API Interview Questions

"An API is a set of codes which allows two or more than two applications to communicate each other internally or externally and provide a result to end users or to another API."

I got some feedback to explain "internally or externally" from definition of API above.

Suppose you are developing an application and it has many components. A component will communicate to other component to get data or provide data. For example: Login component of Facebook application needs to communicate "Profile" component to get user's data. Here these components will communicate to each other through an API. It is internal communication as both components within same application talking to each other's. It is an example of internal communication.

Let's see other aspect now. You download some apps and it will ask you to signup/Login with Facebook on launching app. You click on it to login with Facebook and you are promoted to dialog box where you need to provide some permissions and you are logged in. Here App and Facebook both are different applications but they communicated and it is possible because of APIs. App used Facebook open APIs to provide facility to login using Facebook. It is an example of external communication.

At a very high level, an application has three components:

1. **Back end (Data Layer) : Where data is stored and retrieved.**
2. **Front end (Presentation Layer) : User interface.**
3. **Middle ware (Logic Layer) : It connects front end and back end of application.**

URN – Uniform Resource **Name**
URL – Uniform Resource **Locator**
URI – Uniform Resource **Identifier**

What is Resource?

Resource is a physical or virtual component, which supports/extends/represents/constructs a system. Consider a real time example of Motor Bike. Wheels, breaks, headlight, fuel tank etc all are resources of a Motor bike. All resources have names, location and identifiers.

Take another example of Google. Now append “/maps” in last of base url “https://www.google.com/” as “https://www.google.com/maps“. Google will launch Google Maps. Try same with “/news”. Google will launch Google News. Here “Maps” and “News” are actually resources which has its names and locator.

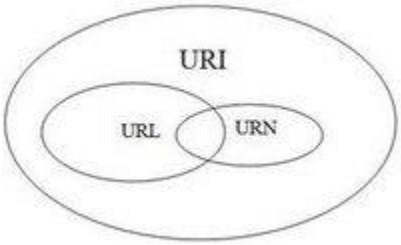
A Uniform Resource Identifier (URI) is a string of characters used to identify a name or a resource on the Internet.

A URI identifies a resource either by location, or a name, or both. **A URI has two specializations known as URL and URN.**

A Uniform Resource Locator (URL) is a subset of the Uniform Resource Identifier (URI) that specifies where an identified resource is available and the mechanism for retrieving it. A URL defines how the resource can be obtained. It does not have to be a HTTP URL (http://), a URL can also start with ftp:// or smb://, specifying the protocol that's used to get the resource.

A Uniform Resource Name (URN) is a Uniform Resource Identifier (URI) that uses the URN scheme, and **does not imply availability of the identified resource**. Both URNs (names) and URLs (locators) are URIs, and a particular URI may be both a name and a locator at the same time.

This diagram (source) visualizes the relationship between URI, URN, and URL:

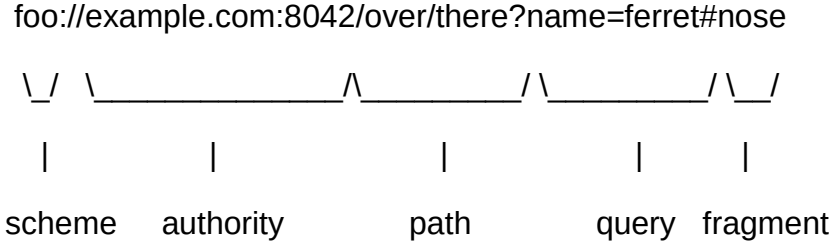


A URN is similar to a person's name, while a URL is like a street address. The URN defines something's identity, while the URL provides a location.

- A URL is a URI that identifies a resource and also provides the means of locating the resource by describing the way to access it
- A URL is a URI
- A URI is not necessarily a URL

URL: ftp://ftp.is.co.za/rfc/rfc1808.txt
URL: http://www.ietf.org/rfc/rfc2396.txt
URL: ldap://[2001:db8::7]/c=GB?objectClass?one
URL: mailto:John.Doe@example.com
URL: news:comp.infosystems.www.servers.unix
URL: telnet://192.0.2.16:80/
URN (not URL): urn:oasis:names:specification:docbook:dtd:xml:4.1.2
URN (not URL): tel:+1-816-555-1212 (disputed, see comments)

URI = scheme ":" hier-part ["?" query] ["#" fragment]



An API is developed to serve a purpose/functionality which may fall in one of these categories:

Creating data , Retrieving data, Modifying data or Delete data.

Above are basic operations performed by APIs. These functions are called as CRUD(Create, Retrieve, Update and Delete) as an acronym. In REST, these operations are called as HTTP methods. There are more HTTP methods other than these.

We will see these HTTP methods here:

1. **GET**
2. **POST**
3. **PUT**
4. **DELETE**
5. **PATCH**

6. **OPTIONS**
7. **HEAD**

GET:

This HTTP method is used to read/retrieve resource representation only. It is called Safe methods as it can not modify information. It should retrieve same information for multiple identical requests until any other API has changed the state of resource. That's why it is also called as idempotent method. Response is returned in JSON/XML format.

POST:

A HTTP POST method is used to create a new resource in collection of resources with a request body passed as a JSON/XML. If resource is created successfully at the end point, it returns a status code 201(Created) (Not always) and returns response body. It may return 200 (OK) and 204 (No Content) status code as well based on how it is created.

*POST is not safe method as it is related to data creation. It is also not **idempotent** and invoking two identical POST requests will result in two different resources containing the same information with just different resource ids.*

PUT:

An HTTP PUT method is used to primarily update the resource information but it also can be used to create a new resource (Depends on API development) if requested resource is not available. If PUT request is made to update resource, it should return 200 (OK) and 204 (No Content) status code. If PUT request is made to create a new resource, it must return a status code 201(Created).

PUT is not a safe method as it performs data creation and modifications but it is idempotent as if we hit the same request again, it operates on same existing resource. But note here that a PUT request can be made as non-idempotent as well.

DELETE:

An HTTP DELETE method is used to delete an existing resource from collection of resources. On successful deletion of resource, it returns 200 (OK) and 204 (No Content) status code. It may return as 202 (Accepted) status code if request is queued.

It is not a safe method as it performs on modification of data. If we hit the same request again after first hit, it will give you 404 (Not Found) . So DELETE request are idempotent after second call onward.

PATCH:

An HTTP PATCH method is used to update information of resource partially. It is different from PUT as PUT updates/replace complete information of resource while PATCH updates some information of resource. It returns 200 (OK) and 204 (No Content) status code

A PATCH method is not safe method as it operations on modification of data. It is also non-idempotent but can be made idempotent.

HEAD:

An HTTP HEAD method is identical to GET method without response body. Instead of response body or resource information, a GET request returns meta information/headers contented in an HTTP GET method. This method can be used for obtaining meta information about the entity implied by the request without transferring the entity-body itself.

OPTIONS:

An HTTP OPTIONS method which is used to get information about allowed operations on given URI. It returns a response header named “Allow” with the list of available operation on given URI.

If an HTTP method does not change/modify the resource information on the server side or perform read only operation, is called a SAFE HTTP Method.

GET, HEAD and OPTIONS HTTP methods are safe methods. This methods perform read only operations. POST, PUT etc are unsafe methods.

Now we will see some status codes which we must know.

200 (OK) => All is GOOD. It is the most positive status code which everyone expects. This status code is thrown when requested operation on server by client is successfully processed.

201 (Created) ==> This status code is thrown when an HTTP method to hit to create a new resource on server and resource is created successfully. This status code makes more sense for HTTP methods which are meant to create new resources.

202(Accepted) ==> If a request is queued for processing or takes longer time to process, this status code is thrown to client. The request might or might not be eventually acted upon, or even maybe disallowed when processing occurs.

204 (No Content) ==> When a request is processed successfully but returns no state representation of resource to be included in the response message body, throws 204 status code. Its success without response body.

400 (Bad Request) ==> It is a client side error made when user submits inappropriate request like malformed request syntax, invalid request message parameters, or deceptive request routing etc. Client should correct request before hitting again.

401 (Unauthorised) ==> When a client submits a request with no or wrong authorisation on a resource which is protected by authorisation , this status code is thrown. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource.

403 (Forbidden) ==> When a user tries to access a resource on which user has no permission, this error will be thrown. It is not like 401. Request and proper authorization is provided in request body but user has no access on request resource.

404(Not Found) ==> If the request resource is not available at given URI or rest api can’t map the client’s URI to a resource, this status code is thrown.

405(Method Not Allowed) ==> When a client calls an HTTP method on a resource which is not applicable to on it, this status code is thrown. For example: If client hits a POST method on a GET resource, it will throw 405. Response includes “Allow” header, which lists the HTTP methods that the resource supports.

406(Not Acceptable) ==> If API can not format data as per client's provided media types in Accept request header, this status code is thrown. For example, a client request for data formatted as application/xml will receive a 406 response if the API is only willing to format data as application/json.

415 (Media Type) ==> If API is not able to process the client's supplied media type, as indicated by the Content-Type request header, this status code is thrown. It indicates that part of the request was in an unsupported format. For example, a client request including data formatted as application/xml will receive a 415 response if the API is only willing to process data formatted as application/json.

500 (Internal Server Error) ==> It is server side error. When server encounters an exception while processing a request, this status code is thrown. Please note here it is not client side error.

501 (Not Implemented) ==> When the server does not recognize the request method and is not capable of supporting it for any resource, this status code is thrown. The server either does not recognize the request method, or it lacks the ability to fulfill the request.

#State The Core Components of an HTTP Response?

Every HTTP response contains four key elements.

- Status/Response Code – These are response codes issued by a server to a client's request. For example, 404 means Page Not Found, and 200 means Response is OK.
- HTTP Version – describes HTTP version, for example-HTTP v1.1.
- Response Header – Includes information for the HTTP response message. For example, Content-type, Content-length, date, status and server type.
- Response Body – It contains the data that was requested by a client to server.

#What is meant by the term environment in postman?

- An environment in postman is a set of key value pairs. You can create multiple environments in postman which can be switched quickly with a press of a button. There are 2 types of environment, global and local.

#Can global scope variables have duplicate names in postman?

- Since global variables are global i.e. without any environment, global variables cannot have duplicate names. Local variables can have the same name but in different environments.

#Which one will be preferred in postman- a global variable or a local variable?

- In postman, if 2 variables have the same name(one being local, other global) then the higher priority is of the local variable. it will overwrite the global variable.
- For all variables, the **initial value** is the one that's syncd to the Postman servers and can be seen by you and your team members. The **current value** is only seen by you and is not syncd to the Postman servers.

#What is a Postman Collection?

- A Postman Collection lets us group individual requests together. Simply it allows us to organize the requests into folders.

#What do you mean by postman monitors?

The postman monitor is used for running collections. Collections are run till specified time defined by the user. Postman Monitor requires the user to be logged in. Monitor reports are shared by users over email on a daily/monthly basis.

#**What do you understand by the term Postman Collection runners?**

A postman collection runner is used to perform Data-driven testing. The group of API requests are run in a collection for the multiple iterations with different sets of data.

#Can local variables be imported in Postman Monitors?

Yes. Postman monitors allow to import local variables but it does not allow to import global variables.

#What is the purpose of Postman cloud if we are working in a company? Why?

A Postman cloud is a common repository of companies to access Postman collections. In Postman cloud, work can be saved instantly after logging in. Anyone from the team can access data/collections from anywhere.

#Why is it not preferred to save work in Postman cloud?

It is not preferred to save your work in Postman cloud as company's work is not allowed to be leaked and remain confidential. Security breaches can be experienced if Postman cloud is used as Postman cloud requires sign in. Therefore Postman Cloud is discouraged for saving work and team workspace is highly encouraged.

#What is the purpose of status code 304?

It means NOT MODIFIED. It is used to reduce network bandwidth usage in case of conditional GET requests. Response body should be empty. Headers should have date, location etc.

#Define status code 201?

It means created, when a resource is successfully created using POST or PUT request. It returns a link to a newly created resource using the location header.

#When do we use global variables, collection variables, and local variables?

Global variables are general purpose variables, ideal for quick results, and prototyping. They are used while passing data to other requests.

Collection variables can be mostly used for storing some constants that do not change during the execution of the collection. They are used for constants that do not change during the execution and also for URLs / authentication credentials if only one environment exists.

Local variables are only available within the request that has set them or when using Newman/Collection runner during the entire execution. They are used whenever you would like to override all other variable scopes.

#How do you remove local variables?

Local variables are automatically removed once the tests have been executed.