CSSE 280: Forms, AJAX, jQuery Homework Exercise

Objectives

- You will implement the N-Squared puzzle in JavaScript using jQuery
- You will use Forms, AJAX, and jQuery to interact with a remote API and display the results of AJAX requests on the same page.

Table of Contents

N-Squared Puzzle with jQuery

Checkout the **nSquaredPuzzleJQuery** project from your Git repository. All your work for this exercise will be done is this project. You will commit and push your solutions back to your repo by the due date.

This assignment is about using jQuery and ES6 to solve the N-Squared Puzzle.

Puzzle Description

A description of the puzzle is given in the **nSquaredPuzzle** assignment that you completed during a previous homework exercise. If you need a refresher on how the puzzle works, you are encouraged to review the description there.

Implementation Details

All the work for this assignment must be done in the **nSquaredPuzzleJQuery** project.

You should not have to modify the HTML or CSS code.

Use **jQuery** and **ES6** to

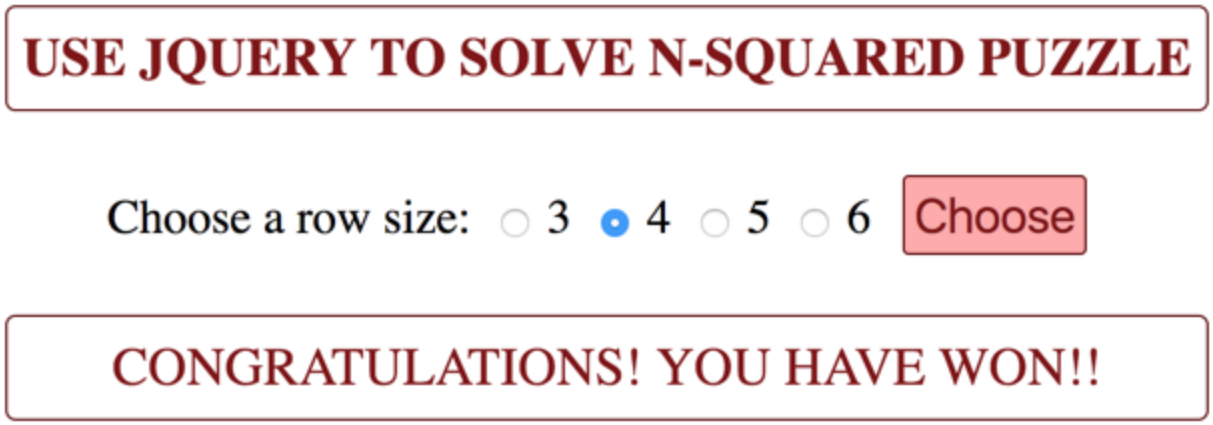
- ★ hide and show DOM elements,
- ★ swap tiles and DOM elements,
- ★ respond to user interactions,
- ★ move tiles legally, and
- ★ notify the user when the puzzle is solved.

Implement as many JavaScript functions and use as many jQuery features as needed to solve this problem.

A reasonable amount of initial code is available in the project you checked out.

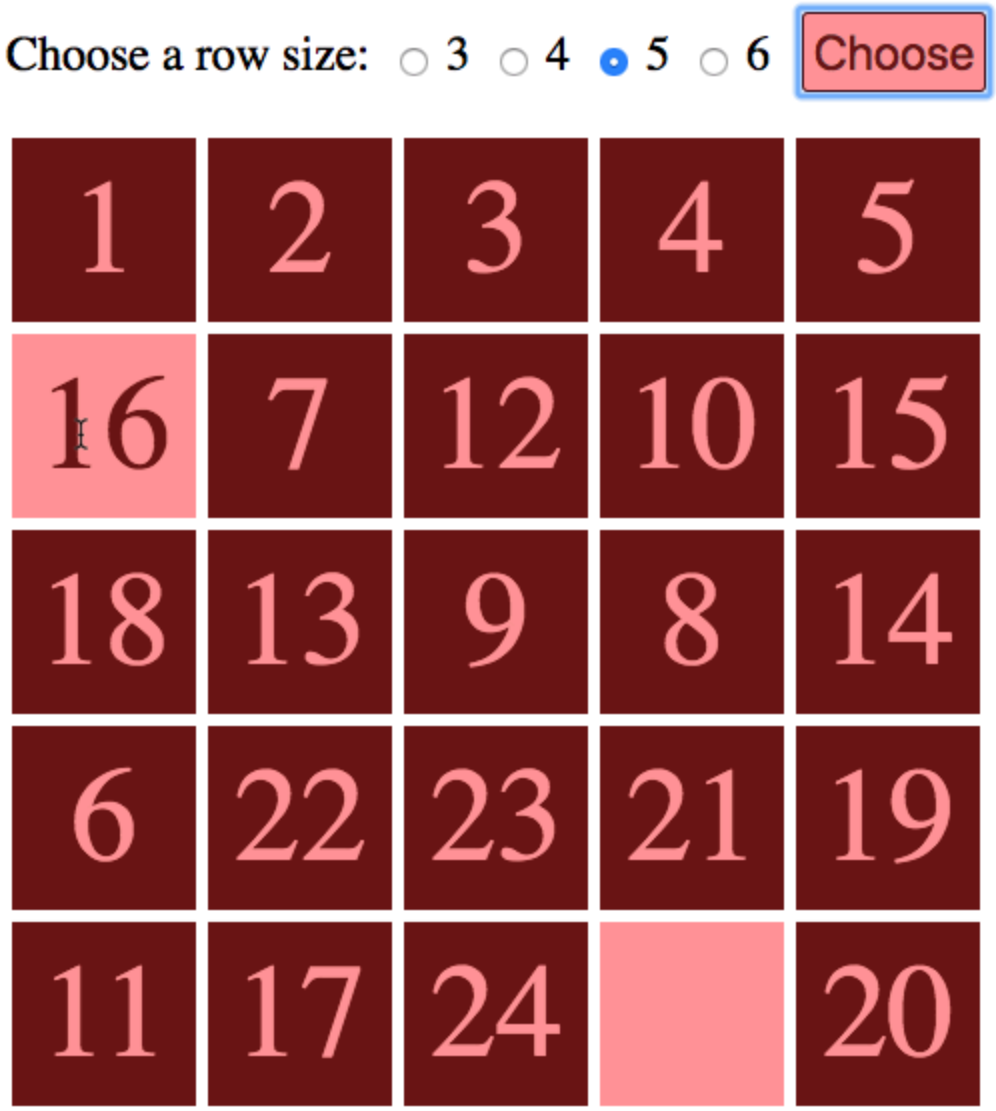
Guidelines and appearance

When you access the web page from which the puzzle will be solved, the figure below depicts what you will see.



The default row size should be 4. Clicking in a different checkbox will change the row size to the indicated value. When the Choose button is clicked by the user, a squared frame of numbered squares with numbers and the empty space in random positions in the frame should be displayed. The congratulatory message should **NOT** be visible. The figure below shows a sample of the game state that the user would see. Tile 16 is hovered over in this figure.

Note: **you should start with a board in the win state**, **and shuffle it by calling your *moveTile()* function**.



The board size (a.k.a frame size), border width, & font size are all given in the JavaScript tile.

When the user clicks on a tile or the **Choose** button described above, a click event is triggered. You can register an event handler to respond to a click event as follows:

```
const sizeEl =$('#sizeChoice);   // Get the Choose button
sizeEl.on('click', getSize);// Click the Choose button
```

OR

```
sizeEl.click(getSize);// Click the Choose button
```

Where **getSize** is the function called in response to the click event triggered on the DOM element with **id == "sizeChoice"**.

Code to exchange the locations of two DOM elements is partially completed in the .js file.

Forms and AJAX

Checkout the ***formsAndAJAX*** project from your Git repository. All your work for this exercise will be done is this project. You will commit and push your solutions back to your repo by the due date.

This assignment is about using HTML5 forms, AJAX, jQuery and ES6 to query an open movie database and display the results from AJAX calls to the same page.

Problem Description

You will create a simple HTML5 form that a user will use to provide a Title, Year, length of plot and response type expected to make AJAX GET requests to the open movie database for information on a movie of interest. The form should initially look as depicted below.

When the user clicks on the search button, **getFormData()** in **main.js** should respond to the click event by extracting the data from the form and using it to make an AJAX GET request.

Unsuccessful searches should clear the form and the dynamically generated content. **Optionally**, a “Movie not found.” message can be displayed where the dynamically generated content would otherwise be displayed.

Note: Only JSON responses should be processed. XML responses should be ignored.

Guidance and appearance

The response from the AJAX request should be displayed **below** the form to look like in the figures below. The success callback from your AJAX request should call **displayMovie(data)** to display the data

Title:	*The Avengers*
Year:	*2012*
Rated:	*PG-13*
Released:	*04 May 2012*
Runtime:	*143 min*
Genre:	*Action, Adventure, Sci-Fi*
Director:	*Joss Whedon*
Writer:	*Joss Whedon (screenplay), Zak Penn (story), Joss Whedon (story)*
Actors:	*Robert Downey Jr., Chris Evans, Mark Ruffalo, Chris Hemsworth*
Plot:	*Earth's mightiest heroes must come together and learn to fight as a team if they are to stop the mischievous Loki and his alien army from enslaving humanity.*
Language:	*English, Russian, Hindi*
Country:	*USA*
Awards:	*Nominated for 1 Oscar. Another 37 wins & 79 nominations.*
Poster:	
Ratings:	Internet Movie Database: *8.1/10* Rotten Tomatoes: *92%* Metacritic: *69/100*

Metascore:	*69*
imdbRating:	*8.1*
imdbVotes:	*1,065,150*
imdbID:	*tt0848228*
Type:	*movie*
DVD:	*25 Sep 2012*
BoxOffice:	*$623,279,547*
Production:	*Walt Disney Pictures*
Website:	*http://marvel.com/avengers_movie*
Response:	*True*

The width and height of the data displayed is determined by your device resolution and on the amount of data that is available for the movie requested.

CSS is provided to style the display of the results. When inspected with Chrome Devtools, here's what a sample of that looks like:

Note that <p> elements with nested spans are used to display most of the key-value pairs in the JSON response.

A element with nested paragraph elements is used to display the Ratings information because the value associated with the Ratings key is different from the value associated with other keys.

NOTE: All these elements should be added in your JavaScript file and they should be appended to the div element with id = "search-by-title-response".

When the user clicks the Reset button, the form should be cleared and the data displayed below the form should be removed. The form can be reset by calling the **resetForm()** function in response to a click event on the Reset button.

Bonus Credit

Implement the CSS classes used in the form to make the form look professional. The amount of bonus credit depends on how professional the form looks.

Submission

Commit and push your solutions to your Git repository in the project that you checked out.

Resources

The Resources column for sessions 16 and 17 from the course schedule provides valuable information that might be helpful to solving these problems.