**Web Services:**

* Service over the web
* For Communication between Applications over web
* Format of communication (protocol) - works between different platforms to exchange info.
* Platform independent communication.

**Constituents:**

1. Provider : Server - Creates Services.
2. Consumer : Client - Requests for a Service
3. Broker : Provides access to registry (Menu- UDDI) to describe, publish & discover web services - (Medium & Format)

**TYPE 1:**

1. XML based Message sent via HTTP
2. Header (Target Info / Authentication ) & Body (Message) sent in wrapper.
3. **SOAP -** Simple Object Access Protocol

**Web Service Standards**

* Ex:Registration Form - XML based interface to describe functionality - Format of message - WSDL
* Ex:Menu - Directory of WSDL - UDDI ( Universal Description, Discovery, Integration )
* Extended : WSIL / WS-I / etc.

**TYPE 2:**

1. Architecture of Resources governed through URL and Transfer protocol.
2. Assigning resources for different Services via HTTP.
3. Format - XML/ JSON/ TEXT etc
4. **RESTful WS -** REpresentational State Transfer (A Design Method)

**Principles of REST architecture**

* Resource (Noun)
* Address to find the resource with identifiers - URI
* Perform actions via HTTP Methods (GET, PUSH,POST, DELETE, etc) (Verb) - transmit message.
* Independent of the state of the server(X storage) - request contains all data to handle itself - Stateless.
* Response from server labeled Cacheable or Non Cacheable - Caching on client side
* Multiple services can be requested : Layering
* Can implement code on the server.(if needed)

Ex: JAVA Web Services (JAX-WS & JAX-RS)

**DIfferences:**

| **SOAP** | **REST** |
| --- | --- |
| * Strict Communication Standard | * Flexible Architecture with constraints |
| * XML + Structured/Bulky Data | * XML, JSON, Plain Text etc + Less Structure/Lighter Data |
| * Can't use REST | * Can use SOAP |
| * Uses Interface | * Uses URL |
| * Difficult to implement + more bandwidth. | * Easy to implement + Lower Bandwidth (Smartphones) |
| * Database Transaction - Yes | * Database Transaction - No |
| * Data Security is Better | * Data Security - Can be implemented |
| * Prefered Enterprise applications | * Designed for Mobile applications |

**API:**

Where you tell the server to perform an action:

Application ←—------- Programming ←—---------Interface

* Using rather than writing program
* Platform Independent
* No need to upgrade . (Application takes care of it you just request for the service)

How to write an API : [www.google.com/search?q=apple](http://www.google.com/search?q=apple)

Example: Zomato Food Order ( API calling APIs )

SOAP and REST are ways to send requests and receive responses over HTTP.