CMSI 370-01
Interaction Design
Fall 2015

Assignment 1029 (due 1103) Feedback

Note that, as a condition for the due date extension, you were still expected to commit something by 1029. This will factor into your *4f* proficiency.

Flanders Lorton

florton / freeflowingvortex@gmail.com

Notes while running (asterisks indicate major observations):

- *** Extra effort at getting a web service relay up and running noted here. +(*4a*, *4d*)
- Nice and clean layout, effectively combining text and image content… +(*3a*)
- …though I think the text content could have been displayed a tad better: e.g., distinct typography between labels and actual values; alternative displays for specific data types like dates, numbers/percentages, durations, etc. (*3a*, *4a*)
- *** Not the best error handling though—things stall when a non-existent player is added. (*3a*, *3b*, *4a*)
- Fun easter egg when looking up your account! **:)** (and I spotted that before I looked at your code, which made it even more fun) +(*3a*, *4a*)
- If the APIs allow it, typeahead functionality would be a good incremental improvement to the lookup user interface. (*3a*, *3b*, *4a*)

Code review:

1. Thank you for using spaces to indent all of your code! +(*4c*)
2. For consistency of notation, also use `var proxy = function () { …` when defining functions. (this is not a *hard* rule, but my own preference, for the aforementioned reason of consistency [and a few others we can talk about sometime if you like]) (*4c*)
3. For function definitions, place a space between `function` and the argument parenthetical. Think of it as a function statement, but without the name in between…there's still a space there, right? (*4c*)
4. JavaScript convention is to start local variable names with a lowercase letter. (*4c*)
5. Semicolons are indeed optional in JavaScript, but stay consistent. (*4c*)
6. Unless adjacent to parentheses, have a space before and after braces. (*4c*)
7. Same with binary operators, as well as assignment. (*4c*)
8. If you use 2 spaces per indent for HTML files, you can afford to indent `head` and `body`. (*4c*)
9. Loading Bootstrap locally isn't really necessary, except in some cases; I would not consider this assignment to be one of them. (*4b*)
10. On the one hand, ideally a custom style like this should still be loaded from a separate CSS file, even if it is this short. On the other, the issue of body width should be handled by Bootstrap—I'm unclear on why this was even considered necessary. (*4a*, *4b*)
11. OK, I see now—if you read the Bootstrap documentation, it says that you should place everything in a `container`. That will take care of the width issue for you. (*3a*, *4d*)
12. Missed indentation here. (*4c*)
13. *** The `center` tag is obsolete—note how it is a *view* element rather than a *model*. Centering should be done in CSS. In Bootstrap, the `text-centered` class takes care of most centering cases, and you can always turn to custom CSS for all others. (*4b*, *4d*)

CMSI 370-01
Interaction Design
Fall 2015

Assignment 1029 (due 1103) Feedback

Note that, as a condition for the due date extension, you were still expected to commit something by 1029. This will factor into your *4f* proficiency.

14. *** You did follow the recommended `$(function () { … });` top-level structure here, but not completely. The wrapping function needs to cover the *entire* code base. Otherwise only that single line is guaranteed to execute after the whole page has been loaded. Plus, all other symbols (e.g., `newImage`, `processTime`, `processPlaytime`) now pollute the top-level namespace (go ahead, type `processPlaytime` directly into the console and see what appears). (*4a*, *4b*, *4d*)
15. Space after `if` (and most other reserved words) please. (*4c*)
16. Break even single-line `if` clauses into multiple lines. (*4c*)
17. In JavaScript, triple equals (===) is the appropriate equality operator. Double-equals is not sufficiently strict and is ultimately a language design flaw. (*4a*, *4c*)
18. Learn how to do these in jQuery; your code will be more compact and fluent. (*4c*, *4d*)
19. Using `$(document).ready` has no effect here—go ahead, try it without this "wrapper." It appears there is a misunderstanding on what this does; look it up on the jQuery documentation when you get the chance, then ask me if things remain unclear. This, combined with note 14, indicates that some clarification/re-reading is called for. (*3b*, *4a*, *4d*)
20. Don't let your code run to arbitrarily long lines. Typically maximum "right margin" these days is 120 characters. Many editors will show you this boundary so that you don't have to think too much about it. Overly long strings can be shortened by inserting concatenations. (*4c*)
21. You have an uncalled-for hardcode in your URL—you won't always be running off localhost! This will break the moment you try to host your stuff somewhere else. Relative URLs may work here, or at the very least, make this easy to change. (*4b*)
22. *** Here is where your error handling would start—if the search yielded no result, the `players` array will have zero elements. Always be cognizant of potential error cases. (*3b*, *4a*)
23. *** These attributes are obsolete. Clues for checking their validity: (a) look them up and see if they are in HTML5; (b) if they pertain to visual properties (and these do), there is a very good chance that they are gone (like `center`). All such HTML identifiers have been moved to CSS. (*4b*, *4d*)
24. Make use of JavaScript truthiness/falsiness—avoid this comparison, and if this is somehow needed specifically, explain it in a comment. (*4a*)
25. An `else` clause is still in the same statement as the preceding `if`, so don't break them up. (*4c*)
26. In particular, look at `jQuery.clone` when you need to create repeated patterns of HTML. (*3a*, *4a*, *4d*)
27. **From this point, I will stop individually marking repeated notes—I think you've seen enough that you know what falls into similar situations. Only new notes after this marker.**
28. Why the change to single quotes here? (*4c*)
29. This is also covered by note #21, and furthermore, observe how the `sid` parameter is redundant. I think I've pointed this out to you already; you can see it in the request. (*4b*)
30. Inappropriate hardcode: what makes 570 so special? Better to give it a descriptive name. (*4b*, *4c*)
31. *** All told, your `submit` button holds the bulk of your logic but is unnecessarily long. It needs to be broken up into more cohesive units. You show an initial level of this with your `newImage`, `processTime`, `processPlaytime`, and `place` functions, and overall you need more of that. For example, each `getJSON` call is

CMSI 370-01

Interaction Design

Fall 2015

Assignment 1029 (due 1103) Feedback

Note that, as a condition for the due date extension, you were still expected to commit something by 1029. This will factor into your *4f* proficiency.

> a self-contained network interaction; those are good candidates for separation. Portions that process the raw response data into the data that you will actually display are also candidates. Remember that decomposition into functions is not solely a matter of encapsulating repeated operations (you got that part right); it is also a form of abstraction, allowing the code reader to see the broader strokes of what you are doing, and letting them choose when to go into detail. I won't go as far as saying that "all functions must be less than *xxx* lines," but in general if your code starts doing another, distinct sequence of activities, chances are good that you need a function call. (*4b*, *4c*)

3a — | …Good overall idea for presenting the data, with a few under-the-hood points of improvement.

3b — | …Error-handling can be better (well, it should exist!), plus your event handlers (counting network responses here) should be decomposed into functions.

4a — + …Overall functionality is excellent, including extra props for the web service relay.

4b — | …Code is generally cleanly structured, with various pinpoint areas of improvement.

4c — / …A good number of suboptimal or inconsistent code presentation choices here, as seen in the code review notes.

4d — | …Overall a good effort at looking things up, with the main pitfall dwelling in some missed jQuery functionality that would have improved your code's conciseness and fluency.

4e — Decent number of commits here, and your messages are mostly descriptive (the low end being "tweaks" and "tiny fixes"). This generally meets the bar, though I would say see if you can become more consistent with your messages, as we progress further. (+)

4f — Started well before 1029, submitted on time. (+)

Updated feedback and proficiencies based on 2015-12-08—*note, this is outside the allowed 2-week resubmission window from the syllabus, but it will be considered as an exception*:

3a — | …Primary points for this outcome, Bootstrap `container` and jQuery `clone`, not addressed.

3b — + …Improved error handling and better function decomposition seen.

4b — | …Assorted improvements noted, but a major pain point remains: globally-scoped names. As a widely *un*recommended practice, this proficiency cannot be a + as long as globals are around.

4c — + …Code presentation is significantly improved.

4d — | …Changes were not as much about new information as just paying more attention to detail in the existing code.