

Overview of the recent and upcoming features of Learn-OCaml

Érik Martin-Dorel

Dept. *Fiabilité des Systèmes et des Logiciels*, Lab. IRIT
Univ. Toulouse III - Paul Sabatier

Wednesday 23 March 2022

Agenda

- 1 Introduction
- 2 Some recent features in learn-ocaml ($\leq 0.14.0$)
- 3 Upcoming features

Agenda

1 Introduction

2 Some recent features in learn-ocaml ($\leq 0.14.0$)

3 Upcoming features

Wrap-up of Learn-OCaml's history

- Originated in the OCaml MOOC (2015 then 2017, 2018, 2019, 2020)
 - Course contents created by Univ. Paris Diderot (Roberto Di Cosmo, Yann Régis-Gianas, Ralf Treinen...)
 - Platform developed by OCamlPro (Benjamin Canou, Çağdaş Bozman, Grégoire Henry, Louis Gesbert, Pierrick Couderc...)
 - Based on openEDX (running everything from the browser)
- 2016: OCamlPro made Learn-OCaml indep./openEDX, AGPL license
- 2018: © transfer to the OCaml Software Foundation, MIT license
- 2022-03-23: 278 PRs integrated from 30 contributors

(see also Louis Gesbert's video at [IRILL seminar](#))

Learn-OCaml home page: summary of its main features



Nickname (\Rightarrow BACKEND): optional name to help identifying the user. Can be for example: FirstLast@Group1.

Token (\Rightarrow BACKEND): unique string (e.g. GX9-HBS-1KS-A1J) serving as login+password (can't be changed!)

Download a .zip archive (\Rightarrow BACKEND) with all submitted exercises (i.e., with the Grade button, not just Sync)

Choose an activity.

"Interactive tutorials": sequence of several topics with snippets that can be automatically copied to a toplevel.

Lectures: sequence of web slides with $\leftarrow \rightarrow$ arrows and ocaml syntax highlighting.

(The Main Feature™) Exercises with description, prelude, template, editor, toplevel, and custom graders.

Client-side toplevel (less useful than Playgrounds!)

Playgrounds: editor, toplevel, and customizable preludes.

Teacher Dashboard (\Rightarrow BACKEND): track the progress of students on the exercises (if they clicked on Grade)

Learn-OCaml exercises repositories: the "exodir" spec. I

```
the-repository
├── exercises
│   ├── index.json    → ordered list of exos
│   ├── tp1
│   │   ├── descr.md → questions
│   │   ├── meta.json → meta-data
│   │   ├── prelude.ml → given code
│   │   ├── prepare.ml → hidden given code
│   │   ├── solution.ml → complete solution
│   │   ├── template.ml → starter code
│   │   └── test.ml → ocaml grader
│   ├── tp2
│   └── ...
├── lessons
│   └── ...
├── playgrounds
│   └── ...
├── tutorials
│   └── ...

```

`$ learn-ocaml build serve --repo=the-repository`

Learn-OCaml exercises repositories: the "exodir" spec. II

```
the-repository/exercises/index.json:
{
  "learnocaml_version": "1",
  "groups":
  { "group1":
    {
      "title": "Some group of exercises",
      "exercises": [ "tp1", "tp2" ]
    }
  }
}
```

cf. https://ocaml-sf.org/learn-ocaml/exercises_format.html

Summary of the architecture of Learn-OCaml

- opam dependencies: ocaml, lwt, and js_of_ocaml
- **client-side** grading (an "exploitation" is still possible currently)
- **static** deployment:
 - a running backend is optional in learn-ocaml !
 - <https://github.com/ocaml-sf/learn-ocaml-public> (GH Pages)
- full-stack deployment (with a running backend):
 - using docker/docker-compose and an [ocamlsf/learn-ocaml](#) image
 - or using the [learn-ocaml-essok](#) project
 - or using the [learn-ocaml](#) binaries (handy for local tests)
 - **provides** Student accounts and Teacher accounts (with a Dashboard)

Agenda

- 1 Introduction
- 2 Some recent features in learn-ocaml ($\leq 0.14.0$)
- 3 Upcoming features

git clone

- Git remote → http(s) protocol, **read-only**

Student: (NicknameChange | Sync | Grade) → git commit

CLI: `git clone $URL/$TOKEN/learnocaml-workspace.git`

```
~/forge/git/learn-ocaml ((tags/v0.14.0))
$ docker run -rm -d -v "$PWD/demo-repository:/repository:ro" -p 8080:8080 ocamlsf/learn-ocaml:0.14.0
[sudo] password for emartin:
6c1614589341172be223ff2cd5e2ae0edafb077647963e3a95cfce0424e7f36f
~/forge/git/learn-ocaml ((tags/v0.14.0))
$ docker logs 6c1614589341172be223ff2cd5e2ae0edafb077647963e3a95cfce0424e7f36f
Learnocaml v.0.14.0 running.
Updating app at ./www
demo [OK]
demo2 [OK]
Learnocaml server v.0.14.0 starting on port 8080
Initial teacher token created: X-JUZ-3S9-11X-YJM
~/forge/git/learn-ocaml ((tags/v0.14.0))
$ #-- ouvre http://localhost:8080, crée un utilisateur (OCO-SB1-B1A-YZG) et répond à l'exo demo ---
~/forge/git/learn-ocaml ((tags/v0.14.0))
$ git clone http://localhost:8080/OCO-SB1-B1A-YZG/learnocaml-workspace.git
Cloning into 'learnocaml-workspace'...
~/forge/git/learn-ocaml ((tags/v0.14.0))
$ cd learnocaml-workspace/
~/forge/git/learn-ocaml/learnocaml-workspace (master u=)
$ ls
demo.ml save.json
```

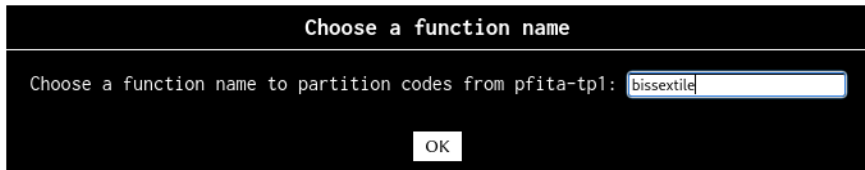
Support OCaml 4.12 (by Louis Gesbert)

- Support OCaml 4.12 (in learn-ocaml's build system *and* UI→toplevel)



Dissimilarity analysis (by Alexandre Moine & Yann R.-G.) I

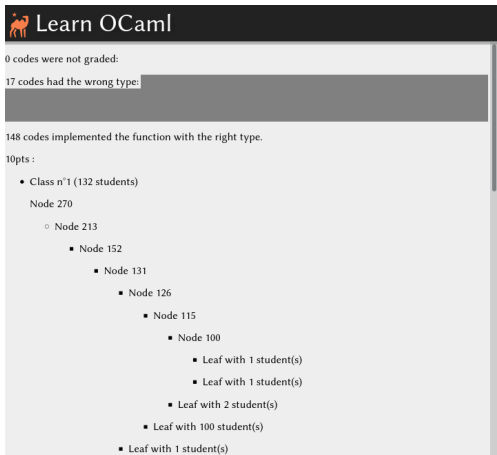
- so-called Learn-OCaml's **partition-view** feature
Teacher Dashboard → **middle-click** on an exercise (e.g. pfita-tp1)
- example input:



The screenshot shows a dark-themed dialog box with the title "Choose a function name". Below the title bar, the text "Choose a function name to partition codes from pfita-tp1:" is displayed. To the right of this text is a text input field containing the word "bissextile". At the bottom center of the dialog box is a white button with the text "OK".

Dissimilarity analysis (by Alexandre Moine & Yann R.-G.) II

- example output:



Learn OCaml

0 codes were not graded:

17 codes had the wrong type:

148 codes implemented the function with the right type.

10pts :

- Class n°1 (132 students)
 - Node 270
 - Node 213
 - Node 152
 - Node 131
 - Node 126
 - Node 115
 - Node 100
 - Leaf with 1 student(s)
 - Leaf with 1 student(s)
 - Leaf with 2 student(s)
 - Leaf with 100 student(s)
 - Leaf with 1 student(s)

Static binaries as release assets (by Louis Gesbert & EMD)

- For each release, static binaries are available for {Linux, macOS}:
<https://github.com/ocaml-sf/learn-ocaml/releases>

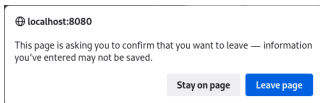
▼ Assets 9

| |
|--|
|  learn-ocaml-client-darwin-x86_64 |
|  learn-ocaml-client-linux-x86_64 |
|  learn-ocaml-darwin-x86_64 |
|  learn-ocaml-linux-x86_64 |
|  learn-ocaml-server-darwin-x86_64 |
|  learn-ocaml-server-linux-x86_64 |
|  learn-ocaml-www.zip |

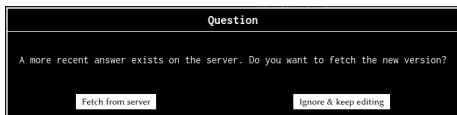
- `learn-ocaml-client`: CLI grader for **Tuareg+Merlin**, faster than JS
- `learn-ocaml`: CLI teacher tool to statically build and serve an instance
- `learn-ocaml-server`: server binary, faster than `learn-ocaml serve`

Fixing a long-standing issue (by Yann R.-G. & EMD)

- Offer better protections against "solution overwriting" (issue #316)
 - Mechanism 1: disable the automatic-implicit saving of the user's code at tab closing; ask for confirmation (OK: Firefox/Chromium/Safari)



- Mechanism 2: when the tab code has been unmodified for 3 minutes, check (upon next keystroke) if a more recent version exists, and ask:



- Mechanism 3: disable the Sync button when the answer is saved (avoid overloading the server with unneeded Sync requests and commits)



Agenda

- 1 Introduction
- 2 Some recent features in learn-ocaml ($\leq 0.14.0$)
- 3 Upcoming features

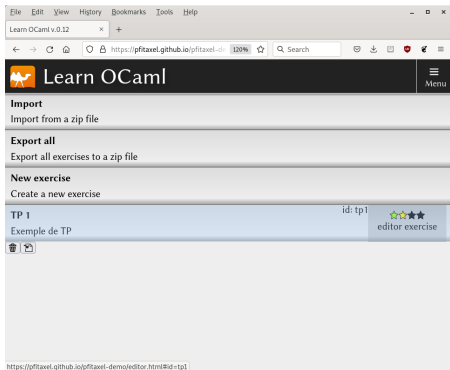
learn-ocaml-editor (by IRIT interns & EMD) I

Objectives:

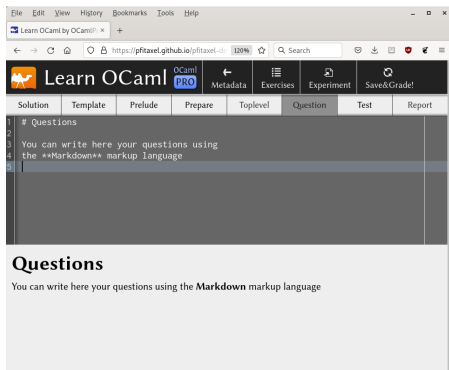
- ease the design of exercises
- reduce the "development loop" time

Links:

- Pending PR: [#295](#)
- Preview: <https://pfitaxel.github.io/pfitaxel-demo/>



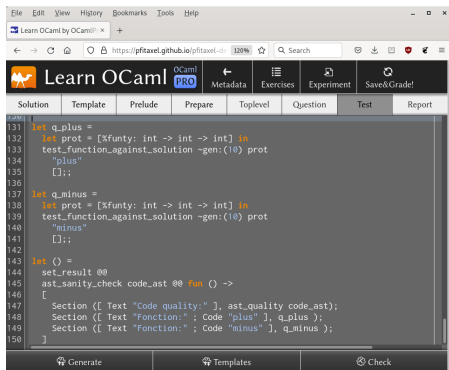
learn-ocaml-editor (by IRIT interns & EMD) II



The screenshot shows the Learn OCaml editor interface. The top menu includes File, Edit, View, History, Bookmarks, Tools, and Help. The browser address bar shows the URL <https://pftazel.github.io/pftazel-d/>. The main toolbar contains icons for Metadata, Exercises, Experiment, and Save&Grade!. Below the toolbar is a navigation bar with tabs for Solution, Template, Prelude, Prepare, Toplevel, Question, Test, and Report. The 'Question' tab is active, displaying a text editor with the following content:

```
1 # Questions
2
3 You can write here your questions using
4 the **Markdown** markup language
5
```

Below the text editor, the heading **Questions** is followed by the instruction: "You can write here your questions using the **Markdown** markup language".



The screenshot shows the Learn OCaml editor interface with the 'Test' tab active. The text editor displays OCaml code for testing functions:

```
131 let q_plus =
132 let prot = [%funty: int -> int -> int] in
133 test_function_against_solution ~gen:(10) prot
134 "plus"
135 [];
136
137 let q_minus =
138 let prot = [%funty: int -> int -> int] in
139 test_function_against_solution ~gen:(10) prot
140 "minus"
141 [];
142
143 let () =
144 set_result @@
145 ast_sanitary_check code_ast @@ fun () ->
146 [
147 Section ([ Text "Code quality:" ], ast_quality code_ast);
148 Section ([ Text "Fonction:" ; Code "plus" ], q_plus );
149 Section ([ Text "Fonction:" ; Code "minus" ], q_minus );
150 ]
```

At the bottom of the editor, there are buttons for Generate, Templates, and Check.

learn-ocaml.el *a.k.a.* learn-ocaml-mode

- by Manuel Cabarcos-Baulina, Louis Ayroles, EMD
- Separated software project:
<https://github.com/pfitaxel/learn-ocaml.el> (MIT license)
- Emacs/Tuareg front-end that is integrated **in the MELPA distribution**:
<https://melpa.org/#/learn-ocaml>
- Use `learn-ocaml-client` (whose dev. was started by Louis Gesbert)
- Advantages:
 - a standard OCaml **IDE UX** (using **Merlin+Eldoc**) for coding exercises
 - the students store their `.ml` files directly on their workstation
 - **grading** w.r.t. a Learn-OCaml server (TOKEN or mail/passwd)
 - **very quick feedback** (no JS is involved anymore)
- Bottlenecks:
 - Pending PR: [#458](#) (lifts a limitation, paves the way to `.cmo` JS grader)
 - Needs a few small additions to `learn-ocaml-client`
 - & *Needs more users interested to test it!*

`use_passwd/use_moodle` (by IRIT interns & EMD)

- Strong authentication (by e-mail/password, that can be changed)
 - Add `"use_passwd":true` in `server_config.json`
 - SMTP protocol: send automatic mails for e-mail/password change
 - support in Tuareg/Merlin/learn-ocaml-mode : OK
- Moodle auth (Moodle admin rights unneeded, just teacher ones) :
 - Add `"use_moodle":true` in `server_config.json`
 - OAuth & LTI protocols: students authenticate via a click in Moodle
- Remaining tasks:
 - Migrate the `oauth-moodle` branch : OCaml 4.05 \rightarrow 4.12
 - Add the `irmin` dependency (requiring OCaml > 4.08)
 - Extend the documentation
 - Release `learn-ocaml.el` accordingly

Supporting Vg-based exercises in Learn-OCaml

- <https://opam.ocaml.org/packages/vg/>
- "Declarative 2D vector graphics in OCaml"
- Learn-OCaml support: on-going implementation by Étienne Marais, Émile Rolley, Yann R.-G.
- Bottlenecks:
 - Use the rendering engine of browsers?
Reimplement a rendering engine within OCaml?
 - Understand how to generate a test report that is as useful as possible for students?

Multi-part exercises

- (issue #331) Feature wish: Organize an exercise into sub-parts, where one can navigate using left and right arrows. . .
- (issue #395) Feature wish: Handle several different graders for the same exercise. . .
- Objectives:
 - Support these two feature wishes in one go
 - Make design choices with a special focus on **compatibility** (the implementation code of a single-part exo should be kept as is; a multi-part exo should be partly rendered by a non-multi-part-aware server)
- Milestone:
 - The design has been completed and is satisfactory: PoC devised by Yoan Mollet (former L3 intern at IRIT), EMD, YRG.
 - To do: full-stack implementation (JS, backend, learn-ocaml-mode): **M1 intern at IRIT** (May-July 2022)

Teacher feedback stream

- Observation as a teacher: sometimes it is annoying to **note some bug** in the student's code (from the Teacher Dashboard) **and be stuck to give any feedback from the webapp** (read-only view)
- Many ideas would be possible, but constraints:
 - it should be simple enough to implement
 - it should be compatible with all frontends (including learn-ocaml-client's REST API → no web-sockets)
- **Yann R.-G.** is working on a PoC related to the following idea:
 - create a "teacher feedback stream" gathering comments + excerpts of the student's code (a bit like "GitHub's PR diff comments" ?)
 - which can be fetched by the student anytime, and marked as read.