

**Introduction to GraphQL**

**Course Number:** RCT-134  
**Duration:** 2 days

**Overview**

This Introduction to GraphQL training teaches attendees GraphQL basics, both backend (server-side handling of GraphQL requests) and frontend (web UI using React). For the backend, this class uses Node.js, TypeScript, and Apollo Server. For the front end, the course uses React, TypeScript, and Apollo Client.

**Prerequisites**

All attendees must be familiar with

* Modern JavaScript and/or TypeScript
* The basics of HTTP and REST
* Node.js and asynchronous programming
* React

**Materials**

All GraphQL training attendees receive comprehensive courseware.

**Software Needed on Each Student PC**

* Google Chrome and/or Firefox
* Other modern browsers as desired
* IDE/development environment of your choice
* Other free software and lab files that Accelebrate would specify, including Node.js

**Objectives**

* Master the GraphQL language and syntax
* Understand GraphQL schemas and the GraphQL SDL (schema definition language)
* Build backend services using GraphQL and Node.js
* Send queries and mutations to any GraphQL service
* Integrate GraphQL into a web-based frontend application

**Outline**

* Introduction
* GraphQL: Query Language and a Type System
* Unlike REST, GraphQL uses a Single Endpoint
* The Simplest Type: Scalars
* Add Comments to a Query
* Types: Scalars, Objects, Lists
* GraphQL Playground (Ctrl+space to show available fields; Cmd+click to GTD)
* A GraphQL API Queried Using Plain HTTP POST (curl/fetch)
* Schema Definition Language (SDL)
* Building a GraphQL Server (typeDefs and resolvers)
* Deeply Nested Field Selections
* Query Arguments (e.g., task ID)
* Aliases (e.g., the same field with different params)
* Multiple Queries Within a single request
* Argument/Parameter Defaults
* Named Queries (typically use UpperCamelCase)
* Mutations (e.g., createAccount)
* Query Variables
* Async Resolvers; Persisting Data to a Database
* Custom Resolver Chains for Deep Nesting and Circular Relationships
* Context, Headers, and Auth
* Union Types and Inline Fragments
* typename Field Implicit on all Types
* GraphQL Interfaces
* Building Frontend GraphQL Applications with React
  + Apollo Client
  + Tagged template literals and gql``
  + Using variables to pass dynamic input
  + Params received at `mutation()` level
  + Using React hooks: useQuery and useMutation
  + Refresh a query after a mutation
  + Mutations for authentication
  + Context in Apollo Server
  + Fragments
* Conclusion