Racket Language and raco tools

CONCEPTS

A file starts with #lang racket (or racket/base, typed/racket, ...). Running racket <file> requires that module and runs its main submodule if present.

Write #lang racket in app.rkt
↓ `raco make`
Compile compiled/*.zo bytecode
↓ `raco test`
Test test submodule
↓ `raco exe`
Ship executable + distribute

racket with a file argument does not open a REPL. Add -i to run the file and then stay in the REPL.

RUN RACKET

`racket`
Start the REPL

`racket` *`<file.rkt>`* **`[`***`<args>`***`...]`**
Run a module with arguments

`racket -e '`*`<expr>`***`'`**
Evaluate an expression and print it

`racket -i` *`<file.rkt>`*
Run a file, then enter the REPL

THE REPL

`(exit) / Ctrl+D`
Leave the REPL

`,enter` *`<file.rkt>`*
Enter a module's namespace

`,require` *`<module>`*
Require a module

RACO MAKE AND EXE

`raco make` *`<file.rkt>`*
Compile a module and its dependencies

`raco exe -o` *`<out>`* *`<file.rkt>`*
Build a stand-alone executable

`raco distribute` *`<dir>`* *`<exe>`*
Package an exe with its runtime files

TESTS

`raco test` *`<file.rkt>`*
Run a module's test submodules

TESTS (cont.)

`raco test` *`<dir>`*
Test all modules in a directory

`(module+ test ...)`
Add tests inside the module

`(require rackunit)`
Load the unit-test library

`(check-equal?` *`<actual>`* *`<expected>`***`)`**
Check two values are equal

PACKAGES

`raco pkg install` *`<pkg>`*
Install a package from the catalog

`raco pkg install --auto` *`<pkg>`*
Install and fetch missing dependencies

`raco pkg update` *`<pkg>`*
Update a package

`raco pkg remove` *`<pkg>`*
Uninstall a package

`raco pkg show`
List installed packages

`raco pkg install --link` *`<dir>`*
Link a local directory as a package

TOOLS AND DOCS

`raco docs` *`<term>`*
Search the documentation

`raco fmt` *`<file.rkt>`*
Print the formatted program

`raco fmt -i` *`<file.rkt>`*
Format in place

LANGUAGE BASICS

`#lang racket`
Full language with common libraries

`#lang racket/base`
Small, fast-loading language

`(require` *`<module>`***`)`**
Import a library or file

`(provide` *`<id>`* **`...)`**
Export bindings from a module

`(define (`*`<f>`*** *`<x>`***`) ...)`**
Define a function

`(module+ main ...)`
Code run by racket <file>