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.

Chris Franco *cf7 / spe861@gmail.com*

Notes while running (asterisks indicate major observations):

- Better coordination/feedback is needed between the channel selection radio buttons and the message-posting panel—i.e., when no channel is selected, posting should be visibly disabled. (*3a*)
- Ditto with the team list viz the invitation to a channel. (*3a*)
- Overall layout shows some Bootstrap use, but can use some additional design thought. Aside from the coordination already mentioned above, maybe the channel selection check boxes can be collapsed or displayed as a modal, so that they don't dominate so much vertical space at the top. It seems like the channel displays should be the stars of the show because they reflect the chat streams. The other sections, for posting a message and inviting team members, should then be supportive of the central content area. Dependent information such as channel or user selection should be integrated into their actual action areas of typing a message or inviting a user. (*3a*)

Code review:

1. You indented with spaces across the board—the right choice! (*4c*)
2. For HTML, it is OK to set the indent to 2 spaces due to the typically deep nesting that happens with these kinds of files. (*4c*)
3. 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*)
4. Oops, misspelled end tag! (*4a*, *4c*)
5. Bootstrap has a predefined `class` called `hidden`, which obviates the need for an explicit `display: none` inline style. (*4d*)
6. Avoid unnecessary comments. This one is particularly unnecessary because it's pretty obvious that you have a URL. Parameters are also pretty clearly parameters (for someone who knows what `getJSON` does) so that comment is not necessary either. (*4c*)
7. In JavaScript, there isn't a real need to preallocate an array to a particular length. Just `[]` suffices. (*4a*, *4c*)
8. Consider using `map` here. It will make the code more concise. (*4a*, *4c*)
9. This one, in turn, is a good candidate for `reduce`. (*4a*, *4c*)
10. Stick with `===` and `!==` for comparisons. (*4a*)
11. Break your code into multiple lines when an individual line is too long for the current medium. A typical modern limit is a column width of 120 characters. (*4c*)
12. Is `toString` necessary here? Seems like `result.user.name` is probably already a string. (*4a*, *4c*)
13. jQuery has an `empty` method which is more fluent than setting an element's `html` content to the empty string (*4a*, *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. Notice here that you have multiple calls to the same jQuery object. When you see this pattern, you can use *currying* to make the code more compact: (*4c*, *4d*)
```
$("#settings-status-display")
    .empty()
    .append("* Please select channels to open *");
```
15. Conversely, when you are doing the same thing to multiple elements, consider grouping them into a single [composite] selector: (*4c*, *4d*)
```
$("#settings-status-display, #refresh-status, #post-status-display").empty();
```
16. From this point, note that many other portions of the code that correspond to the notes in this code review so far. Presumably you can recognize additional applicable instances.
17. Oops, indented too far here. (*4c*)
18. Looks like you want to use `forEach` here. (*4a*, *4c*)
19. Consider using the CSS `transition` property for this effect. (*3a*, *4c*)
20. This is what many refer to as "callback hell"—multiple texted network requests and inline response-handling functions. You can alleviate this somewhat by breaking out the callbacks into named functions of their own. (*4a*, *4b*)

3a — | …Functional layout but not necessarily intuitive or robust.

3b — + …You generally set up events properly. A point of improvement, though minor because this is your first time with this, is to alleviate callback hell.

4a — + …Quite functional, despite the not-so-optimal design.

4b — | …Yep this is mainly those inline styles.

4c — | …Code is generally structured well but can benefit from learning more JavaScript and jQuery idioms. Learning these (many listed above) will shorten your code.

4d — | …Good job with figuring out the Slack API on your own; on the other hand, it doesn't look like you explored jQuery and Bootstrap much.

4e — Version controlling like a pro! Pacing and messages are both excellent here. (+)

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