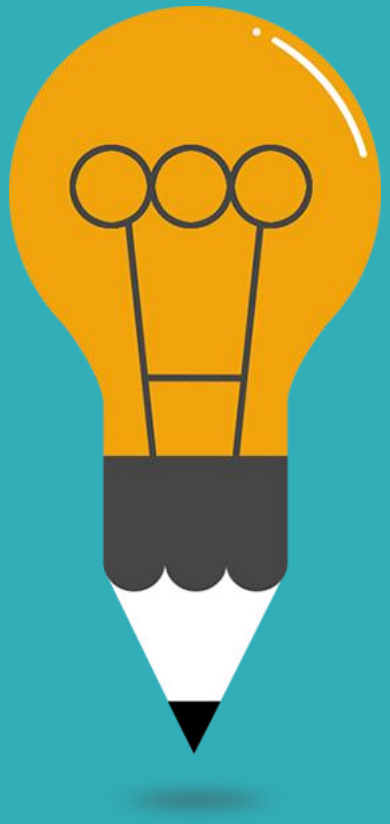
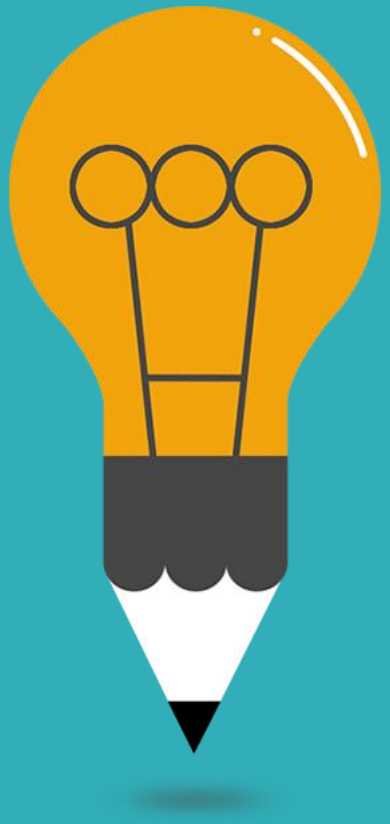


API Fundamentals and REST API Testing

Python Based Testing



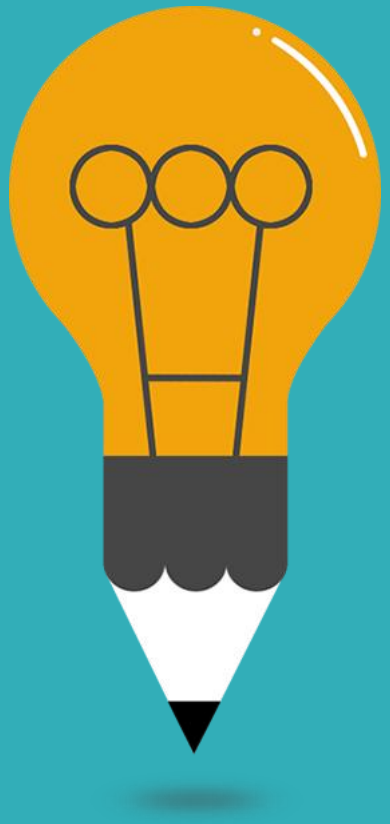
General Guidelines

Guidelines

01 Follow the sequence maintained in the course.

02 Increase or decrease the speed of the video.

03 Complete the course in 1-2 Days.



Webservices and API

Web Services

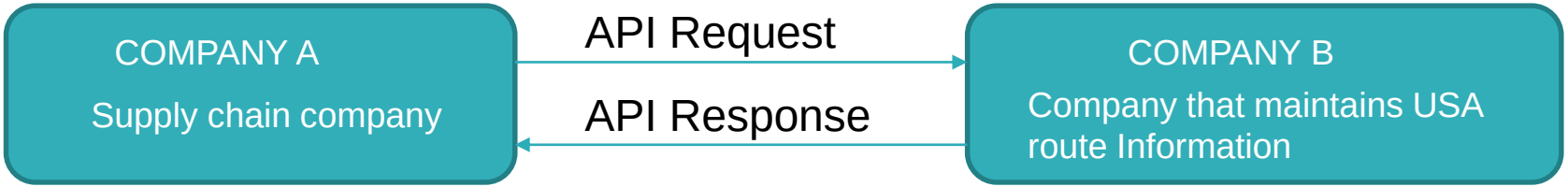
01 What is a Web Service ?

02 Understand the Webservice with an example

03 What is the Importance of Web Service? Why is it required ?

04 What are the advantages of a Web Service.

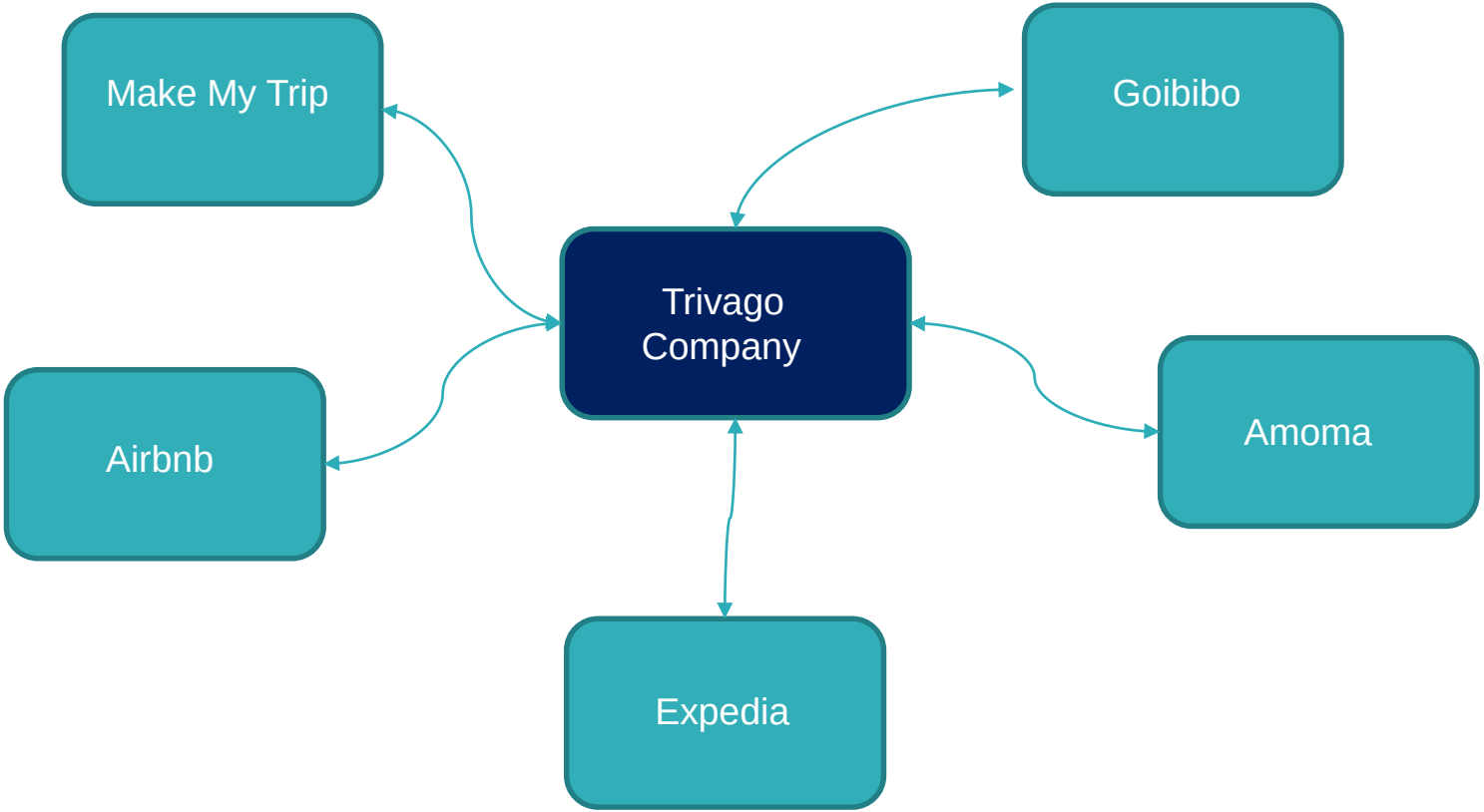
Web Services Example



- Planning to optimize the Transportation Cost.
- Need to improve the Overall Supply Chain Visibility.

- Provides Latest Route Details
- Max Load on the Road
- Tariff (Tax/Toll) on the Road
- Docking Information Cost

Web Services Example



Web Services Salient Features

Platform Independent

Client Server Communication.

Uses Web Standard Protocol / format for communication between machines

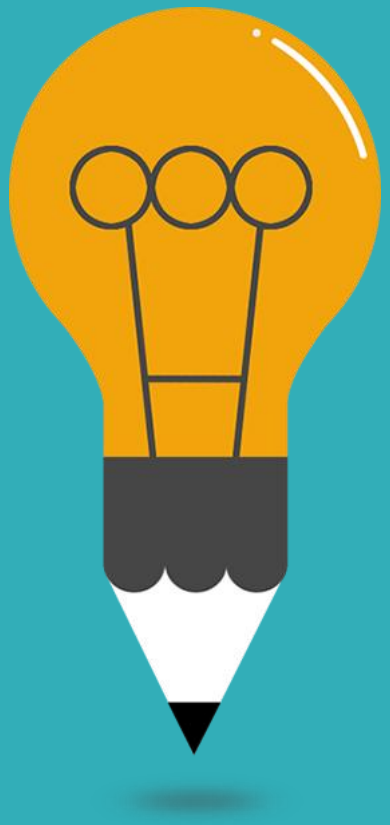
Available over Internet or private Networks.

There are lot of tools provided these days just to measure API Performance.

Uses HTTP Protocol methods like GET, POST, DELETE, PUT etc.

Trending Functionality and is used by most of the companies

Provides alternative for ETL Workflows



Types of API

API

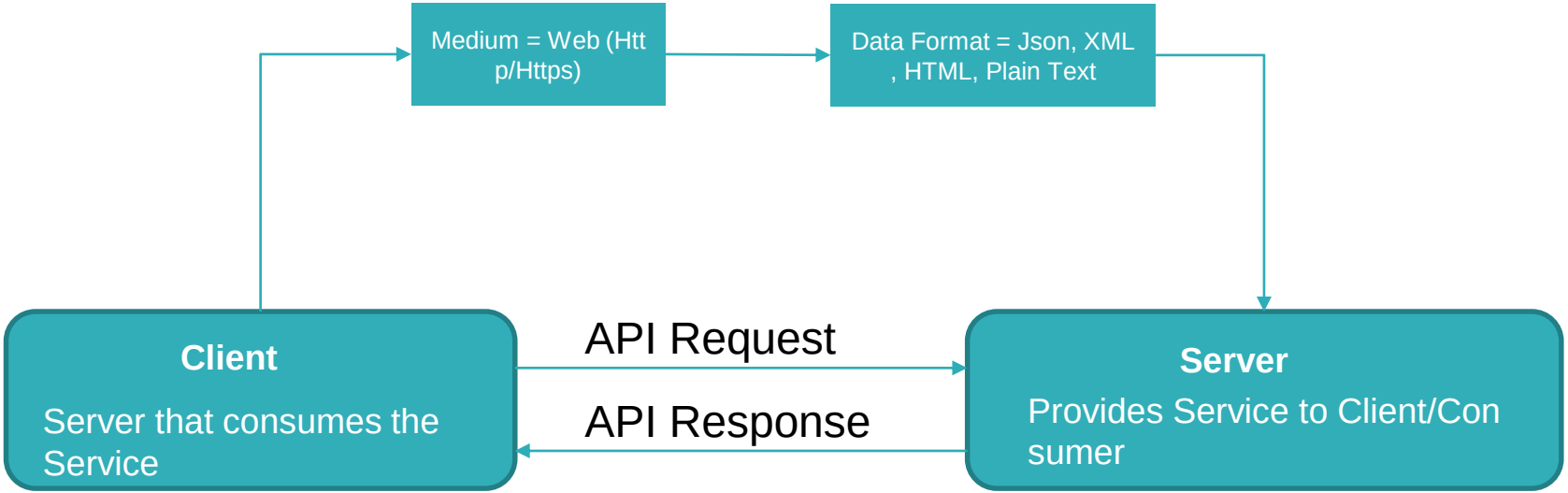
01 Types of API ?

02 What is Client Server Model?

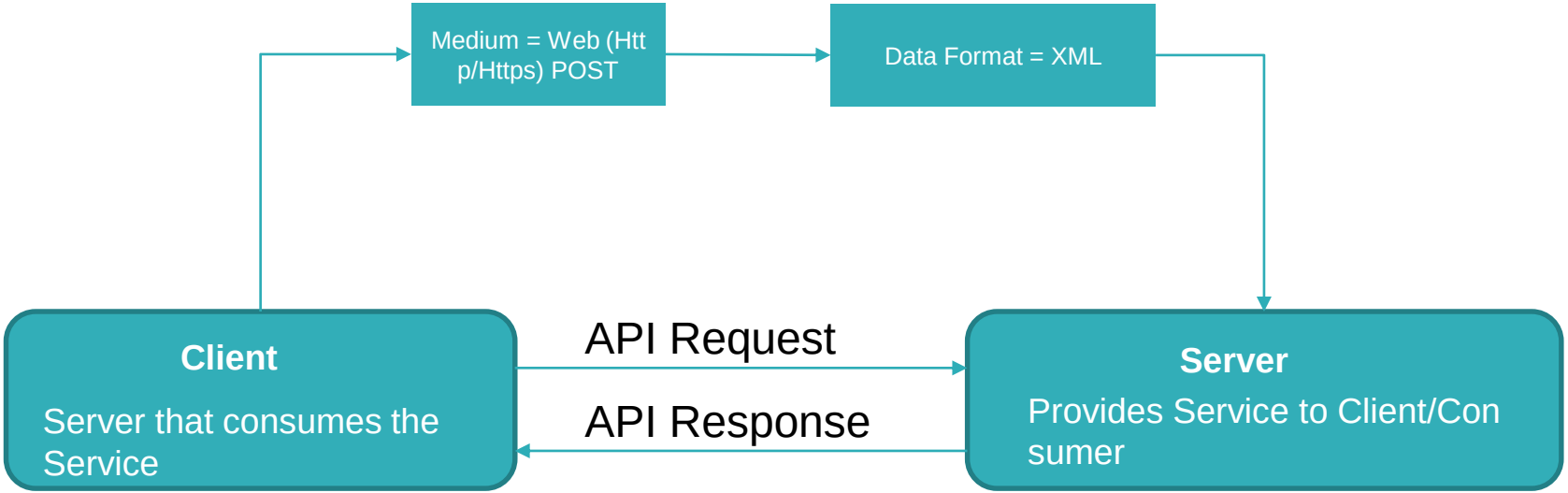
03 Difference Between SOAP and REST API ?

04 Medium and the Format of the Data used in SOAP/REST API?

Web Services Representation

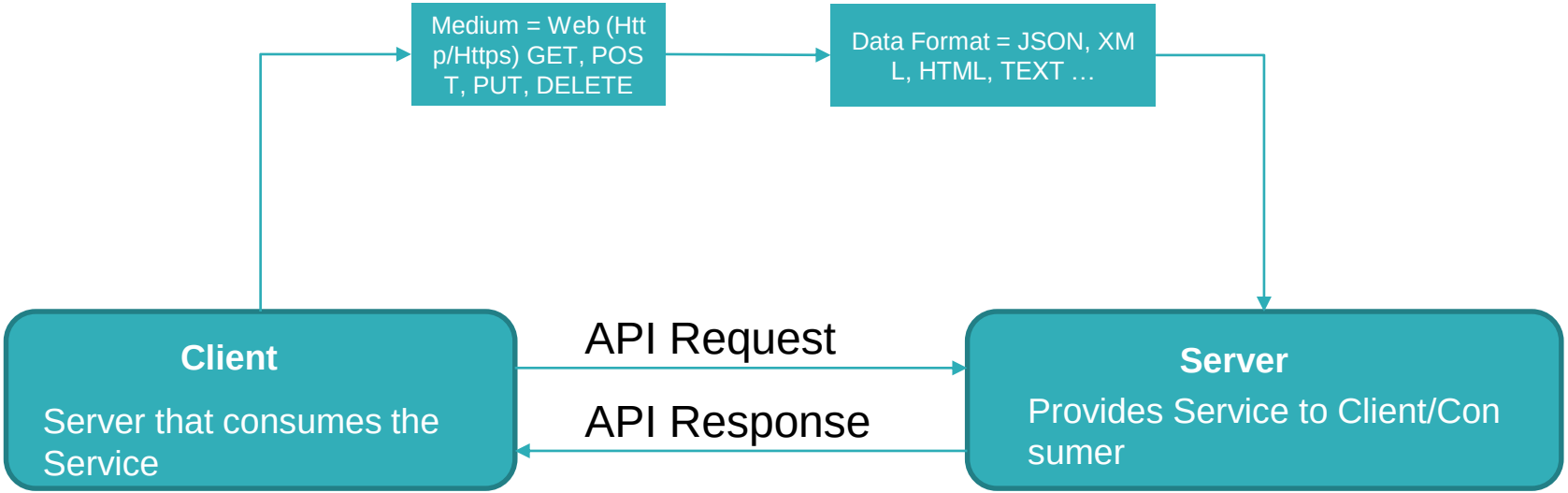


SOAP Web Services

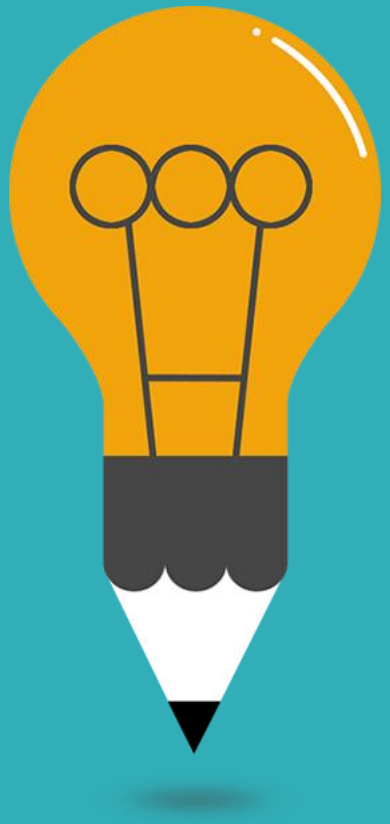


Simple **O**bject **A**ccess **P**rotocol

REST Web Services



Representational **S**tate **T**ransfer



REST API

REST API

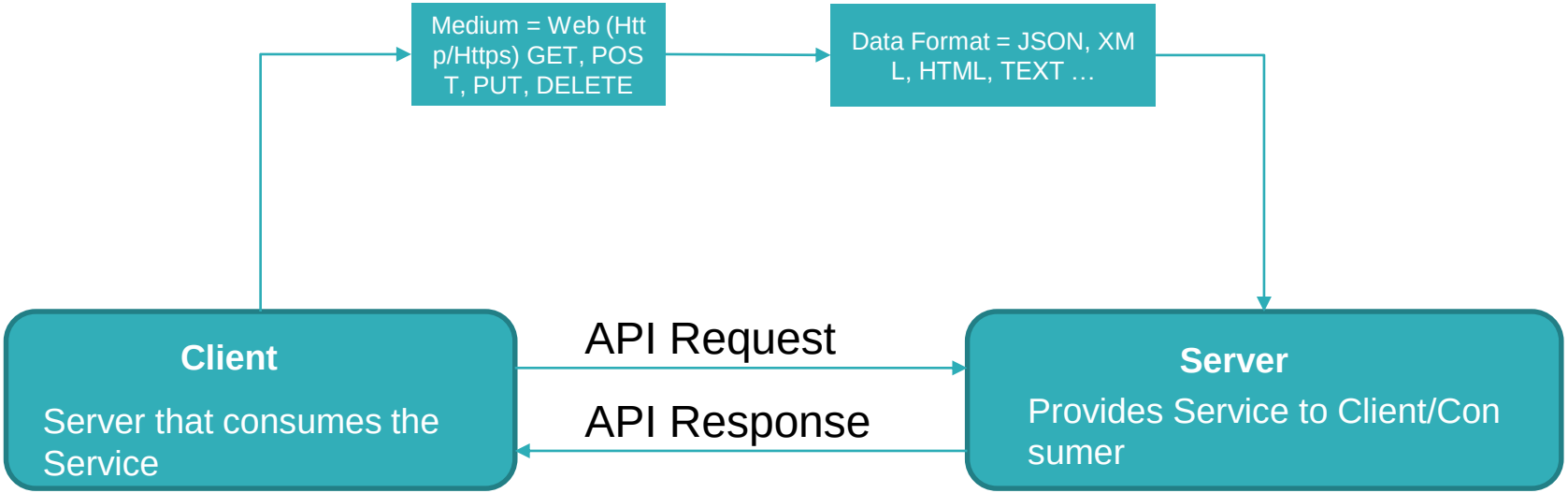
01 Explore REST API in Detail. Example of REST API ?

02 REST Architecture Style/Principles.

03 6 Guiding Principles of REST Architecture

04 What are the advantages of a Web Service.

Recap of REST Web Services



Representational **S**tate **T**ransfer

REST API

01

What is a REST API/Webservice?

It is a Webservice that allows exchange of data between 2 different Servers /Applications using the **REST Principles.**

02

What are all the REST Principles?

Client Server, Uniform Interface, Stateless, Cacheable, Layered System and Code on Demand (Optional).

03

What is Meaning of Representational, State and Transfer ?

The same resource data is to be retrieved or represented in different formats, depending upon the format requested.

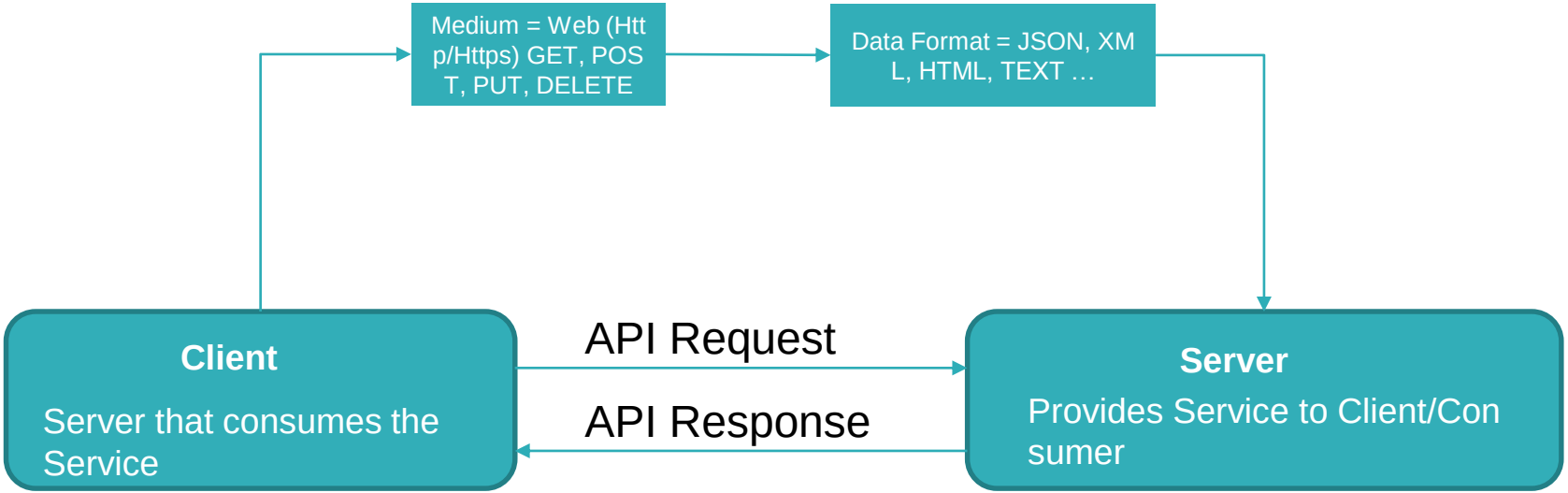
REST Principles

Client Server	It is a Webservice that allows exchange of data between 2 different Servers/Applications using the **REST Principles.**
Uniform Interface	The Uniform interface constraint defines the interface between clients and servers (Identification of Resources (URI), Manipulation of Resources, Self-Descriptive Messages).
Stateless	The Client Server works in Stateless Model**.** The Information related to the session is not persisted or remembered by the Service Provider.
Cacheable	The response data could be cached in the Client/Consumer Side.
Layered System	The Format of the data should not be broken if there are any components that exists between Client and Server (Ex: Proxy Server, API Gateway etc).
Code On Demand	Provides the ability for the client to download the code and execute directly on the client side.

REST Principle 1

Client Server

It is a Webservice that allows exchange of data between 2 different Servers/Applications using the **REST Principles.**



Representational **S**tate **T**ransfer

REST Principle 2

Uniform Interface

The Uniform interface constraint defines the interface between clients and servers (Identifi cation of Resources (URI), Manipulation of Resources, Self-Descriptive Messages).

HTTP

GET

POST

PUT

DELETE

URI

http://<Domain>/state
http://<Domain>/state/{name}
http://<Domain>/state/NY
http://<Domain>/state?Name=NY

http://<Domain>/Tariff
http://<Domain>/Tariff/{Farecategory_id}
http://<Domain>/Tariff/Low_rate

http://<Domain>/Tariff/Low_rate/State

State

Name

Tolls

Toll Name

Tariff

Fare Category ID

Fare Category Name

Other Modules

GET http://<Domain>/state — List All states
GET http://<Domain>/state/NY — Details of NY State
DELETE http://<Domain>/state/NY — Delete NY State
POST http://<Domain>/state — Add a new state called NJ
{ “State” : “NJ”}
PUT http://<Domain>/state/NY — Modify the description of NY
{ “State” : “NJ” , “Desc”:”New Jersey”}

REST Principle 2 …cont.

Uniform Interface

The Uniform interface constraint defines the interface between clients and servers (Identifi cation of Resources (URI), Manipulation of Resources, Self-Descriptive Messages).

- ✓ A **resource** in the system should have only **one logical URI** and that should provide a way to fetch related or additional data**.**
- ✓ Any single resource should not be too large and contain each and everything in its representation.
- ✓ The resource representations across system should follow certain guidelines such as naming conventions, link formats or dat a format (xml or/and json).
- ✓ All resources should be accessible through a common approach such as **HTTP GET** and similarly modified using a consistent approach (**POST, PUT, DELETE etc**).
- ✓ **Self Descriptive Messages** : A message should have enough information to let the server know how to process it (i.e. the typ e of request, mime types, etc.)

REST Principle 3,4,5,6.

Stateless

The Client Server works in Stateless Model**.** The Information related to the session is not persisted or remembered by the Service Provider.

- ✓ No need to store any data in the Server nor the state of the session.
- ✓ If the call fails, then the REST API will not be Auto Triggered.
- ✓ The Past activity of the consuming Session need not be remembered in the Server side.
- ✓ **Ex**: Activity carts Information, User Session ID (Authentication Code) etc.

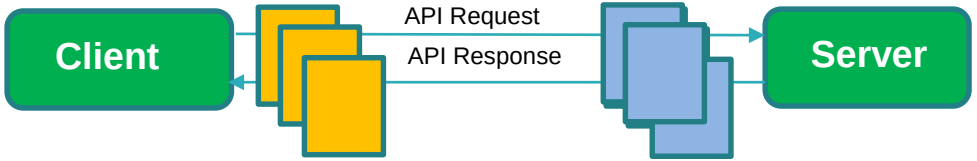
Cacheable

The response data could be cached in the Client/Consumer Side.

- ✓ The State of the session is managed and maintained in the Client side.
- ✓ The details of Caching is explained in next module.

Layered System

The Format of the data should not be broken if there are any components that exists between Client and Server (Ex: Proxy Server, API Gateway etc).



Code On Demand

Provides the ability for the client to download the code and execute directly on the client side. It is an **OPTIONAL** Principle.

Representational State Transfer

GET /State/NY
Accept: Json

GET /State/NY
Accept: XML

Output:

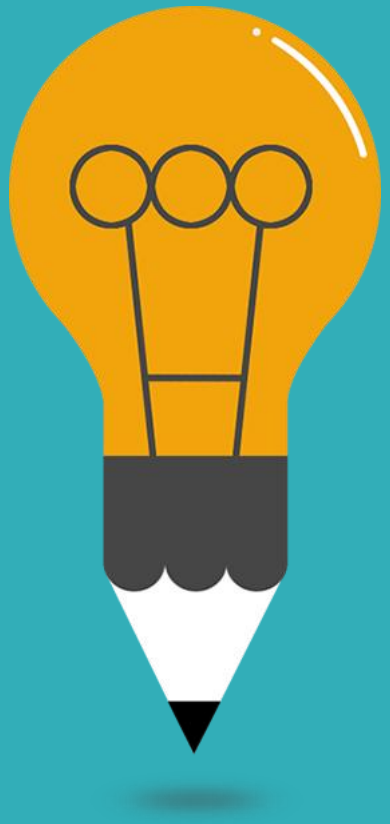
```
{
  “Statename”:”NY”,
  “Description”:”New York”,
  “Coast”:”East”,
  “Population”:”5M”
}
```

Output:

```
<State>
  <Statename>NY </Statename>
  <Description>New York <Description>
  <Coast>East </Coast>
  <Population> 5M </Population>
</State>
```

✓ Representation is the state/format of the resource.

✓ The Common Formats are JSON, XML, HTML, YAML, GHTML etc.



SOAP API

SOAP API

01 What is SOAP API?

02 Components and Rules of SOAP Webservices.

03 Example of a SOAP Webservice

SOAP Webservice

01

What is a SOAP API / Webservice?

It is a Webservice that follows the guidelines of the **SOAP webservice Specifications**. It is called as **S**imple **O**bject **A**ccess **P**rotocol.

02

What are all the SOAP Webservice Specifications?

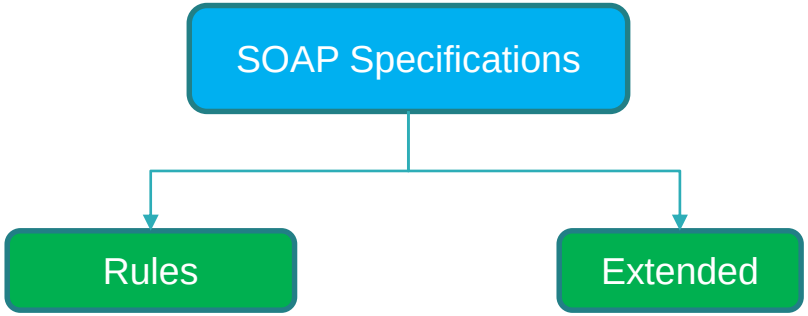
SOAP Rules, **WSDL** (Web Service Definition Language) and **UDDI** (Universal Description, Discovery and Integration) .

03

Who is responsible for dictating the SOAP Web Service Specification?

World web consortium (https://www.w3.org/TR/soap/). It is a community that develops and standardize the communication over the web.

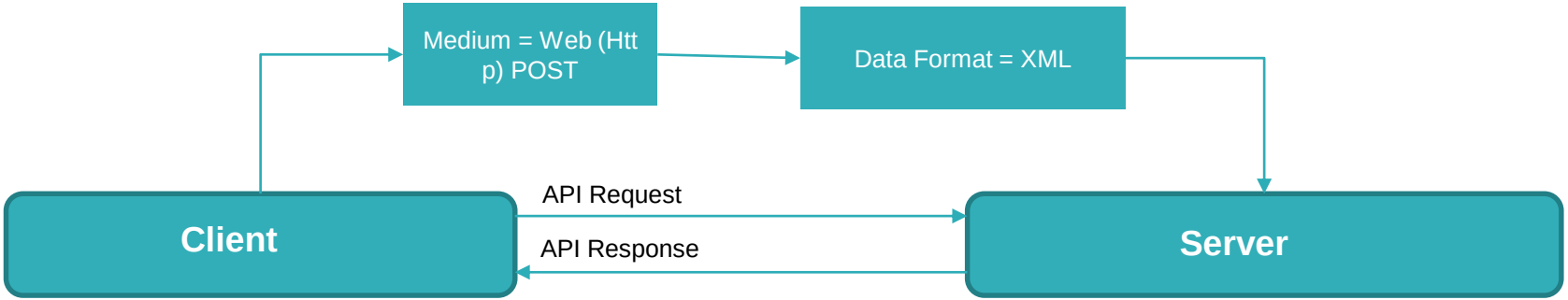
SOAP Specifications



- ✓ SOAP Rules
- ✓ WSDL
- ✓ UDDI

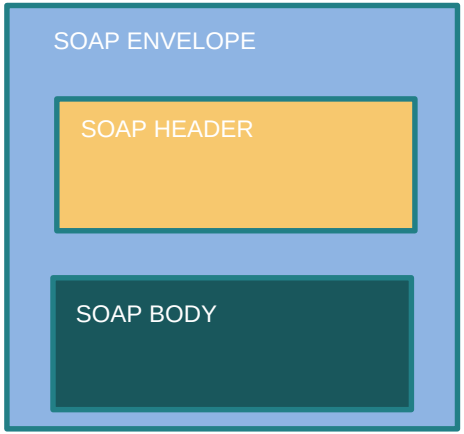
- ✓ WS-Addressing
- ✓ WS-Policy
- ✓ WS-Security
- ✓ WS-Federation
- ✓ WS-ReliableMessaging
- ✓ WS-Coordination
- ✓ WS-AtomicTransaction
- ✓ WS-RemotePortlets
- ✓ ….. Others

SOAP Web Services



Simple Object Access Protocol

- Communication between Client and Server happens in XML Format.
- The Format of the XML file should adhere to the SOAP standards.
- The SOAP XML contains 3 blocks namely **SOAP Envelope**, **SOAP header** and **SOAP Body**.



SOAP Message Example

```
<?xml version='1.0' Encoding='UTF-8' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
                env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2007-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
                env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
   <n:name>Fred Bloggs</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2007-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2007-12-20</p:departureDate>
     <p:departureTime>mid-morning</p:departureTime>
     <p:seatPreference></p:seatPreference>
   </p:return>
  </p:itinerary>
 </env:Body>
</env:Envelope>
```

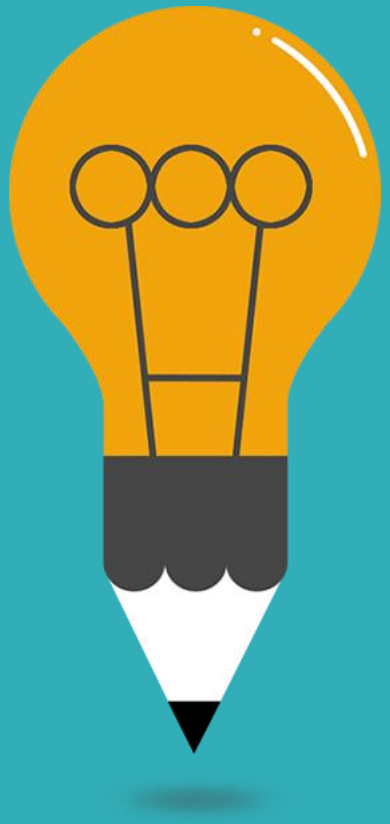
SOAP Message Example

```
POST /Quotation HTTP/1.0
Host: www.xyz.org
Content-Type: text/xml; charset = utf-8
Content-Length: nnn

<?xml version = "1.0"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
   SOAP-ENV:encodingStyle = "http://www.w3.org/2001/12/soap-encoding">

   <SOAP-ENV:Body xmlns:m = "http://www.xyz.org/quotations">
      <m:GetQuotation>
         <m:QuotationsName>MiscroSoft</m:QuotationsName>
      </m:GetQuotation>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Request

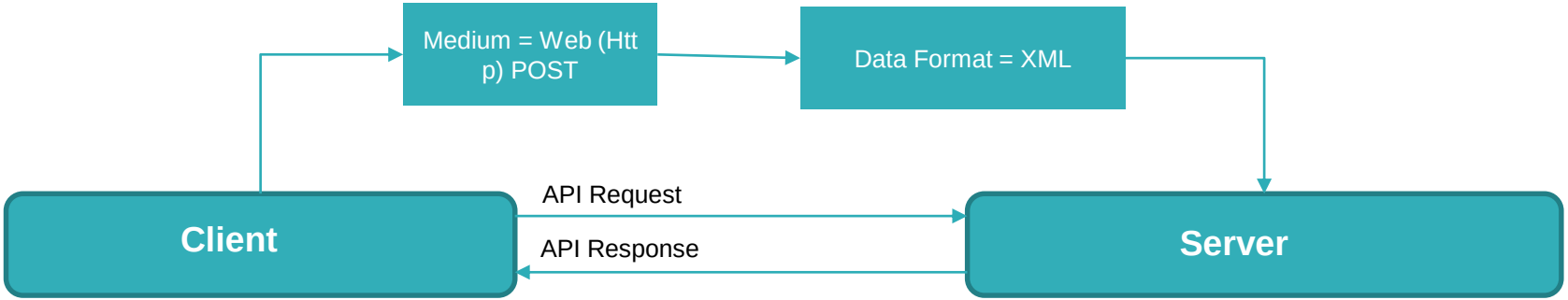
Response

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset = utf-8
Content-Length: nnn

<?xml version = "1.0"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
   SOAP-ENV:encodingStyle = "http://www.w3.org/2001/12/soap-encoding">

   <SOAP-ENV:Body xmlns:m = "http://www.xyz.org/quotation">
      <m:GetQuotationResponse>
         <m:Quotation>Here is the quotation</m:Quotation>
      </m:GetQuotationResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```



WSDL and UDDI

WSDL and UDDI



- WSDL stands for **W**eb **S**ervices **D**escription **L**anguage.
- WSDL is an XML notation for describing a web service.
- A WSDL definition tells a client how to compose a web service request and describes the interface that is provided by the web service provider.
- It includes the End Point, Method name, parameters required, data type of parameter, PORT Information, Binding Information etc.
- Machine Readable. Tools present to read the WSDL file and convert into more Human readable format.

WSDL Live example

Sample WSDL File

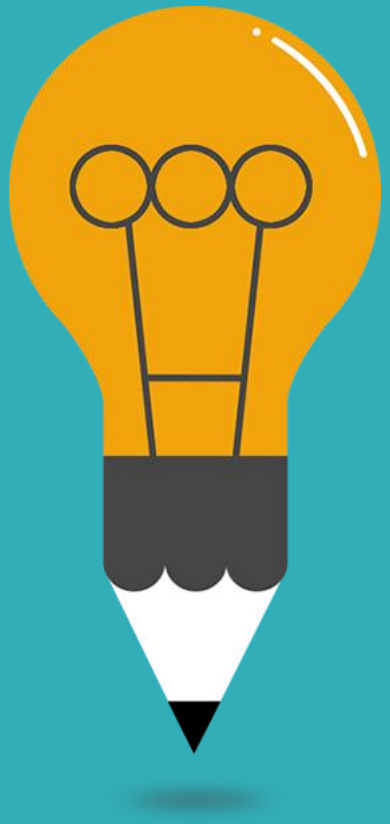
- https://svn.apache.org/repos/asf/airavata/sandbox/xbaya-web/test/Calculator.wsdl

WSDL Parser

- https://tomi.vanek.sk/

UDDI

- **UDDI** stands for Universal Description, Discovery, and Integration.
- **UDDI** is an XML-based standard for describing, publishing, and finding web services.
- It is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet.
- It is a Dictionary where the users would come and publish the Soap Webservices.



Difference between REST and SOAP API

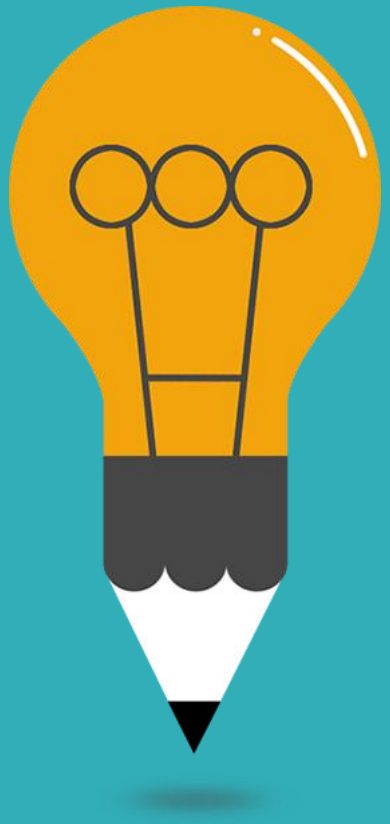
REST Vs SOAP

REST

- ✓ Format used in JSON, XML, HTML, etc.
- ✓ REST is Architecture Style.
- ✓ Supports all HTTP requests like GET, POST, DELETE, PUT etc.
- ✓ Less Data is transferred across network as the data is stored in JSON format.
- ✓ Representational State Transfer
- ✓ Data Driven Approach (Use Nouns: Ex: Dept)
- ✓ Stateless.
- ✓ API Calls can be cached.
- ✓ Only HTTP Protocol.
- ✓ Better Performance.
- ✓ Built in Error handling do not exist

SOAP

- ✓ Format used in XML Only.
- ✓ SOAP is Standardized Format/Protocol.
- ✓ Supports Only HTTP POST.
- ✓ More Data is transferred across network as the data is stored in XML format.
- ✓ Simple Object Access Protocol.
- ✓ Function Driven approach (Use Verbs: Ex: GetDepts)
- ✓ Both Stateful and Stateless.
- ✓ API Calls cannot be cached.
- ✓ HTTP, SMTP, UDP and Others.
- ✓ High Security, Standardization and hence used in Payment gateways and Financial Services.
- ✓ Built in Error handling and Error codes are assigned.



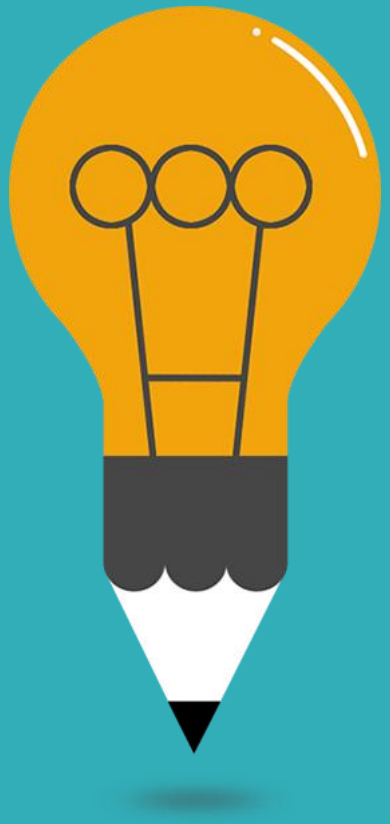
What existed before REST and SOAP

Pre REST/SOAP Era

- ✓ DCOM (**Distributed Component Object Model**). Developed by Microsoft.
- ✓ CORBA (**Common Object Request Broker Architecture**)
- ✓ RMI (**Remote Method Invocation**)

Disadvantages

- ✓ **RMI**: Strictly Java. Cannot use with other code outside Java.
- ✓ Difficulty in using DCOM/CORBA to work over Internet firewalls and on unknown and insecure machines



Basics of HTTP

HTTP Protocol

- Hyper Text Transfer Protocol.
- Latest Version is HTTP 2.0, Most used Version is HTTP 1.1
- TCP is the Underlying Protocol and is the foundation of WWW
- Default PORT that it listens to is 80.
- It works in REQUEST and RESPONSE Mode

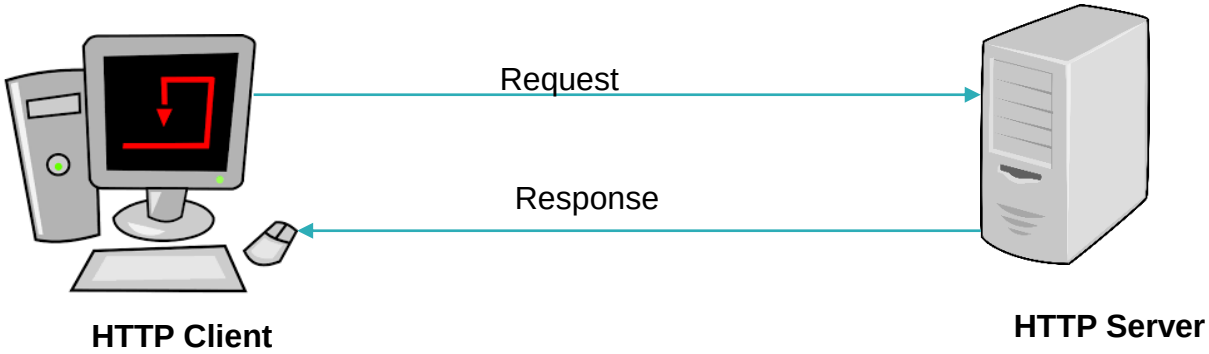
HTTP Overview

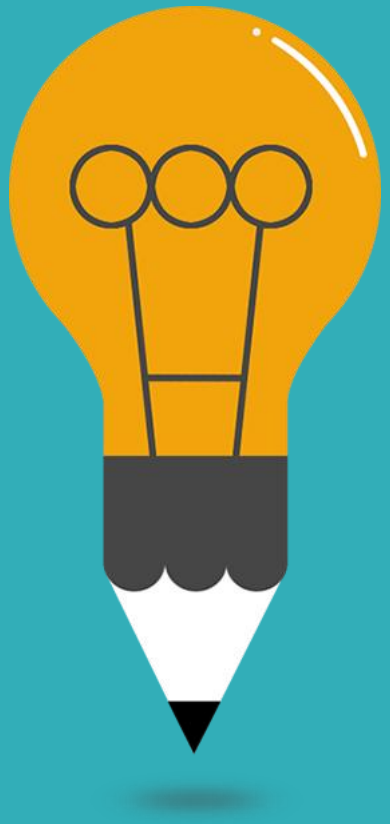
Client sends the requests in the HTTP standards. All the information required by the HTTP Server should be sent by the HTTP client in the request.

This request should include all the data elements like what resource is being requested and should adhere to the HTTP Standards.

Server processes the HTTP requests and then sends back the results as a HTTP response. It also includes return code to tell the client if the response was successful or not.

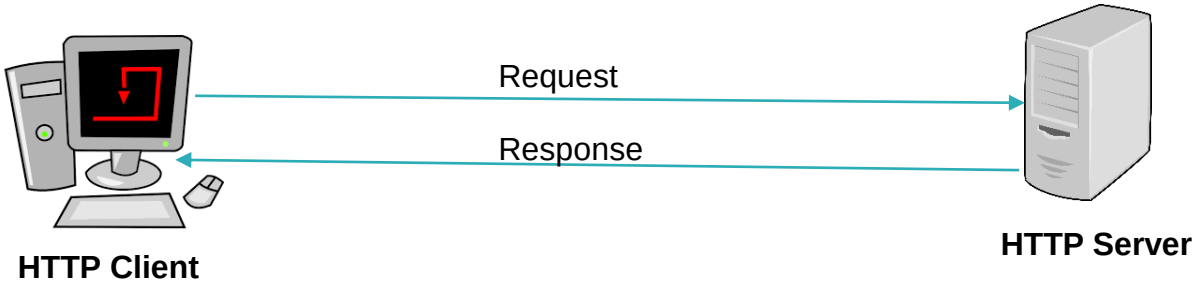
This response should include all the data elem ents that could be understood by the client to c onsume it.

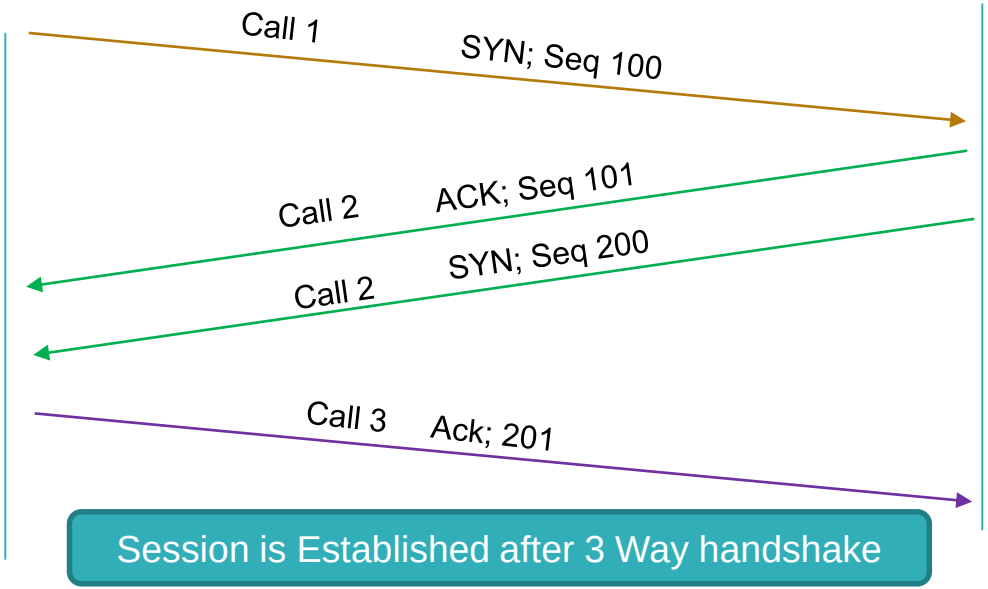




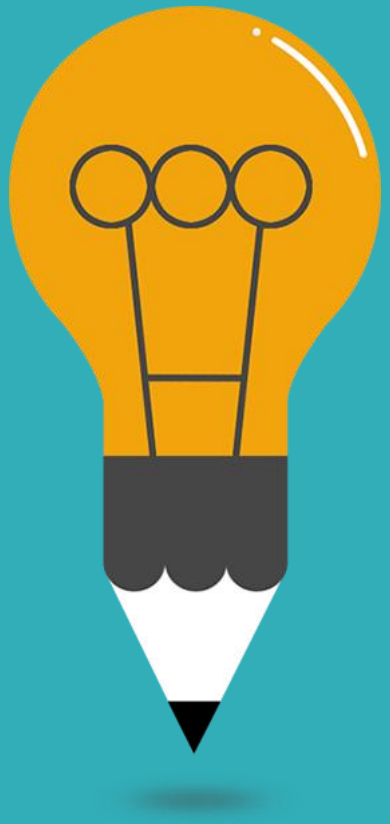
3 Way TCP Handshake

3 Way Handshake (TCP)



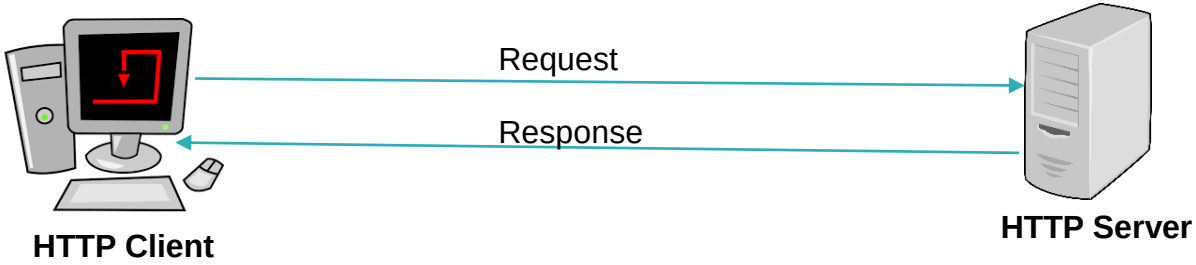


Session is Established after 3 Way handshake



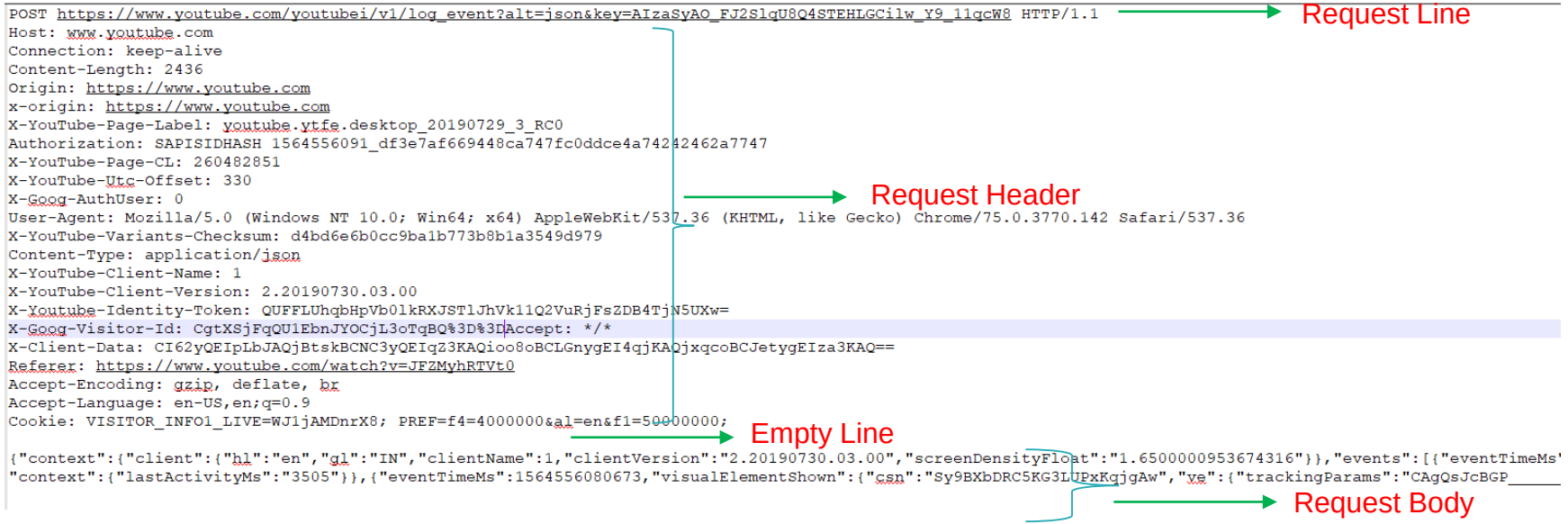
Request Header / Body

Request Message Format

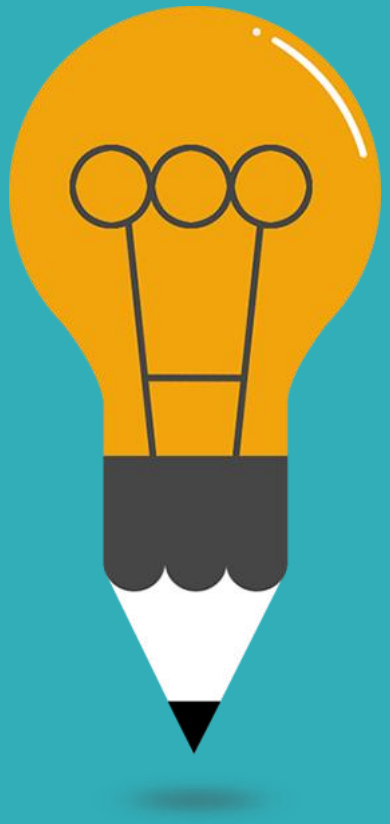


1. Request Line
2. Request Header
3. Empty Line
4. Request Message Body

Sample Request

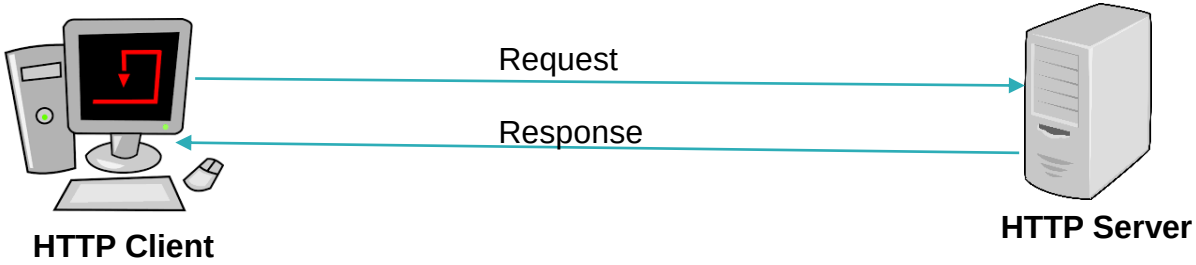


Let us now Look at sample Request in Fiddler



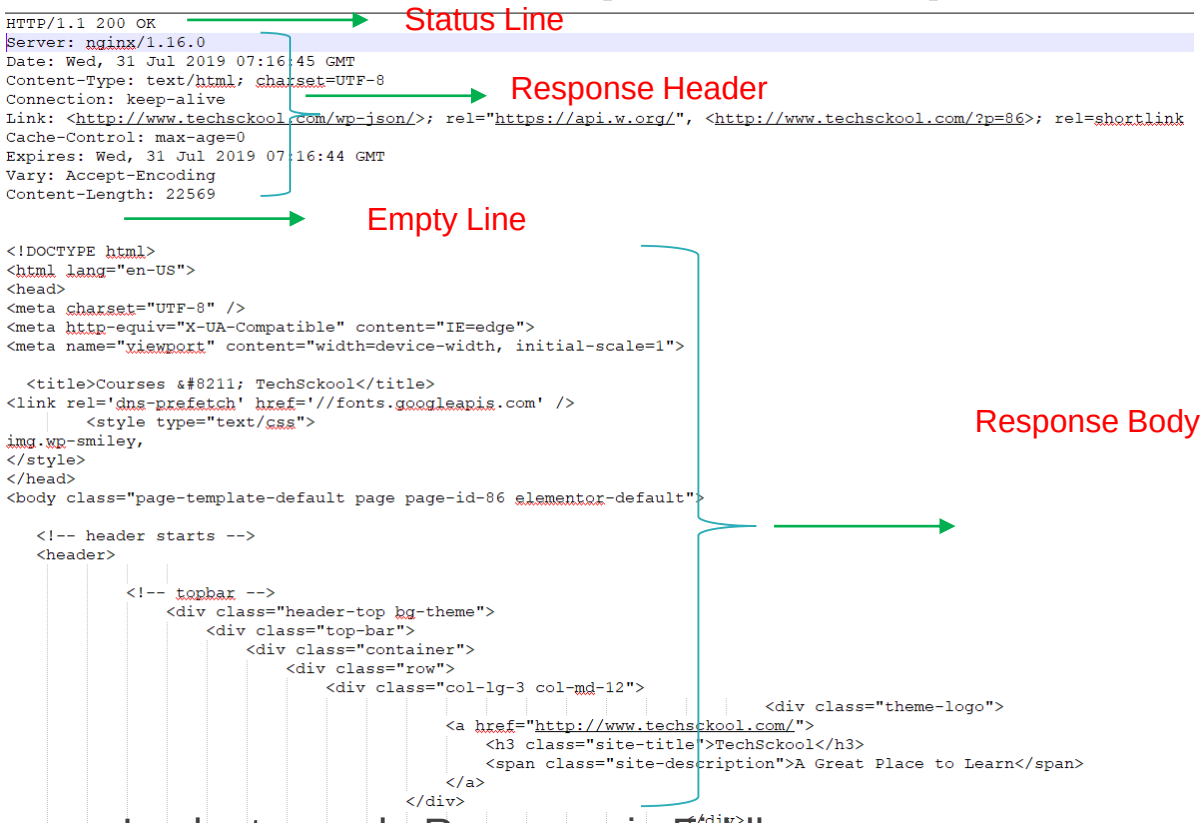
Response
Header / Body

Response Message Format

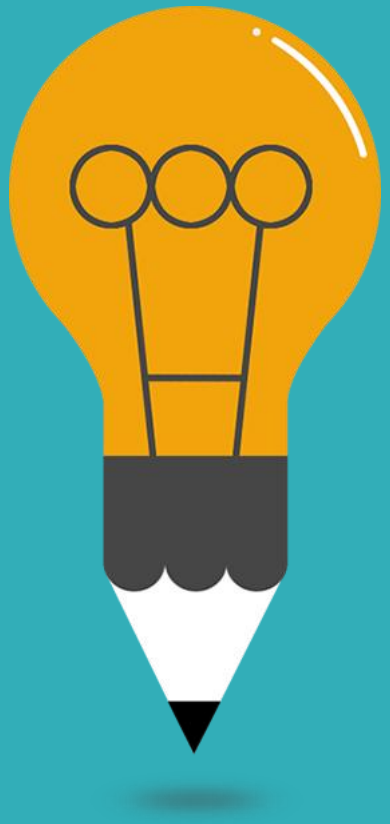


1. Status Line
2. Response Header
3. Empty Line
4. Response Message Body

Sample Response



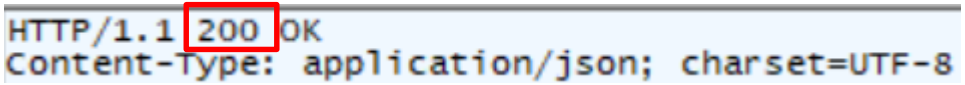
Let us now Look at sample Response in Fiddler



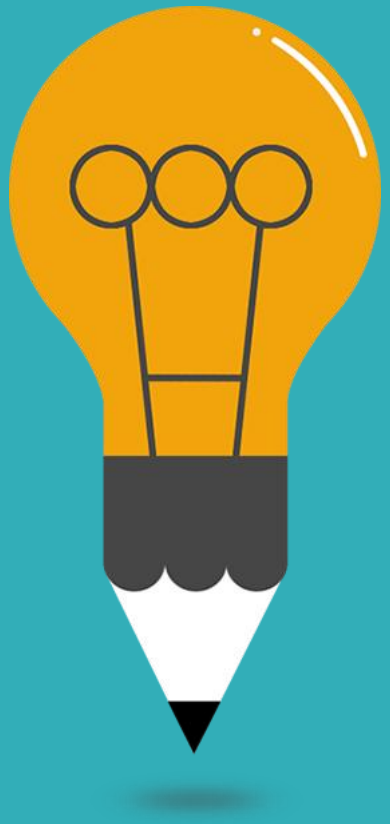
HTTP Status Code

HTTP Status Code

- Status codes are issued by a server in response to a client's request made to the server. This would tell if the request is successful or not.

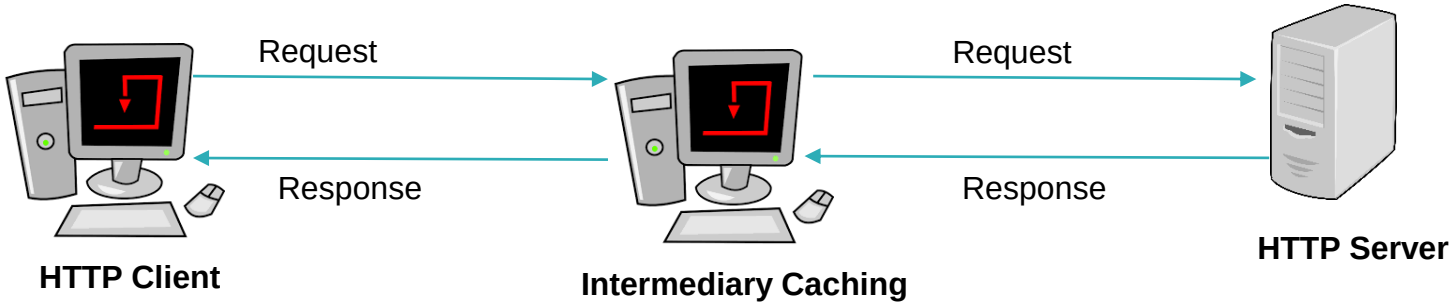


Status Code	Meaning
1XX	Informational Message (Ex: 102)
2XX	Success (Ex: 200)
3XX	Redirection (Ex: 301, 304)
4XX	Client Error (Ex: 404)
5XX	Server Error (Ex: 500, 502)



HTTP Cache

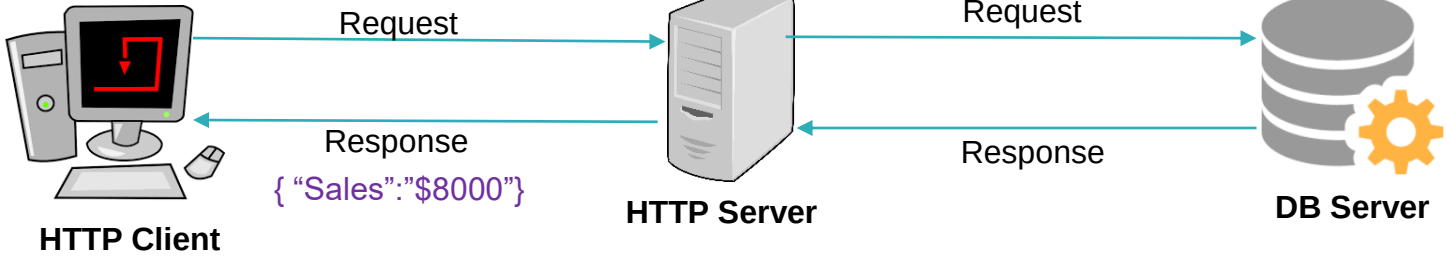
HTTP Cache



1. Caching helps to reduce the response time of the request.
2. Caching helps to reduce the Network Bandwidth Utilization.
3. Caching could be done at:
    - Client
    - Intermediary Server
    - Web Server

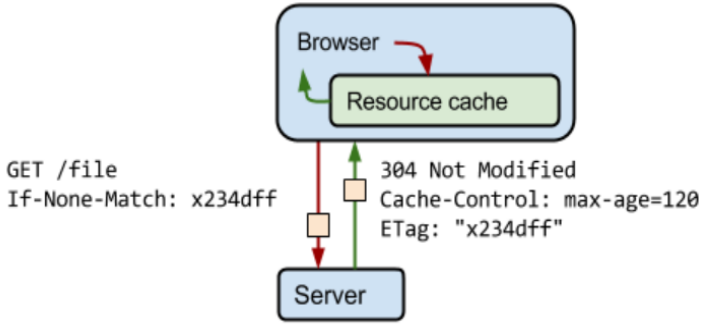
HTTP Cache

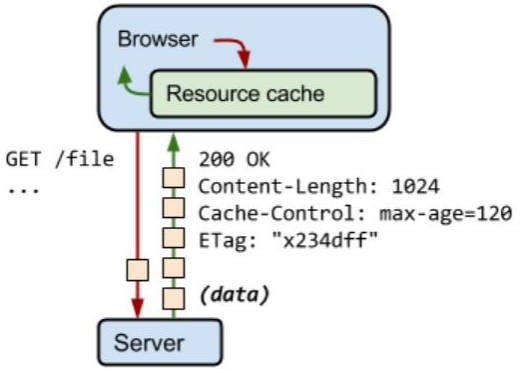
http:/Domain/SalesData



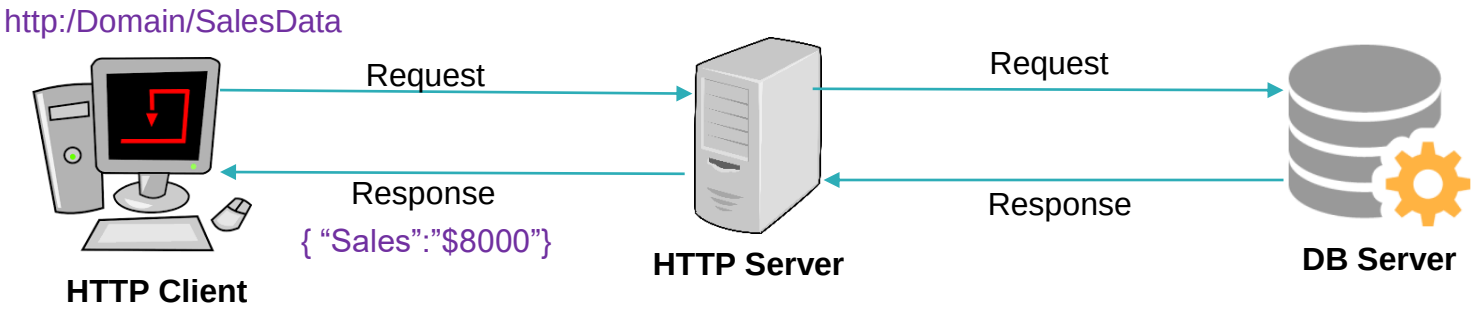
Browser CACHED

{ “Sales”:”$8000”}





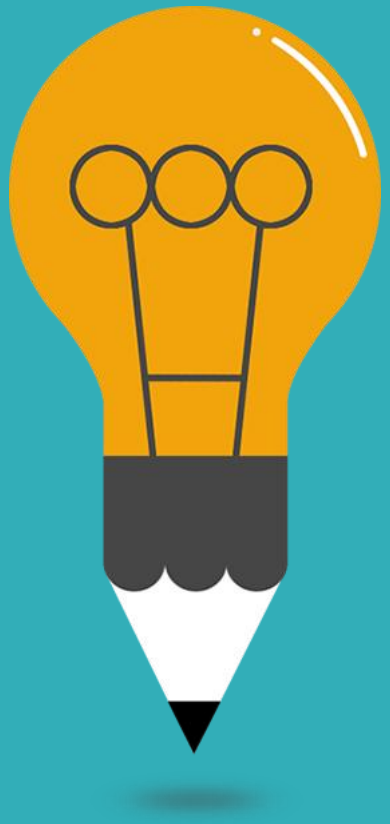
HTTP Cache Control Header Options



Browser CACHED

{ “Sales”:”$8000”}

Cache-Control	Meaning
no-cache	First check with Server and then read from cache.
no-store	Completely ignore CACHE.
max-age	Time in seconds to read from Cache
public	It can be cached even if it has HTTP Authentication associated with it.
private	Cacheable only in Client Browser and not in Interme diary server(CDN)



JSON Fundamentals

JSON Fundamentals

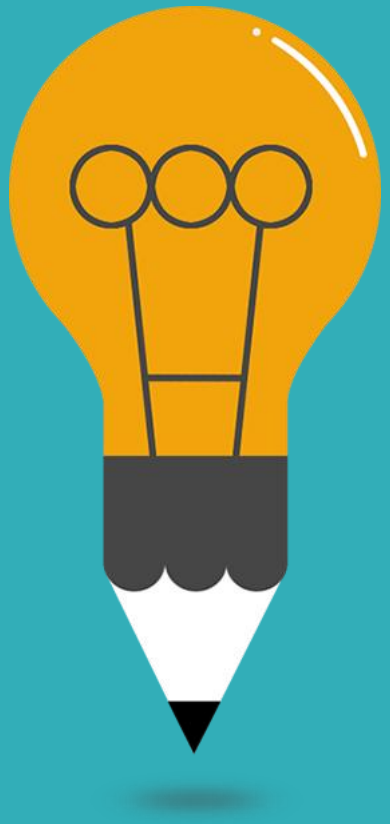
- **JSON** stands for JavaScript Object Notation
- It is human-readable data.
- It stores data in **Key/value** pair.
- It is the most popular data format used in **Web/API.**
- It is Language Independent.
- The Key/value pairs are separated by “,”
- Square brackets hold arrays and values are separated by “,”

JSON File Example

School	St Joseph High School	
Class	9	
City	Bangalore	
ID	**Name**	**Age**
1	Sohan	16
2	Abhay	15
3	John	16
4	Nancy	15

https://jsonformatter.org/

```
{ "School" : "St Joseph High School",
  "Class":"9",
  "City":"Bangalore",
  "Students":[  { "ID":"1",
                  "Name":"Sohan",
                  "Age":16
                },
                { "ID":"2",
                  "Name":"Abhay",
                  "Age":15
                },
                { "ID":"3",
                  "Name":"John",
                  "Age":16
                },
               { "ID":"4",
                  "Name":"Nancy",
                  "Age":15
                }
             ]
}
```



XML Fundamentals

XML Fundamentals

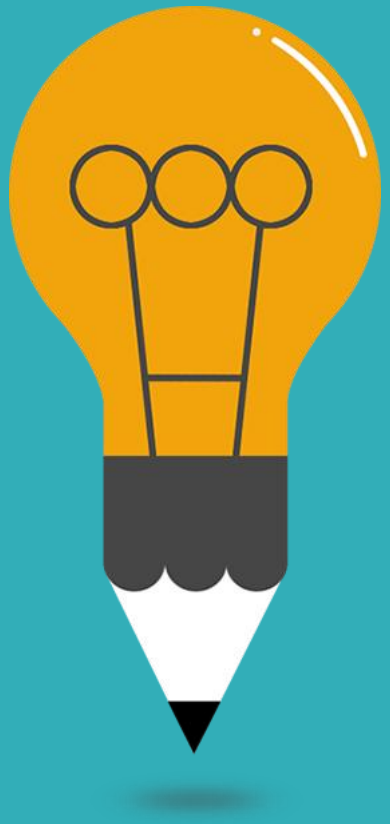
- **XML** stands for **E**xtensible **M**arkup **L**anguage
- XML was developed by an organization called the World Wide Web Consortium (W3C)
- XML is the only format used by the SOAP API's.
- Any type of data can be expressed as an XML document.
- It is Language Independent.
- It is both Human Readable and Machine Readable.
- It uses markup symbols like <data> </data> , <password> </password>.

XML File Example

School	St Joseph High School	
Class	9	
City	Bangalore	
ID	**Name**	**Age**
1	Sohan	16
2	Abhay	15
3	John	16
4	Nancy	15

http://xmlbeautifier.com/

```
<?xml version = "1.0"?>
<School-info>
    <School>St Joseph High School</School>
    <Class>9</Class>
    <City>Bangalore</City>
    <Students>
        <Student>
            <ID> 1 </ID>
            <Name> Sohan </Name>
            <Age> 16  </Age>
        </Student>
        <Student>
            <ID> 2 </ID>
            <Name> Abhay </Name>
            <Age> 15  </Age>
        </Student>
        <Student>
            <ID> 3 </ID>
            <Name> John </Name>
            <Age> 16  </Age>
        </Student>
        <Student>
            <ID> 4 </ID>
            <Name> Nancy </Name>
            <Age> 15  </Age>
        </Student>
    </Students>
</School-info>
```



Types of HTTP Methods

HTTP Methods

- There are 4 main types of HTTP methods.