# API Design Styles

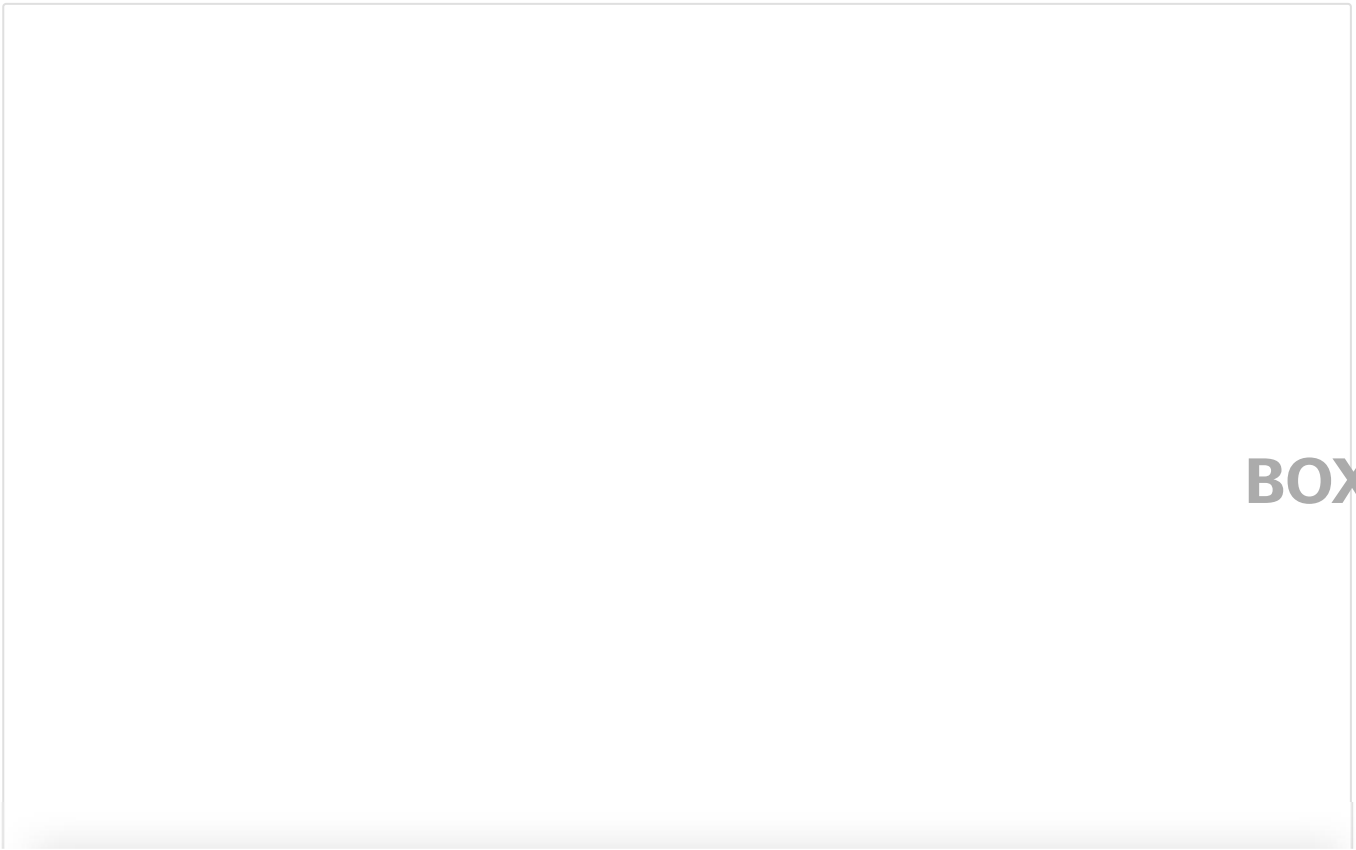
4.2.1

## Types of design styles

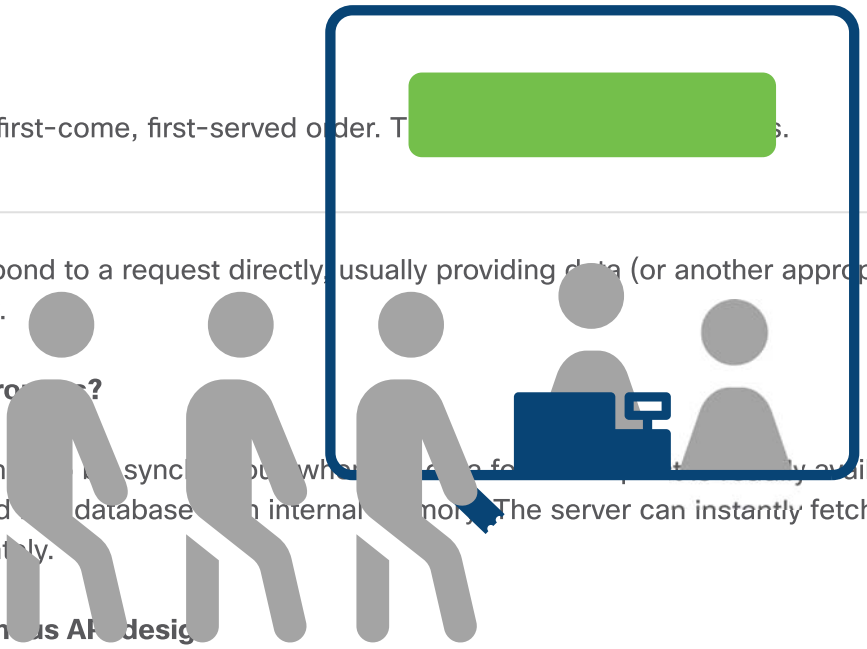
APIs can be delivered in one of two ways: synchronously or asynchronously. You need to know the difference between the two, because the application that is consuming the API manages the response differently depending on the API design. Each design has its own purpose, but also its own set of complexities on the client and/or server side. A product's set of APIs may consist of both synchronous and asynchronous designs, where each API's design is independent of the others. For best practices, however, the logic behind the design should be consistent.

4.2.2

## Synchronous APIs



Tickets are sold in a first-come, first-served order. T s.



Synchronous APIs respond to a request directly, usually providing data (or another appropriate response) immediately.

**When are APIs synchro e?**

APIs are usually design sync ou whe fo available, such as when the data is stored database n internal mor The server can instantly fetch this data and respond back immediat ly.

**Benefits of a synchron us AP desig**

Synchronous APIs enable the application to receive data immediately. If the API is designed correctly, the application will have better performance because everything happens quickly. However, if it is not designed correctly, the API request will be a bottleneck because the application has to wait for the response.

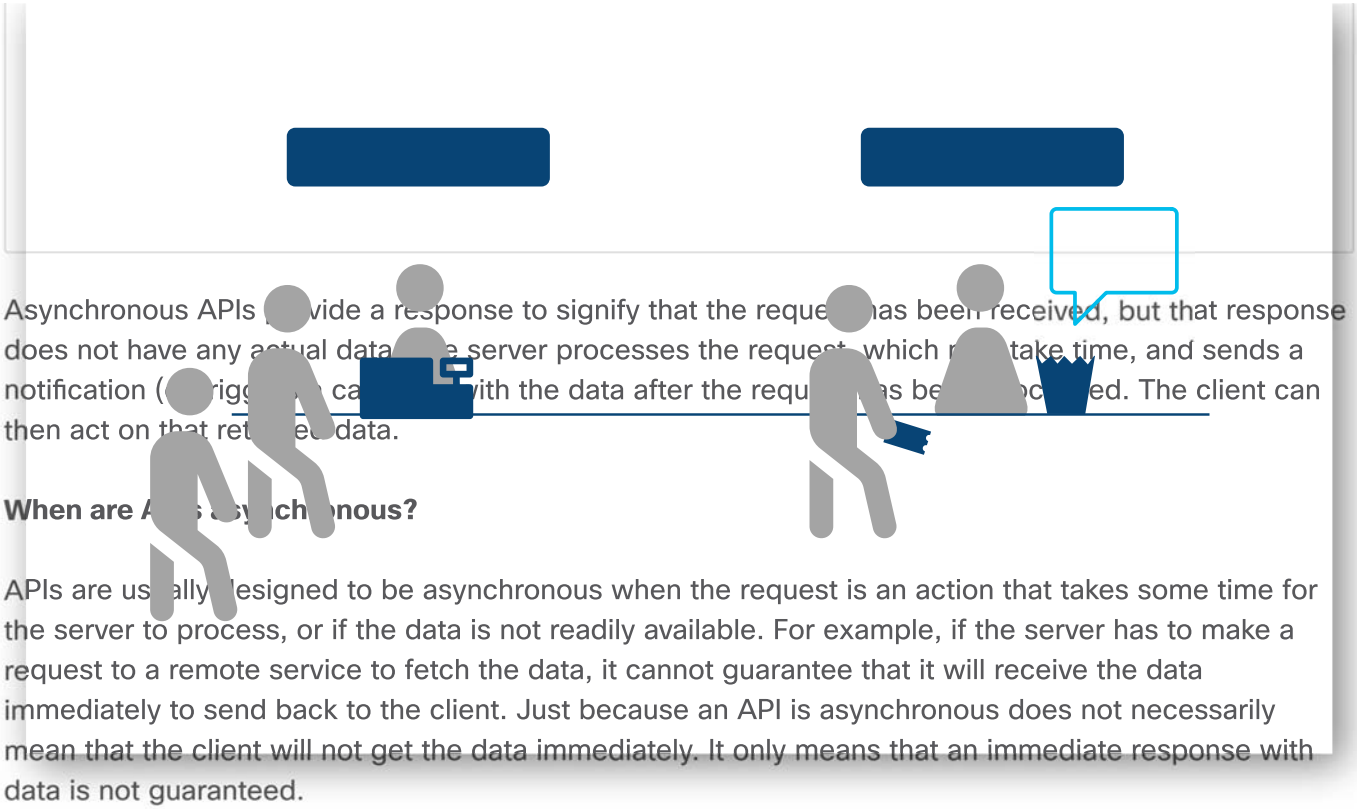
**Client side processing**

The application making the API request must wait for the response before performing any additional code execution tasks.

4.2.3

## Asynchronous APIs

ORDER HERE



Asynchronous APIs provide a response to signify that the request has been received, but that response does not have any actual data. The server processes the request, which may take time, and sends a notification (trigger) callback with the data after the request has been processed. The client can then act on that returned data.

**When are APIs asynchronous?**

APIs are usually designed to be asynchronous when the request is an action that takes some time for the server to process, or if the data is not readily available. For example, if the server has to make a request to a remote service to fetch the data, it cannot guarantee that it will receive the data immediately to send back to the client. Just because an API is asynchronous does not necessarily mean that the client will not get the data immediately. It only means that an immediate response with data is not guaranteed.

**Benefits of an asynchronous API design**

Asynchronous APIs let the application continue execution without being blocked for the amount of time it takes for the server to process the request. As a result, the application may have better performance because it can multi-task and make other requests. However, unnecessary or excessive use of asynchronous calls can have the opposite effect on performance.

**Client-side processing**

With asynchronous processing, the design of the API on the server side defines what you want to do on the client side. Sometimes the client can establish a listener or callback mechanism to receive these notifications and process them when they are received. Depending on the design of the application, your client may also need a queue to store the requests to maintain the order for processing. Other API designs need the client to have a polling mechanism to find out the status and progress of a given request.

4.1 Introducing APIs

4.3 API Architectural Styles