Software and Programming Languages

Software

Computers do not understand human language. They do understand 1s and 0s, [bits and bytes](https://lms.clarusway.com/mod/lesson/view.php?id=886). The computer programs fill this gap. A computer program, also called software, is a **set of commands** that a computer should execute one by one.

Software is a set of instructions, data or programs used to operate computers and execute specific tasks. Opposite of hardware, which describes the physical aspects of a computer, the software is a generic term used to refer to applications, scripts, and programs that run on a device. The software can be thought of as the variable part of a computer and hardware the invariable part.

| **https://docs.google.com/uc?id=1YGQ2wp7KNk7jGR13lC-CDG1v6uJTFakQ** |
| --- |
| *Software Architecture* |

Q: What is Software?  
A: Software is a set of instructions, data or programs used to operate computers and execute specific tasks. Opposite of hardware, which describes the physical aspects of a computer, software is a generic term used to refer to applications, scripts and programs that run on a computer.

Software and Programming Languages

Machine Language

The language that the computers understand, is called the **machine language**. As mentioned, this language has only two letters which are 0 and 1. The binary code sequence of these numbers has special meaning for computers. Different types of computers have different types of machine language. Because it’s hard to understand and use for people, other types of languages are developed. These are **Assembly Language (low level)** and **high-level programming languages**.

| **https://docs.google.com/uc?id=1dtVdb7AmEOwBrQkTf6l4bJBI4tpwh5p-** |
| --- |
| *Machine Language Example* |

Software and Programming Languages

Assembly Language

Assembly Language uses short descriptive words, know as **mnemonic**, (hatirlatici, ipucu) to represent each of the machine language commands. For example, mnemonic add means to add two numbers. Adding numbers 2 and 3 and getting the result in assembly code is as such:

add 2, 3, result

Assembly Language is not useful for computers. For that reason, it should be translated into machine language. The translator is called  **“assembler”**. (çevirici program, programlama dili)

Assembly Language is not much different than the machine language. And knowing the commands is not just enough. You have to know how the CPU and memory work. So it is also not much use for humans. Fortunately, this type of languages are no longer used. Therefore high-level languages are developed in place of this low-level language.

| **https://docs.google.com/uc?id=1NDjsqHfc8tPpxFTEH5zyS4FpqtTxVrJR** |
| --- |
| *Assembly Language and Machine Language* |

Q: What is Assembler?  
A: An assembler is a program that converts assembly language into machine code. It takes the basic commands and operations from assembly code and converts them into binary code.

High Level Languages

High-level languages emerged in the 1950s. These languages are **similar to human-level languages**. Much easier to understand than low level and machine language.

The commands in high level programming languages are called **statements**. There are many programming languages, each of which has upsides and downsides.

A program in a high-level language is called **source program or source code**. As in the assembly language, the source code also should be translated to machine code before it gets executed. High-level languages should use tools called **compiler derleyici or interpreter** to make this translation. An interpreter or a compiler reads one statement from the source code and turns it into machine code. Then the execution happens right away.

Q: What is the difference between compiler and interpreter?  
A: A compiler takes entire program and converts it into object code which is typically stored in a file. The object code is also refereed as binary code and can be directly executed by the machine after linking. An Interpreter directly executes instructions written in a programming or scripting language without previously converting them to an object code or machine code.

Libraries

Real-life libraries give you a book for "prepared" specific topic information. You don't have to write again the same book its already prepared by someone and you can use it, you can do whatever you want with this information. Software libraries and real-life libraries are similar. A software library generally consists of pre-written code, classes, procedures, scripts, configuration data and more. Typically, a developer might manually add a software library to a program to achieve more functionality or to automate a process without writing code for it. For example, when developing a mathematical program or application, a developer may add a mathematics software library to the program to eliminate the need for writing complex functions. All of the available functions within a software library can just be called/used within the program body without defining them explicitly. Similarly, a compiler might automatically add a related software library to a program on run time.

| **https://docs.google.com/uc?id=1o_mQMEP6TFfZUSX3rAdEqkQABvQ8GvnE** |
| --- |
| *Library* |

Q: What is a software library used for?  
A: A software library generally consists of pre-written code, classes, procedures, scripts, configuration data and more. So Software Libraries save programmers' time from writing routine processes' codes every time. For example, when developing a mathematical program or application, a developer may add a mathematics software library to the program to eliminate the need for writing complex functions

### Packages and Package Managers

**Software Package** is a set of software that fulfills a particular function, for example, installation on the desktop. Packages automatically manage software and [libraries](https://lms.clarusway.com/mod/lesson/view.php?id=917) on your system. You can install packages from source (from web site/DVD or USB) or package managers.

**Package Managers** are the tool used to manage Software and [Libraries](https://lms.clarusway.com/mod/lesson/view.php?id=917) (together called Packages) in your OS. They install, upgrade and uninstall packages easily. There are several types of package managers. The most popular ones are Linux System package managers and programming language package managers.

Frameworks

Frameworks are meant to make the life of a developer easier by supplying tools and design patterns to accomplish a task in an expressive and relatively simple way. Frameworks are like the father of software [libraries](https://lms.clarusway.com/mod/lesson/view.php?id=917). Frameworks are collections of [libraries](https://lms.clarusway.com/mod/lesson/view.php?id=917), classes, functions, and constants designed to complete a task easier. They give you a collection of different tasks codes and **avoid you from reinventing the wheel**.

Frameworks give you a working environment. For example, you want to open a coffee shop, you can buy another coffee-shop and operate it or you can build it from zero. If you buy another coffee shop you don't have to pay attention to accessories, coffee shops interior architecture, etc. Frameworks supply pre-written software working environments.

For instance, while you are thinking to write your own mobile application with Javascript programming language, the React-Native framework gives you the code that every mobile application necessary functions such as visualization to your application at a smartphone.

Frameworks dictate a specific look to an application and limit the design choices a developer can make in favor of making the code easier to read and write.

| **https://docs.google.com/uc?id=1xkUqVh5BEmNZsK-WMOIFSzJaEfOg9-4T** |
| --- |
| *Some Frameworks* |

Frameworks

Model-View-Controller (MVC)

**The Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components is built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

**Model**, the data represented in some way, usually from a database.  
**View**, what the user sees. For example the web page you look at.  
**Controller**, the code that manipulates the data in the database. For example insert/delete/update user information to your web site.

| **https://docs.google.com/uc?id=1zcy_-Nrb4so8pQujxj83PMffZ8YeYbz0** |
| --- |
| *MVC Model* |

Backend vs. Frontend

Introduction

Frontend and Backend development terms are related to **web and application development**. Before you build your web site you should first develop functionalities that what web site should do and second you should visualize it very well.

Frontend and Backend are the two most popular terms used in web development. These terms are very crucial for web development but are quite different from each other. Each side needs to communicate and operate effectively with the other as a single unit to improve the website’s functionality.

| **https://docs.google.com/uc?id=18AsCfwGqHX-wb6gVfnHLI8gAECwVAGv7** |
| --- |
| *Backend and Frontend* |

Q: What is Frontend and Backend?  
A: **The frontend** of a website or application is what you see and interact with on your device screen. Also referred to as “client-side”, it includes everything the **user experiences** directly: from text and colors to buttons, images, and navigation menus.  
  
**The backend** (or “server-side”) is the portion of the website you don’t see. It’s responsible for **storing and organizing data**, and ensuring everything on the client-side actually works. The backend communicates with the front-end, sending and receiving information to be displayed as a web page.

### Backend

Backend Development refers to the server-side (we will explain later on the [Servers](https://lms.clarusway.com/mod/lesson/view.php?id=1015) section) of development where you are primarily focused on how the site works. Making updates and changes in addition to monitoring the functionality of the site will be your primary responsibilities.

This type of web development usually consists of three parts: a **server**, an **application**, and a **database**. Code written by backend developers is what communicates the database information to the web browser. Web browser is your web pages "window application" like Google chrome, Opera, Safari etc. It just prints web site. Anything you can’t see easily with the eye such as [databases](https://lms.clarusway.com/mod/lesson/view.php?id=995) and [servers](https://lms.clarusway.com/mod/lesson/view.php?id=1015) is the work of a backend developer. Backend developer positions are often called programmers or web developers.

### Frontend

Frontend development manages everything that users visually see first in their web page or application. Frontend developers are responsible for the look and feel of a site. Frontend developers primarily use three languages: HTML, CSS, Javascript.

Here are a few examples of frontend developer job titles:

* **Web designer**, designs websites with different tools such as photoshop, fireworks or wordpress.
* **User Interface (UI) Designer** is basically a visual designer and is generally focused on design.
* **User Experience (UX) designers** work in the front-end, studying and researching how people use the sites. Then they make changes through a lot of testing.