Front-end and Back-end

Front-end

We have a *client* and a *server*. The client is the **browser**, and the server is the **webserver**.

- The client can also be an application too!

Client sends *requests* and the server gives back *responses*.

- The part on the browser is called **front-end**.
  - That is what the user *sees*.

It is about the **presentation**.

- How things look.
- Images, content, structure.

On the web, it is written by using HTML, CSS and JS.

Back-end

Back-end is about *how things work*.

- What are you going to present in the front-end?

Business Logic and Data are on the back-end.

- Database is here.

When a user wants to add something to the website, the form is sent to the back-end.

Generally the used languages are PHP, Ruby, Python, Java, .NET, etc.

Website

Let's open a website and go through stuff to see what is what.

Frontend

All about Presentation!

- Not only the web. What about mobile applications?
  - Java, Kotlin for Android
  - Objective C and Swift for IOS
  - What else?

Although these are not generally called as front-end, they are.
The difference is, we are using a different language now.

- All the user interface we see is *rendered directly on the device of the user*.

Frontend programming

- While building an application, there will be design teams.
  - UX, UI
- This design will come to front-end developers and they will *code* (realize) these images.
  - By using **html, css and js**.
- If you want to be a front-end engineer, you should be very good at CSS and JS.

Frontend programming > js

- Javascript is a *client side* language.
  - We can see the javascript code in a website by checking the source.
- The code works on **our** computer, not on the server.
- There are different libraries of Javascript.
  - The original one is called *vanilla* javascript.

Frontend programming > js

- Instead of using vanilla JS, many developers and systems are using different libraries.
- The most common library is React by Meta.
  - In the old times, there were JQuery, CoffeeScript, etc.
    - It is still being used. Because **maintenance**.
  - Angular by Google has a user base.
  - VueJS is also very popular.
- However, React is the golden standard.

Frontend programming > js

- There is also **Typescript**. It is a strongly typed programming language that builds on Javascript.
  - You write in Typescript, it compiles to Javascript.
  - Helps write better code.
- If you want to be a front-end programmer, you must improve yourself on Javascript.
  - Try to start with *vanilla js* and then move on to libraries like React.

Frontend programming > css

- Front end programming is not only about Javascript. CSS is also very important.
  - If you want to be a good front-end programmer, you should improve yourself on CSS.
  - Do not rely on simple tutorials and try to move beyond them.
- There are interviews done on Turkish on Youtube about front-end development. Watch to see what they want.
  - LinkedIn Jobs is also good to see what companies want.
- There are not much front-end developers in Turkey.

Frontend programming > mobile

- Mobile development is mainly front-end.
  - The business logic is *backend* and it is not written specifically for Mobile.
- However, *mobile application* still needs to *look good* and *reach* backend data.
- This is what mobile programming is.
  - Apart from gaming of course.
- Therefore, jobs on mobile development deals with the *presentation* too.

Frontend programming > mobile

- It is not *mandatory* to create mobile applications with the native languages.
  - React Native
  - Xamarin
  - ..
- You can code in React(Js), C# and make it multiplatform.
- This is good for small companies or personal stuff.
  - However, enterprises *usually* go for native languages.
  - So it would be good for you to learn them and improve yourself on them.

Frontend programming > good to know

- CSS Pre-processors such as Sass and Stylus.
- JS Libraries and Frameworks (Lodash; Angular, React, Vue, etc.)
- Build tools (npm, Webpack, etc.)

Frontend developers are not going to be involved with server-side languages, database queries, server configurations or server-side business logic such as *user auth*, *order handling* etc.

Backend programming

Called *backend* because it is working in the *back*; not on our computer.

- Remember that frontend *runs* on our computer, our browser.
- Backend on the other hand runs on a *server* which we call a **webserver**.

This server can be on the *cloud* or it can be on our own computer.

Backend programming

For example, when we are using an app, the **data** is not on our computer. It resides on a different computer and we *see* them by pushing/clicking on the *frontend*.

- In other terms, data is in the backend, and we use *frontend* to reach that data!
- Backend is not only about *data storage*.
  - **Logic!**

Backend programming

There are some things that we should not do on frontend.

- Interactions with the file system.
- Server-side input validation
- Database interactions, etc.

Frontend should only contain code that is related to the *presentation*. It should *fetch* data, *send* data and *present* data to the user.

Backend programming > languages

You can use different programming languages for backend. Almost all languages are suitable, but some are better equipped.

- PHP: A web programming language. Designed for this task only. It was very popular (still is), but nowadays people who learn web programming are not learning PHP.
- Python: It is not a *web programming* language but there are *frameworks* to help build web applications such as **Flask** and **Django**. Very popular especially with Startups.
- cont.

Backend programming > languages

- Java: One of the most popular languages used in web. Has a very good web framework called Spring (Boot).
  - Many large applications are using Spring Boot.

Although it is possible to make a web application using *C* or *C++* it is not a standard. We generally use scripting languages or compiled languages like Java and C#.

- Javascript can also be used as a backend programming language with *NodeJS*!. Very good and popular for *fullstack* development. NodeJS let's us use Javascript outside the browser and *Express* framework helps us develop web applications.

Backend programming > good to know

- Server-side languages & frameworks (PHP, Python, Java, etc.)
- Databases & Query Languages (SQL, ORM, etc.)
- **Basic** frontend knowledge
- Server configuration

Backend programming > works on

- Data validation (security)
- Notifications
- Business Logic (User auth, etc.)
- Data storage, database access
- Scheduled processes (cronjobs, etc.)

Does not care about advanced CSS or Javascript, build tools, etc.

They will not work on client-side stuff in general.

Fullstack development

In big enterprises or applications, *frontend* and *backend* are separated.

- Frontend developers are backend developers are different!
- This helps in terms of expertise.

If these are not divided *or* if you are doing both the *backend* and the *frontend* it is called **fullstack development**.

- You are responsible for both frontend and backend.

It is good to start with fullstack when you are learning. Learn some CSS and JS and a backend language to build a website.

- You can always use CSS frameworks like Bootstrap to help you.

Communication

The frontend and backend are communicating with each other.

- By using HTTP requests.

Frontend sends data to backend. Backend *validate* that data, store it in the database or do some additional stuff and saves, etc it.

- BFF

Going online.

What happens when we write a url and press enter?

Going online

This is a very simple example which can be further evaluated. We will talk about what happens behind the scenes when we are trying to connect to a website.

People may have different answers to this question:

- A dba may only care about what happens to the database.
- A backend engineer will care about the webserver.
- A frontend engineer will care about the presentation and the communication between fe & be.
- A devops may care about the CI of the system
- A mechanical engineer may even start with the keystroke!
- ...

Going online

The Internet runs on TCP/IP.

Browser is the client. It talks with a server. It is the same with a phone.

We use URL to connect to these websites. URL: Uniform Resource Locator

- It has a protocol: HTTPS
- Has a domain name:
- And *path*

Going online

We know that the URL needs to change into an IP address.
By doing a simple `ping`, we can learn the IP address.

- `ping google.com`

DNS: Domain Name Servers

- Yellow pages for the Internet

While pinging, we don't care about the path or the protocol. You know the main number. It's like calling the school line. Path is the room number.

- You can ask to be connected to somewhere. They have the information for that.

Going online

The purpose of `ping` command is to see whether a web site is *up* or not.
However, it is the shortest way to learn an IP address.

We can use `nslookup` command to get that info too. We will see more. The first *server* is the DNS server.

- If you did not specify any DNS server, the default is your router output.
- We can use `server 8.8.8.8` command and check the name again.
    - We will get the same result.

Going online

So, when we try to enter a website, the first thing that's happening is *DNS Lookup*.

- That converts the URL into an IP address.
- Sometimes that IP address will be *cached*; so that we don't lookup the same thing over and over again.

We can also learn some information about the website by using `whois` command.

- This should be installed on your computer.
- Or you can check via website.

DNS

Actually, the browser first checks it *cache*.

- If there is no cache, it asks the operating system.
  - Remember *hosts* file.
- Then we do the DNS lookup.
- ISPs have their own DNS servers.
  - We connect to them by default.
  - But we can change them to OpenDNS, GoogleDNS, etc.

Going online

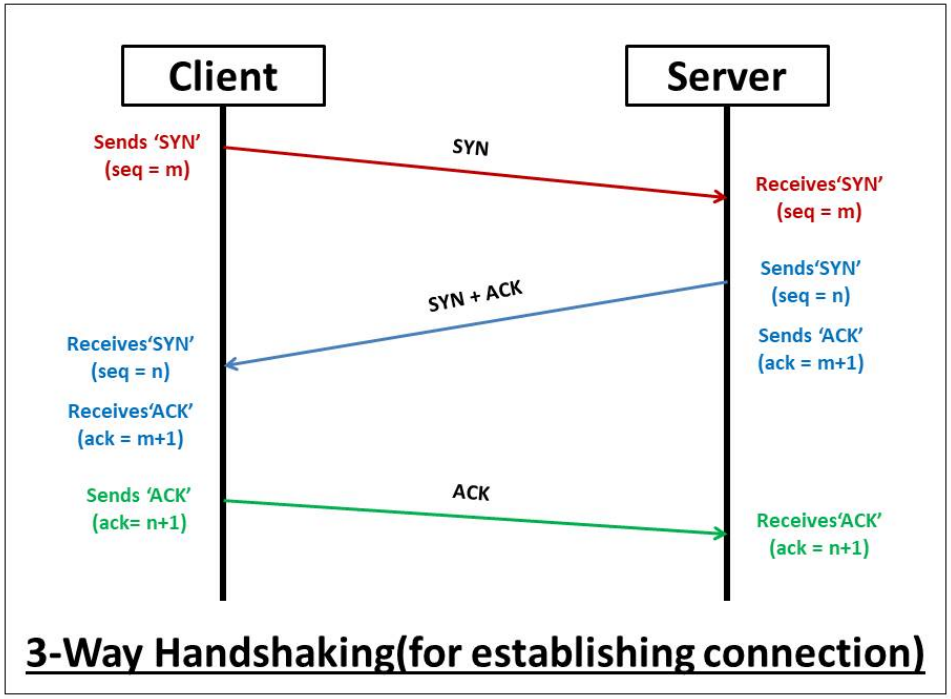
After getting the IP address, browser will do *Three-way handshake*.

- It will go and talk to a specific port.
- The website name is the *building* and these ports are just different doors.

For example, if you are using HTTP, you use the port number 80. If you are using HTTPS, port number is 443.

- (B): So, browser talks to 443 and asks to synchronize with it. (SYN)
- (S): Sends back SYNACK.
- (B): (ACK)

Going online



Going online

Browsers work on Application Layer (Remember the OSI model.)

At this point we have seen that there are a lot of stuff going on behind the scenes which are somewhat secret to us.

However, we can actually see some of them by looking at our Developer Tools in browsers.

Connecting to a website

Let's open *Network* tab and refresh.
We will see a lot of rows. Each of them is an HTTP Request.

First it gets the website and while trying to show that, it realizes that it needs to load more things such as Js and Image files.

We can also do these things from the *command line*. We can use `curl` command to make an HTTP request/call.

Connecting to a website

E.g. we can get the certificate information of a website `curl -iv https://google.com`

There are some information here too.

You can learn a lot about *curl*.

Connecting to a website

In the developer tools, there is *Headers*.

These are *key-value*.

- There are a lot of information here.
- We see some stuff similar to our curl output.

HTTP codes

When you `curl` to a website, you will see some response codes.

- 200 OK
- 300 Moved to some place
  - 301 Moved Permanently
  - 302 Moved Temporarily
- 400 Error
- 500 Page Not Found
  - My error.

In order to understand this better, we need to try some stuff!

Curl & HTTP response codes

```
curl -vi arel.edu.tr
```

- We get a 302, redirect.

```
curl -vi http://arel.edu.tr
```

- We get a 302, redirect to **https://**

```
curl -vi https://arel.edu.tr
```

- What is the problem?!

Bad config !

Curl & HTTP response codes

Actually, when we try to open this website from the browser, we cannot! It loads another webpage: `istanbularel.edu.tr`

```
curl -vi istanbularel.edu.tr
```

- 302 Redirect to https version.

```
curl -vi https://istanbularel.edu.tr
```

- 301 Redirect (Moved Permanently.) It gives us a version with **www**.

```
curl -vi https://www.istanbularel.edu.tr
```

- Now we get 200 OK!