Emissions Frontend (Angular) Documentation

The Angular frontend of the **Emissions project** provides a **responsive** and **interactive interface** for visualizing and filtering emission data. At the center of the application is the **standalone** `AppComponent`, which handles the **loading of emission data** from the backend, manages **user-defined filters**, and communicates the processed data to child components. The component's template includes **conditional rendering** to display *loading* and *error* states, and provides **dropdowns** for filtering by *country*, *activity*, and *emission type*. Below the filters, a **table** displays the filtered emission records, and a custom `EmissionsChartComponent` visualizes emission trends over time using the `ngx-charts` library for **dynamic line chart rendering**.

The frontend's **styling** is implemented with SCSS to ensure a **clean and modern layout**. The host container applies **padding**, sets a **minimum height**, and defines a **background color** for the page. Filter elements are styled as **flexible blocks** with consistent spacing, padding, and focus effects for accessibility. Tables are designed with a **professional aesthetic**, including **hover effects** and responsive handling for smaller screens. The `EmissionsChartComponent` occupies the full width and has a fixed height of 400 pixels, creating a consistent visualization area. Media queries ensure the interface remains **usable and readable** on both desktop and mobile devices.

From a **functional perspective**, the frontend relies on **services** and **models** to interact with the backend API. The `EmissionsService` manages **HTTP requests** to fetch emission data and supports optional filtering parameters for *country*, *activity*, and *emission type*. The `Emission` interface defines the structure of the data with properties for year, emissions amount, emission type, country, and activity. When the `AppComponent` initializes, it fetches all emission records and extracts **unique values** for countries, activities, and emission types to populate the filter dropdowns. Changing a filter triggers the recalculation of the **filtered dataset**, which is then passed to the `EmissionsChartComponent`. This approach ensures a clear **separation of concerns**, where the service focuses on **data retrieval** and the component manages **presentation and user interaction**.

The project includes **modern development tools** and **deployment configurations**. The `Dockerfile` and `docker-compose.yml` enable running the frontend in a **containerized environment**, guaranteeing **consistent builds** and **easy setup** across systems. Environment variables are dynamically generated with a Node.js script, allowing configuration of **API endpoints** and **default filters** without changing the source code. Angular's **standalone component design**, **TypeScript typing**, and integration with **RxJS** for asynchronous data handling provide a **maintainable**, **scalable**, and **type-safe frontend application**. This architecture allows the application to be extended easily with **additional visualizations** or **advanced interactions** while keeping the codebase organized and efficient.