**Beginner API deep dive: What is an API? -** [The Educative Team](https://educative-inc.medium.com/?source=post_page-----f79fdca458ce--------------------------------) Aug 11, 2022

APIs are a fundamental piece of modern web application design. They allow you to access services from many different companies and sources through a single web browser. Many people interact with APIs and don’t even realize it, such as a “Sign in with Facebook” button on a website or an app using your mobile camera.

If you’re interested in jumping into web app development, good API knowledge is essential to [landing a developer position](https://www.educative.io/blog/learning-company-api?eid=5082902844932096). Today, we’ll help you get started with APIs by exploring what an API is and what types you’ll see in the modern development environment.

**Here’s what we’ll cover today:**

* What is an API?
* Benefits of APIs
* Types of APIs
* What to learn next

**What is an API?**

API stands for **Application Programming Interface** and acts as a stable intermediary between two systems. Just as a user interface (UI) connects a user to a system, APIs connect one system to another system or one software component to another. Most APIs are known as **Web APIs** because they link web pages to other applications or databases.

APIs are often visualized as waiters in a restaurant, where the customer is one system and the kitchen is another.

The waiter API tells the customer what services are available, can relay requests to the kitchen, and can access restricted information like the order’s position in the queue. The customer doesn’t need to know the specifics of how this restaurant works or how their orders are processed and can instead simply placeOrder and trust that the waiter API will translate that to fit their backend kitchen system.

Diagram

Description automatically generated

From this, we can see that APIs provide a layer of abstraction to allow other systems to make API requests, or calls, without needing access to backend specifics.

For example, operating systems have APIs that let developers alter the appearance of a button. The developer doesn’t need to know what type of system the user has or manually code the behavior that makes it a button. Instead, they can simply ask the API to render a button and the API will translate that call into terms the operating system understands.

This abstraction is great for developers because it means they can design their product for the stable API rather than and don’t have to worry about manually implementing the feature for each system or their app breaking if a backend system is changed.

There are different types of APIs that developers will often use to make their apps more connected with outside technologies.

For example, [Facebook](https://www.educative.io/blog/cracking-top-facebook-coding-interview-questions?eid=5082902844932096) API may allow users to share goals from their fitness app or Stripe API may allow an app to utilize a trusted third-party transaction service to complete purchases.

Overall, APIs increase connectivity among our devices and allow developers to focus more on design and less on implementation.

**Benefits of APIs**

Abstraction isn’t the only benefit that APIs offer. Let’s explore some of the best benefits you can get with the use

**External Usage**

If your app becomes popular, other technologies may ask for an API to interact with your system. As a result, you’ll essentially receive free publicity, and eventually users, because it introduces your product to new populations of users.

For example, web applications may include a “Buy with Amazon” button to provide an easy next step for customers. This improves the customer experience of app users. At the same time, [Amazon](https://www.educative.io/blog/why-amazon-interview-question?eid=5082902844932096) benefits because it exposes users to their brand every time they log in, even if it’s on a non-Amazon application.

**Security**

Since APIs act as a buffer between systems, they allow you to choose which behaviors you allow and which you do not. Instead of all or nothing, users can make a more customized choice based on their privacy preferences.

For example, you may want to use an app’s Twitter API integration to post a picture on social media but do not want the app to know your location. With APIs, you can use one service (the post) without automatically allowing another (location sharing). Since there is a set number of APIs that can access a mobile location, it’s easy for the OS to identify when it’s attempted and request approval from the user. This choice will make your end users more comfortable and make their experience better.

**Faster Production**

APIs make it easier for programmers to build applications quickly because they streamline the number of devices they have to consider and allow them to import external APIs to add certain key functionalities. For example, you could create an app that allows users to find events of rising artists in their area.

The primary interest of these developers should be to ensure that the event creation system works well and that the users can browse the results with ease. While they still need a map and sales system, they can use partner APIs like [Google](https://www.educative.io/blog/google-coding-interview?eid=5082902844932096) Maps API to render an accurate map and a Salesforce eCommerce API to handle sales.

**Automation**

APIs are also incredibly useful for automation testing. Before, testers had to rely on custom scripts, outside web services, and direct integrations to handle their testing needs. While this works at a small scale, the direct integrations make testing environments inflexible when dealing with different tools and limited to low amounts of tests. The abstraction provided by APIs allows testers to create general testing suites in either XML or JSON files and use them on any system integrated with their API.

**Types of APIs**

APIs are categorized by three factors: who is supposed to use them, what protocols they use, and how they’re integrated with other aspects of the system. Knowing these types will help you use APIs, steer [API design](https://www.educative.io/blog/python-fastapi-tutorial?eid=5082902844932096), and even during current API management. We’ll look at each different API examples to understand the differences between each type.

**Release policies**

APIs allow a level of access to whoever integrates them, however, we don’t want to provide the same access to consumers as we do to enterprise clients or other internal teams. As result, we have release policies that define how much access the API grants into the system.

There are three tiers of access: Private APIs, Partner APIs, and Public APIs.

* **Private APIs** are internal APIs that are only used within the company. They offer the highest level of access and allow those in the same work ecosystem to access data, edit functionalities, and see some backend complexity if they wish.

Private APIs are great for large software development companies like Microsoft that are split into many different teams because it allows all teams to access a standard set of web services.

* **Partner APIs** are shared with select business partners to allow approved uses to access the API. For example, eBay could use a Partner API system to allow vetted vendors to shop from their listings, but low-quality or unsafe vendors cannot.

This allows the app developers to enforce quality control standards and keep their brand reputation high by only allowing high-standard apps to use their APIs.

* **Public APIs**, sometimes called “Open APIs”, are available for use by the public in a similar format to open-source software. Most platform APIs like Microsoft or Apple’s API Cocoa are Public to allow developers to easily make apps on their platforms.

**API Protocols**

Protocols are a set of standardized rules that define how APIs should be constructed to make it easier to implement new APIs quickly and in a consistent format.

* [**REST APIs**](https://www.educative.io/blog/what-are-rest-apis#rest-api?eid=5082902844932096) stands for Representational State Transfer. It is primarily used to share documents in web services. REST applications are designed to prioritize the scalability of components and the simplicity of interfaces.

To do so, REST rules dictate that you only use CRUD functions, regardless of the complexity of the command. That means REST applications use simple HTTP methods like GET, POST, DELETE, and PUT.  
While it may seem counterintuitive to neglect a portion of your tools, it ultimately forces you to describe complex behaviors in simple, scalable terms.

* **RPC** stands for Remote Procedure Call and is designed to make API calls to server methods rather than share documents. In other words, REST APIs focus on pulling resources from the server to perform actions on the client-side, while RPC focuses on sending action requests for the server to perform. This leads to a more consistent experience but is not as scalable.

RPC is an older style of protocol and is largely overlooked in favor of REST APIs.

* **SOAP** is an acronym for Simple Object Access Protocol. It’s a standard communication protocol that allows communication across different operating systems. It’s primarily used to create, update, and pull data from shared databases.

SOAP is designed to prioritize versatility by supporting all major operating systems and web service programming languages such as Python, JavaScript, and Java.

**API Architectural style**

Application architecture is the rules and design principles that steer the structure and flow of a system. Protocols are one aspect of a system’s architecture. If protocols are the rules for micro-level API messaging, architectures are the macro-level plan for how everything fits together.

There are two major types of architecture that dominate the web services field:

* **Service-Oriented Architecture (SOA)** is the older of the two architectures. Unlike previous monolithic apps that handled everything, SOA apps outsource some functions to different apps that are all loosely coupled through an integration pattern.

While SOA solved many of the issues of monolithic structures, the lack of centralized control could introduce complexity if the other coupled apps are updated.

* [**Microservice Architecture**](https://www.educative.io/blog/microservices-architecture-tutorial-all-you-need-to-get-started?eid=5082902844932096) is the iteration of SOA. It maintains the loosely coupled structure but adds a common communication system, often via RESTful APIs. This system allows separate components to communicate faster by cutting data conversions between services. Microservice Architectures are almost always used in conjunction with REST APIs.

They’re gaining popularity because each service can be iterated and improved without the risk of affecting other services. Each is isolated enough to be adaptable and is abstracted such that changes to one service do not break others.

**What to learn next**

APIs can be tricky to understand as a beginner but will quickly become a valuable tool for you as a web application developer. As you continue to explore APIs, your next step is to learn to use some of the popular APIs, like:

* [**Stripe API**](https://www.educative.io/courses/integration-stripe-api?eid=5082902844932096) to enable in-app purchases
* [**Auth0 API**](https://www.educative.io/courses/integrate-auth0-java-servlet?eid=5082902844932096)for client authentication
* [**Facebook API**](https://www.educative.io/courses/facebook-login-graph-api?eid=5082902844932096) for Facebook integration
* [**Algolia**](https://www.educative.io/courses/start-algolia-api?eid=5082902844932096) for fast search results
* [**Spotify API in Python**](NULL)for gathering music statistics
* [**Slack API in Python**](https://www.educative.io/courses/automations-slack-api-python?eid=5082902844932096)for Slack automation and integration

To help you learn your next API fast, Educative has created an **API course collection**. This collection features an ever-growing list of short courses, like the ones linked above, that help you pick up APIs quickly with hands-on practice. In August 2022 alone, we released **over 30 courses for APIs** across many different industries and uses. In the end, you’ll earn a certificate that shows your mastery of the API that you can share on your LinkedIn and resume!

**Continue reading about APIs on Educative**

* [How learning a company’s API helps you in your interview](https://www.educative.io/blog/learning-company-api?eid=5082902844932096)
* [Python REST API tutorial: Getting started with FastAPI](https://www.educative.io/blog/python-fastapi-tutorial?eid=5082902844932096)
* [3 Features beginner devs overlook when building APIs](https://www.educative.io/blog/three-overlooked-api-features?eid=5082902844932096)