Most of these resources cover the same topics as the class, but from different perspectives, in a different orders, and using different styles of pseudocode. The variation in viewpoints is intentional; having ideas explained several different ways is more helpful than always seeing them from the same perspective. But the other differences can be problematic.

To address the rearrangement of material, I've tried to list readings under the days that we will cover their content and also include descriptions of what part of the day's content they correspond to.

As for the pseudocode, it may take some additional background to read other styles. We use a style first codified by Christophe Fiorio and since used by both mathematicians and theoretical computer scientists; it might reasonably be called the Fiorio style. In contrast, most of these resources use ad-hoc or ALGOL-like styles. For this reason, I've included some readings that introduce ALGOL-style pseudocode.

Finally, if you find additional resources that you think should be added to this list, send me a link via e-mail and a description of which parts correspond to which days. If I agree, I'll make the change and award an additional 1% to your final grade (up to a maximum of 3%).

August 26: Basic Factoring

Some of this content anticipates the next lecture.

- `http://www.eecs.wsu.edu/~cs150/tdd.htm` **by Diane Law**

  This is a very nice high-level introduction to the process of factoring. It ends with a long example in ad-hoc, but highly legible pseudocode.

- `http://ee.hawaii.edu/~tep/EE160/Book/PDF/Chapter1.pdf` **by Bharat Kinariwala and Tep Dobry**

  Section 1.3.1 gives an overview of factoring and basic operations. The presentation uses structural diagrams instead of outlines, and there are flow charts to supplement the ad-hoc pseudocode.

- `http://www.scifac.ru.ac.za/javabook/ch02.htm` **by Pat Terry**

  Section 2.1 goes over factoring informally as well as introducing ALGOL-style pseudocode.

August 28: Loops

- `http://www.eecs.wsu.edu/~cs150/tdd.htm` **by Diane Law**

  Diane's example is worth revisiting as it contains a factoring into variably many quotient subproblems.

- `http://www.scifac.ru.ac.za/javabook/ch02.htm` **by Pat Terry**

  Section 2.3 works through a factoring that leads to a loop and further discusses ALGOL-style pseudocode.

  Sections 2.4.3–2.4.6 go over the different ways to write loops in ALGOL-style pseudocode.

  Section 2.5 gives some guidance about writing loops and mentions some common mistakes. It also talks about recursive algorithm design, which is another perspective on quotient subproblems.

  Section 2.6 works through a more complicated factoring involving loops.

September 2: Basic Dynamic Programming

- `http://20bits.com/article/introduction-to-dynamic-programming` **by Jesse Farmer**

  This is a pretty accessible walkthrough of dynamic programming, though it does cover a little more content than we saw in class, and the examples are in Python, which makes some of them a little tricky to read.

September 4: Bisection Example

- `http://en.wikipedia.org/wiki/Bisection_method` **by various authors**

  This is the Wikipedia article on the example problem.

September 9: Maps and Strings

The first two readings refer to objects rather than maps. The differences between objects and maps don't really come up until the course following this one, nor do they affect the applicability of these readings.

- ***Eloquent JavaScript*** **by Marijn Haverbeke**

  Chapter 4 from "Data Sets" through "Objects" presents maps in JavaScript. The section "Methods" requires knowledge from the next lecture, but can be safely skimmed on the first read.

- `http://www.scifac.ru.ac.za/javabook/ch02.htm` **by P. D. Terry**

  Section 2.2 gives an informal description of why objects are useful. The same arguments apply to maps.

- `http://en.wikipedia.org/wiki/Associative_array` **by various authors**

  This is the Wikipedia article on maps.

September 16: Functions

- `http://www.eecs.wsu.edu/~cs150/reading/functions.htm` **by Diane Law**

  The material up through the section "Two Kinds of Functions" explains functions well. The rest of the material is specific to the C programming language and not applicable to our class.

- ***Eloquent JavaScript*** **by Marijn Haverbeke**

  Chapter 3, up through the section titled "Parameters and Scopes" plus the section "The Call Stack" cover the same content as the lecture, but in JavaScript syntax.

  The section "Functions as Values" gives some warnings about confusing functions with the variables that contain them.

The section titled "Growing functions" gives advice on when and when not to use functions.

The section "Functions and Side Effects" explains the difference between pure and impure functions. However, note that the claim "There'd be no way to write a pure version of `console.log`" is actually incorrect; there's a trick taught in advanced computer science courses (using an I/O monad) that the author did not consider.

Chapter 4 up through "The Final Analysis" combines maps and functions to solve a problem titled "The Weresquirrel". Note that the author's answer depends on a quirk of JavaScript (and also many other languages) called "mutability", which we won't get to until the next unit; it's discussed in the section "Mutability".

Chapter 5, from the beginning up through the section "Abstracting Array Traversal", covers higher-order functions. (The following section "Higher-Order Functions" depends on material from the next lecture, and, oddly enough, does not add much to the discussion of higher-order functions.)

The sections from "Filtering an Array" through "Great-Great-Great-Great-..." give examples where higher-order functions are useful.

- `http://www.scifac.ru.ac.za/javabook/ch02.htm` **by P. D. Terry**

  Section 2.8 argues the merit of using first-order functions.

September 23: Closures and Captures

- ***Eloquent JavaScript*** **by Marijn Haverbeke**

  The sections "Nested Scope" and "Closure" in Chapter 3 explain capturing.

  The section "Binding" in Chapter 5 shows a JavaScript shorthand for creating certain kinds of closures.

- `http://en.wikipedia.org/wiki/Closure_%28computer_programming%29` **by various authors**

  This is the Wikipedia article on closures.

September 25: Applications and Web Programming

- `http://www.usability.gov/how-to-and-tools/methods/scenarios.html`, `http://www.usability.gov/how-to-and-tools/methods/use-cases.html`, **and** `http://www.usability.gov/how-to-and-tools/methods/requirements.html` **by the Department of Health and Human Services**

  These pages cover documentation of scenarios, use cases, and requirements.

- `https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Canvas_tutorial` **by various authors**

  This is the MDN canvas tutorial, which is worth looking at before starting your photo framer.

- ***Eloquent JavaScript* by Marijn Haverbeke**

  The section "Compatibility and the Browser Wars" in Chapter 12 presents caveats worth knowing, especially since they also apply to the lecture and handout material.

  Chapters 1–4 cover JavaScript syntax.

  The section "HTML" in Chapter 12 covers HTML.

  The sections "Styling" and "Cascading Styles" in Chapter 13 discuss CSS.

  Chapter 13 covers the parts of the DOM you need to know (and some related parts) in the sections "Document Structure", "Finding elements", "Changing the Document", and "Attributes". The section "Layout" may also be useful reading. (Note that the recommendation to "[prefix] the names of such made-up attributes with `data-`" is in fact standard; you should always follow this advice.)

  Chapter 14, except for the bit about web workers, covers information on events that is relevant to the course.

  Chapter 16 may be useful as a supplement to the MDN canvas tutorial.

October 2: Validation, Verification, and Debugging

- `http://www.inf.ed.ac.uk/teaching/courses/st/2009-2010/resources/category_partition_1.html` **by Conrad Hughes**

  This page walks through an example of the Category-Partition Method. It uses the terms "independently testable feature (ITF)", which in our case is always a function, and "`int`", which is an integer with limited magnitude.

- `http://www.eecs.wsu.edu/~cs150/prog/debug.htm` **by Diane Law**

  While not all of this content is applicable to our class, it contains a wealth of good debugging advice.

- ***Eloquent JavaScript* by Marijn Haverbeke**

  Chapter 8, up through the section "Debugging", contains further advice on dealing with bugs.

October 9: Search

- `http://www.redblobgames.com/pathfinding/a-star/introduction.html` **by Amit Patel**

  This page provides an interactive overview of search, primarily in the context of tile-based game maps.

October 14: Discrete-Event Simulation

- `http://en.wikipedia.org/wiki/Discrete_event_simulation` **by various authors**

  This is the Wikipedia page on discrete-event simulation.

