Assignment #2 - JavaScript, jQuery and jQuery Plugins

This Assignment will be completed in your groups, however you must submit your homework with an individual readme file. Only the readme needs to be unique – the lab may be the same for each group member. Different groups must have different submissions however. As usual, all referenced materials (tutorials, guides, documentation, etc.) must be cited.

Groups will start this assignment in class, Thursday October 13th.

Assignment #2 – Hexed! Due Monday, October 24, 2016 at 11:59:59 pm (EoD)

For this assignment, using JavaScript and jQuery, you will be creating a single-player game used to help teach new Web practitioners hexidecimal color.

Collaboration

Groups will submit the same assignment to LMS, with your team number and all of your names in the readme; The Readme files are to be written up individually and contain a discussion of the lab as well as individual answers to the questions below.

jQuery Plugins

Your solution will be developed as a jQuery Plugin. Plugins are used to extend jQuery's capabilities: for example, all of jQuery UI's widgets are actually implemented as plugins. In this case, your plugin will create a new instance of the game inside of the target element.

Your plugin may only depend on jQuery and jQuery UI.

The Game: Hexed!

After including your plugin as a regular JavaScript file, calling $(Element).hexed(settings) will initialize the game in that Element, adding any necessary markup within (remember: you can add classes and ids to these created elements and style them using a separate style sheet to be included with your plugin).

The settings parameter accepts an object literal that defines gameplay.

difficulty: Ranges from 0-10, 0 being the easiest. Defaults to 5.
turns: Any positive number. Defaults to 10.

Gameplay

Players are be presented with a color swatch with RGB values generated at random, and will have to come as close as they can to guessing its hex value for each of the red, green and blue values. A series of three jQuery UI sliders will be presented below (have a look at the Vertical Slider and Simple Colorpicker examples), and can be used to adjust the hex value (00-FF), shown in a row of three text boxes below. When the sliders are moved, the hex values are updated, and when the hex values are updated, the sliders move accordingly (handling invalid input is left as a design decision).

After pressing the CheckIt! button, the submitted color will be displayed below, along with the percent error for each red, green and blue value. This error percentage will be used to calculate the score, which is based on how long it took to guess and how accurate the guess was (more below). Clicking "Next" will present a new color, until the predetermined number of turns have passed, at which point the final score is presented and the user is prompted to play again (using the same or different settings).

Scoring

To compute how far off a guess was from the actual value, take the average of the absolute value percentages off from each color component (so a response 1% off on red, -2% off on green and -3% off on blue nets an average 2% off from the actual result).

Percentage off is calculated as follow:

(|expected value – actual value| / 255) * 100

Once the percentage off from the solution is found, the scoring formula is as follows:

((15 – difficulty – percent_off) / (15 – difficulty)) * (15000 – milliseconds_taken)

After each guess, the score earned should be added to a visible running tally, along with how many points were earned for that color. If the score would be less than zero for a color, it should be counted as zero. Scores should not have more than 2 points precision. (ie round to the nearest 100^{th})

Readme.txt

In addition to providing your name and RCS at the top and citing any sources used at the bottom of your README.txt, answer the following questions:

1) What are the advantages to writing a jQuery plugin over regular JavaScript that uses jQuery?
2) Explain how your jQuery plugin adheres to best practices in JavaScript and jQuery development.
3) Does anything prevent you from POSTing high scores to a server on a different domain? If so, what? If not, how would we go about it (written explanation/pseudocode is fine here)?

Submission

Name your plugin as team#-hw2.js, and any associated CSS as team#-hw2.css, along with your README-RCSID.txt, contained in a ZIP file to LMS. The functioning program should be in an homework2 directory under your websys directory – zip that directory and submit it under the name Team#-RCSID-homework2.zip, where Team# is your team's number and RCSID is your RCSID.

Grading Rubric

Semantic HTML	10 Points
Use of CSS	10 Points
Game Initialization (Plugin)	10 Points
Game State	10 Points
Color Sliders	10 Points
Score Calculation	10 Points
Overall Presentation	10 points
Code Style	10 points
Readme	20 points
Total: 100 points	

Challenge: JSON & Web Storage

For a potential recovery of 15 points, We want to save high scores locally!

At the end of the game, provide a form for users to provide their name and save their score. Once the user clicks on the "Submit Score" button, use the Web Storage API to store a list of high scores as a JSON string in the user's localStorage (you'll have to parse and stringify the JSON each time you want to load and save the high score list). Each high score should include the following, collected at the end of each game:

- Player Name
- Difficulty
- Turns
- Score
- Timestamp

Create a new page, Team#Scores.html, using HTML5. This page will read from localStorage to retrieve the high scores list, and print them in a neatly formatted table, sorted by score, then timestamp.

In your README.txt, answer the following question:

4) Now that you're used Web Storage, what other information would you store there in other Web-based applications? Is there any information you wouldn't store?

Challenge Rubric

Use of JSON	3 points
Use of HTML5 & Web Storage	7 points
Readme	5 points

Your solution should be in a folder under your websys folder named homework2

Zip this folder – it will include everything you needed to get the game running.

Submit as a ZIP file, named Team#-RCSid-homework2.zip