# Level Up - Notions Développement web

| | |
|---|---|
| Class | |
| Created | @Mar 29, 2021 9:57 AM |
| Materials | |
| Reviewed | ☐ |
| Type | |

Le développement web regroupe les activités permettant aux internautes, via un navigateur web, d'accéder à des interfaces consommant de l'information fournie par une API. Ce dernier pouvant être utilisé sur de nombreux appareils aux écrans de tailles différentes (smartphone, tablette, ordinateurs, ...).

- **le fonctionnement du Web et du protocole HTTP**

En bref – Internet est un réseau permettant aux ordinateurs qui y sont connectés de partager de l'information via différents services. Le Web est un de ces services. Il permet d'accéder à des pages Web localisées sur un serveur identifié par une adresse IP grâce au protocole HTTP et auxquelles les internautes accèdent grâce à leur URL via un navigateur.

**TCP/IP protocol :** Trasmission Control Protocol and Internet Protocol.

→ **IP adresses** (computer adresses : ours + the one to which we send stuff)

4 numbers

#.#.#.#

(all range from 0 to 255 / or 8 bits of information (32 bits in one adress- there only can be 4 billion devices)→ in binary 255 it's 8 ones : one after the other)

**IPv4 protocol** = 32 bit adresses → still pretty commun

TODAY we have **IPv6** (128-bit adresses) → many devices this

**Port numbers** allow to transfter files(service FTP=21 port), e-mails (SMTP=25), web pages (HTTP=80)

1.2.3.4 : 80

(first four⟶ IP adress, 80 is the port- HTTP, we are sending a webpage)

>But usually we don't write IP adresses we write an **URL** (exemple : https://www.example.com - this is a **host** = the page to which I want to connect)

▼ **DNS (Domain Name System):** a mapping/translation between URLS and corresponding IP. DNS are servers present on the internet.

▼

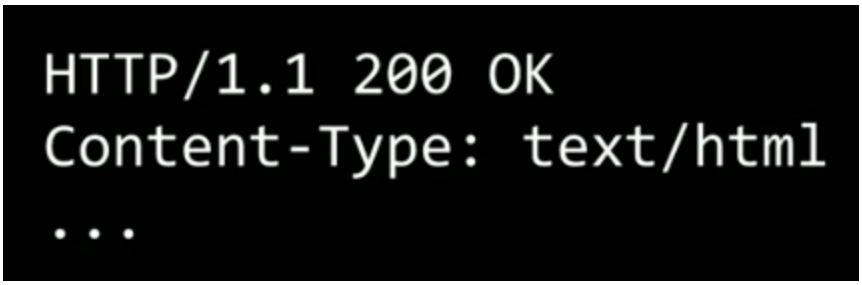
**http requests :**

"Hyper Text Transfer Protocol" - HTTP is a protocol on how machines communicate with each other - a machine does a request and hopefully gets an answer back ⟶ **request-response cycle**

Request looks like this :

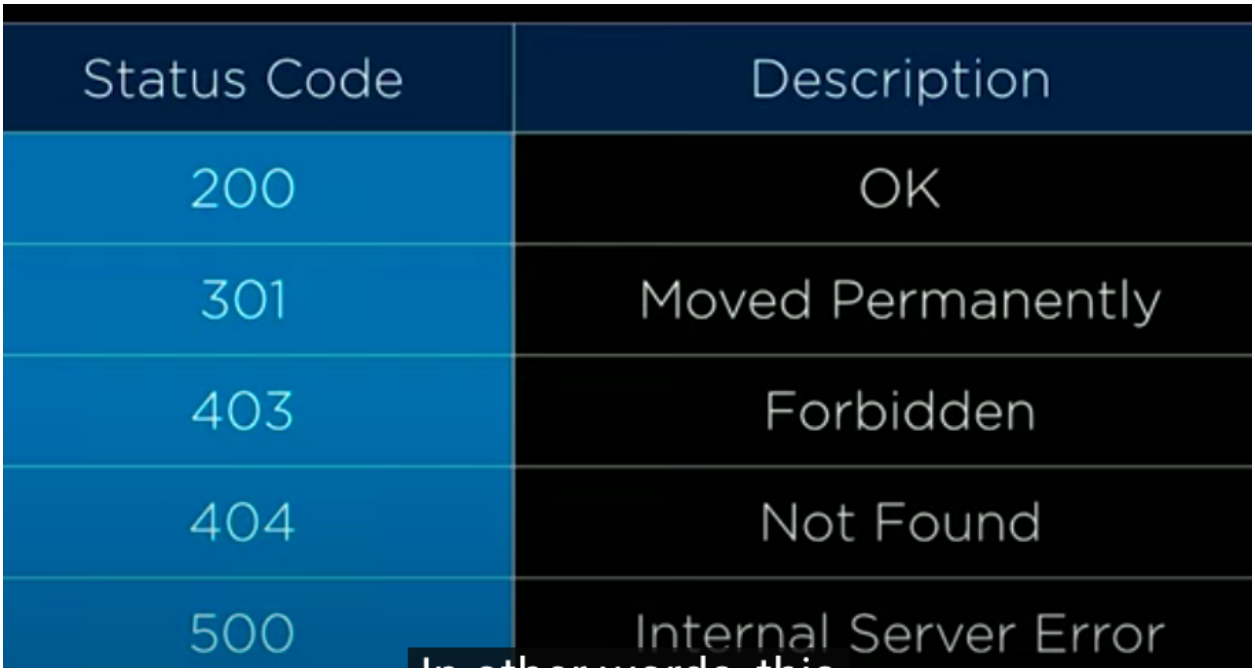

We specified http version 1

Response :



200 - means OK (it's a status code)

Other famous **status codes** :



| Status Code | Description |
|---|---|
| 200 | OK |
| 301 | Moved Permanently |
| 403 | Forbidden |
| 404 | Not Found |
| 500 | Internal Server Error |

301 : redirection (exemple from http to https)

HTTP : it's about what's inside

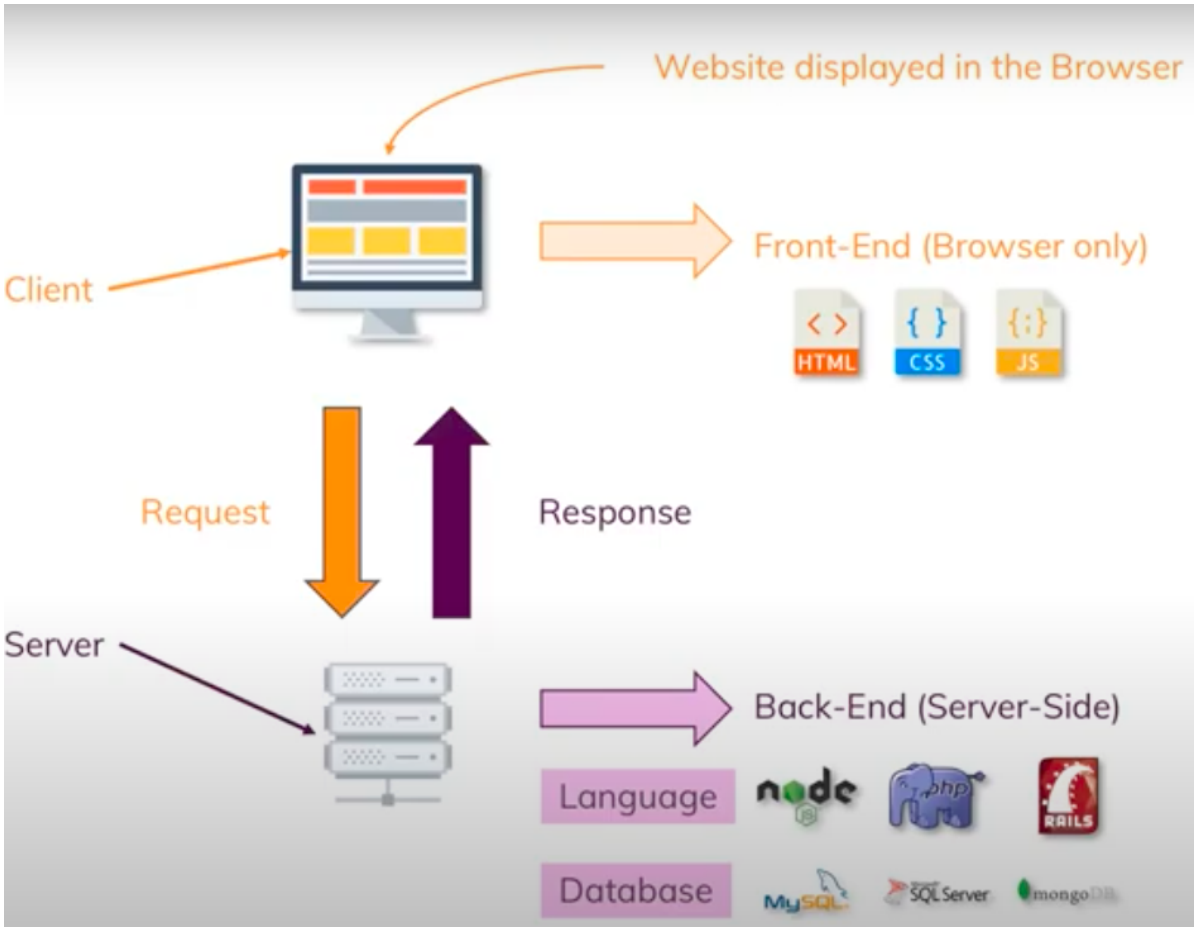
Exemples of http requests :

- Clicking "enter"
- clicking on a link
- Click on Google search button
- Writing an URL

HTTPS : information is encripted, connection more secure than HTTP

A **server =** a computer/dedicated machine that receives http requests and responds with something → it sends it back to my **browser (navigateur web)** that renders the content so it makes sense to me (ex : transforms code in a web page)

- 404 Error : is the server response, he cannot give us anything




- **les langages lus par les navigateurs web et qui sont donc utilisés pour développer les sites et applications web (HTML, CSS, JavaScript, PHP, ...)**

**Quels sont les rôles respectifs du HTML, du CSS et du JavaScript sur une page web ?**

En bref – Un navigateur web ne sait lire que certains langages et ces derniers ont chacun leur rôle dédié.

**HTML** : is the NOUN of the we page → the structure of the web page, **Hyper Text Markup Language** (we markup a document to describe its structure - at the beginning created for universities to share research papers)

Le HTML permet de définir le squelette sémantique d'une page web et contient le texte (la donnée) à afficher

**CSS** : ADJECTIVE - is the style, describes the web page

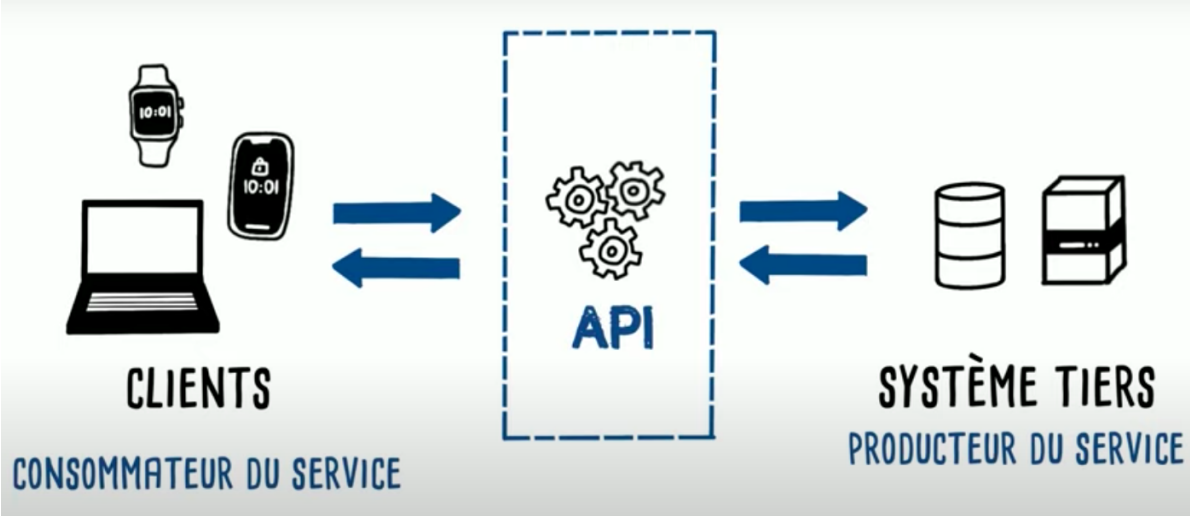
Le CSS permet de mettre en forme ce texte (cette donnée).

**JS :** The verb - action

JavaScript permet lui d'accéder aux éléments HTML de la page pour les rendre dynamiques et permettre à l'utilisateur d'interagir avec eux.

Ces 3 langages sont aujourd'hui utilisés pour développer des interfaces web. Côté serveur on peut utiliser des langages tels que le PHP, Ruby ou encore JavaScript grâce à Node.js.

- **ce qu'est une API, comment elle est consommée par un site ou une application web et comment on les développe**
- **Qu'est-ce qu'une API ? Quelles sont les 4 grandes fonctionnalités les plus communément fournies par une API ?**



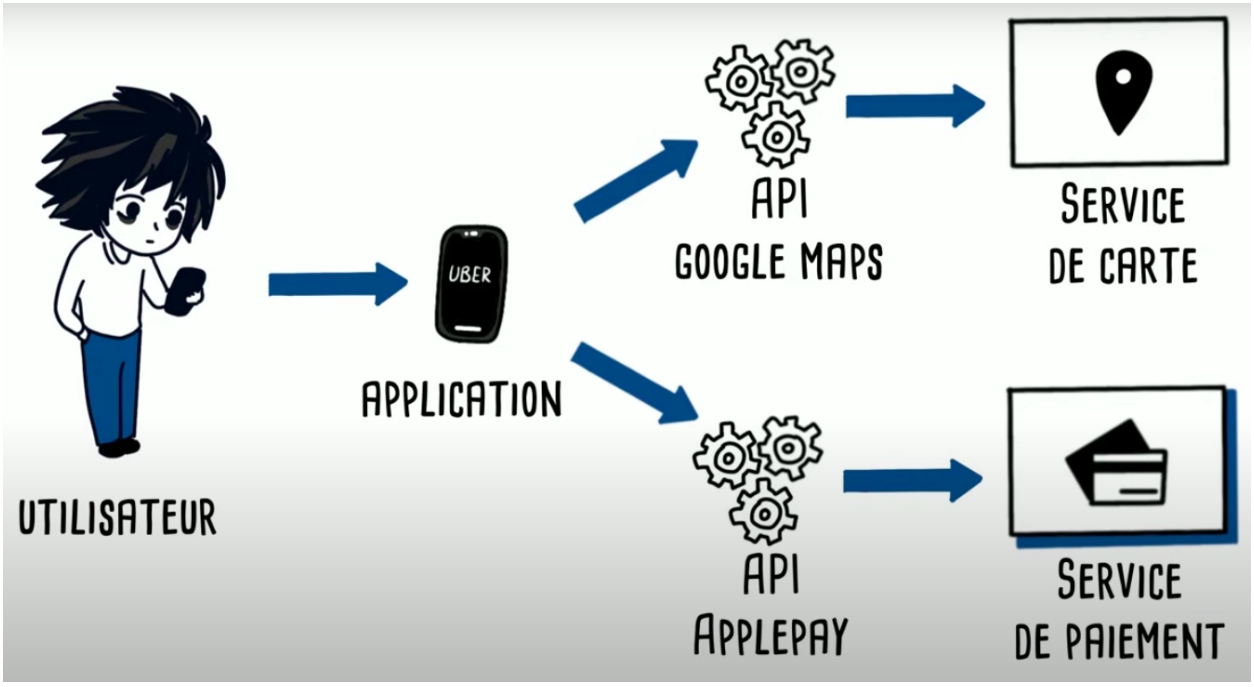

Une API, ou Application Programming Interface, n'est rien de plus qu'un programme avec lequel on échange de l'information. C'est l'intermédiaire entre la donnée (qui est stockée dans des bases de données) et l'interface qui formate et fournit cette donnée à l'utilisateur. Une API doit nous permettre de lire de

l'information, d'en créer, de la mettre à jour et de la supprimer. Ce sont les fonctionnalités **CRUD pour Create, Read, Update et Delete.**

> L'acronyme CRUD se réfère à la majorité des opérations implémentées dans les bases de données relationnelles. Chaque composante de l'acronyme peut être associée à un type de requête en SQL ainsi qu'à une méthode HTTP.

| Operation | SQL | HTTP |
|---|---|---|
| Create | INSERT | POST (en) |
| Read (Retrieve) | SELECT | GET (en) |
| Update (Modify) | UPDATE | PUT (en) |
| Delete (Destroy) | DELETE | DELETE (en) |

Exemple : Uber (les travaillaurs d'Uber ne gèrent pas les API, c'est des services à part)



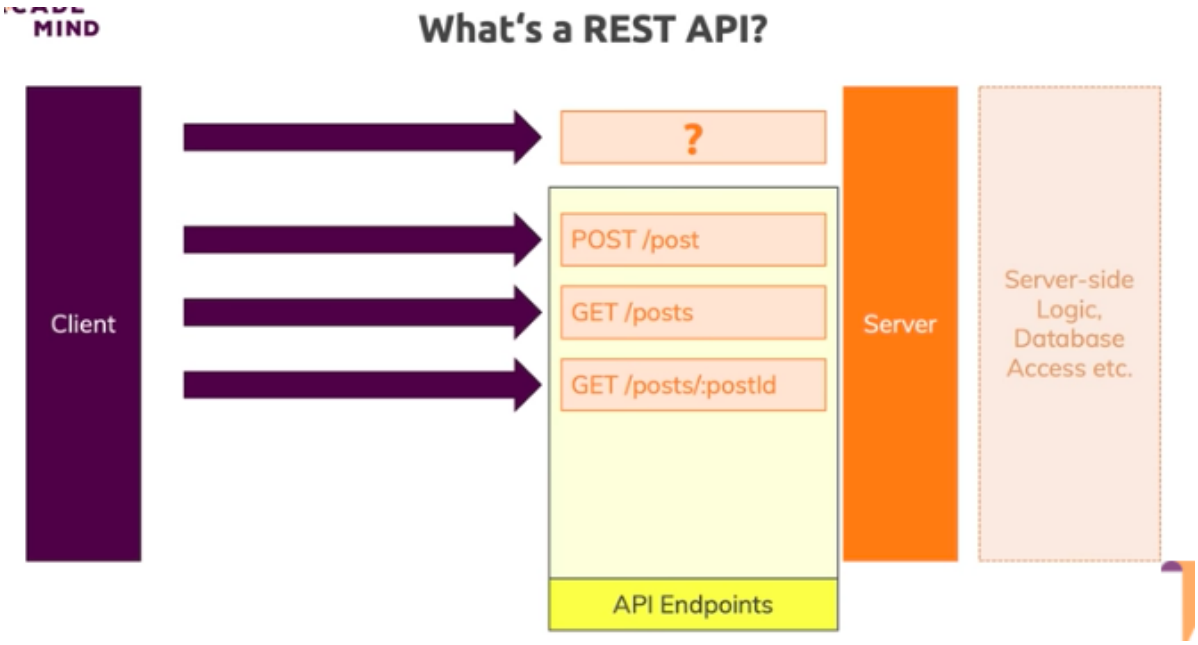
- Les API facilitent la vie du développeur - il crée des fonctionnalités connectées à des API
- L'APi est aussi une opportunité business pour le dev
- La DOCUMENTATION API : le developpeur doit définir ses requêtes en suivant une documentation très précise de l'API décrivant le cadre d'usage de l'API (fonctionnement et comment utiliser l'API...)

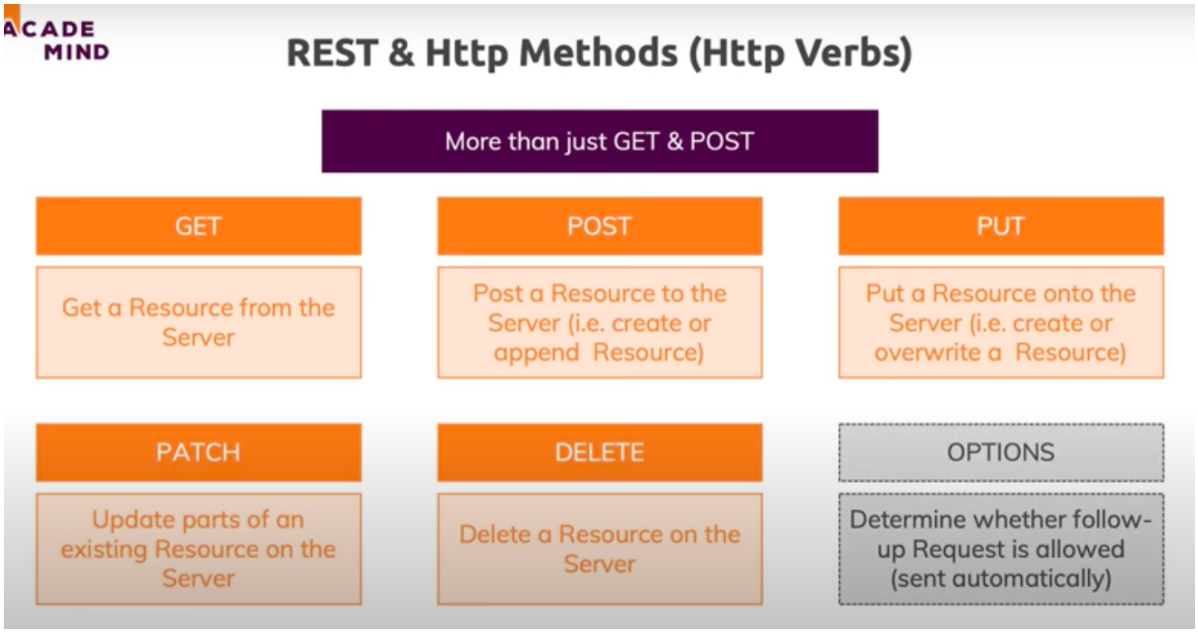
- **Quel protocole réseau utilise un site/une application web pour communiquer avec une API ?**

La plupart des API sont conçues sur la base des normes Web. (Toutes les API distantes ne sont pas des API Web)

Les API Web utilisent en général le protocole HTTP pour leurs messages de requête et fournissent une définition de la structure des messages de réponse. Les messages de réponse se présentent la plupart du temps sous la forme d'un fichier XML (Extensible Markup Language) ou JSON. Ces deux formats sont les plus courants, car les données qu'ils contiennent sont faciles à manipuler pour les autres applications.

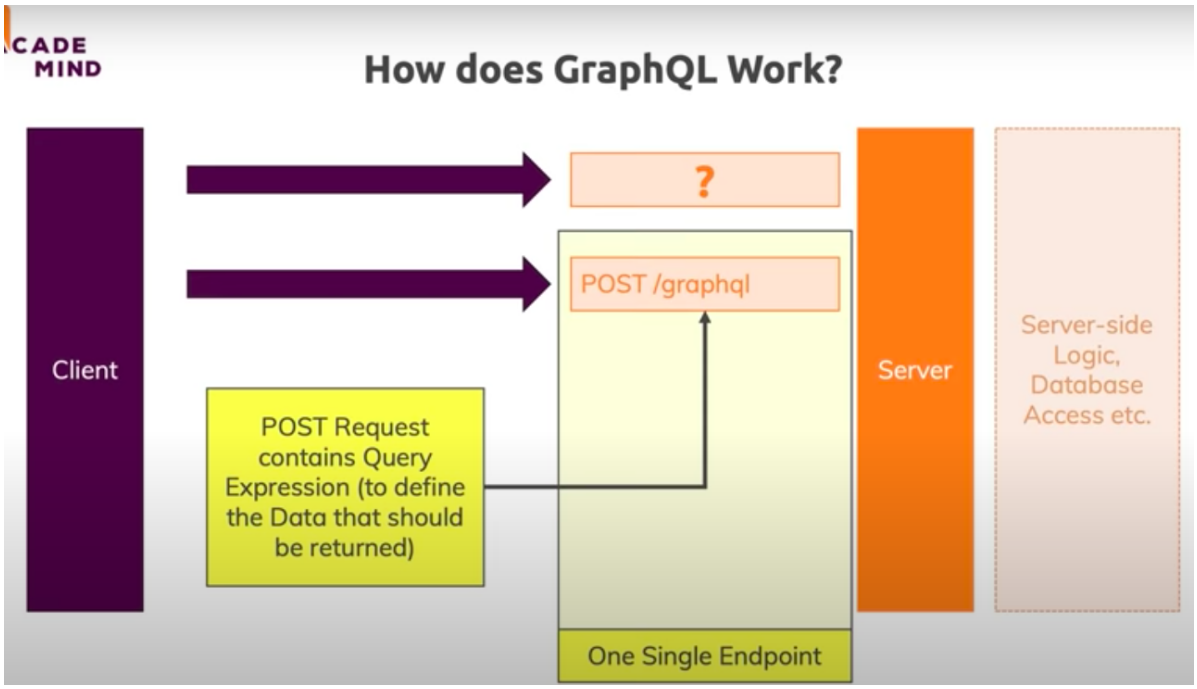
- **API REST** : API construite en respectant les standards du web. REST est une architecture web basée sur le HTTP (protocole de référence déifnissant communications sur le web) → l'API c'est le quoi, le REST c'est le comment : les échanges sont établies à travers de requettes HTTP au serveur
- Pas toute les API utilisent le protocole REST, mais l'utiliser garantit une meilleur intégration aux communications du web
- Another kid of API : GraphQL API (only 1  andpoint) vs REST (multiple endpoints)




GRAPHQL

- Only one endpoint that is an action POST

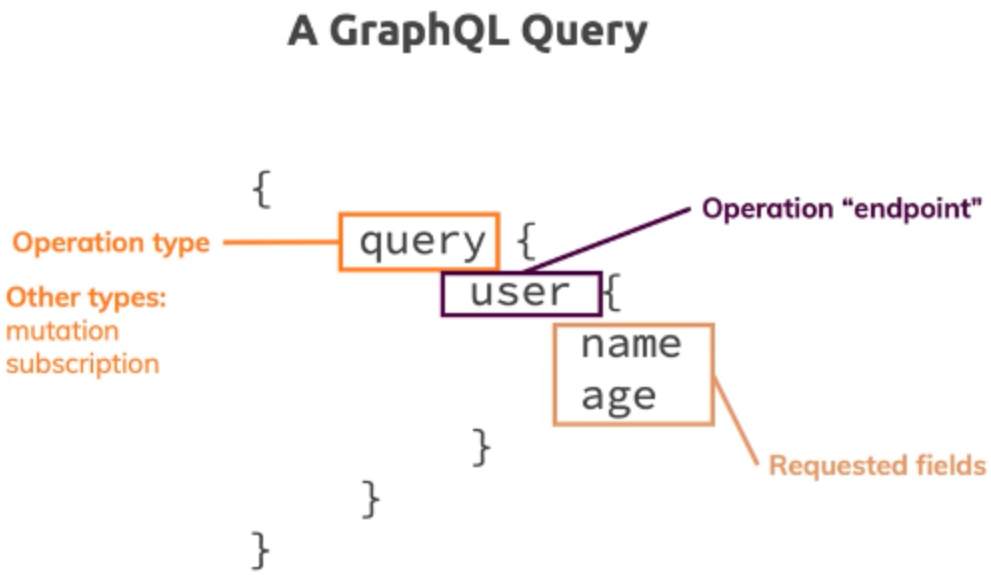



OPERATION TYPE

query = get some data

mutation = edit some data

subsription = set up live suscription




BIG ADVANTAGE of GraphQL : you target the fields you want to get (ex : only name and age), not possible with REST (you fetch too much); with GraphQL often you are more specific - but REST easier to understand and a lot more used because they are easier to use

**le fonctionnement des bases de données, de manière basique, afin de comprendre d'où vient la donnée consommée par les sites et applications web et comment elle est traitée**

- DATA = any facts related to an object in consideration
- **DATABASE** = systematic collection of data (it's organised so data management is easy)
- **DBMS (DATABASE MANAGEMENT SYSTEM)**: a collection of programs which enables its users to access database, manipulate data, & help in representation of data + helps control access to the database by various users

EXEMPLE :


4 types of dbms :

- hierarchical dbms: parent-child way of storing data (rarely used today, structure like a treestructure and nodes). exemple: windows registry used in winddows xp
- network dbms (supports many relationships)
- object oriented relatonal dbms : data to be stored is in forms of objects (data has attributes⟶ gender and age and methods that define how to use these attributes)
  - Postgree SQL
- MOST POPULAR IN THE MARKET: relational dbms = defines database relationships in forms of tables, also known as relations. Rel DBMS usually have predifined data types thatt they can support.
  - Exemples : My SQL, Oracle, My SQL server

**SQL (Structured Query Language) =** the standard language for dealing with Relational Databases (language that communicates with them)

- Used to insert, search, update and delete database records.
- It helps optimizing and maintenance of databases and much more
- MYSQL, ORACLE? MS SQL SERVER, SYBASE use SQL
- Sql syntaxes used in these databases are almost similar (but some databases use dfferent syntxes and even proprietary sql syntaxes)

- **et enfin les notions de déploiement, c'est-à-dire comment une application web est "mise en ligne" et rendue accessible sur le web à une adresse URL.**

https://medium.com/swlh/localhost-to-com-deploying-a-web-app-for-beginners-ea05b0213eb7

We're going to look at how to rent a server, connect it to a domain, and run Apps on it.

1. **Setting up a VPS**

- VPS (Virtual Private Server): A small, virtual computer running on the Cloud, that you can pay a fee to use. You'll rent one of these to run your App on.

1. **Buying a Domain**

- The human readable name that represents an IP address; for example: 'medium.com'. You'll need to buy a domain that people will use to reach your App.
- You can get a Domain from any one of countless registrars. I'll be using Google Domains for this guide. Head to https://domains.google.com/registrar/search. After logging in with a Google account, you can search for available domains on the 'Get a new domain' page.

1. **Connecting the Domain to the VPS**
2. **Connecting to the VPS using SSH**

   SSH: A protocol used to run commands on a Linux server over the internet.
3. **Setting up the Web App**

   the App is the software that you want to deploy. It accepts HTTP requests and provides some sort of response. This could perhaps be a Web API, or something that serves a Website. The specific technology used doesn't matter, but I'll be using Node.js.
4. **Setting up a Reverse Proxy**

For our needs, this can be defined as a piece of software that runs on your Server, that intercepts all HTTP requests before passing them on to your App.

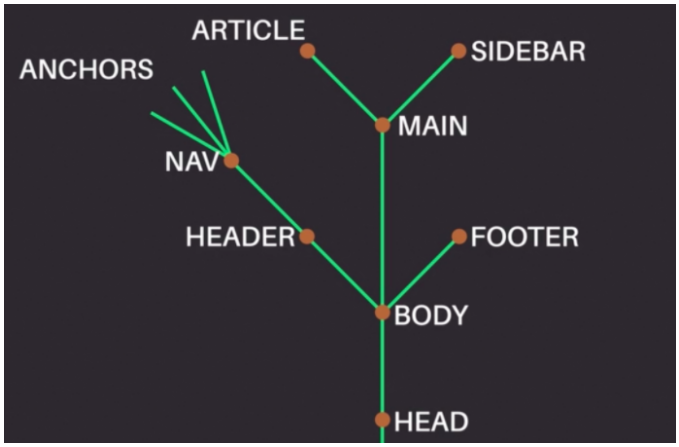
5. **Securing the App with HTTPS**

   The ‘S’ in HTTPS shows that the communication is encrypted, using keys from a Certificate Authority.

## Qu’est-ce que le DOM ?

Le DOM, qui signifie Document Object Model (c'est-à-dire "modèle d'objet de document", en français), est une interface de programmation qui est une représentation du HTML d'une page web et qui permet d'accéder aux éléments de cette page web et de les modifier avec le langage JavaScript.

Dans le DOM, on commence toujours par un élément racine qui est le point de départ du document : la balise <html> . Celle-ci a pour enfants les balises <head> et <body> qui ont donc un parent commun : la balise <html> ! Vous trouverez ensuite le contenu de votre page dans la balise <body> sous forme de liens, boutons, blocs, etc.




Avec une interface de programmation nous permettant de parcourir le DOM, nous allons pouvoir **interagir** avec lui. Ces interactions comprennent :

- La modification du contenu d'un élément précis ;

- La modification du style d'un élément ;
- La création ou la suppression d'éléments ;
- L'interaction avec les utilisateurs, afin de repérer des clics sur un élément ou encore de récupérer leur nom dans un formulaire ;
- Etc.

→ LE DOM : Cela vous permet d'avoir une navigation fluide et agréable, qui n'est bien sûr pas pour nous déplaire.

→ le DOM permet que le site change et évolue en continu sans à avoir besoin de recharger la page

**Qu'est-ce qu'une requête ? Nommer 4 types de requêtes.**

Sur le Web, les clients, comme votre navigateur, communiquent avec les serveurs Web en utilisant le protocole HTTP. Ce protocole contrôle la façon dont le client formule ses demandes et la façon dont le serveur y répond. Le protocole HTTP connaît différentes méthodes de requête.

https://www.ionos.fr/digitalguide/hebergement/aspects-techniques/requete-http/

# 1) GET

GET est l' « ancêtre » des requêtes HTTP. Cette méthode de requête existe depuis le début du Web. Elle est utilisée pour **demander une ressource**, par exemple un fichier HTML, au **serveur Web.**

Si vous entrez l'URL *www.exemple.com* dans votre navigateur, celui-ci se connecte au serveur Web et lui envoie la requête GET → **GET /index.php**

La requête GET peut recevoir des **informations supplémentaires** que le serveur Web doit traiter. Ces paramètres d'URL sont simplement ajoutés à l'URL. La syntaxe est très simple :

- La chaîne de requête est introduite par un « ? » (point d'interrogation).
- Chaque paramètre est nommé, il se compose donc d'un nom et d'une valeur : « Nom=Valeur ».

- Si plusieurs paramètres doivent être inclus, ils sont reliés par un « & ».

Voici un exemple : sur le site Web d'une entreprise de logiciels, « Windows » est saisi comme plateforme et « Office » comme catégorie pour afficher les offres correspondantes du serveur :

```
GET /search?platform=Windows&category=office
```

## 2) POST

Si vous souhaitez envoyer de grandes quantités de données, par exemple des images, ou des données confidentielles de formulaires au serveur

. Cette méthode n'écrit pas les paramètres de l'URL, mais les ajoute à l'en-tête HTTP.

Les requêtes POST sont principalement utilisées pour les **formulaires en ligne**. Voici un exemple de formulaire qui prend le nom et l'adresse électronique et les envoie au serveur en utilisant POST :

```
<html>
<body>
<form action="newsletter.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

## 3)HEAD

La méthode HEAD est utilisée pour interroger l'en-tête de la réponse, sans que le fichier ne vous soit envoyé immédiatement. → C'est utile, par exemple, si des fichiers volumineux doivent être transférés : grâce à la requête HEAD, le client peut d'abord être informé de la **taille du fichier** et seulement ensuite décider s'il veut recevoir le fichier.

Exemple :

```
HEAD /downloads/video1.mpeg HTTP/1.0
```

Réponse HTTP:

| | |
|---|---|
| age | 96529 |
| cache-control | max-age=604800 |
| content-length | 1495 |
| content-type | text/html; charset=UTF-8 |
| date | Fri, 06 Mar 2020 14:53:39 GMT |
| etag | "3147526947+gzip" |
| expires | Fri, 13 Mar 2020 14:53:39 GMT |
| last-modified | Thu, 17 Oct 2019 07:18:26 GMT |
| server | ECS (dcb/7F83) |
| vary | Accept-Encoding |
| x-cache | HIT |

# 4) OPTIONS

Avec la méthode OPTIONS, le client peut demander **quelles méthodes** le serveur supporte pour le fichier en question.

```
OPTIONS /download.php
```

Exemple :

| | |
|---|---|
| allow | OPTIONS, GET, HEAD, POST |
| cache-control | max-age=604800 |
| content-length | 0 |
| content-type | text/html; charset=UTF-8 |
| date | Fri, 06 Mar 2020 14:06:08 GMT |
| expires | Fri, 13 Mar 2020 14:06:08 GMT |
| server | EOS (vny/0452) |

content length = 0 (aucun fichier n'a été sauvegardé)

Dans le champ « allow », nous apprenons que le serveur prend en charge les méthodes OPTIONS, GET, HEAD et POST