Gregoire Sacha
Hai An Mai

"Illusion of Choice" Project Report

Our project consists of subpages, where for each page we focused on a different theme. The website includes subprojects: Conway's game of life, visualization of sorting algorithms, visualization of random event distribution, a typing game, and chess. For each project, we will explain our approach and problems we had in corresponding paragraphs. The chess subpage was mostly edited by Sacha, while the others were mostly edited by An.

Conway's Game of Life

This idea was inspired from the course CSE101, where we coded the rules of Conway's Game of Life. It was the first subproject that we worked on, which is why the JavaScript code is probably not the most optimal. The most important thing about this page was to encode the board and its cells. The Game of Life relies on checking each alive and dead cell and all of their neighbors, therefore we had to store each cell's coordinates. The next challenge was to correctly implement all the types of boards on which the game will be played on, and for that we used helper functions that were repeating to avoid the same code as much as possible. The last challenging part here was to implement a sleep/wait function, because JavaScript does not have a built in sleep function. We managed to find the correct line of code online, but it was a bit annoying that there wasn't a simpler way to do this.

Visualization of sorting algorithms

The first thing we did was implement a randomizer of columns. In JavaScript, there is no built-in random integer generation and shuffle a list function (like in Python), thus we had to implement it by ourselves. The second problem was the idea of switching two blocks. We struggled a bit on that, but in the end we managed to create a function that switched two blocks (with their values and corresponding objects). The last thing we needed to implement was sorting algorithms, which we copied most of the algorithm from this website. We realized that it would be time inefficient to code all the algorithms from scratch. Also, one of the biggest problems we had for this subproject was again a sleep function, because this time we were doing a lot of comparisons and changing elements (from a list), we couldn't use the same method as in the **Conway's Game of Life, so** we had to use async functions and await calls, because otherwise the code would calculate something beforehand, which we did not want to.

Visualization of random events distribution

This idea was motivated by the Central Limit Theorem. At first we were thinking about how we could visualize the number of occurrences nicely on a graph. Then we found out about JavaScript Graphing Libraries, which was the exact thing we were looking for. This made the subproject much easier. The only challenge we had was to simulate a random event (sum of dice), which was not very difficult, as we used the same idea of random function as in **Visualization of sorting algorithms**. For this subproject, we also added some extra functions, like changing the types of dice and also a possibility of changing probability distribution of each side of the die, which was not very hard to implement.

Gregoire Sacha
Hai An Mai

Typing game

Inspired by Monkey Type. For this idea, we needed some API to generate random words for us. Fortunately we found an API: Random Words, which greatly helped us on the main part of this project. A couple of challenges we had on this were checking if the typed character was correct, changing to the next word, and deleting the progress. At first, we wanted to change the span of words every time a character was typed, but we quickly realized that it would get very complicated when the user would type a lot of wrong characters. Thus, we referred to creating a div for each letter, that way we could easily track if the character was typed correctly or not. We also added a class *active word,* which helped us track the current progress. With that, implementing the deleting a character part was not very difficult. We wanted to also implement a cursor line that would follow the newest character, but it was too difficult and time consuming, so we decided to abandon this idea.

Chess.come

This subpage does exactly what you would expect it to; it allows you to locally play chess. Some of the more intricate rules of chess, such as castling, checks, and en passant, were not coded since this would have been time consuming, and did not seem like the main point of this project (something not extremely difficult nor impressive, just annoying to implement).

Some features:

- Movements are allowed through either clicking or dragging pieces.
- The selected piece is highlighted in green, while legal moves are highlighted in red.
- The board flips after every move to make it easier for both players to play.
- A tab with the history of all moves is available on the right hand side.
- A restart button is available on the left to restart the game whenever you want.
- Once the game is over, the winner will appear on the left tab (below the restart button). This will not end the game, since some pretty fun things can happen if the players decide to continue playing (with their own rules, of course).
- A couple of secrets that you will have to discover on your own by reading the code (these are not intended to be found by players, although it would be cool if they did find them).

Some of the struggles:

- Cropping the picture of the pieces. I spent at least 2 hours trying to do it in CSS or JS, just to end up failing to get it to work and spending another 30 minutes cropping them by hand.
- Getting the divs to appear in the right place. I have only tested this on a few computers, so I hope it works on yours.
- Remembering all the rules of chess, all the functions I defined, what was already implemented in the code, etc. I have never worked on a coding document as long as this one, so I suppose it was inevitable for me to struggle.

Gregoire Sacha
Hai An Mai

Homepage

We also added an icon which returns back to the homepage. The icon is included in each subpage. We also added a sliding animation for each selection, for which we used display: flex. It was a bit difficult to get the final result, but in the end we got what we wanted.

Note: Unfortunately our website is not very compatible with smaller devices. This would have been a disaster to implement, but it does mark something we could work on in the future to make our website even better.
We hope you enjoy our project!