CSCI-E31: Introduction to Web Application Development Using Node.JS

Course Number/Term: CSCI E-31, Spring 2020
Meeting Time: Online (no set meeting times)

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

Course website:
[revision 2019.11.14, subject to change]

Course Description

This course provides an introduction to web application development by way of JavaScript and the Node.js environment. Students learn the basics of server-side web development using the MEAN stack (MongoDB, Express, Angular, Node.js). The course introduces students to models of software development that apply to server-side web development, including the application server (Node.JS), Model View Controller (MVC) frameworks using Express.js, front-end frameworks (Angular), and databases (MongoDB). The course includes setting up a Node.JS environment, building web APIs and full-stack JavaScript applications using the MEAN stack, and following good application development practices. Experience with server-side application development is not required, though knowledge of client-side web development (HTML/CSS/JavaScript) is important.

Overview

The course serves as an introduction to server-side Web application development, as well as an introduction to the technologies involved in server-side JavaScript development.

Through a series of examples and projects, students will learn about Web application architecture, with specific attention to JavaScript frameworks and tools. The course will begin with the Node.JS environment and advanced features of JavaScript that make it suitable to certain types of web applications. Building step-by-step, students will use JavaScript to progressively build upon basic Node.JS to first create a simple Web server, then adding increased application functionality with Express, persistence of data with Mongo, and client-side MVC architecture with Angular.

Goals

This course is designed to introduce students to server-side Web programming in general, and then specifically, to explore the use of advanced JavaScript techniques and application frameworks. Students will learn:

- Foundational Skills:
  - Installation/configuration of Node.JS development/test environment, locally and in the cloud
  - Basic use of git to manage code and deploy to a cloud hosting service
  - Advanced and/or new features of JavaScript including ES6
  - Understanding of Node.JS module and package system
- Web application architecture:
  - Structure of a Node.JS application
  - HTTP servers and services, and the lifecycle of an HTTP request
  - How to use middleware to process Web requests

    - Model-View-Controller (MVC) architecture (via Express)
    - Designing and using a database (via MongoDB)
    - Client-side application frameworks (via Angular)
- Practicing a sustainable path to learning
    - The Node.JS ecosystem is evolving rapidly and changes in the JavaScript core, new and improved application frameworks, and perceived best practices are a constantly moving target. By a focus on the fundamentals underlying the methods and frameworks we study, students will be better prepared to learn and adapt as the technology evolves.

The course will consist of four units. Each unit will focus on a particular aspect of Web application development using Node.JS:

1. Development operations

   Note: This course is not focused on DevOps in a comprehensive way, but it will include enough to support the students' ability to install and use Node.JS and related tools, maintain a code repository, and deploy code to a cloud service.
2. Node.JS and Server-side Fundamentals
3. Server-side application design
4. Angular and client-side application design

Who Should Take This Course

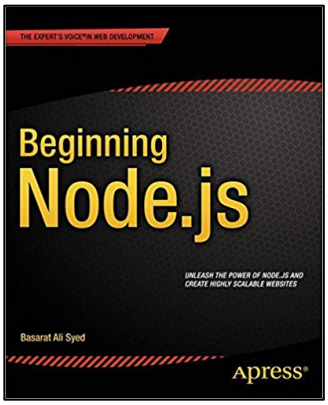
This is an introductory course to server-side application development, as well an introductory course to the Node.JS ecosystem and the "MEAN stack". Students are expected to have basic skills in Web development and the JavaScript language, including understanding of JavaScript concepts such as variable scope, objects and object inheritance, JSON, callbacks, and debugging JavaScript code.

The course will introduce students to the basics of the lifecycle of an HTTP request, and how server-side Web applications work. Students who have prior experience in server-side development will benefit from learning how JavaScript's features and architecture support Web application development, how to develop Model-View-Controller (MVC) applications, design and development of REST APIs, and the specifics of how to develop full-stack JavaScript applications using Node.JS, Express, MongoDB, and Angular.

Course Materials

Optional Books

There are two books we'll use in this course, both available for free online via Harvard's Safari Books subscription. Purchase copies if you enjoy having physical books, but for purposes of this course, the free digital editions are sufficient. **Note: NodeJS technology is rapidly changing – there are details in these books that are already not entirely current, but the specific information we'll use them for is still quite relevant and important.**

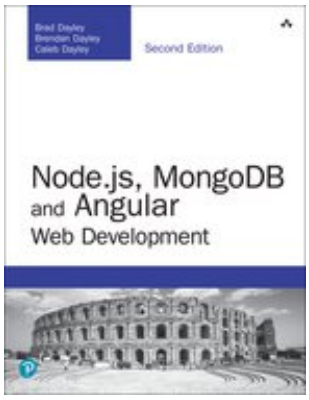


We'll use Beginning Node.JS by Basarat Ali Syed in the first few weeks of the course to get acquainted with Node's general architecture and approach to application development.

Beginning Node.JS

by Basarat Ali Syed
ISBN-13: **978-1484201886**

Available for free 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/9781484201879



We'll refer to **Node.js, MongoDB and Angular Web Development, 2nd Edition** by Brad, Brendan, and Caleb Dayley as a general reference at times throughout the course.

Node.js, MongoDB and Angular Web Development, 2nd Edition

by Brad, Brendan, and Caleb Dayley
ISBN-13: 978-0-13-465564-2

Available for free from the Harvard University Library via Safari Books Online (login with HUID and PIN): https://proquest-safaribooksonline-com.ezp-prod1.hul.harvard.edu/book/web-development/9780134655642

Digital Ocean Hosting

Students will need an account on Digital Ocean (DO) and will create and manage a Digital Ocean server for publishing their working Web applications. The cost of this will be about $5 per month. There are credits for students sometimes available through bundles (check out the Github Student Developer Pack) that may make the DO hosting free for the duration of this course.

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:

- Readings from the assigned textbook and various Web resources
- 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.
- An optional one-hour section meeting led by one of the course Teaching Fellows.
- A set of tasks that employ techniques from the lessons of the week, sometimes in support of the final project, which students work on throughout the semester. Deliverables will usually be expected from students every two weeks.
- 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.

Final Project: The course culminates in a final project of the student's choice. Much of the classwork and semi-weekly deliverables in the course will be progressively building towards the final project.

Graduate Credit: Students taking the course for graduate credit will have one additional assignment over the duration of the course. In this assignment, graduate students will demonstrate a deeper level of understanding by creating teaching materials or practice sets that exercise concepts covered in the prior weeks of the course.

Expected Time Commitment

A standard classroom version of this course would include 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.

While it's impossible to predict the specific time commitment for an individual, some reasonable targets for "typical" 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. Some students choose to attend more than one section per week.
- Two to six hours on the weekly assignments, which are designed to exercise your understanding of individual concepts introduced in the lesson and build towards your final project.

Accessibility

The Extension School is committed to providing an accessible academic community. The Accessibility Office offers a variety of accommodations and services to students with documented disabilities. Please visit http://www.extension.harvard.edu/accessibility-student-services for more information.

Assignments & Evaluation

The course assignments will consist of deliverables roughly every two weeks. Each assignment will consist of a reflection survey, a question or two requiring analysis and/or demonstrating understanding of the week's topics, and most weeks, a code deliverable demonstrating understanding and building towards the final project.

The topics associated with each assignment are noted below. These are subject to change. Details will be provided within the course Canvas site.

1. Assignment 1: NodeJS environment is set up, use of github, successful deployment to Digital Ocean server
2. Assignment 2: Node modules, npm and packages, package.json
3. Assignment 3: ExpressJS: routing and templates
4. Assignment 4: MongoDB and data models
5. Assignment 5: REST APIs in Express, CRUD operations (Create, Read, Update, Delete)
6. Assignment 6: Adding Angular to your application
7. Assignment 7: Final Project

Late Policy

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.

Academic Integrity

Harvard Extension School expects students to understand and maintain high standards of academic integrity. Breaches of academic integrity are subject to review by the Administrative Board and may be grounds for disciplinary action, up to and including requirement to withdraw from the Extension School and suspension of registration privileges.

A Note about Academic Honesty that appears in other Harvard CS courses,[1] :

Do not submit as your own work a program based on the work of another! Violations of this rule is plagiarism**; it is dishonorable behavior, and the penalty for it is requirement to withdraw from Harvard College.**

Two obvious "exceptions" to this rule may be noted in passing. Courses sometimes supply the main idea or even some of the text of a program that is to be completed as an exercise; naturally, students are expected to use this assistance. And there is merit in "copying from oneself" in a course that develops cumulative programming skills. Here programs differ from papers; no author would want to write two different pieces with several paragraphs in common, but with computer programs, this is not unusual. A skill taught in programming courses is how NOT to reinvent the wheel; when a small phrase or short sentence has proven useful and reliable in one program, a programmer should feel free to reuse it if the same thing needs to be said in another program. Such clauses play the role of aphorisms; they make a point but they are not the main point of the piece being written.

Of course, neither of these examples obscures the basic point that a program submitted as original work should not have been derived from the work of another unless the course has specifically permitted this.

How much help on a programming exercise may you obtain before you are stealing, rather than being assisted? Teaching fellows and user assistants know the limitations of what is fair and legitimate; their goal is to help you understand how to solve your own problem, not to solve it for you. If you seek help from other students you are treading on much thinner ice. When a student answers a simple factual question which could have been answered out of a manual, no violation of principle is involved; it is not dishonest to ask another student the value of PI or the statement of the Pythagorean Theorem. But the more your request is for part or all of the solutions to the programming exercise itself, rather than for general factual information, the less acceptable it is. In the extreme case one student asks for and receives the actual text of a program which both were to have created independently; in this case both are guilty of academic dishonesty.

In the Harvard College Handbook for Students is a section related to collaboration:

It is expected that all homework assignments, projects, lab reports, papers, theses, and examinations and any other work submitted for academic credit will be the student's own. Students should always take great care to distinguish their own ideas and knowledge from information derived from sources.

PLAGIARISM

Plagiarism is the theft of someone else's ideas and work. It is the incorporation of facts, ideas, or specific language that are not common knowledge, are taken from another source, and are not properly cited.

Whether a student copies verbatim or simply rephrases the ideas of another without properly acknowledging the source, the theft is the same. A computer program written as part of the student's academic work is, like a paper, expected to be the student's original work and subject to the same standards of representation. In the preparation of work submitted to meet course, program, or school requirements, whether a draft or a final version of a paper, project, take-home exam, computer program, placement exam, application essay, oral presentation, or other work, students must take great care

[1] The principal author of this section is Prof. Harry R. Lewis, Gordon McKay Professor of Computer Science.

to distinguish their own ideas and language from information derived from sources. Sources include published and unpublished primary and secondary materials, the Internet, and information and opinions of other people.

Extension School students are responsible for following the standards of proper citation to avoid plagiarism. A useful resource is *The Harvard Guide to Using Sources* prepared by the Harvard College Writing Program and the Extension School's tips to avoid plagiarism.

INAPPROPRIATE COLLABORATION AND OTHER ASSISTANCE

Collaboration on assignments is prohibited unless explicitly permitted by the instructor. When collaboration is permitted, students must acknowledge all collaboration and its extent in all submitted work. Collaboration includes the use of professional or expert editing or writing services, as well as statistical, coding, or other outside assistance. Because it is assumed that work submitted in a course is the student's own unless otherwise permitted, students should be very clear about how they are working with others and what types of assistance, if any, they are receiving. In cases where assistance is approved, the student is expected to specify, upon submission of the assignment, the type and extent of assistance that was received and from whom. The goal of this oversight is to preserve the status of the work as the student's own intellectual product. Students should remember that the Writing Center is available to assist them with assessing and editing their own work.

CHEATING

Students may not copy other students' work, computer programs or parts of programs, or exams. To avoid any suggestions of improper behavior during an exam, students should not communicate with other students during the exam. Neither should they refer to any books, papers, or use electronic devices during the exam without the permission of the instructor or proctor. All electronic devices must be turned off during an exam.

DUPLICATE ASSIGNMENTS

Students are expected to submit work that is done solely for each course in which they enroll. Prior written permission of all instructors is required if students wish to submit the same or similar work in more than one course.

Students who repeat a course must have the instructor's approval to reuse or resubmit work that they previously submitted for the same course.

Weekly Schedule

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

WEEK	LESSON DATE	TOPICS
UNIT 1: Development Operations		
1		Understanding Node • Installing Node.JS • Getting used to the command-line • Try out Node.JS, REPL
2		Managing and Deploying Code • Git and github • Ssh keys • Setting up and deploying to Digital Ocean
UNIT 2: Node.JS and Server-side Fundamentals		

WEEK	LESSON DATE	TOPICS
3	Node.JS modules and packages	
4	JavaScript Objects and Inheritance Streams and Events Connect and Middleware	
UNIT 3: Server-side application design		
5	HTTP • Create a Web server • Connect • Middleware	
6	Express.JS • MVC Architecture • Express Configuration • Routing	
7	Express.JS • Templates and template languages • Express Middleware • Handling HTTP form data, request types	
8	Persisting Data • Database Basics • Install Mongo, Mongoose • create and use models, • basic create/query • mongo CLI	
9	REST APIs • REST APIs • CRUD (Create, Read, Update, Delete) in Express to manage your application's data	
Unit 4: Client-side Frameworks and Angular		
10 - 13	Introduction to Angular • Angular MVC Architecture • Adding Angular to your Project • Debugging Angular Applications Angular Project Design • Introduction to TypeScript • Angular Data model • SPA (Single age Applications) Key Angular Features • Data Bindings • Directives • Modules vs. Components • Routing • Error Handling	
14	Other Node Applications • Sockets for real-time communication • TBD	