General Assembly
Course Curriculum

BACK-END WEB DEVELOPMENT

GA.CO/BEWD

BEWD

Back-End Web Development
Table of Contents

3 Overview

4 Students

7 Curriculum
Projects & Units

11 Frequently
Asked Questions

13 Contact
Information

OVERVIEW

THE FRAMEWORK

A web developer that creates client-side websites can only go so far without back-end logic. Creating a web application has never been more simple with Ruby on Rails. Yukihiro Matsumoto designed the Ruby programming language with the programmer in mind and wanted it to be easy, fun and productive. Using Rails, beginners can quickly create web applications that communicate with both the front-end of a site, and back-end data stores.

In this 10-week program, students will practice building rails applications, in addition to developing their own ideas into functional web applications (minimum viable products). This course will give aspiring entrepreneurs and career changers the confidence to speak Rails jargon and to continue to learn about programming fundamentals. The focus of this course is programming in Ruby and creating Rails web applications.

However, GA students will gain the skills needed to be successful during the course, and beyond. In addition to teaching Rails, this course also includes lessons on programming fundamentals, command line basics, Git, GitHub, and database schemas.

» Design and implement functional web applications using Rails
» Practice solving problems like a developer by writing object oriented programming code with Ruby
» Integrate third party APIs/GEMs and write the logic required to customize solutions on the web
» Use GitHub and Heroku to share your application on the web
» Communicate web application ideas to teammates and other stakeholders

STUDENTS

ENTREPRENEURS	This course provides entrepreneurs without a programming background, the vocabulary to communicate web application ideas to other potential stakeholders. Upon completion of this course entrepreneurs will have built a small version of their app idea.
ASPIRING RAILS DEVELOPERS	This course is an introduction to Ruby on Rails. Although it is not a fast track to web dev positions like WDI, those who wish to change careers will have opportunity to practice and gain proficiency to actively participate in the Rails community and continue learning. Upon completion, aspiring rails developers will have a functional Rails application that they can add to their growing portfolio. In addition they will have the skills to continue learning and perfecting their craft.
INDIVIDUALS SEEKING TO PROGRESS IN THEIR CURRENT CAREER	Project Managers and Front-End Web Developers will find this course useful as it will help them better communicate with the Rails developers they currently work with. Upon completion, these students will have experienced the challenges and requirements.
PROGRAMMING ENTHUSIAST (HOBBYIST)	For those looking to get their feet wet and explore the world of programming, this course will offer you an introduction to object oriented programming and web development. In addition to offering opportunities to practice these new skills.

PROJECTS

FINAL PROJECT

For the final project students use knowledge of back-end web development to produce a rails web application that will be hosted on Heroku and usable by anyone that visits the site. The objective of this project is to:

» Demonstrate understanding of all topics covered during this course
» Apply knowledge gained during this course by building a web application from the ground up

The final encapsulates everything learned in this course. GA wants to make sure you are successful in your venture, so to stay on track we create milestones to help your design and build your application. In addition, instructors will help define scope and guide process.

Projects must include:

» About me page that describes your app
» Security Features: login credentials must be validated before users can gain access to your application
» Heroku hosting
» Ruby Gems. Use a Ruby Gem (Rails defaults don't count). Some ideas to look into: devise or paper clip, carrier wave.
» Handle invalid data. Models in your application should validate data and handle incorrect inputs (i.e. validate sign up information, verify valid email addresses and secure passwords).
» Make use of associations. At least two of your models should be associated.

PROJECTS

BEST PRACTICES

Even though it is not part of the requirements, students should keep mind and demonstrate their understanding of best practices listed below:

» Clean and Readable Code. Instructors and TAs should be able to read and follow code easily. Maintain clean and readable code including: consistent indentation, code commenting (e.g. when closing tags, demarcating sections of code, describing possibly ambiguous code choices) and use proper and consistent naming conventions.
» RESTful. Implement some non-default (defaults are: add, create, update, destroy) RESTful routes to demonstrate your understanding of RESTful routing in Rails.

UNITS

UNIT 1: RUBY	» Working Like a Developer	*Lesson 1*
	» Basics of Computational Thinking	*Lesson 2*
	» Collections and Iteration	*Lesson 3*
	» Working With Collection and APIs	*Lesson 4*
	» Classes and Objects	*Lesson 5*
	» Using Modules and Exception Handling	*Lesson 6*
UNIT 2: RAILS	» My First Rails App, ER Diagrams, MVC	*Lesson 7*
	» Controllers and Views	*Lesson 8*
	» Models, Migrations, and Active Record	*Lesson 9*
	» Forms	*Lesson 10*
	» Review and Project Lab Session	*Lesson 11*
	» Authentication Part I and Managing Users	*Lesson 12*
	» Identifying and Coding Associations	*Lesson 13*
	» Starting Your Project (Potential Guest Speaker)	*Lesson 14*
	» Authentication Part II: OmniAuth (OAuth/Ruby)	*Lesson 15*
	» Finishing Touches: Front-End Templates	*Lesson 16*
	» Precious Gems	*Lesson 17*
	» Working with Third Party APIs	*Lesson 18*
	» Project Lab Session	*Lesson 19*
	» Final Project Presentation	*Lesson 20*

1 RUBY

1 WORKING LIKE A DEVELOPER
» Recognize the difference between a website and a web application
» Utilize the shell commands to create and change directories
» Define Git and articulate the benefits of using it
» Configure Git on computer and push/pull from a repository

2 BASICS OF COMPUTATIONAL THINKING: VARIABLES AND CONDITIONALS
» Define computational thinking and translate instructions into basic pseudo code
» Define variables and data types
» Utilize conditional logic and variables to create a simple Ruby program

3 COLLECTIONS AND ITERATION
» Define arrays, hashes and differentiate between the two
» Describe iteration in computational thinking and relate to loops
» Apply loops to manipulate collections (arrays and hashes)

4 WORKING WITH COLLECTION AND APIS
» Use loops, arrays, and hashes to work with JSON returned by API
» Practice and apply all skills learned thus far

5 CLASSES AND OBJECTS
» Create and implement custom classes
» Control the scope of class methods and variables

6 USING MODULES AND EXCEPTION HANDLING
» Create classes that use inheritance to share behavior
» Define how modules and mixins extend class behavior

2 RAILS

7 MY FIRST RAILS APPLICATION, ER DIAGRAMS, AND MVC FRAMEWORK
» Define how DNS servers, web servers and web browsers work together to deliver a Rails app
» Describe HTTP URI structure and Rails request handling cycle
» Define framework and explain how Ruby works with Rails
» Create a basic Rails application using scaffolding
» Review Rails application file structure

2 RAILS (CONTINUED)

8 CONTROLLERS AND VIEWS
» Explain the MVC structure and differentiate between views and controllers
» Describe the request response cycle and the flow of control in a Rails app
» Identify where to put CSS and JavaScript files

9 MODELS, MIGRATIONS, AND ACTIVE RECORD
» Describe a model's function in the MVC framework
» Define migrations and apply rake commands
» Implement migrations to update tables
» Operate Activerecord to CRUD records in the database

10 FORMS
» Utilize form builders to add search capabilities
» Define validations and implement them in a model
» Describe partials and identify the best instance to use them

11 REVIEW AND PROJECT LAB SESSION
» Practice and challenge understanding of Rails thus far

12 AUTHENTICATION PART I AND MANAGING USERS
» Describe the authentication process in a Rails app
» Implement Devise Gem to add authentication to an app
» Describe the use of the before_filter and implement it to restrict user access to pages

13 IDENTIFYING AND CODING ASSOCATIONS
» Analyze rails apps and identify model associations
» Utilize documentation to code associations

14 STARTING YOUR PROJECT (POTENTIAL GUEST SPEAKER)
» Describe and write user stories
» Draft wire frames from user stories
» Identify how to use wire frames and user stories to design a rails app

15 AUTHENTICATION PART II: OMNIAUTH (OAUTH / RUBY)
» Investigate third party authentication gems
» Practice reading API documentation

2 RAILS (CONTINUED)

16 FINISHING TOUCHES: FRONT-END TEMPLATES — TWITTER BOOTSTRAP, HEROKU, AND CARRIER WAVE

» Utilize front-end templates to style an app
» Deploy a project to heroku
» Investigate file upload gems

17 PRECIOUS GEMS

» Instructor's session to introduce useful Gems to students

18 WORKING WITH THIRD PARTY APIS

» Practice reading third party API documentation

19 PROJECT LAB SESSION

» Students work on final project before presentation

20 FINAL PROJECT PRESENTATION

» Present final project web application to class

FAQS

WHY IS THIS COURSE RELEVANT TODAY?	A web developer that creates client-side websites can only go so far without back-end logic. This course introduces students to web application programming which allows one to create robust, back-end Rails applications that communicate with both the front-end of a site, and back-end data stores. Students will learn how to integrate with third party APIs and write the logic required to drive customized solutions on the web.
WHAT PRACTICAL SKILL SETS CAN I EXPECT TO HAVE UPON COMPLETION OF THE COURSE?	This course is an introduction to web development with Ruby on Rails. By taking this course you will gain the skills necessary to build and deploy a Rails web application with user logins and sign-ups. This course will first teach you core programming fundamentals, in Ruby, that are essential for building web apps. In addition, you will learn how to approach and solve a problem like a developer. Students also leave this course with a better understanding of how the Internet works.
WHO WILL I BE SITTING NEXT TO IN THIS COURSE?	This is a beginner level course, and therefore a wide range of people sign up. In the past we've had: Designers, Journalists, Front End Web Developers, Bloggers, Entrepreneurs, Project Managers, and Students.
WILL THERE BE ANY PRE-WORK?	We'd prefer that students have a basic knowledge of HTML and CSS. If you are interested in taking the course but don't know anything about HTML and CSS, we can set you up with tutorials prior to class. Students will have to go through a command line tutorial before the first day of class.
SHOULD I COME EQUIPPED WITH ANYTHING?	Preferably a Mac, but a PC will do. Motivation to learn how to code.
HOW MUCH OF THE COURSE IS FOCUSED ON BACK-END DATABASES?	This course will provide an overview of database tables and their structure. In addition, it provides a high level overview of how Rails retrieves and stores data over the Internet. SQL is not needed to create a Rails app; therefore the course will not cover it. However, it will introduce how to manage data in Rails.

FAQS

HOW MUCH PREREQUISITE KNOWLEDGE IS REQUIRED TO TAKE THIS COURSE?

We'd prefer that students have a basic knowledge of HTML and CSS. If you are interested in taking the course but don't know anything about HTML and CSS, we can set you up with tutorials prior to class.

WHAT ARE THE MAIN TOPICS OR THEMES THAT ARE COVERED THROUGHOUT THE COURSE?

Ruby programming fundamentals; Rails: mvc, scaffolding, routing, creating models, views, controllers, embedding ruby logic in your app, front-end templates, and structuring data.

WHAT IS BACK-END WEB DEVELOPMENT?

Back-end web development is the logic behind an application. The backend of a web app, stores and retrieves data over the Internet to a database. For example, when you take a picture and upload it to Facebook that entire process is enabled by back-end web development. Your photo is saved on Facebook's database and retrieved when your friends visit your wall. When you post a tweet, that is also back-end functionality. The ability to save data (text, images, video) in a web application is enabled by back-end web development.

WHY DOES THE COURSE FOCUS ON RUBY AND RUBY ON RAILS?

Learning to code is one of the most in demand job skills of the 21st century. We chose Ruby and the Ruby on Rails framework for a number of reasons:

» The syntax of the Ruby programming language is extremely beginner friendly. It reads almost like English.
» The Ruby on Rails community is really beginner friendly and there are plentiful learning resources.
» There is a huge and growing demand for Ruby on Rails developers at startups. A big reason is because the Ruby on Rails framework cuts out a lot of the mundane tasks of web development. Rails allows you to be productive quickly, which is key for startups who are sensitive about time to market.

We want to give you the foundation so that you are able to continue programming beyond this course. Therefore we introduce Ruby first and practice programming basics before we begin to create web applications. Learning Rails involves understanding databases and how your computer interacts with the Internet in order to send and receive information.

CONTACT

GET IN TOUCH

NEW YORK
nyc_admissions@ga.co
+1 (805) GANYC-01
Course application

BOSTON
boston_admissions@ga.co
(412) 852-2664
Course application

SAN FRANCISCO
sf_admissions@ga.co
(415) 294-0866
Course application

LONDON
london_admissions@ga.co
+44 (0) 7703 002 046
Course application

WASHINGTON, DC
dc_admissions@ga.co
(734) 363-8339
Course application

SYDNEY
sydney_admissions@ga.co
+61 2 0420 438 649
Course application

LOS ANGELES
la_admissions@ga.co
(424) 543-7083
Course application

HONG KONG
hk_admissions@ga.co
+852 9473 3608
Course application