This code base is better!

Both of us have comfortable experience with Javascript and Node.js, which will allow Richard, the one new to the codebase, to easily onboard. In addition, Javascript has a console.table function which helps us visualize the game board quite easily.

In regards to the confidence in a working implementation of the core Santorini components -- including the Board, RuleChecker, and Player -- we have passing test cases written for their core implementations of their respective interfaces in the Javascript code base. In addition, the code base we are using has done historically better on the “Test Fest” than the others.

In regards to features in implementation, we are listing several decisions in the implementation that we really like in this code base:

- Use of monads/maybes for return of values when invalid input was provided. This flexibility in passing maybes gives us room to choose where we’d like to raise Errors in the case of invalid input.
- In the player component, we liked a way of managing the color for a single player, and then using that information to generate a ‘color’ + 1 and a ‘color’ + 2.

In regards to the design of the components, we particularly liked how well organized this code base was. Each library had a very well defined purpose that aligned with the interfaces that the entire class has been implementing. This allowed Richard to onboard faster and makes the code base easier to understand.