UDACITY

NANODEGREE PROGRAM SYLLABUS

Full Stack JavaScript Developer

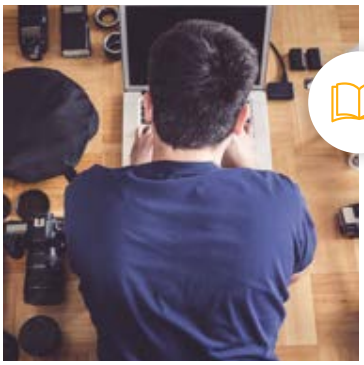
Overview

You'll master the skills necessary to become a successful full stack developer. Learn how to build UI and UX, create APIs and server side business logic, and develop the persistence layer to store, process and retrieve data.

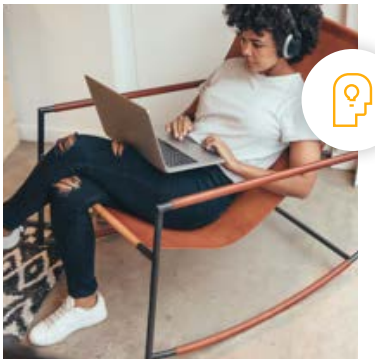
Educational Objectives:
Students who graduate from the program will be able to:

- Build client-side experiences and applications using Angular, collecting data from users and from backends, providing rich user interactions and organizing code and data.
- Build server-side executed code with TypeScript and integrate with 3rd party code such as Angular's Server Side Rendering.
- Leverage Express.js to architect and build APIs that power dynamic functionality and to generate and supply data to web and mobile clients.
- Persist data to a database, query and retrieve data, and pass this data all the way through to various client devices.

Estimated Time:
4 Months



Prerequisites:
HTML, CSS, Basic JavaScript, JSON



Flexible Learning:
Self-paced, so you can learn on the schedule that works best for you.

Need Help?
udacity.com/advisor
Discuss this program with an enrollment advisor.

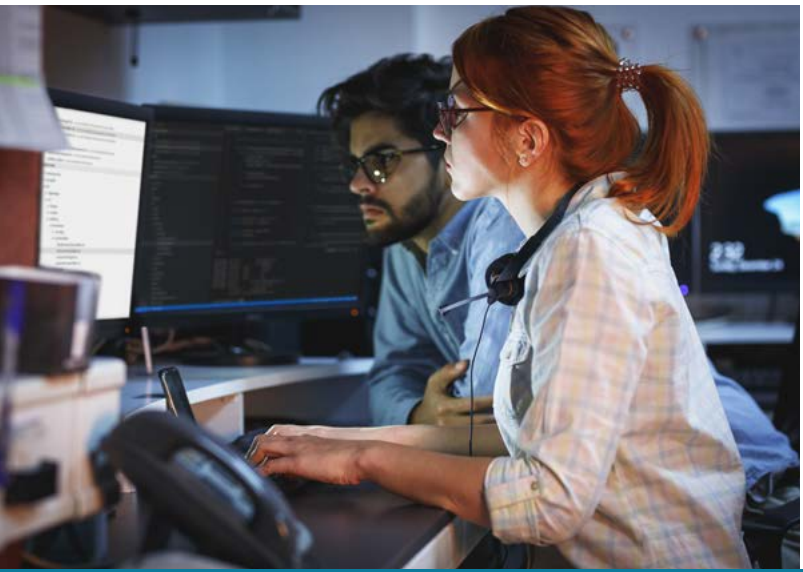
*The length of this program is an estimation of total hours the average student may take to complete all required coursework, including lecture and project time. If you spend about 10 hours per week working through the program, you should finish within the time provided. Actual hours may vary.

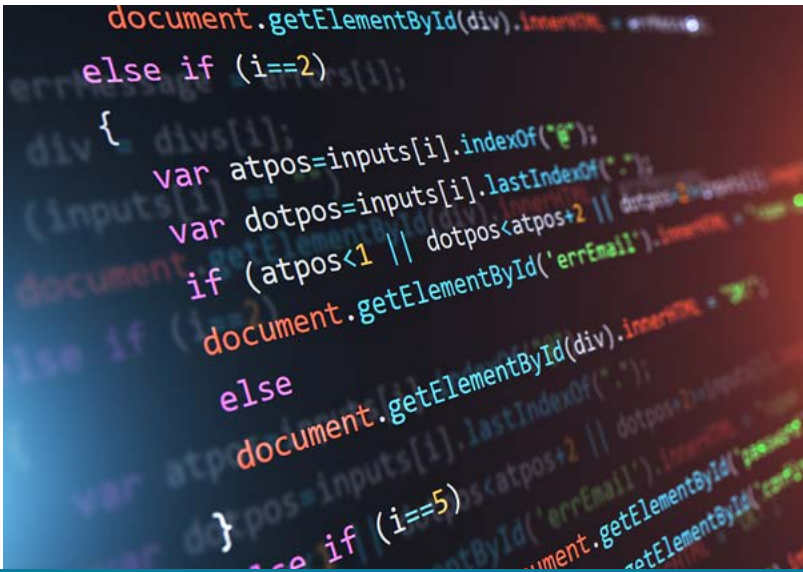
Course 1: Backend Development with Node.js

There are quite a few technologies involved to build the backend of an application that's enterprise ready. This course introduces the fundamental tools needed to build a basic API in a way that is both scalable, and maintainable. The course will go through working with Node.js and the core modules available, writing TypeScript for developer error reduction, testing with Jasmine to introduce unit testing in a Test Driven Development environment and working with Express as a framework for building APIs.

Course Project :
Image Processing API

Students will be designing an API for image processing that allows the user to visit a url and using url parameters, resize the image based on the parameters provided. Upon viewing an image that's already been resized, a cached image will be served. This is the core functionality seen in placeholder image websites and can be implemented with a frontend to better serve appropriately sized images. The API presents the first opportunity to pull together the technologies of the course and tie them together in a commonly used application.





		LEARNING OUTCOMES
LESSON ONE	**Getting Started with Node.js**	• Understand why to use Node.js. • Install and understand how Node.js is updated. • Understand the event loop and control how asynchronous code is processed. • Use Node.js REPL to write js expressions, then just Node.js to run a js file. • Extend JavaScript by using global variables and functions not available in the browser. • Use NPM init to create a package.json/lock to install dependencies and configure scripts.
LESSON TWO	**Developing with TypeScript**	• Take advantage of strict typing to reduce error by installing TypeScript to work with Node.js. • Create valid, formatted, readable TypeScript by configuring ESLint/Prettier to work with TypeScript. • Type variables, functions and objects with TypeScript. • Manage async/await, promises and error handling with TypeScript.
LESSON THREE	**Unit Testing with Jasmine**	• Install Jasmine and configure it to work with JavaScript after TypeScript has been compiled. • Organize, write and run unit tests. • Create asynchronous tests and use Supertest to perform endpoint tests.
LESSON FOUR	**Building a Server**	• Build a server by applying the top features of Express's root app object. • Improve an application by creating and applying middleware. • Take advantage of the file system by learning to write and read files from disk.

Course 2: Creating an API with PostgreSQL and Express

This course covers the primary skills required for API development. Students will build a RESTful JSON API with Node and Postgres. Along the way, you will cover essential topics like databases and querying, API architecture, database migrations, REST, CRUD, creating a testing environment, password hashing and route authorization via JWTs. By completing the exercises and course content, students will gain the knowledge to create a secure, well organized API from scratch and learn skills JavaScript developers use every day.

Course Project :
Build a Storefront Backend

Students will imagine themselves as a full stack developer at a small company, asked to craft an API to meet a set of requirements created by business stakeholders and to be consumed by a front end developer coworker. You will use skills and understanding from the course to discern the best architecture, endpoint structure and database schema to complete the task. Modeled after a workplace environment and true to life task, in this project you will demonstrate that you can reason through the design of an API and are capable of writing the code and logic necessary to meet requirements.

		LEARNING OUTCOMES
LESSON ONE	**Databases and SQL**	• Understand what databases are. • Gain visibility into the most popular types of databases, their respective strengths and use cases. • Become familiar with Relational Databases. • Connect to a database with interactive terminal psql. • Write basic database queries with SQL. • Create, structure and fill a Postgres database. • Know what CRUD is and how it pertains to database actions. • Understand when and how to use SQL filters and foreign keys.

LESSON TWO	**Creating an API with Postgres Connection**	• Set up a Node application to connect to a postgres database. • Install a Node package dotenv for environment variables. • Learn best practices for using environment variables. • Create Models in Node. • Understand the need for database migrations. • Install a Node package for database migrations. • Set up a testing environment with a separate testing database. • Testing models with Jasmine.
LESSON THREE	**Creating an API with Express**	• Understand RESTful APIs and how to design one. • Understand the full stack request/response loop through the API. • Be introduced to what CORS is and when it is necessary. • Write Express custom middleware. • Write clean, well organized Express routes that mirror API structure.
LESSON FOUR	**Authentication and Authorization in a Node API**	• Be familiar with password hashing, salt and pepper. • Understand why these practices are important for data security. • Install a Node package Bcrypt and learn to set up Bcrypt for password hashing at user creation. • Understand what JWTs are and how they work. • Implement route authorization with JWTs.
LESSON FIVE	**SQL for Advanced API Functionality**	• Understand database relationships and when to implement them. • Create RESTful routes that demonstrate relationships between data via well organized routes. • Write custom model and handler methods for extending API endpoints. • Understand SQL Joins and result ordering. • Write informational API endpoints to support dashboard functionality. • Organize more advanced business logic in a Node API for cleaner code and separation of concerns.

Course 3: Angular Fundamentals

In Angular Fundamentals, students will learn the most important and foundational skills for building Single Page Applications (SPAs). You will discover the architecture of an application, explore how to retrieve and flow data throughout an application, and see how applications scale in a maintainable and performant way. Upon completion of the course, you will be able to build new and expand existing Angular applications with new components and features, architect an Angular application for clarity and maintainability while following best practices and create and use dependencies such as services and third-party libraries to enrich and extend applications.

Course Project :
My Store

In this project, students will build a full single-page ecommerce application with Angular called MyStore. Your application will contain a variety of different Angular components that communicate with each other, such as a product list component that renders a list of items for which a user can shop. You'll pull this data by making requests to a backend API, then populate your page with items that can be added to the shopping cart.

You'll build and nest these components in a logical structure for optimal navigation and routing, such as bringing users to a product detail page. Through using services, among other tools, you'll be able to share data with any component that needs it, such as your shopping cart. Your application will also be able to handle and respond to user input, through Angular's powerful template-driven forms.

		LEARNING OUTCOMES
LESSON ONE	**Angular Overview**	• Set up and install the Angular CLI. • Scaffold a new application using the Angular CLI. • Generate a component. • Organize components into closely-related sets of capabilities (i.e., modules).
LESSON TWO	**Components**	• Build a component by defining a class containing application data and logic. • Build templates to represent an application's user interface. • Use data binding to re-render the application by detecting changes in state. • Extend the style or behavior of HTML elements with directives. • Use the Angular router to handle navigation between views.
LESSON THREE	**Libraries & Services**	• Extend Angular's base functionality by importing libraries and modules. • Use dependency injection to provide custom services and capabilities. • Find, add and manage an application's dependencies.
LESSON FOUR	**Data**	• Fetch data via HTTP with the HTTP Client Module. • Collect user input by building template-driven forms. • Validate user input.

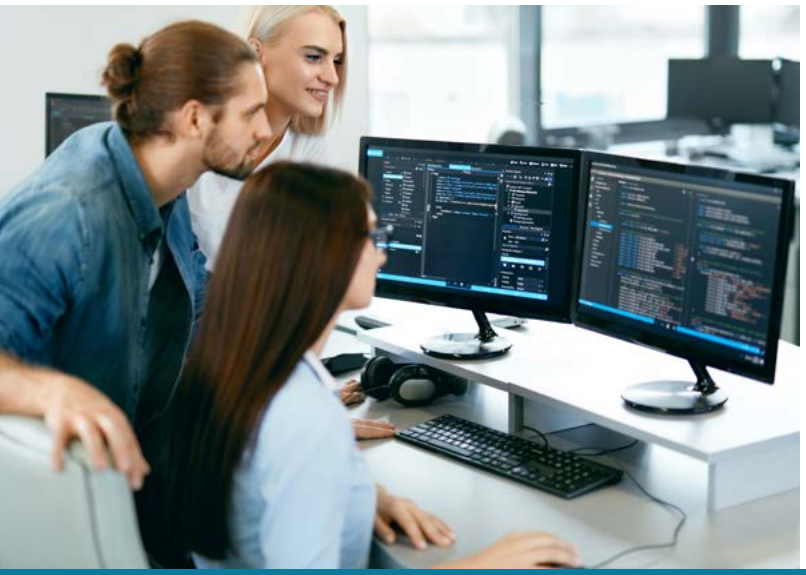
Course 4: Deployment Process

Being able to deploy your own application is a skill that is often overlooked by developers, thus making it a rare and valuable skill to have! This course will teach the necessary knowledge to create your own production environment and automate the deployment of code to it. By building an automated pipeline and scripts students will gain insights into the world of automated deployments that has been revolutionizing how fast companies are able to deliver features to their customers.

Course Project :
Hosting a Full Stack Application

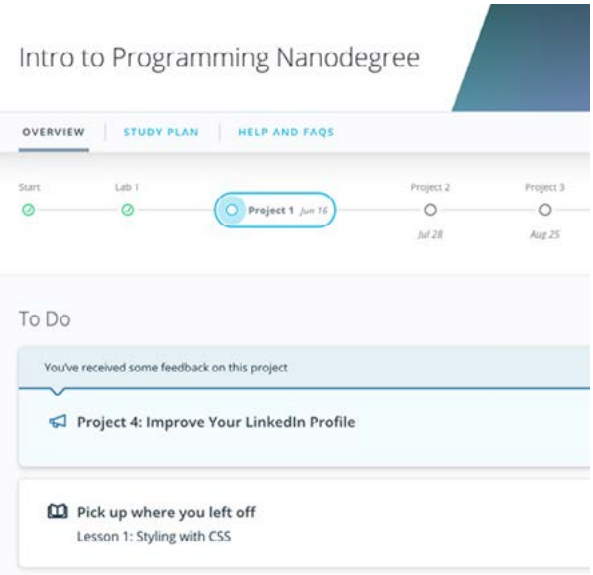
Students will be hosting and creating an automated deployment pipeline for a full stack application. You will have the option of using your own project or using a provided full stack application. During the project students will provision AWS infrastructure and prepare the application for hosting it.

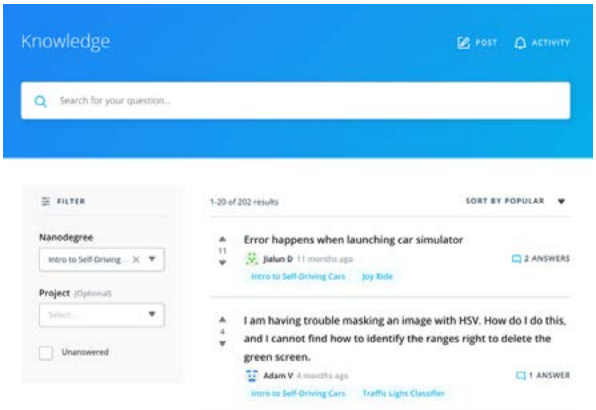
After this is done, you will move on to creating different scripts for deploying the application and creating a deployment pipeline. This project will expose students to a key process that is present on most software development projects.

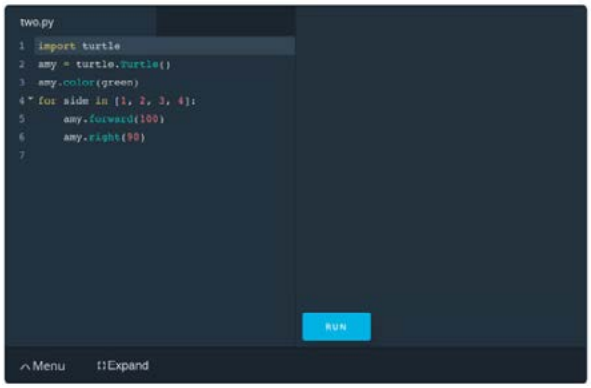


	LEARNING OUTCOMES	
LESSON ONE	**Setting Up a Production Environment**	• Create and configure a Postgres database in RDS. • Start an elastic beanstalk environment using the pre-configured node template. • Create environment properties in Elastic Beanstalk. • Create an S3 bucket and configure it for web hosting.
LESSON TWO	**Interact with Cloud Services**	• Ensure an application is healthy by using the Elastic Beanstalk CLI. • Apply code changes in an S3 bucket by using the AWS CLI to update the bucket content. • Apply code changes in Elastic Beanstalk by using the EB CLI to deploy a new application version.
LESSON THREE	**Write Scripts For Web Applications**	• Create scripts to build back-end and front-end applications. • Create scripts to deploy different pieces of a full stack application. • Create scripts that ensure a build follows quality standards. • Create a root level script to install all application dependencies.
LESSON FOUR	**Configure and Document a Pipeline**	• Connect a repository to CirceCi. • Create a manual approval step for deployments. • Create an automated integration process. • Create documentation using markdown files and diagrams. • Create an automated deployment process.

Our Classroom Experience







REAL-WORLD PROJECTS

Build your skills through industry-relevant projects. Get personalized feedback from our network of 900+ project reviewers. Our simple interface makes it easy to submit your projects as often as you need and receive unlimited feedback on your work.

KNOWLEDGE

Find answers to your questions with Knowledge, our proprietary wiki. Search questions asked by other students, connect with technical mentors, and discover in real-time how to solve the challenges that you encounter.

WORKSPACES

See your code in action. Check the output and quality of your code by running them on workspaces that are a part of our classroom.

QUIZZES

Check your understanding of concepts learned in the program by answering simple and auto-graded quizzes. Easily go back to the lessons to brush up on concepts anytime you get an answer wrong.

CUSTOM STUDY PLANS

Create a custom study plan to suit your personal needs and use this plan to keep track of your progress toward your goal.

PROGRESS TRACKER

Stay on track to complete your Nanodegree program with useful milestone reminders.

Learn with the Best

Guillaume Bibeau Laviolette

SOFTWARE ENGINEER

Guillaume is a software developer that has worked for companies such as Shutterstock and Filevine as a software and cloud engineer. He obtained his bachelor's of statistics and probability at McGill University in Canada.

Rachel Manning

FULL STACK DEVELOPER

Rachel is a full stack freelance developer and educator. As an advocate for continued learning, she is passionate about mentoring women and the underserved community in technology.

Alyssa Hope

SOFTWARE ENGINEER

Alyssa is a full stack developer who was previously the lead instructor at a coding bootcamp. With a degree in International Communications, her passion is to express thoughts well, whether in code or writing.

Andrew Wong

FULL STACK ENGINEER

Andrew is a full stack engineer who enjoys making the world a better place through code. He first discovered his passion for teaching as an instructor at App Academy, and continues to enjoy empowering students to advance their education.

All Our Nanodegree Programs Include:

EXPERIENCED PROJECT REVIEWERS

REVIEWER SERVICES

- Personalized feedback & line by line code reviews
- 1600+ Reviewers with a 4.85/5 average rating
- 3 hour average project review turnaround time
- Unlimited submissions and feedback loops
- Practical tips and industry best practices
- Additional suggested resources to improve

TECHNICAL MENTOR SUPPORT

MENTORSHIP SERVICES

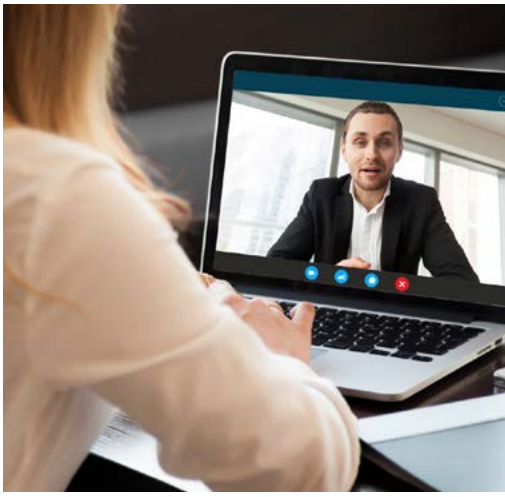
- Questions answered quickly by our team of technical mentors
- 1000+ Mentors with a 4.7/5 average rating
- Support for all your technical questions

PERSONAL CAREER SERVICES

CAREER SUPPORT

- Resume support
- Github portfolio review
- LinkedIn profile optimization

Frequently Asked Questions



PROGRAM OVERVIEW

WHY SHOULD I ENROLL?
This program was designed to help you take advantage of the growing need for skilled developers. Prepare to meet the demand for qualified professionals that can deliver modern web-based experiences.

WHAT JOBS WILL THIS PROGRAM PREPARE ME FOR?
The need for a strong programming culture in an enterprise organization is greater than ever. The skills you will gain from this Nanodegree program will qualify you for jobs in several industries as countless companies are trying to keep up with digital transformation.

HOW DO I KNOW IF THIS PROGRAM IS RIGHT FOR ME?
The course is for individuals who are looking to advance their developer careers with the cutting-edge skills to build a modern web applications.

ENROLLMENT AND ADMISSION

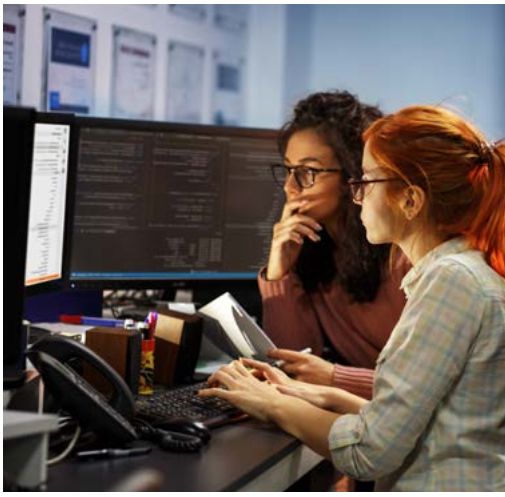
DO I NEED TO APPLY? WHAT ARE THE ADMISSION CRITERIA?
No. This Nanodegree program accepts all applicants regardless of experience and specific background.

WHAT ARE THE PREREQUISITES FOR ENROLLMENT?
A well-prepared student will meet the following prerequisites:
- Apply fundamental programming concepts (loops, conditionals, arrays, objects, functions) to write code that can be executed from the command line
- Use object-oriented programming features within JavaScript (i.e., build classes, instantiate objects, leverage the `this` keyword, etc.)
- Use HTML and CSS to build and style basic static webpages
- Use and configure HTML tags (attributes) and listen to DOM events
- Write simple programs in JavaScript using syntax, loops, conditionals, etc.
- Write asynchronous JavaScript calls with async/await syntax
- Read and write a JSON object

IF I DO NOT MEET THE REQUIREMENTS TO ENROLL, WHAT SHOULD I DO?
Students who do not feel comfortable in the above may consider taking Udacity's Intro to Programming Nanodegree program to obtain prerequisite skills.

FAQs Continued

TUITION AND TERM OF PROGRAM



HOW IS THIS NANODEGREE PROGRAM STRUCTURED?

The Full Stack JavaScript Developer Nanodegree program is comprised of content and curriculum to support 4 projects. We estimate that students can complete the program in 4 months working 10 hours per week.

Each project will be reviewed by the Udacity reviewer network. Feedback will be provided and if you do not pass the project, you will be asked to resubmit the project until it passes.

HOW LONG IS THIS NANODEGREE PROGRAM?

Access to this Nanodegree program runs for the length of time specified above. If you do not graduate within that time period, you will continue learning with month to month payments. See the **Terms of Use** and **FAQs** for other policies regarding the terms of access to our Nanodegree programs.

CAN I SWITCH MY START DATE? CAN I GET A REFUND?

Please see the Udacity Nanodegree program **FAQs** for policies on enrollment in our programs.

SOFTWARE AND HARDWARE

WHAT SOFTWARE AND VERSIONS WILL I NEED IN THIS PROGRAM?

There are no software and version requirements to complete this Nanodegree program. All coursework and projects can be completed via Student Workspaces in the Udacity online classroom.