JavaScript assignment (home project)

Submit Assignment

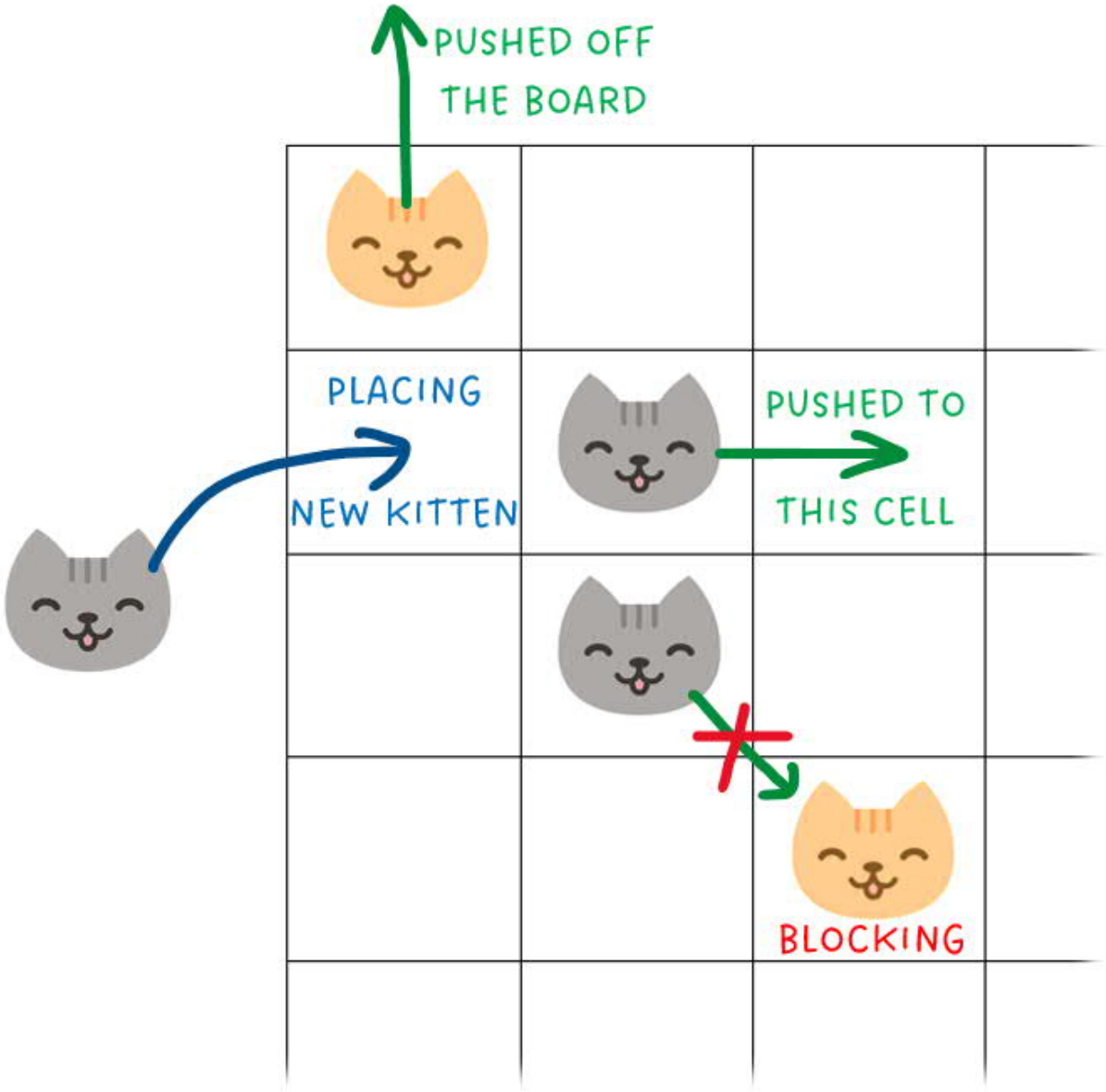
Due Sunday by 11:59pm **Points** 20 **Submitting** a file upload **File Types** zip
Available until May 14 at 11:59pm

Once upon a time, there was a princess in Nowhereland who really loved kittens. One day the kittens of the royal court sneaked into the beautiful young maiden's bedroom and started playing with each other on the princess' comfortable mattress. Her bed was very flexible, which created another problem: the jumping kittens kept bumping into each other and pushing their furry little playmates.

Task

Your task is to create a **simplified version** of the board game *Boop* as a browser-based application written in native JavaScript. To understand the task, you may take a look at the **rule book of the original game (https://www.smirkandlaughter.com/_files/ugd/693f33_5cd5b748ac194385a4fac5fb168076e7.pdf)** — just don't forget to add the scoring system and ignore the special rules about (big) cats.

- The game board consists of a **6x6 grid of square** cells.
- The game is played by **two players** against each other, who take turns placing kittens on the field. The kittens of the two players must be different, e.g. in their colour or pattern.
- The players have **8 kittens** each, which are initially placed on the **bench** outside the game board.
- Placing a new kitten on the board pushes the kittens (regardless of whether the kitten is yours or the oppenent's) in all adjacent cells **one cell away** from the location of the placement. (See image below!)
- There is only one case when the kitten is not pushed away — if in the cell where it would arrive is already taken by another kitten.
- The kittens on the edge of the board **are also pushed away** — they fall off the board and return to the bench of the player who placed them.



- The goal of the game is to have three kittens of the same color (belonging to the same player) next to each other horizontally, vertically or diagonally **after the pushing finishes**. In this case, the player gets one point, and the three kittens are returned to their bench. (Attention! It is also possible that after pushing, three of the opponent's kittens land next to each other — in this case, the point goes to the opponent!)
- The game can finish in two ways:
  - If any player reaches a specified **target score**. In this case this player is the winner.

- If any player has **all their kittens on the field at the same time**. In this case, this player automatically loses, as they cannot play any move.

Implementation

A **sophisticated look** is important, but this does not mean that it should be overdone. You can either use a minimalistic design or even create your own style sheets boosted with various graphic elements. There are no set expectations for appearance, but it is extremely important that **all game elements must be recognizable and usable!**

It might be helpful to create a **static prototype** first. This does not require programming, only knowledge of HTML and CSS. Think about how you will display your elements on the page. *(e.g. What technology will you use to implement the layout of the board and the bench? Grid? Flexbox? Table? Absolutely positioned elements? The decision is only yours!)*

It is important that the entire game must be on a **single HTML page** — so any switching between screens *(e.g. game board and settings)* should be done by dynamically hiding or displaying the appropriate elements!

The next step can be figuring out **what data is needed and what data structure** it should be stored in *(e.g. How to store the game board's current state?)*. Once the data structure has been decided, you may start to think about the **operations** that need to be performed on the data *(e.g. detecting if three kittens are aligned well)*, recognise the **events** that belong to these operations *(e.g. What type of event is triggered on the page and which element triggers it?)*, and finally begin to implement the **event handler** functions. When developing the logic of the game, it may be a good idea to proceed in small steps: solve only one thing at a time.

Purity of the submitted work

The submitted solution must be **your own, individual, independent intellectual product!** This does not mean that you are not allowed use the documentation or look for a vague idea on the Internet, but in case of a significant code match with other students or a large amount copying from external sources, your home project will be **rejected without any possibility of retake/correction!** (In such case, the affected student(s) will not be able to complete the course in this semester in any way!)

In order to avoid unwanted matches, we hereby strongly request that you DO NOT publish your solution to any public storage, GitHub or any other platform at least until the deadline of late submission! In case of code matches, we will not look for the source: all involved students will be penalised equally, since forwarding or publishing the solution of an evaluation is also considered a violation of the academic rules! *(ELTE Academic Regulations for Students, Regulations on the Faculty of Informatics, Section 377/A)*

NO JavaScript framework or library can be used to implement the application — e.g. any use of jQuery, React, etc. is prohibited.

The completed project must be uploaded to the course's Canvas interface as a single *.zip* file. The archive must contain **all files** (e.g. HTML, CSS, media, JavaScript) that are needed to run and evaluate your submission.

In addition to the files of the application, your upload is also **REQUIRED** to contain a `README.md` file **based on the template found here**. You need to fill in the statement and the self-check list in the file. By submitting this file, the student declares that he/she accepts the possible consequences cheating, and the self-check list

makes the evaluation more accurate and faster. Please put an `[X]` in front of every task that you have solved at least partially.

If the uploaded archive is missing the statement, the self-check list or any neccessary files, the submission will not be graded, which will result in failure to complete the course! It is the student's own responsibility to verify in a timely manner that the submission is not incomplete!

Evaluation

20 points can be obtained by solving the basic task. There are required tasks, without which the work will not be accepted. An additional 5 points can be obtained for extra tasks. If someone solves everything, they can get up to 25 points for the assignment.

Criteria related to the JavaScript home assignment for getting your final course grade: it is required to meet all minimum requirements AND have at least 8 points (40%) after the deduction of points due to late submission. (Please consider what is easier for you: solve the minimum requirements by the deadline; or hand in a version with more work two weeks later and get less points with due to the deduction.)

Mandatory tasks (must be completed, 8 pts)

- Other: the `README.md` file from the "Purity of the submitted work" section is included and properly filled in (0 pts)
- Game: the empty board and 8 kittens per player appear (1 pt)
- Game: the two players alternately place a kitten on the board from their own bench (1 pt)
- Game: a new kitten can only be placed on an empty cell (1 pt)
- Game: after placing a new kitten, adjacent kittens are pushed away correctly — this includes that kittens pushed off the board are returned to the corresponding player's bench (2 pts)
- Game: the game detects when three kittens are adjacent in any direction — this includes that the three kittens are returned to the bench and the points are counted (2 pts)
- Game: the game ends when any player reaches 5 points (1 pt)

Basic tasks (12 pts)

- Game: the game ends when a player has all their kittens on the field at the same time (1 pt)
- Game: before placing the kittens, the game indicates (e.g. by changing the background colour of the cells) which kittens will be pushed away (1 pt)
- Game: kittens can also be placed by dragging and dropping them from the bench to the board (2 pts)
- Game: the pushing of the kittens is animated (1 pt)
- Starting screen: the name of the players and the number of points required to win can be entered before starting the game (1 pt)
- Starting screen: the board's dimensions (e.g. 7x7) and the number of kittens per player can be set (1 pt)
- Starting screen: the most recent games are displayed — players, date and time, final result (1 pt)
- Starting screen: the results of recent games will remain even after closing the page (1 pt)
- Game: when the game ends, a new game with the same settings can be started with a single button press without reloading the page (1 pt)
- Meowing: activities (e.g. starting a game, placing a kitten, pushing, earning a point) have sound effects (1 pt)
- Other: nice appearance and code organisation (1 pt)
- **Late submission: -0,5 pts / started day!**

Extra tasks (additional 5 pts)

- Saving: by clicking a button during the game, the current state can be permanently saved, and can later be loaded and continued from the starting screen (2 pts)
- Timer: each player has a timer with 2 minutes of thinking time, which counts down while waiting for the given player's move (like a chess clock) — the player who runs out of thinking time automatically loses (3 pts)