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.

Dustin Kane

ranneyd / dustinpkane@gmail.com

Notes while running (asterisks indicate major observations):

- Nicely done, very cohesive and well-executed! +(*3a*, *3b*, *4a*, *4d*)
- Bootstrap is used well overall, and nicely integrated to Twitch's own widgets. +(*3a*, *4a*, *4d*)
- Nothing much else to say, really…pretty much operates as advertised. +(*3a*, *3b*, *4a*, *4d*)
- The "Options" modal can use a few refinements: most importantly, better validation for *width*, *height*, *top*, and *left*, and less importantly, more vertical space between rows. (*3a*, *3b*, *4a*)
- The channel meta data (viewers, total views, followers) can use a little bit more layout, perhaps a tabular arrangement or enclosure in a `well` or `panel`? More of a refinement really than anything else. (*3a*, *4d*)

Code review:

1. You indented with spaces on HTML, but not CSS nor JavaScript. (*4c*)
2. If you use 2 spaces per indent, you can afford to indent `head` and `body`. (*4c*)
3. *** Unless you have a really good reason, the current practice is to place JavaScript at the *end* of the `body` element. Primary reasons are: (a) processing JavaScript holds up page rendering, so putting this in the beginning will keep the page content from appearing sooner; (b) a lot of JavaScript depends on the full instantiation of the page anyway, so top-loading it runs the risk of accessing elements that have not been created yet. (*3a*, *4a*, *4b*, *4d*)
4. Addendum to note #2: If you use 2 spaces per indent, you can afford to indent correctly **:)** (*4c*)
5. Acknowledged; yes, that would be one of the [few] reasons that justify inline styles (as you know now, I do this in the *boxes* code for the same reason). The comment explaining this is also appropriate. +(*4b*, *4d*)
6. Now *this* inline style is avoidable: the `navbar-right` class takes care of it. When not in a `nav`, Bootstrap also has a workalike class for this called `pull-right`. (*3a*, *4d*)
7. Yes, this is a common resize implementation pitfall. The fix is indeed to track a resizing drag from an element that covers the entire area, and not the element being resized at all. +(*3a*, *3b*)
8. Don't allow lines to get excessively long—there is no guarantee that your code will be displayed in something that wraps lines automatically. 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*)
9. One helpful rule with code spacing is to "space like you're being proofread"—i.e., unless syntactically meaningful, apply whitespace the way you would to natural language content. That means adding space with most punctuation. There are exceptions, but this is a nice starting point. (*4c*)
10. Corollary to note #9: CSS is more readable if you add a space after the colons (to separate property names from their assigned values). (*4c*)
11. Another one for readability: blank lines between CSS blocks. (*4c*)
12. *** You didn't follow the recommended `$(function () { … });` top-level structure here, and thus all variables defined at this layer (e.g., `global_channel`, `channels`, `resetChannels`) now pollute the top-level

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.

namespace. Not recommended unless you have a reusable library here (and even then, modern JavaScript libraries have mechanisms for avoiding that, too). (*4a*, *4b*, *4d*)

13. 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*)
14. 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*)
15. Unless adjacent to same-sided parentheses, have a space before and after braces. (*4c*)
16. Space after `if` (and most other reserved words) please. (*4c*)
17. An `else` clause is still in the same statement as the preceding `if`, so don't break them up. (*4c*)
18. In JavaScript, triple equals (`===`) is the appropriate equality operator. Double-equals is not sufficiently strict and is ultimately a language design flaw. (*4a*, *4c*)
19. For consistency of notation, use `var functionName = 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*)
20. Suggested alternative: (*4b*, *4c*)

```
var KNOWN_TYPES = [ "width", "height", "top", "left" ];
$("#chat-div").css(KNOWN_TYPES.indexOf(type) + 1 ? type : "opacity", val);
```

21. It's more readable and less error prone to always enclose potential compound blocks of code in braces, even if said block is just one line. (*4c*)
22. Why is this being assigned to `caption`? It's already an argument to the function. Plus, the resulting variable ends up in top-level namespace. (*4a*, *4b*)
23. How about: (*4b*, *4c*)

```
channels.forEach(addChannel);
```

3a — **+**

3b — **+**

4a — **+**

4b — **|** …A few minor rearrangements and refactorings are justifiably called for, as stated in some code review notes. Most importantly, wrap the entire *main.js* codebase inside a closure.

4c — **|** …First there's the tabs. Beyond that, things are generally readable but can use some additional spit and polish as mentioned in the code review notes.

4d — **+**

4e — Version control, you've got that down. (**+**)

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