**Project: FormBuilder**

**Work Begins**

git checkout v1-hard-coded

* v1-hard-coded: Entire form hardcoded
* v2-from-config: Entire form from configuration
  + PROBLEM: We can now only show an input field wrapped in a .form-group how about in a navbar?
* v3-more-config-not-dry: Adding more configuration (placeholder & hint)
  + BUG: config.hint should probably also be implemented for secret and toggle? How to keep this DRY?
* v4-going-all-out: Setting up a framework that works with React/Angular and Bootstrap/Foundation

**The Business steps in**

Since our FormBuilder project is so successful, team M will also start using it. Unfortunately, these guys do not use Bootstrap but Foundation. Oh, and they also use React instead of Angular.

Much needed functionality is missing also:

* Validation, tooltips, ...
* Single layout is boring: TableLayout, ColumnLayout
* Confirmation dialogs, selection/search dialogs, wizards, ...

**Exercise DesignPatterns**

Ideally, we would split up the project in separate npm packages so that the Angular team does not create a dependency on the React library.

* formbuilder-core:
  + **Bridge** (Structural): To achieve loose coupling between react/angular and bootstrap/foundation
  + **Builder** (Creational): Turn configuration into classes/components
  + **Abstract Factory** (Creational): Create the correct classes from angular/react and bootstrap/foundation
  + **Decorator** (Structural): Add extra functionality to controls with HigherOrderComponents (React only)
  + **Composite** (Structural): The Abstract Factory may return a Composite: Once of our controls may contain controls of their own
    - **Visitor** (Behavioral): If it did return a Composite, we could add extra behavior in an OCP way (validation, layouting, tooltips, ...)
    - **Chain Of Responsibility** (Behavioral): Events may be handled at different levels (the control itself, its parent, the main window)
  + **Mediator** (Behavioral): React/Foundation/... is getting complex... Do we need a Mediator
  + **Memento** (Behavioral): We can store the state of the controls in localStorage and restore it on demand
  + **Observer** (Behavioral): Users of our library may want to respond as changes happen (ex: user input)
* formbuilder-angular/react: The controls in Angular/React
* formbuilder-bootstrap/foundation: The Html markup in Bootstrap/Foundation
  + **Strategy** (Behavioral): Implement the interfaces of formbuilder-core in Angular/React/Foundation/Bootstrap
  + **State** (Behavioral): Have controls behave/look differently based on disabled / error / focused / ...
  + **Adapter** (Structural): Have an existing npm package replace some functionality of our framework (Validation, ...)

**Which design patterns could we use where and to what purpose?**