**Basic Terminology**

1. **Runtime Environment**

A runtime environment creates the necessary infrastructure for software applications to run by providing resources, libraries, and services needed for execution, allowing developers to focus on writing application logic without having to manage low-level details.

In other words, A runtime environment is a platform or system that provides the necessary resources and services for executing and running software applications. It includes various components and libraries that allow software programs to function correctly and interact with the underlying hardware and software infrastructure. In essence, a runtime environment creates an execution environment for the software code to operate within.

**Examples of runtime environments include:**

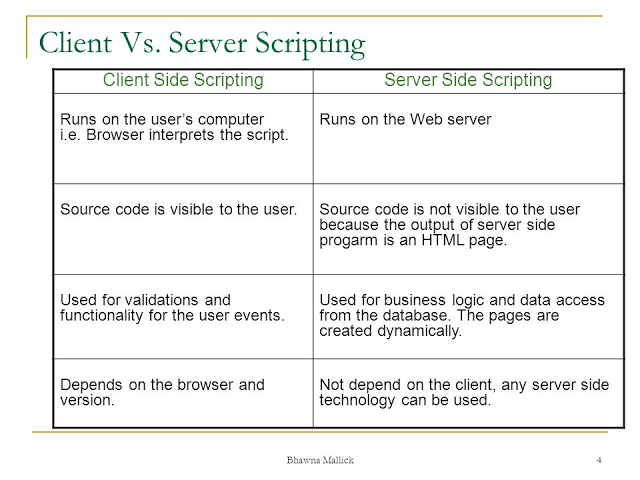
**JavaScript Runtime (Node.js):** Node.js provides a runtime environment for executing JavaScript code on the server side.

**Java Virtual Machine (JVM):** The JVM is a runtime environment that allows Java code to run on various platforms.

**Python Interpreter:** The Python interpreter is a runtime environment for executing Python scripts.

**.NET Runtime (Common Language Runtime):** This runtime environment allows languages like C#, Visual Basic, and F# to run on the .NET framework.

1. **Client side Scripting v/s Server side Scripting language**



The client-side scripting language involves languages such as HTML, CSS and JavaScript. The server-side scripting language involves languages such as PHP, ASP.net, Ruby, ColdFusion, Python, C#, Java, C++, etc.

Server-side scripting is more secure than client-side scripting as the server side scripts are usually hidden from the client end, while a client-side script is visible to the users.

1. **Client-Side-JavaScript v/s Server-Side-JavaScript**

There are 3 major types of JavaScript namely:

**Client-Side JavaScript (CSJS) --** an extended version of JavaScript that enables the enhancement and manipulation of web pages and client browsers.

CSJS allows you to make interaction with DOM and adding events to HTML elements.

**Server-Side JavaScript (SSJS) --** an extended version of JavaScript that enables back-end access to databases, file systems, and servers.

SSJS allows you to build more scalable, event-driven and non-blocking applications. Example Node js.

**Core JavaScript --** the base JavaScript language

Client-Side JavaScript (CSJS) and Server-Side JavaScript (SSJS) are dependent on the core JavaScript and cannot work without it.