D09 - Ruby on Rails Training

Real Time

Summary: In order to make your pages more fun and the UX more pleasant, you need to work on the client side, but it doesn't know Ruby. This is why you will have to use JavaScript.
*The first part of the day will aim to make the **`CRUD`** more dynamic with some **`AJAX`**. The second part will be about multi-user real time with WebSockets provided through ActionCable (a new feature in Rails 5).*

Contents

Chapter I

Preamble

Since we're going to deal with real time and JavaScript, I've made a list of games in JavaScript:

- Wipeout-like
- Text based rpg
- the ultimate mouse killer
- Oldschool Survival Rpg
- Puzzle game
- co cow coW cOW COW

Chapter II

Ocaml piscine, general rules

- Every output goes to the standard output, and will be ended by a newline, unless specified otherwise.
- The imposed filenames must be followed to the letter, as well as class names, function names and method names, etc.
- Unless otherwise explicitly stated, the keywords `open`, `for` and `while` are forbidden. Their use will be flagged as cheating, no questions asked.
- Turn-in directories are `ex00/`, `ex01/`, . . . , `exn/`.
- You must read the examples thoroughly. They can contain requirements that are not obvious in the exercise's description.
- Since you are allowed to use the `OCaml` syntaxes you learned about since the beginning of the piscine, you are not allowed to use any additional syntaxes, modules and libraries unless explicitly stated otherwise.
- The exercices must be done in order. The graduation will stop at the first failed exercice. Yes, the old school way.
- Read each exercise FULLY before starting it! Really, do it.
- The compiler to use is `ocamlopt`. When you are required to turn in a function, you must also include anything necessary to compile a full executable. That executable should display some tests that prove that you've done the exercise correctly.
- Remember that the special token `";;"` is only used to end an expression in the interpreter. Thus, it must never appear in any file you turn in. Regardless, the interpreter is a powerfull ally, learn to use it at its best as soon as possible!
- The subject can be modified up to 4 hours before the final turn-in time.
- In case you're wondering, no coding style is enforced during the `OCaml` piscine. You can use any style you like, no restrictions. But remember that a code your peer-evaluator can't read is a code he or she can't grade. As usual, big functions are a weak style.
- You will NOT be graded by a program, unless explictly stated in the subject. Therefore, you are given a certain amount of freedom in how you choose to do the

exercises. However, some piscine day might explicitly cancel this rule, and you will have to respect directions and outputs perfectly.

- Only the requested files must be turned in and thus present on the repository during the peer-evaluation.
- Even if the subject of an exercise is short, it's worth spending some time on it to be absolutely sure you understand what's expected of you, and that you did it in the best possible way.
- By Odin, by Thor! Use your brain!!!

Chapter III

Today's specific instructions

You must use `rails 5` all day long. Gems facilitating the AJAX (or just making it for you) are prohibited.

Non-exhaustive list:

- "best_in_place"
- "better-edit-in-place"
- "super_inplace_controls"
- "rest_in_place"
- "on_the_spot"
- "edit_mode"
- "best_in_placeish"
- "crest_in_place"
- ...

You will have to justify the use of any non rails-built-in Gem during the evaluation if your assessor requires it.

`ANY` additional JavaScript library is prohibited. Your `application.js` doesn't contain any of the following imports:

```
jquery
jquery_ujs
turbolinks
```

Chapter IV

Francis_1

	Exercise 00
Exercise 00:Francis_1	
Turn-in directory : *ex*00/	
Files to turn in : `Xnote`	
Allowed functions :	

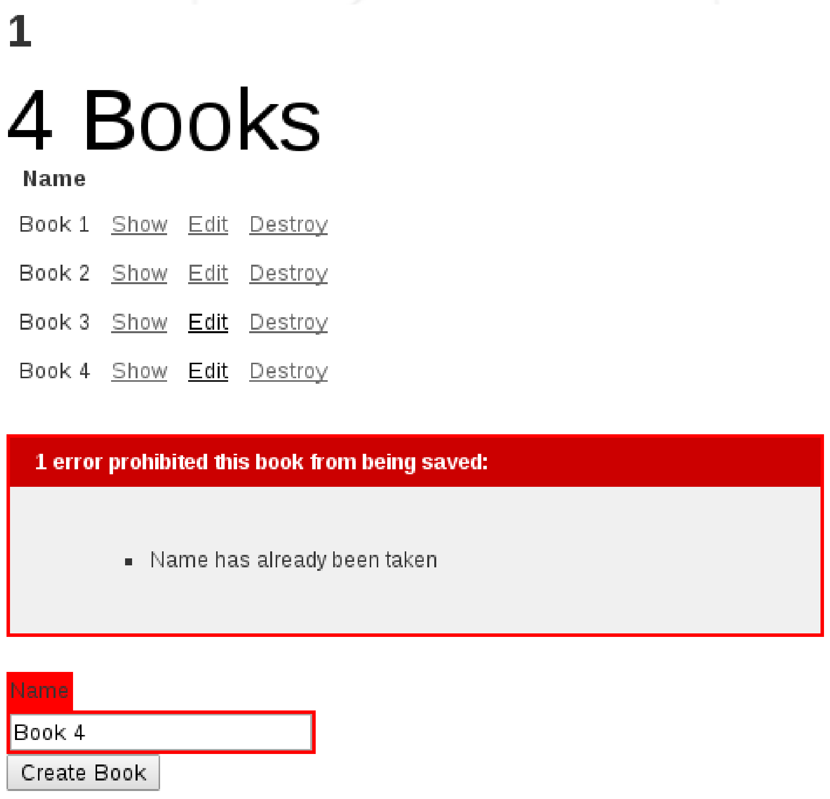
You must create a library listing books. The application is named "Xnote". For this exercise, a scaffold is enough:

```
rails g scaffold book name
```

Then you will:

- Create a unicity validation rule on 'name' of 'book'.
- Allow the add of books in AJAX:
  - the form will appear clicking the 'link_to' pointing on 'new_book_path'
  - the 'books' list data are updated when the form is submit (it must also appear)
- See errors. For instance, if you want to use a name that's already in use.

Example :


1

4 Books

Name			
Book 1	Show	Edit	Destroy
Book 2	Show	Edit	Destroy
Book 3	Show	Edit	Destroy
Book 4	Show	Edit	Destroy

1 error prohibited this book from being saved:

- Name has already been taken

And **everything** should work **without** refreshing the whole page. To verify that, you must put in your layout:

```
##ex00/Xnote/app/views/layout/application.html.erb:
...
<body>
  <% $refresh ||= 0 %>
  <h1><%= $refresh +=1 %></h1>
  <%= yield %>
</body>
...
```

A bit like a bug, your page will include a "refresh count" that must stick to `1`.

Chapter V

Francis_2

	Exercise 01
Exercise 01:Francis_2	
Turn-in directory : *ex*01/	
Files to turn in : `Xnote`	
Allowed functions :	

Now you know the drill, you will be able to do the same with the "link_to" pointing on the destroy method.
Clicking on it should, after a confirmation popup, delete the entry from the DB and update the list.

`the global variable must always be 1!`

Chapter VI

Francis_3

	Exercise 02
Exercise 02:Francis_3	
Turn-in directory : *ex*02/	
Files to turn in : `Xnote`	
Allowed functions :	

And now, this is the "edit" method's turn to work without reloading the page...

You must insert the form as the first line of the table matching the edited 'book' and allow errors to be displayed. When submitting a book already submitted, validation errors must appear.

`the global variable must always be 1!`

Chapter VII

Francis_4

	Exercise 03
Exercise 03:Francis_4	
Turn-in directory : *ex*03/	
Files to turn in : `Xnote`	
Allowed functions :	

Let the CRUD aside for this exercise.
In the header, create a count for the (total) number of books. Is must be updated each time the DB is modified, whether a book is added or deleted.

`the global variable must always be 1!`

Chapter VIII

ChatOne

	Exercise 04
Exercise 04:ChatOne	
Turn-in directory : *ex*04/	
Files to turn in : `Chat`	
Allowed functions :	

Until now, we only had to produce parts in AJAX, which is just a pattern. Now, let's head towards the multi-user together.

Make a chat application that includes user messages authenticated with the 'devise' Gem, that appear in real time to all the logged-in users. It will surprisingly enough be named: "Chat".

A piece of advice: ActionCable is very good at managing this. Use it!

`Open several windows in 'invisibility' mode and register with different logins.`

Design this application so it can deal with `a lot` of real-time traffic. You must implement a system that will set the tasks in buffer.

`the 'ApplicationJob' or 'Active Job' were not just made for our canid friends.`

Chapter IX

ChatTwo

	Exercise 05
Exercise 05:ChatTwo	
Turn-in directory : *ex*05/	
Files to turn in : `Chat`	
Allowed functions :	

Using the base of the application you've just created, implement the ChatRoom concept: rooms that will keep what comes in. You will make sure that as soon a user is logged in, they can create a chatroom.

This user is considered the sole creator of this ChatRoom. Deleting this user also deletes all the rooms they have created and the messages they include. Posted messages only appear in the room they were initially created in.

Chapter X

ChatThree

	Exercise 06
Exercise 06:ChatThree	
Turn-in directory : *ex*06/	
Files to turn in : `Chat`	
Allowed functions :	

Always in the same "chat" application, create a notification system represented in `Chat/apps/views/layouts/application.html.erb` by a list where an entry is added for each new message **if** the message's author is not the logged in user. It's just logic: you're not gonna get notifications for your own messages. This must be applied to all the chatrooms the user is logged in to.

In real time, you will always and on every page, display a list of notifications and a count of the total number. You can get `NO` notification of your own messages. This must work in multi-user and simultaneously.

As long as you're at it, why not adding a little sound to your notification.