WEB-215 Project 11

Before You Begin

- Read Chapters 11 and 12 and complete all tutorials if you haven't already done so.
- Rename the downloaded/unzipped project folder to **Last-First-Proj11** (substitute your first/last name).

Examine the default browser rendering

- Open the HTML file in a browser and notice the following:
    - The page heading reads *Project 11 Puzzle* with no background color
    - The puzzle is complete
    - You can't drag the puzzle pieces around
- Click the **Scramble Puzzle** button and notice the following:
    - The page heading remains unchanged
    - The puzzle pieces have been scrambled
    - The pieces still can't be dragged around

Examine the default code

1. HTML
    a. Links already exist to the CSS, jQuery, and puzzle.js files.
    b. The puzzle is comprised of 9 `<li>` tags – each with an `id`.
        i. Note that the ID's indicate the correct order of the puzzle pieces.
        For those of you wondering – HTML 4 prohibited ID's from starting with a number. HTML5 removes that restriction. While it is now legal in HTML5, CSS doesn't support it. If you wrote a CSS rule to target #1, the rule would not work. So – the ID's used in this project are for JavaScript's use – not for CSS.
2. CSS
    a. The `.placeholder` class is the only rule not currently in effect. It should be used later to format the puzzle.
3. JavaScript (puzzle.js)
    a. DOCUMENT READY binds the click event to the HTML button. When the button is clicked, the `scramblePuzzle()` function is called. Additionally, the `puzzle()` function runs.
    b. `scramblePuzzle()` scrambles the puzzle pieces by randomly rearranging the `<li>` tags. It also changes the appearance of the `<h1>` and makes the `<ul>` sortable (if you use the jQuery `sortable` UI).
    c. Note the puzzle pieces are still not draggable/sortable because the jQuery UI has not been implemented.

What to do – in no particular order

Get Ready

1. First – there is no need to edit the existing HTML, CSS, `$('document').ready()`, or the `scramblePuzzle()` function. All of your work should be restricted to the `puzzle()` function.
2. Decide if you want to use the `draggable`, `droppable`, and/or `sortable` interactions. Then, download and link what you need from the jQuery UI page.

Write Code

3. Get the puzzle pieces movable. Use the jQuery UI (`draggable`, `droppable`, and/or `sortable`).
4. Set the options to:
    a. Apply the `.placeholder` class (already in the CSS) to identify the drop zone
    b. Display the cursor as the four-headed *move* icon
5. As soon as a user starts to drag a puzzle piece, the following should happen:
    a. `<h1>` text should change to "Puzzle in progress…"
    b. `<h1>` background color should change to red
6. As soon as a user drops a puzzle piece, the following should happen:
    a. Determine if the puzzle is solved. (Are the pieces in the correct order? Remember the `<li>` tag's `id's` should be in order if the puzzle is correct!)
    b. If the puzzle is not yet solved, do nothing.
    c. If the puzzle is solved (determining if it is solved is the challenging part), do the following:
        i. Change the `<h1>` text to "Complete!"
        ii. Change the `<h1>` background color to green
        iii. Stop the puzzle pieces from being draggable
7. There *should* be nothing to write for this next part. But there might be.
    a. After the puzzle is solved, you should be able to click the Scramble Puzzle button and play again. The `scramblePuzzle()` function already takes care of this, provided your `puzzle()` function used the `sortable` UI. If you used something other than the sortable UI, you will need to edit line 14 of the JavaScript code.
8. Delete all comments from the puzzle.js file.

Submission

- Remove the PDF instructions from your submission folder. Only project files should be submitted – not project instructions.
- Zip your **Last-First-Proj11** and submit it via Moodle.