Architecture pattern to help make things reusable and testable.

Container → Input/Output ← Presentation

- Container = how things work. Loading and Dispatching.

- Presentation = presenting the data, how things work, but doesn't deal with observables.

NgRx - Reducing Boiler Plate

NgRx Schematics works with the CLI. Blueprints that make it very easy to use.

NgRx Data

• Becoming an official NgRx package soon

• Really nice way to build out NgRx Schematics. Should look into it.

What belongs in the store: SHARI

• **Shared -** shared state is accessed by many components and services

• **Hydrated -** state that is persisted and hydrated from storage

• **Available -** state that needs to be available when re-entering routes

• **Retrieved -** state that needs to be retrieved with a side effect

• **Impacted -** state that is impacted by actions from other sources

What doesn't go in the store:

• Forms… WTF?

• Data that can't be serialized

• State that is solely owned by a component doesn't belong in the store.

Ideally

Application can be booted up with a specific state to recreate any scenario.

NgRx Container / Presentation

Architecture pattern to help make things reusable and testable.
Container → Input/Output ← Presentation

- Container = how things work. Loading and Dispatching.

- Presentation = presenting the data, how things work, but doesn't deal with observables.