Yale x Flatiron School Web Development Bootcamp

CPSC S1xx: Introduction to Full-Stack Web Development

Course Overview

This 10-week course introduces students to full-stack web development. Students develop a foundation in programming fundamentals through web development using Ruby and Javascript. Students will work with APIs (Application Programming Interfaces), understand the basic concepts of object-oriented programming, become proficient in database modeling and ORM (Object Relational Mapping), apply the MVC (Model View Controller) framework, and execute full application development lifecycle.

The course includes three modules—A, B, and C—and a final project. Within each module, students will complete a series of problem sets to help them develop the knowledge to complete that module. These problems sets increase in complexity over the course.

Module	Module Title	Student Hours*	Grading Weight
Pre-work & Module A	Programming Fundamentals: Ruby Basics and Object-Oriented Programming	100 hours in Pre-work + 120 hours in Module A	30% Based on Module Assessment
Module B	Web Frameworks: Web Development with Rails	190	25% Based on Module Assessment
Module C	Javascript: Creating Interactive and Performant Front-Ends with Javascript	85	25% Based on Module Assessment
Student Project	2 Week Student Project	100	20% based on Project Rubric
	Program Totals:	**595 (495 outside of Pre-work)**	**100%**

*Student Hours assumes time spent inside and outside of class

Grading Assessments

Assessments are graded on a scale from 1 to 5. To calculate a student's final grade, they should multiply each grade by the percentage weight for that module and sum those numbers. For example if a student received an 4 on Module A, a 3 on Module B, a 5 on Module C, and a 3 on their final project their final grade would be (4*0.3)+(3*0.25)+(5*0.25) + (3*0.2) = 3.8 and this student would receive an B.

>= 4.0 = A

> 3.0 and < 4.0 = B

>= 2.0 and < 3 = C

>= 1.5 and < 2.0 = D

< 1.5 = F

Academic Integrity: Programming, like composition, is an individual creative process. Individuals must reach their own understanding of the problem and discover a path to its solution. During this time, discussions with friends are encouraged. However, when the time comes to write the code that solves the problem, such discussions are no longer appropriate -- the program must be your own work (although you may ask teaching assistants or lab assistants for help in debugging).

Do not, under any circumstances, copy another person's program. Writing code for use by another or using another's code in any form violates the University's academic regulations and will be dealt with harshly. *We will carefully screen programs for evidence of copying, using a combination of automated tools and instructor review.*

Major Equipment: Ideally Apple Computer with Mac OS X 10.11.6 or later, but any laptop with minimum 4 GB (8 GB is preferable).

Pre-work & Module A – Programming Fundamentals: Ruby Basics and Object-Oriented Programming

Students will begin exploring version control using git commands including with cloning, branching, merging, rolling back commits, forking, and submitting pull requests. We will also learn explore the history of software development from Waterfall to Version control and discuss the historical significance of open sourced tools like Linux.

Students will then learn fundamental concepts in programming including repls, methods, loops, variables, variable scope, conditionals, blocks and iterators, case statements, arrays, scope, hashes, regular expressions, iterators, enumerables, data structures, nesting, etc. Topics will build in complexity and provide the foundation for the rest of the course.

Students will learn to embrace error messages as clues and gain a fundamental appreciation for failure as the only way to learn and progress. Students gain experience in debugging with various gems and tools designed to track down issues in code.

After gaining a deep understanding of ruby basics, students will gain experience with Object Oriented Programming and understand how it allows programmers to bundle code and create reusable objects and methods, allowing for increasing complexity in software. Object Oriented programming is the design pattern that most languages use to create software. While this pattern is learned in Ruby, we re-visit object oriented design in Javascript later in the class to ensure that students are learning the pattern of object oriented programming.

Deliverable: A solid foundation in Ruby, continued experience with git, experience reading and understanding unit tests, enhanced problem-solving skills and an ability to encapsulate code in classes and objects.

Prerequisite: None
Hours: 220

Module B – Web Frameworks: Web Development with Rails

Before discussing web development, students must first understand data persistence. Saving data in a structured way that maps to object oriented design is handled through an object relational mapper. ORM (Object Relational Mapping) allows programmers to query and manipulate data from a database using an object-paradigm. Students learn to write manipulate data using the Ruby language. We will gain an appreciation for the structure of a database, how to map out tables, and the difference between the various table relationships. Students will learn how to wireframe database structures, as well as how to link their applications to a database. This unit covers sql, domain modeling, relational database theory, schema architecture, and the Object Relational Model, include the ActiveRecord pattern.

After obtaining an understanding of HTTP and Rack, students will learn Sinatra. Sinatra is a Domain Specific Language (DSL) written in Ruby for building web applications on top of Rack. This framework provides students with exposure to design patterns in web applications. The topics covered in this unit include architectural patterns such as REST (Representational State Transfer), MVC (Model-View-Controller), HTML Forms, ERB (Embedded Ruby) and template rendering, and application environments.

Having a foundation in the Ruby language as well as the architecture of the world wide web, students will use Rails to build complex functional web applications from the ground up. They will learn the file structure of Rails, how to set up their own databases, how to draw routes, create Rails forms, gain an understanding of the asset pipeline, and bring it together by integrating their front-end design skills.

Students will also have the ability to take on more advanced concepts such as authorization, validation and callbacks. Once students grasp the basic functionality of Rails, they will spend time building out their own Rails applications, moving through the entire process from idea to execution

Deliverable: A multi-model Ruby on Rails powered web application that integrates nested forms and URL routes to a multi-relational data set.

Prerequisite: Module A
Hours: 190

Module C – Javascript: Creating Interactive and Performant Front-Ends with Javascript

Javascript (EMCAScript) powers the user experience of the web. Students will learn the basics of the javascript syntax, its functional architecture, along with a few approaches to the object model. They will then learn the Document Object Model (DOM) Javascript API provided by the browser to dynamically interact with HTML. This unit will focus on jQuery, the most popular Javascript library, to aid students in learning how to collect user input, manipulate the DOM with animations and injection, and send Asynchronous AJAX requests for a rich user experience.

Deliverable: A Rails application that includes examples of DOM Manipulation, AJAX HTML Loading, and AJAX Form Submissions.

Prerequisite: Module B
Hours: 85

Student Project

During this portion of the course students are tasked with working alone to create a project. This course provides an in-depth opportunity for student to demonstrate their learning accomplishments and put them into practice.
Students begin with an initial pitch session. Students are tasked with pitching three different ideas that they are passionate about. Instructors take the pitches and then choose the final project that the students work on. Instructors choose projects based on difficulty and feasibility given the time constraints of the course.

Students are then paired with an instructor as their product manager. The student works through various milestones to complete the project.

Deliverable: Final project

Prerequisite: Modules A, B, C
Hours: 100