Summer 25 Web Programming
Team Project 2: Fifteen Puzzle
Proposal Due Date 07/25/2025
Project Due Date 08/02/2025

Submission Requirements

You may discuss the assignments with other students in the class. However, as stated in the academic honesty policy, your written answers must be your own. Please list the names of any students you discussed the assignment with.

How to Submit

1. Log into **iCollege**.
2. Select the class to view its Dropbox folders.
3. Choose the correct folder for the given assignment and upload your file there.

Important: Please submit your link for **Project 2 URL via Codd Server, GITHUB REPO LINK, and PPT SLIDES** to iCollege. **NO Link = NO GRADE.** All members must submit as well as discuss a major JS feature/functionality that was implemented by you.

Overview

All groups have been assigned. Please refer to **iCollege** or **Discord** for any new additions. This project presents an excellent opportunity to enhance your teamwork skills, which are highly sought after in the professional realm. (Read *Becoming a Successful Team Member*.)

The project will involve delivering a presentation lasting approximately **15 to 20 minutes** on course-related materials or topics. Each group has the freedom to select a topic of interest, allowing creativity to flourish and showcasing your efforts to the class. While the project doesn't need to be overly complex, demonstrating exceptional logic is essential. I encourage everyone to explore their creativity and enjoy the process!

1st group meeting Agenda

1. **I encourage you to remain with your current team, allowing for partial changes if necessary as we did for Project 2.**
2. Designate a leader within your team who will serve as the main point of contact with the instructor.
3. Engage in a brainstorming session to generate and share ideas collectively.

4. Strategize and outline how your team will collaborate and communicate effectively throughout the project.
5. Select an individual responsible for integrating the various parts completed by team members into a cohesive whole.
6. Define and agree upon the specific responsibilities for each team member to ensure a clear distribution of tasks. For instance, determining who will handle which aspects of the project.

Here are the key directives for your project:

Suggestions for development and presentation

Team Coordination: Choose one team member as the leader responsible for coordinating the project and reporting to the instructor.

Presentation: Each team must prepare a presentation lasting 15 - 20 minutes. At the beginning of the presentation, the leader should present PowerPoint slides and the txt document that will list the url codd link along with the list of details below.

- Leader's Name
- Project Name
- Description: A one-sentence summary of your project
- All team members (last and first names) and their respective project responsibilities.
- Each team member should summarize the benefits of the project methodology noted below

Project Submission:

It is strongly (required) that, before the presentations, each team member copies the project files to the CODD server.

Create a folder containing all project files and place the code on GitHub. Ensure all the code work, from inception to completion, is uploaded on GitHub. Please note that we may randomly check GitHub at any time to review the work and any communication related to each assigned project.

Note: Your project should focus on HTML, CSS for front-end design, and PHP without the use of JavaScript or e-commerce-related concepts.

Should you have any questions or need further clarification, feel free to ask for assistance.

Feel free to structure your team in any way that suits your project's needs. One approach could involve roles like user, designer, coder/programmer, and tester. Alternatively, you might consider a setup with an architect/chief programmer overseeing a team of programmers, each dedicated to a specific part of the program.

The presentation format should be as follows:

PowerPoint Slide Show:

Use this to introduce the problem, outline the team structure, and discuss the project's objectives and challenges.

Demo Run of the Program:

Showcase the program in action. Provide a live demonstration to illustrate its functionality and how it addresses the identified problem.

Display and Explanation of Source Code via Slides:

Share the source code via slides to delve into the technical aspects of the program. Explain the key sections, methodologies, and any innovative or crucial parts of the codebase.

Remember, the goal is to present a comprehensive overview of the project, from the initial problem through the practical demonstration of the solution, and finally, a detailed exploration of the underlying code. This structure will help your audience understand your project's problem-solving process and technical implementation.

Required Key for the design see the following: >> Scrum methodology

To effectively execute the project, please adhere to the following guidelines within the Scrum methodology:

Brief Assessment of Scrum Benefits: Each team member **MUST** Submit a 1-ppt slide providing an assessment of how the Scrum framework has positively influenced your project.

Note:

- All team members act as designers collaborating with users to establish program requirements.
- Produce a user interface sketch.
- Develop the program design, specifying classes, fields, methods, objects, etc. Provide pseudo code for all methods.
- All team members are responsible for creating the interface and writing code.
- All team members participate as testers, creating a test plan that includes procedures, test data, bug tracking and reporting methods, and bug priority assignments. They might also assist in bug fixes.
- All team members must assemble the PowerPoint slide show, integrating inputs from the entire team.

Planning and Communication:

- Determine a schedule, estimating hours for each project phase.
- Define communication and coordination protocols (when, where, and how).
- Utilize available class time for teamwork, with support from me as needed. Effective communication via email is encouraged.

Decide the responsibilities of each team member. e.g. (To-do list) - Team Roles and Responsibilities:

Designers (All Members)

a) Collaborate with the User to define program requirements.
b) Create user interface sketches.
c) Design the program, outlining classes, fields, methods, objects, etc.

d) Develop pseudocode for all methods.

Programmers (All Members)

a) Implement the program by writing code for all components.

Testers (All Members)

b) If applicable, formulate a comprehensive test plan.
c) Include test procedures, test data, bug tracking, and reporting methods.
d) Prioritize identified bugs.
e) May assist in bug fixing.
f) Compile information from team members for the PowerPoint presentation.

Devops deployment methodology (Required)

a) Implement the DevOps deployment methodology, conducting daily sprint meetings.
b) Concentrate on specific work areas.
c) Provide a detailed summary of how the Kanban methodology was applied and its benefits. Include this information in the PowerPoint presentation with a labeled section on the advantages of the Scrum methodology. (State a different approach from your last project we will check)

Graduate Students

a) **You Must** introduce creative extra features or unique concepts into the project you MUST NOTE in your presentation.

Project Planning

b) Collaboratively establish a project schedule.
c) Estimate hours required for each project phase.
d) Determine communication methods, timing, and locations for coordination.
e) Utilize Discord as a primary communication platform for all team members.

Throughout the course, guidance and support will be available, including dedicated class time. Feel free to reach out for assistance, and utilize Discord as a means of communication and collaboration.

Presentation Day Requirements - Summary

a) Prepare a PowerPoint Presentation of 15- 20 Minutes - Please refer to the specified requirements below.
b) Utilize conference call recording software like Zoom, WebEx, or Microsoft Teams to deliver your presentation.
c) Establish a YouTube channel and post your video following the provided guidelines - Refer below for specific instructions.
d) Obtain the video link source and submit it to the designated Dropbox location listed on i-College.

PowerPoint slide show should include the following:

- ✓ User - statement of problem, and general requirements (inputs, outputs, etc.)
- ✓ Design - Overview of the solution, key design features, JS concepts, user interface, UML class diagrams, pseudo code using Transforms, Transitions, & Animation.
- ✓ Testing - (if applicable) how tested (e.g., test plan, data used, tracking and reporting bugs, bugs fixed/not fixed, etc.)
    - ✓ **Give your Group a Team Name that's tied to Web Technologies Development Project.**
    - ✓ You will choose the presenters in order. **All members must be involved in the presentation.** Interchange - One person will do the slide show, and a second team member will demo the or each team member may wish to present his own work.
    - ✓ Please use PowerPoint to present be sure to insert your code snippets with slides.

About the YouTube Channel

Create a YouTube channel that will be used to present your work

Title Video: i.e. "Name of the task CSS Project 03_TeamName "

- ✓ **This video should range from 15 - 20 minutes.**
- ✓ **Every team member must participate in this video and must cover a key feature.**
- ✓ **Create a channel at YouTube and name it as your group name**

List of links to use and determine your recording presentation software

1. Zoom
2. Microsoft Teams
3. WebEx
4. Additional Software you can consider
5. **Discord**

How to create a YouTube channel? Let's start with the basics

1. Sign into YouTube and click on the user icon at the top right of the screen.
2. Click on the gear icon to get to your account's YouTube Settings.
3. Click on Create a new channel.
4. Then choose "Use a business or other name"
5. Add your Brand name and click Create.

- ✓ Once ready, upload the video to your channel.
- ✓ Include the link to this channel with your submission and you will
- ✓ Incorporate the video for the core of your PowerPoint presentation.
- ✓ Use this channel for uploading future videos.

Grading Criteria: Please see the Rubix Excel sheet used in Project 1.

If you fulfill the specified requirements (please refer to the Requirements section above), you will be awarded full credit. To receive credit, your team must adhere to the designated submission guidelines for both the paper and presentations. Merely submitting the files to iCollege will not be considered adequate. It is imperative that you also upload the project to the CODD server, and all team members must ensure their projects are posted on Dropbox. Additionally, it is advisable for all members to share the Dropbox links to ensure accurate grading and reporting of your grades.

The Project title: Fifteen Puzzle

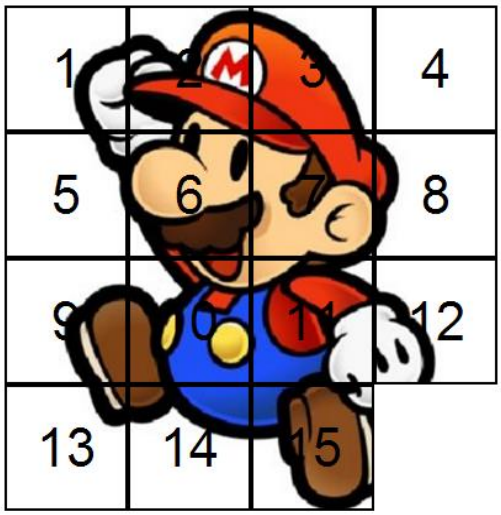
Suggestions for development and presentation

- You may organize your team any way you like. One way is user, designer, coder/programmer, and tester. Another way is an architect/chief programmer with a team of programmers each of whom works on one part of the program.
- The presentation could be structured as follows:
  - a PowerPoint slide show to introduce the problem
  - a demo run of the program
  - a display and explanation of some of the code
  - a question and answer period

This Project is about JavaScript's and event handling. You must match in appearance and behavior the following web page (between but not including the thick black lines):

Fifteen Puzzle

The goal of the fifteen puzzle is to un-jumble its fifteen squares by repeatedly making moves that slide squares into the empty space. How quickly can you solve it?



American puzzle author and mathematician Sam Loyd is often falsely credited with creating the puzzle; indeed, Loyd claimed from 1891 until his death in 1911 that he invented it. The puzzle was actually created around 1874 by Noyes Palmer Chapman, a postmaster in Canastota, New York.

Background Information:

The **"Fifteen Puzzle (Explanation 1)"** **(Explanation 2)** (more generally called the Sliding Puzzle) is a simple classic game consisting of a 4x4 grid of numbered squares with one square missing. The object of the game is to arrange the tiles into numerical order by repeatedly sliding a square that neighbors the missing square into its space.

You will write the JavaScript code for page fifteen.html that plays the Fifteen Puzzle. You create a background image of your choosing, displayed underneath the tiles of the board. Choose any image you like, so long as its tiles can be distinguished on the board. Please see the file names below which you should have on the Codd server:

- fifteen.js, the JavaScript code for your web page
- background.jpg, your background image, suitable for a puzzle of size 400x400px

You CAN create your own XHTML or CSS code in this assignment or instead, use the CSS we provide. Write JavaScript code that interacts with the page using the DOM. To create its appearance, write appropriate DOM code to change the styles of onscreen elements by setting classes, IDs, and/or style properties on them.

Milestone 1

Appearance Details:

In the center of the page is a set of tiles representing the playable Fifteen Puzzle game. Each tile is 100x100 pixels in total size, including a 2px black border around all four of its sides. (This leaves a 96x96 pixels visible area inside the tile.) Each tile displays a number from 1 to 15 in 32pt text using the default sans-serif font available on the system. When the page loads, initially the tiles are arranged in their correct order, top to bottom, left to right, with the missing square in the bottom-right. The tiles also should display a chunk of the image background.jpg, assumed to be located in the same folder as your page. Useful CSS code. < (GSU VPN may be needed to Access)

Please ensure that your background image appears on the 15 puzzle pieces. By adjusting the background position on each **div** which is a container, you'll be able to show a different piece of the background on each of the 15 puzzle pieces. One confusing thing about the background-position property is that the x/y offsets shift the background behind an element, not the element itself; this means that the offsets are the negation of what you may expect. For example, if you wanted a 100x100px div to show the top-right corner of a 400x400px background image, you would set its background position to -300px 0px.

Centered underneath the puzzle tiles is a Shuffle button that can be clicked to randomly rearrange the tiles of the puzzle. The last content on the page is a right-aligned paragraph containing two links to the W3C validators and (**optional**) JSLint. These are the same images and links as used in the previous assignments. The images should not have borders.

All other style elements on the page are subject to the preference of the web browser. **The screenshot in this document is a demo.**

Milestone 2

Behavior Details:

When the mouse button is pressed on a puzzle square, if that square is next to the blank square, it is moved into the blank space. If the square does not neighbor the blank square, no action occurs. Similarly, if the mouse is pressed on the empty square or elsewhere on the page, no action occurs.

Whenever the mouse cursor hovers over a square that can be moved (one that neighbors the blank square), its appearance should change. Its border color should change from black to red. Its text should be underlined and should be drawn in a green color of#006600. (This can be done by creating a CSS class called **movablepiece** that would represent these styles.) Once the cursor is no longer hovering over the square, its appearance should revert to its original state. Hovering the mouse over a square that cannot be moved has no effect.

When the Shuffle button is clicked, the tiles of the puzzle are randomized. The tiles must be rearranged into a solvable state. Please note that many states of the puzzle are not solvable; for example, it has been proven that the puzzle **cannot be solved** if you switch only its 14 and 15 tiles. We suggest that you generate a random solvable puzzle state by repeatedly choosing a random neighbor of the missing tile and sliding it onto the missing tile's space. A few hundred such random movements should produce a shuffled board. Your shuffle algorithm should be relatively efficient; if it takes more than a second to run or performs a large number of unnecessary tests and calls, you may lose points.

The game is not required to take any particular action when the puzzle has been won, that is when the tiles have been rearranged into the correct order. THIS PUZZLE MUST BE SOLVEABLE OR YOU FAIL!!!!

Milestone 3

Fifteen Puzzle: Database Integration and Enhanced Features

The "Fifteen Puzzle" project, as outlined in the provided document, primarily focuses on front-end development using HTML and CSS, with PHP for server-side logic, and explicitly notes the exclusion of JavaScript for certain aspects. While the current scope emphasizes core game mechanics and presentation, integrating a database like MySQL can significantly enhance the game's functionality, user experience, and administrative capabilities.

This **Milestone** proposes how MySQL database concepts will be applied to extend the Fifteen Puzzle, along with "**awesome**" additional features that would leverage this database integration, including the introduction of an administrative role.

MySQL Database Concepts and Design

A relational database like MySQL will allow for persistent storage of user data, game statistics, preferences, and content management. Here's the proposed schema:

1. users Table

This table would store information about registered players and administrators.

- **user_id**: INT (Primary Key, Auto-increment) - Unique identifier for each user.
- **username**: VARCHAR(50) (Unique, Not Null) - User's chosen username.
- **password_hash**: VARCHAR(255) (Not Null) - Hashed password for security.
- **email**: VARCHAR(100) (Unique, Not Null) - User's email address.
- **role**: ENUM('player', 'admin') (Default 'player', Not Null) - Defines the user's role (player or administrator).
- **registration_date**: DATETIME (Not Null) - Timestamp of user registration.
- **last_login**: DATETIME - Timestamp of the user's last login.

2. game_stats Table

This table would record the performance of each game played by users.

- **stat_id**: INT (Primary Key, Auto-increment) - Unique identifier for each game session record.
- **user_id**: INT (Foreign Key referencing users.user_id, Not Null) - Links the game session to a specific user.
- **puzzle_size**: VARCHAR(10) (e.g., '4x4', '5x5', '6x6', Not Null) - The size of the puzzle played.

- **time_taken_seconds**: INT (Not Null) - The time taken to solve the puzzle in seconds.
- **moves_count**: INT (Not Null) - The total number of moves made to solve the puzzle.
- **background_image_id**: INT (Foreign Key referencing background_images.image_id) - The ID of the background image used for this game.
- **win_status**: BOOLEAN (Not Null) - TRUE if the game was won, FALSE if abandoned/lost.
- **game_date**: DATETIME (Not Null) - Timestamp when the game was completed.

3. user_preferences Table

This table would store individual user settings and preferences.

- **preference_id**: INT (Primary Key, Auto-increment) - Unique identifier for each preference set.
- **user_id**: INT (Foreign Key referencing users.user_id, Unique, Not Null) - Links preferences to a specific user.
- **default_puzzle_size**: VARCHAR(10) (Default '4x4') - User's preferred default puzzle size.
- **preferred_background_image_id**: INT (Foreign Key referencing background_images.image_id) - User's preferred default background image.
- **sound_enabled**: BOOLEAN (Default TRUE) - User's preference for in-game sounds/music.
- **animations_enabled**: BOOLEAN (Default TRUE) - User's preference for animations.

4. background_images Table

This table would manage available background images for the puzzle. **Images themselves are NOT stored directly in the database.** Instead, their URLs are stored, and the images would reside on the web server or a CDN.

- **image_id**: INT (Primary Key, Auto-increment) - Unique identifier for each background image.
- **image_name**: VARCHAR(100) (Not Null) - A descriptive name for the image (e.g., 'Mario World', 'Nature Scene').
- **image_url**: VARCHAR(255) (Not Null) - The URL path to the image file.
- **is_active**: BOOLEAN (Default TRUE) - Flag to enable/disable an image (e.g., for admin to remove options).
- **uploaded_by_user_id**: INT (Foreign Key referencing users.user_id) - If user-uploaded, links to the uploader.

Additional Extra Features Leveraging Database Integration - Applying a MySQL backend, the Fifteen Puzzle can evolve from a standalone game into a more engaging and personalized experience.

Undergraduate Students: You are required to implement the Admin ROLE plus any two features from the provided list.

Graduate Students Only: You are required to implement the ADMIN ROLE plus any four features from the provided list.

---

1. **User Accounts and Secure Authentication:**
    - **Registration/Login:** Users can create accounts, log in securely using PHP for backend processing (hashing passwords, session management).
    - **Personalized Experience:** Game progress and preferences can be saved and loaded across sessions.
2. **Global and Personalized Leaderboards:**
    - **Best Times/Moves:** The game_stats table allows for querying and displaying leaderboards based on fastest times or fewest moves for different puzzle sizes.
    - **Friends Leaderboards:** (Advanced) If a "friends" system were implemented, users could compare scores with their friends.
3. **Customizable Game Experience:**
    - **Persistent Preferences:** Users' chosen puzzle sizes, background images, and sound settings (from user_preferences) would be remembered for future sessions.
    - **Multiple Backgrounds (Enhanced):** Beyond the current requirement of 4, the background_images table allows for an unlimited number of background images, managed centrally. Users could browse and select from a gallery.
4. **Game History and Analytics:**
    - **Player Progress:** Users can view their past game statistics, track their improvement over time, and see their personal bests.
    - **Overall Game Metrics:** Administrators can analyze overall game play data (most popular puzzle sizes, average completion times, etc.).
5. **Admin Role and Content Management System (CMS):**
    - **User Management:** An admin dashboard would allow administrators to view, edit, or deactivate user accounts (e.g., for moderation).

    - **Content Management:** Admins could add, update, or remove background images from the background_images table, making them available or unavailable to players without code changes.
    - **Game Statistics Oversight:** Admins could monitor overall game performance and identify trends.
    - **Announcements/News:** (Optional) An admin could post news or updates visible to all players.

6. **User-Generated Content (e.g., Custom Puzzle Images):**
    - Users could upload their own images (stored on the server, with URL in background_images table) to create personalized puzzles, fostering community engagement. This would require robust server-side validation and moderation capabilities, potentially managed by the admin role.

7. **Achievements and Badges:**
    - Based on game_stats, the system could automatically award achievements (e.g., "First Win," "Solved 10 Puzzles," "Fastest 4x4 Solver"). These could be stored in a new achievements table.

Admin Role Functionality

The role column in the users table would differentiate between regular players and administrators. An admin user would have access to a dedicated backend interface (PHP-driven) to perform the following actions:

- **User Account Management:**
    - View all registered users.
    - Edit user details (e.g., change username, reset password).
    - Deactivate or activate user accounts.
- **Background Image Management:**
    - Upload new background images (files saved to server, URL and metadata stored in background_images table).
    - Edit existing image details (name, active status).
    - Delete images (removes from database and server).
- **Game Statistics Monitoring:**
    - View aggregate game statistics (e.g., total games played, average time, most popular puzzle).

    - View individual player statistics.
    - Identify top players for leaderboards.
- **System Configuration (Advanced):**
    - Potentially modify game parameters (e.g., default puzzle size options, maximum moves for a "win").

Implementing these features will transform the Fifteen Puzzle into a more dynamic, interactive, and community-oriented web application, providing a richer experience for players and robust management tools for administrators.

Milestone 4

Key Features:

In addition to the previous requirements:

> You must specify the key features you used during your presentation failure to do so will result in points deductions on an average of 10 points each one that is missing Requirements

- **End-of-game notification**: Provide some sort of visual notification when the game has been won; that is, when the tiles have been rearranged into their original order. An alert is not sufficient; you should modify the appearance of the page. You may display an image(s) if you like, but there is no way to turn them in, so put them on your web space and use full path URLs when linking to them.
- **Ability to slide multiple squares at once**: Make it so that if you click any square in the empty square's row or column, even ones more than one spot away from the empty square, all squares between it and the empty square slide over. (Much more pleasant to play!) If you do this extra feature, make it so that all movable squares (including ones several rows or columns away from the empty square) are highlighted on hover as described before.
- **Animations and/or transitions**: Instead of each tile immediately appearing in its new position, make them animate. You can do any sort of animation or other styling you like, as long as the game ends up in the proper state after a reasonable amount of time.
- **Game time with some music file**: Keep track of the game time elapsed in seconds and the total number of moves, and when the puzzle has been solved, display them along with the best time/moves seen so far. The Music files should be a sound to create some adrenalin
- **Multiple backgrounds**: Provide several background images (at least 4) to choose from. The game should choose a random background on startup, and should have a UI (such as a select box) by which the player can choose a different image while playing. Host your additional backgrounds on the web server and link to them using full path URLs.
- **Different puzzle sizes**: Place a control on the board to allow the game to be broken apart in other sizes besides 4x4, such as 3x3, 6x6, 8x8, 10x10 etc..
- **Cheat button**: This will solve and show the shuffle that was created **not a static page** to display the titles in order.

- **UNDERGRAD – MUST IMPLEMENT ANY 2 OF THE EXTRA FEATURES**
- **Graduate Students Only – MUST IMPLEMENT ANY 3 OF THE EXTRA FEATURES**

At the top or bottom of your html file, display a comment saying which extra feature(s) you have completed. If you implement more than one, comment which one you want us to grade (the others will be ignored). Regardless of how many additions you implement, the main behavior and appearance should still work as specified.

Development Strategy and Hints:

Past students claim that this program is hard! We suggest the following development strategy:

- Make the fifteen puzzle pieces appear in the correct positions without any background behind them.
- Make the correct parts of the background show through behind each tile.
- Write the code that moves a tile when it is clicked from its current location to the empty square's location. Don't worry initially about whether the clicked tile is next to the empty square.
- Write code to determine whether a given square can move or not (whether it neighbors the empty square). Use this to implement the highlighting style that occurs when the user's mouse hovers over tiles that can be moved. This will require you to keep track of where the empty square is at all times.
- Write the shuffling algorithm. We suggest first implementing the code to perform a single random move; that is, randomly picking one square that neighbors the empty square and moving that square to the empty spot. Get it to do this one time when Shuffle is clicked, then work on doing it many times.

Here are some hints to help you solve particular aspects of the assignment:

- Many students have redundant code on this program because they don't create helper functions. You should consider writing functions for common operations, such as moving a particular square, or for determining whether a given square currently can be moved. The ***this*** keyword can be helpful for reducing redundancy.
- At some point you will find yourself needing to get access to the DOM object for the square at a particular row/column or x/y position. We suggest you write a function that accepts a row/column as parameter and returns the DOM object for the corresponding square. It may be helpful to you to give id values to each square, such as "square_2_3" for the square in row 2, column 3, so that you can more easily access the squares later in your JavaScript code. (Such id values would need to change as squares move.) You should use these IDs in appropriate ways. If you need to get access to the DOM object for the square at row 2, column 3, build the proper ID to find it. Another approach is to store the puzzle pieces into an array or to use $$ to find the piece with the right x/y position.
- You can convert a String into a number using the parseInt function. This also works for Strings that end with non-numeric content. For example, parseInt("20 percent") returns 20.
- You can generate a random number from 0 to N, or randomly choose between N choices, by saying **Math.floor(Math.random() * N).**
- I suggest that you do not explicitly make a div to represent the empty square. Keep track of where it is, either by row/column or by x/y position, but don't create an actual element for it.

- I suggest that you not store your squares in a 2-D array. This might seem to be an appropriate structure because of the 4x4 appearance of the grid, but it will be difficult to keep such an array up to date as the squares move.

Implementation and Grading:

Once again Submit your assignment to iCollege and post the link on each team member's course web site.

Please have a secondary link that will display a summary of work details for all group members as to who did what. Also, report all hours spent toward completion of the project.

You should make extra effort to minimize redundant code. If you are not cautious about avoiding redundant code, capturing common operations as functions, etc., it is easy for your code size and complexity to grow. You can reduce your code size by using the ***this*** keyword in your event handlers.

Follow reasonable stylistic guidelines. For example, utilize parameters and return values properly, correctly use indentation and spacing, and place a comment header on top of the file, atop every function explaining that function's behavior, and on complex code sections.

Global variables are allowed, but it is not appropriate to declare lots of them; values should be local as much as possible. If a particular value is used frequently throughout your code, declare it as a global "constant" variable named IN_UPPER_CASE and use the constant throughout your code.

You should only use material discussed in the textbook or lectures unless given permission by the instructor.

You should separate content (XHTML), presentation (CSS), and behavior (JS). As much as possible, your JS code should use styles from the CSS rather than manually setting each style property in the JS. Implement the behavior of each onscreen control using JavaScript event handlers defined in your script file. For full credit, you must write your code using unobtrusive JavaScript, so that no JavaScript code, onclick handlers, etc. are embedded into the XHTML code.

You may be able to find Fifteen Puzzle games written in JavaScript on the internet.

Please do not look at any such games. Using or borrowing ideas from such code is a violation of this course academic integrity policy. We have done searches to find several such games and will include them in our similarity detection process.

Remember to place the link solution to this assignment online on each team member’s web page.

Also, All Team member can use Google Plus or GIT HUB to collaborate. If you'd like me to see the communication in your circle, send me your Google plus info and I'll add you to the circle.