Homework 4

Web-based Sudoku Puzzle Solver

This project gives hands on practice with JavaScript programming and updating an HTML page using the Document Object Model (DOM). In this project you will update the DOM to display a solved Sudoku puzzle. The solve function is done for you. You must implement the function that displays the board and the function that calls an API to display a new puzzle.

Sudoku Puzzles

The objective of Sudoku is to fill a 9x9 grid with digits so that each column, each row, and each of the nine 3x3 sub-grids that compose the grid (also called ”boxes”, ”blocks”, or ”regions”) contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution. To learn more about Sudoku and how to solve by hand see this short youtube video: https://www.youtube.com/watch?v=OtKxtvMUahA

Depth-First Search Solution

The code that solves the Sudoku puzzle is given to you using the following algorithm:

```
solve()
for each grid cell
   if grid cell is empty
      // we try possible numbers
      for numbers 1 – 9
          check if number satisfies Sudoku requirements
         if number satisfies Sudoku requirements
             set grid cell to number
             if(solve() == true)
                   return true; //we continue
             else
                 set grid cell back to 0 //try next number
     return false; //this is returned if no number works so we backtrack
return true; //returns true when all cells are filled with correct values
```

Project Set Up

Create a folder called hw4. Download starter code from Blackboard: index.html, sudoku.css, sudoku.js. and put the three files in hw4.

Your solution should go in sudoku.js only. The only file you need to modify is sudoku.js. Do not modify index.html. If you'd like to modify the sudoku.css to change the design of the table or button feel free, but it will not be graded.

Instructions

Display the Board

1. The index.html file contains the html of the table and the button. The original partially completed sudoku puzzle should be displayed on load.
2. When the user clicks the solve button, the table should display the solved puzzle.

Load a New Board

3. When the user clicks the Load New Puzzle button, an API call should be made to https://sudoku-api.vercel.app/api/dosuku
4. After the call to the API, JSON will be returned. Nested in the JSON is a new puzzle. Print out the JSON using console.log and determine how to drill down into the JSON to get a new 9x9 grid. Assign it to the array sol and a new puzzle will be displayed.

See this link for an example of what your solution should look like:

https://66fb389950fb4d5c74e870e7--earnest-dolphin-7ba212.netlify.app/

Submission

Upload your code to Netlify.

Submit sudoku.js to blackboard.

If Blackboard does not accept the JavaScript file, you can save your code as a text file and submit that. **In the text submission box**, enter the URL to your sudoku website on Netlify.

Grading

80 points - Implementation

35 points - On load index.html displays the original partially filled grid sudoku puzzle.

10 points - The puzzle is solved and displayed when the user clicks the solve button.

35 points – A new puzzle is retrieved from the API and displayed when the user clicks Load New Puzzle.

20 points

The code is in the correct files and uploaded correctly to Netlify and/or blackboard.

For more information on the API see https://sudoku-api.vercel.app/