110-1 Web Programming

Hackathon 1 MineSweeper

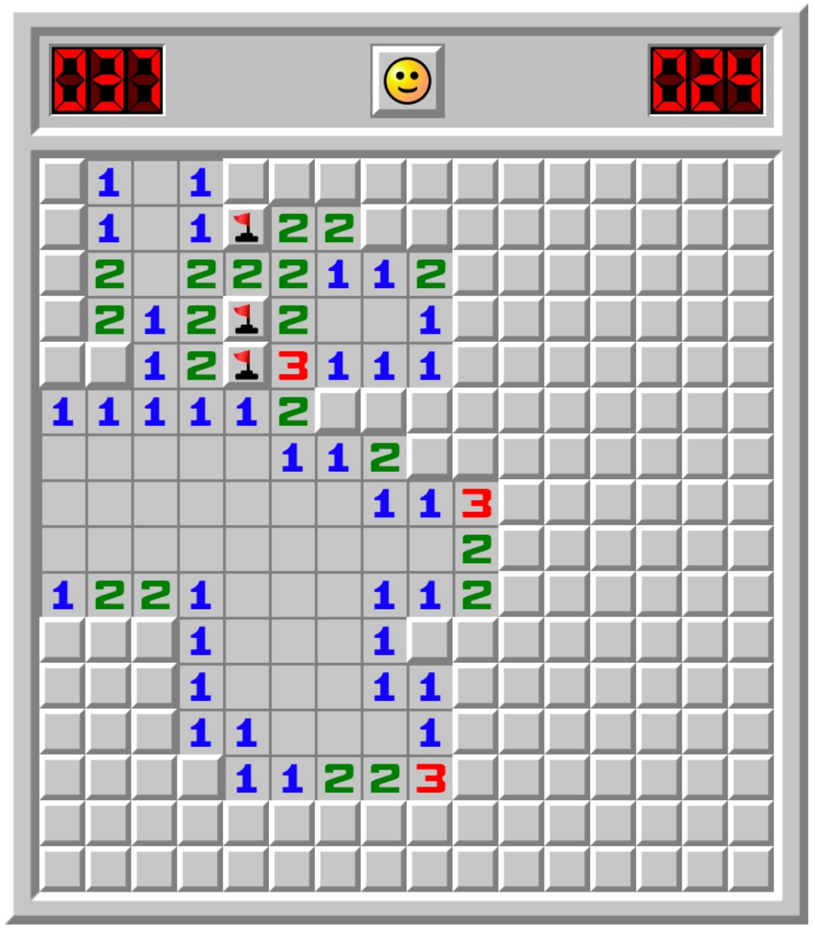
2021. 11. 2　9:10 AM ~ 12:10 PM

TA: 呂承樺、鄭謹譯

Introduction & Rules

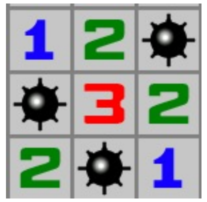
Introduction

- Minesweeper is a single-player puzzle video game. The objective of the game is to clear a rectangular board containing hidden “mines” without detonating any of them, with help from clues about the number of neighboring mines in each field. The game originates from the 1960s, and it has been written for many computing platforms in use today. It has many variations and offshoots.



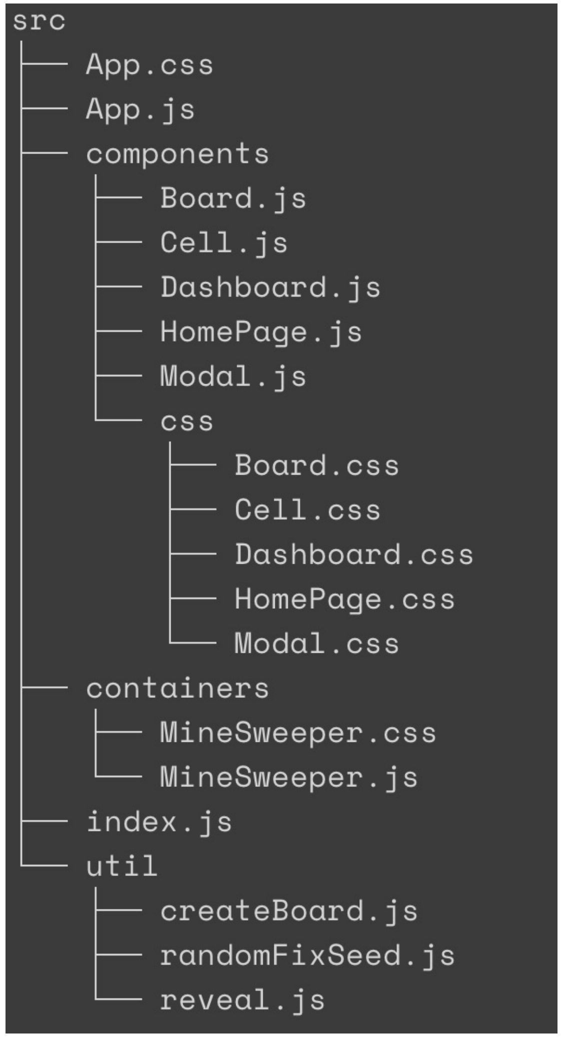
Our rules of MineSweeper

- Click “Start Game” to begin the game.
- Use the left click button on the mouse to select a cell on the board. If you hit a mine, you lost.
- The numbers of the cells represent how many mines are adjacent to a cell. For example, if a cell has a “3” on it, then there are 3 mines around that cell. The mines could be above, below, right, left, or diagonal to the cell.
- Avoid all the mines and reveal all the empty cells to win MineSweeper.
  - Use the numbers to determine where the mine(s) could be.
  - You can right click a cell with the mouse to place a flag where you think a mine is. This allows you to avoid that spot.

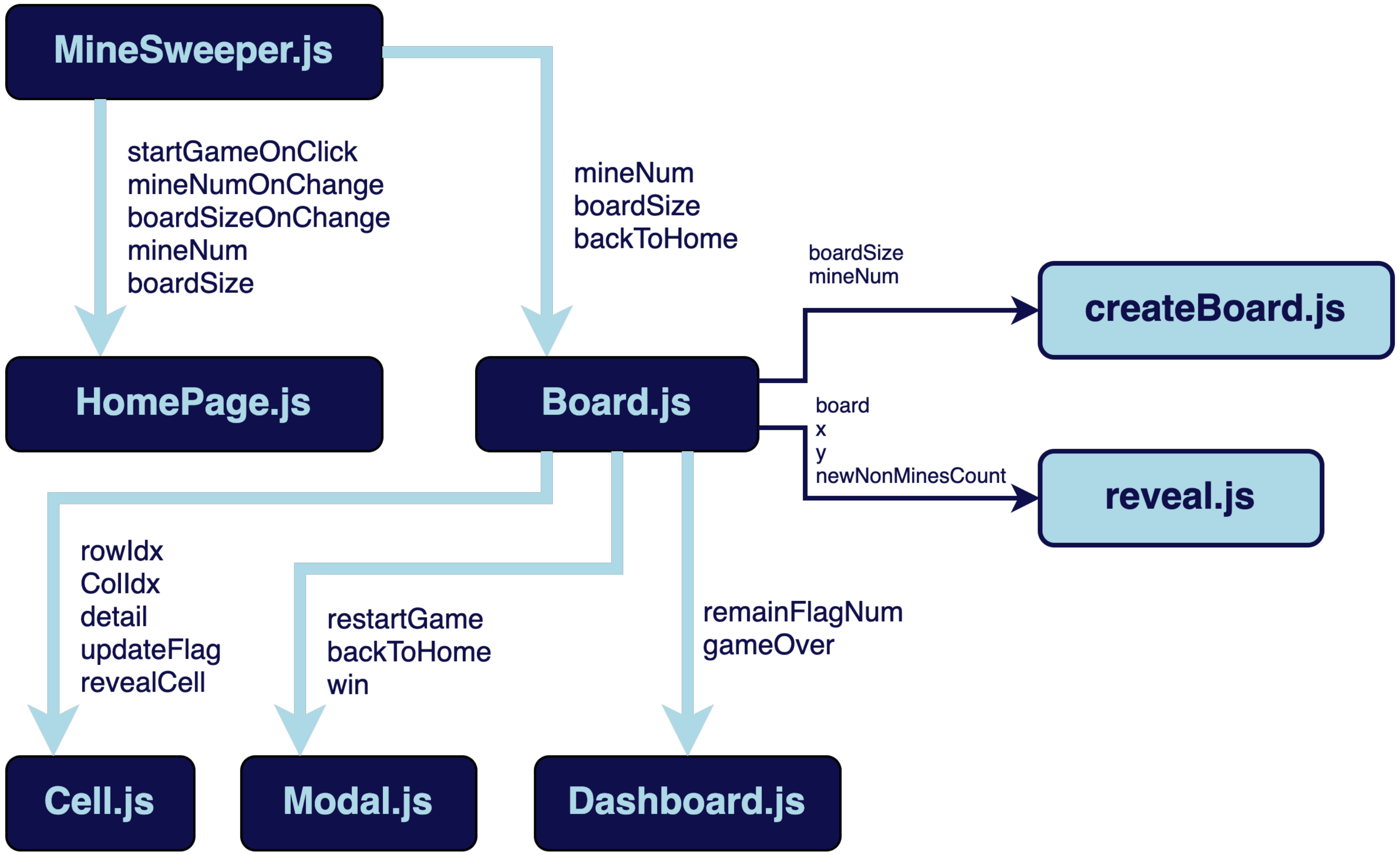


Structures and Diagrams

File Structure of MineSweeper



```
src
├── App.css
├── App.js
├── components
│   ├── Board.js
│   ├── Cell.js
│   ├── Dashboard.js
│   ├── HomePage.js
│   ├── Modal.js
│   └── css
│       ├── Board.css
│       ├── Cell.css
│       ├── Dashboard.css
│       ├── HomePage.css
│       └── Modal.css
├── containers
│   ├── MineSweeper.css
│   └── MineSweeper.js
├── index.js
└── util
    ├── createBoard.js
    ├── randomFixSeed.js
    └── reveal.js
```



* All the files have been created already, you **DO NOT** have to create or delete any file on your own.

TODOs

TODO 1 — HomePage Start (10%)

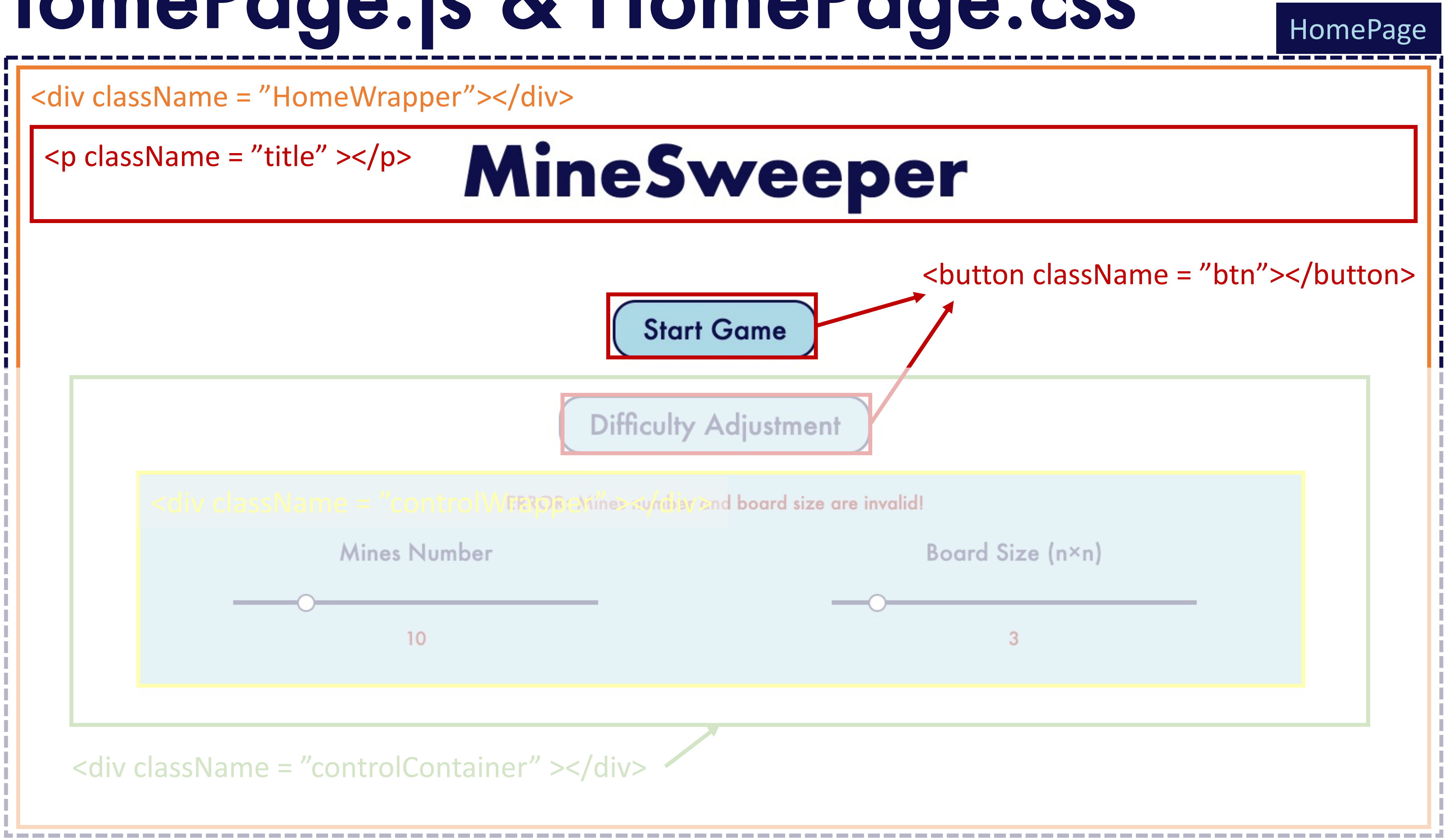
- 1-1: Try to implement the HTML of HomePage. (5%)
  - You will see a blank page if you try to implement this project in the beginning.
  - You should add something to HomePage.js and MineSweeper.js.
  - The structure of this page is shown in next page, please follow the diagram strictly.
    - As a reminder, **this TODO only contains the "Start Game" button and the reaction of pressing this button.** Difficulty Adjustment is left in TODO 6.

Notice: Do not add additional space before and after the string

```
<p>Correct Example</p>
<p> Wrong Example</p>
<p>Wrong Example </p>.
```

Difficulty: ★☆☆☆☆

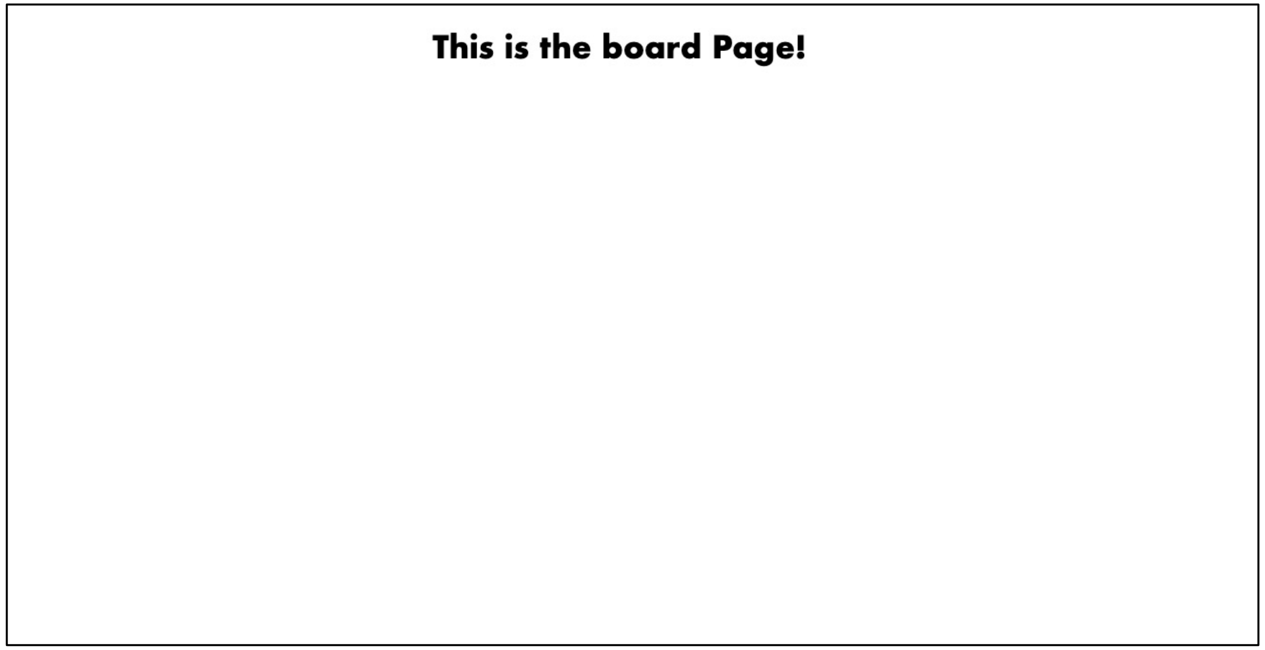
HomePage.js & HomePage.css


HomePage
<div className = ”HomeWrapper”></div>
<p className = ”title” ></p>
MineSweeper
<button className = ”btn”></button>
Start Game
Difficulty Adjustment
<div className = ”controlWrapper” ></div>
Mines Number
Board Size (n×n)
10
3
<div className = ”controlContainer” ></div>

ERROR: Mines number and board size are invalid!

TODO 1 — HomePage Start (10%)

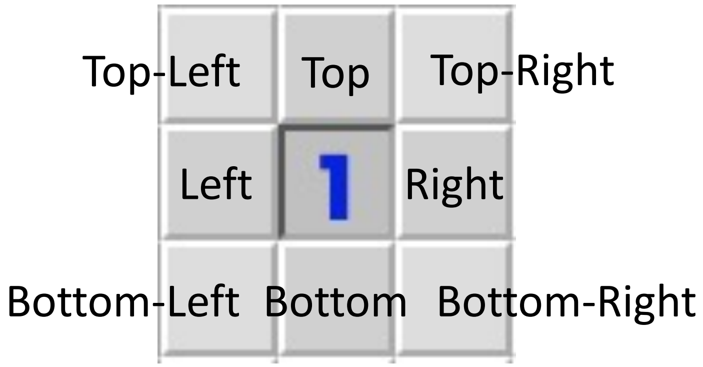
- 1-2: Make sure you can start a game when pressing the "Start Game" button. (5%)
  - *startGameOnClick* in MineSweeper.js is a function to make the game start and also make the screen switch from HomePage to Board.
  - After you press the "Start Game" button, you will see the page as below.

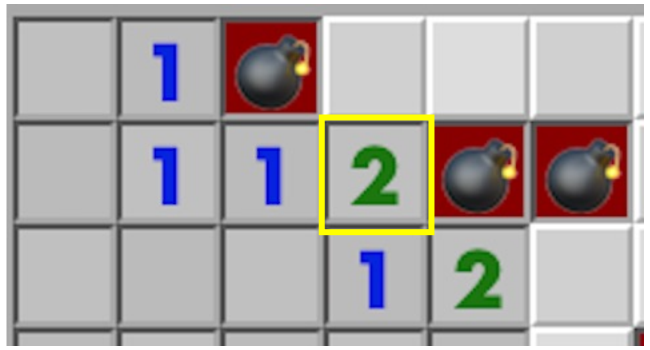


Difficulty: ★☆☆☆☆

TODO 2 — createBoard.js (10%)

- Counting the value of each cell in the Board to complete createBoard.js. (10%)
  - The value is the number of mines around the cell. The definition of the position is shown in the left figure below.
  - Take the right figure below as an example. Since the value of "Top-Left" and "Right" of the yellow block is '💣', so the value of the yellow block must be set to "2".
  - You can use the function *printBoard* for testing.





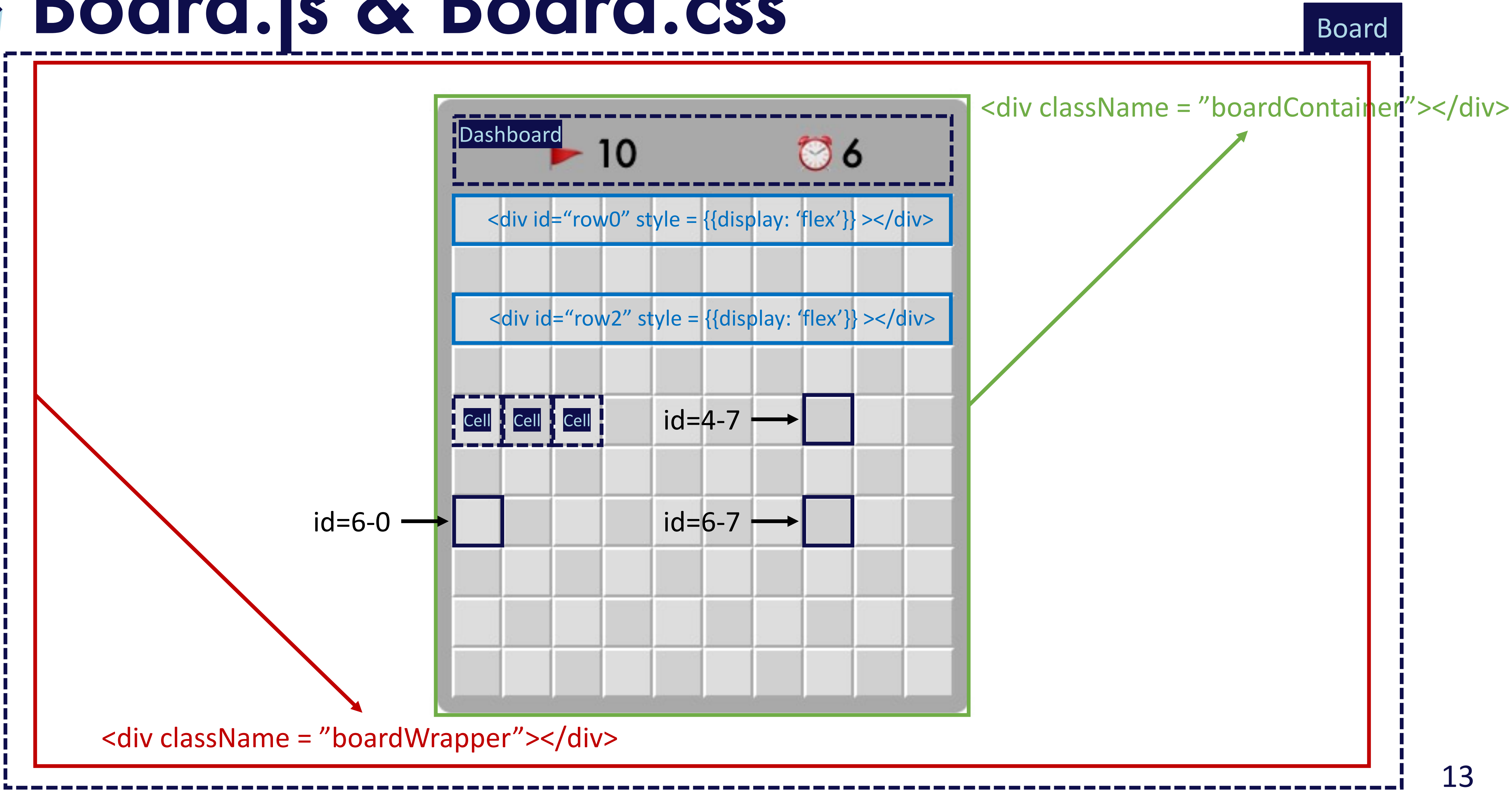
Difficulty: ★★☆☆☆

TODO 3 — Board.js (20%)

- 3-1. Implement the Board using the component Dashboard in Dashboard.js and Cell in Cell.js which is done. (10%)
  - Use the function in createBoard.js to implement *freshBoard*
  - The board is built by 2-dimensional Cell. Use the map function to finish it.
  - Follow the diagram in next page strictly. Remember adding id to some tags.
  - To get all points, you do not need to implement the timer in Dashboard.
- 3-2. Implement the function *updateFlag* of right click to flag a cell. (10%)
  - You can only flag the cell if it is not revealed.
  - If the cell is already flagged, you should unflagged it if you right click it.
  - Also remember to update the board and the remainFlagNum.

Difficulty: ★★★★☆

Board.js & Board.css


Board
<div className = ”boardContainer”></div>
Dashboard
10
6
<div id=“row0” style = {{display: ‘flex’}} ></div>
<div id=“row2” style = {{display: ‘flex’}} ></div>
Cell
Cell
Cell
id=4-7
id=6-0
id=6-7
<div className = ”boardWrapper”></div>

TODO 4 — Reveal Cells (25%)

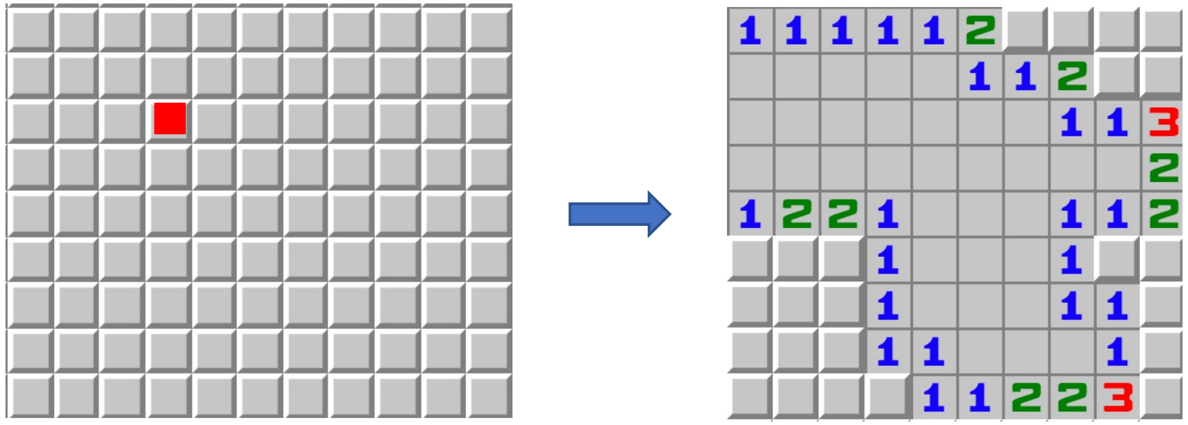
- 4-1. Complete the *revealCell* function in Board.js. (10%)
  - Do not reveal the cell if
    - the cell has revealed
    - the cell is flagged
    - won or lost
  - You should consider two conditions:
    - If the cell you want to reveal is one of the mines' location, show all the mines that have not been flagged and you lost the game.
    - If the revelation of this cell will result in winning the game.

Notice: This implementation may be related to the function in reveal.js

Difficulty: ★★★★☆

TODO 4 — Reveal Cells (25%)

- 4-2. The outcome of revelation (function in reveal.js). (15%)
  - If the cell is already revealed, do nothing. And if the value of the cell is not 0, only show the cell value. (5%)
  - If the value of the cell is 0, you should revealed all the values of adjacent cells, and if the values are 0 again, you should consider that 0 and revealed all its adjacent cells, until no other revealed value is 0, as below graph. (10%)
    - As an example or a hint: You can consider the adjacent cells respectively. If the value of any adjacent cell is 0, that cell must be revealed. Perhaps, you can consider this question into two stage. First, to check the value of each adjacent cell. Then, reveal the cells that must be revealed at the same time.



Difficulty: ★★★★☆

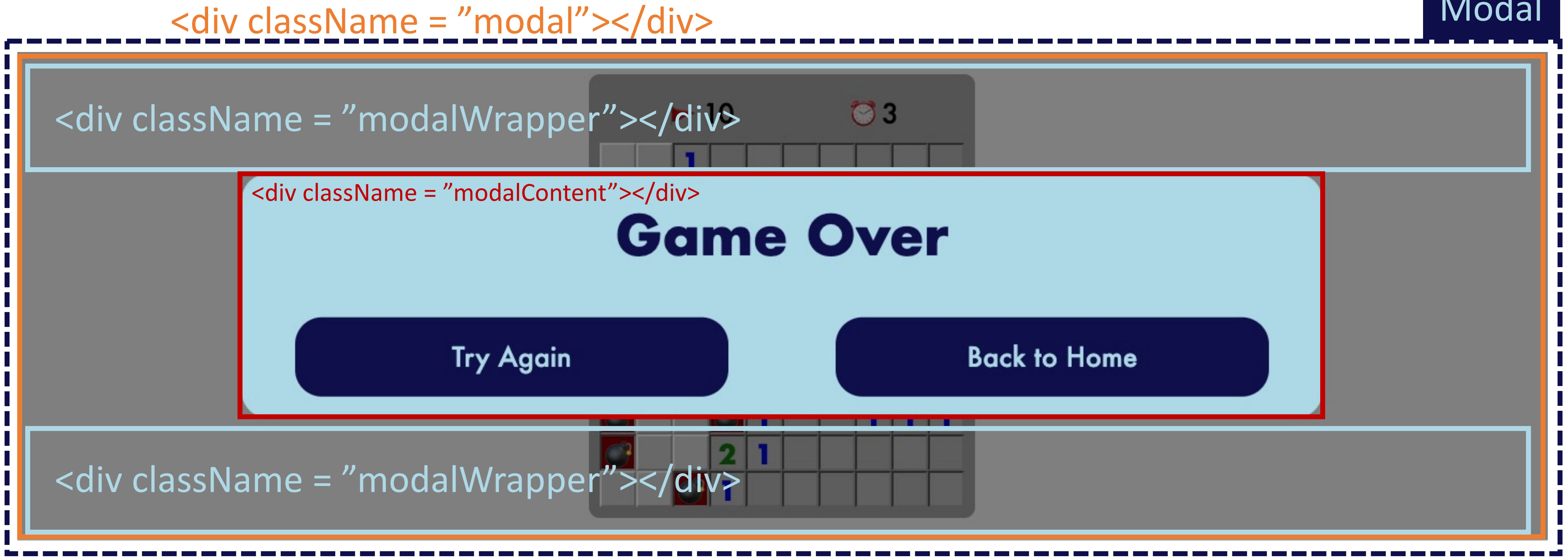
TODO 5 — Modal (15%)

- 5-1: Implement the html of Modal in modal.js. (10%)
  - The structure of Modal must follow the diagram in next page strictly.

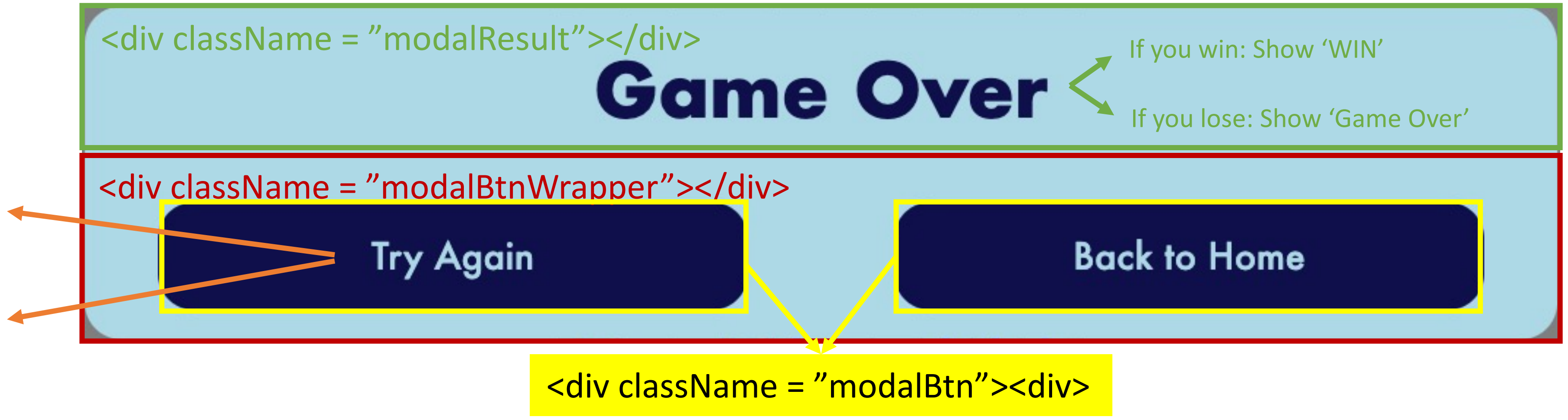
Difficulty: ★☆☆☆☆

Modal.js & Modal.css

Modal



<div className = ”modal”></div>
<div className = ”modalWrapper”></div>
<div className = ”modalContent”></div>
Game Over
Try Again
Back to Home
<div className = ”modalWrapper”></div>



<div className = ”modalResult”></div>
Game Over
If you win: Show ‘WIN’
If you lose: Show ‘Game Over’
<div className = ”modalBtnWrapper”></div>
Try Again
Back to Home
<div className = ”modalBtn”><div>

If you win: Show ‘New Game’

If you lose: Show ‘Try Again’

TODO 5 — Modal (15%)

- 5-2: Two buttons. (5%)
  - Use the function *freshBoard* to implement *restartGame* in Board.js
  - *backToHomeOnClick* (in MineSweeper.js) will switch the page back to HomePage and reset all the customized settings (default).

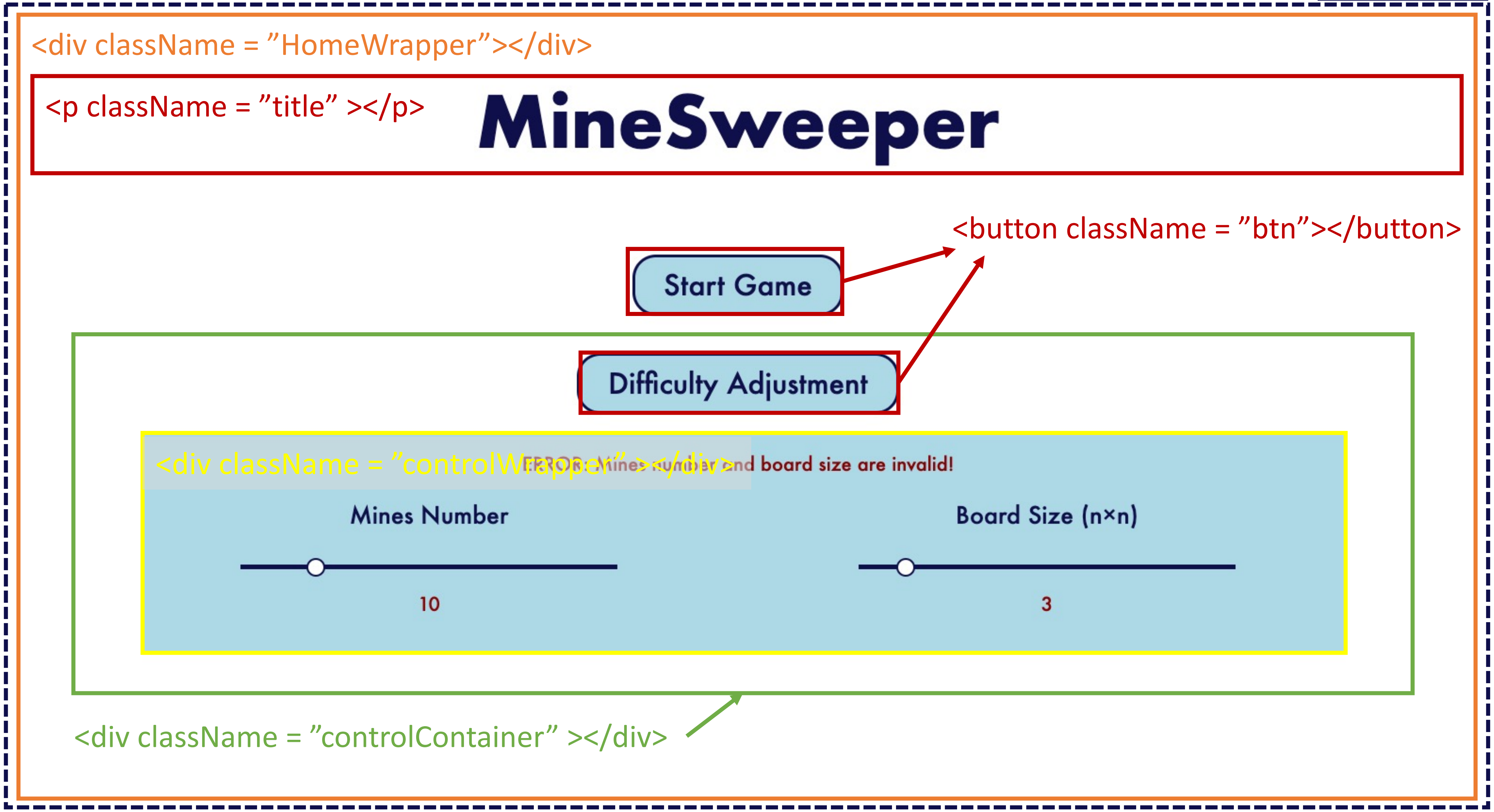
Difficulty: ★☆☆☆☆

TODO 6 — HomePage Control (15%)

- 6-1: Try to implement a button called “Difficulty Adjustment” and its functions. (5% for structure, 5% for input value update, 5% for error checking)
  - Initially, the control panel is invisible. *[showPanel = false]*
  - If you press the “Difficulty Adjustment” button, the control panel will show under the button. And then, if you press it again, the control panel will be invisible as expected.
  - In the control panel, you should implement two input sliders (see next page for detailed structure). The two input sliders can control the number of mines and the board size of the game. (1 ≤ mineNum ≤ 50, 1 ≤ boardSize ≤ 20)
  - If the two input values are invalid to build a new game, which means that the number of mines is bigger than the square of board size, show the Error messages and change the value of input sliders into darkred. At this time, you cannot start the game when pressing the “Start Game” button.

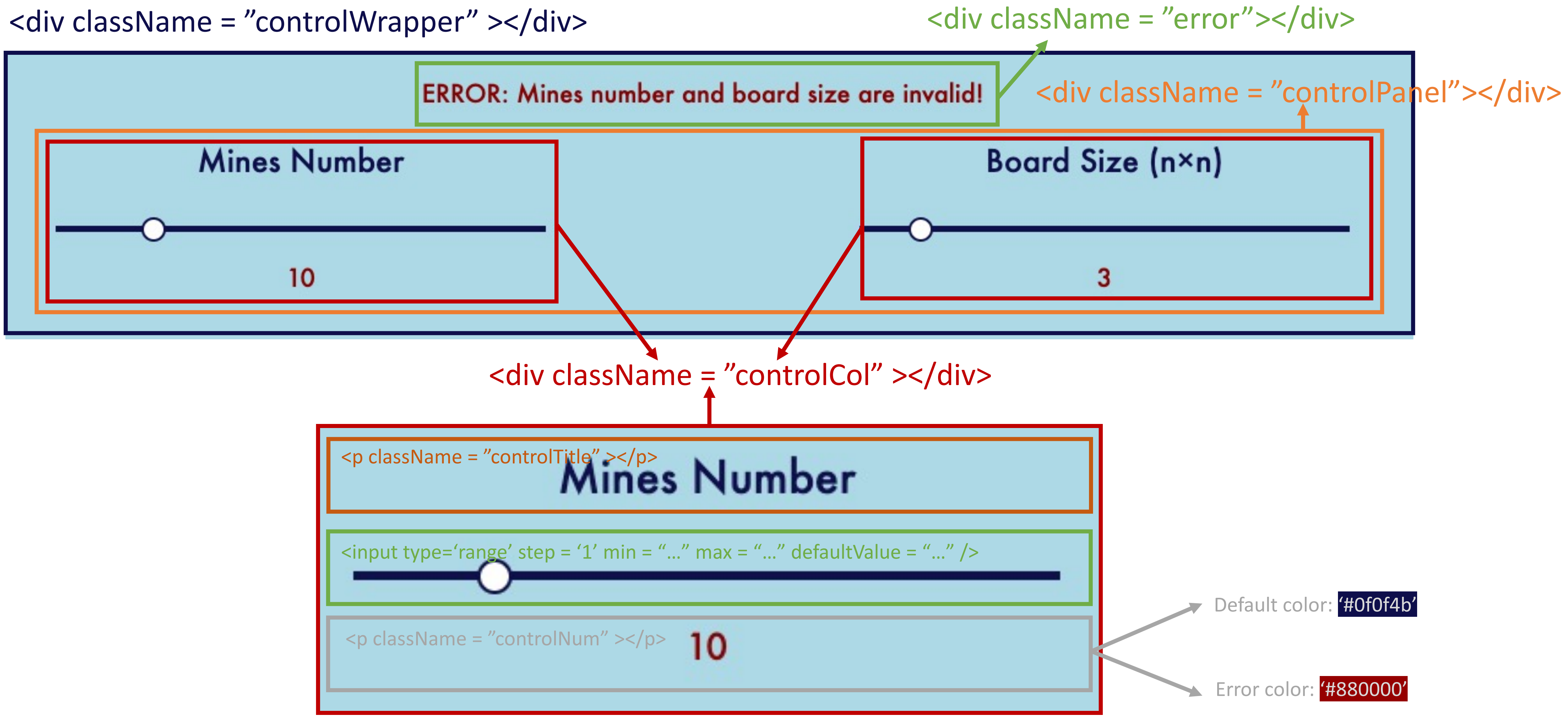
Difficulty: ★★★☆☆

HomePage.js & HomePage.css



```
<div className = ”HomeWrapper”></div>
<p className = ”title” ></p>
<button className = ”btn”></button>
<div className = ”controlWrapper” ></div>
<div className = ”controlContainer” ></div>
```

HomePage.js & HomePage.css



<div className = ”controlWrapper” ></div>
<div className = ”error”></div>
ERROR: Mines number and board size are invalid!
<div className = ”controlPanel”></div>
Mines Number
10
Board Size (n×n)
3
<div className = ”controlCol” ></div>
<p className = ”controlTitle” ></p>
Mines Number
<input type=‘range’ step = ‘1’ min = “…” max = “…” defaultValue = “…” />
<p className = ”controlNum” ></p>
10
Default color: ‘#0f0f4b’
Error color: ‘#880000’

TODO 6 — HomePage Control (15%)

- 6-2: *mineNumOnChange* and *boardSizeOnChange* (both in MineSweeper.js) are two functions to make the game customized and change the input values synchronously.
  - You should use these two functions to update the values of difficulty.

Difficulty: ★★★☆☆

TODO 7 — Dashboard.js (5%)

- Complete the Dashboard (5%)
  - Make sure the clock will start counting when the game start.
  - Also, if the game win or game over, the clock will stop.
  - If the game is restarted, the clock must be adjusted to 0 and start counting again.

Difficulty: ★★★★★

Tests

Cypress

- We only provide a part of testcases
- Almost all tags in ”Static pages” are tested in those provided testcases.
- Passing all testcases definitely unequal to scoring 100% in hackathon.
  - There are some private tests

- ***“npm run test” will perform all tests provided in this hackathon.***
- You can try to implement more tests by cypress.

Cypress Random Issues

- The random function is written in util/randomFixSeed.js
- The default random seed is “over my dead body”
  - Provided run tests are based on this seed.
  - These tests will also be included in our scoring recipe.

- You can modify the seed to perform other tests.
  - Seed as a string.