**Parameters vs Arguments:**

void Foo(int i, float f) { // Do things } >> void Bar() { int anInt = 1; Foo(anInt, 2.0); }

Here i and f are the parameters, and anInt and 2.0 are the arguments.

//https://www.freecodecamp.org/news/node-js-what-when-where-why-how-ab8424886e2/

**Process**

A process is a program under execution, i.e. a running program and is created when a program starts execution. A process can have multiple threads.

**Threads**

A thread is the smallest sequence of programmed instructions that can be managed independently by a scheduler, which is typically a part of the operating system.

The primary difference is that threads within the same process run on a shared memory space, while processes run in separate memory spaces.

**Multi-processing**

Multiprocessing is the use of two or more CPUs (processors) within a single computer system. Now, as there are multiple processors available, multiple processes can be executed at a time.

**Multi-threading**

Multi-threading is an execution model that allows a single process to have multiple code blocks (threads) running concurrently within the “context” of that process.

**Thread pool**

A thread pool is a group of pre-instantiated, idle threads which stand ready to be given work. By maintaining a pool of threads, the model increases performance and avoids latency in execution, due to frequent creation and destruction of threads for short-lived tasks.

**1. Explain different types of Web Services.**

There are basically two types of web services:

**SOAP** (Simple Object Access Protocol) Web Services: It is also referred to as transport-independent messaging protocol whose main purpose is to transfer a message, and is based on XML protocol.

**RESTful** (Representational State Transfer) Web Services: It is developed to fulfill the shortcomings of SOAP and to make the web services more effective.

**Write the difference between API and Web services.**

API (Application Programming Interface): It acts as an interface between two devices so that they can communicate with each other without any user intervention. All APIs are not web services.   
Web Service: It facilitates interaction between two devices over a network. All Web services are APIs.

| **API** | **Web Service** |
| --- | --- |
| It can be online or offline. | It must use a network. |
| They are lightweight architecture. | They require SOAP to send and receive network data therefore, are not lightweight architectures. |
| It supports HTTP/HTTPS protocol, XML and JSON. | It supports HTTP protocol and also supports XML. |

**Define Statelessness & its advantages.**

Statelessness is basically a condition or restriction where RESTful web services are not allowed to keep a client state on the server as per the REST architecture. Clients are responsible to pass their context to the server. To process the client’s request, the server then further stores this context.  **Advantages:**

* No need to maintain previous interactions with clients.
* Independent treatment of each method request.
* Less complexity and simplified application design.

**Define SOAP & its advantages.**

SOAP (Simple Object Access Protocol) is an XML-based protocol that is used to access web services. It is simply used to interchange data or information between two devices using request and response based on XML format over transport protocols like HTTP, SMTP, etc. **Advantages:**

* Language and Information independent.
* Can be written in different programming languages or operating systems.
* Provide data transport for web services.
* Can extend HTTP for XML messaging.
* Defines and uses its own WS security.
* Easy to debug and eliminate firewall issues.

**What are the different elements of the SOAP Document or message?**

The SOAP message is basically an ordinary XML document consists of three parts as given below:   
  
**SOAP Envelope**: It is a mandatory element that identifies XML documents as a SOAP message. It simply defines the start and the end of the message.    
**SOAP Header**: It is an optional element that contains header information.    
**SOAP Body**: It is a mandatory element that contains call and response information. It includes XML data consisting of the message that is being sent.

**Why is Web service important?**

Web services are very important as they provide standardized web protocols I.e., HTTP or HTTPS simply to interoperate, communicate, and exchange information. Advantages include:

* Allows devices to talk to each other and share data or services between themselves.
* Makes the application platform and technology independent.
* Uses standardized standard protocol for communication.
* Can be used at the same time by many client applications.

**Loosely Coupled vs Tightly Coupled:**

**Promises** are the ideal choice for handling asynchronous operations in the simplest manner. They can handle multiple asynchronous operations easily and provide better error handling than callbacks and events.

**Web Applications**

A web application is a program that runs on a server and is rendered by a client browser, using the internet to access all the resources of that application. It can be broken down into 3 parts **Client, Server & Database**

**Client:** The user interacts with the front-end part of a web application. The front-end is usually developed using languages like HTML and [CSS](https://www.simplilearn.com/11-css-secrets-rar400-article) styles, along with extensive usage of JavaScript-based frameworks like [ReactJS and Angular,](https://www.simplilearn.com/react-vs-angular-vs-vue-article) which help with application design.

**Server:** The server is responsible for taking the client requests, performing the required tasks, and sending responses back to the clients. It acts as a middleware between the front-end and stored data to enable operations on the data by a client. Node.js, PHP, and [Java](https://www.simplilearn.com/best-java-programs-article) are the most popular technologies in use to develop and maintain a web server.

**Database:** The database stores the data for a web application. The data can be created, updated, and deleted whenever the client requests. [MySQL](https://www.simplilearn.com/mobile-and-software-development/fundamentals-of-mysql-training) and [MongoDB](https://www.simplilearn.com/tutorials/mongodb-tutorial/what-is-mongodb) are among the most popular databases used to store data for web applications.

Architectural Constraints of RESTful API: There are six architectural constraints which makes any web service are listed below:

Uniform Interface

Stateless

Cacheable

Client-Server

Layered System

Code on Demand

The only optional constraint of REST architecture is code on demand. If a service violates any other constraint, it cannot strictly be referred to as RESTful.

Uniform Interface: It is a key constraint that differentiate between a REST API and Non-REST API. It suggests that there should be an uniform way of interacting with a given server irrespective of device or type of application (website, mobile app).

There are four guidelines principle of Uniform Interface are:

Resource-Based: Individual resources are identified in requests. For example: API/users.

Manipulation of Resources Through Representations: Client has representation of resource and it contains enough information to modify or delete the resource on the server, provided it has permission to do so. Example: Usually user get a user id when user request for a list of users and then use that id to delete or modify that particular user.

Self-descriptive Messages: Each message includes enough information to describe how to process the message so that server can easily analyses the request.

Hypermedia as the Engine of Application State (HATEOAS): It need to include links for each response so that client can discover other resources easily.

Stateless: It means that the necessary state to handle the request is contained within the request itself and server would not store anything related to the session. In REST, the client must include all information for the server to fulfill the request whether as a part of query params, headers or URI. Statelessness enables greater availability since the server does not have to maintain, update or communicate that session state. There is a drawback when the client need to send too much data to the server so it reduces the scope of network optimization and requires more bandwidth.

Cacheable: Every response should include whether the response is cacheable or not and for how much duration responses can be cached at the client side. Client will return the data from its cache for any subsequent request and there would be no need to send the request again to the server. A well-managed caching partially or completely eliminates some client–server interactions, further improving availability and performance. But sometime there are chances that user may receive stale data.

Client-Server: REST application should have a client-server architecture. A Client is someone who is requesting resources and are not concerned with data storage, which remains internal to each server, and server is someone who holds the resources and are not concerned with the user interface or user state. They can evolve independently. Client doesn’t need to know anything about business logic and server doesn’t need to know anything about frontend UI.

Layered system: An application architecture needs to be composed of multiple layers. Each layer doesn’t know any thing about any layer other than that of immediate layer and there can be lot of intermediate servers between client and the end server. Intermediary servers may improve system availability by enabling load-balancing and by providing shared caches.

Code on demand: It is an optional feature. According to this, servers can also provide executable code to the client. The examples of code on demand may include the compiled components such as Java applets and client-side scripts such as JavaScript.

Rules of REST API: There are certain rules which should be kept in mind while creating REST API endpoints.

REST is based on the resource or noun instead of action or verb based. It means that a URI of a REST API should always end with a noun. Example: /api/users is a good example, but /api?type=users is a bad example of creating a REST API.

HTTP verbs are used to identify the action. Some of the HTTP verbs are – GET, PUT, POST, DELETE, GET, PATCH.

A web application should be organized into resources like users and then uses HTTP verbs like – GET, PUT, POST, DELETE to modify those resources. And as a developer it should be clear that what needs to be done just by looking at the endpoint and HTTP method used.

URI HTTP verb Description

api/users GET Get all users

api/users/new GET Show form for adding new user

api/users POST Add a user

api/users/1 PUT Update a user with id = 1

api/users/1/edit GET Show edit form for user with id = 1

api/users/1 DELETE Delete a user with id = 1

api/users/1 GET Get a user with id = 1

Always use plurals in URL to keep an API URI consistent throughout the application.

Send a proper HTTP code to indicate a success or error status.

Note : You can easily use GET and POST but in order to use PUT and DELETE you will need to install method override. You can do this by following below code :

HTTP verbs: Some of the common HTTP methods/verbs are described below:

GET: Retrieves one or more resources identified by the request URI and it can cache the information receive.

POST: Create a resource from the submission of a request and response is not cacheable in this case. This method is unsafe if no security is applied to the endpoint as it would allow anyone to create a random resource by submission.

PUT: Update an existing resource on the server specified by the request URI.

DELETE: Delete an existing resource on the server specified by the request URI. It always return an appropriate HTTP status for every request.

GET, PUT, DELETE methods are also known as Idempotent methods. Applying an operation once or applying it multiple times has the same effect. Example: Delete any resource from the server and it succeeds with 200 OK and then try again to delete that resource than it will display an error message 410 GONE.