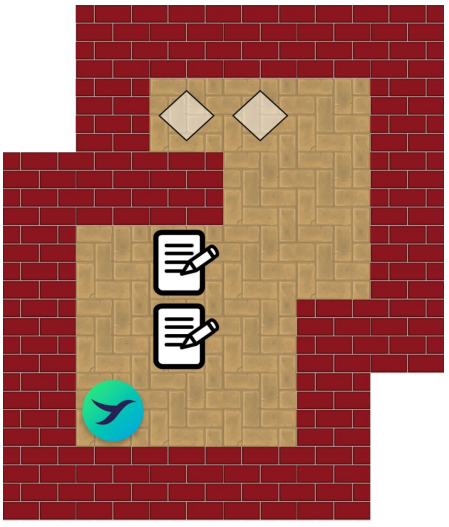
FormBird - EasySend Candidate Exercise

Hi there! Welcome to EasySend :)
Thanks for taking the time to do this exercise.
In this exercise you're going to play FormBird, a very addictive game! This document describes the steps for installing the exercise and the tasks you'll need to complete.

Installation

You'll need to clone the game's repo to your machine. Make sure to ask for permissions to access it. https://bitbucket.org/easy-send/form-bird-exercise
Follow the steps in the repo's README.md to install and run it.
It's best to run the exercise on Google Chrome, or any other evergreen web browser.

Game Goal



In FormBird, your goal is to move all forms:

To their destinations:

By pushing them with the player:

You can control your player by using the arrow keys of your keyboard.
Once all forms get to their destinations, the game is won (cheers!)

Exercise Tasks

You are provided with several tasks to complete. Each tasks requires you to make some modification to the game's source code. You're allowed to edit **all the source files** of the project, and introduce any external library you wish to use.
Here are the tasks:

Task 1: Move Left

The game's creators seem to have made an error that caused the left button not to work properly. Try to fix the bug to make sure the player can go left!

Task 2: Animations :)

When the player or a form moves, they do it with no animation. Add a nice movement animation.
Tip: Try to use only CSS.

Task 3: New Game Button

In this task you'll introduce the "new game" button - a button that starts a fresh new game with the initial game state (as described in game-state.js). Add the button to the main game page's UI below the 'moves' counter. There's no need to add a special design to the button.
Note: Try to think of a solution that doesn't require reloading the page (this causes the entire app to reload and is bad for performance 🤓). Instead, try to restore the game's state while the app is running 👏.

Task 4: Make isTileTypeAt more efficient

Check out the function isTileTypeAt in static/js/main.js. What does it do? Why is it not very efficient? Replace it with a more efficient implementation, you are allowed to create additional data structures if needed.

Task 5: 🔁 Undo and Redo Moves

Add an Undo & Redo buttons under the new game button (still no need for any special design for the buttons). These buttons allow the player to go back to previous moves if they think they made a mistake. Make sure to undo the entire game state, including the player's position, the forms' positions, and the moves counter.
It should be possible to use the undo button several times in a row to go back to previous game states.
Tip: Try to use that fact that the entire game state is saved in the gameState object.
There's no need to try to make this feature especially memory efficient.

Task 6: Save Game State Locally

When your game window closes, your entire game progress is lost, and that's a shame! Implement a solution to save the game's state after every move, so that the game's progress will be saved even after closing and opening the browser, or after refreshing the page.
In this task you are only required to save the progress for the same browser used to play the game in the first place.

EasySend

Task 7: Save Game State Globally

Sometimes saving a copy of your progress on the local browser is just not enough (i.e. your computer might get stolen by the cookie monster).
Make the game state persistent in a way that'll be stored in the server side so that every browser that connects to the website will see the same game progress.
Tip: You can use the placeholder game state endpoints at 'api/game_state_api.py'

Task 8: Undo & Redo Hit Again

Going back to the undo & redo task - how would you make your solution more memory efficient?

Task 9: Realtime fun!

In task this you'll need to add support for sharing the game state across different sessions in real time! That means two (or more!) players that open the game at the same time, should see exactly the same game state, and every move should reflected in realtime for all connected users.
Tip 1: You'll need to make changes to both the server and client for this task. You're also allowed to completely replace the server if you feel more comfortable with other programming languages / frameworks.
Tip 2: Use websockets to make the magic happen!

Task 10: Scaling Up

After your great success in all previous tasks, FormBird became a worldwide hit!
🚀🚀🚀 Thousands of players are playing it worldwide at any moment. We'll need to boost our server in order to keep up with the demand. Design an architecture for running the game server on more than one machine. The same game state should be shared even for users that use different servers.