2. Introduction to Node.js

2.1. How does a local static web page work?

Prerequisites to understand Node.js:

- Before we start learning **Node.js**, we must understand why **Node.js** is necessary to begin with. To understand why Node is necessary, we must clearly understand the **problem** it is trying to solve. To understand the problem it is trying to solve, we must clearly understand **how a website works in detail**. Once we know how websites work, it becomes much easier to see **why Node.js is useful** and **what it helps us do better or faster**.
- Let's go over all the steps that are involved in making any website work

How does a website work? (How do local static web pages work?)

- **What is a webpage?**
  - **Definition**: A web page or a webpage is a text document that is properly tagged with HTML and is viewed in a browser.
  - **What is the purpose of a webpage?** The main purpose of a web page is to display a text document in a well-organized and visually meaningful way to the user in a browser
- **Webpage vs a web application vs desktop application**:
  - **A web page** is a text document properly tagged with HTML. **Example**: Evangadi's website
  - **A web application/app**: It is a computer software that needs an internet connection or some sort of network to function properly. A web app uses data saved in remote computers (server) and will be using a browser to perform its function.

    Example: Facebook, Gmail

- **A desktop application/app**: It is a software that runs locally on your computer, meaning you need to install it on need to install it on the hard drive. Unlike web apps, desktop apps are not accessible anywhere from a web browser. **Example**: Adobe Photoshop

- **The Browser/Web Browser**:
  - **What is a browser?** It is a software that loads documents from a remote server and displays them to users that are available on the internet (world wide web). As a webpage is simply a text document, we would need help to create some useful visual representation of the text. A software that helps us create a useful visual representation of a text document is called a Browser. This means a web page without the help of a Browser is as helpless as a text document. **Note**: The browser is a software made of many software/programs that control various aspects of the browser
  - **Why do we need a browser?** The main role of a web browser is to create a visual representation of text documents based on the HTML tags used in the text document.
  - **Browser rendering**: This is a process by which the browser takes your hypertext markup language document or HTML document, converts the document to human readable format, and displays it on the browser. In short, browser rendering is a process by which the browser creates a visual representation of the HTML text document so that the users can view it on a browser screen.

- **Why do the texts in a webpage need to be tagged with HTML?** For a text to be viewed in a web browser, it must be tagged with HTML tags. When an HTML file is saved locally, all the browser needs is a way to locate the file containing the HTML. The way we achieve this is by giving a URL (Uniform Resource Locator) on the browser's address bar. Since we are trying to locate the file from a local computer, this URL is just the path to the HTML file. The path is structured by the Operating System on the computer.

  Example: /Week-1/clock/index.html
- **Browser's address bar**: The browser has a place to let us provide the location/path of the file we want to view on the browse called the address bar. A browser's address bar is located at the top of the browser. Once we provide the location of the file in the address bar, the browser will search for the file, find it, and start rendering it in a way that makes more visual sense for human eyes.
- **Browser locating files included in the HTML file**: If additional resources (such as image, CSS, or JS files) are included within the HTML file, the browser locates these resources using the file paths/URLs we specify in the HTML file. If a resource/file is located on the local computer, there is no problem for the browser to find and render it without additional help.
- **How does the browser render JS files?**
  - We know that the browser has built-in HTML and CSS interpreters to understand and render HTML and CSS files as soon as it finds the ".html" or ".css" file extensions.

- If the required file is a JavaScript file, it means the browser needs help understanding the code in the JavaScript file.
- We have said that the browser is made of different software. For the browser to understand JS files and render them, it will use one of its software called the browser engine. The job of any browser's engine is to take the JavaScript code downloaded from a remote or local computer and interpret and compile it into something the browser can understand.
- Here is a diagram showing the interactions of an HTML file, a CSS file, and a JavaScript file with the Browser.

- So far, we have discussed how to display text documents stored in a local computer in a browser. In the next section, we will see how things are handled when the text documents are stored on a remote computer/ server.

2.2. How does a remote static web page work? Requesting/receiving data from remote computer

How Do Remote Static Web Pages Work?

A **static web page** is a simple web page stored on another computer called a **remote server**. When you try to open that page, your browser sends a **request** to that remote computer asking for the page file. The remote computer then sends the file back to your browser. Once your browser receives the file, it can **display (or render)** the web page on your screen.

Static vs dynamic webpages: At the end of the day, it is an HTML page that is displayed on the browser in both cases. However, there is a difference between static and dynamic web pages when it comes to the type of HTML content they serve.

- **Static webpages**: The word static refers to something that is fixed, that doesn't move or change in any way. A static page is an HTML text document with the content already written on it. A static website contains simple HTML pages and supporting files (e.g., Cascading Style Sheets (CSS), JavaScript (JS)) hosted on a web server. When the web server receives a user request for a webpage, it sends the HTML page directly to the requesting browser. **Note**: Static pages require manual updating of the HTML document before their contents change.
- **Dynamic webpages**: A dynamic webpage is a page that displays different content each time it is viewed by a user's input, the time of day, language, location, etc. The content of a dynamic page is not yet fully written and tagged as HTML. Rather, it is generated when the browser requests the document.
    - When the web server receives a user request for a dynamic page, it does not send the page directly to the requesting browser as it would do with a static page. Instead, it passes the page to the application server, which then reads the code on the page, finishes the page according to the code's instructions, and removes the code from the page
    - **How does a dynamic website display different content?** The HTML of dynamic pages changes due to a server-side script/logic we write to instruct the remote server to execute before it sends the content to the user's browser.

      For instance, backend developers write a logic that instructs the remote server to obtain the French version of the webpage/HTML and display it if a user is sending a request from a French-speaking country.

Requesting and receiving data from a remote computer: When a static web page is saved on another computer (called a **remote server**), your **browser has to ask** that computer for the file. This is called making a **request**.

The server then **sends the file back**, and your browser **displays** the web page on your screen.

No matter where the file is — on your computer (**local**) or a different one somewhere else (**remote**) — **you must give the browser the correct web address (URL)** so it knows exactly where to look.

- **Locating the server/remote computer containing the website file**: The most common way of doing this is using the domain name of the server computer. You can locate the server using its IP address too
    - **IP (Internet Protocol) address**: It is the address of the server/ remote computer that has the website file. **Note**: The IP address is, in fact, the address of your internet router device. If we have many devices using one router for the internet, all these devices have the same IP address. IP address is a combination of special numbers that look like this: 63.245.215.20. But these numbers are not very easy to remember for the human brain. That is why we needed to have domain names.
    - **Domain name**: It is the address of the server computer that has the website file. A domain name is basically an IP address. However, because an IP address is a series of numbers that is not easy to remember for humans, a domain name was created as a human-friendly version of an IP address.

- There are global standards to follow when giving a domain name to a server computer. These standards are governed by a controlling body called ICANN.
    - **ICANN**: It manages IP addresses and domain names by registering and controls them
    - **Domain Name System (DNS)**: It is a system that translates the human readable domain names into to machine readable IP addresses whenever a user sends a request to the remote server.
    - You can read more about DNS here:
        - https://www.cloudflare.com/learning/dns/what-is-dns/

What happens when a user types a web address on a browser?

- When you type a web address into your browser, the browser looks at the DNS. DNS will compare the website's domain name entered with the IP address it has in its database before the browser can retrieve the website.
- Once the browser has the correct IP address of the server containing the requested webpage, it will send an HTTP request message to the server, asking the server to send it a copy of the website
- If the server approves the browser's request, the server sends the browser a "200 OK" message, which means "Of course you can look at that website! Here it is", and then starts sending the website's files to the browser as a series of small chunks called data packets
- Finally, the browser assembles the small chunks into a complete web page and displays it to you

How to locate/identify a website file from the server computer? Once we locate the location of the server that contains the website file, we need to locate the specific location of the website file from that computer. That specific file location should also be included in the URL we provide to our browser.

- If no particular file location is specified, the directory index file (index.html) located in the public_html folder will be the first file the server loads.

  Example: If you type in simply evangadi.com or https://www.aevangadi.com/, the server will load the directory's index file.

- If a user specifies, for instance, the "courses" location in their search, the courses.html file located in the public_html folder will be the one the server computer loads.

  Example: https://www.evangadi.com/courses.html

2.3. How does a remote static web page work? Browser getting content of the file using TCP/IP layers

Getting the file from a remote computer/server: At this point, we have the location of the file we are looking for. Getting the website file from a local computer was not a problem because the computer's operating system can be used to get the file. But in cases where the website file is located on a remote server, how will the computer containing the browser receive the file from the remote computer? It is at this point that the TCP/IP network protocol comes in handy

Transmission Control Protocol (TCP): TCP is a communications standard/protocol that enables application programs and computing devices to exchange messages over a network. It is designed to send packets across the internet and ensure the successful delivery of data and messages over networks

The Internet Protocol (IP): IP address is a unique numerical value given to every device that is connected to the internet/network that uses the Internet Protocol for communication. Every device has an IP address that uniquely identifies it and enables it to exchange data with other devices connected to the internet. **Note**: The IP address in fact the address of your internet router device. If we have many devices using one router for the internet, all these devices have the same IP address. An IP address is a combination of special numbers that look like this: 63.245.215.20

Port number: A port number is a numerical value assigned to an application or data type that is executing in the computer/device. Usually, there are multiple processes running in the device at a time. When a device receives data, the port number helps to identify the process that requires the received data.

- We have seen that an IP address is used to locate the exact server computer to which a request is sent. But how can we identify the correct application within the identified computer/server? Now, we can use the port number to identify the exact application.
- In short, a port number is added to the IP address to help in locating the application that is executing in the computer.

IP vs Port number

- **Similarity:** Both are numerical labels, and both are used for identification purposes when networked devices communicate to exchange data. A port number is "tacked on" to the end of the IP address. For example, in the "192.168. 1.67:80":
  - The "192.168. 1.67" is the IP address that represents the device
  - The number after the semi column, "80", is the port number that identifies the kind of data that is received, which is HTTP
- **Difference**:
  - **IP address**:
    - It is a numerical value assigned to a device/computer. So, it helps to identify a specific device/computer
  - **Port number**:
    - It is a number assigned to a data/application that is executing in a device/computer. Thus, it helps to identify a particular data type or application (such as MySQL, TCP) executing in the device/computer

Hypertext Transfer Protocol (HTTP): HTTP is a member of the TCP/IP family that is specifically designed to transfer mainly text files through the internet.

TCP/IP Model: The Model allows communication over large distances and helps us determine how a specific computer is connected to the internet and how data should be transmitted between them.

Web server: To be able to send or receive data using the HTTP protocol through the TCP/IP network protocol, we need a software that understands the HTTP protocol and helps us send the HTML text from the server computer to the browser requesting the document. The software that receives the HTTP request and handles the delivery of the requested file is called a **web server**.

Understanding the 5 networking layers of the TCP/IP model: The TCP/IP has five layers that are used to manage the transportation of data between computers on the internet. Below are the layers and what they are responsible for

- **Application layer**: This is the layer with which the user usually interacts. This layer allows users to interact with other internet-connected software applications such as the Browser, CyberDuck, Outlook, etc.
    - Each of these internet-connected applications uses data transfer protocol to format the data to be sent. **Example**: Browser/HTTP, CyberDuck/FTP, Outlook /SMPT, etc.
    - The data to be sent starts to be packed in this layer
    - Once the data is formatted following one of the protocols, it then gets passed to the layer that manages the transportation. At this layer, what is going to be sent is called DATA
- **Transport layer**: The transport layer builds on the messages that are received from the application layer. This layer helps in the transfer of data from the source host/machine to the destination host/machine. The main protocol of this layer is called the TCP protocol.
    - TCP is a reliable connection-oriented protocol that transmits data from the source to the destination machine without any error. A connection is established between the peer entities before transmission. At the sending host, TCP divides an incoming byte stream into segments and assigns a separate sequence number to each segment. At the receiving host, TCP reorders the segments and sends an acknowledgment to the sender for correct receipt of segments. TCP also manages flow control so that a fast sender does not overwhelm a slow receiver.

- This layer adds the port number and sequence to the data. **For example**, port 80 will be added to the data if it is going to be sent through HTTP. This port will be later used by the receiving machine to determine which specific application is responsible for handling the request. In this case, it will be received by a web server application
- **Segment**: The data that is sent at this layer is called a segment.
    - **Segment** = Data + (port and sequence information)

■ **Network interface/access layer**: This layer helps you to define details of how data should be sent to the receiving device using the network. The network layer includes the powerful Internet Protocol (IP), and it adds the source and destination IP addresses to the data. This layer responds to service requests from the transport layer and issues service requests to the data link layer.

- **Packet**: At this layer, what is going to be sent to the receiving device is called a packet
    - Packet = segment + IPs

■ **DataLink layer (Ethernet)**: This layer adds the sender and/or receiver's MAC addresses to the data. It also adds some error-checking mechanisms to be used by the receiver. At this layer, what is going to be sent is called a frame

- **Frame** = packet + Mac address

- **MAC (Media Access Control) address**: It is a unique 48-bit serial number assigned to a network card (circuitry) of any device connected by internet or ethernet. A MAC address identifies a device from every other internet-connected device.
- **MAC address vs IP address**: Remember we said earlier that every device that is connected to the internet has a unique IP address. We also said that IP address is the address of the router and that multiple devices connected to a single router

might have similar IP addresses. Because MAC address is unique to any internet-connected device, the router can manage each device connected to it using their unique MAC address

- **Physical Layer**: The physical network layer specifies the characteristics of the hardware to be used for the network. Basically, the cables. This is where the frame leaves the sender's machine.

2.4. How does a local dynamic web page work?

Dynamic web page: We have defined dynamic page earlier under section 2.2. Simply, a dynamic web page is a page that contains content that is dynamically generated during run time. When we covered a static page earlier, we mentioned that a static page is an HTML text document with the content already written on it. But in the case of a dynamic page, the content on the page is not fully written and tagged as HTML yet. Rather, it is generated when the browser requests the document.

How logic/scripts determine the page to be generated in dynamic pages:

- **Backend developer pre-writes conditions**: In a dynamic webpage, the content that is going to be sent to the browser and displayed is determined based on multiple conditions that are included in the file to be requested. The conditions are pre-written by the developer who builds the page. It is the developer who determines what needs to be generated based on the conditions of that webpage. In technical terms, these conditions are called **the logic behind generating the content of that page**.

- **Backend programming language to communicate with the computer/server**: To be able to generate the content following logic, we need to use a programming language that can directly communicate with the processor of the computer. Programming languages that have a compiler to help them directly communicate with the computer are called Backend programming languages. The most common ones are: PHP, JAVA, C++, JS, etc...
    - **For example**, if we want to write a PHP program on our computer, we also need to install the PHP compiler on the computer to help us translate the PHP syntax into something the computer understands
    - Just like we use the .html extension to determine the type of an HTML file, other programs also have their unique extension that they use to quickly determine the type of the file. For example, PHP uses the extension .php.

Creating a webpage with content that generates dynamically: Please watch the class demo regarding the /clock/time.php

- **Dynamic page using PHP**: Now, instead of creating a static HTML file, let's create a webpage with content that is going to be dynamically generated using PHP.
    - Now, let's go ahead and open this file with our browser. First, it has a different extension than ".html". Therefore, the browser doesn't know what to do with it. Since PHP is also a text file, the browser would just treat it as a text file and display all the text as is. However, for the browser to understand the PHP code and display the correct current time, it needs two things:
        - It needs something that understands a file with a ".php" extension (a web server).
        - It also needs to get the resulting document as HTML. Because that is what it understands.

- **Web server**: Here is where a software known as a web server comes into play. This software knows that the browser needs to receive only an HTML document, and it also understands that the browser doesn't know what to do with the PHP code
    - Usually, a web server and a browser communicate with each other using the HTTP protocol. That means they both follow the standards defined in the HTTP protocol to understand each other. There are multiple protocols in the computer/internet world. For example, the FTP protocol, SMTP protocol, HTTPS protocol, and so on. We have discussed before that any data/application has a port number assigned to it when the data is transported between networked computers. **For example**, the port ID of all HTTP data is 80. **Note**: You can change the IDs/port numbers if you want to. However, if you don't change them, the default values are used. **For example**, the commonly used alternative port IDs for HTTP are port 8080 and port 8888, and the default is 80. When you install a web server on your computer, it defines two things:
        - The first one is the root directory, where it serves the files
        - The second one is the port it is listening to. (Port 80 by default). Whenever there is a request on port 80 of that computer (or the port that is being listened to by the web server), it is this software that receives and handles HTTP requests

- Going back to our PHP page above, for our browser to receive the dynamically generated value of the PHP page and display it correctly, we must do these two things first:
    - The php file needs to be inside the root directory of the web server. We can choose this location depending on the software we use.
    - Our browser needs to send an HTTP request to our web server and ask for the generated document.

        NOTE: If you are using MAMP, you can also set up the root directory for the web server.
- Finally, if the browser sends an HTTP request to the web server and asks for the content of the time.php file, the web server receives the request, checks the extension of the requested file, and determines where to send it to be processed. In our case, it is PHP. Then, the web server passes it to PHP to compile and return the result. Then, the web server receives the result and sends it back to the browser as HTML. The browser then renders the result based on the returned HTML.

How do remote dynamic web pages work? Remote dynamic web pages also work the same way as a local dynamic web page. There needs to be a compiler and a web server installed to be able to generate the dynamic content and send it back to the client that requests the data.

- One additional challenge when a web page is available on a publicly accessible domain is that multiple people can request to access the same page at the same time. This phenomenon is called "concurrent request"
- Being able to handle multiple requests at the same time is very critical, especially when your website has a lot of visitors. One of the ways to determine the effectiveness of a web server is to check its ability to handle concurrent requests.

2.5. Concurrent HTTP requests and synchronous/blocking architecture

How web servers traditionally handle concurrent requests:

- **The Common Gateway Interface (CGI)**: This serves as a middleware/way between HTTP servers and external applications, located on the server, that are responsible for generating web content. When a request for dynamic content is sent from the browser, the CGI method is used to generate dynamic content.
    - **Note**: These CGI resources can be written using any language that can interface with the CGI. (Eg: C, C++, Java, PERL., etc..). CGI is an extension of the HTTP protocol.
- **Why the CGI method is not preferred for concurrent requests**: When the web server receives the request, it passes the control to the CGI application. When a web server passes a request to the CGI, there will be a new process for every request. This means that each time an HTTP server receives a request, it initiates a new process, which is very resource intensive if multiple requests are sent to the server at the same time. To be specific, if the CGI application is written using PHP, then the server needs to run a new PHP on every request. This approach gets very expensive as lots of CPU power is needed to handle many concurrent requests

How modern web servers handle concurrent requests: Different web servers tried different methods to solve this problem of handling concurrent requests. Most implemented the concept of threading.

Note: Threading is possible only if the programming language supports threading in the first place.

- **Threading**: It is a way for a program/app to split itself into two or more simultaneously running tasks. Technically speaking, threads are the virtual code/application that divide the physical core of a CPU into virtual multiple cores. A single CPU core can have up to 2 threads per core, and if your computer CPU is a dual core (has 2 cores), it means it has a maximum of 4 threads.

- **Why threading?** Threads allow the processor to perform multiple tasks at the same time, making the tasks faster. It is because of threading that you can browse the web while you listen to music simultaneously on your computer.
- **How are threads created?** Every time you open an application, it creates a thread that will handle all the tasks of that specific application. At the core of a CPU, there is a single thread (which is a code) that creates another thread and allocates tasks to the newly created thread. A good analogy for this is when you open a new browser tab, a new tab is created from the existing tab. Each new tab will handle what any browser tab does
- **Threads vs processes**: A process is any program in execution. Thread means a segment of a process. Threads are distinguished from processes in that processes are typically independent. Multiple threads, on the other hand, typically share the state information of a single process and share memory and other resources directly.
    - **Example:** The most common web server for Java is called Tomcat. It applies the concept of threading. This is possible because Java supports multiple threading. Tomcat runs 200 threads at the same time, meaning it can handle up to 200 concurrent requests at the same time without delay. If there would be a delay, it would start at the 201st concurrent connection. Running 200 threads at the same time is an ok number for small sites. But it quickly becomes a problem if we are thinking to scale up
- **Thread blocking/synchronous architecture**: Synchronous architecture uses thread blocking, meaning a single thread is allocated to handle one request while other tasks remain idle until the first thread completes the task. This means that when a request arrives from the browser, that request is received by the web server and is allocated to a thread to handle that request. Most of the time,

requests involve getting data from the database, which involves querying the database and getting the data. In such cases, other tasks must wait until a thread gets the data from the database

- **How thread blocking works**: When a thread enters a section of code or method that can only be executed by one thread at a time, that thread locks the section of code. That means other threads must wait until the first thread leaves that section of code. When a thread has the code locked, it's called a **blocking thread** because it is blocking other threads from executing the code
- **Pros of synchronous architecture**: In Synchronous Architecture, the thread that sends the query sits and waits until it gets the data back from the database. This means that it cannot accept another request and is blocked from other requests. This has its advantages as you don't have to worry about any other thing in the meantime. In addition, writing synchronous programs is objectively easier than writing asynchronous programs.
- **Cons of Synchronous Architecture:** One of the disadvantages of this architecture is SPEED. If there are many concurrent requests, it causes a delay. A good analogy for this would be if a waiter refuses to bring out any of the dishes you ordered until each dish was fully prepared. To avoid the delay, we need to add more hardware to handle the requests. Which makes it an expensive approach. Adding additional resources could have been avoided if we could have a way to avoid the time the CPU sits idle waiting for the response.

- Another disadvantage of this architecture is that if one thread fails, it can cause the failure or suspension of another thread.

- **Examples of frameworks with synchronous architecture**: Frameworks like Asp.Net and Rails follow a synchronous architecture by default.
- **Speed, the main problem of a synchronous architecture**: The Main problem that arises from this architecture and in general on web servers is how to improve the speed of serving pages when there are a lot of concurrent requests. Node was created to solve this problem.

2.6. Inception of NodeJS: non-blocking/asynchronous architecture

In this section, we will discuss Node.js. We will not start by defining what Node.js is, as it will be too technical to define it without understanding the very problem Node.js was created to solve. Therefore, our approach will be to explain the problem that existed before Node.js and the technology used behind Node.js to solve the problem

What problem was Node.js created to solve? Node.js was created to improve the speed of serving pages when there are a lot of concurrent requests

- **Conceptual solution for the above problem**: Designing a non-blocking or asynchronous Architecture.

Node uses a non-blocking/asynchronous architecture: Asynchronous programming relies on a non-blocking input operation that is not executed in a hierarchical or sequential order. This means that asynchronous operations can run multiple tasks concurrently on a single thread. This simply means you can serve another client while the previous request is being handled.

How can Node run multiple tasks concurrently with one thread?

- **Event Loop Model**: Even if Node is single threaded, it has borrowed the event loop model from the JavaScript callback mechanism to enable it to do many different things at any given time.
    - The event loop is the heart of the Node.js processing model as it is the one that allows Node.js to handle concurrent tasks with one thread without blocking.
    - An event loop is an event-listener inside the Node.js environment. When the event loop detects those events, it triggers a callback function. In short, the role of this loop is to schedule which task/ request our one thread should be performing at any given point in time using callback functions.
- **Sockets:** This is because asynchronous architecture uses threads with sockets (software that establishes bidirectional communication between a server and one or more client programs.

How requests get concurrently handled in non-blocking/ asynchronous architecture:

- When a request arrives, that request is received by the web server and is allocated to a process to handle that request
- If the request involves querying the database, it sends the query to the database. Then, instead of waiting for the response, it just takes note of the request and goes back to receiving another request.
- The system that tracks the requests is called an event queue
- What makes this event queue effective is that it has a mechanism to keep track of the status of the request sent to the database. When the database returns the data, the queue receives and stores it along with a flag to notify the original thread that the data is ready.

- This avoids the time that is wasted by the CPU waiting for the response

Qualities a programming language needs to have to solve the speed problem while handling multiple requests:

- First, the language needs to have a good compiler as it is going to be used on the server.
- Second, the language needed to be able to handle a non-blocking input/output request and to run asynchronously.
- Third, the language should be a language that most people were already using.

  Note: Most languages that were popular at the time already had a good compiler. But the problem was that their architecture was mostly synchronous/ blocking. **Example**: Languages like Ruby and Python wouldn't be a good fit because they use synchronous/thread blocking architecture

Why JavaScript chosen to solve the speed problem:

- JavaScript could use the non-blocking/asynchronous architecture.
- JavaScript was already known by many people.
- **Drawback of JavaScript**: The only drawback for JavaScript at the time was that JavaScript did not have a good compiler that was ready to be deployed on a server.
- **Development of the JavaScript compiler**: Before Node.js was created, JavaScript codes were only executed by browsers (because of the well-developed JavaScript compilers browsers have). Before Node, there was no way to create a dynamic website on a server using JavaScript. This was because JavaScript was not understood by servers (which are computers), and there was no way that JavaScript instructions were to be used to instruct our servers.

Therefore, the solution was to create a compiler that could translate the JavaScript instructions for the server computer. Having such a compiler meant that JavaScript could be used to write server-side applications with access to a computer's operating system, file system, and everything else required to build fully functional applications

- **Refreshing your knowledge on compilers/interpreters**: Computer programs are written in human-understandable languages called High-level languages. However, instructions given to the computer in any high-level language cannot be understood by the computer directly. This is because the computer understands only the language of 0's and 1's (binary code). Thus, the code written in any of the high-level languages needs to be converted to a machine-understandable code called **machine language** for your computer to understand it. That is why we have compilers and interpreters, computer programs that convert a code written in a high-level language to machine language.
    - **Compiler**: A compiler is a computer program that converts the human-readable code into a machine language code before executing it. This means that the execution of the code/ instruction happens only after the entire instruction is compiled/ translated. In the case of compilers, all the errors are shown at the end of the compilation, and the entire script will not run until the error is resolved
    - **Interpreter**: An interpreter works like a compiler. However, in the case of an interpreter, the translation takes place while the code is being executed. Thus, each line of code is converted to machine language, which is then understood by the computer and then finally executed to give us the desired output.

Meaning that during interpretation, errors are displayed line by line. The script/code runs until the error is found and proceeds further on resolving

- **V8 engine made to compile JS code outside of browsers**:
  - Ryan Dahl (creator of Node.js) took out the V8 engine, the fastest JavaScript compiler, from the Chrome browser and created another software that can execute JavaScript outside of the browser. He wrote that application using C++ and called it **Node.js.**
  - Ryan took already existing C libraries and included them in the Node.js package along with a way for JavaScript developers to access the C libraries using JavaScript. This allowed JavaScript developers to access the file system, communicate with the network, etc., on top of what was already provided by V8. He included these additional modules inside of the core Node.js application and released it to the public
  - **Note**: While V8 was used inside of a browser, there was no need for JavaScript codes to access the operating system or the file system of a computer. Until Node.js was created, the main purpose of JavaScript was to just manipulate the DOM object. Since the DOM object was already provided by the browser, all was well, and no one bothered with the file system or the other things that other programming language compilers handle
- **So, what is Node.js:** Node.js is just a runtime environment. To be very specific, it is a JavaScript runtime environment.
- **What is a runtime in programming?** It is an environment where your code/application will be executed/run.

- **Node.js as JS runtime environment**: Basically, this means we will use Node to run our JavaScript code, just like we used browser's runtime environment to run our JavaScript code. For a long time, JavaScript code could only be executed in a browser and was used exclusively for creating front-end applications. Before Node, backend applications used to only be created using languages, such as Java or PHP, that could run on a computer (WITHOUT a browser). Thanks to Ryan Dahl, in 2009, the Node runtime environment was created for the purpose of executing JavaScript code without a browser, thus enabling programmers to create full-stack (front-end and back-end) applications using only the JavaScript language
- **Single thread, the main drawback of Node.js for multiple requests**: Node.js is considered single threaded because by default any JavaScript code runs on a single thread. Since Node uses only one thread to handle requests, it comes with its drawback of delay/slowness
- **Node.js shouldn't be used for CPU intensive applications**: Initially, Node.js was created to handle multiple tasks (such as webpage loading) with speed using a single thread but by applying an asynchronous processing. However, Node was not created with the assumption that it would be used to handle concurrent CPU intensive tasks. When a time-consuming task is received by the server (such as sorting, computing, calculations, graph …), the request must be processed by the CPU/thread itself. We know that Node is single threaded. That means Node will need to wait for the thread to finish the sorting/computing request, then send it to the compiler, receive it from the compiler, and display the response in the browser.

While this single thread is processing this CPU intensive request, if a simple request, such as loading a website, comes, this request will have to wait until the thread finishes the CPU intensive task. This will cause delay

- Therefore, we should avoid using Node for CPU intensive projects. Node is effective for applications that send and receive a lot of data from external services like Database

Getting started with Node.js:

- **Installing Node**: Before installing, check if you have already installed Node by typing the following command on your terminal: *node –version*
  - If the command returns a version number, it means you have Node installed already.
  - If you don't have Node downloaded and installed, go to the link below to download and install the latest stable version: https://nodejs.org/en/
- **Running your JavaScript code on Node**: Remember, Node is just a runtime environment to compile JavaScript. The code that you write is just regular JavaScript. The only difference is that instead of running the code on the browser, like we were used to so far, we are now going to run the code directly on the computer using the Node compiler.
- **Node Package Manager (NPM)**: Let's try to understand the three components of NPM to understand NPM. **The three components of NPM are**: website, registry, and Command Line Interface (CLI).
- First and foremost, NPM is an online repository (website) for the publishing of open-source Node.js projects. Using this website, you can find packages, view documentation, and share and publish packages NPM's official website is:
  - https://www.npmjs.com/

- Second, NPM is a CLI that helps in interacting with the NPM for installing, updating, and uninstalling packages and managing dependencies. It is a command line utility for interacting with said repository that aids in package installation, version management, and dependency management. This is the command line that helps in interacting with the NPM for installing, updating, and uninstalling packages and managing dependencies.
- Third, NPM is the world's largest software registry with over 800,000 code packages. It has a website to look through these registered packages.

How do we use NPM?

- **Installing NPM**: The first thing we need is to have it on our computer. NPM comes with Node.js. So, if you have downloaded and installed Node.js, it means that you already have NPM in your computer
    - **To check if you have NPM installed,** type on your terminal
        - npm –v
        - If the command returns a version number, it means you have NPM installed already
- **Package.json file**: This file typically resides at the root directory of a project and represents various metadata/information relevant to the project. The relevant information includes the name of a package, its version, description, license, and keywords.
- **How to create a package.json file**: Go to your project and open your terminal, then run the following command:
    - npm init
- **Basic NPM commands**:
    - **Installing packages locally**
        - npm install packageName --save-dev
        - **Example**: npm i express –save

- **Installing packages globally**
    - npm install -g package_name --save-dev
- **Updating packages:**
    - **Updating a specific package**
        - npm update package_name
    - **Updating every package**
        - npm update
- **Uninstalling packages**
    - npm uninstall package_name
- **Installing from package.json**
    - npm install