Process

By

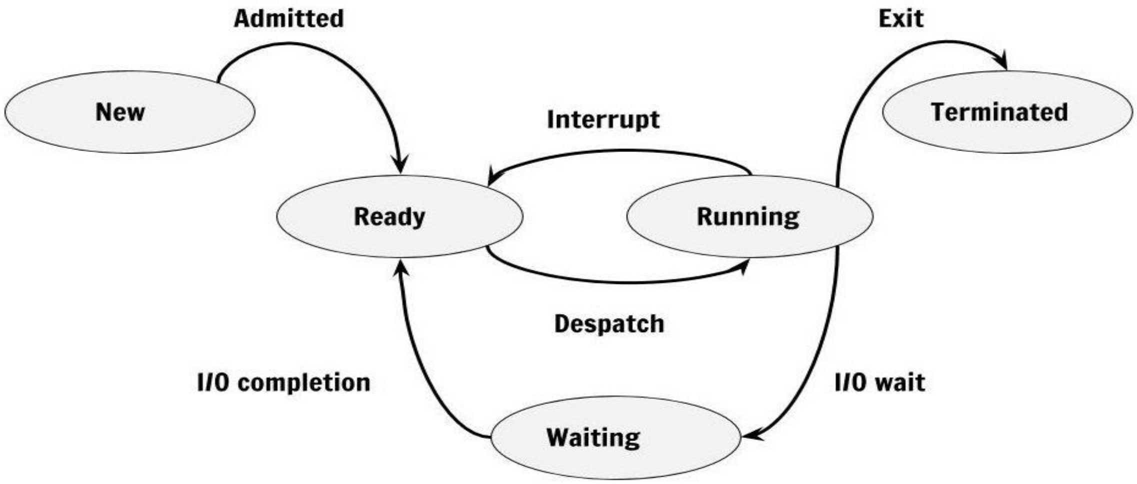
G. M. Al-Arafat. Shaown.

Process:

A process is an active program i.e. a program that is under execution. It contains the program code, program counter, process stack, registers etc.

Process States:

The different states that a process is in during its execution are explained using the following diagram –



New- The process is in new state when it has just been created.

Ready - The process is waiting to be assigned the processor by the short term scheduler.

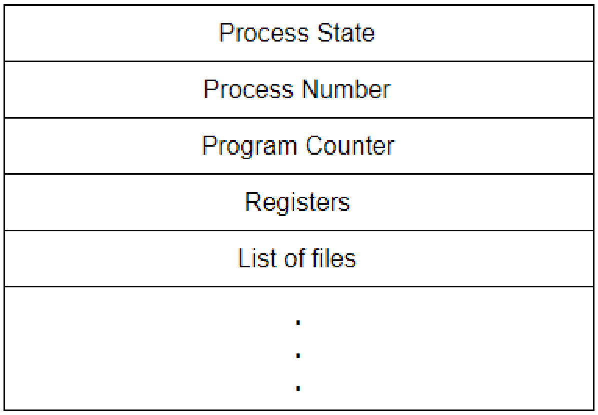
Running - The process instructions are being executed by the processor.

Waiting - The process is waiting for some event such as I/O to occur.

Terminated - The process has completed its execution.

Process Control Block:

A process control block is associated with each of the processes. It contains important details about that particular process. These are as follows –



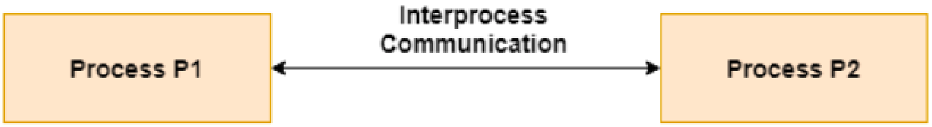
Process State
Process Number
Program Counter
Registers
List of files
. . .

- **Process State** - This specifies the process state i.e. new, ready, running, waiting or terminated.
- **Process Number** - This shows the number of the particular process.
- **Program Counter** - This contains the address of the next instruction that needs to be executed in the process.
- **Registers** - This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.
- **List of files** - These are the different files that are associated with the process.

Inter-process communication:

Inter-process communication is the mechanism provided by the operating system that allows processes to communicate with each other. This communication could involve a process letting another process know that some event has occurred or the transferring of data from one process to another.

A diagram that illustrates inter-process communication is as follows –



Synchronization in Inter-process Communication:

Synchronization is a necessary part of inter-process communication. It is either provided by the inter-process control mechanism or handled by the communicating processes. Some of the methods to provide synchronization are as follows −

- **Semaphore**
  A semaphore is a variable that controls the access to a common resource by multiple processes. The two types of semaphores are binary semaphores and counting semaphores.
- **Mutual Exclusion**
  Mutual exclusion requires that only one process thread can enter the critical section at a time. This is useful for synchronization and also prevents race conditions.
- **Barrier**
  A barrier does not allow individual processes to proceed until all the processes reach it. Many parallel languages and collective routines impose barriers.
- **Spinlock**
  This is a type of lock. The processes trying to acquire this lock wait in a loop while checking if the lock is available or not. This is known as busy waiting because the process is not doing any useful operation even though it is active.

Approaches to Inter-process Communication:

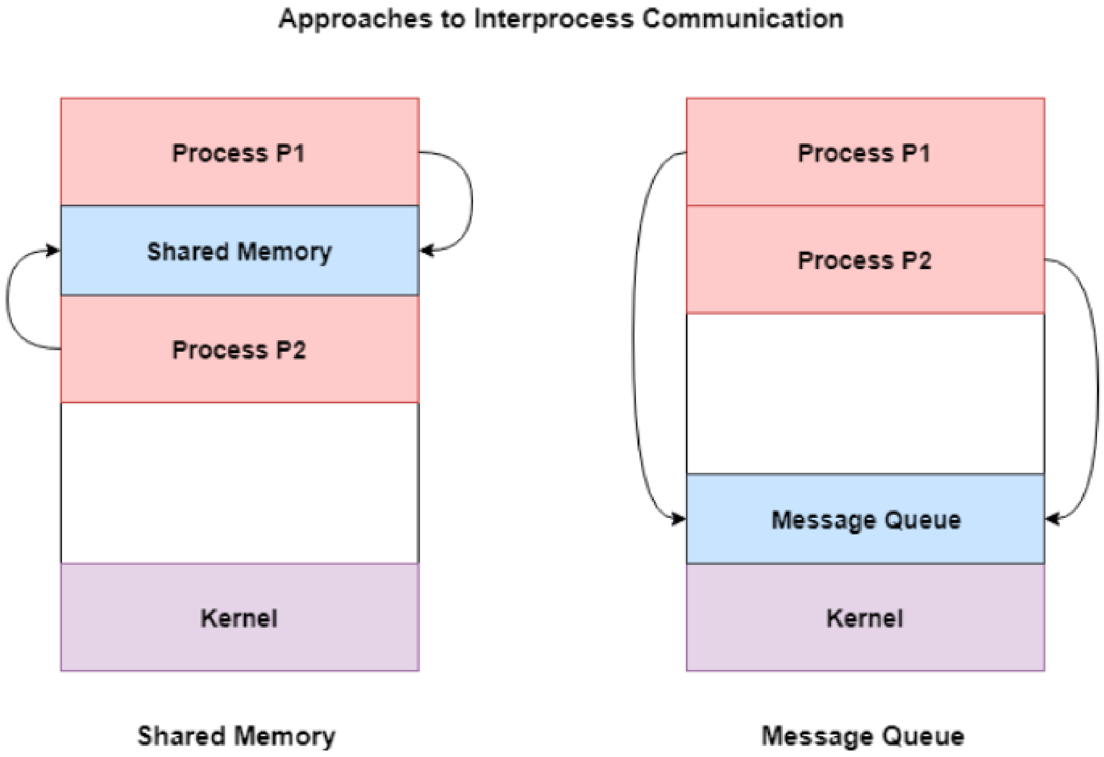
The different approaches to implement inter-process communication are given as follows −

- **Pipe**
  A pipe is a data channel that is unidirectional. Two pipes can be used to create a two-way data channel between two processes. This uses standard input and output methods. Pipes are used in all POSIX systems as well as Windows operating systems.
- **Socket**
  The socket is the endpoint for sending or receiving data in a network. This is true for data sent between processes on the same computer or data sent between different computers on the same network. Most of the operating systems use sockets for inter-process communication.
- **File**
  A file is a data record that may be stored on a disk or acquired on demand by a file server. Multiple processes can access a file as required. All operating systems use files for data storage.
- **Signal**
  Signals are useful in inter-process communication in a limited way. They are system messages that are sent from one process to another. Normally, signals are not used to transfer data but are used for remote commands between processes.

- **Shared Memory**
  Shared memory is the memory that can be simultaneously accessed by multiple processes. This is done so that the processes can communicate with each other. All POSIX systems, as well as Windows operating systems use shared memory.

- **Message Queue**
  Multiple processes can read and write data to the message queue without being connected to each other. Messages are stored in the queue until their recipient retrieves them. Message queues are quite useful for inter-process communication and are used by most operating systems.

A diagram that demonstrates message queue and shared memory methods of inter-process communication is as follows –



Process scheduling:

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

Categories of Scheduling:

There are two categories of scheduling:

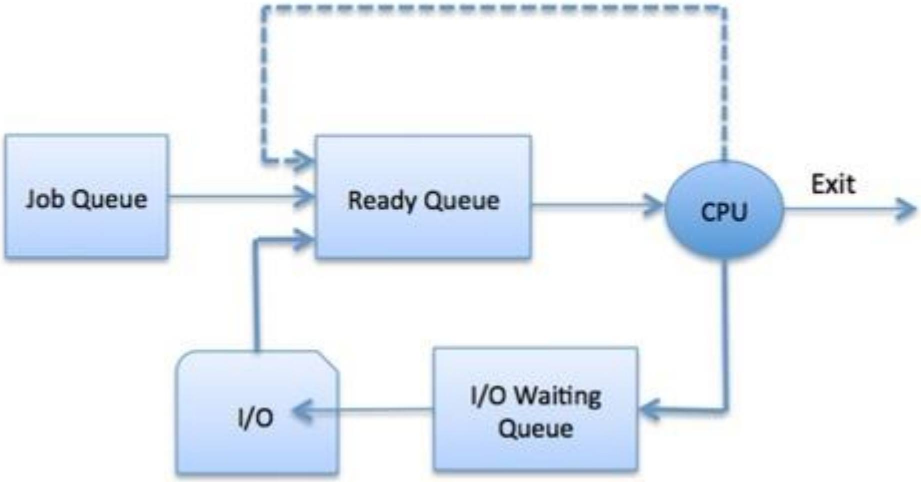
1. **Non-preemptive:** Here the resource can't be taken from a process until the process completes execution. The switching of resources occurs when the running process terminates and moves to a waiting state.
2. **Preemptive:** Here the OS allocates the resources to a process for a fixed amount of time. During resource allocation, the process switches from running state to ready state or from waiting state to ready state. This switching occurs as the CPU may give priority to other processes and replace the process with higher priority with the running process.

Process Scheduling Queues:

The OS maintains all Process Control Blocks (PCBs) in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

The Operating System maintains the following important process scheduling queues −

- **Job queue** − This queue keeps all the processes in the system.
- **Ready queue** − This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues** − The processes which are blocked due to unavailability of an I/O device constitute this queue.



The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). The OS scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system; in the above diagram, it has been merged with the CPU.

Two-State Process Model

Two-state process model refers to running and non-running states which are described below −

S.N.	State & Description
1	**Running** When a new process is created, it enters into the system as in the running state.
2	**Not Running** Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Schedulers

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types −

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

Long Term Scheduler:

It is also called a **job scheduler**. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

Short Term Scheduler:

It is also called as **CPU scheduler**. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

Medium Term Scheduler:

Medium-term scheduling is a part of **swapping**. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called **swapping**, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. Such as:

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Round Robin(RR) Scheduling

<u>FCFS Scheduling Algorithms</u>:

First come first serve (FCFS) scheduling algorithm simply schedules the jobs according to their arrival time. The job which comes first in the ready queue will get the CPU first. The lesser the arrival time of the job, the sooner will the job get the CPU. FCFS scheduling may cause the problem of starvation if the burst time of the first process is the longest among all the jobs.

Example:

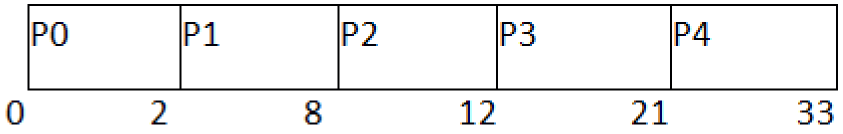
Let's take an example of The FCFS scheduling algorithm. In the Following schedule, there are 5 processes with process ID P0, P1, P2, P3 and P4. P0 arrives at time 0, P1 at time 1, P2 at time 2, P3 arrives at time 3 and Process P4 arrives at time 4 in the ready queue. The processes and their respective Arrival and Burst time are given in the following table.

The Turnaround time and the waiting time are calculated by using the following formula.

- **Turn Around Time = Completion Time - Arrival Time**
- **Waiting Time = Turnaround time - Burst Time**

The average waiting Time is determined by summing the respective waiting time of all the processes and divided the sum by the total number of processes.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
0	0	2	2	2	0
1	1	6	8	7	1
2	2	4	12	10	6
3	3	9	21	18	9
4	6	12	33	29	17



Avg Waiting Time=31/5

****Note: Follow the class lecture for better understanding****

Shortest Job First (SJF) Scheduling:

Till now, we were scheduling the processes according to their arrival time (in FCFS scheduling). However, SJF scheduling algorithm, schedules the processes according to their burst time.

In SJF scheduling, the process with the lowest burst time, among the list of available processes in the ready queue, is going to be scheduled next.

However, it is very difficult to predict the burst time needed for a process hence this algorithm is very difficult to implement in the system.

Example:

In the following example, there are five jobs named as P1, P2, P3, P4 and P5. Their arrival time and burst time are given in the table below.

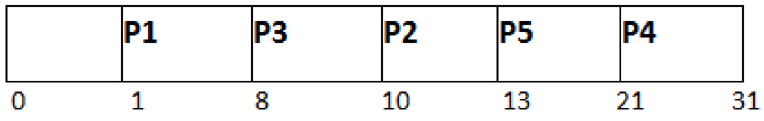
The Turnaround time and the waiting time are calculated by using the following formula.

- **Turn Around Time = Completion Time - Arrival Time**
- **Waiting Time = Turnaround time - Burst Time**

PID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	1	7	8	7	0
2	3	3	13	10	7
3	6	2	10	4	2
4	7	10	31	24	14
5	9	8	21	12	4

- Since, No Process arrives at time 0 hence; there will be an empty slot in the Gantt chart from time 0 to 1 (the time at which the first process arrives).
- According to the algorithm, the OS schedules the process which is having the lowest burst time among the available processes in the ready queue.
- Till now, we have only one process in the ready queue hence the scheduler will schedule this to the processor no matter what is its burst time.
- This will be executed till 8 units of time. Till then we have three more processes arrived in the ready queue hence the scheduler will choose the process with the lowest burst time.
- Among the processes given in the table, P3 will be executed next since it is having the lowest burst time among all the available processes.

  So that's how the procedure will go on in shortest job first (SJF) scheduling algorithm.



Avg Waiting Time = 27/5

****Note: Follow the class lecture for better understanding****

Shortest Job First (SJF) Scheduling preemptive version:

This Algorithm is the preemptive version of SJF scheduling. In SRTF, the execution of the process can be stopped after certain amount of time. At the arrival of every process, the short term scheduler schedules the process with the least remaining burst time among the list of available processes and the running process.

Once all the processes are available in the ready queue, No preemption will be done and the algorithm will work as SJF scheduling. The context of the process is saved in the Process Control Block when the process is removed from the execution and the next process is scheduled. This PCB is accessed on the next execution of this process.

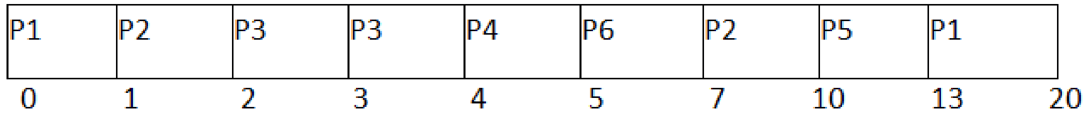
Example:

In this Example, there are five jobs P1, P2, P3, P4, P5 and P6. Their arrival time and burst time are given below in the table.

The Turnaround time and the waiting time are calculated by using the following formula.

- **Turn Around Time = Completion Time - Arrival Time**
- **Waiting Time = Turnaround time - Burst Time**

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	0	8	20	20	12
2	1	4	10	9	5
3	2	2	4	2	0
4	3	1	5	2	1
5	4	3	13	9	6
6	5	2	7	2	0



Avg Waiting Time = 24/6

****Note: Follow the class lecture for better understanding****

Round Robin (RR) Scheduling: (Follow the Class lecture)

Client-Server Model/System:

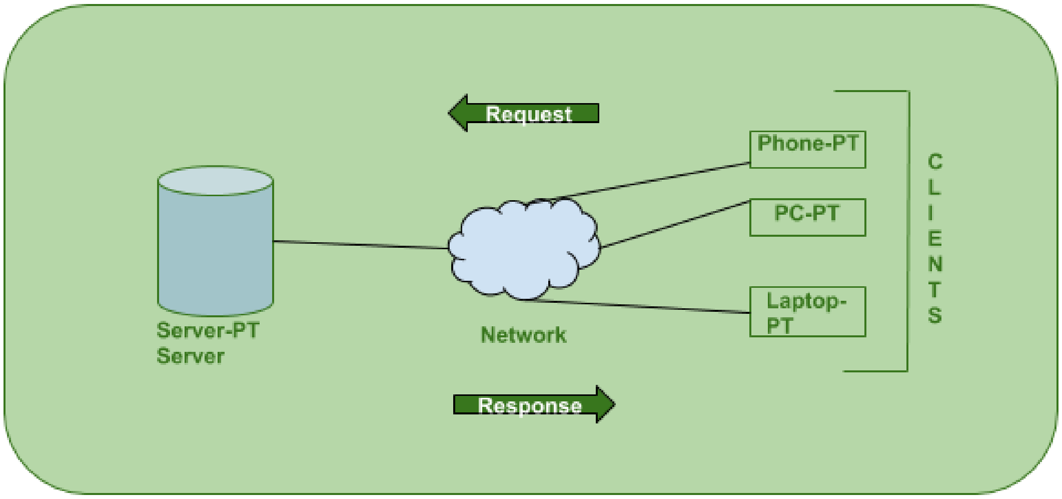
In the client-server architecture, when the client computer sends a request for data to the server through the internet, the server accepts the requested process and deliver the data packets requested back to the client. Clients do not share any of their resources. Examples of Client-Server Model are Email, World Wide Web, etc.

How the Client-Server Model works?

In this article we are going to take a dive into the Client-Server model and have a look at how the Internet works via, web browsers. This article will help us in having a solid foundation of the WEB and help in working with WEB technologies with ease.

- **Client:** When we talk the word Client, it mean to talk of a person or an organization using a particular service. Similarly in the digital world a Client is a computer (Host) i.e. capable of receiving information or using a particular service from the service providers (Servers).
- **Servers:** Similarly, when we talk the word Servers, It mean a person or medium that serves something. Similarly in this digital world a Server is a remote computer which provides information (data) or access to particular services.

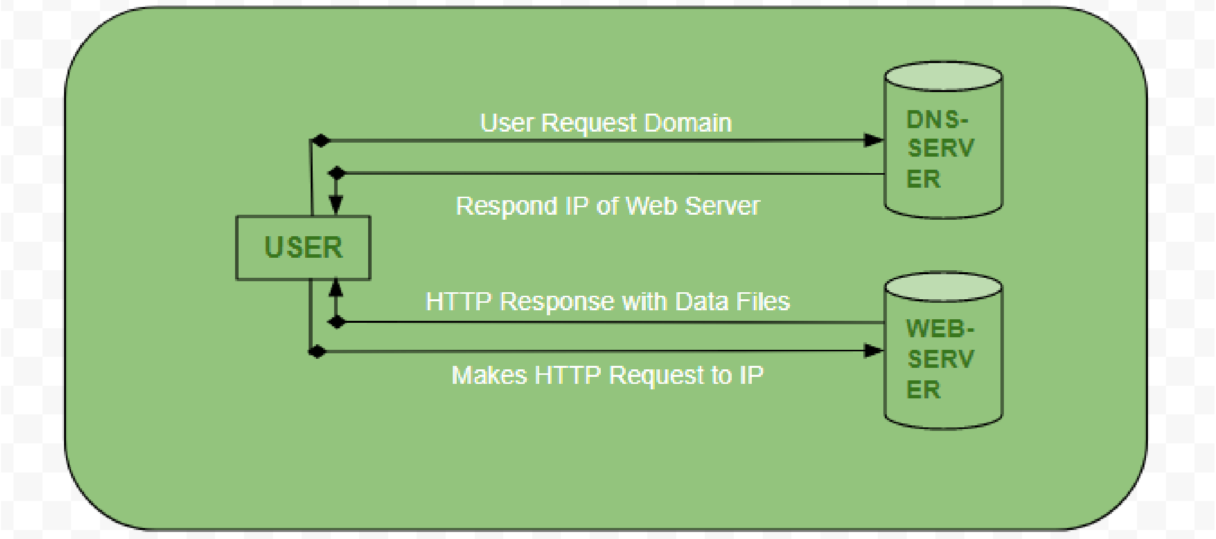
So, it’s basically the Client requesting something and the Server serving it as long as it’s present in the database.



How the browser interacts with the servers?

There are few steps to follow to interact with the servers a client.

- User enters the **URL** (Uniform Resource Locator) of the website or file. The Browser then requests the **DNS** (DOMAIN NAME SYSTEM) Server.
- **DNS Server** lookup for the address of the **WEB Server**.
- **DNS Server** responds with the **IP address** of the **WEB Server.**
- Browser sends over an **HTTP/HTTPS** request to **WEB Server's IP** (provided by DNS server).
- Server sends over the necessary files of the website.
- Browser then renders the files and the website is displayed. This rendering is done with the help of **DOM** (Document Object Model) interpreter, **CSS** interpreter and **JS Engine** collectively known as the **JIT** or (Just in Time) Compilers.


User Request Domain
DNS-SERVER
Respond IP of Web Server
USER
HTTP Response with Data Files
WEB-SERVER
Makes HTTP Request to IP