

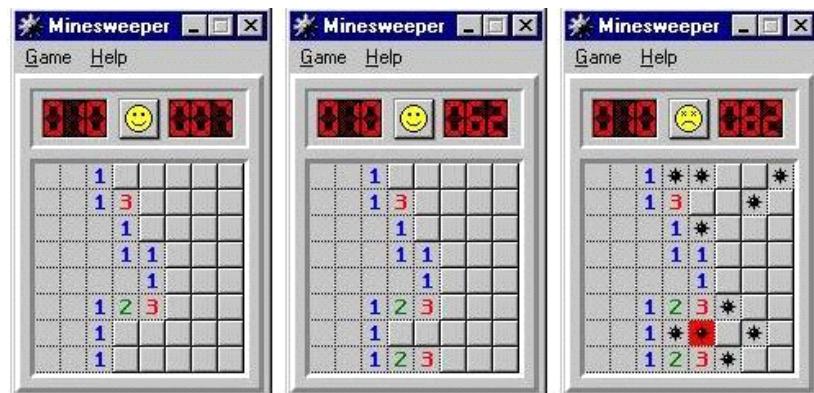
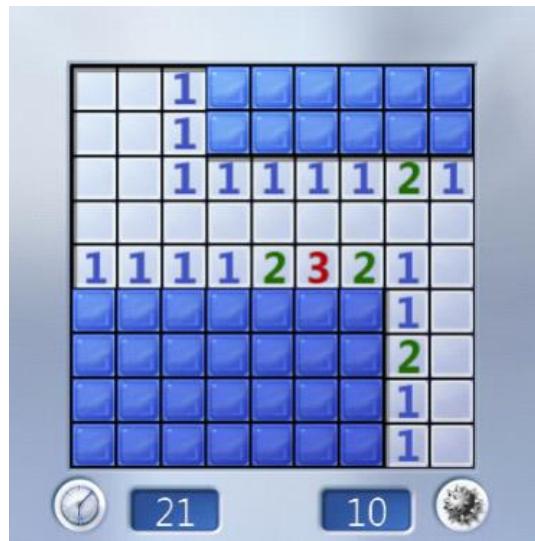
# Sprint 1 Challenge

## Mine Sweeper

### Preface

Let's create a super version of the **Minesweeper game**,

First, play [the game](#) a little bit, get to know it and relax



## Software Delivery Phases - Instructions

In this sprint **we work alone**, we code our own solution and bring our own knowledge and tools - please respect the rules.

Delivery is done through GitHub and Dropbox.

We will have 4 delivery points:

- *1<sup>st</sup> delivery:* Sunday 21:30
- *2<sup>nd</sup> delivery:* Monday 21:30
- *3<sup>rd</sup> delivery:* Wednesday 21:30
- *Final delivery:* Thursday 21:30

The sprint base-score will be determined by this delivery – please do your best to have a working game

- *Bonus (optional):* Saturday 21:30
- *Presentation:* Monday 17:30

We will go through all projects, review the feature and get some cheering up from everyone.

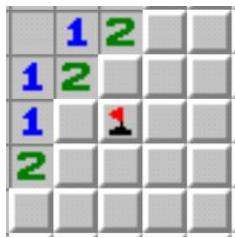
## Minesweeper – Basic intro

The goal of the game is to uncover all the squares that do not contain mines without being "blown up" by clicking on a square with a mine underneath.

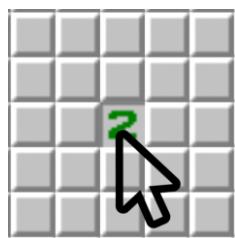
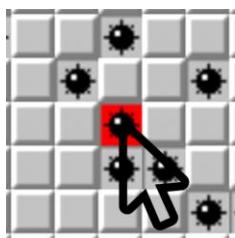
Our Minesweeper basic functionality is based on the [reference game](#)

## Functionality and Features

- Show the board
- Left click **reveals** the cell's content
- Right click flags/unflags a suspected cell (cannot **reveal** a flagged cell)



- Game ends when:
  - LOSE: when clicking a mine, all the mines are revealed
  - WIN: all the mines are flagged, and all the other cells are shown
- Support 3 levels of the game
  - Beginner (4 \* 4 with 2 MINES)
  - Medium (8 \* 8 with 14 MINES)
  - Expert (12 \* 12 with 32 MINES)
- Expanding: When **left clicking** on cells there are 3 possible cases we want to address:
  - MINE – reveal the mine clicked
  - Cell with neighbors – reveal the cell alone
  - Cell without neighbors – expand it and its 1<sup>st</sup> degree neighbors



## Development - Tips and Guidelines

As you know, there is usually more than one way to approach a challenge.

But as a guideline, we suggest having the following functions (it is ok to have more functions as needed).

<code>onInit()</code>	This is called when page loads
<code>buildBoard()</code>	Builds the board Set the mines Call <code>setMinesNegsCount()</code> Return the created board
<code>setMinesNegsCount(board)</code>	Count mines around each cell and set the cell's

	minesAroundCount.
<code>renderBoard(board)</code>	Render the board as a <table> to the page
<code>onCellClicked(elCell, i, j)</code>	Called when a cell is clicked
<code>onCellMarked(elCell)</code>	Called when a cell is right-clicked See how you can hide the context menu on right click
<code>checkGameOver()</code>	Game ends when all mines are marked, and all the other cells are shown
<code>expandShown(board, elCell, i, j)</code>	<p>When user clicks a cell with no mines around, we need to open not only that cell, but also its neighbors.</p> <p>NOTE: start with a basic implementation that only opens the non-mine 1<sup>st</sup> degree neighbors</p> <p>BONUS: if you have the time later, try to work more like the real algorithm (see description at the Bonuses section below)</p>

Here are the global variables you might be using:

<b>gBoard</b> - A Matrix containing cell objects: Each cell: { <b>minesAroundCount</b> : 4, <b>isShown</b> : false, <b>isMine</b> : false, <b>isMarked</b> : true }	The model
<b>gLevel</b> = { <b>SIZE</b> : 4, <b>MINES</b> : 2	This is an object by which the board size is set (in this case: 4x4 board and how many mines

<pre>}</pre>	to place)
<pre>gGame = {     isOn: false,     shownCount: 0,     markedCount: 0,     secsPassed: 0 }</pre>	<p>This is an object in which you can keep and update the current game state:</p> <ul style="list-style-type: none"> <li><b>isOn</b>: Boolean, when true we let the user play</li> <li><b>shownCount</b>: How many cells are shown</li> <li><b>markedCount</b>: How many cells are marked (with a flag)</li> <li><b>secsPassed</b>: How many seconds passed</li> </ul>

## Development - How to start?

Breaking-down the task to small tasks is a key success factor.  
In our case – we recommend starting from the following steps:

### Step1 – the seed app:

- Create a 4x4 gBoard Matrix containing Objects.
- Set 2 of them to be mines
- Present the mines using `renderBoard()` function.

### Step2 – counting neighbors:

- Create `setMinesNegsCount()` and store the numbers
- Update the `renderBoard()` function to also display the neighbor count and the mines
- Add a `console.log` – to help you with debugging

### Step3 – click to reveal:

- When clicking a cell, call the `onCellClicked()` function.
- Clicking a safe cell reveals the `minesAroundCount` of this cell

### Step4 – randomize mines' location:

- Add some randomicity for mines location

- After you have this functionality working– its best to comment the code and switch back to static location to help you focus during the development phase

Step5 –

- Add a footer with your name
- Upload to git

Continue to **Functionality and Features**, then to **Further Tasks**, and if you went that far, do go ahead and check the **Bonus Tasks**.

## UI Guidelines

This sprint is not a UI-centered project, however, do your best to make it look nice

## Further Tasks

### First click is never a Mine

The first clicked cell is never a mine

HINT: We need to start with an empty matrix (no mines) and then place the mines and count the neighbors only on first click.

### Lives

Add support for “LIVES” -

The user has 3 LIVES:



When a MINE is clicked, there is an indication to the user that he clicked a mine. The LIVES counter decreases. The user can continue playing.

### The Smiley button

Add the smiley button - clicking the smiley resets the game here are some smiley ideas:

- Normal 😊
- Sad & Dead – LOSE 😞 (stepped on a mine and have no life left)
- Sunglasses – WIN 😎

### Bonus Tasks – if time permits

#### Add support for HINTS

The user has 3 hints



When a hint is clicked, it changes its look, example:

Now, when a cell (unrevealed) is clicked, the cell and its neighbors are revealed **for a second**, and the clicked hint disappears.

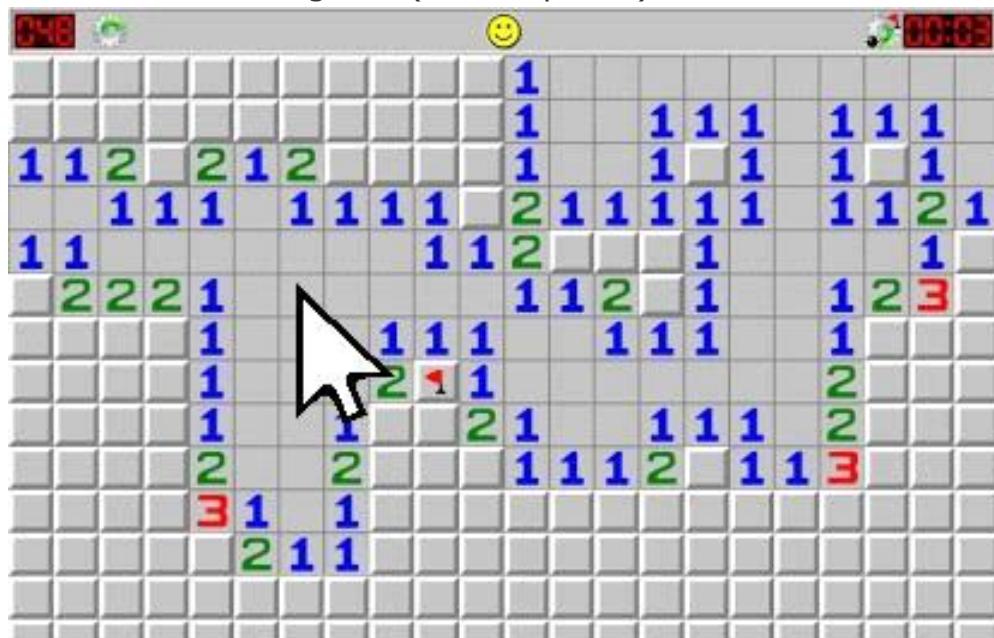
#### Best Score

Keep the best score in [local storage](#) (per level) and show it on the page

## Full Expand

When an empty cell is clicked, open all empty cells that are connected and their numbered neighbors

Expand like in the real game ("Full expand"):



Think about a recursion.

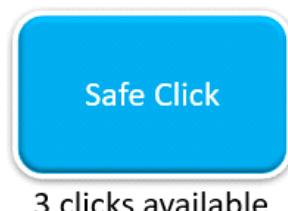
## Safe click

Add a **Safe-Click** Button:

The user has 3 **Safe-Clicks**

Clicking the **Safe-Click** button will mark a random covered cell (for a few seconds) that is safe to click

Present the remaining **Safe-Clicks** count



### **Manually positioned mines**

Create a “manually create” mode in which user first positions the mines (by clicking cells) and then plays.

### **Undo**

Add an “UNDO” button, each click on that button takes the game back by one step (can go all the way back to game start).



### **DARK MODE**

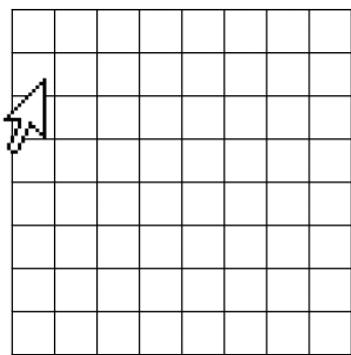
Implement Dark-Mode for the game

### **MEGA HINT**

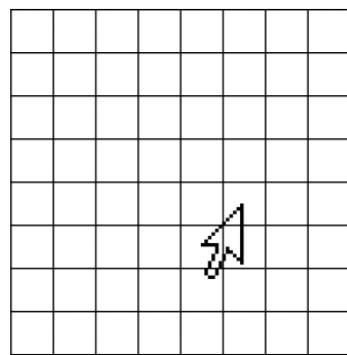
Mega-Hint works only once every game. It is used to reveal an area of the board for 2 seconds. Functionality description: (1) Click the “Mega Hint” button (2) then click the area’s top-left cell (3) then click bottom-right cell. The whole area will be revealed for 2 seconds.

## Mega Hint

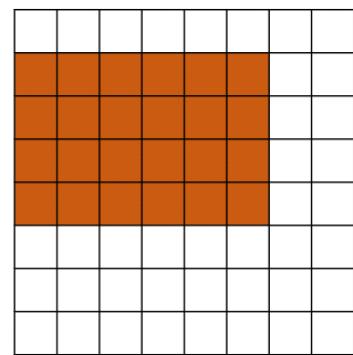
Step1: Click the Mega-Hint button



Step2: Click the area's top-left corner



Step3: Click the area's bottom-right corner



Result: the selected area's content will be revealed for 2 seconds

## MINE EXTERMINATOR



Clicking the “Exterminator” button, eliminate 3 of the existing mines, randomly. These mines will disappear from the board. We will need re-calculation of neighbors-count

**זהירות – הספרינט הזה ייקח אותך רחוק משחשבות**

