BITS Pilani

Full Stack Application Development

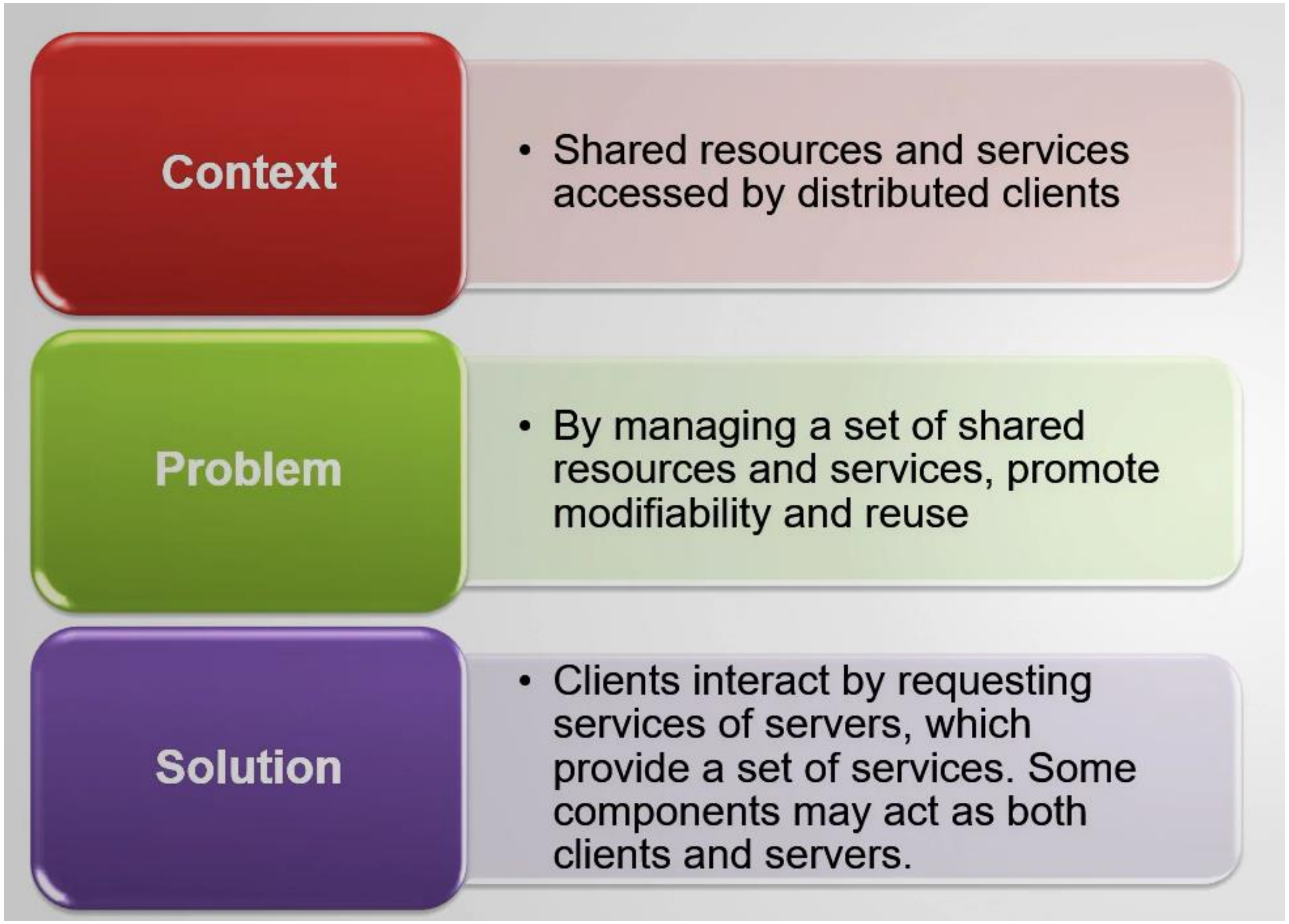
Module 2: Understanding Basic Web Applications

innovate achieve lead

BITS Pilani
Pilani Campus

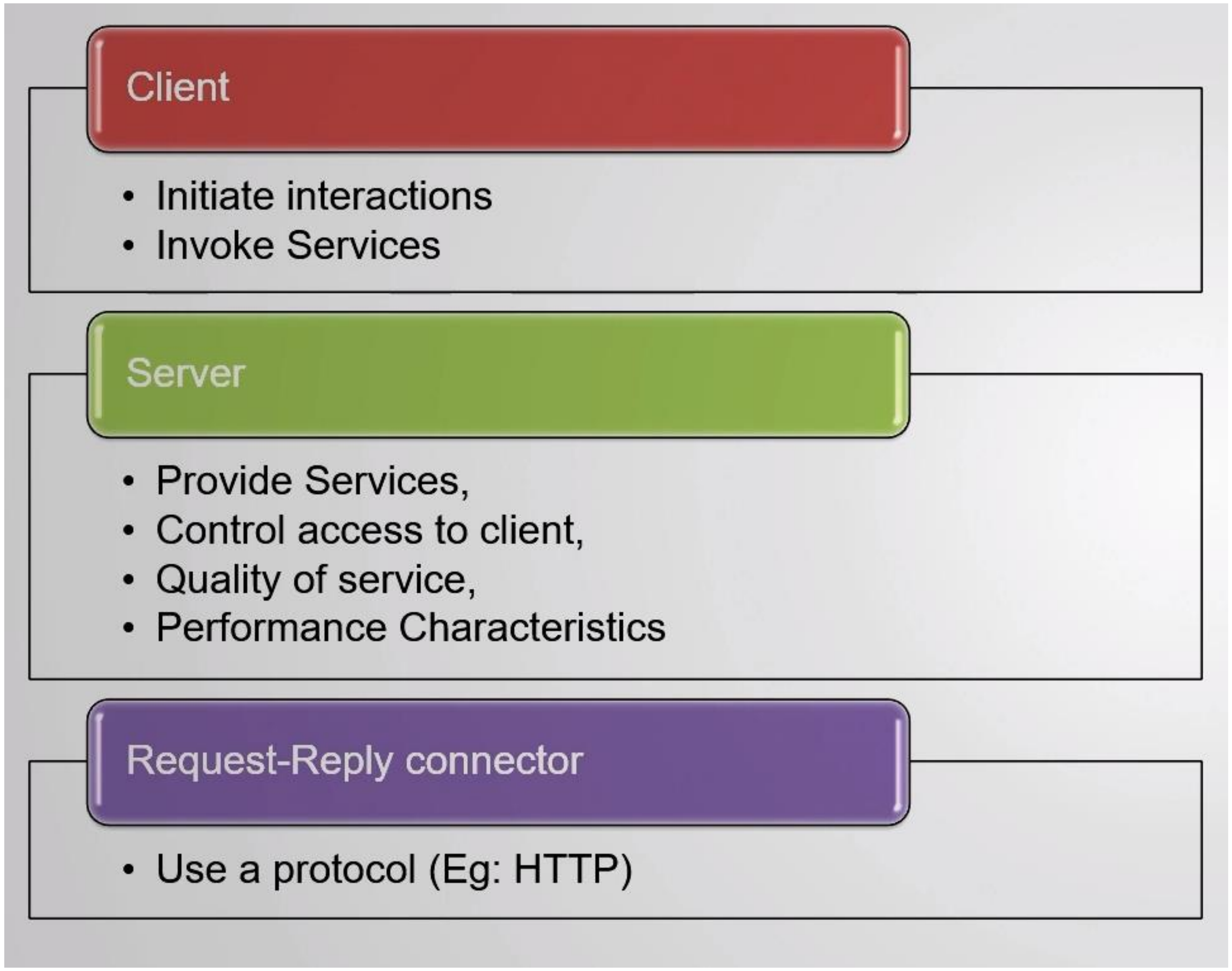
Client-Server Architecture

Client Server Pattern



Context	• Shared resources and services accessed by distributed clients
Problem	• By managing a set of shared resources and services, promote modifiability and reuse
Solution	• Clients interact by requesting services of servers, which provide a set of services. Some components may act as both clients and servers.

Client-Server Architecture



Client

- Initiate interactions
- Invoke Services

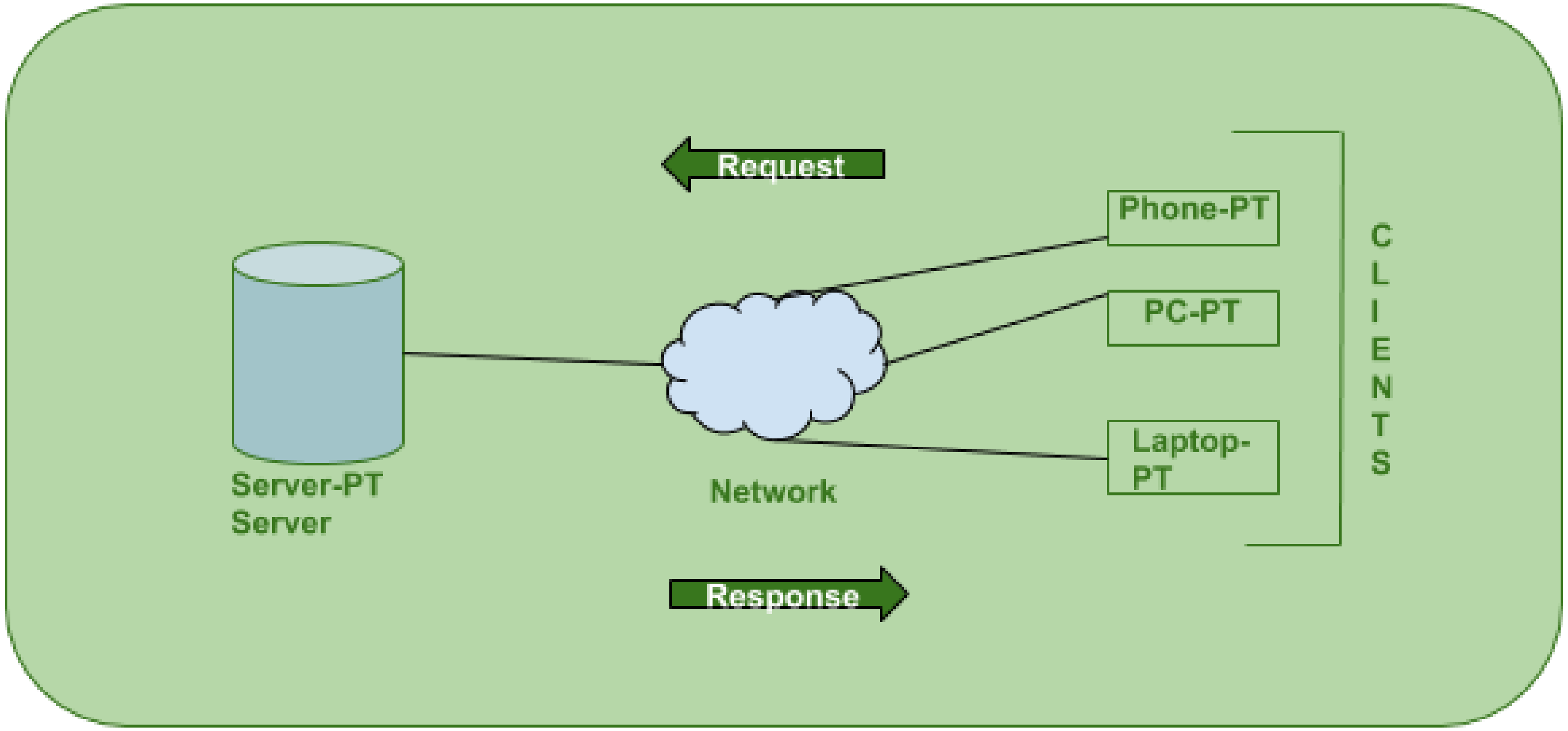
Server

- Provide Services,
- Control access to client,
- Quality of service,
- Performance Characteristics

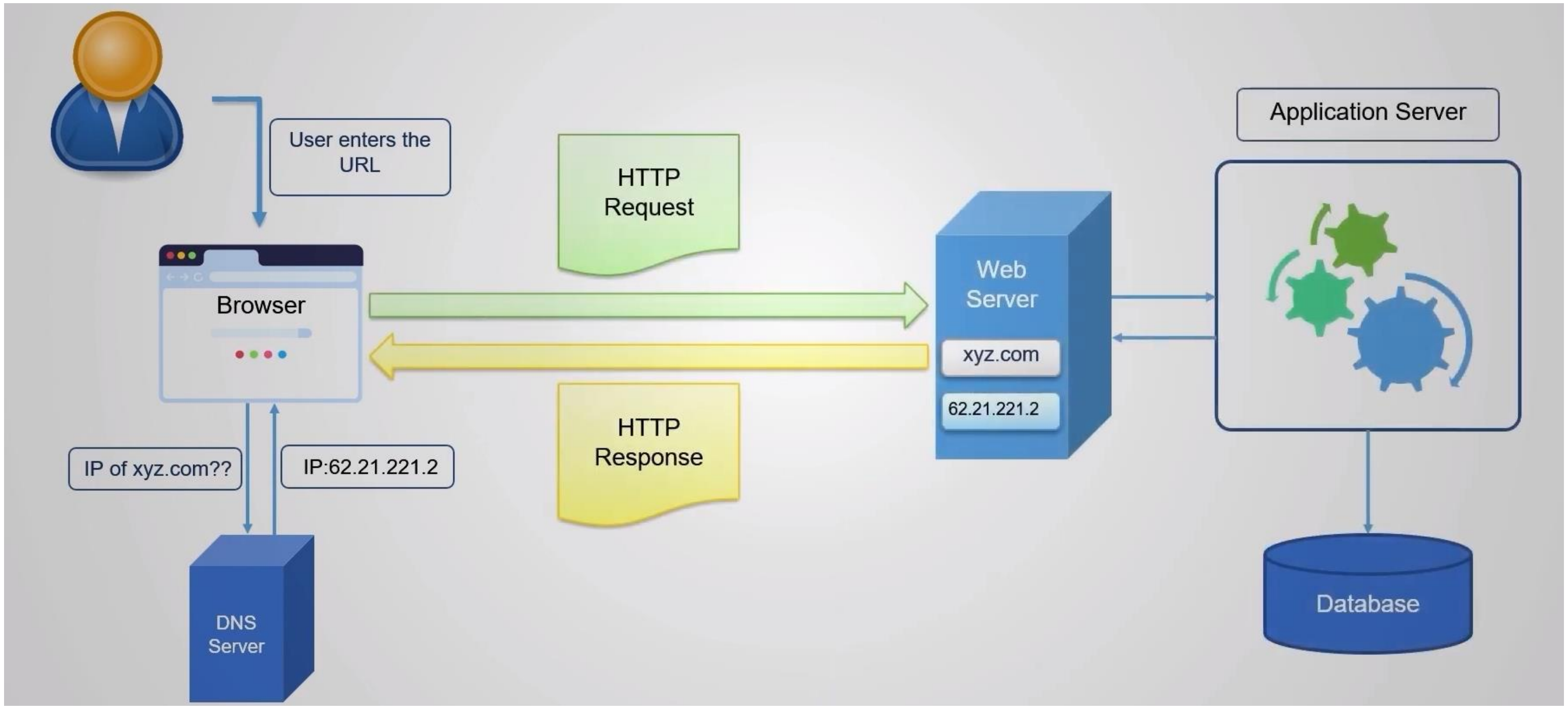
Request-Reply connector

- Use a protocol (Eg: HTTP)

How the browsers interact with Servers?



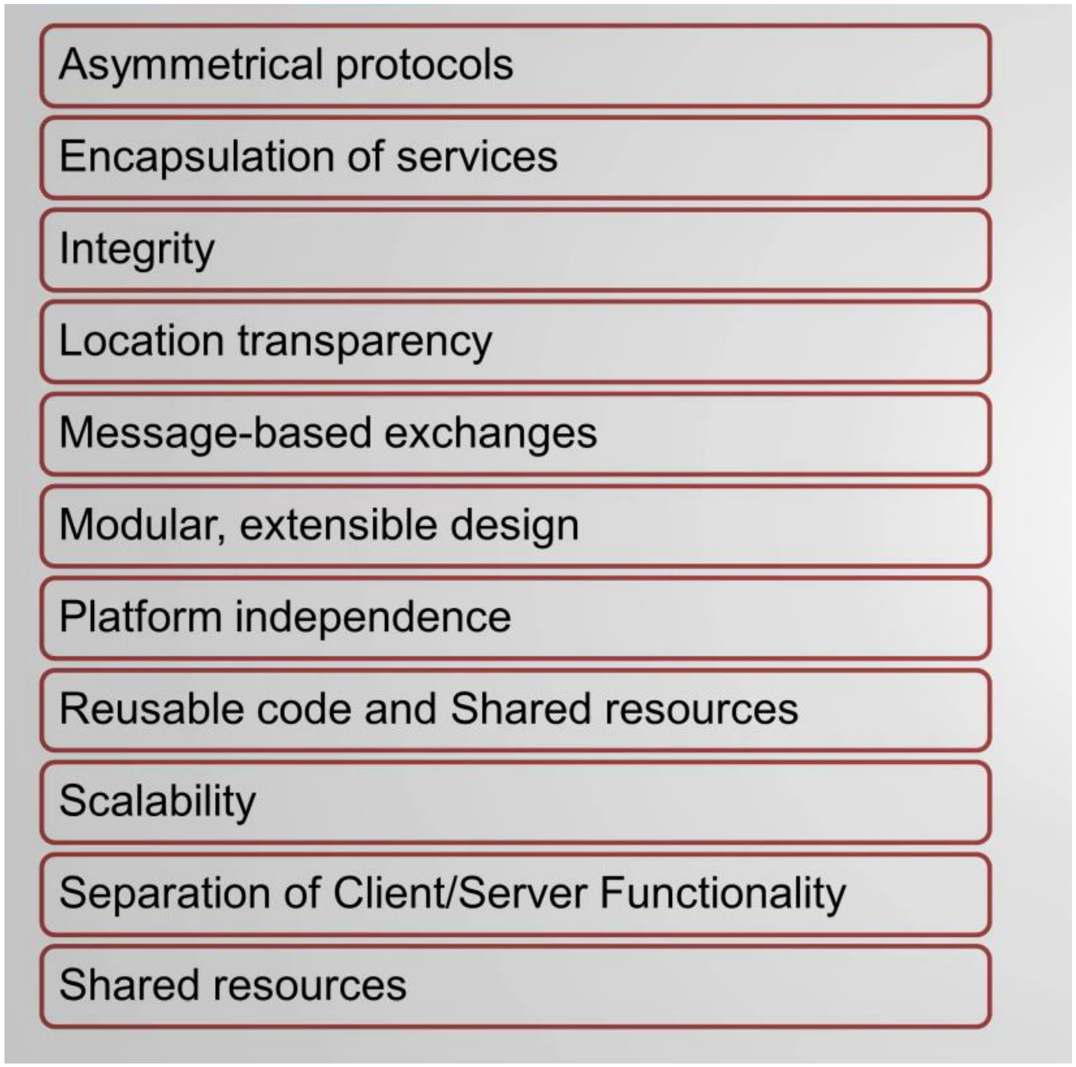
Client-Server Interaction



Answer

1. User enters the **URL**(Uniform Resource Locator) of the website or file. The Browser then requests the **DNS**(DOMAIN NAME SYSTEM) Server.
2. **DNS Server** lookup for the address of the **WEB Server**.
3. **DNS Server** responds with the **IP address** of the **WEB Server**.
4. Browser sends over an **HTTP/HTTPS** request to **WEB Server's IP** (provided by **DNS server**).
5. Server sends over the necessary files of the website.
6. 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.

Characteristics


- Asymmetrical protocols
- Encapsulation of services
- Integrity
- Location transparency
- Message-based exchanges
- Modular, extensible design
- Platform independence
- Reusable code and Shared resources
- Scalability
- Separation of Client/Server Functionality
- Shared resources

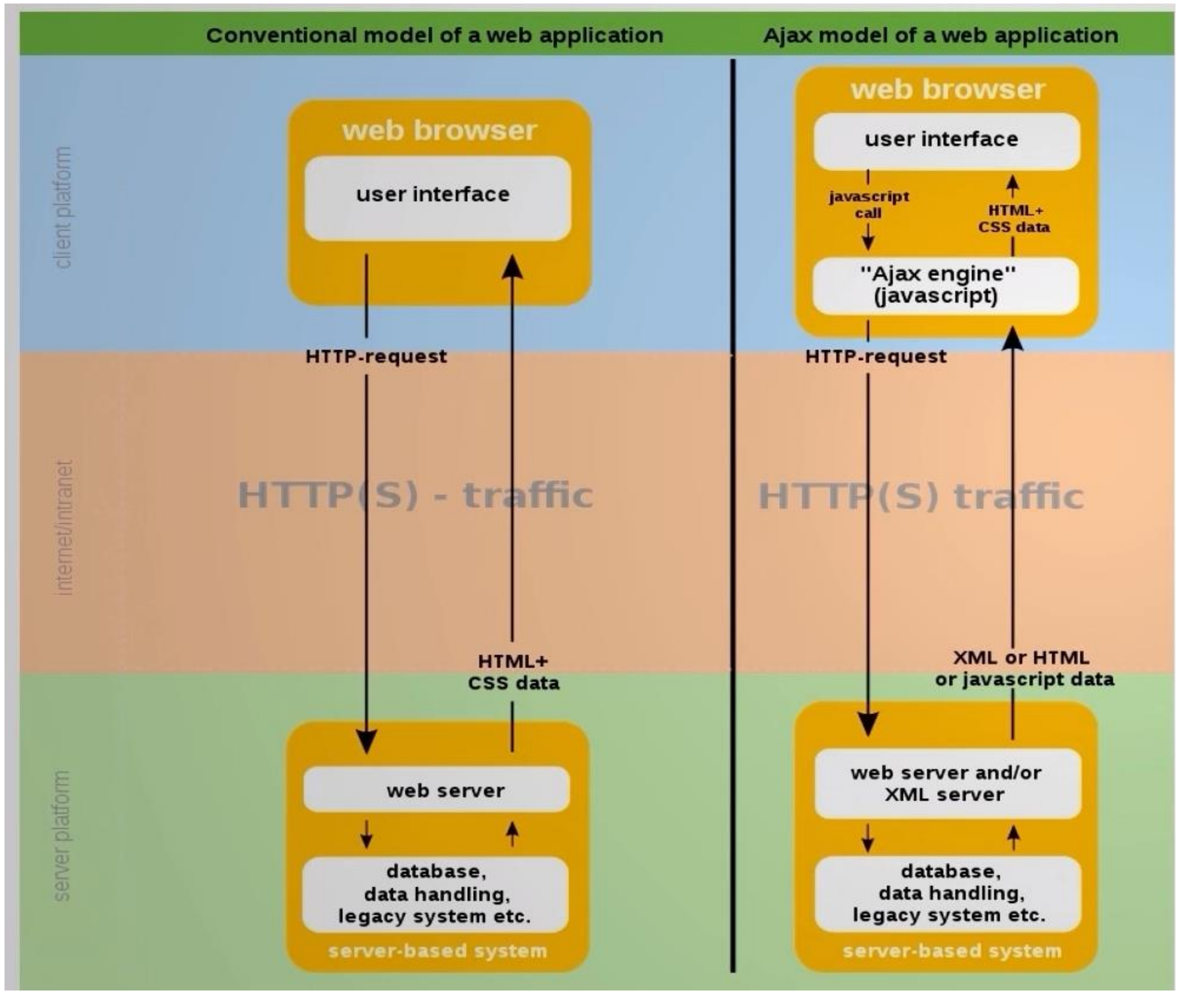
Advantages

- Centralized system with all data in a single place.
- Cost efficient requires less maintenance cost and Data recovery is possible.
- The capacity of the Client and Servers can be changed separately.

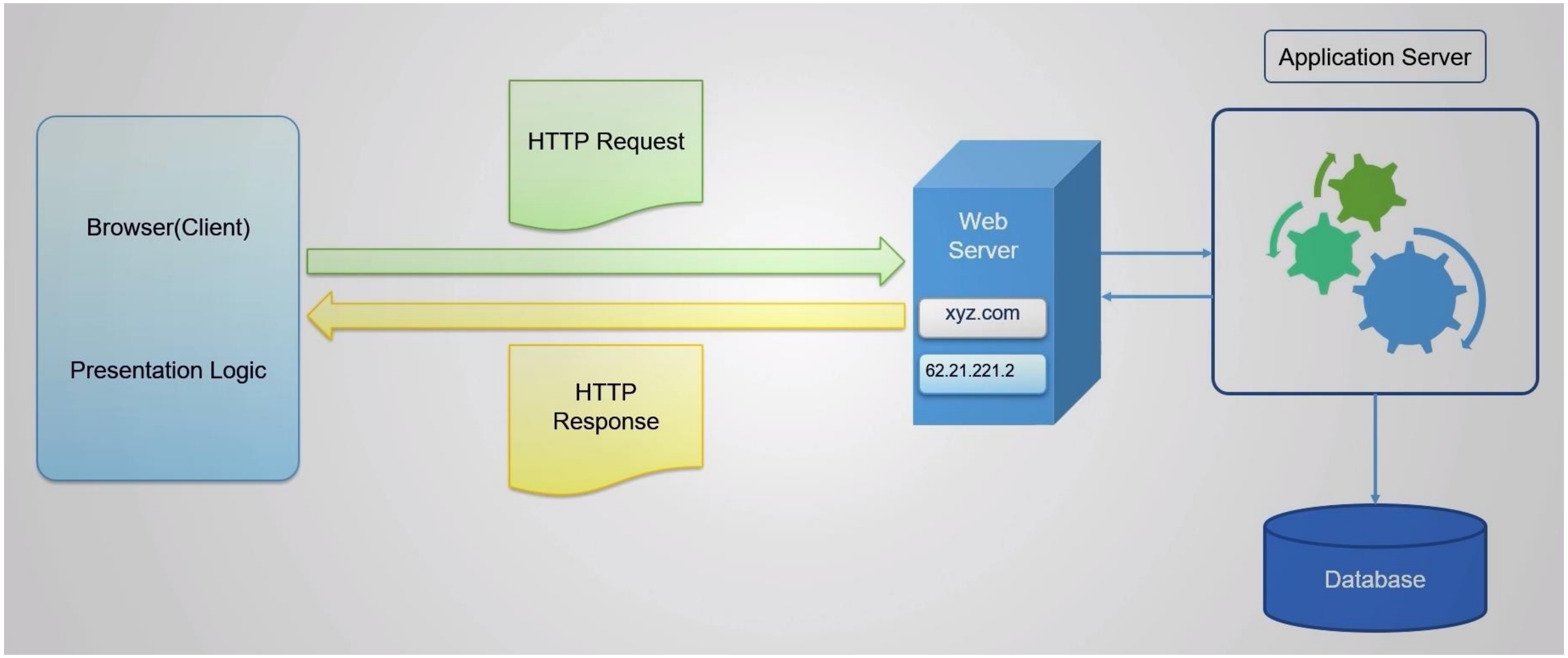
Disadvantages

- ❑ Clients are prone to viruses, Trojans and worms if present in the Server or uploaded into the Server.
- ❑ **Single point of failure**: If the server fails, all the clients will be unable to access the system.
- ❑ **Network dependence**: The system relies on a reliable network connection between the client and the server.
- ❑ Server are prone to Denial of Service (DOS) attacks.
- ❑ Data packets may be spoofed or modified during transmission.
- ❑ Phishing or capturing login credentials or other useful information of the user are common and MITM(Man in the Middle) attacks are common.

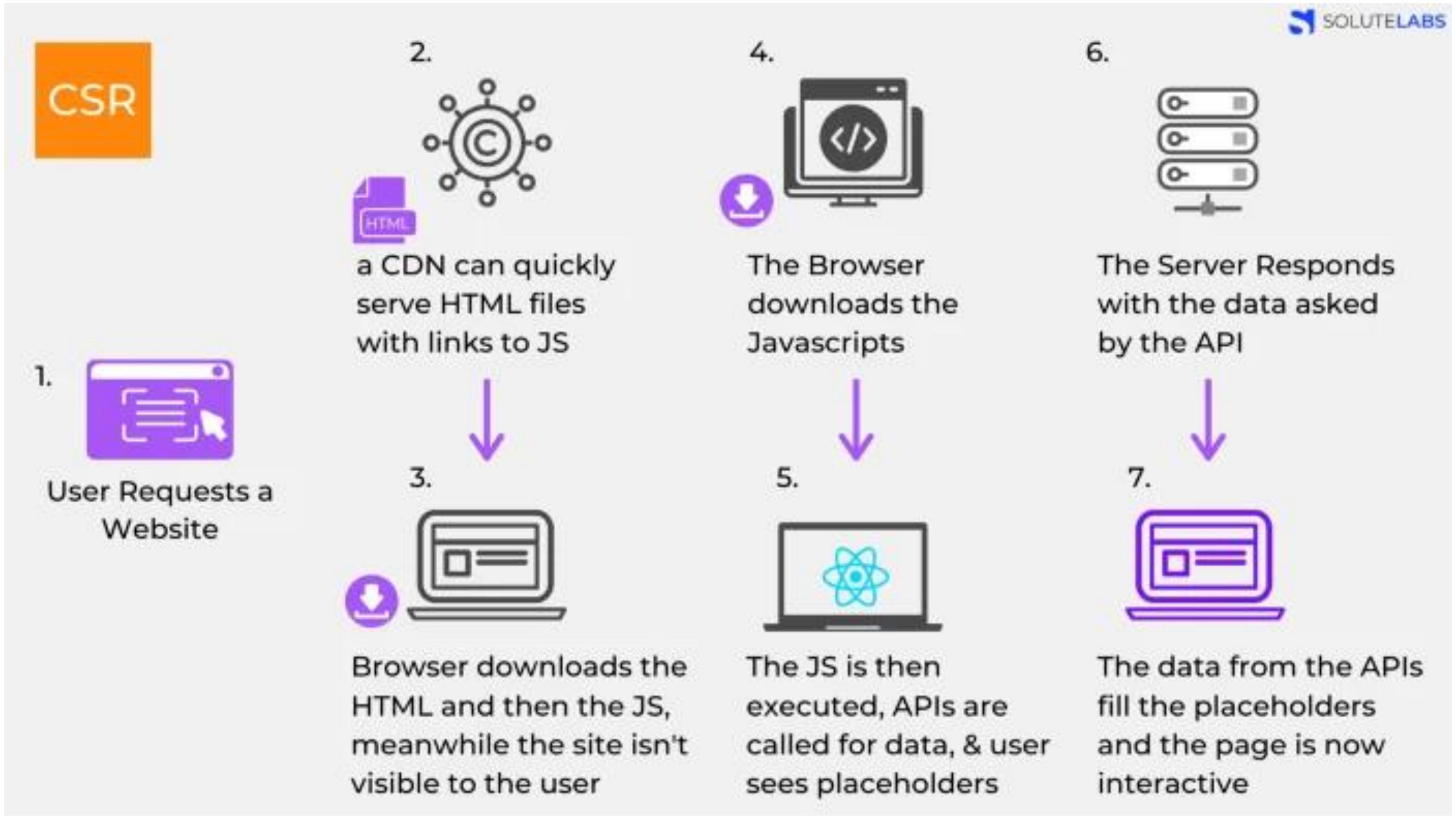
Model of Web Application



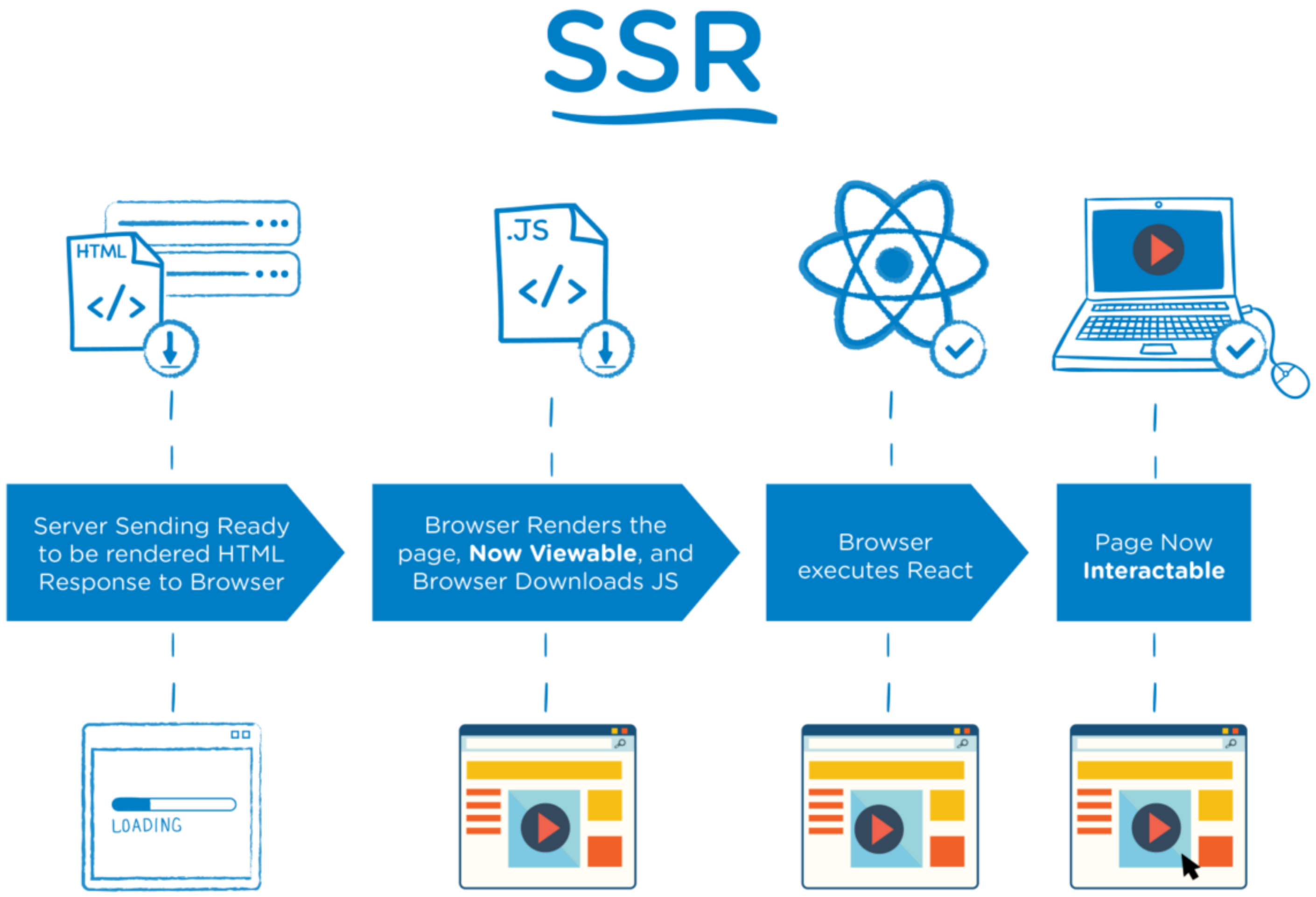
Client-Side Rendering



Client-Side Rendering


SOLUTELABS
CSR
1.
User Requests a Website
2.
HTML
a CDN can quickly serve HTML files with links to JS
3.
Browser downloads the HTML and then the JS, meanwhile the site isn't visible to the user
4.
The Browser downloads the Javascripts
5.
The JS is then executed, APIs are called for data, & user sees placeholders
6.
The Server Responds with the data asked by the API
7.
The data from the APIs fill the placeholders and the page is now interactive

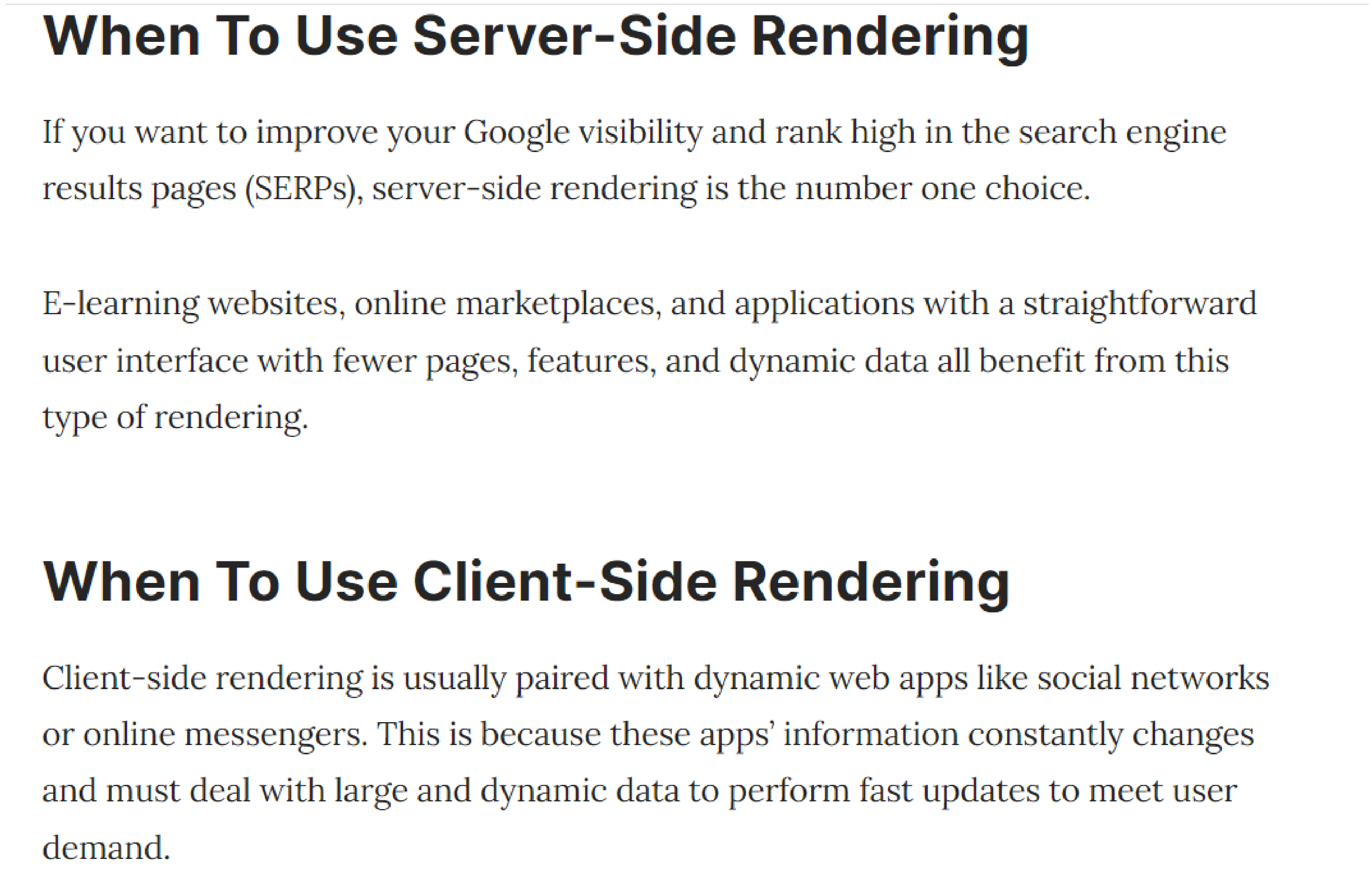
Server-Side Rendering


SSR
HTML
.JS
Server Sending Ready to be rendered HTML Response to Browser
Browser Renders the page, **Now Viewable**, and Browser Downloads JS
Browser executes React
Page Now **Interactable**
LOADING

Question

Which one is better? CSR or SSR?

When to Use?



When To Use Server-Side Rendering

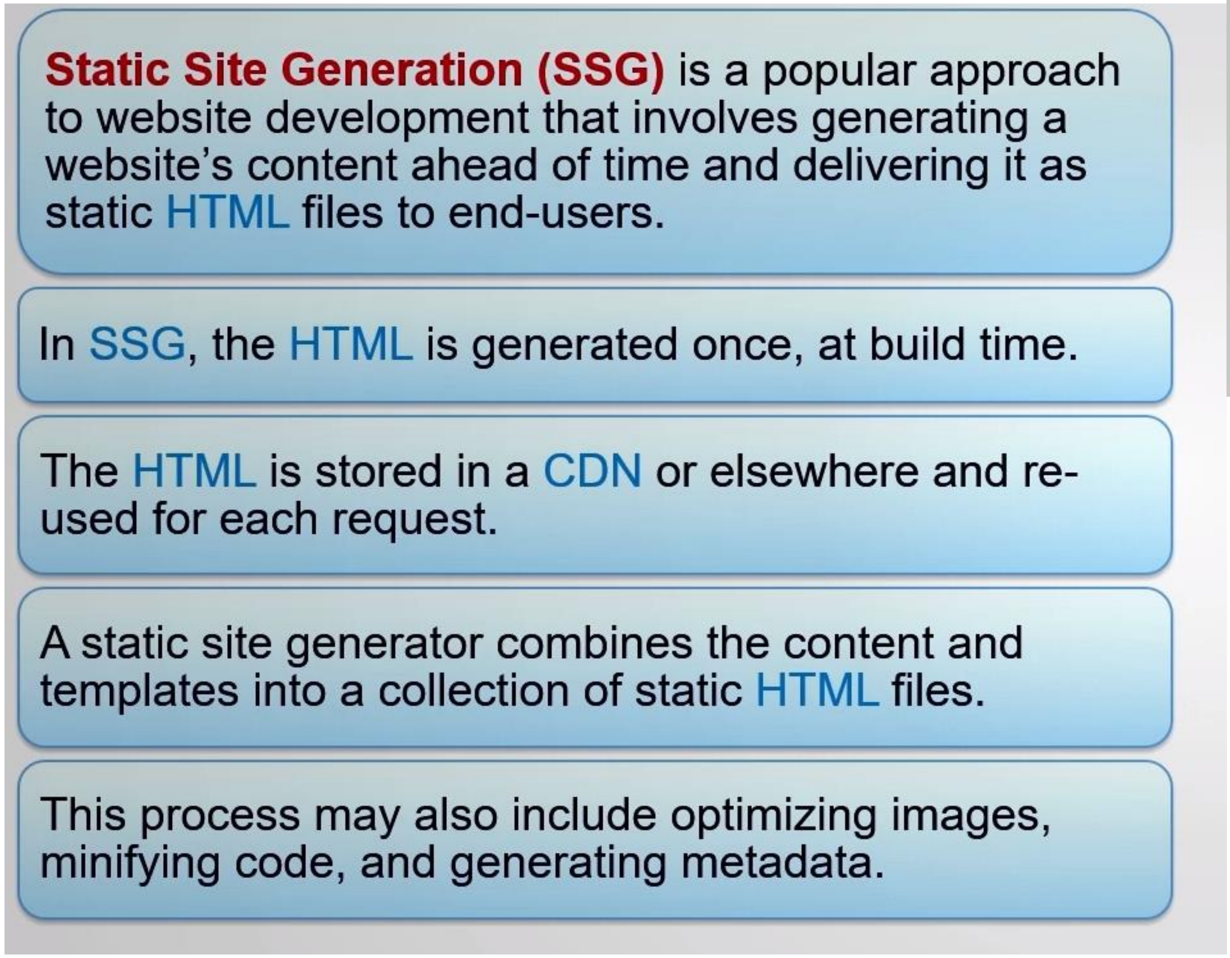
If you want to improve your Google visibility and rank high in the search engine results pages (SERPs), server-side rendering is the number one choice.

E-learning websites, online marketplaces, and applications with a straightforward user interface with fewer pages, features, and dynamic data all benefit from this type of rendering.

When To Use Client-Side Rendering

Client-side rendering is usually paired with dynamic web apps like social networks or online messengers. This is because these apps' information constantly changes and must deal with large and dynamic data to perform fast updates to meet user demand.

Static Side Generation



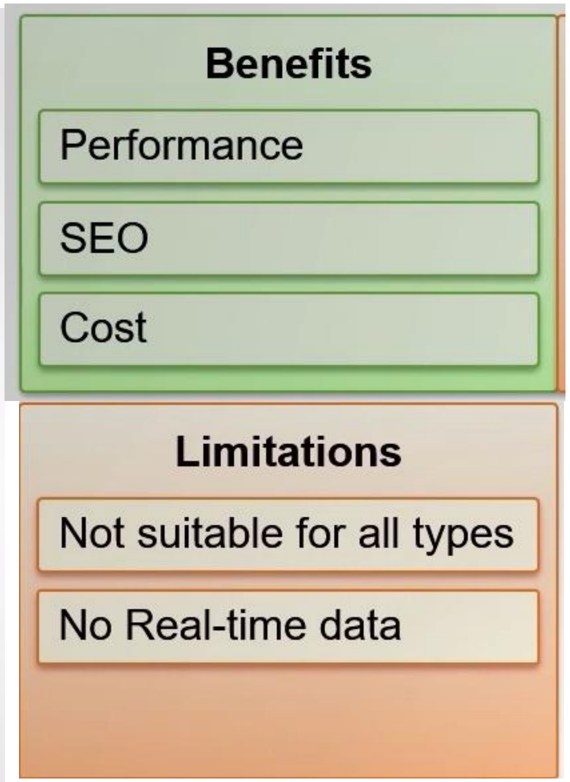
Static Site Generation (SSG) is a popular approach to website development that involves generating a website's content ahead of time and delivering it as static HTML files to end-users.

In SSG, the HTML is generated once, at build time.

The HTML is stored in a CDN or elsewhere and re-used for each request.

A static site generator combines the content and templates into a collection of static HTML files.

This process may also include optimizing images, minifying code, and generating metadata.



Benefits

- Performance
- SEO
- Cost

Limitations

- Not suitable for all types
- No Real-time data

Case Study : Fat Server Thin Client

1. Fat Server (Backend):

1. The **core component** within the architecture.
2. Installed and configured with **key applications and processes**.
3. Handles most of the computational tasks and data processing.
4. Typically equipped with **greater memory**, **hard drive space**, and **faster CPUs**.
5. **Stores the database** and **business logic**.
6. Acts as the central hub for managing requests from clients.

2. Thin Clients (Frontend):

1. These are the **client devices** (such as desktops, laptops, or mobile devices).
2. They have **minimal processing power** and rely heavily on the server.
3. Thin clients primarily handle **user interface rendering** and **user input**.
4. They connect to the server over a network (e.g., LAN or the Internet).
5. Examples include web browsers, terminal emulators, or lightweight mobile apps.

https://naveenkumarmuguda.medium.com/fat-server-thin-client-ac79f33c28f0

Group Discussion

Group Discussion

Imagine you're part of a team designing the communication system for a new Mars rover mission. This rover will be exploring a remote region of the Red Planet, tasked with collecting scientific data and sending it back to Earth. Communication with Earth will be crucial, but it presents several challenges:

- **Extreme distance:** Mars is millions of kilometers away, resulting in significant signal delay (up to 22 minutes one-way).
- **Limited bandwidth:** Sending large amounts of data through interplanetary space is costly and requires careful optimization.
- **Harsh environment:** Dust storms, radiation, and other factors can interfere with signal transmission.
- **Autonomous operation:** The rover needs to make decisions on its own and communicate them in real-time without constant human intervention.

You need to

Design a client-server architecture for the rover's communication system that addresses these challenges. Here are some specific considerations:

- **Client-side (rover):**
  - What type of client software will run on the rover? How will it handle limited processing power and storage?
  - How will data be prioritized and compressed for efficient transmission?
  - How will the rover handle communication disruptions and re-establish connections autonomously?
- **Server-side (Earth):**
  - What server infrastructure is needed to receive and process data from the rover?
  - How will real-time communication and command delivery be achieved despite signal delay?
  - How will data be stored, analyzed, and made accessible to scientists?

Network Communication (optional)

- What communication protocols will be used to ensure reliable data transfer over vast distances?
- How will error correction and redundancy be implemented to handle data loss or corruption?
- How will the system adapt to changing network conditions and potential interference?

Your Answer

- **Technical feasibility:** Does the design utilize existing technologies and address the technical challenges realistically?
- **Robustness and reliability:** Can the system withstand the harsh Martian environment and unexpected situations?
- **Efficiency and optimization:** Does the design minimize bandwidth usage and maximize data transfer efficiency?
- **Autonomy and adaptability:** Can the rover operate independently and respond to changing circumstances effectively?
- **Security and scalability:** Does the design incorporate necessary security measures and allow for future expansion?