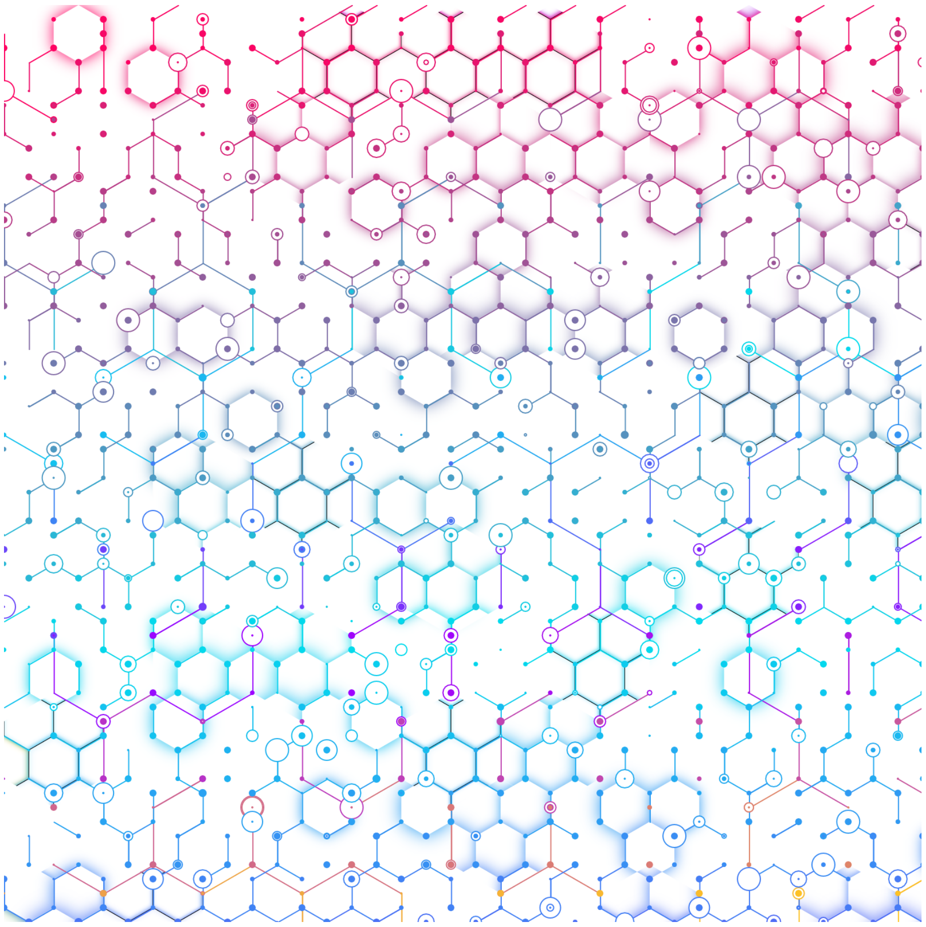
Schemers Competition 2019

Static Blog Generator

The goal of the competition is to build static blog generator using one of the many Scheme programming language implementations.

All the features are described along the number of points given for their completion. Code style will also be scored by a jury.



(austerity by /u/shvembldr [instagram])

Official Mailing List

Sponsors

- Diamond: TBD
- Gold: TBD
- Silver: TBD
- Bronze: TBD

Prizes

TBD

Specifications and Scoring

Scheme Implementation Bonus

- Classic: 50 pts
- R5RS: 25pts
- R6RS: 25pts
- R7RS: 75pts

The Right Thing

> *Simplicity — the design must be simple, both in implementation and interface. It is more important for the interface to be simple than that the implementation be simple.*
>
> *Correctness — the design must be correct in all observable aspects. Incorrectness is simply not allowed.*
>
> *Consistency — the design must not be inconsistent. A design is allowed to be slightly less simple and less complete to avoid inconsistency. Consistency is as important as correctness.*
>
> *Completeness — the design must cover as many important situationsas is practical. All reasonably expected cases must be covered. Simplicity is not allowed to overly reduce completeness.*

Richard P. Gabriel

It is highly recommend to have a Continuous Integration facility hooked into the git repository whatever the platform you are developing for. Example, on source hut check man.sr.ht/builds.sr.ht/. Otherwise, describe in the documentation the steps required to run your project.

All submission must come with a `Makefile` with the following targets:

- `make init` will setup a local environment for running the blog generator from source that do must not require superuser privileges.
- `make check` will run any unit tests you have written.
- `make doc` must build the documentation.
- `make render` must render the source files into html as described in the following specification.
- `make server` will spawn a webserver at port 8000 suitable to browser the generated website.
- `make pdf` must render the blog in `.pdf` format (if implemented)
- `make epub` must render the blog in `.epub` format (if implemented)

The following scoring takes into account the fact that any public feature must be documented.

Getting started

- Render an index page sorted by date chronologically (50 points)
- Render posts based on the following formats: sxml, skribe, org-mode, markdown, html (25pts per supported formats)
- Syntax highlighting for scheme code with rainbow delimiters (50pts)
- Posts feed in chronological order: RSS, ATOM (25pts per supported formats)

Forward

- Add support for keywords the best way you can (25 points per supported formats)
- Add pages for keywords listing posts with the given keyword in chronological order (50 pts)
- Add a feed per keyword (25pts per supported formats)

Beyond

- Add support for posts in mutliple languages (100pts)
- Add support for custom templates (100pts)
- Add support for `.pdf` rendering of the whole blog (50pts)
- Add support for `.epub` rendering of the whole blog (50pts)
- Add a favicon (100pts)

The worse-is-better

> *Simplicity — the design must be simple, both in implementation and interface. It is more important for the implementation to be simple*

> *than the interface. Simplicity is the most important consideration in a design.*
>
> *Correctness — the design must be correct in all observable aspects. It is slightly better to be simple than correct.*
>
> *Consistency — the design must not be overly inconsistent. Consistency can be sacrificed for simplicity in some cases, but it is better to drop those parts of the design that deal with less common circumstances than to introduce either implementational complexity or inconsistency.*
>
> *Completeness — the design must cover as many important situations as is practical. All reasonably expected cases should be covered. Completeness can be sacrificed in favor of any other quality. In fact, completeness must be sacrificed whenever implementation simplicity is jeopardized. Consistency can be sacrificed to achieve completeness if simplicity is retained; especially worthless is consistency of interface.*

The goal of this part of the challenge is to be **creative**. *Preliminary hints* are given to the three expected features in some form of latin riddles.

Animata magna minimis

The first mysterious hint was not revelead yet!

Voluptaria cupiditatis parens

The second mysterious hint was not revelead yet!

Fake Bibendum

The last mysterious hint was not revealed yet!

Calendar

- 2019/07/26: Official start of the competition.
- 2019/09/01: *first mysterious hint* will be revealed!
- 2019/09/30: Last call for participation, after that date nobody can enter the competition.
- 2019/10/01: **second mysterious hint** will be revealed!!
- 2019/11/01: ***last mysterious hint*** will be revealed!!!
- 2019/12/14: End of the competition
- 2019/12/22: The score board will be published.

Teams

There must a maximum of three persons in a team.

- ...
- ...
- ...

FAQ

Should the source of my program be freely available?

Yes.

Can I use pandoc?

If you find it is useful: yes!

What can I do ahead of time?

There is nothing like ahead of time, the competition is already started!

Can I fork an existing project?

Yes, but you must actually produce some Scheme code.

What happens if I'm not going to finish in time?

Commit your code! If anything works, it will be scored based on that.

What is in it for you?

More Scheme code!

What is fun?

Scheme programming language is fun!