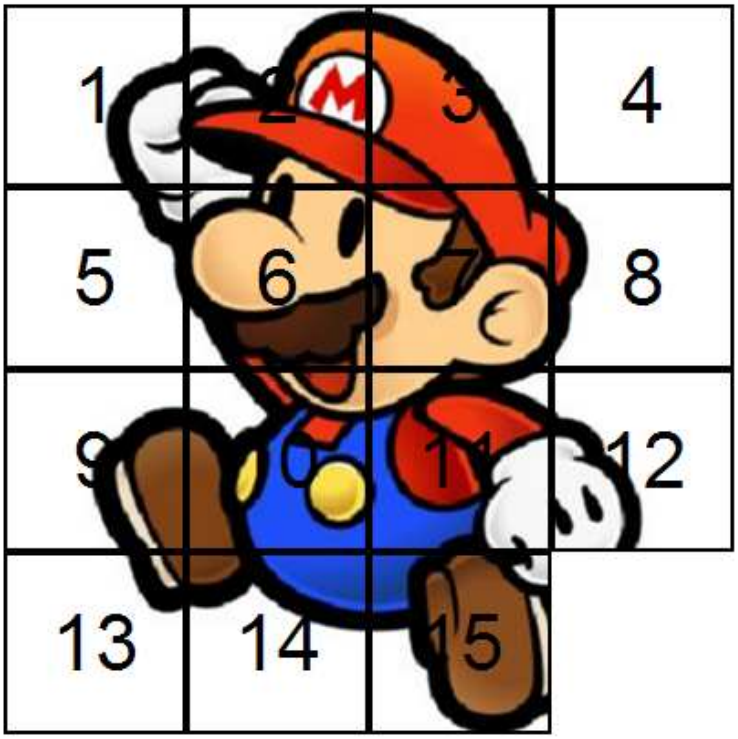
University of Washington, CSE 190 M, Spring 2007
Homework Assignment 4: Fifteen Puzzle
due Tuesday, April 24, 2007, 11:59pm electronically

This assignment tests your further understanding of Javascript and its interaction with XHTML and CSS, particularly events and the Document Object Model (DOM). You must match in appearance and behavior the following web page (between but not including the thick black lines):

CSE 190 M 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?



Shuffle

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" (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 empty space.

The main task for this assignment is to write the Javascript and CSS code for a web page to play the Fifteen Puzzle, stored in a file named `fifteen.html`. This page must match the appearance and behavior specified in this document, though some variations and add-ons are will be described.

You will also submit a background image of your own choosing. Your game should be configured to display this background image underneath its tiles. You can choose any image you like, so long as its tiles can be distinguished on the game board.

In total you will turn in the following files:

- `fifteen_style.css`, the style sheet for your web page
- `fifteen_script.js`, the Javascript code for your web page
- `background.jpg`, your background image (suggested filename; can also be in GIF or PNG format)

You will not submit any `.html` file or directly write any HTML code in this assignment. Instead, we will provide you with the HTML code to use, which should not be modified. (You can modify it temporarily on your machine for testing, but your code should work with the provided file unmodified.) To modify the page's behavior and content, write Javascript code that interacts with the page using the DOM. To modify its appearance, write appropriate CSS styles if possible or DOM code if necessary. We will provide you with a starter version of `fifteen_style.css` that you may modify in any way you like.

Appearance Details:

All images on the page and mentioned in the following text are hosted on the web in the folder: `http://www.cs.washington.edu/education/courses/cse190m/07sp/homework/4-fifteen_puzzle/images/`

The overall page has a white background. The preferred font for text on the page is the default serif font available on the system, in size 14pt. The top of the page contains a level 1 heading left-aligned within the page, with 10px of margin above and below it. Below this heading and at the bottom of the page are paragraphs describing the Fifteen Puzzle game, displayed using the browser's default paragraph formatting.

Between these paragraphs is the set of tiles representing the playable Fifteen Puzzle game. Each tile is 100x100 pixels in total size, with a 2px black border around all four of its sides. (This leaves 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 display a chunk of the image `background.png` (or another image of your choosing, as long as you turn it in).

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. These are the same images and links as used in the previous assignments. The images should not have borders.

When the page is resized, the only change that should occur onscreen is that the paragraphs might adjust their line wrapping if the width changes.

All other style elements on the page are subject to the preference of the web browser. The screenshots in this document were taken on Windows XP in Firefox 2.0, which may differ from your system.

The page's title text is `Fifteen Puzzle`. The page should also have a "Favorites icon" of `fifteen.gif`.

Behavior Details:

When the mouse button is pressed on a square of the puzzle, if that square is next to the blank square, it is moved into the blank space. If the square on which the mouse was pressed 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 border should change from black to red. Once the cursor is no longer hovering over the square, its border should revert to black. 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. 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.

The game takes no special action when the puzzle has been won, that is, when the tiles have been rearranged into their correct order.

Extra Features / Creativity Aspect:

To encourage you to be creative on this assignment, we will award you **+1 free late day** (up to a maximum of 2) for each of the following possible additions to your page. You are not required to do any of these.

- **Improved game flow**: The game is more fun if you can actually detect when the user has "won" it. Keep track of game time and number of moves, and show the best time/moves on the screen.
- **Ability to slide multiple squares at once**: In many implementations of this game, if you click aany 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. This is a nice way to let the user make multiple similar moves without manually clicking each tile.
- **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.
- **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 or 6x6.
- **Other**: If you have an idea for a creative addition to this program, please ask us and we may approve it for late day credit.

Note that regardless of how many of these additions you choose to implement, you may not submit the assignment more than 3 days past its listed due date. Also recall that you must make any such changes without modifying the HTML page file directly.

Implementation and Grading:

Submit your assignment online from the turnin area of the course web site. Because the turnin system does not work well with `.js` files, you should package your files into a `.zip` archive (instructions) and turn in this archive. For reference, our solution has 44 lines of CSS and 118 lines of Javascript.

Express all stylistic information on the page in CSS using your style sheet file. For full credit, your style sheet must successfully pass the W3C CSS validator. Format your CSS so that it is as readable as possible, similarly to the examples shown in class. Also place a comment in your CSS file containing your name and section.

Your Javascript code should follow reasonable stylistic guidelines similar to those you would follow on a CSE 14x programming assignment. In particular, you should minimize the number of global variables, utilize parameters and return values properly, correctly utilize the HTML DOM objects, correctly use indentation and spacing, and place a comment header at the top of your Javascript file and atop every function explaining that function's behavior. You should only material that has been discussed during the first four weeks of the course, unless given explicit permission from the instructor.

Three aspects of your Javascript style deserve particular emphasis. The first is that you should minimize redundant code. The second is that you should exercise good procedural decomposition, breaking down lengthy operations into multiple functions. The third is that you should not use Javascript as a substitute for CSS. As much as possible, place stylistic information into CSS stylesheets rather than placing it into Javascript code.

Here are some hints to help you solve particular aspects of the assignment. It may be helpful to you to give `id` values to each square of the board, such as `"square_2_3"` for the square in row 2, column 3, so that you can more easily access these squares later in your Javascript code. (Such `id` values would need to change as squares moved on the board.) Also recall that you can convert a String into an integer using the `parseInt` function. This function also works for Strings that begin with an integer and end with non-numeric content. For example, `parseInt("100px")` returns the number `100`. Also recall that you can generate a random number from 0 to MAX, or randomly choose between MAX choices, by saying `Math.floor(Math.random() * MAX)`.

Please do not place a solution to this assignment online on a publicly accessible (not password protected) web site. Doing so is a violation of the course academic integrity policy. Please see the course web site for directions on setting up password protection on your web space.

We realize that it is possible to find Fifteen Puzzle games written in Javascript on the internet. Please do not look for or examine source code of any such games. Examining such code or submitting it (or your own code derived from it) in any form is a violation of our course academic integrity policy. We have found many of the most common such games and will include them in our automatic similarity detection.