HOP

Scheme-like Web Apps

http://hop.inria.fr

Overall architecture

- Scheme2js - compiles scheme to javascript
- Web browser is responsible for all GUI interaction
- A web "broker" is responsible for application logic
  - Also responsible for communicating with other brokers for distributed computing
- HOP is currently implemented as a web server but presumably could be something else

Stratums

- A stratum defines the separation between data and code
  - (In scheme isn't code also data?)
- Main stratum
  - Application logic, executes on the server
- GUI stratum
  - User interface logic, executes on the client
- Both strata can be defined in the same file

Going between strata

- The escape character "~" takes you in to the GUI:
  - (<P> :onclick ~(alert "foo"))
- The escape character "$" takes from the GUI back to the server:
  - (let ((foo "bar")) (<P> :onclick ~(alert $foo)))

Example @ http://reason.local:8080/HopEx/Escape.hop

Services

- created with a define-server directive
- provides an entry point for client to fetch data from the server after a page load

Example @ http://reason.local:8080/HopEx/Svc.hop

Hello World

- Slightly more complex service example
- Reads files

Example @ http://reason.local:8080/HopEx/Hello.hop

Other stuff

- HSS - hop CSS which allows HOP code to be embedded in CSS files
- Sqlite - HOP provides an Sqlite module for serialization
- No tightly integrated serialization layer