Introduction to NodeJS
Week 1 - Day 1

Topics Covered

1. Introduction to Server-side Development
2. Full Stack Web Development

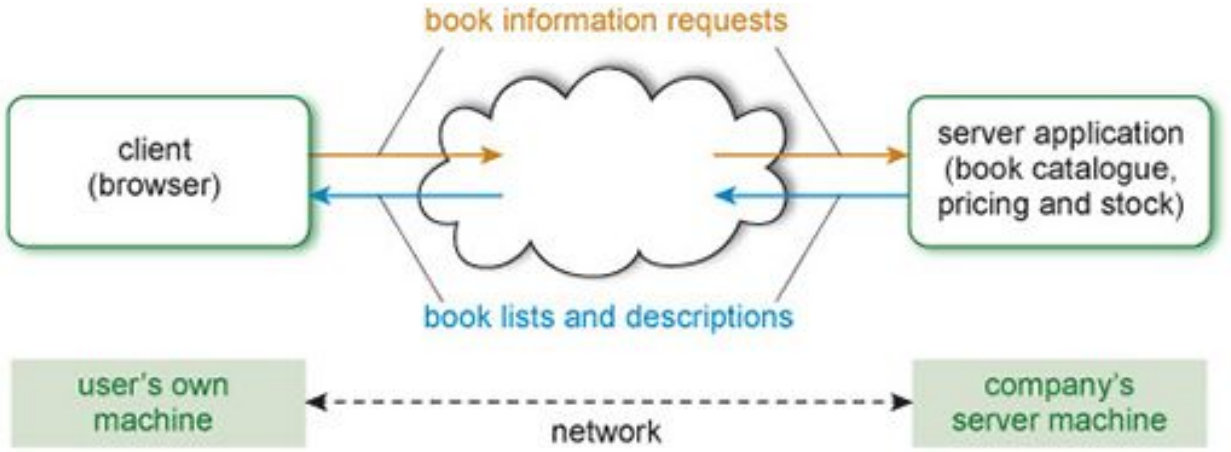
Introduction to Server-side Development

What exactly is server-client architecture, and why is it needed?

A client-server architecture divides an application into two parts, 'client' and 'server'. The server part of that architecture provides the central functionality: i.e., any number of clients can connect to the server and request that it performs a task. The server accepts these requests, performs the required task and returns any results to the client, as appropriate.

Consider an online bookstore as an example. The application allows a user to search and look at the details of a large range of books, and then to order a book. The application software provides an interface and a means of selecting or finding a book's details, as well as displaying book information and allowing a book order to be generated.

The application could take the form of a single 'chunk' of software downloaded from the web. However, if the software is one monolithic item, then every time anything is changed or updated, the entire application has to be redistributed again. An improvement might be to split the application into two parts. One part, the client, can provide the interface for users and be distributed to them. The other part can be kept and run on the company's own server machine. The client application can display information and be used to pass information to the server for searching, such as the title of a book. This simple client-server architecture is also commonly called 'two-tier architecture'.



The catalogue of book information can be held centrally on the server and then be easily updated. This allows other 'centralised' information to be maintained and sent to clients, such as the stock level of each book. Users of the client will find it much simpler and smaller to work with than the complete application. At the same time, the company will have better control and be able to, for example, monitor usage of the server application itself.

There are also different distributions of functionality across a two-tier architecture. For instance, suppose you have a client that accesses your bank account online. If that client is a web browser, for example, it can be used to request and display your accounts' statements from the bank's server. The information you may obtain is restricted to the pre-defined views of the information provided by the server. So, while the server's information might include your account balance, if you want to find out the total payments in and out over the last week or year, then you will still have to calculate it yourself, based on the figures the server provides. Alternatively, you might access your bank server over the internet using another, 'more intelligent' client, such as a mobile app. This client might include a facility to extract figures from your bank statements and to perform whatever calculations you require. Such a client might also create bar or pie charts that display your income and expenditure across different categories that you define.

The web browser with little functionality of its own is often termed a thin client while the more intelligent client is usually termed a thick (or 'thicker') client.

The client that is distributed to users may change, while the server part can be a centralised component that maintains dynamic, global data in a consistent and secure way for the organisation and for users to access and use.
(Source)

Other interesting reads:

➔ https://spin.atomicobject.com/2015/04/06/web-app-client-side-server-side/

Full Stack Web Development

Difference between front-end and back-end development

Meaning of Frontend vs Backend

The frontend is the part of the website users can see and interact with such as the graphical user interface (GUI) and the command line including the design, navigating menus, texts, images, videos, etc. Backend, on the contrary, is the part of the website users cannot see and interact with. It's all about how everything works.

Role of Frontend vs Backend

Both play a crucial role in web development and although they have their fair share of differences, they are like two sides of the same coin. The frontend is all about the visual aspects of the website that a user can see and experience. On the contrary, everything that happens on the background can be attributed to backend web development. It's more like an enabler for frontend web experience.

Developer of Frontend vs Backend

Web designer is the most common job title for the frontend web development and the role of a web designer is to design and rebuild websites keeping the visual aspects in mind. Backend developers are the ones who make sure the data and systems requested by the frontend application or software are efficiently delivered. Backend developers handle everything that happens in the background.

Essentials of Frontend vs Backend

Frontend is also referred to as the "client-side" as opposed to the backend which is basically the "server-side" of the application. The essentials of backend web development include languages such as Java, Ruby, Python, PHP, .Net, etc. The most common frontend languages are HTML, CSS, and JavaScript.

What is a Full Stack Developer?

A full-stack developer is a web developer or engineer who works with both the front and back ends of a website or application—meaning they can tackle projects that involve databases, building user-facing websites, or even work with clients during the planning phase of projects.

Full-stack web developers:

- Are familiar with HTML, CSS, JavaScript, and one or more back end languages.
- Most full-stack developers specialize in a particular back end programming language, like Ruby or PHP or Python, although some, especially if they've been working as a developer for a while, work with more than one.

As the line between what can be done on the front end versus back end increasingly blurs, more developers are becoming what we call "full-stack." A lot of employers (especially agencies who work on different kinds of sites) are looking for developers who know how to work on all the parts of a site, so they can use the best tools for the job regardless of whether it's technically "front end" or "back end." Hence the rise in companies listing full-stack development on job requirements.

Now, contrary to what a lot of people think, "full-stack" doesn't necessarily mean a developer is writing all of a site's code themselves. Many full stack web developers spend the majority of their time in either the front or back end code of a site.

But the point is that they know enough about the code across the entire stack that they can dive in anywhere if needed. And some full stack developers code entire websites, including both the front and back end, but usually only if they are working freelance or are the only developer working on a project.

Some interesting reads:

- ➔ https://medium.com/@sagarajkt/the-fundamentals-of-front-end-and-back-end-development-5973ac0910cf#:~:text=Backend%20developers%20handle%20everything%20that%20happens%20in%20the%20background.&text=Front%2Dend%20is%20also%20referred,Net%2C%20etc.
- ➔ https://skillcrush.com/blog/front-end-back-end-full-stack/
- ➔ https://medium.com/@alexkatrompas/the-hard-truth-about-the-full-stack-developer-myths-and-lies-945ffadeeb8c
- ➔ https://hackernoon.com/6-essential-tips-on-how-to-become-a-full-stack-developer-1d10965aaead