To: Matthias Felleisen
From: Jakob Hain and Kevin Zhang
Subject: Racket+Docs Overview

We plan to build a Racket language extension. Our extension adds support for documentation "mixed in" with Racket code, like how Javadoc adds support for documentation "mixed in" with Java code. This documentation can be compiled into HTML.

The language itself will have the same base syntax and semantics as Racket. But it will have additional, optional forms for adding documentation. The programmer can add documentation for *defines* and *define-syntaxes* with *define-docs*, and create data definitions with *define-data*. These forms will contain other forms for signatures, purpose statements, examples, accumulators, generative descriptions, and effects.

Racket+Docs programs can theoretically be run completely ignoring the documentation forms - the forms won't affect the runtime semantics of the language at all. However, when the program is compiled, the compiler will use the documentation to create a function, **(compile-docs path)**. This function will generate HTML documentation via Scribble at the given path. In the future, the compiler will verify that the data types used in function signatures actually exist, and report errors if they don't. It will run the examples to make sure they actually work, and report errors if they fail. And it will resolve references to identifiers spliced in raw text, generating arrows pointing to their binds.

Of course, the documentation forms won't just be used by the compiler. They'll also provide useful information to developers reading the code directly. The ultimate purpose of Racket+Docs is for developers to write documentation which they can easily understand, and which the compiler can "understand" too.

Language Specification: The language's specification consists of a *vocabulary and grammar*, *scoping rules*, *and* *semantics*.

Milestones:

- Recognize the language's forms, and parse them correctly. The rest of the Racket+Docs program should still run. For now, recognizing the forms will print out the relevant information.
- Generate Scribble code from user-defined documentation by calling a function. First print the Scribble, Then, compile the Scribble code into a collection of webpages, and write them to a given path.
- Add validation. Recognize type signatures and type-check, using turnstile or typed racket. Make DrRacket verify examples, using rackunit.
- Slightly modify the parser so it automatically recognizes raw text (e.g. in purpose statements), so the text doesn't need to be quoted. Also make DrRacket recognize expressions spliced in the text, color them differently, and resolve their references.
- Add kinds and type variables.

Github Repo: https://github.com/FDWraith/Racket-Docs