## Part 2: Comparison exercise (30% of grade)

Compare custom responsive design, created without any frameworks as we did at the start of this course, with one of the big three responsive design frameworks (UIkit, Foundation, Bootstrap) or jQuery Mobile. Other frameworks may be approved; ask your question in the [Assignment 3 Questions](https://canvas.harvard.edu/courses/19812/discussion_topics/159821) thread. Framework should include CSS and JavaScript and should not focus on just one area (like only a grid system).

**Rationale**: It's a code course -- why write a paper? Because when I read capstone and thesis proposals, it's clear that students don't understand why they're choosing one technology over another. This is good practice in structured thinking about technology choice. Even if you don't complete a degree at Extension, you will be asked about technology choices throughout your career. It's nice to have the skill set to critically evaluate all choices and pick the best one for the problem at hand.

1. **Create a comparison grid with at least 5 criteria**. You may have as many criteria as you wish, but 5 is the minimum. State what each criteria is clearly. State how that criteria is implemented/addressed (or not) in the items you're comparing. Which is better based on that criteria?

* Try to go a bit beyond the obvious and terse in your comparison. Example of the obvious:
  + Problem: Creating columns of equal heights
  + Foundation: comes with Equalizer component
  + Custom: Must download and include your own JS solution
  + Which is better: it depends!
* A better way of dealing with this would be:
  + Problem: Addressing common responsive design problems requiring a JavaScript-based solution, like equal height columns or responsive images
  + Foundation: ships with several Javascript-based components to address common responsive design problems. All components are designed to work together without conflict and integrate easily with the framework using HTML attributes.
  + Custom: Must research and identify Javascript-based components to address common responsive design problems. Potential exists for conflicts between components. May not integrate with HTML attributes, meaning you must tweak or modify JavaScript to work with your HTML. Alternatively, you could code your own, which adds to development time.
  + Which is better: ...
* In other words, don't be a community college student. Go beyond and show me what a Harvard graduate student does when thinking about a list of criteria.

2. Once this list of 5 criteria is generated, then **write a 1-2 page paper explaining which is better,** responsive design frameworks or custom responsive design.

* If you have a clear winner for most situations, defend your answer. Why is your choice the best? Why should everyone use it?
* If you don't have a clear winner, or if you think the answer is "it depends", explain. What does it depend on? What is an ideal project for each approach?

Remember **this is a critical thinking exercise** driven by your opinions of what's important when building a responsive website. There is no correct answer, but there are answers not sufficiently supported by data. An A answer will demonstrate the data (the criteria grid) that supports your opinion (the paper).

Grading criteria for the paper:

* Please no more than 2 pages. Say it in fewer words. I'm not going to play games with fonts and margins. I'm asking you to write clearly and compactly to make your argument.
* Good grammar, spelling, and punctuation count. You're Harvard grad students, for goodness sake.
* Have you made a good, supported argument?
* Have you made an argument that someone could follow who may not know much about responsive design? Imagine you're talking to a CTO or other technical manager who may not write the code for this, but who may have a reasonable technical background. (You don't need to explain what HTML, CSS, Sass, and compiling are, for example.)
* Hint: management considerations might be worth considering -- documentation, development time, cost of ownership, technical debt, stability of a given open source project, etc., in addition to the more obvious technical considerations.