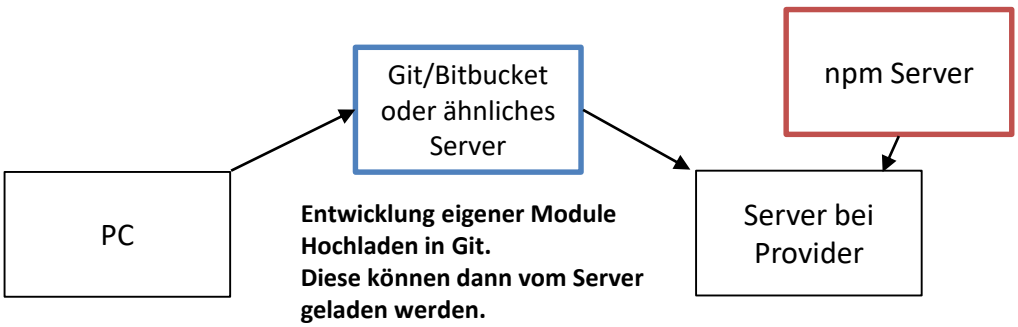
NPM

Startprojekt und npm

- Zur Installation von Astro können Startprojekte herangezogen werden. Das erste Startprojekt heißt “1_AstroBasic”.
- Astro nutzt NodeJS und den integrierten **Paketmanager npm** (Node Package Manger), der mit Node installiert wird.
- Dieser erlaubt das **komfortable Laden von JS-Bibliotheken**.
- Die Datei package.json enthält die Pakete und beim Ausführen von “`npm i`” werden diese
- Zum Entwickeln werden die Daten der Webseite lokal gespeichert.



- Für Produktivsysteme wird der Quellcode auf einen Quellcodeserver wie gitlab, bitbucket oder github geladen.
- Auf dem Server kann dies gezogen werden.



Dies wird auf einem Produktivsystem so gemacht. Hier benötigen wir es zur lokalen Entwicklung nicht.

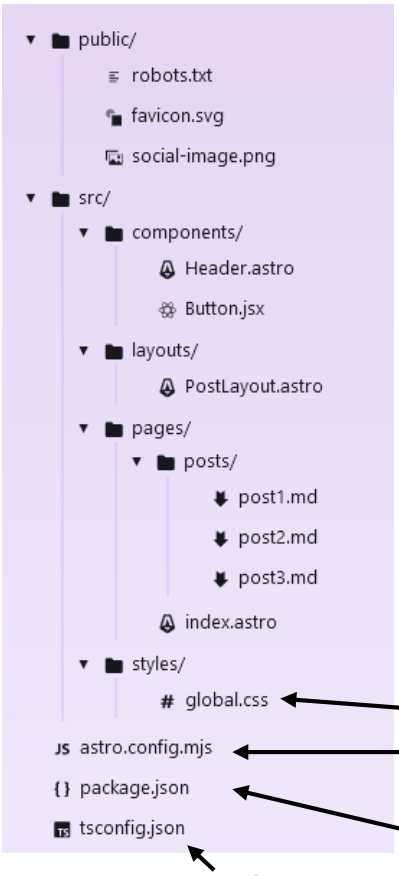
- Installation von npm Paketen erfolgt mit

  npm i <paketname>

- Pakete sind auf npmjs.com zu finden und beschrieben.

Astro

Projektstruktur von Astro: Quelle https://docs.astro.build/en/core-concepts/project-structure/



Der Public-Ordner enthält Objekte wie Bilder, die public zugänglich sind.

Components sind wiederverwendbare Teile einer Siete, z.B. Header, Navbars, Footers.

Der Aufbau ist **filebased**. Das heißt, dass die Links so wie die Ordnerstruktur aufgebaut ist.

Die Seite post1.md wird als www.mysite.de/posts/post1.md aufgerufen. Daher müssen Entwickler sich nicht um die Verlinkung kümmern.

Styles enthält global geltende CSS

Enthält die Konfigurationen von Astro

Package.json enthält eine Liste von Paketen, die zur Ausführung des Projektes nötig sind. Durch den Befehl „npm i“ werden alle Pakete installiert.

Enthält die Konfigurationen für Typescript und Paths

Als Pages werden folgende Formate unterstützt.

- .astro : Intelligente SSR Komponenten
- .md : Markdowns
- .mdx (with the MDX Integration installed) : Erweiterte Markdowns
- .html : HTML Seiten
- .js/.ts (as endpoints) : Server-Endpoints in Javascript / Typescript.

Layouts bilden eine Art Rahmen für die Seiten.

• Als Beispiel Template wird nofuss von nofuss.pages.dev (MIT Lizenz) verwendet.

Astro - Aufbau

WEBENTWICKLUNG 2
one step ahead

Layouts

- Layouts sind wiederverwenbare Templates für Seiten.
- Die Pages werden durch den Slot-Tag in die Layouts injiziert. So kann ein Rahmen geschaffen werden und eigene neue Seiten an diese Stelle platziert werden.
- https://docs.astro.build/en/core-concepts/layouts/

```
---
import BaseHead from '../components/BaseHead.astro';
import Footer from '../components/Footer.astro';
const { title } = Astro.props
---
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <BaseHead title={title}/>
  </head>
  <body>
    <nav>
      <a href="#">Home</a>
      <a href="#">Posts</a>
      <a href="#">Contact</a>
    </nav>
    <h1>{title}</h1>
    <article>
      <slot /> <!-- your content is injected here -->
    </article>
    <Footer />
  </body>
</html>
```

Elemente, die auf mehreren Seiten zu sehen sein sollen, könne so verbleiben.
Beispielsweise wird die Navbar über den Inhalt, der getauscht werden soll, eingefügt.

Beim Slot-Tag wird die Seite dann eingefügt.

Astro – Aufbau einer einfachen Seite


```astro
---
const { content } = Astro.props;
---

<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width" />
        <link rel="icon" type="image/svg+xml" href="/favicon.svg" />
        <title>{content.title}</title>
    </head>
    <body>
        <slot />
    </body>
</html>
```

Beim Aufruf /overview wird in Slot overview.astro injiziert.

Die 3 Striche --- umschließen einen **Javascript Block ein,** der beim Aufruf **der** Seite **auf dem Server ausgeführt wird.**

Imports machen es möglich, die importierten Komponenten innerhalb der Komponente zu nutzen.

Definition einer Datenstruktur

```astro
---
import Layout from "@layouts/Layout.astro";
import Features from "@components/Features.astro";
const homepage = [
  {
    title: "Super schnell",
    content: "Server Side Rendering ist für rein darstellende Seiten sehr schnell.",
    icon: "charm:rocket",
  },
  {
    title: "Einfach zu nutzen",
    content: "Wenn man weiß, wie es geht, ist es einfach zu nutzen. Aber bis dahin müssen wir uns einige Konzepte anschauen.",
    icon: "ph:code-bold",
  }
];
---

<Layout content = {{title: "Web2"}}>
  <section class="grid py-10 place-items-center text-center">
    <h1 class="sm:text-4xl text-3xl font-bold mb-6 dark:text-white">Web2 Template</h1>
    <p>Web2: Wir bauen schon mal mit dem Backend eine Seite.</p>
    { homepage.map((homepage) => <Features title={homepage.title} icon={homepage.icon} content={homepage.content} />) }
  </section>

</Layout>
```

Auf diese Datenstruktur kann innerhalb der Datei zugegriffen werden.
Die Map Funktion wandelt die Datenstruktur in Komponenten Feature um.

Jeder Komponente können Props (Properties) mitgegeben werden.
Hier 3 Props: title, icon, content.

Es werden 2 Instanzen von Features gebildet, die untereinander dargestellt werden.

```astro
---
import { Icon } from "astro-icon";

export interface Props {
  title: string;
  icon: string;
  content: string;
}

const { title, icon, content } = Astro.props;
---

<section class="py-14 sm:flex sm:items-center sm:justify-evenly text-center sm:text-left">
  <Icon name={icon} class="overflow-hidden inline-flex sm:m-8 w-28 h-28" />
  <header class="mt-8 max-w-lg sm:my-0 mx-auto sm:mx-0">
    <h1 class="sm:text-4xl text-3xl font-bold mb-4 dark:text-white">{title}</h1>
    <p class="max-w-10 font-normal leading-7 dark:text-zinc-300 sm:max-w-sm md:max-w-xl">{content}</p>
  </header>
</section>
```

Die Props werden über ein Interface bekannt gemacht.

Und über Objektdestrukturierung zugänglich gemacht.

Tailwind.css bietet einen Vielzahl an CSS-Klassen an.