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.

Mondo Yamaguchi

myamagu1 / mondo.yamaguchi@gmail.com

Notes while running (asterisks indicate major observations):

- Your overall design comes across as clean and functional, if a little utilitarian. However, in the spirit of this being an "operational console" for the API, utilitarian is acceptable if it is executed well and the presentation is polished. (*3a*)
- Further, it appears that you have chosen to unconditionally constrain your layout to a specific width, which may work well on mobile but wastes space on desktop. This is unfortunate because one of the very points of Bootstrap is to provide a minimal-effort *responsive* layout system that expands/contracts according to the available window width. (*3a*, *4d*)
- In contrast, the buttons are a little too wide relative to the layout. I can see why you made that choice, but if you look at most button layouts (and guidelines documents), there tends to be a limit on how wide they can be. Otherwise they become indistinguishable from, say, banners or headings. (*3a*)
- Good job on learning how to do clean, Bootstrap forms and tables! As additional refinement for your tables, note that numeric values like level and money tend to be easier to understand if right-justified. (*3a*)
- *** Here is a major missing behavior: dynamic updates. When a character is created, edited, or deleted, that action is not reflected in the list of characters without a re-hide/show. On the one hand, the layout of the interface does not imply this this should necessarily happen, but on the other hand, most interfaces that deal with lists of data (as this one is) do tend to work that way (the polished ones, at least). (*3a*, *3b*, *4a*)
- *** Another major gap: data validation. All buttons remain enabled all the time, but they are not always applicable. For example, one should not be able to click the *Create* button if insufficient information has been placed in the "Create a new character" fields. Ditto for editing and deleting—unless an ID is provided at a minimum, the user should not be allowed to trigger an edit or delete operation. (*3a*, *3b*, *4a*)
- When viewing a single character, the tabular view is less appropriate. After all, there is just one character! I would have made this an opportunity to play with Bootstrap's other layout/display classes to put together something more like a "character profile" or "character card" display. (*3a*, *4a*)
- *** This does not stick out so much given the way the user interface is laid out, but if you pause to think about it, no modern list-of-data interface performs an edit or delete operation by asking the user to supply an ID. Instead, with the list of data already present, the user should be allowed to just choose the character to edit or delete, without having to copy-paste the ID. Allowing the user to make the choice is also less error prone—i.e., it avoids operations using an invalid ID. Remember that the menus-forms-dialogs interaction style is all about *recognition* rather than *recall*. (*3a*, *3b*, *4a*, *4d*)
- *** The character display does not clean up after itself: every time you click *Show*, the list of characters unconditionally appends to the list, even if the list already shows characters. (*3a*, *3b*)

Code review:

1. *** Indent with spaces, not tabs. (*4c*)
2. *** The `center` and `font` tags are obsolete—note how they pertain to a *view* rather than a *model*. Centering and fonts should be done in CSS. The same goes for view-specific attributes like `align`. In Bootstrap, the

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.

`text-centered` class takes care of most centering cases, and you can always turn to custom CSS for all others. (*4b*, *4d*)

3. Miscoded tag: the slash should be in the beginning, not the end. (*4a*)
4. Avoid inline style attributes—these should be placed in a separate CSS file. You may need to assign additional classes so the styles get assigned to just the right element(s). Or, scan the Bootstrap documentation for built-in classes that you may find useful. For example, I know that there are certain classes already defined to produce a specific text follow and/or accompanying background. (*4b*)
5. "`table`" is not the best ID for a table element! Give it something more descriptive. (*4c*)
6. *** I'm seeing a pattern now with the `width: 500px` inline style—it's everywhere! This very fact should have raised a huge red flag for you, either triggering additional reading so you can figure out how to avoid it, or if you couldn't figure it out yourself, having you ask me. Develop a sense for when a piece of code appears to repeat itself unnecessarily—it is a sign that there is probably a better way to do it. (*3a*, *4b*, *4d*)
7. Looks like you have a dangling tag there. And sometimes capitalized too? (*4a*)
8. Avoid names with numbers! This is a sign that your names are not sufficiently descriptive. (*4b*, *4c*)
9. A common [negative] side effect of not-so-descriptive names (note 8) is: names with numbers (note 8)! Note that a more descriptive naming scheme would have avoided that. (*4b*, *4c*)
10. *** From a DRY-ness perspective, the add and edit sections require some thought. If they are largely the same, differing only in that the edit area modifies an existing character, then the code is easier to maintain if you actually base the two sections on the same code. You would need to write code that "clones" (jQuery has a function for that) then "adapts" (you'd have to write this one yourself) the forms between adding and editing (e.g., presence of an ID field, adjustments to the labels and placeholders, etc.), but in exchange you will have one code base for displaying and reading player properties. Based on the functionality, I would say the latter is what matches in this situation. (*4b*)
11. *** A partial consequence of note 10 is that your IDs are replicated, and if you are not careful, you might miss some adjustments. Your `label for` attributes in the character modification section are an example of this—you forgot to adjust them after copy/pasting the character creation code! (*4a*, *4b*)
12. Concatenated HTML code is functional, but not always the easiest to maintain. Look into a combination of jQuery's `clone`, `find`, and `text` methods to give your code more structure. (*4b*, *4d*)
13. Note also how your "infinite character list" behavior would be easy to fix here—just empty the table before adding more rows to it! (*3a*, *4a*, *4d*)
14. See, you *do* know about `empty`. That makes the "infinite character list" oversight even more glaring. (*4a*)
15. 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*)
16. *** This is one possible place where the error checking would go. Before diving into the web service API invocation, examine the data you are about to send first, and reject anything that would cause an error. Better yet, check the data *as the user modifies it*, so that you can preemptively enable or disable the action button depending on input validity. (*3b*, *4a*)

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.

17. Don't allow lines to get excessively long. A good maximum line length these days is 120 characters. Get your editor to help you here; many of them let you set such a limit. (*4c*)

3a — / …Generally clean layout, but not exactly following the intent of Bootstrap and employing some obsolete tags and attributes.

3b — / …Fundamental event handling is there, but error handling/reporting/avoidance, plus displays that don't clean up or update well, are big gaps.

4a — / …Functionality is there but very basic; next-level functions like the aforementioned errors, better use of selections instead of manually-typed IDs, and more appropriate displays beyond tabular formats are all notably missing.

4b — / …Glaring mix-up of MVC concerns, particular M and V (i.e., inline styles; obsolete tags and attributes). There is also the question of how add/modify code should be organized.

4c — / …In addition to the tabs, your code presentation has some spacing inconsistencies, names with number suffixes, and an occasional tendency for lines to get too long.

4d — | …You did a decent job with picking up some new Bootstrap that was not shown in class, but judging from some layout coding choices there was more to learn.

4e — Good pacing and spacing of commits, with descriptive messages. Keep doing it that way **:)** But do remember to push, too! (**+**)

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