CSCI-E3: Introduction to Web Programming Using Javascript

Course Number/Term: CSCI E-3, Spring 2015
Meeting Time: Online (no set meeting times)
Location:

Instructor: Larry Bouthillier
lbouthillier@fas.harvard.edu
Teaching Fellows: Each student will be assigned to a TF
Office Hours: TBD
Section Meetings: Multiple section meetings led by course TFs will be scheduled, including in-person meetings in Cambridge as well as online meetings using Web conferencing, starting in week 2.

Course website: https://canvas.harvard.edu/courses/1847
[revision 2014.10.24]

Overview

This course provides an introduction to web development by way of the essential language and runtime environment that powers modern web interfaces. This is an entry-level programming course, and no prior programming experience is assumed. Exposure to basic HTML and CSS will be helpful, but is not required. Students with no HTML experience will face a steeper learning curve, and are encouraged to spend some time with online HTML learning resources such as CodeAcademy prior to the start of this class.

Through a series of examples and projects, students learn basic programming concepts while building an understanding of the power and complexities of Javascript, which can perplex even experienced web developers. The course provides a solid foundation in computer programming in Javascript: syntax and data structures, conditionals, objects, scope and closures, AJAX, the DOM, and event handling. Students gain an understanding of the popular libraries that power rich web applications such as JQuery and others. Upon completion, students are prepared to use Javascript in their projects, write their own or extend existing Javascript libraries, and build rich web applications using these powerful tools.

Goals

This course is designed to introduce students to programming in general, and then specifically, to explore the use of Javascript programming to add complex behavior to Web sites and Web applications. Students will learn:

- Foundational Skills:
  - be able to write, deploy, debug and run Javascript code in the context of client-side Web pages
  - have an understanding of the structure and syntax of programming languages, with a focus on Javascript's unique capabilities and techniques
  - algorithmic thinking, and be able to break a complex problem into smaller, solvable chunks
- Applications of Javascript to solve real-world problems:
  - be able to write Javascript programs to add useful behavior to Web pages
  - be able to use and extend popular libraries like JQuery
  - Javascript's role in Web architecture
- Learning how to learn
  - be able to understand and use common Javascript references to discover and use new APIs and information
  - be able to construct a well-written and complete request for programming help, and to answer a request for help

The course will consist of four units. Each unit will focus on a particular aspect of programming:

1. Programming fundamentals
2. The Javascript language
3. Javascript and the behavior of Web pages
4. Javascript libraries and advanced applications

Who Should Take This Course

While no programming experience is required for this course, it will help if you are comfortable with your computer: sending and receiving emails, attaching files, navigating the Web, finding your way around the file system (Windows Explorer or the Mac Finder). Basic knowledge of HTML will be very helpful. While the HTML isn't the focus of the course, and students will not be evaluated on their ability to write complex HTML, we'll be using simple HTML in many examples and assignments.

Some students in this course will have some experience with programming and Web development. Others will be starting with no programming experience at all. No matter what your experience level coming in, the goal of this course will be to build your skills starting from where you're at.

If you've never done any programming, you'll learn some basics that will apply to any programming language. If you've done some Web development before and used "found Javascript" in your pages, this course will contain some review. But you'll learn some of the basics you may have missed in your previous experience, and then get into some of the nuances of Javascript that you need to know to get to writing libraries and modifying your own scripts. It's these nuances that define real expertise in Javascript.

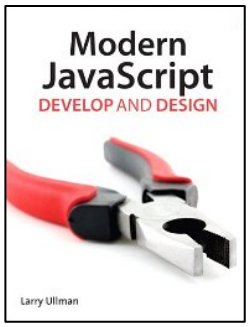
Why Did We Choose Javascript?

Let's face it – many of us started using Javascript to spiff up our Web pages without really understanding how it works. Sure - for simple things, it's often possible to copy-and-paste and putter around with the code and make things work.

But to really understand what Javascript can do, why JS libraries are written like they are, how to build Web experiences with complex behavior, we have to first go back to some basics that apply to nearly all computer programming languages, and then explore some of the features that make Javascript somewhat unique among programming languages. This creates an excellent opportunity to explore the fundamentals of programming while building practical, useful skills in a popular area of software development.

Course Materials

This book is strongly recommended for students in this course, and there will be references to its pages in every week's lessons. The book is available to students from the Harvard Library Safari Books Online subscription, but there's nothing quite like a physical copy of your own.



Modern Javascript: Develop and Design

by Larry Ullman
ISBN-10: **0321812522**

Available from Amazon, Barnes and Noble, and other booksellers
Also available from the Harvard University Library via Safari Books Online (login with HUID and PIN): http://proquest.safaribooksonline.com.ezp-prod1.hul.harvard.edu/book/programming/javascript/9780132905848

How to Take This Course

Though this course is an online course, it is not a self-paced course. Materials and assignments will be published on a weekly schedule. Each week will consist of:

- A set of course Canvas pages that weave together readings, code examples, brief video lessons, web references and other material to deliver the content and concepts of the week's topic. There are no long-format video lectures – this course is specifically built for online delivery, with all of its video produced specifically for this version of the course. The professor will be present in the course on an ongoing basis.
- Readings from the assigned textbook and various Web resources
- An optional one-hour section meeting led by one of the course Teaching Fellows.
- Learning to program is a bit like learning to play a musical instrument or learning to speak a foreign language. You can't learn to do it without spending time at the keyboard: designing, coding, debugging, and occasionally getting frustrated along the way. To this end, each Unit will contain a Practice Set assignment that requires you to do research, design solutions, and/or write small bits of code.
- Most weeks will include a peer assessment activity, in which students review Practice Set submissions by other students (anonymously). This activity increases student's familiarity with viewing code and recognizing its structure; as well as exposes students to a variety of possible solutions and approaches to a single problem.
- Interaction in the class discussion area on Piazza. Piazza will be the primary place for informal discussion, Q&A, and other interaction between instructors, TAs, and students.

Unit Problem Sets: The Unit Problem Sets will contain multiple problems of increasing difficulty, and it's expected that the problems will be done over several week's time. These problems will be designed to provide experience with real-world applications of the Javascript concepts learned in the unit. Students will have several weeks to complete these assignments.

In addition, there will be a final project. Students taking the course for graduate credit will have an additional requirement for each Unit Homework and the project.

Expected Time Commitment

The standard classroom version of this course included a two-hour lecture each week, plus a one-hour optional section, and the assigned readings and problems. The online course will not take less time than the classroom course. The time commitment will vary according to students' prior programming experience, as well as their motivation to push themselves to the most solid understanding of the course material. That said some reasonable targets for weekly student time commitment follow:

- Two to three (undistracted) hours to go through the lessons in Canvas each week, reading carefully, examining and understanding the example code, watching the video lessons and demonstrations.
- One hour for section meeting (optional, but highly recommended), at which TAs will work through code examples, present additional material that may help with Unit Problem Sets, answer questions and generally offer support.
- Up to one hour on the practice sets, which are designed to exercise your understanding of individual concepts introduced in the lesson.
- Unit Problem sets will generally have deadlines 2-5 weeks after they are made available to students, so having discipline to do some work each week will make a huge difference to your workload. Assume 1-5 hours per week, depending on the problem set.

Assignments & Evaluation

Description	Points
Weekly Activities (small assignments, practice sets, peer review)	150 points
Four Unit Problem Set assignments over the course of the term (150 points each)	600 points
Final Project	150 points
TOTAL	**900 points**

Assignment Submission: Weekly activities such as Practice Sets will typically be due Mondays at 11:59pm and will not be accepted late. Together, the Practice Sets are designed to be worth about a full letter grade of your final course grade. Missing the deadline on a couple due to work or family commitments will not damage your final grade very much, but doing them regularly will provide essential practice, keep you engaged with the course week-to-week, and enhance your final grade.

Unit Problem Sets will be due at 11:59pm on the published due date, which will always be on a Monday before the next Unit starts. We recognize that students have lives that may sometimes present challenges to getting homework done on time, therefore each student is given 5 late days for homework at the beginning of the semester. A late day extends the individual homework project deadline by 24 hours without penalty. You can use them all at once on a single assignment, here and there for an extra day when you need it, or not at all.

For homework that's late in excess of the five "grace" days, there will be a 10% penalty for homework less than 2 days late, and 20% for less than five days late. Homework more than 5 days late will not be accepted.

The final project is due at 11:59pm the Monday night of the final class week. No late submissions will be accepted on the final project.

Weekly Schedule

This is a general guide to the course structure and content. Weekly details are subject to change.

WEEK	LESSON DATE	TOPICS
UNIT 1: Programming Fundamentals		
1	January 27	Introduction Course objectives, structure and processes Communication, Assignments, Support Program design: Plan before you code
2	February 3	Basics of the Javascript environment • The browser and Web page execution cycle • Authoring and debugging code • The roles and relationships between HTML, CSS and Javascript
UNIT 2: The Javascript Language		

WEEK	LESSON DATE	TOPICS
3	February 10	Basic data types, variables, objects, and mathematical operations
4	February 17	Control structures, conditionals, looping, functions
5	February 24	Data and data structures • Objects • Arrays • Dates and other built-in data objects
6	March 3	More data structures • Functions, objects, and data • JSON • Advanced control structures
7	March 10	The real power of JS – a deep dive into functions • Inheritance, prototypes, function literals, scoping and closures
March 17: Spring Break Week – No Classes		
Unit 3: Javascript and the behavior of Web pages		
8	March 24	Making Web pages behave: manipulating the DOM
9	March 31	Working with Browser Events • Script loading, responding to keyboard input or mouse activity, scrolling
10	April 7	Forms and AJAX
11	April 14	Using Javascript Libraries for Advanced Behavior • JQuery and others • Animations, AJAX, form and data handling
Unit 4: Javascript Libraries and Advanced Applications		
12	April 21	Understanding How Libraries Work • Library Architecture and design patterns • Writing a JQuery plugin
13	April 28	Other kinds of libraries • Media players, layout managers • Writing your own library
14	May 5	Javascript and multimedia
15	May 12	Final Projects due (night before) Possible in-class meeting for project presentations