# 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, similar to how Javadoc adds support for documentation "mixed in" with Java code.

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 forms describing *defines*, and create data definitions via the new form *define-data*. There will be explicit forms for signatures, purpose statements, important sections (e.g. accumulators or effects), examples, and commentary.

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 forms to generate HTML documentation via Scribble. The compiler will use arrows to track data definitions and their usage inside of documentation forms. In addition, 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.

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](http://www.apple.com/), and [scoping rules](http://www.apple.com/)*.*

*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. For now, this function can only be called within the definitions pane.
* 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. Also make DrRacket recognize expressions within the text, color them differently, and resolve references.
* Add type variable support.