Assignment 2: JavaScript

Due 12 May by 17:00 **Points** 100

COMP284 Scripting Languages (2020-21) -- Assignment 2: JavaScript

Your task for this assignment consists of two parts:

1. Develop a JavaScript program that provides the functionality stated in the **Requirements section** below.
2. Make the JavaScript program that you have created accessible and usable via the URL

   `https://student.csc.liv.ac.uk/~<your user name>/game.html`

   taking care of the requirements set out in **Submission and Setup section** below.

Requirements

The JavaScript program implements a simple game that consists of three stages, *setup*, *play* and *end*. During the *play* stage the game proceeds in *rounds*. The game is played on a grid with 10 x 10 cells, surrounded by a wall, and involves our hero, controlled by the user, and a bunch of evil killer robots, controller by the computer (that is, your program). Our hero hunts for treasure on the grid while the killer robots hunt our hero. The user and your program are the two *players* of the game.

The game always starts in the *setup* stage. During that stage the user is shown the grid and can place four different types of objects on the cells of the grid:

- by clicking on a cell and typing a number between 1 and 9, a *treasure* is placed on a cell, the number indicates the *value of the treasure*;
- by clicking on a cell and typing the letter "o", an *obstacle* is placed on a cell;
- by clicking on a cell and typing the letter "h", the *hero* is placed on a cell.
- by clicking on a cell and typing the letter "k", the *a killer robot* is placed on a cell.

There is no limit on the number of treasures, obstacles and killer robots, but there is only one hero. No cell can contain more than one object. If theuser types a character that is not among 1 to 9, "o", "h" and "k", an error message should be shown.

In addition to the grid, there must be a button that allows the user to end the *setup* stage of the game. If the user tries to end the *setup* stage of the game without placing the hero, then an error message should be shown and the user remains in the *setup* stage. Otherwise the game continues with the *play* stage.

At the start and during the *play* stage, the user is again shown the grid, initially with all the objects that have been placed on the grid, plus additional *status information*: The number of the *round* currently played, the *number of treasures* still on the grid, the *user's score*, the *killer robots' score*. Initially, round 1 is played and both the user's score and the killer robots' score is 0. In addition, there must be a button that allows the user to end the *play* stage at any time.

While in the *play* stage, the game proceeds in rounds, each round starting with the user's turn followed by the computer's turn.

During his/her turn, the user can attempt to move the hero horizontally or vertically on the grid by typing one of four letters:

- "a" attempts to move the hero one cell to the left,
- "d" attempts to move the hero one cell to the right,
- "w" attempts to move the hero one cell up,
- "s" attempts to move the hero one cell down.

If the user types any other character, then an error message should be shown, the user's turn does not end, and the user can type another character. If the attempted move would result in the hero ending up outside the grid or on a cell occupied by an obstacle, then an error message should be shown, the attempt to move fails, the hero does not move, and the user's turn is over. Otherwise, the attempted move is successful and the hero changes cells. If the hero ends up on a cell occupied by a killer robot, then our hero dies, the game is over and the game proceeds to the *end* stage. If the hero ends up on a cell that contains a treasure, then that treasure is removed from the grid and the value of the treasure is added to the user's score (if the last treasure is removed, then the game proceeds to the *end* stage). After the hero changed cells and possibly collected a treasure or died, the user's turn is again over.

During the computer's turn your program attempts to move each of the killer robots in some order. Unlike the hero, the killer robots are not only able to move horizontally and vertically but also diagonally. Just like the hero, each killer robot only moves at most one cell in a turn. If the hero is in a cell immediately surrounding a killer robot, then that killer robot must move to the cell occupied by the hero; the hero dies, the computer's turn and the round ends, and the game moves to the *end* stage. If the hero is not in a cell immediately surrounding a killer robot, but one or more of those cells contains a treasure, then the killer robot must move to one of those cells, the treasure on that cell is removed from the grid, and the value of the treasure is added to the killer robots' score. If none of the surrounding cells contains the hero nor a treasure, then a killer robot can move to an arbitrary surrounding empty cell. A killer robot is not allowed to stand still if it can move and the order in which killer robots attempt to move must be such that the number of robots that move is maximised. However, if a killer robot cannot move at all, then the computer should simply proceed to the next killer robot. Once an attempt has been made to move each of the killer robots, the computer's turn and the current round ends, and the number of rounds played is increased by one.

Whenever the number of rounds played, the number of treasures remaining on the grid, the user's score, or the killer robots' score changes, the *status information* shown by the program must be updated.

The *play* stage ends if one of the following conditions becomes true

- the user ends the *play* stage by activating the button provided for that;
- the hero dies;
- there are no treasures left on the grid;
- neither the hero nor any of the killer robots is able to move.

Once the *play* stage has ended, the game is in the *end* stage. In the *end* stage the program determines the outcome of the game. The outcome is a *win* for the user if the hero is still alive and the user's score is higher than the killer robots' score; the outcome is a *win* for the computer if the hero is dead or the killer robots' score is higher than the user's score; otherwise, the outcome is a *draw*. The program should display a message indicating the outcome of the game and then stop. During the *end* stage the program should not react to any user input or actions.

Additional requirements and comments:

- The bulk of your JavaScript code should be in a JavaScript library called `game.js`. Before submitting your solution, you should create a copy of `game.js` named `game.pretty.js` in a directory other than your `public_html` directory, say, your home directory. Then make the file `game.js` indecipherable for humans using the command

  ```
  uglifyjs $HOME/game.pretty.js --mangle --compress > $HOME/public_html/game.js
  ```

  Make sure that after performing this operation your game still works. Also make sure that `game.pretty.js` can only be read by yourself.
- It is possible that during the *setup* stage the user does not place any treasures on the grid. On entering the *play* stage your program should recognise that, immediately proceed to the *end* stage, and declare that **the game has ended in a draw.**
- It is also possible that during the *setup* stage the user does not place any killer robots on the grid. The game still proceeds as described above, only that nothing happens during the computer's turn.
- You should carefully analyse in which situations the hero and the killer robots might not be able to move in order to correctly end the *play* stage in such a situation.
- Ideally your program would move the killer robots in such a way that they increase their chances of killing the hero. This could be done by each killer robot trying to decrease the distance to the hero with each move. But you could also implement a strategy by which the killer robots try to `encircle' the hero in order to increase their chances to kill the hero. They could also `guard' one specific treasure, knowing that the hero must eventually try to collect it.
- JavaScript engines differ from browser to browser. You should make sure that your system works in all commonly used browsers (e.g., Google Chrome, Mozilla Firefox, Microsoft Internet Explorer 9 or higher) and on all commonly used platforms (e.g., Linux derivatives and Microsoft Windows).

- Your JavaScript program should only depend on your own code. JavaScript libraries/frameworks must not be used.
- Your code should follow the **COMP284 Coding Standard (https://cgi.csc.liv.ac.uk/~ullrich/COMP284/notes/COMP284CodingStandard.pdf)**. This includes pointing out which parts of your code have been developed with the help of on-line sources or textbooks and references for these sources.

A script that deals satisfactorily with these additional requirements and comments, in addition to providing the basic functionality required, will receive higher marks.

Submission and Setup

Submit all relevant files (game.html, game.js, game.pretty.js, any CSS file, and local images) via the departmental submission system at **https://sam.csc.liv.ac.uk/COMP/Submissions.pl?module=comp284 (https://sam.csc.liv.ac.uk/COMP/Submissions.pl?module=comp284)** (COMP284-2: JavaScript).

Make game.html, game.js, and other files (but not game.pretty.js) accessible via the departmental web server.

The files submitted must be identical to those set up on the departmental web server. Furthermore, no alterations are allowed to the latter after files have been submitted. If a submitted file and the corresponding file on the departmental web server have different timestamps, then the later timestamp will be used to determine lateness. This applies even if the earlier file is used for marking.

Permissions of the files in your filestore must be such that no other user can view their contents in the filestore.

Deadline

The deadline for this assignment is

To be confirmed

Earlier submission is possible, but any submission after the deadline attracts the standard lateness penalties. Please remember that a strict interpretation of `lateness' is applied by the Department, that is, a submission a minute after the deadline is considered to be a day late.

Assessment

This assignment will address the following learning outcomes of the module:

- Develop computer-based or client-side web-based applications using an appropriate scripting language.

This assignment will contribute **50%** to the overall mark of COMP284. Failure on this assignment may be compensated by higher marks on other assignments for this module.

Marks will be awarded according to the following scheme:

- Submission, Setup, Error-freeness: 10
- Quality of the interface design: 8
- Correctness and quality of the implementation of the *setup* stage: 12
- Correctness and quality of the implementation of the *play* stage: 48
- Correctness and quality of detecting the end of the game and of the *end* stage: 10
- Code layout, commenting, and quality of code: 12

In more detail, the requirements above translate into about 32 criteria that your system must satisfy. Marks are given according to the extent to which the system is observed to behave in the expected way and produces correct results, and, to a lesser extent, how well the code is written. Code that has no observable effect will typically receive no marks.

As stated above, the University policy on late submissions applies to this assignment, as do the University policy on coursework submission (available at **https://www.liverpool.ac.uk/media/livacuk/tqsd/code-of-practice-on-assessment/appendix_Q_cop_assess.pdf** **(https://www.liverpool.ac.uk/media/livacuk/tqsd/code-of-practice-on-assessment/appendix_Q_cop_assess.pdf)**) and the University policy on academic integrity (available at **http://www.liv.ac.uk/student-administration/student-administration-centre/policies-procedures/academic-integrity/** **(http://www.liv.ac.uk/student-administration/student-administration-centre/policies-procedures/academic-integrity/)**). You should follow the **COMP284 Lab Rules** **(https://cgi.csc.liv.ac.uk/~ullrich/COMP284/notes/COMP284LabRules.pdf)** to ensure that you do not breach the latter policy.

Feedback

You can expect individual feedback for this assignment about three weeks after the deadline. Generic feedback will be provided 7 days after the deadline. You should take that to be the date referred to in Section 5.1.4.(i) of the UG Computer Science Student Handbook.