Mon June 25, 2018

Code Immersives Term 2 Project: DUE DATE: Mon July 16 - Three Options

Counts 60% of grade for **Course05 Advanced Javascript** and is in lieu of equivalently weighted exams. The other 40% of the grade is, as always, based on attendance.

OPTION 1: Javascript OOP Application of your choice: Max score 120 points

Ideas: a voting app, a survey, a quiz, a game, a learning interaction, a slideshow, a to-do list, a chat box — or anything else. Project should NOT be a full-fledged website — a single app, living on one page, with a narrow purpose is most appropriate. Think *widget*. You can feel free to use any past class projects as a point of departure to build upon. You do NOT have to hit all 20 items on this list to get an "A" :

- 90 points out of 120 points (75%) earns an A-
- 95 points (79%) is an A
- 100 points (83%) earns an A+

The application does NOT have to be sophisticated or advanced… just needs to hit items on the checklist, look and perform decently, and make sense as a thing that we can interact with. *Less is more!* A simple app that works will score higher than a complicated or ambitious one that does not work and cannot be demonstrated properly. Max point values itemized. Partial credit.

1. OOP Application with class and constructor(), constructor takes at least one argument : +25
2. Use of map, filter, reduce (any one) : +5
3. AJAX-JSON to load data into app : +5
4. Dynamic HTML elements using document.createElement('div'), etc. for min 3 elements : +5
5. Use of dynamic image with new Image() instantiation : +5
6. Use of dynamic audio with new Audio() or document.getElementById('audio') : +5
7. Use at least two Math methods, Math.random(), Math.floor(), etc. : +5
8. Use of at least 2 array methods: splice(), slice(), push(), pop(), join(), shift(), unshift() : +5
9. Use of at least 1 string method : charAt(), indexOf(), split() : +5
10. Use of Object Sorting by Property Key : +5
11. Use of setTimeout() or setInterval() / clearInterval() : +5
12. Distribution: FTP to Website : +5
13. Distribution: Push to Github : +5
14. CRUD call to Database using PHP-MySQL : +5
15. CRUD call that saves result of user interaction (vote, score, etc) : +5
16. Frontend: use of CSS Grid or Flexbox : +5
17. Frontend: Overall appearance and use of styles / CSS : +5
18. Technical communication: ReadMe.txt file that describes app: +5
19. Technical communication: pseudo-code algorithm : +5
20. Technical communication: public presentation of project (10 minutes) : +5

OPTION 2: Javascript Jukebox OOP Application: Max score 120 points

Instead of OPTION 1, you can do the Jukebox / MP3 Player, using the provided starter files. The Jukebox has the following functionality that you will need to implement on a point-scoring basis: **120 pts**

1. OOP **Jukebox.js** file, with **constructor()**, taking exactly two arguments, as indicated by the instantiation line in the html file: **new Jukebox('app', songs)** if all this Jukebox ever can do is say Hello World, you get: +25 points
2. HTML page with only one DOM element: ** </app>** : +10
3. Use of drag and drop to drag a quarter into the coin slot and play a song : +10
4. Dynamically generated HTML to render the 12 Jukebox buttons +5
5. Getting the big background image to appear in the main app div: +5
6. Buttons play the correct song when clicked. Album cover appears the whole time: +5
7. Dynamically generated div to display song data with each song played: +5
8. Dynamic generated div to display album covers; the code that does the transform perspective is in blackjack.css : +5
9. Dynamically generated audio player with document.createElement('audio') and with this required method called on the audio object: setAttribute("controls", "controls"): +5
10. Math methods, Math.random() and Math.floor() to generate random numbers to play random songs. : +5
11. Array method splice() to remove the just-played song from the array so that it doesn't get repeated by Autoplay All : +5
12. Autoplay All triggered by dragging the quarter into the slot — all 12 songs will play, one right after another with no repeats: how to make another song start as soon as one is done? +5
13. Mouseover-mouseout functions to show-hide album cover when mousing over a button : +5 On mouseout of a button, if a song is already playing, revert to showing that album cover: +5
14. Frontend: Overall appearance and use of styles — all CSS may be done in JS : +5
15. Technical communication: ReadMe.txt file that describes app: +5
16. Technical communication: pseudo-code algorithm : +5
17. Technical communication: public presentation of project (10 minutes) : +5

IMPORTANT: PUH-LEASE DO NOT SHARE OR UPLOAD THESE COPYRIGHTED SONGS AND/OR ALBUM COVER ART — NO FTP TO WEBSITE OR PUSHING TO GITHUB !

OPTION 3:
Do Both projects and get an average of the 2 PLUS a 15 point (whole letter grade) bonus added to score. So, if you got a 70 on one and an 80 on the other, you would get 75 + 15 = 90.

Which one should you do?
The Jukebox has a more focused checklist than does OPTION 1: There is no backend, no CRUD, no AJAX-JSON, no map, filter, reduce methods, no sorting of objects by keys, no FTP, no GitHub, no CSS Grid, no CSS Flexbox. All that said, however, the Jukebox does pose its own challenges and is no means easy to implement… after all it's Advanced Javascript OOP !

… **Or should you do both?**
If you have time, doing both projects would be an EXCELLENT idea. The two checklists are quite different, which means that between the two projects, you would get quite a workout in many aspects of not only Javascript, but also HTML, CSS, AJAX-JSON and PHP-MySQL. The combo OPTION 3 also opens the door to a max score of 135. Anything above 125 is an A+++

Whatever you choose, good luck! jQuery will continue in lab through Thursday June 28, but from July 2 - 12 all labs will be reserved for project work. That is quite a bit of time, so if you can finish one with time to spare, you really should challenge yourself to do the other project as well. Good luck!