YOUR PASSPORT TO MODERN WEB DEVELOPER JOB

THE WEB ESSENTIALS

INTRODUCTION

This e-book is a part of a roadmap toward getting or starting a professional modern web developer career. The collection of e-books represent steps you need to follow to achieve this goal.

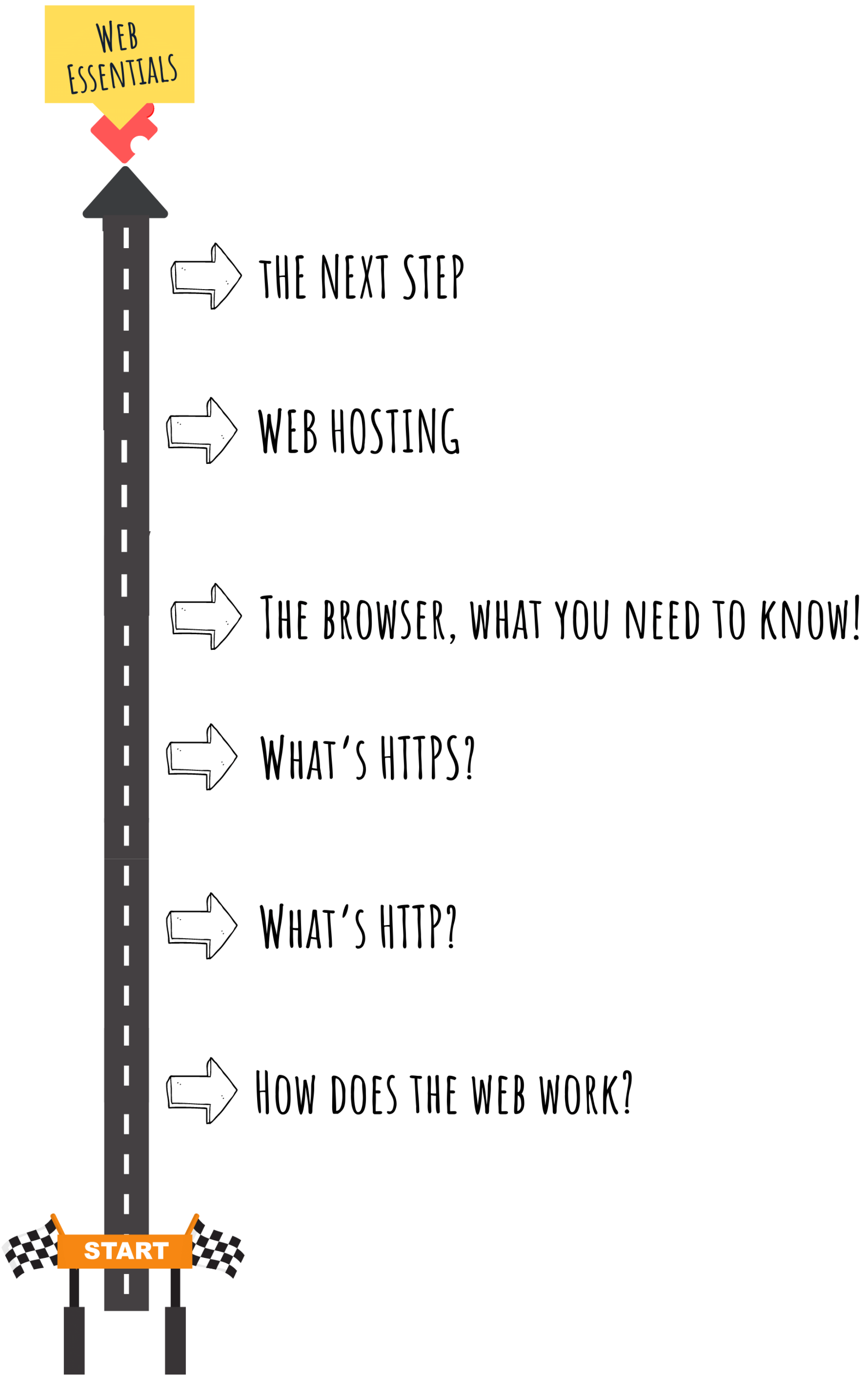
This collection has two main goals. First of all, provide a clear and solid web development roadmap—to fix the lack of a complete roadmap to web developer jobs in the universities and schools. I'm a software engineer who studied and completed a computer science master's degree. However, the school and university schedule was still not complete to build a solid knowledge base to work in this field. In addition to this, tech subjects are not following the new trends as the technology field moves constantly. There are new things every day. Secondly, I met many people through social media and other platforms that want to change their job to a web developer because of the market's constant demand for this job or financial reasons. Their problem is that they don't know where to start and don't have a clear roadmap to follow that can guarantee building the right skills for this job. If you are in this case, then this collection is perfect for you.

The collection will ensure that you don't get confused by the immense amount of web content over the Internet by following its clear and complete roadmap. It has all the steps and skills you need to learn in the proper order.

Following the collection step by step will make you ready for a web developer job. At the end of the collection, you'll receive professional hacks and tips e-book to write your first professional tech resume and succeed in your interview.

Of course, I'd love to get in touch with you through social media and help you have a fun and pleasant web developer journey. All you need is your determination. Feel free to get in touch with me with any questions.

Web Essentials Roadmap



I. How does the web work?

Before learning any skill about the web, we first need to understand how the web works. This chapter will clearly explain how the web works to have a clear mental picture for the following information.

The web or the Internet is not about just visiting an URL (Uniform Resource Locator) by entering it in the browser. Many things happen behind the scene, and that's what we will expose to you.

Let's start with a classical website visiting. Let's say we want to go to www.fam-front.com.

Many steps need to happen so that you can see the website. Here are the steps that happen behind entering an URL and hitting enter.

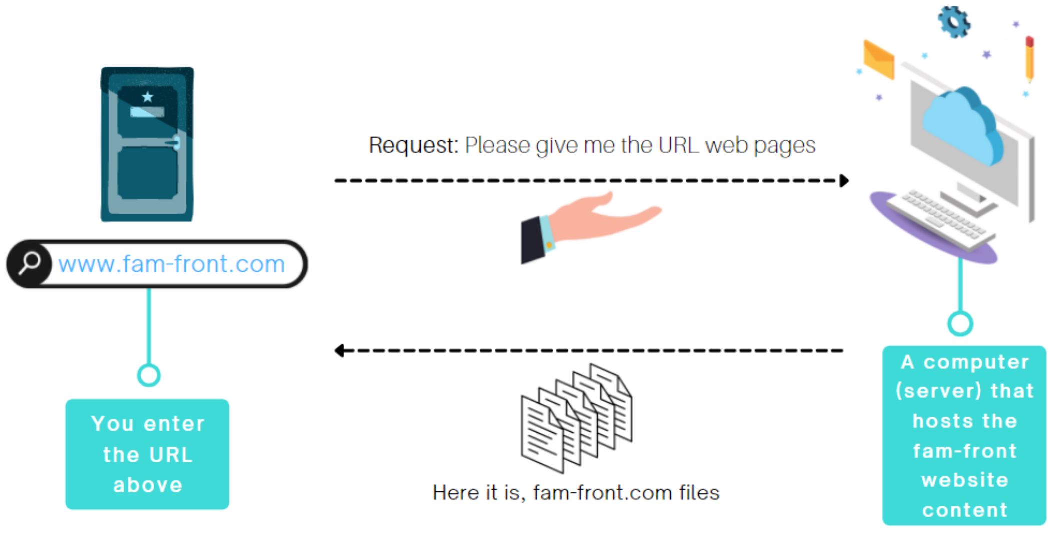
1. The URL, the entry door

The web story starts from an URL. We type a URL, hit enter, and the magic occurs. What happens behind? How is a page on the Internet linked to an URL? How is it built? And much more questions that we'll be answering in this e-book. So, no worries, by the end, you'll have the big picture in mind.

a. Understand why we need an URL on the Internet

Let's see a website as a book. The book you're visiting is not on your laptop or desktop. Since we can call it on the Internet, which means that somehow it is stored somewhere, it is held in another computer so that everyone on the Internet can access it by its entry door, the URL. Because that other computer serves as the book (website), we call it the server.

Let's see a mental picture of the whole process:


www.fam-front.com
You enter the URL above
Request: Please give me the URL web pages
Here it is, fam-front.com files
A computer (server) that hosts the fam-front website content

With:

URL is the door or the **address** to a website

The computer where the website content is stored is called the **server**

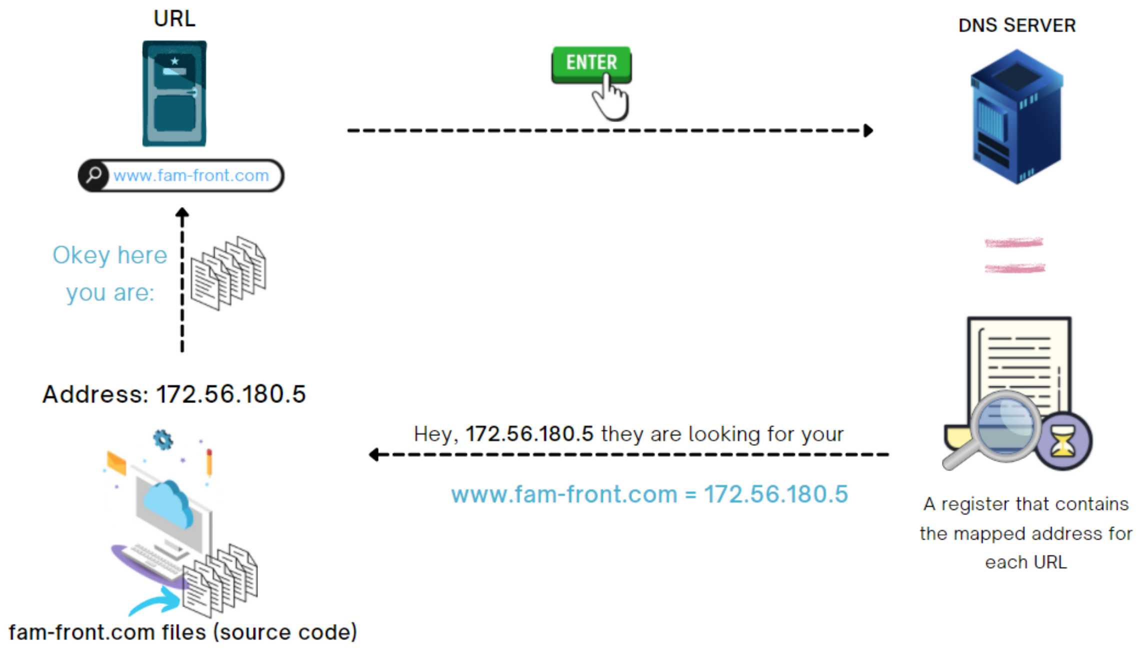
b. Now, how does the server know that he has the files for the entered URL?

Good question, Nah? Well, there are more hidden steps between the user requesting an URL and the server answer the request.

To get the website, you're looking for. There are three main conditions:

- Condition #1: The server hosts our website (a computer), which means our website files (source code) are somewhere on the server.
- Condition #2: The URL has its translated address (the IP address) on the DNS server.
- Condition #3: That IP address is nothing but the hosting server's authentic and unique address. Thus the connection is established.

Let's picture it together:


URL
www.fam-front.com
ENTER
DNS SERVER
Okey here you are:
Address: 172.56.180.5
Hey, 172.56.180.5 they are looking for your
www.fam-front.com = 172.56.180.5
A register that contains the mapped address for each URL
fam-front.com files (source code)

FYI

- **IP** stands for **Internet Protocol**
- Every server (computer) on the Internet has a **unique IP address** (exactly like your home address) so that we can call it when we want.

What's DNS?

When you hit enter to request a URL, the DNS (Domain Name System) gets to work. It's a server, a simple computer that got a register of each URL on the Internet and its corresponding address. Since a computer address is something like: 172.56.190.3, called the IP address.

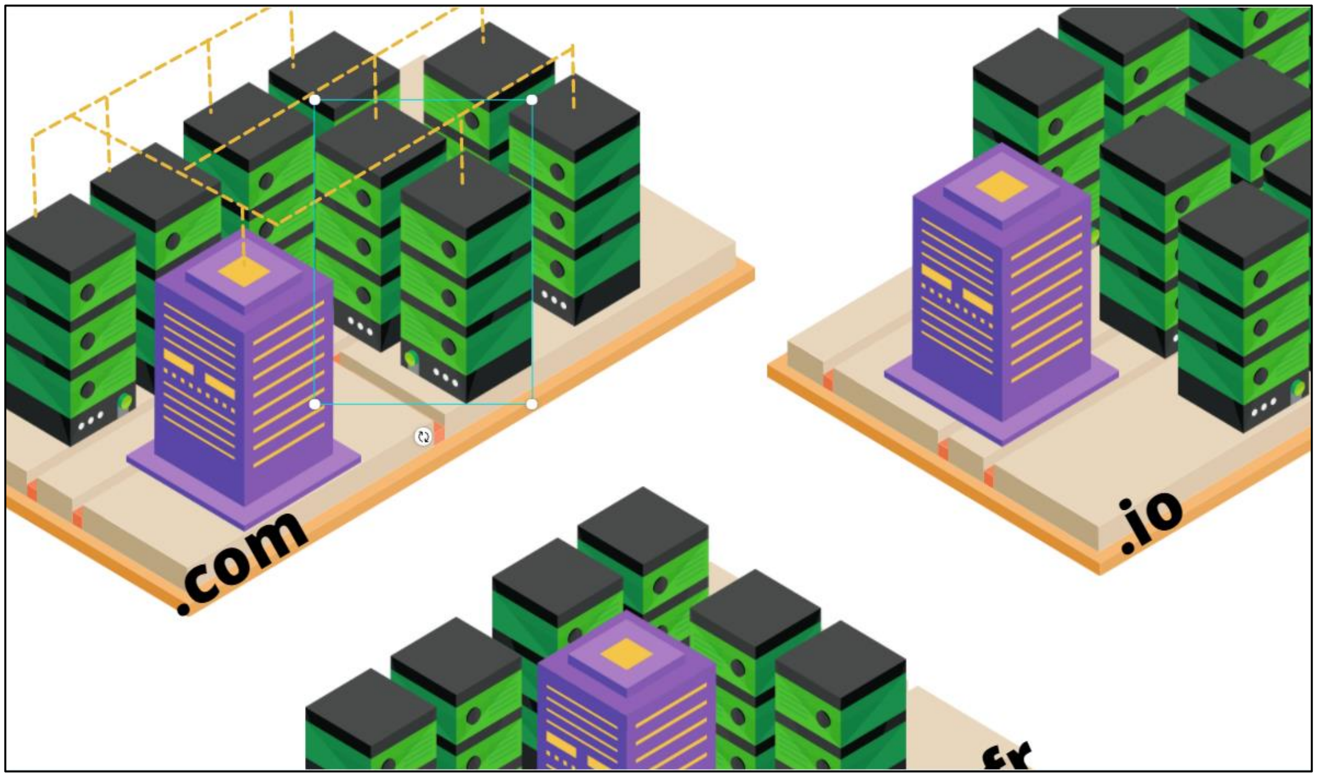
Are you wondering why we need this?

Well, because we are humans. If we don't give human names to our website, no one will remember these complex addresses. That's why we need a sort of register that will link the human address (URL name) to the actual server address (the IP address). Got it?

So, is there one DNS server to contact each time?

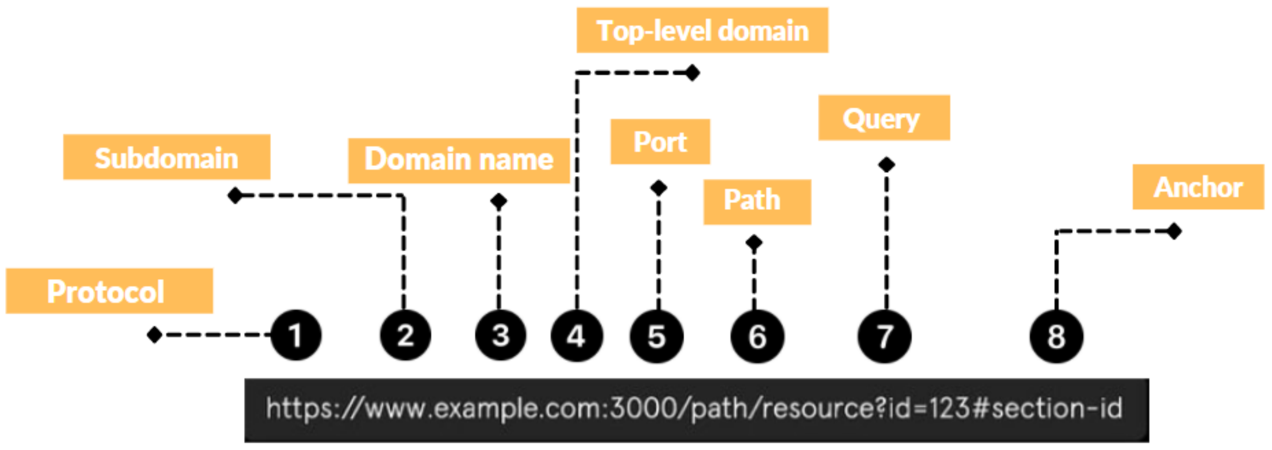
The answer is no. Imagine the whole world connected people request the same server. It's like doing this:

We need multiple servers around the world to dispatch the enormous amount of demands. In each area, there is the master DNS server that the browser contact to get the IP address. After then, the master DNS server can communicate with the servers it leads to getting the IP address.



c. Understand URL composition

Let's now discover the URL composition:



We will learn more about this composition later in the following e-book.

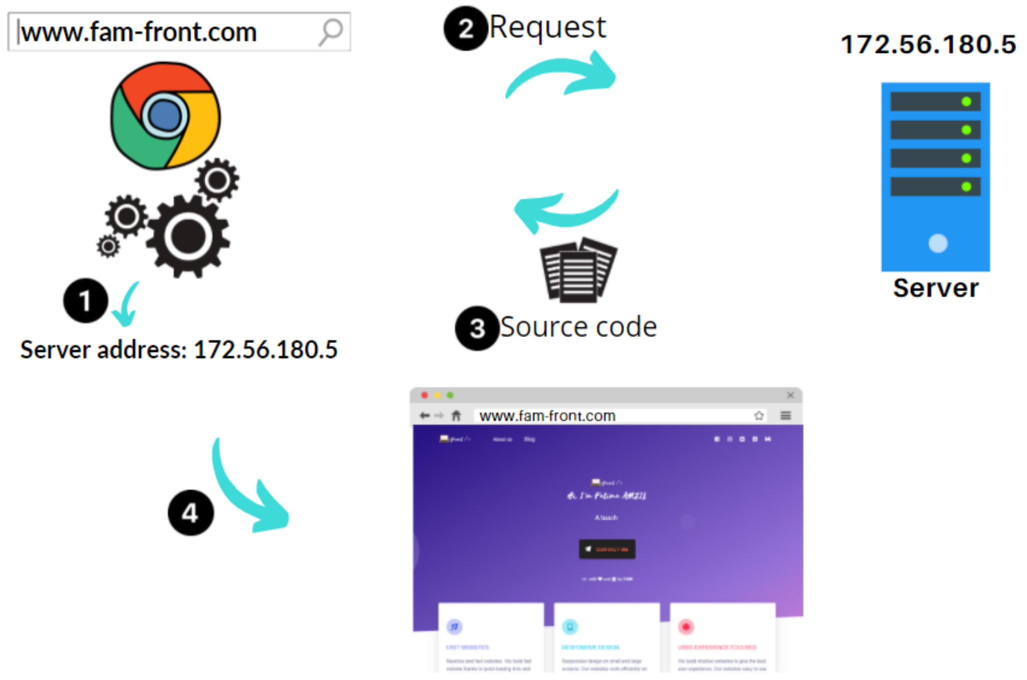
2. Wow, the page is rendered

In the second step, after getting our website source code, we see the page on the browser. So what happens when the browser receives the website files? How does it display it?

The browser plays a vital role in this step. The browser comes with the domain addresses. It already knows the IP address corresponding to the typed URL. After then, the browser goes ahead and requests that server with the found IP address.

Having the IP address means having the location or the address of the hosting server that contains your website code. The browser understands your website's code. It transforms your files: HTML, CSS, and JS files to what you see at the end!

We will get to know more about how things are rendered in the following e-book. So no worries, you are on the right path, step by step, to building global and solid knowledge.



The sent request is not just a simple: 'Hi there, please send me the source code for this IP address". It's a package of data that contains all the information that the server may need to accept our request and send back the website source code.

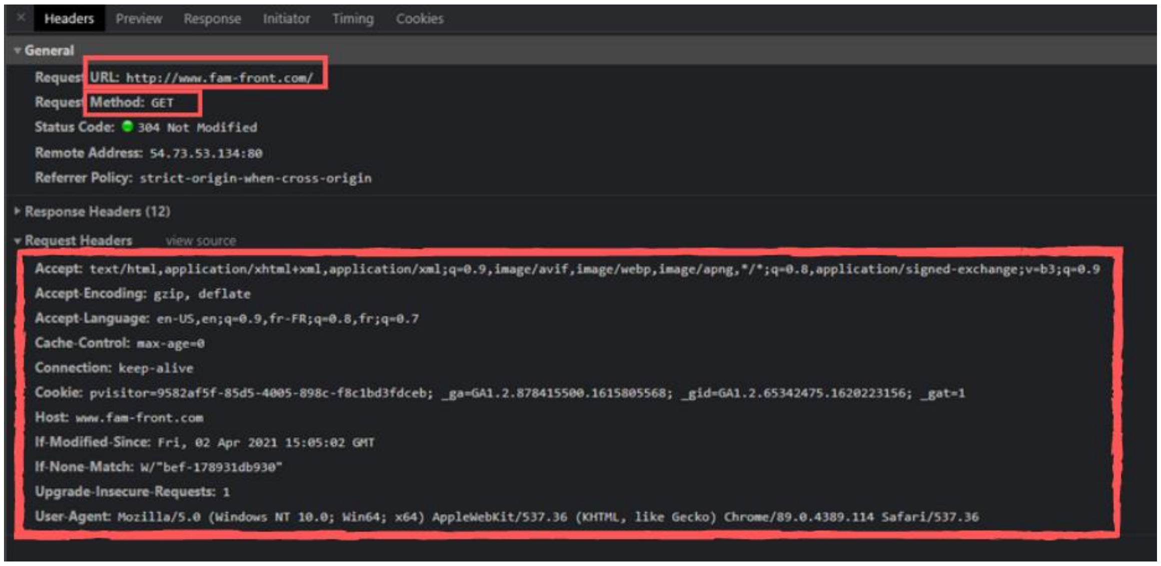
The data are sent via a protocol known as the HTTP (HyperText Transfer Protocol). That's what we are going to learn in the next section.

II. What's HTTP?

To communicate with other people, you speak a language, right? Well, HTTP is the spoken language between websites on the Internet. It is a standardized protocol that defines what a request (and response) has to be. In other terms, it determines how we can communicate data for the World Wide Web. You may be wondering why an URL works the same even without typing the HTTP protocol. Well, this is working thanks to the browser that auto-completes the URL with the HTTP protocol.

So a request or response on the Internet is not a simple question or answer. It is a technical thing that can contain data, metadata, and how a request should be submitted to make it work properly. The same applies to the response. It can be a website like the case of www.fam-front.com or return some data or nothing.

Let's see what a request looks like by checking Chrome's developer tool:



```
General
   Request URL: http://www.fam-front.com/
   Request Method: GET
   Status Code: 304 Not Modified
   Remote Address: 54.73.53.134:80
   Referrer Policy: strict-origin-when-cross-origin
Response Headers (12)
Request Headers      view source
   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
   Accept-Encoding: gzip, deflate
   Accept-Language: en-US,en;q=0.9,fr-FR;q=0.8,fr;q=0.7
   Cache-Control: max-age=0
   Connection: keep-alive
   Cookie: pvisitor=9582af5f-85d5-4005-898c-f8c1bd3fdceb; _ga=GA1.2.878415500.1615805568; _gid=GA1.2.65342475.1620223156; _gat=1
   Host: www.fam-front.com
   If-Modified-Since: Fri, 02 Apr 2021 15:05:02 GMT
   If-None-Match: W/"bef-178931db930"
   Upgrade-Insecure-Requests: 1
   User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.114 Safari/537.36
```

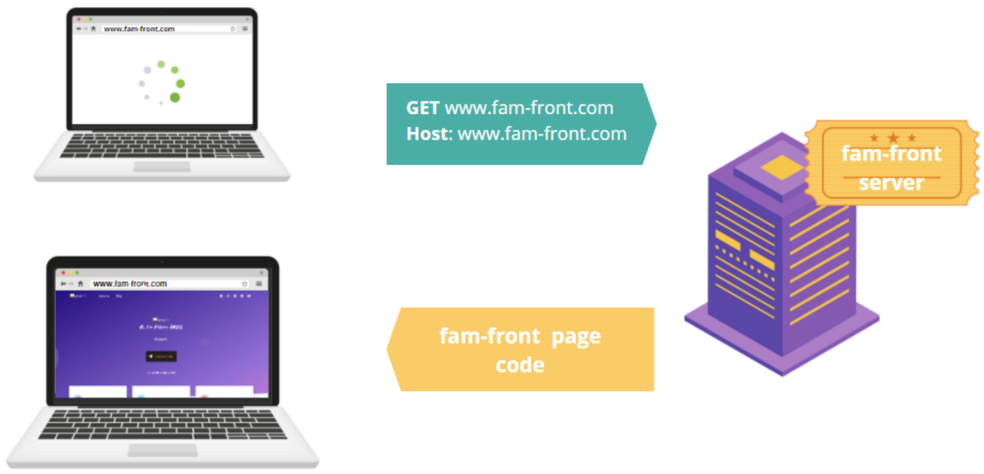
In this example, I requested the website www.fam-front.com. As you can see, a typical HTTP request message contains the following:

- The URL: www.fam-front.com
- The method: GET
- The hosting server
- Request headers

The most important to understand here is:

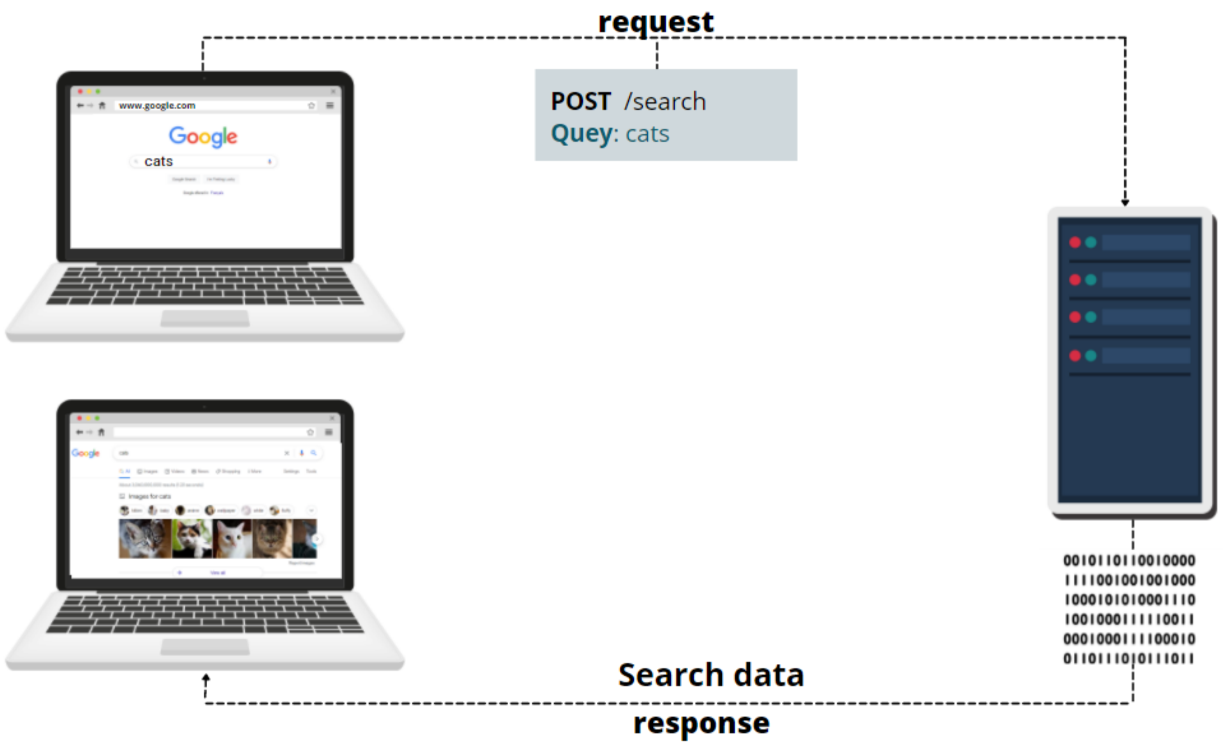
```
GET      Document-name
HOST:    website-name
```

We need to specify the method (ex. GET) and the hosting server where the website is stored. Here is a mental picture to help you keep this in mind:



What if we don't want to "GET" a web page but send our name and password to log in?

The method *GET* used so far is not the only one existing in HTTP. We can use different method's names based on our needs and what we want to do. A concrete example for this case is searching on Google:



Since we want to send what we are looking for to the Google server. We need to specify a different HTTP method which is the POST method. As you can see in the example, we need to give the information (cats) in the request's query. The server processes the request. Search for *cats* and send back the data (the result of the search).

Thant's not the only method we can use. GET and POST are the most popular and the most used ones. But, there are other methods such as:

- PUT: used to update or replace data in the database
- PATCH: used to edit or modify to fix something.
- DELETE: used to delete data.

So if we want to sum up:

- HTTP is the protocol and the spoken language on the Internet to communicate and exchange data between web-based applications.
- HTTP is like a messenger of the web.
- HTTP protocol delivers content like images, files, videos, audios, documents…etc.

- Browsers speak the HTTP protocol and respect it to communicate and display web applications for the user.
- HTTP is connectionless, which means that the client (browser) disconnects from the server after a client request. When the response is ready, the server re-establish the connection with the client to deliver the answer for its request.

Internet is open to everyone. If I can visit a website, you can too. As I showed earlier, the sent request is visible in plain text, which means hackers can use this information against us. That's why a website should be safe and secure on the Internet to prevent a hacker attack. We need to use a secured communication channel that already exists named HTTPS (HTTP Secure).

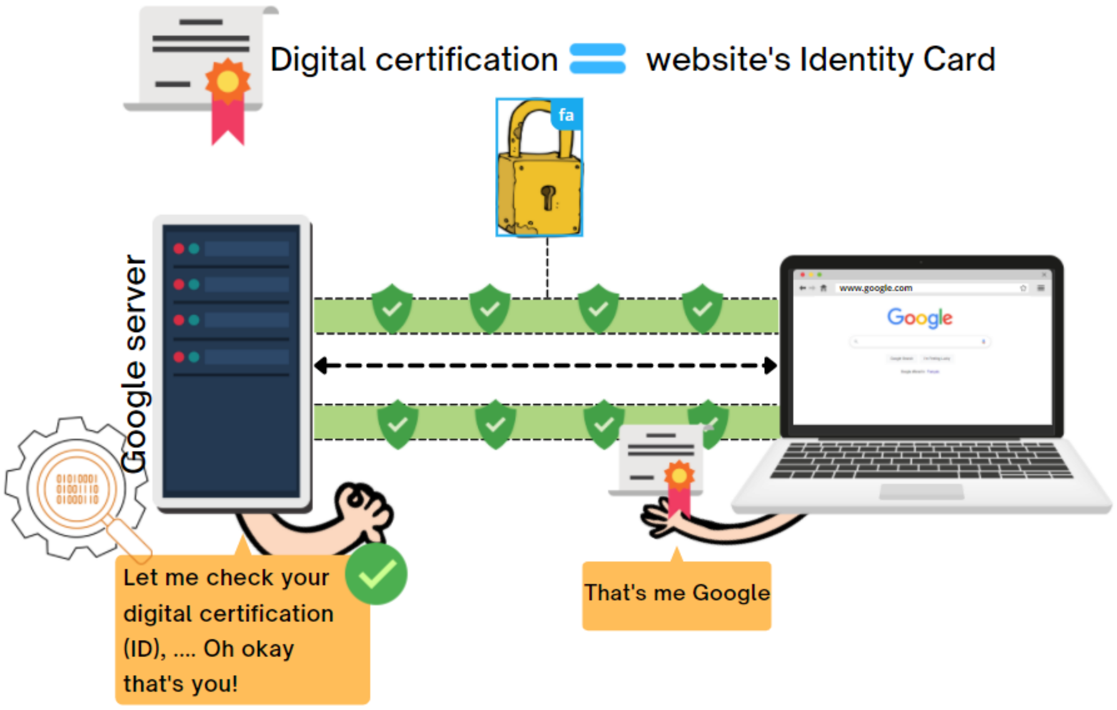
III. What's HTTPS?

To have a secured communication, we need the SSL (Secure Socket Layer) and its successor, the TLS (Transport Layer Security). Think of both SSL and TLS as a wrapper that protects your communication from snooping, tampering, and other hackers' attack.

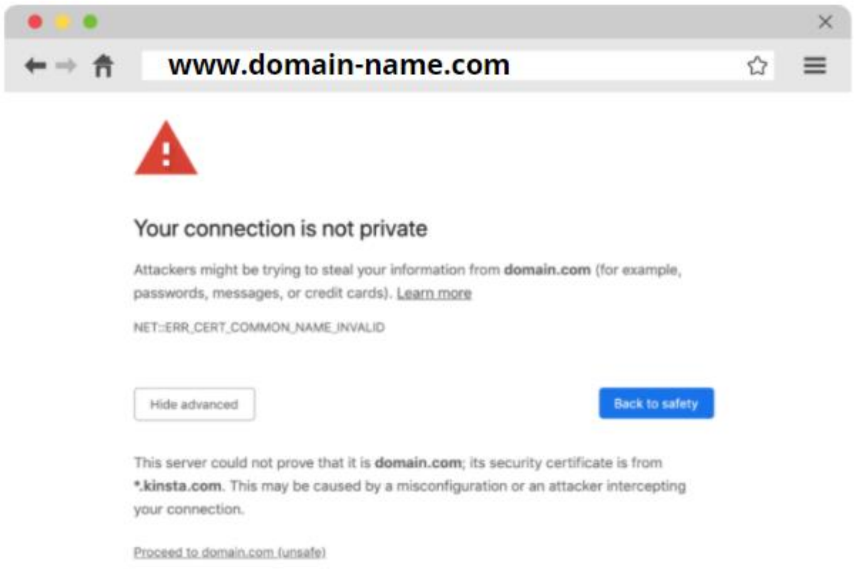
When do we know if a website is secure and that TLS and SSL are active?

Well, you can quickly check that by looking for the little lock next to the HTTPS in the URL:

Thus, the HTTPS protocol ensures that your requests on the Internet are safe and protected.


Digital certification = website's Identity Card
fa
Google server
www.google.com
Google
Let me check your digital certification (ID), ... Oh okay that's you!
That's me Google

The browser asks the Google server to engage a secure connection by providing a digital certification (like an ID card for humans) that proves that the one who wants Google page source code is Google itself. If the certificate is not valid, the browser will warn you with this page that you may have already seen:



Your connection is not private

Attackers might be trying to steal your information from **domain.com** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_COMMON_NAME_INVALID

This server could not prove that it is **domain.com**; its security certificate is from ***.kinsta.com**. This may be caused by a misconfiguration or an attacker intercepting your connection.

Proceed to domain.com (unsafe)

FYI

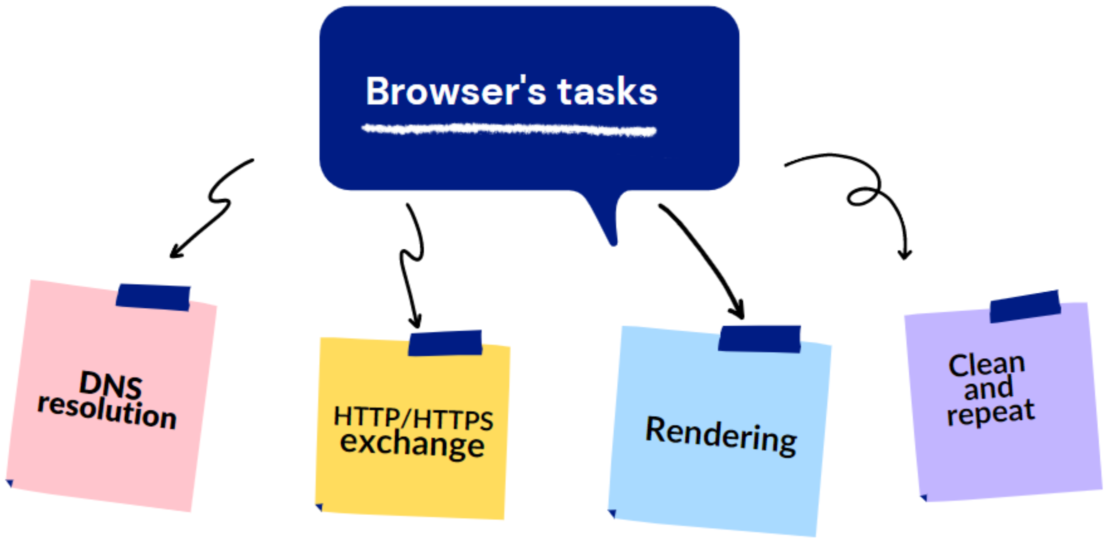
Digital certifications are provided by **certificate authorities**. It's like a government that issues your ID and passport.

IV. The browser, what you need to know!

1. How does a browser work?

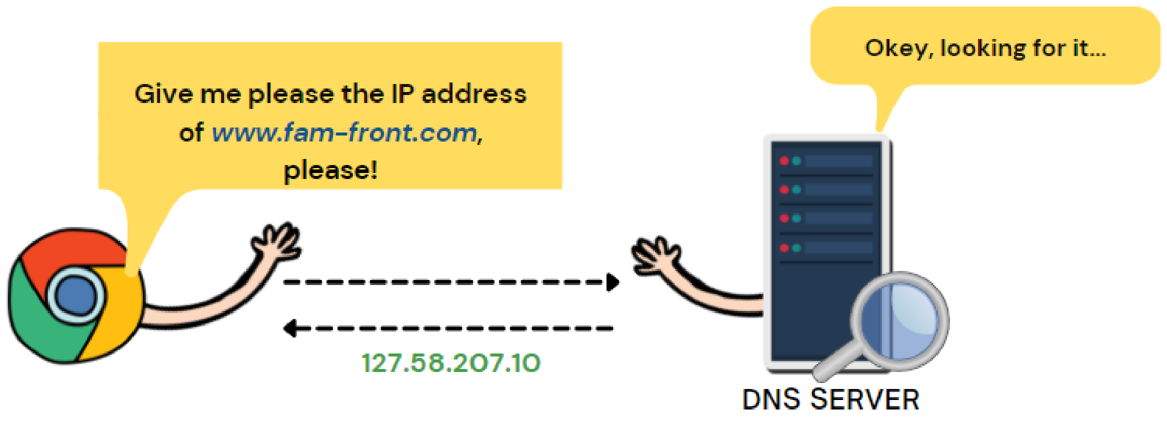
We have discovered what happens behind the scene when making a request and how web pages communicate on the Internet. But, there is still a puzzle that is missing which is the browser. Do you wonder what its job in the whole story?

Well, the browser plays an important role that consists in doing the following tasks:



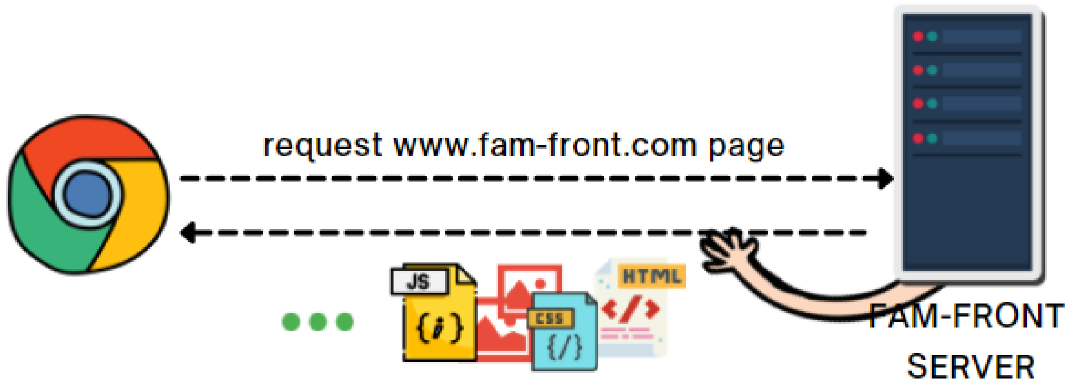
Step 1: DNS resolution

I mentioned this earlier. DNS is like a register that has for each IP address its corresponding domain name. The browser is the one who resolves this task. That means that if I published a website online, I don't need to manage to get the IP address from my website *www.fam-front.com*.



Step 2: HTTP/HTTPS exchange

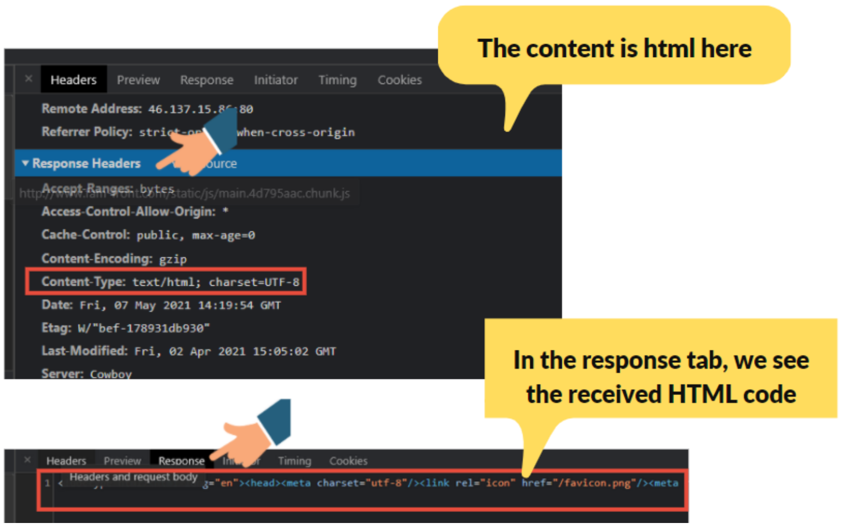
At this step, the browser has the server address. The communication can get started using the internet language (HTTP) or secured language (HTTPS) to talk between web-based applications. The browser acts as a client. It requested a website and waits for the server to process its request and send a response. The response can be HTML code, images, videos, files, audios …etc.



Step 3: Rendering

This step is the most exciting one when you get to see the result of your request. In this example, what we receive are HTML, images, JS, and CSS files. But that's because I requested access to my website. There are cases when we get data from a Database instead of HTML code. It depends on what we have asked for.

To know what we are waiting for as a response, we need to check the response body. The Content-Type tells us what content we have received. If it's text/html, that means we received an HTML page.



```
Remote Address: 46.137.15.8‹:80
Referrer Policy: strict-or     when-cross-origin
▾ Response Headers          ource
Accept-Ranges: bytes
Access-Control-Allow-Origin: *
Cache-Control: public, max-age=0
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Date: Fri, 07 May 2021 14:19:54 GMT
Etag: W/"bef-178931db930"
Last-Modified: Fri, 02 Apr 2021 15:05:02 GMT
Server: Cowboy
```

```
1 < Headers and request body g="en"><head><meta charset="utf-8"/><link rel="icon" href="/favicon.png"/><meta
```

FYI

To access the request and response as I did above, after hitting enter to get a website, hit **F12**. In the **network** tab, you'll see all requests and responses.

Don't worry if you don't have a clear idea of how it works, will come back with more details about it in the next handbook.

If you are not familiar with HTML, JS, and CSS, no worries, we'll get to talk about them in detail in the coming e-book.

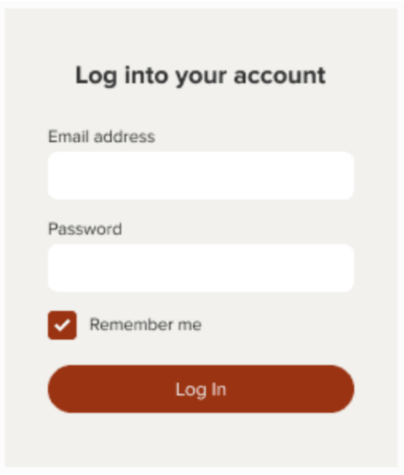
Step 4: Clean & repeat

Repeat endlessly with other requests.

2. What's web cookies and session?

a. Web cookies

A web cookie is also called a browser cookie. It's not something you can eat, Hahaha. A cookie is a small piece of data that a server sends to your browser. The browser may then store it and send it back to the same server. You're wondering what the purpose of this cookie is? Well, it's used to tell requests came from the same browser.



Here is an example to understand the power of a cookie on the Internet. Have you ever checked the little checkbox to tell the browser to remember your login information? That information is stored in a cookie that is communicated with your request, and the server knows you right away. That's why we often label the checkbox "remember me".

So, this is all that it is for cookies?

Well, it's a no. There are other purposes for web cookies, such as:

- Save preferences (theme, settings, etc.)
- Tracking (record and analyzing your behavior to show you ads based on your behavior on the Internet)

- The last one is managing sessions. An example of that could be saving your shopping carts, a game score, etc.

FYI

- Cookies were once used for general **client-side storage**.
- Cookies are **sent with every request**, so they can **worsen performance** (especially for mobile data connections).
- Nowadays, it is more recommended to use **modern storage APIs**.
- Modern APIs for client storage are the Web Storage API (**localStorage** and **sessionStorage**) and **IndexedDB**.

b. Web sessions

A session is also used to store data. The difference between cookie and session relies on the following:

COOKIES	SESSIONS
• Are client-side (saved on the browser) files that contain user information. • Not dependent on the session. • Cookie expires depending on the lifetime you set for it. • The maximum cookie size is limited (4KB).	• Are server-side files that contain user information. • Session is dependent on Cookie. • Session ends when a user closes his/her browser. • With session, you can store as much data as you like.

Note that

We will learn more and get to use cookies and sessions in the next steps of the collection. So no worries, we will code and develop fantastic projects together.

V. Web hosting

We talked several times about the server that hosts a website. Do you wonder how it works and what happens behind the scenes? Well, this section will clarify this part for you!

First, we need to know what hosting is, right?

Hosting

A server hosts a website when this server allocates space on a webserver to store this website's files. So, if you want to publish and make your website online on the Internet, you need a server that will host your website file and stay connected to access it whenever you want to.

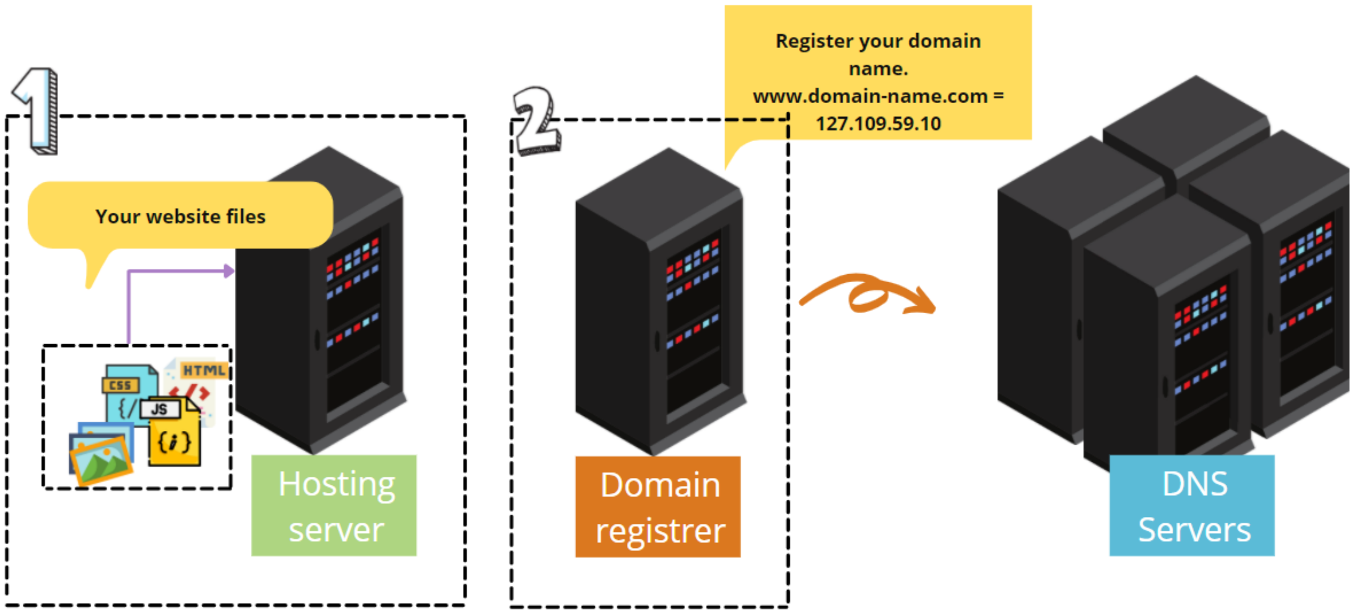
What if you want to host your website on your computer?

Well, that's possible, but you need to make your computer a server that will be open and connected 24/24 so that you can see your website on the Internet.

What do you need to do concretely to make your website online?

You need two main things:

- A hosting server
- A domain name (A domain registrar)



The hosting server will make sure to provide your website source code when needed (when you request visiting your website).

The Domain registrar is an accredited organization that sells domain names. Your domain name is then registered in DNS servers. It is after then recognized on the web.

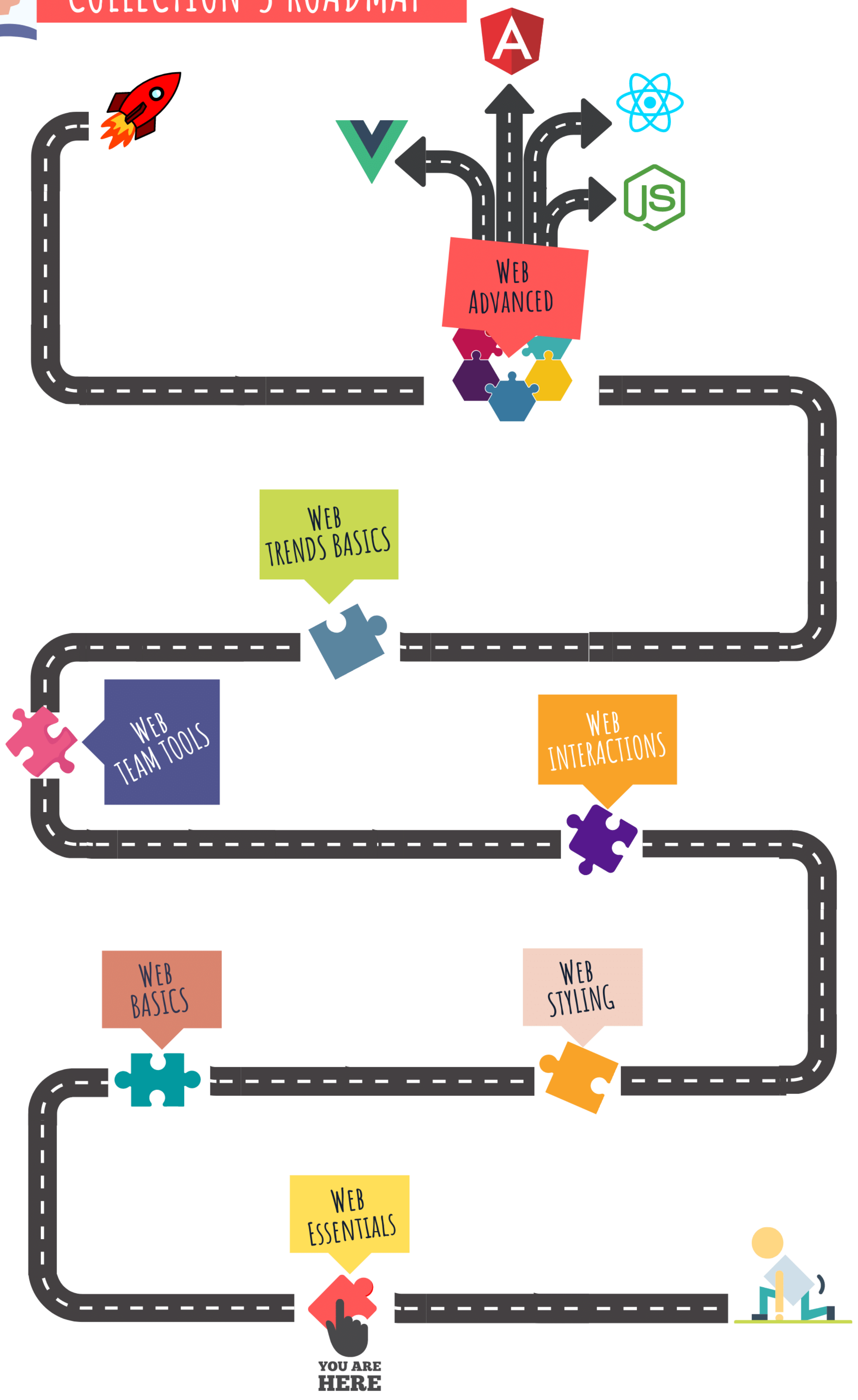
The next step

The following handbook is about the web basics. We will see how we can render things on a web page with HTML. The guide we'll talk about:

- HTML5 fundamentals
- Best practices
- Forms
- Semantic HTML
- Accessibility
- SEO basics

YOUR PASSPORT TO MODERN WEB DEVELOPER JOB
THE
WEB
BASICS
YOUR PATH TO A PROFETIONNAL MODERN WEB DEVELOPER

COLLECTION'S ROADMAP



WEB ADVANCED
WEB TRENDS BASICS
WEB TEAM TOOLS
WEB INTERACTIONS
WEB BASICS
WEB STYLING
WEB ESSENTIALS
YOU ARE HERE