Solitaire

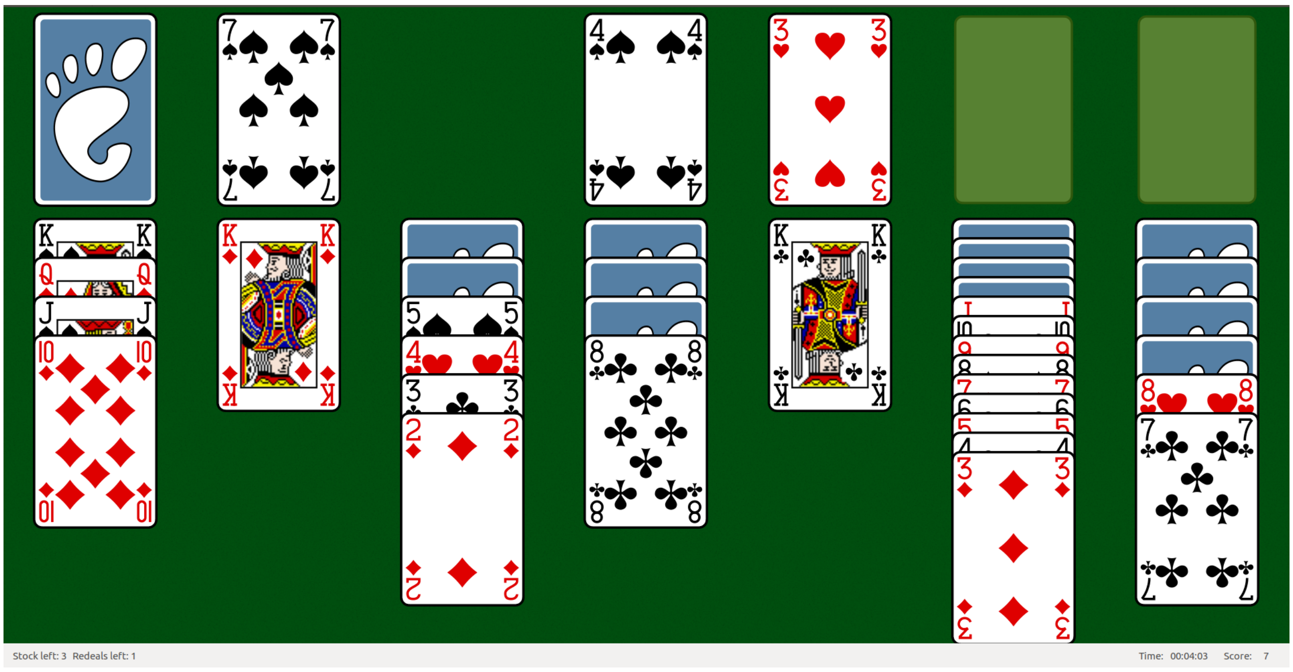
Solitaire online game by Michael Bausano.

Github

Link

How does Klondike work

Klondike is the most known of all solitaire card games. The goal is to sort all cards of the same suit on top right piles in ascending order. There are seven piles with a total of 28 cards in the beginning and a deck with 24 cards.



Project specs

My project is game web site including authorization, scoreboard, comment & help section and the game itself. Solitaire is very simple card game for one player with moves, move validation and multiple similar piles with important differences. This makes it ideal for my first project as I can demonstrate various programming ideas.

The project has been tested on **PHP 7** machine with **Chrome v60** and resolution of **1920x1080**.

It stands and falls with **.htaccess** and ModRewrite option.

If you would like to test the website with some dump data, please open `factory.js` file in root directory and a new anonymous window, copy the file code into your **DevTools** console and reload the page.

This report contains only the required minimum. However I would love to discuss my project more in the lab session.

Authorization

There is Login link in the navigation menu. If the user is logged in, it user gets redirected to a logout page. A guest gets redirected to a login & register page. Both forms have validation and if they enter invalid values, the site tells them what have they done wrong.

Related screeshots:

- auth
- logged
- login_fail
- register_fail

Help

This page includes hints for the game, brief history and a comment section. Only users can comment. You can delete your comment and your message has to be at least 10 characters long. You can also add tags to your message (up to 3, separated by space).

Error handling

All forms have validation functions and user feedback. If user enters an invalid URL, an error page with error message "Wups! This page does not exist :(" appears.

Scoreboard

Upon clicking the Highscore link in the menu, a simple table sorted by number of points is printed. If a current user has a score in the **TOP 10**, the line is highlighted.
The score is based on the time that passes since the beginning of the game and the number of moves a player did.

```
Score = 1000 - 2 * clicks - seconds
```

Related screenshots:

- scoreboard

Game

There are 13 piles I call **registers**. All registers extend *pack data type* class and are indexed. Current game is saved after every player move, so player can leave the a game and return anytime to continue.

Starting position

When the game starts, there are 24 cards in **D**[**0**] register. **F** and **D**[**1**] registers are empty. On each of **P**[**0** - **6**] registers, there are *x+1* cards and only the top one is revealed.
Available player actions:

- one in **D**[**0**] → **D**[**1**]
- ∀ in **D**[**1**] → **D**[**0**]
- { , } in **P**[**x**] ←→ **P**[**y**]
- one in **D**[**1**] → **P**[**x**], **F**[**x**]
- one in **P**[**x**] ←→ **F**[**x**]

The player's goal is to move the cards around the registers so that he manages to reveal all cards on the table. The game ends when all 52 cards are in **F** registers.

Timer

There is a timer running in the background. The timer triggers each second, decreasing the players score by one and printing it out.

Move validation

`Pack` class has an abstract method `validate` that is called in `push` method of main Vue component. This class is overriden in each register class by desired validation conditional.

Related screenshots:

- game_in_progress
- game_over
- new_game
- not_logged

Game development

I would like to say a few words about how my progress on the game went. Let me start with saying that having a game as a project task is very risky for people who tend to procrastinate a lot. I honestly believe that I have spent more time playing my game than coding it. Every time I had a new feature to implement in my mind, I would just start a new game and say "this is definitely the last try" to myself spending another hour playing Solitaire.
The project was done in 3 mayor runs. If you would like to see how the progress was done in more details, please see the project's GitHub commits.

First run

I had do to the structure, thus I did `Router` and `Blader` class. Afterwards, I created the layout, navigation and footer.

Second run

I started working on content. I started with Authorization, which was fairly easy to do and followed up with Scoreboard and Help section.

Third run

Last but not least was the game itself. I did all necessary classes: `Pile`, `Deck`, and `Flush` that all extended `Pack`. Afterwards I styled the layout. Next I had to do was solve event emitting and `push` method. When everything worked I did the game saving and loading.

Bugs

I have fixed all bugs I have found by endless hours of my playtime except one that I call rather an Easter egg or hidden feature than a bug. I am leaving it here for future myself to find.

External libraries and frameworks

Vue.js

I decided to go for Vue framework since I have most experience with it and I didn't want to use vanilla JS. For the 3rd Coursework I am thinking either Angular or React, still not decided really.

Font-awesome

I am loading more than 80 KB of information just because of 1 single icon. This was more of a recession on people using robust libraries on something that could be done way easier and data-wiser.

Bootstrap CSS

I didn't really use Bootstrap CSS framework either, could have omit it, because I did some really bad practice of combining Bootstrap Grid and Flex. Thing to consider in future.

Lodash

Amazing JS library which I absolutely adore that brings you tons of data type related helper functions.