Game Of Dragons

Semesteroppgave 2 by Bjørn-Ivar Skuggen

In this assignment we got the task to create a board-game using HTML/CSS and JavaScript. The task was to create a logo, tokens for the players, design a landing page, board-game page, and a winner page.

**Logo** (seen at top of document)

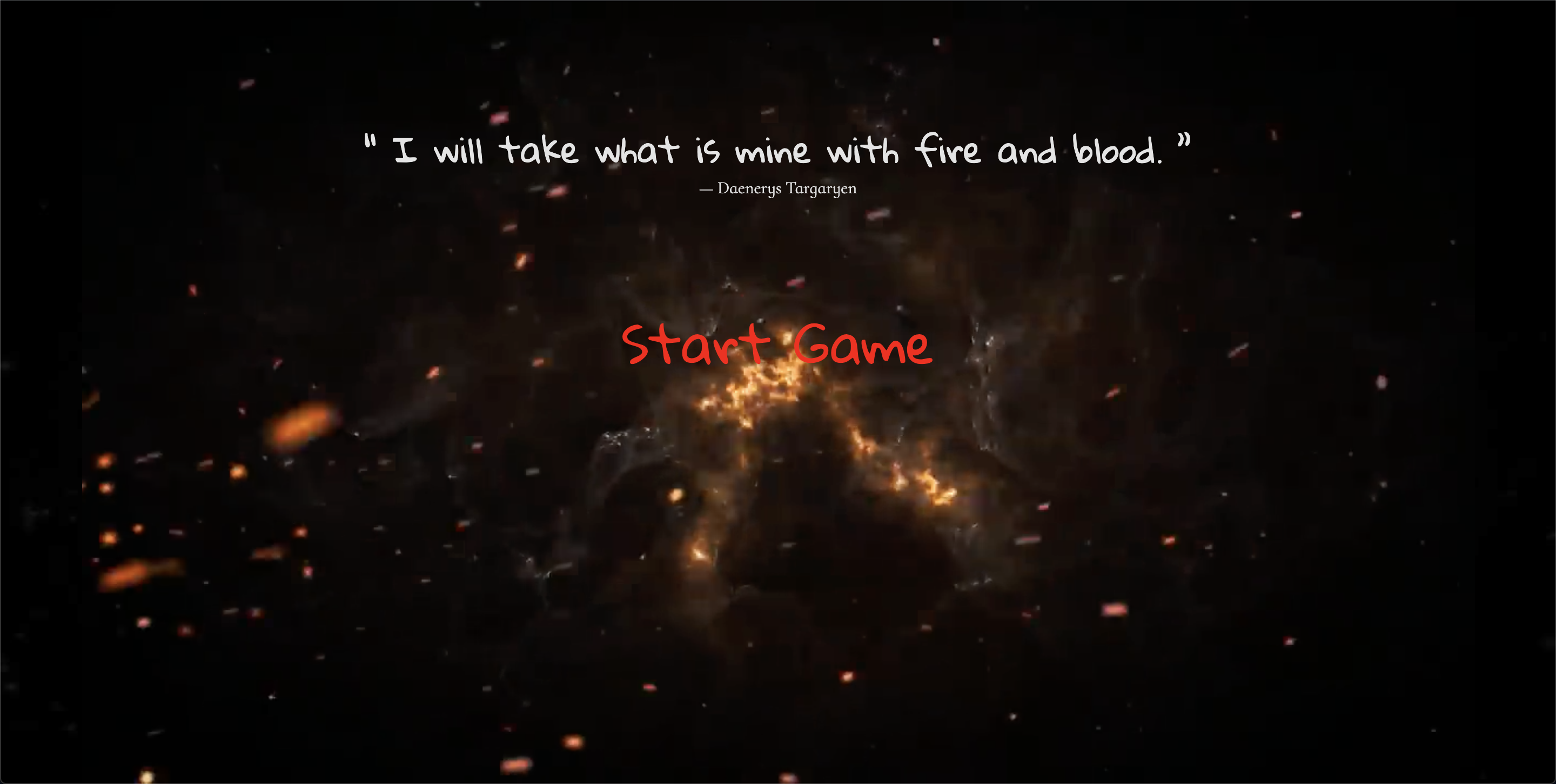
Since i named my game “Game Of Dragons”, the logo had to represent that. I decided to go with two dragon wings surrounding the name. The wings are also in a cartoon-look to match the tokens that will be in the game. The red in the logo is to represent the danger of the dragons.

**Tokens / Characters**

The characters are chosen from the provided API. To create the tokens for these characters, I started off by finding an image that would fit my game well. I then gave the whole image a cartoon effect, cropped out the face, created a black/white layer, masked out the background, and finally adding the gradient background. Some tokens also got a little makeover by the brush tool. This way I got to use several of the techniques from Design 2. I created them looking like cartoon characters to make the game more childish and not so serious, but also so I could make characters that were recognisable from the series.



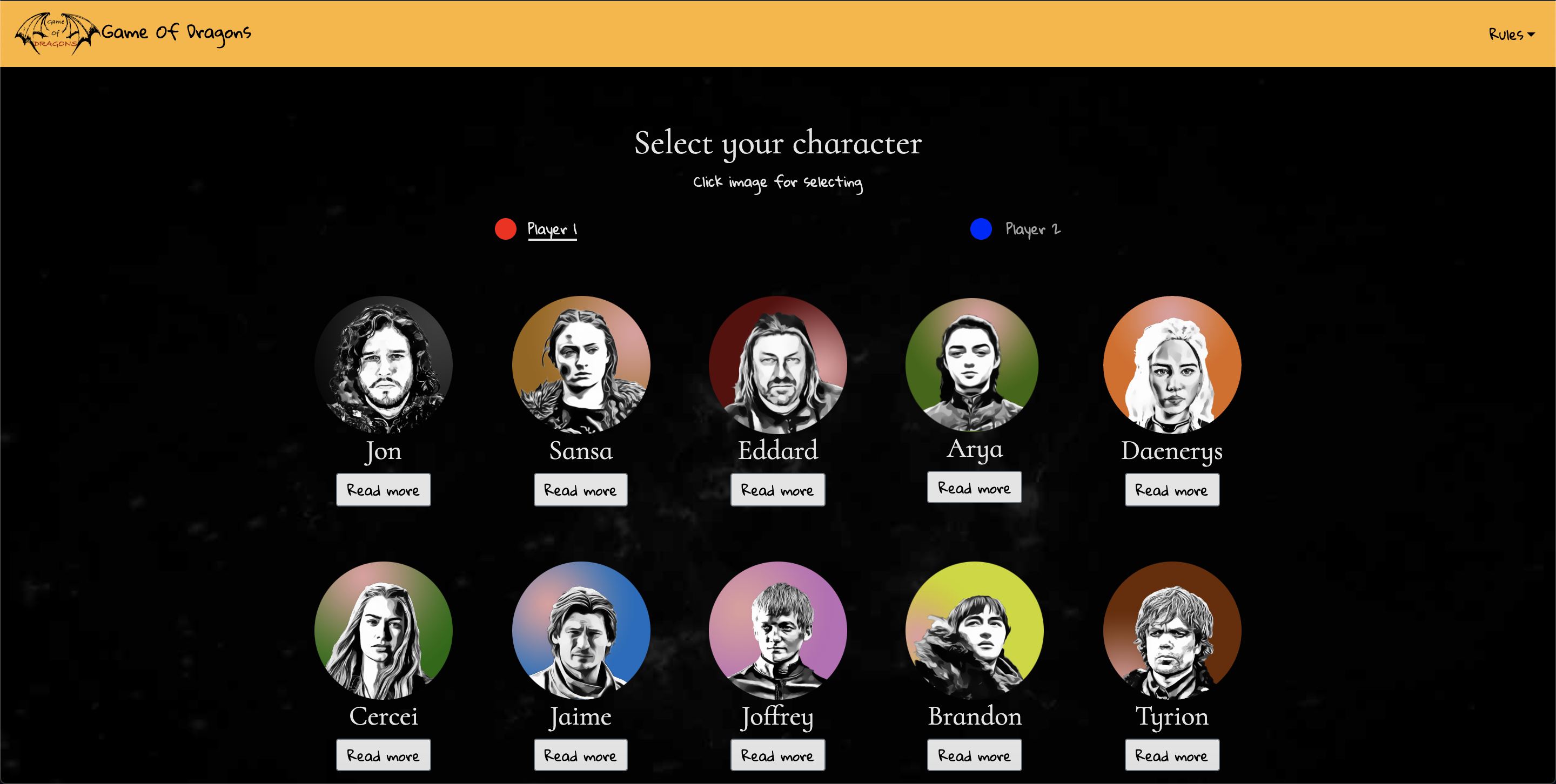
**Landing page**

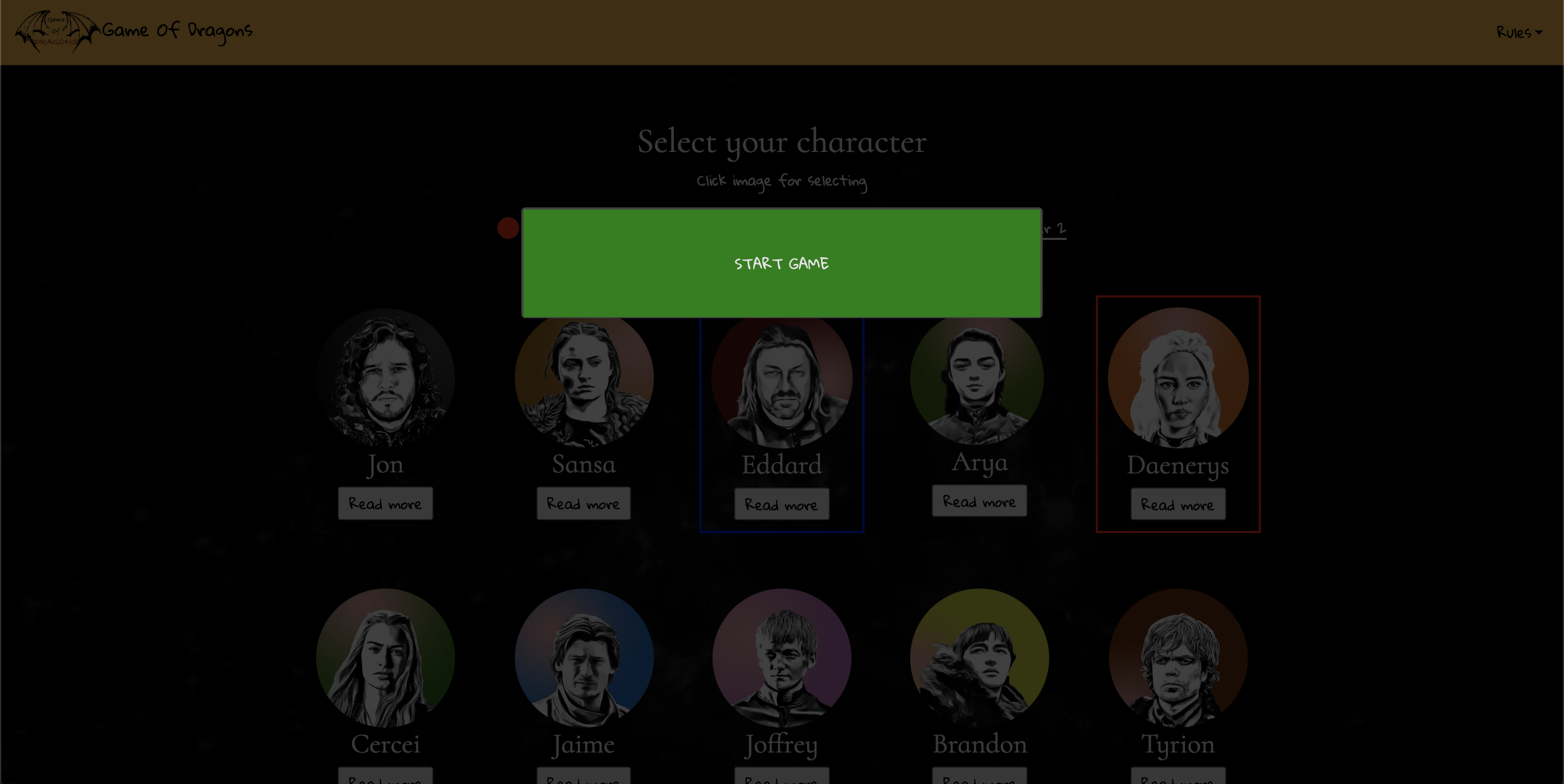
The landing page is pretty simple. To give the site a little suspense from the start, the only thing that shows for the first four seconds is a quote by Daenerys Targaryen on a dark video background of sparks and flames. After four seconds, the “Start Game” button appears, and the game is ready to begin.

**SelectCharacter page**

In the SelectCharacter page, all you and your opponent have to do, is to select your character for the game. When clicking “Read more” you can read the information provided from the API in a pop-up modal. To select the character, the user clicks the image , and are presented with the option to confirm or deny their choice. If user clicks “Nope” they get to choose another character, if they click “YEAH” the character is sent to sessionStorage and it is players 2s turn to choose.

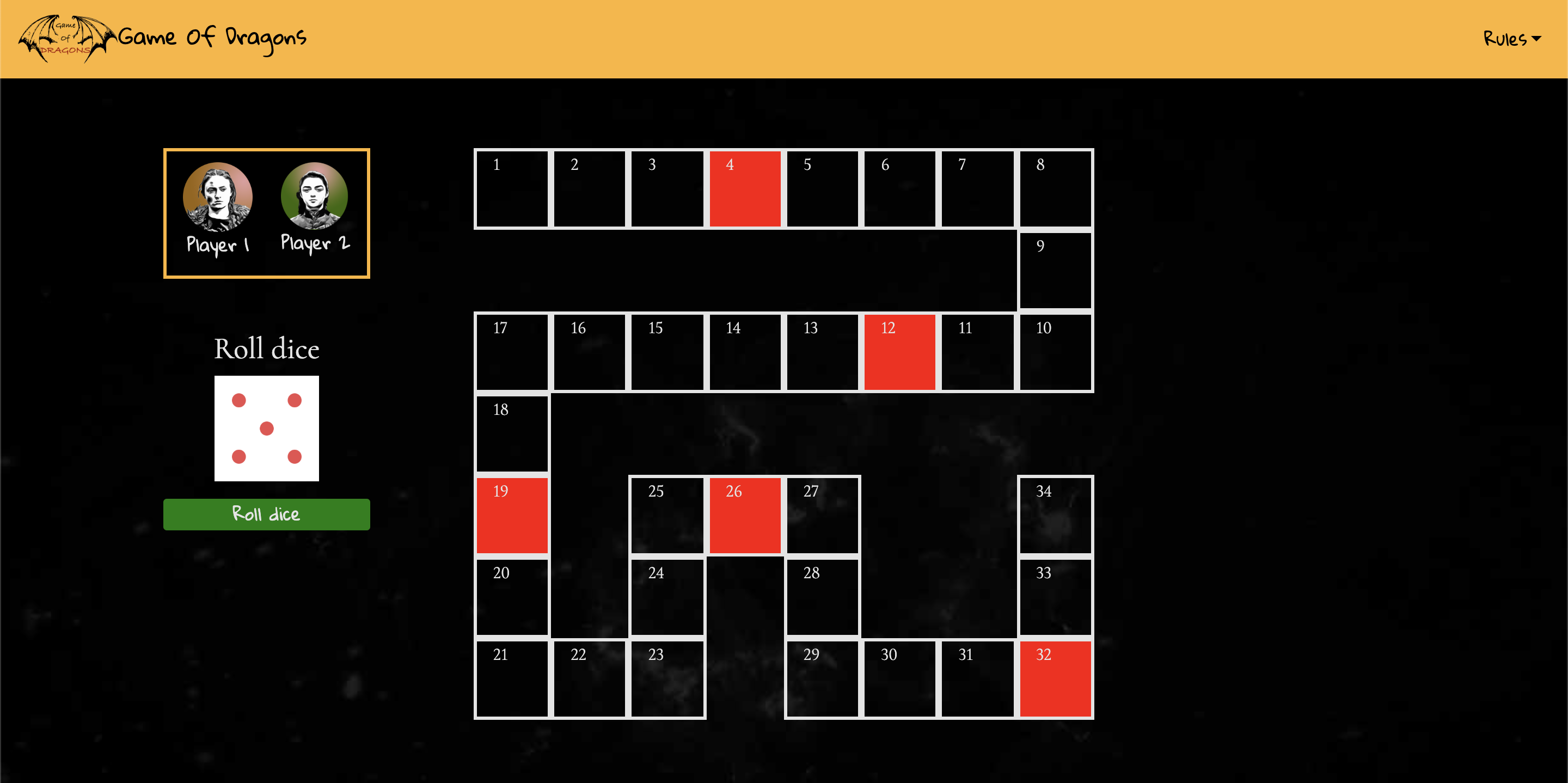
The layout is simple: Image —> Name —> Read more.

The user should never be in doubt of which character they are looking at.

When both players has chosen, the info is passed to the sessionStorage, and the “Start Game” button is displayed.

**Game page**

The game page is where the fun starts. The players roll the dice, the characters move, they hit traps, and there will be a winner. The layout of the site is made to make the user understand the path of the trail and understand where the goal is. The traps are easily spotted, and a red modal pops up when someone hits them. The tokens moves around the board, and when one of them reaches the goal, the winner modal pops up. I chose a modal instead of a new site because it fits better in my whole site as I am using modals for most of the happenings throughout the site.

The board itself, and the dice has sharp edges and are square unlike many other components in the site. This is done to create some contrast, and not make everything look so alike. The visual look is also more pleasing when the tiles are put together without any distance between the edges.

All the sites are created so you can tab your way through the site. Also WCAG tests has been done.

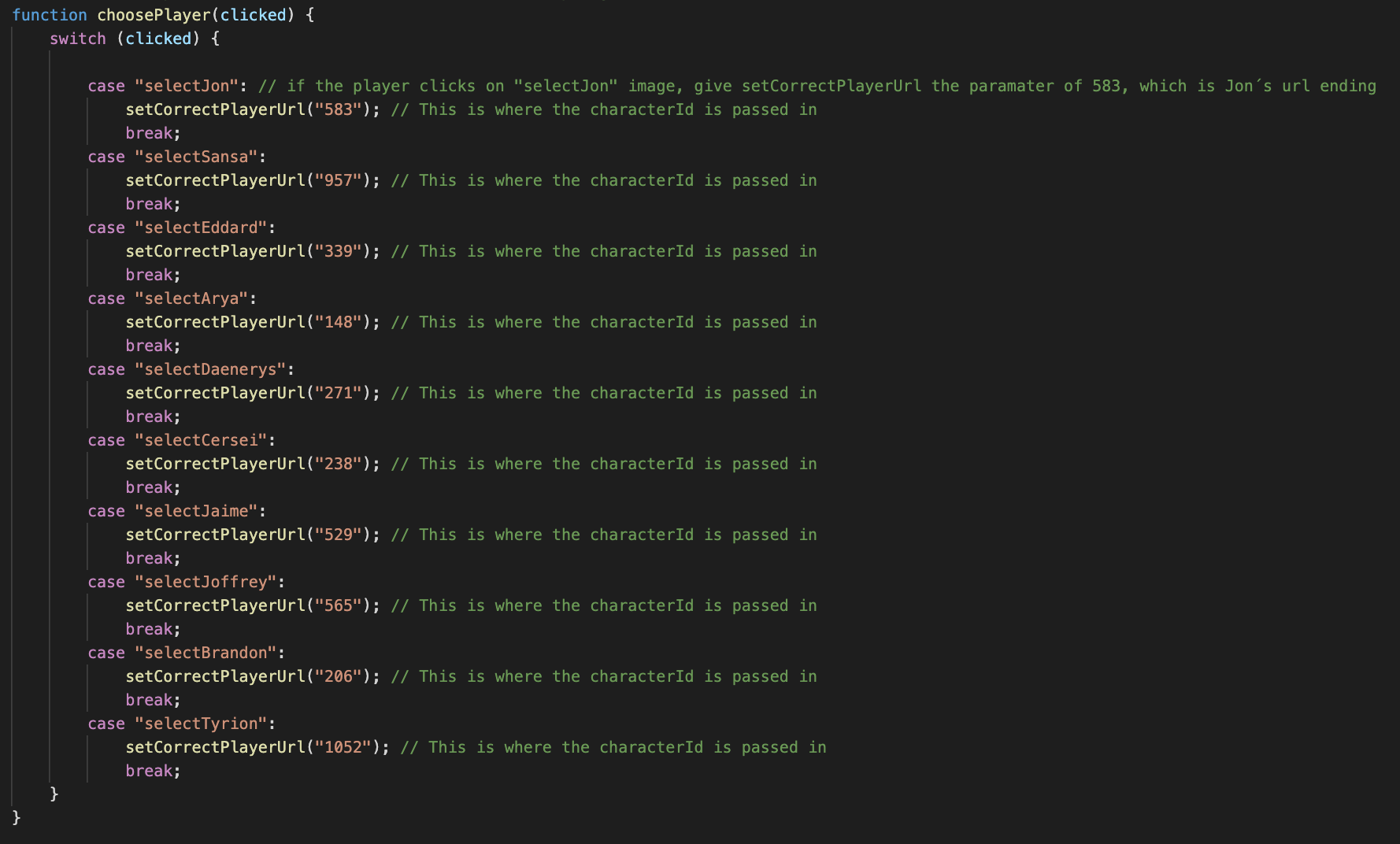
**The code**

When I first started this project, I got some fast results because of shortcuts and going for solutions that “just works”. After about two weeks in, the easy solutions and quick fixes finally caught up to me. Unstructured and unreadable code made the whole project fall apart as I could not find a way to make it work. I therefore had to start all over again.

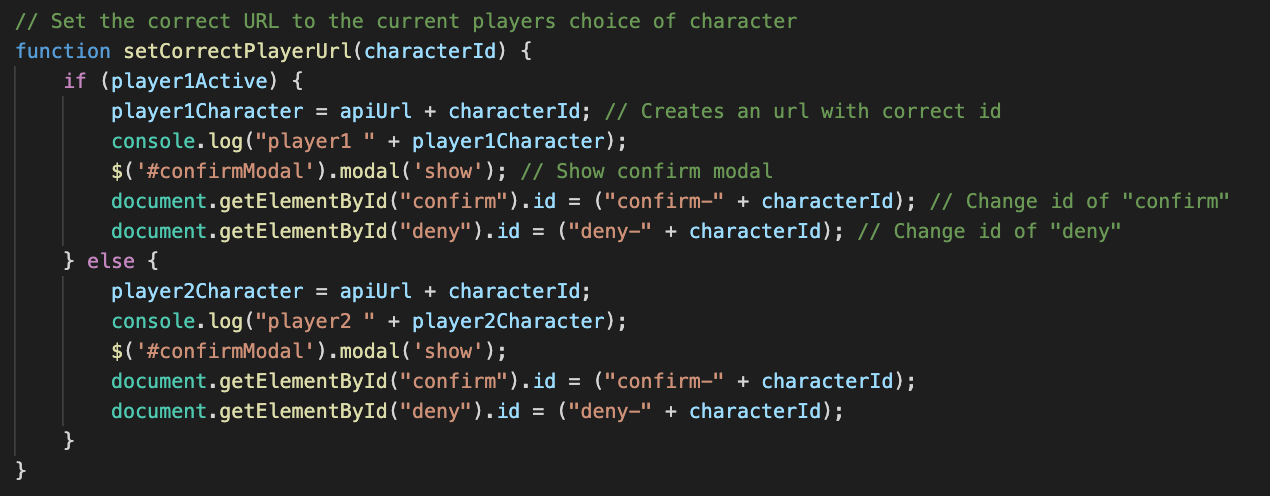
As you can see, the old code contained functions inside functions. It all turned out to be a mess.

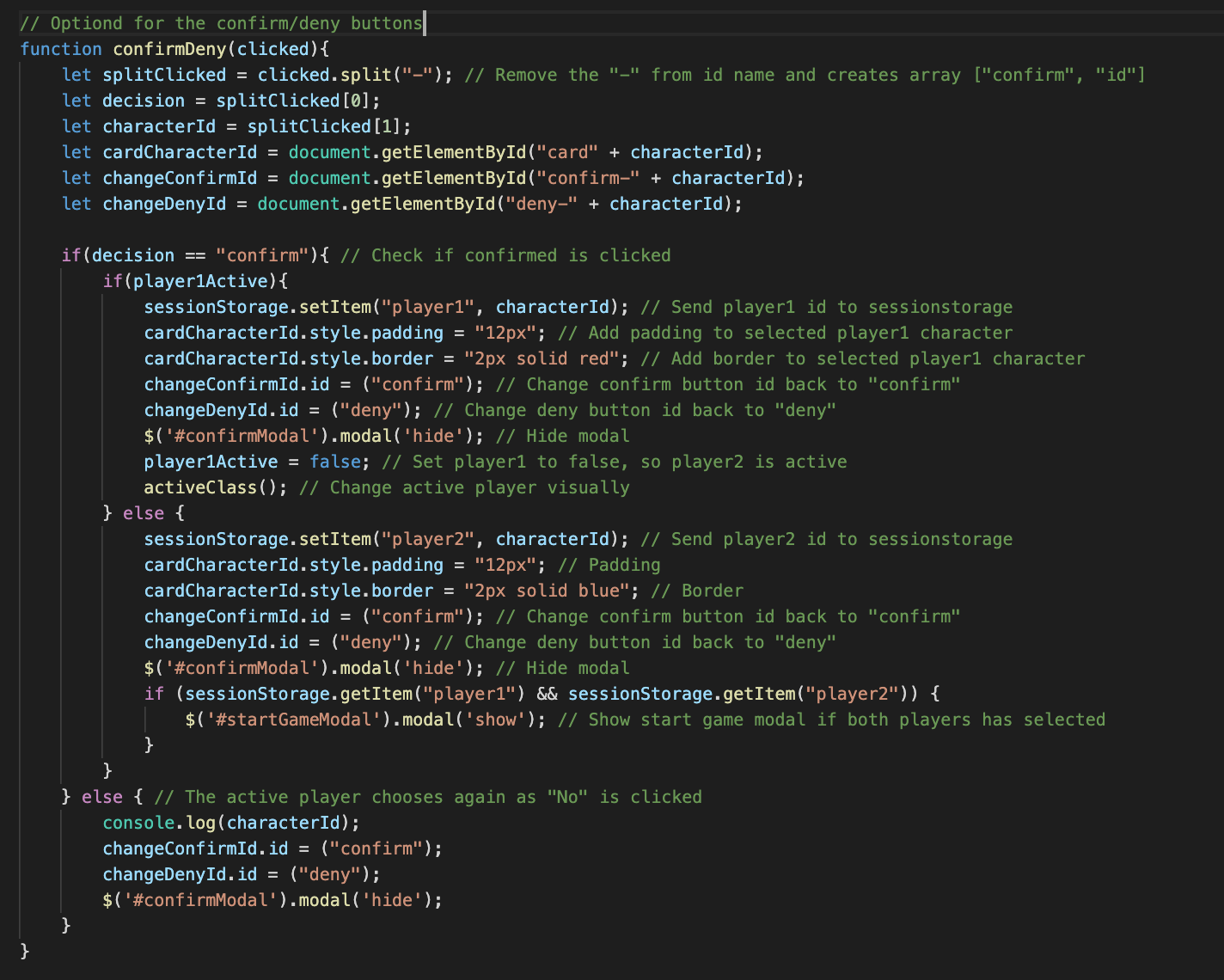
On my new try, everything started off a little better. I now had the understanding of how to structure it, and make the code understandable. I kept some of the ideas from the first code, but made sure it would make sense to use it.

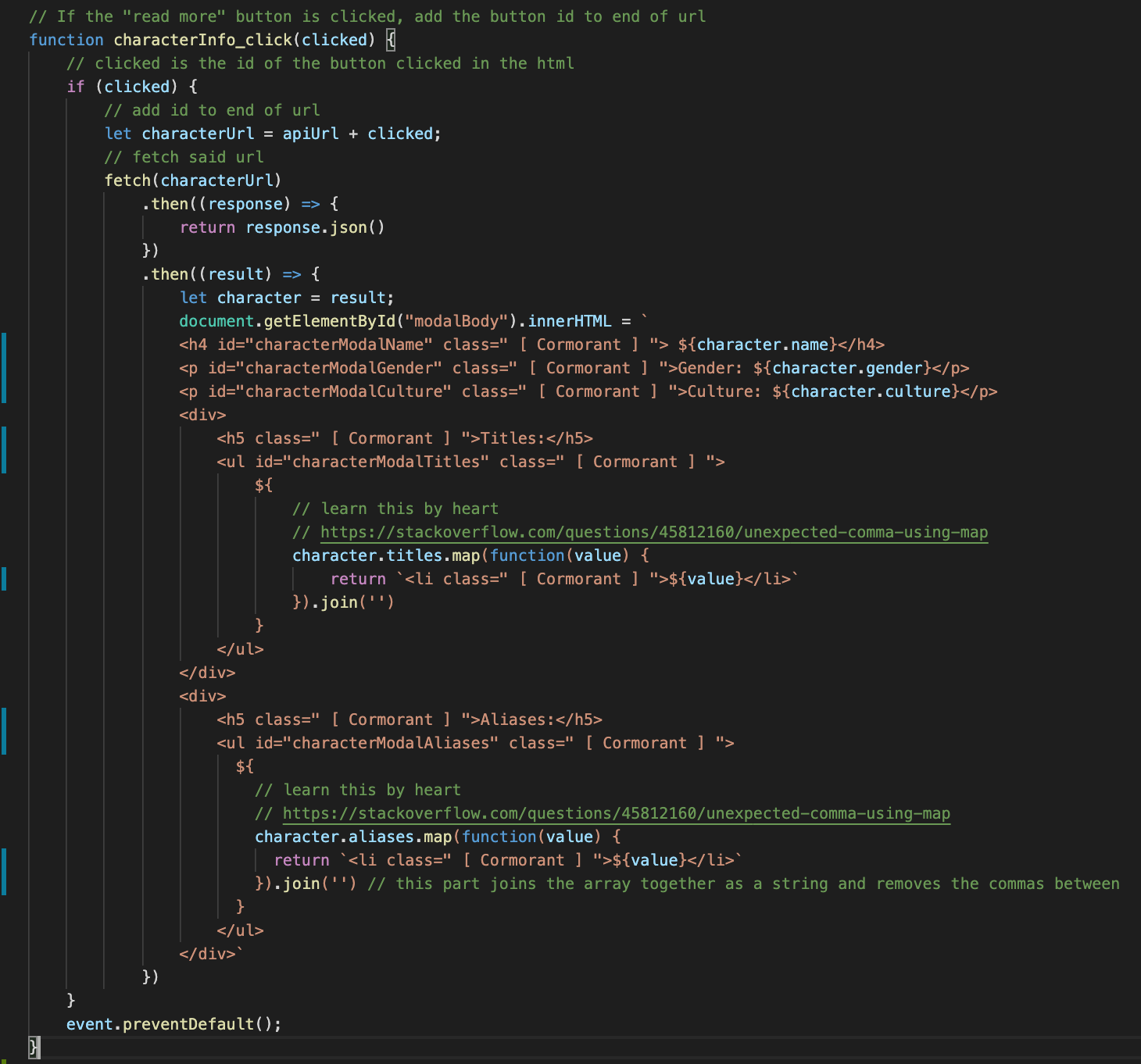
**Selecting the characters**

To select the characters, I decided to use a switch statement for this. That is because i can create a case for each image clicked, and pass in the characters url ending to the function setCorrectPlayerUrl. I will then be able to pass the info further into sessionStorage at a later point.

The characterId from choosePlayer is passed into setCorrectPlayerUrl, and here I display the modal with the confirm or deny options.

When the modal is displayed, the user have to make a final choice; to accept the current player, or change to another one. The click of either of the buttons trigger the confirmDeny function. If the user clicks confirm (YEAH), the characterId is sent to sessionStorage, and some styling is added to the chosen character. It also changes player if confirm is clicked. If deny (Nope) is clicked, it removes modal and gives the player to choose again. The reason of doing it this way, is so the user can decide again if he regrets his choice.

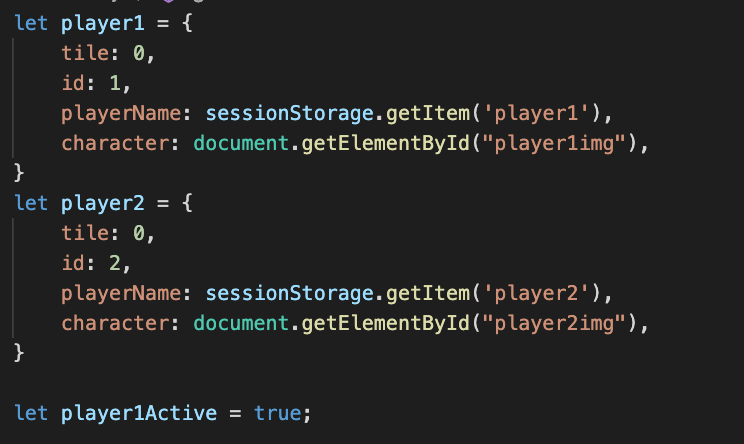
**Read about the characters**

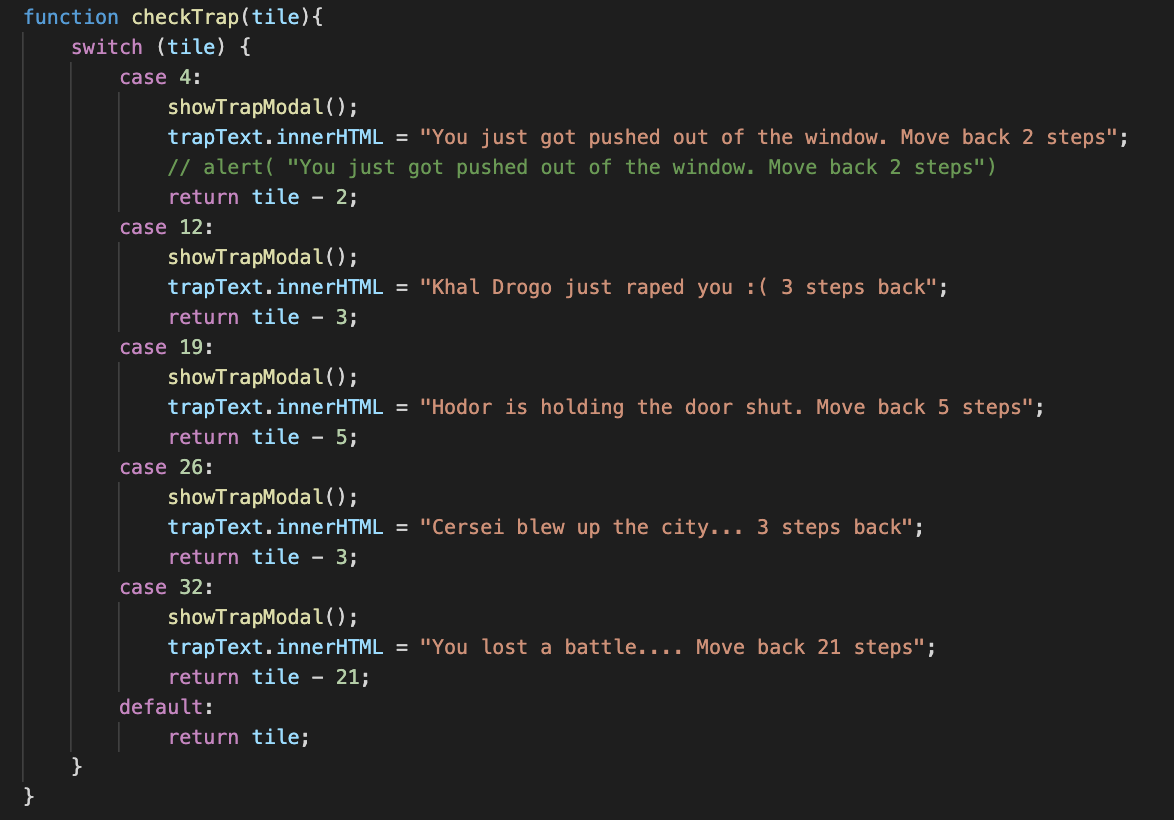
When the user clicks the “Read more” button, the url ending of the api link is added, and a fetch with the characterUrl is run. The result, which is converted to json, from the API is then added to the DOM, and the user can read a little about the characters.

When both users has chosen a character and the info is sent to the sessionStorage, the users are directed to the game site.

**The game**

I started off by defining the players ; player1 and player2. These variables contained an object with the properties tile, id, playerName, and character. The tile property will be changed throughout the game, as every dice roll will affect this value (if the player is active of course). I also created a variable with a boolean to check which player is active or not.



To run the game, everything depends on the game function. This function is triggered on the “Roll dice” button in the DOM. Every time the dice is rolled, the active player gets the dice value added to their tile property. If player.tile is equal to a trap, the checkTrap function is triggered. In this function. I also decided to go for the switch statement for this. That is because i can have a case for each trap, so if the player hits the same tile as the case number, the function triggers, and the user is warned with a modal and is sent back as many steps as the trap function returns.

The rest of the game function are based on if/else statements, and triggers functions if there are matching info.

**Conclusion**

This was a challenging, frustrating, but also fun project. I started off good, but met the wall halfway in. This gave me a setback, but also the opportunity to do things the correct way without any shortcuts and quick fixes. I am happy with having a working game, but if i had not made a mess in the first place, I think I would have a much better result in both design and functionality throughout the game.