Web Application Processing – Server, Client

Web Application

1. A piece of software which can be accessed from a browser.
2. Multiple users, same version of App.
3. Doesn't Need to be installed.
4. For a web app to operate it needs – web server, Database

Web Server

(serves web content) Physical Device with Network Connection can host multiple Apps

1. Listens
2. On a Port
3. For a Request
4. Transport Protocol (HTTP, HTTPS)
5. Response
6. Resource – HTML, JSON, PDF, Image, Document, etc.

web client receives response from web server.

Browser, Mobile Application

Something that can talk to a server.

Client $\xleftrightarrow{\text{Protocol (HTTP)}}$ Server

Web Server — A network Application listening to some port.

Protocol — set of agreed upon rules between client and server.

URL (Uniform Resource Allocator)

https://www.amazon.com/orders/123

protocol | addr. of web server | addr. of resource

54.192.0.0/16

Default Port Addr: 80 (HTTP), 443 (HTTPS)

HTTP Requests

GET
POST
PUT
DELETE

GET /orders/123
Host: 54.192.0.0:16
User-Agent: (Client Data) (Browser, Operating Sys Info)
Connection: keep-Alive
|Body X

Responses

HTTP/1.1 200 OK
Date:
Server:
Content-length:
" -Type:
Connection: closed

1XX Informational
2XX Success
3XX Redirect
4XX Client Error
5XX Server "

Static Vs Dynamic

HTTP

i) Stateless
ii) Connectionless
iii) Application layer protocol
iv) TCP/IP - based
v) Media Independent

MIME Type

Multipurpose Internet Mail Extension

text/plain
html
csv

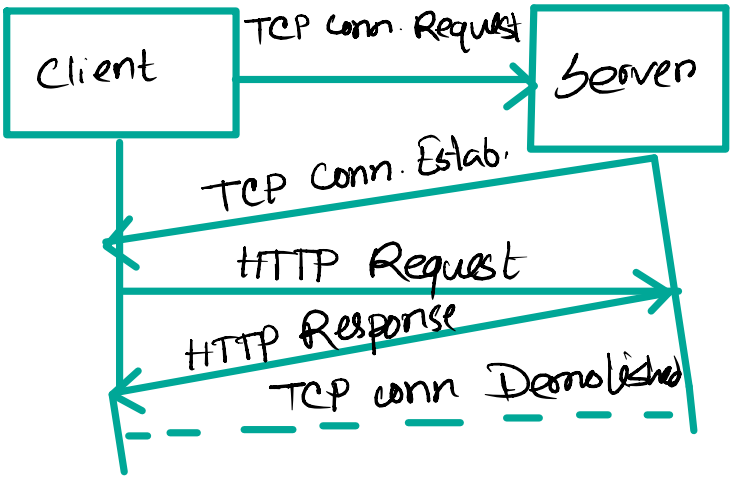
images/jpeg
/png

video/mp4

application/pdf
/zip

32-bits

Source Port	Destination Port	→ 16 bits
TCP/IP Packet format		



HTTP methods

- GET
- HEAD
- POST
- PUT
- DELETE
- CONNECT
- OPTIONS
- TRACE
- PATCH

HTTP 1.0 / 1.1 — single / multiple Request, Response.

Status Codes

1xx	101 Protocol Switch
2xx	200 OK
3xx	Request needs redirection
4xx	401 Unauthorized
5xx	500 Server failed

URL Fragments

https://www.swe4537.com/home?theme=dark#projects

URL scheme — host — Resource Path — Query String — Fragments

What is Javascript?

- HTML is a markup language that we use to structure and give meaning to our web content e.g., defining paragraphs, headings, and multimedia contents, etc.
- CSS is a language of style rules that we use to apply styling to our HTML contents eg., font size, background colors, etc.
- JS is a scripting language that can update and manipulate both HTML and CSS, which, in turn enables you to create dynamically updating content, control multimedia, animate images, and so on.

Scripting languages do not need compiling step.

- Server: JS, Python, PHP
- Client: JS, JQuery, AJAX

Why is it called JavaScript?

- To make websites live by utilizing client side processing power.
- Livescript, then JavaScript
- JS doesn't make websites dynamic. ✗
  Responsive ✓

JavaScript Engine

- software component that executes JS code.
- Intially, Interpreters.
- Built by modern web browser vendors.
- modern engines use Just in-time compilers.

comments // /* */
variables var, let
constants universal, local
Naming conventions
Data Types
Type Conversion

semi-colon, prettier

Number — float, integer, Infinity, Neg. Infinity, NaN
Big Integer
String
Boolean
Null
Undefined
Symbol
Object

primitive – single Val.
non-primitive – multiple

✓Infinity (+ve /0)
✓Neg Infinity (-ve/0)
✓Not a Number (data types where mathematical operations aren't possible)

1/ " 1" = 1 (spaces before and after it is possible)
1/ " 1 2 3" = NaN
string in a '+' operation results in a string.
'5' + 2 = 52
2 + '5' = 25

✓Null – has a value – null
✓Undefined – is not assigned a value.
✓BigInt (Number)
12345n

JSON (JavaScript Object Notation)

{ key: value, key.value }
must be string

Typecast

DataType (data)
String (123)
x.toString()

Operators

Unary ++ , -- , + , -
Comparison > , < , >= , <= , !=
Logical && , || , !
Binary + , - , / , % , **
Bit-Wise & , | , ^ XOR , ~ , << LS , >> RS
Equality == , ===
Assignment

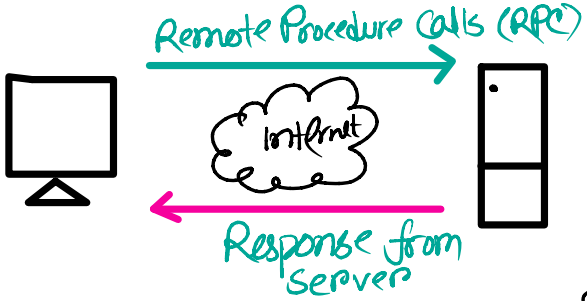
If-else
JS functions

Web Services

A web service is a software system designed to support interoperable machine-to-machine interaction over a network. Other systems interact with the web systems in a manner prescribed by its description using SOAP messages conveyed using HTTP with an XML Serialization.

If an application can be accessed over a network using a combination of protocols like HTTP, XML, SMTP, etc. then it is a web service.

It acts as an abstraction layer separating the user of the code and the programming language specific details of how the application code is actually invoked.



The client would invoke a series of web service calls which would host the actual web service.

RPC A web service can have multiple methods to provide different functionality. These methods are called via RPC.

Data is transferred between the client and server as XML (Extensive Mark-up language, Easy to Understand Understood by many PLs)

Web services use something known as SOAP (Simple Object Access Protocol) for sending XML data between Applications. Data is sent over normal HTTP.

Data is called SOAP message.

Two types of web services:

1. SOAP web services
2. Restful Web services
   Representational State Transfer

SOAP

platform & OS independent messaging protocol.

- XML Document
- Structure follows a specific pattern, not the content.
- Best Part is sent via HTTP (content-type: text/XML
- light-weight data interchange protocol

Root Element identifies XML as SOAP

```
<Envelope>
    <Header> </Header> [Routing data, client info]
    <Body> </Body> [content]
</Envelope>
```

Authentication

So, for message passing web services use SOAP over HTTP.

WSDL (Web Services Description Language) keeps the functionality of a web service.

- XML Document (Not SOAP)
- 3 parts
  - → Message - Data Elements for Each Operation
  - → PortType (Operation, Input-Output)
  - → Bindings - Protocol (transport, encoding)
    - transport → SOAP/http
    - encoding → SOAP

UDDI - Universal Description, Discovery & Integration (UDDI) stores the list of web services. All clients have access to this.

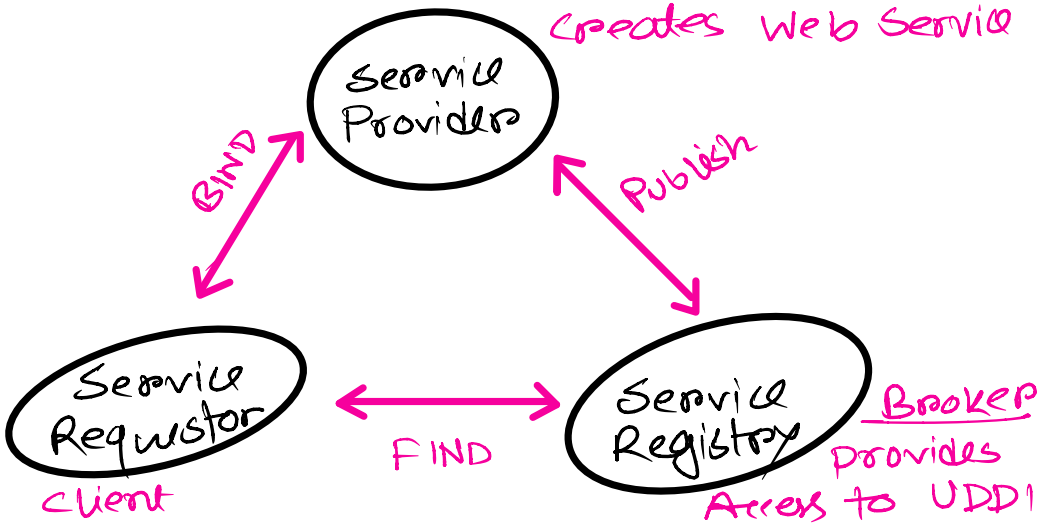
Advantages of Web Services

1. Exposing business functionality on the network.
2. Interoperability among applications.
3. Standardized protocol stack that everybody understands.
4. Reduction in cost of communication

previously used - EDI B2B
Electronic Data Interchange for Business 2 Business.

Protocol Stack

1. Service Transport (Responsible for message passing)
   HTTP, SMTP, FTP, BEEP
   Box Extensible Exchange Protocol.
2. XML Messaging (Encode messages to a common format)
3. Service Description (Describes the public Interface WSDL)
4. Service Discovery (UDDI) - for centralizing services into a common registry



Characteristics

OS, network, Platform

XML Based – Eliminates Dependency

Loosely Coupled – If client or server changes, doesn't affect the communication

Synchronous / Asynchronous

Ability to support RPC

Supports Document Exchange → Simple to Complex
'String' 'Book'

Restful Web Services

Everything is a resource. You can access, manipulate.

RPC
SOAP X

URI	HTTP VERB	DESCRIPTION
api/users/1	GET	Gets first user.

Doesn't Require XML, WSDL, SOAP.
URL is enough. Not structured/protocol-based.

Service Exposure

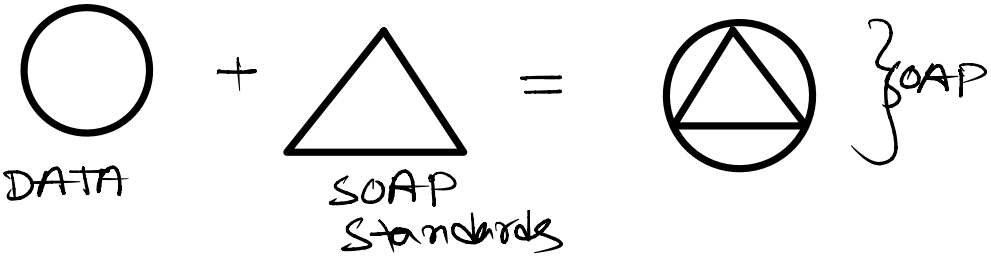
Uses URI to locate resources. Doesn't require a seperate document to transport messages.
HTTP is enough.
less Bandwidth.

Message format

XML
JSON
Any Other.

Conventions

Noun - Endpoint/Resources
Request using HTTP action verb
plural form of resources
HTTP Response code



Differences

1.	SOAP follows strict guidelines, since Protocol.	
2.	" uses XML.	Uses XML, JSON, any.
3.	RPC call,	URL is enough
4.	Difficult to Implement	
5.	More Bandwidth	
6.	Easier to perform transactions.	

Application Programming Interface and Web Service

API is a software intermediary that allows two application to talk to each other.

According to World Wide Web Consortium (W3C), web services provide standard means of operating between different applications over a network running on variety of platforms and/or frameworks.

Internet/organizational

All web services are APIs, but not all APIs are not web services.

Example: Iphone OCR is an API, not web service.

Criteria	API	Web Service
Need of Network	Not Always	Always
Lightweight Arch	✓	X
Mode of Communication	Any	SOAP, Rest, XML-RPC
Protocol Restriction	WebAPI – HTTP	HTTP
Hosting	Devial Internet	Internet

Model View Controller

Transition from web technologies to Architectural Pattern

View - Responsible for visual elements
Model - Database/storage
Controller - set of methods

N-Tier Architecture / Multi-Tier

A program that is distributed among N platforms over a distributed network.

Refers to number of Tiers.

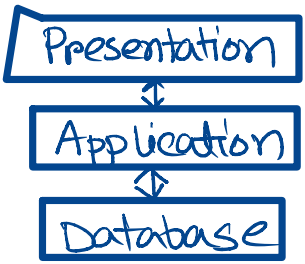
Most common - 3 Tier.

i) User interface - Client
ii) Business Logic - Centralized Platform
iii) Required Data - Database Server

Provides

Scalability	Effort
Security	Security
Fault Tolerance	
Reusability	
Maintainability	
Availability	

A multi-tier architecture can be sub-divided into 3 parts.



Terminologies

Distributed Network

Components located at different network computers co-ordinate and communicate with each other via message passing. A collection of nodes situated at different locations, but appear to the user as a single system.

Platform

A system where programs can run.

1-Tier
2-Tier | Client Server Architecture
3-Tier

Software Architectural Pattern

1. Layered/N-tier/Multitier

Basic layers – Presentation, Business, Database

All the components are interconnected but they do not depend on each other. Each layer performs a single and specific role.

2. Basic Client-Server Architecture (2 tier)

Based on processing load on clients, it can be of two types.

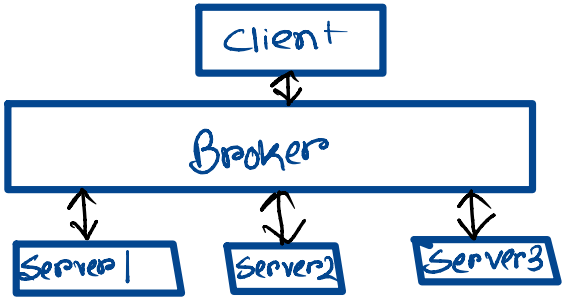
1. Thick client | Video Editing App, Gaming
2. Thin client | Streaming App, Ecommerce

3. Master-Slave Architecture

Master (R/W or R)
Slave (Read)
| Database Servers

Read is faster.

4. Broker

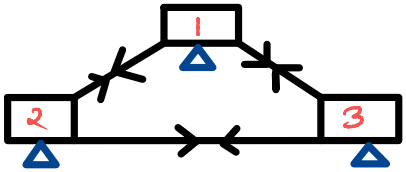


A broker component co-ordinates request and responses between clients and servers.

Broker has the details of the servers and the services they provide.

5. Peer-to-Peer

- Doesn't distinguish between client or server.
- Each node can act like a client or server depending on whether the node is requesting or providing service. No central server is required.



BitTorrent
BitCoin
File Sharing

Proxy

Forward Proxy (Anonymity of client)

- Block malicious data from reaching server
- Block access to certain sites (by organization / subscription)
- Cache responses

Reverse Proxy (Anonymity of server)

- sits on the server side & acts as a middleman
- load balancing
- maintaing traffics
- cache
- X DDOS attack.

Session based authentication

1. What is authentication? Who you are?
2. What is authorization?

You can be authenticated using username and password, but that doesn't authorize you to enter into the website.

Authentication can be done based on 3 factors.

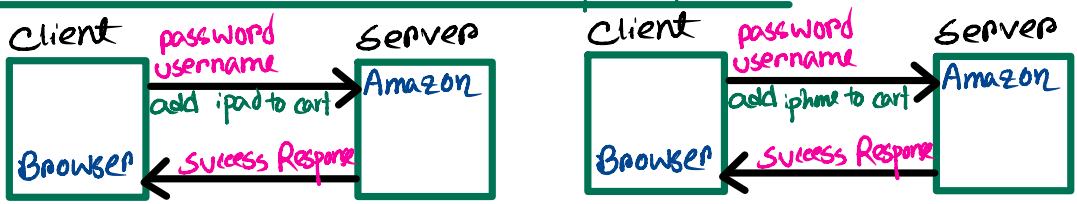
- something that you know
- " " " have
- " " " $\boxed{\text{are}}$ Biometric

single factor
two-factor | combination of any of these two
multi-factor

ATM Money Withdrawal - two factor

something that you have - ATM Card
" " " know - password

Authentication from server's perspective



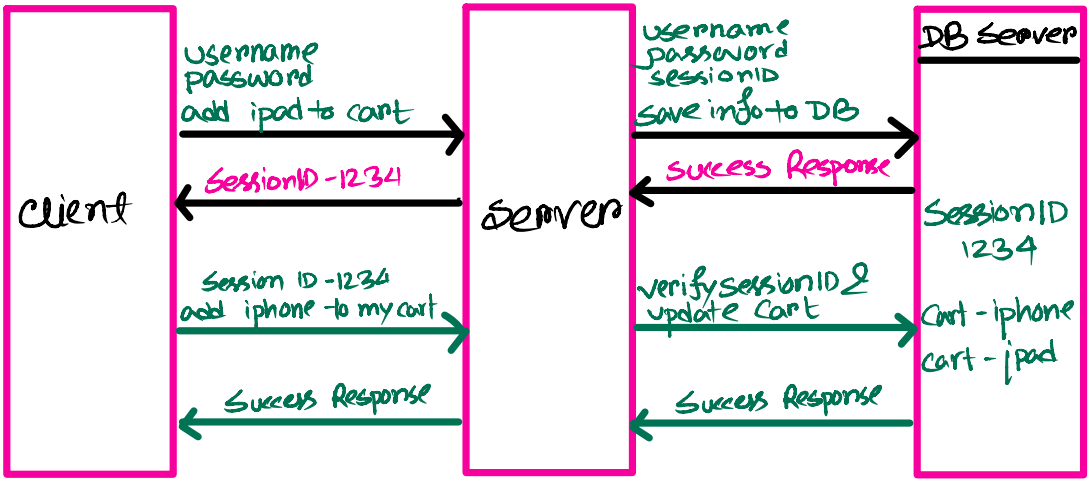
Since, HTTP is stateless for both of these requests, user'll have to send username and password multiple times.
Sending sensitive information over a shared line multiple times is not safe.

Solution? | Encrypted username, pass
Cryptography basics
AES - DES, Safety. $2^{128}, 256$

Solution: Session-based Authentication

Definition is application specific. (from login until logout)

Usually, closing a tab/browser ends a session, but refreshing the page does not.



Issues with infinite session ID

Cross-site Request forgery

You have to transfer sessionID with every request and it is stored in Browser storage.

	Local Storage	Session Storage	Cookie
Storage Limit	5MB	5 - 10MB	4KB
Access	Client + Server	Client + Server	✓ Server - Client Not Always
Validity	Lifetime	Browser/Tab Close	Automatically gets added with HTTP Header Key-Value Pair

HTTP only

Secure flag

XSS Attacks

- HTTPS/SSL

Cookies can be of three types.

- Session cookie
- Permanent cookie
- Third-party cookie | so that other websites can show you refined suggestions, e.g., browsing habits.

Limitations of Session-Based Authentication

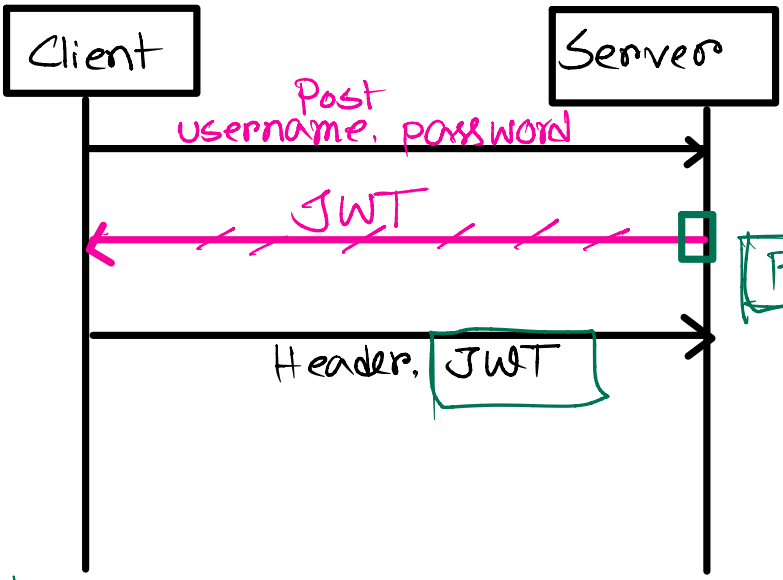
1. Distributed Systems | The requested has to be validated using same server.
2. Performance Issue | Database store/retrieval after every request
3. Cookie fraud

Token-based Authentication

Problems with session-based
- Distributed Systems
- Session compromise
- Database Look-up

sessionID = value

JWT (JSON web Token) | Json



public key
private key
Hand shaking protocol

private key

Advantages
1. Can be generated anywhere
2. Authorization
3. Eavesdropping X More secure

JWT

- Header
  - Token type – "JWT"
  - Algorithm – "RSA"
- Payload
- Signature (Integrity)
  - symmetric – 1 key (client, server)
    - HMAC
  - Asymmetric – public key | source verification
    - private | Data Integrity
    - RSA, ECDSA

Header
Payload
encrypt

Payload

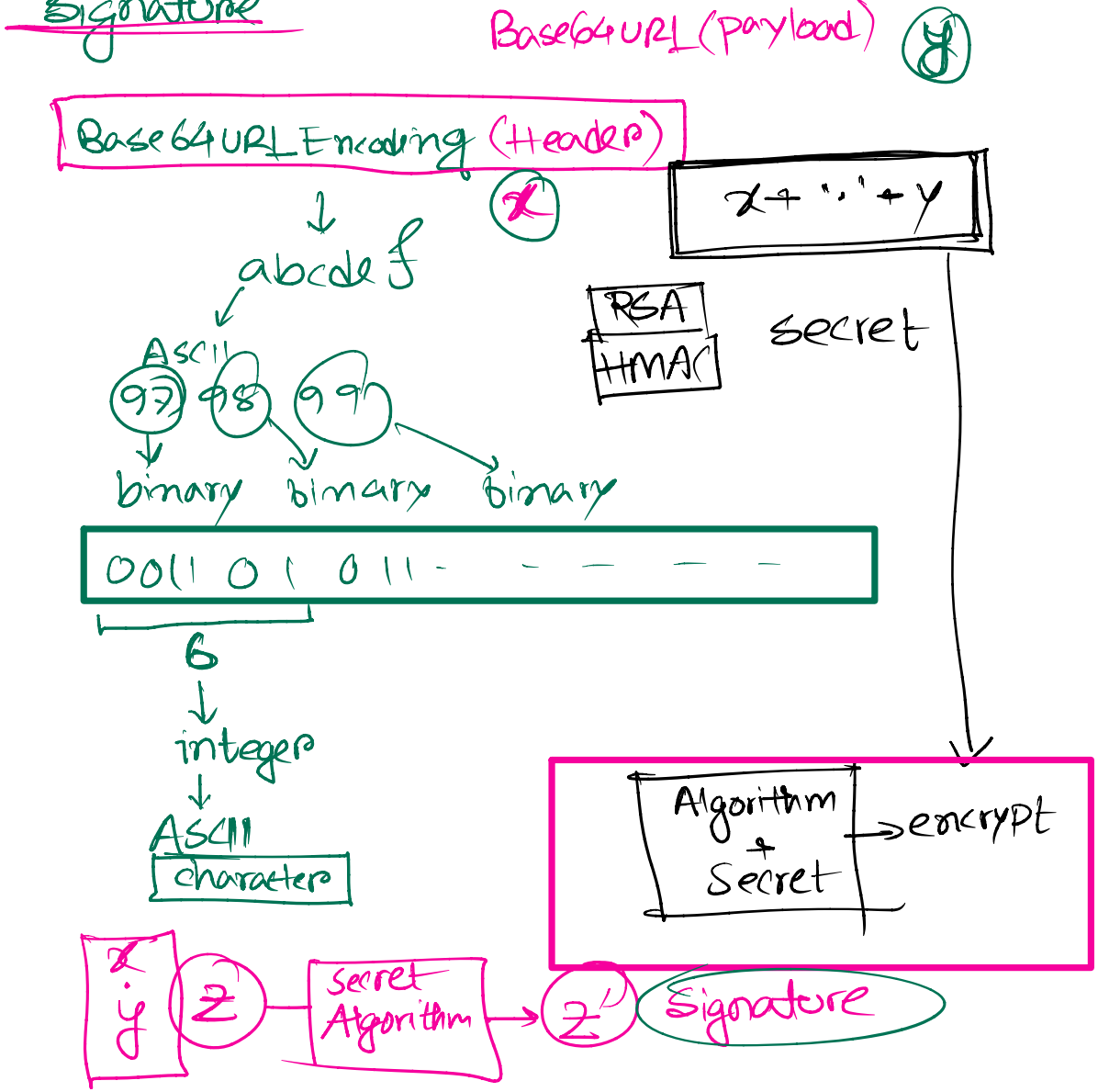
Basic/Default

- issuer
- subject (authentication, authorization, etc.)
- audience
- expiration time
- Not before
- issue at
- JWT id

- username
- role – admin

unencrypted

Signature


Base64URL(payload)
y
Base64URL Encoding (Header)
x
x + '.' + y
abcdef
ASCII
97
98
99
RSA
HMAC
secret
binary
binary
binary
0011 0110 011- - - - -
6
integer
ASCII
characters
Algorithm + Secret
encrypt
x.y
z
Secret Algorithm
z'
Signature

JWT

Header
Payload
Signature

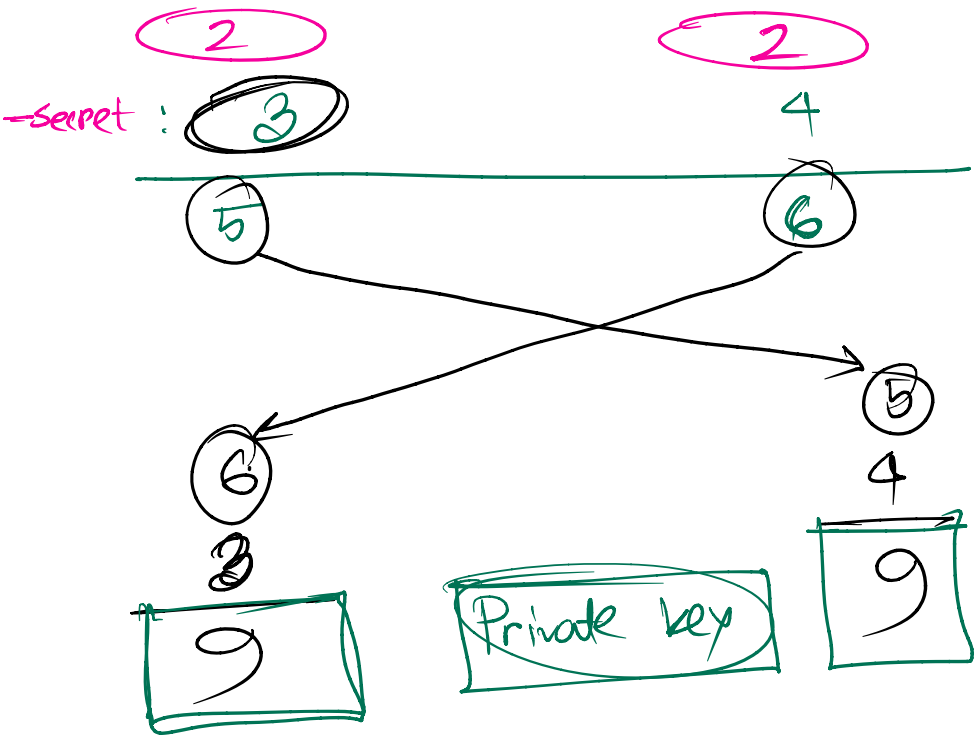
Cookie

Local Storage

Disadvantages

1. Takes more space / Increased Bandwidth

Diffie Hellman Key Exchange



Advantage

No DB lookup
Saving on DB space
More secure
Easier, flexible
Automatically prevents CSRF

Disadvantage

Non-Revocable
Bottlenecked against one secret key.
Data visible to everyone

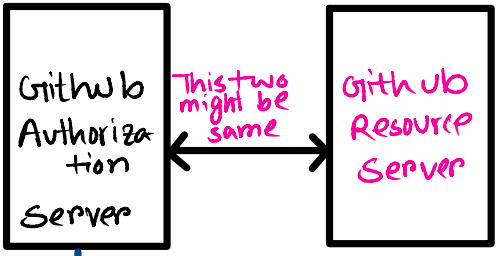
Open Authorization 2.0

four Important Entities.

i) Client
ii) Resource Owner
iii) Authorization Server
iv) Resource Server

Resource owner – The applicant
Redirect/ Callback URL – iut.com
Response Type – Authorization Code | Short lived key
Authorization Server → Client
Scope – Get repository details
Client – iut

Resource owner



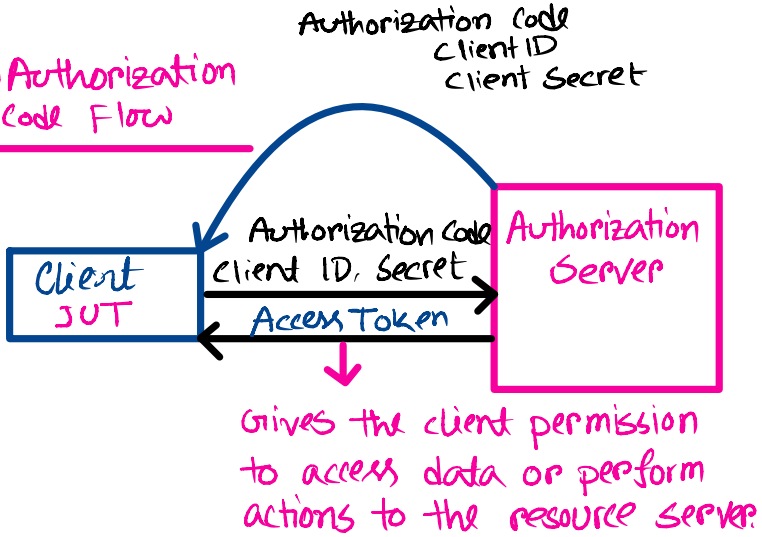
might be a 3rd party that resource server trusts

Consent

Do you want iut.com to access your public & private repository details?

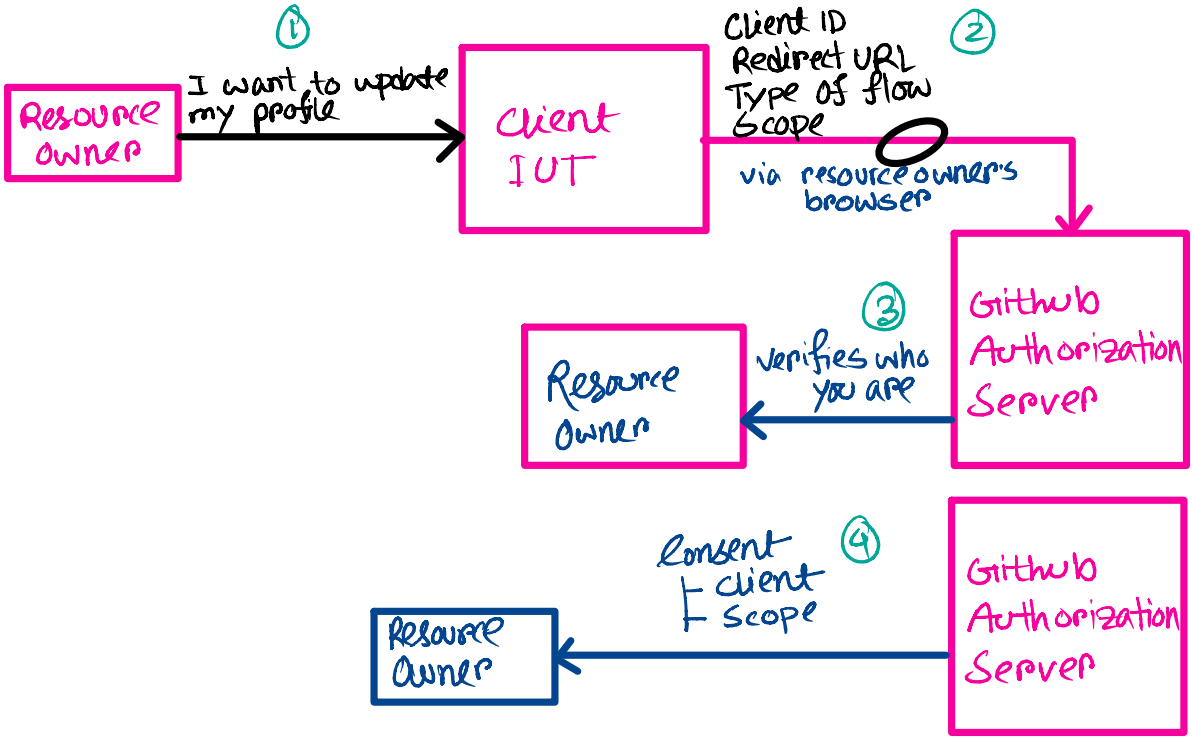
client defines what information about the resource owner to take from resource server. Before providing, resource server shows consent.

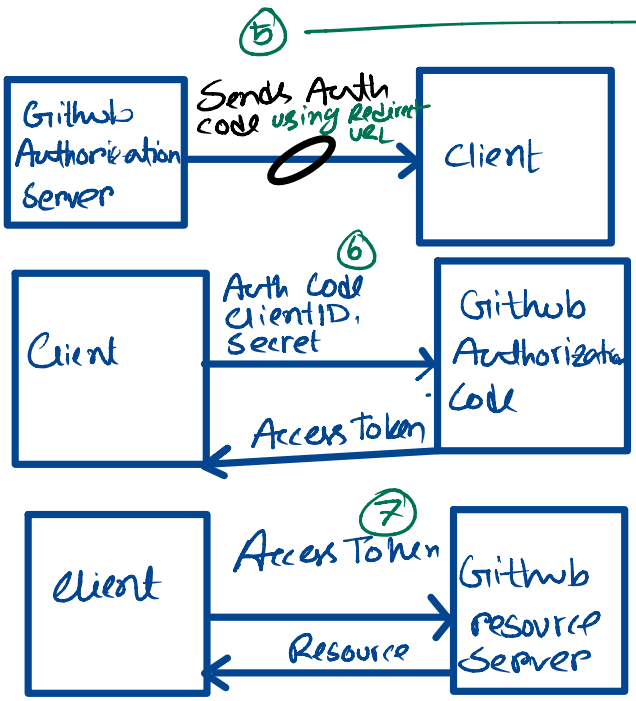
① Authorization Code Flow



OAuth flows

1. Authorization Code Flow
2. Implicit Flow
3. Resource Owner Password Credentials Flow
4. Client Credentials Flow





In implicit flow, it will send access code in this step to client.

In resource owner credential code flow, client will send resource owner's username & password in step 2.

In client credentials flow, client accesses resource owner's public data by directly communicating with the resource server.

Difference between OAuth 1.0 & 2.0

1. Terminology (User > Client, Service provider > Resource Server)
2. No backward compatibility
3. No encryption or signature checking in OAuth 2.0. But requires SSL.

Push Notifications

1. can be triggerred locally by an open application.
2. can be pushed from server to client while app is not running (push notifications on the web)

Push Notifications requires the help of two APIs.

1. Notification API
   displays system notifications to the App.

2. Push API
   allows a service worker to handle push messages from a server, even when app is not running.

   - listens for upcoming push notifications from server.
   - requires very low energy.

Important Terminologies

Notification – A message displayed as pop up to the user.

Push Message – A message sent from server to client.

Push Notification – A notification created in response to a push message.

Push Service – Responsible for routing push notifications from service to client. } different for each browser.

Notification Types

- Alert user to an important event.
- Display an icon and a small piece of text that user can click to interact.
- Integrate buttons, sounds, vibrations, etc..

Workflow

Server —push message→ push Service → client

↓

Service Worker

listens in the background.

- displays Noti.
- handle inter.
- sleep

Notification API

I) Invocation API
II) Interaction API

controls how to make your notifications appear.
controls what happens when you interact with it.

Steps

1. Request Permission
2. Display Notification
3. Listen for events

PUSH API

push service
user grants permission
subscribe app to browsers push service

sends an object
to ~~server~~ client
- endpoint
- public key

Only works on HTTPS

server $\xrightarrow{\text{Secure}}$ push service $\xrightarrow{\text{Secure}}$ client