launch code

Learn to Code / Get a Job / Hire Talent

CodeCamp

Course Overview

The goal of this class is to teach you the skills you need to launch a career in technology. The course has been built to provide you with a solid foundation in the most commonly-used areas of computer programming and to provide you with a strong, well-rounded skill set. It also emphasizes skills that are essential for success as a professional developer, such as problem-solving and the use of developer tools.

Students should expect to spend at least 40 hours a week in class. The class meets 5 days a week for 8 hours. The schedule is designed to provide students with time to work on and complete assigned independent prep work, studio projects, and graded assignments. Assignments will generally be due once every other week.

Evaluation

The course is Pass/Fail. There are two units, and you will receive a status of Pass or Fail for each unit:

> **Pass:** Students who successfully complete all of the graded assignments for a unit will receive a Pass for that unit.

Fail: Students who do not complete the required assignments by the end of a unit will receive a Fail for that unit and will not be allowed to continue to the next unit.

Individual assignments are also scored on a Pass/Fail scale; a score of 1 denotes Pass and a score of 0 denotes Fail.

Class Structure

CodeCamp is structured around a few key activities: independent prep work, in-class lecture / studio work, and graded assignments. These activities are structured to make the most of your studio and assignment work time. It is important to actively engage with each activity. Skipping the prep work or falling behind on assignments can quickly lead to struggle and even failure.

Independent Prep Work

Prep Work is due before each class studio and covers the topics that you will learn about for that lesson. Students must complete prep work on their own before attending the studio. Prep work includes reading and/or video lessons, review questions, and small coding exercises.

In-class Work: Lecture and Studio

Each class combines large-group and small-group activities. The Instructor will lead most large-group activities during lecture time, which may be Q&A sessions, concept reviews, or coding demonstrations. Each student is assigned to a Teaching Assistant (TA), who leads their small-group activities. These activities are typically hands-on coding problems known as studios.

Prep work and in-class work activities are not graded and are not required for course completion. However, you will struggle and most likely fail if you do not engage with all course activities.

Graded Assignments

Graded assignments are larger projects where you demonstrate what you have learned and challenge yourself. Assignments often cover multiple lessons. As outlined above, you must pass all graded assignments in order to complete the course. The course is designed to provide in-class time to work on these assignments.

Course Objectives

This course is broken down into 2 Units:

Unit 1: Intro to Professional Web Development in JavaScript

Unit 2: Skill Track - Back-End Web Development in Java or C#

Upon successful completion of the course, students will be able to:

- Use the foundational structures of code to solve problems in the JavaScript programming language, as well as at least one skill track language (Java or C#).
- Build web applications that utilize a database to manage user data.
- Effectively use common developer tools and implement best practices to write professional-quality code.
- Create programming projects from scratch using in-demand skills and technologies.

Unit 1: Intro to Professional Web Development in JavaScript (7 Weeks)

Unit 1 covers core, universal programming concepts in the JavaScript language with a focus on problem-solving. This will provide students with the conceptual building blocks that every programmer needs.

It also introduces fundamental front-end web programming concepts and provides a brief introduction to Angular, a modern JavaScript framework.

Topics

- Statements and variables
- Data types and typecasting
- Mathematical and comparison operators
- Data structures: arrays and objects
- Code style best practices
- Flow control: conditionals and loops
- Functions and scope
- Unit testing and Test-Driven Development
- Objects and classes
- Terminal usage
- Version control with Git and GitHub

- HTML: tags, forms, and semantics
- CSS: selectors and style rules
- Fundamentals of HTTP
- JavaScript on the Web: DOM, events, forms, fetching remote data
- Intro to Typescript and Angular

Unit 2: Skill Track (7 Weeks)

Unit 2 introduces students to object-oriented programming in either Java or C#. It then covers industry-grade web frameworks in these languages (Spring Boot and .NET MVC, respectively) along with development tools and practices.

Note: The high-level topics covered in the Java and C# tracks are the same, though some lower-level details may differ.

Topics

- Data types and variables; statically-typed languages
- Control flow and collections
- Objects and classes: access modifiers, class construction, and encapsulation
- Unit testing
- Inheritance, interfaces, and polymorphism
- How web apps work
- Model-View-Controller design for web applications
- Views and templates
- Exceptions
- Model classes, model binding, and model validation
- Relational databases (MySQL)
- Object-Relational Mapping
- REST APIs
- Sessions, cookies, hashing, and authentication

Additional Course Details

Getting Help

Modern programming is done collaboratively. Throughout the course, there will be an emphasis on working in groups to tackle problems together. Students will be divided into

small groups, each led by a Teaching Assistant (TA). TAs will be available to assist individual students during class and to help the group through common mistakes. Students can use the class discussion board, where questions can be answered by peers and TAs.

Academic Honesty

Academic honesty is an important focus of this course. Working with others to improve your skills is both acceptable and encouraged, but there is a difference between asking for help and submitting someone else's work. Below are some examples to differentiate between the two:

- If you encounter a bug in your program, a classmate may look at your code to help you identify the source of your problem, but you may not look at the classmate's code to find their solution to that problem.
- You are encouraged to search through sites such as Stack Overflow when trying to debug an error in your code, but you may not search online for complete solutions to your specific assignment.
- Do not post complete programs or functions on the class discussion board. Post only snippets or isolated sections of code. If you need to post a complete program or function in order to get help, do so in a private message to a TA or instructor.

If you have a question about how to work with another student on graded homework, ask the instructor or course staff.