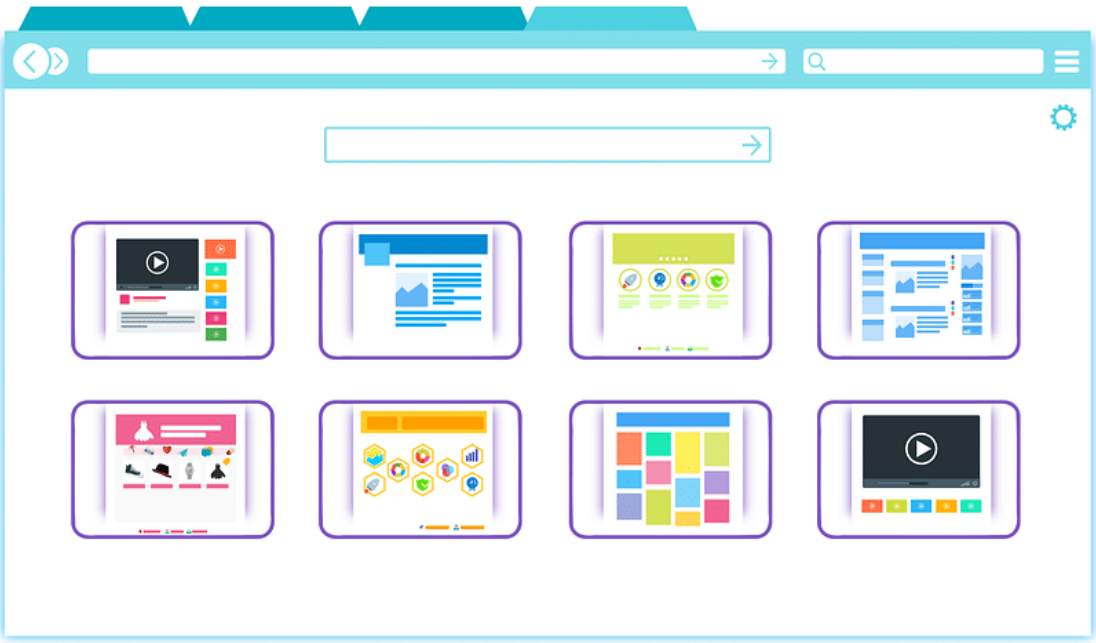
Comment fonctionnent les sites internet

Vous surfez tous les jours mais vous ne savez pas c**omment les sites internet fonctionnent**.
En effet, l'affichage des sites sur votre navigateur WEB se fait tout seul.
Mais derrière cela, il faut configurer des serveurs, créer les pages, administrer le site internet.

Pour les curieux, cet article trace les grandes ligne sur **le fonctionnement des sites internet**.



Définitions : WEB, HTTP, etc

Commençons par quelques définitions.
Un article précédent explique les différentes notions.
Cela peut s'avérer utile si vous confondez les termes **WEB, internet, www ou HTTP**.

Le Web et internet : Qu'est-ce que c'est et les différents

- **Le serveur WEB** : c'est l'application qui permet de délivrer les pages WEB au navigateur internet.
- **Le navigateur internet** : c'est le client que l'on installe pour surfer et afficher les pages WEB. Soit donc en général Mozilla Firefox, Google Chrome, Opera ou Edge.

Introduction

Le serveur WEB est une application que l'on fait tourner sur une machine qui fait office de serveur.
Ce dernier doit fonctionner 24/24 afin de pouvoir délivrer les pages WEB à tout moment.
Dans les sites importants, on peut avoir plusieurs serveurs physiques en redondances.

Le serveur WEB ouvre et écoute sur un port réseau, en général le 80 et 443.
Le navigateur WEB se connecte à ce dernier afin de demander

les pages WEB.

Qu’est-ce qu’un site internet ?

Un site internet est une succession de pages WEB avec des URLs différentes.
Par **URL (Uniform Resource Locator)** est l’adresse WEB de la page.
Par exemple, les liens suivants sont des URL :

- https://www.malekal.com/securiser-microsoft-edge/
- https://www.malekal.com/outils-luminosite-windows10/

Initialement dans les années 90, une page WEB se présente avec du texte, des images.
Des liens Hypertextes permet de passer d’une page à l’autre.
Ces derniers peuvent aussi mener à d’autres sites WEB.
Ainsi le WEB est interconnecté et souvent assimilé à une toile.

Mais depuis les années 2000 avec des connexions internet plus rapides, les sites internet ont beaucoup changé.
Le contenu est devenu plus multimédia avec l’apparition des vidéos par exemple.
Les pages WEB sont aussi devenus beaucoup plus complexes.

Ainsi au final, des applications WEB ont vu le jour.
Par exemple Facebook est une application.
On parle d'application WEB car elle fonctionne sur le navigateur WEB à travers un site internet.
Mais on peut très bien avoir une application installable comme par exemple sur votre Smartphone.

Un mode client et serveur

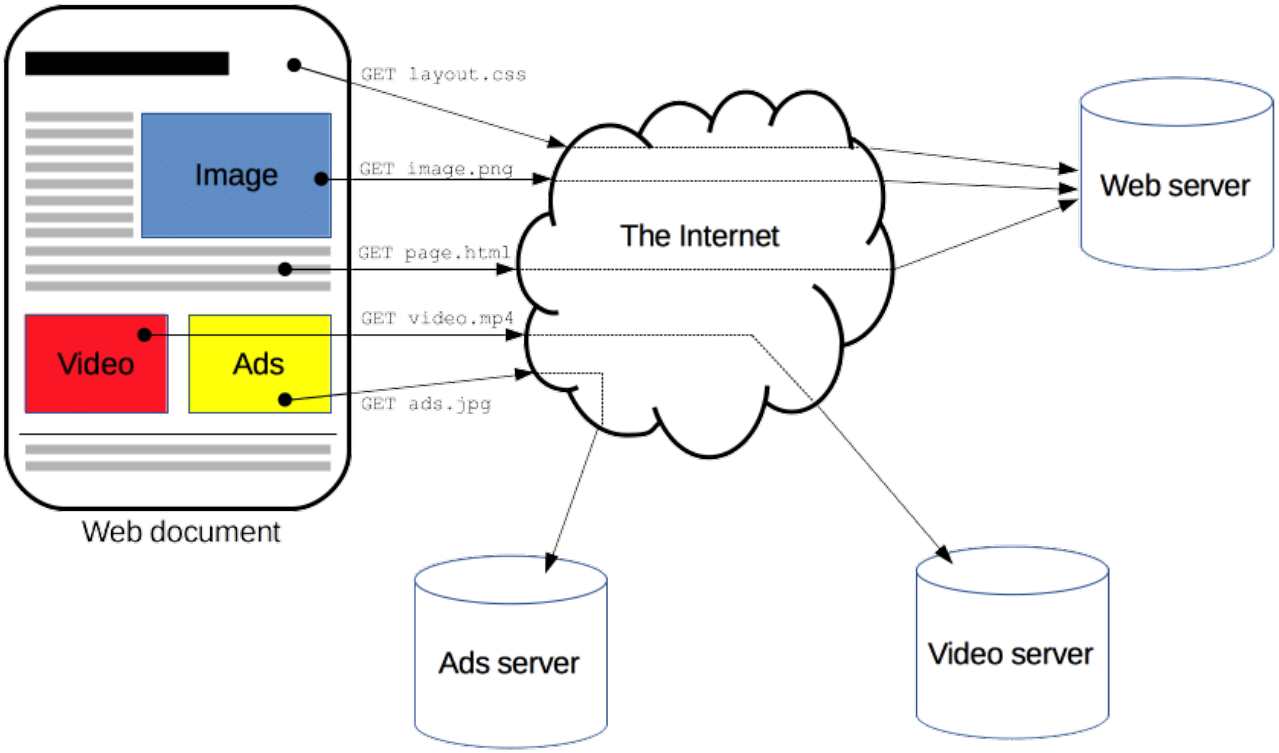
La navigation WEB fonctionne donc en mode client et serveur.
Votre navigateur internet fait office de client qui se connecte à un serveur WEB.
Lorsque l'on tente d'accéder à un site WEB, voici les étapes :

- Votre navigateur effectue une connexion réseau vers le serveur WEB. Pour cela, il peut effectuer une résolution DNS pour trouver l'adresse IP du serveur WEB.
- Il demande alors le contenu d'une page WEB au serveur WEB qui lui donne en retour.
- Puis le navigateur WEB reçoit le code de la page.

- Enfin le navigateur internet se charge du rendu à l'écran grâce à son moteur.

La page WEB peut contenir des liens vers des ressources présentes sur d'autres serveurs WEB.
Par exemple, le site malekal.com peut charger des publicités Google Adsense ou héberger une image d'un autre site internet.
Ainsi au final, pour charger le contenu d'une page WEB, le navigateur internet peut se connecter à de multiples serveurs WEB.

Le schéma suivant donne un aperçu du contenu d'une page WEB.
La page WEB se trouve sur un serveur WEB alors que les images, vidéos et publicités se trouvent sur d'autres.



source
https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

C’est pour cela que lorsque vous chargez une page WEB, en bas à gauche, vous voyez défiler plusieurs adresses WEB différentes.
De plus, le contenu de la page WEB peut se charger en plusieurs temps.
Par exemple, la page et le texte s’affiche et une fois terminée, les images arrivent.

Le navigateur WEB

Nous n’allons pas nous étendre sur ces derniers car il existe déjà un article.
Mais pour résumer, un navigateur WEB est une application que l’on installe dans le système d’exploitation (Windows, Linux, Android).

Ce dernier agit en tant que client pour effectuer une connexion vers le serveur WEB.
Il reçoit alors les pages WEB sous forme de code informatique.
Enfin le navigateur WEB interprète le code afin de composer le contenu de la page WEB à l’écran.
Cette construction se fait à travers le moteur de rendu.

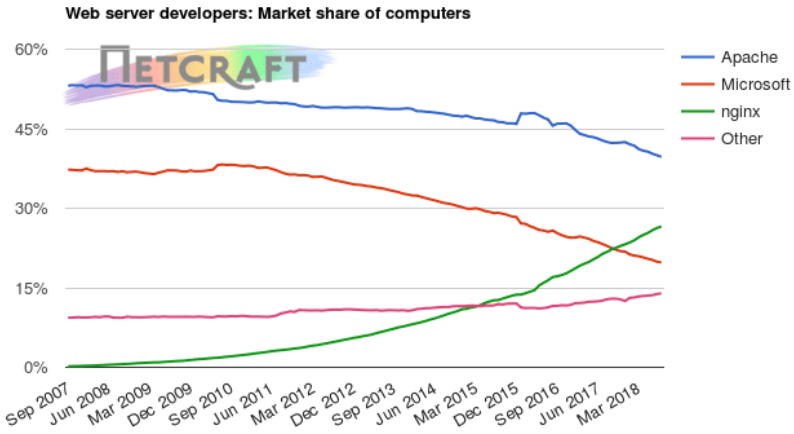
Historique et fonctionnement des navigateurs WEB

Le serveur WEB

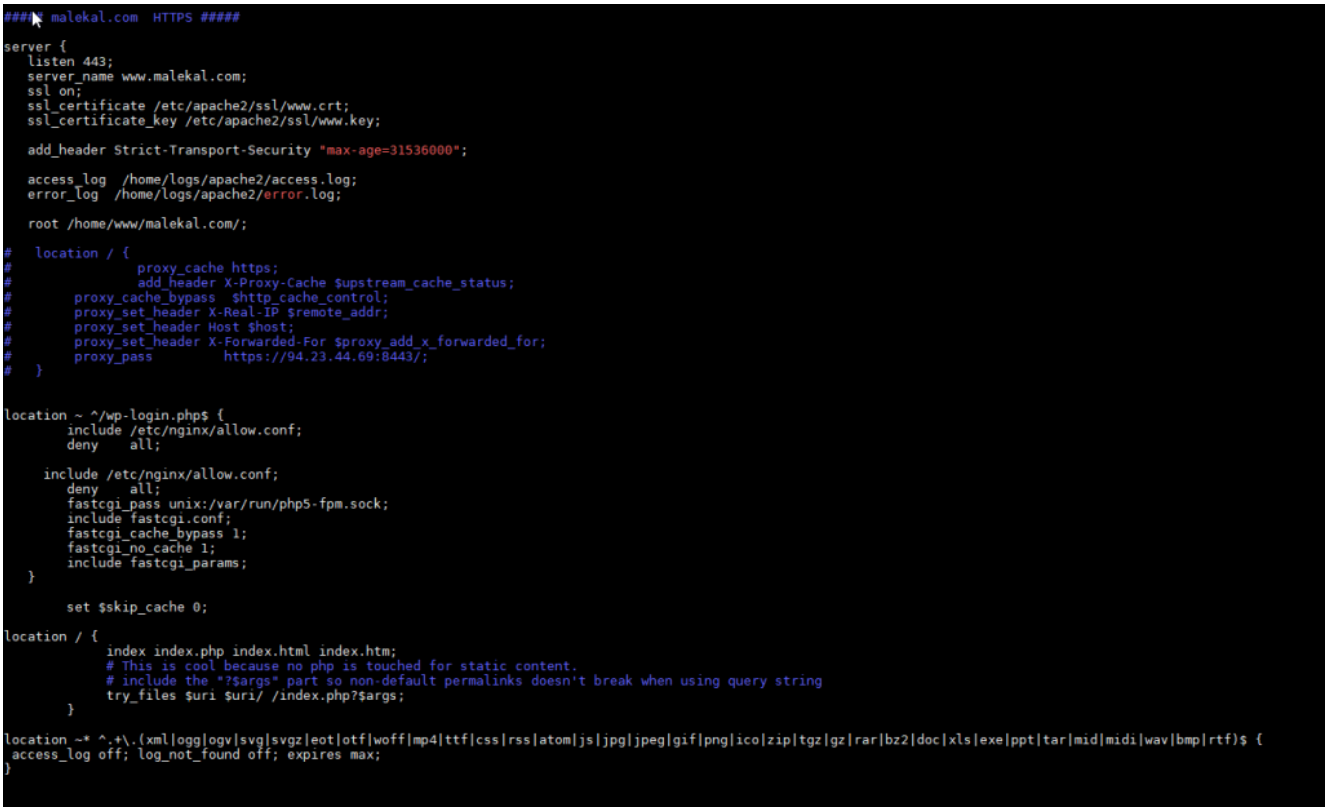
Il existe différentes applications qui font office de serveur

WEB : **Apache, Nginx, IIS** de Microsoft.

Apache a longtemps été le plus utilisé mais baisse depuis plusieurs années au profit de Nginx.
Ce dernier étant réputé plus rapide et fexible.
Microsoft avec IIS voit aussi son nombre de serveurs baissés.
Ainsi Linux reste l'OS le plus utilisé en tant que serveur pour héberger un serveur WEB.



L'administrateur configure le serveur WEB et déclare les sites WEB.
Ainsi on peut avoir des configurations différentes par site.
Ci-dessous, un exemple de déclaration du site malekal.com sur Nginx.



```
#### malekal.com  HTTPS #####

server {
   listen 443;
   server_name www.malekal.com;
   ssl on;
   ssl_certificate /etc/apache2/ssl/www.crt;
   ssl_certificate_key /etc/apache2/ssl/www.key;

   add_header Strict-Transport-Security "max-age=31536000";

   access_log  /home/logs/apache2/access.log;
   error_log  /home/logs/apache2/error.log;

   root /home/www/malekal.com/;

#   location / {
#               proxy_cache https;
#               add_header X-Proxy-Cache $upstream_cache_status;
#       proxy_cache_bypass  $http_cache_control;
#       proxy_set_header X-Real-IP $remote_addr;
#       proxy_set_header Host $host;
#       proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
#       proxy_pass         https://94.23.44.69:8443/;
#   }


location ~ ^/wp-login.php$ {
       include /etc/nginx/allow.conf;
       deny    all;

    include /etc/nginx/allow.conf;
       deny    all;
       fastcgi_pass unix:/var/run/php5-fpm.sock;
       include fastcgi.conf;
       fastcgi_cache_bypass 1;
       fastcgi_no_cache 1;
       include fastcgi_params;
   }

       set $skip_cache 0;

location / {
            index index.php index.html index.htm;
            # This is cool because no php is touched for static content.
            # include the "?$args" part so non-default permalinks doesn't break when using query string
            try_files $uri $uri/ /index.php?$args;
       }

location ~* ^.+\.(xml|ogg|ogv|svg|svgz|eot|otf|woff|mp4|ttf|css|rss|atom|js|jpg|jpeg|gif|png|ico|zip|tgz|gz|rar|bz2|doc|xls|exe|ppt|tar|mid|midi|wav|bmp|rtf)$ {
access_log off; log_not_found off; expires max;
}
```

Lorsque le site est fonctionnel et qu'un navigateur WEB demande une URL sur ce dernier, le serveur WEB répond.
Si le site est mal configuré on peut alors obtenir une erreur 503.
Les erreurs HTTP sont standardisés, vous trouverez quelques informations sur la page suivante : Erreur HTTP et problème de connexion à un site Web

De même pour les sites sécurisés, dits HTTPS.
Si le certificat est périmé, mal créer, on peut alors avoir une erreur SSL.

Le standard HTTP

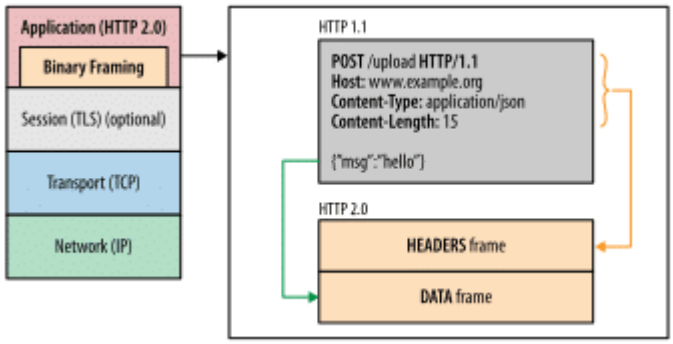
On a donc d'un côté le client WEB et de l'autre le serveur WEB.

Mais comme ces derniers communiquent ?
A travers le protocole standard de communication reste HTTP.

Voici quelques informations concernant ce dernier.
Ce protocole régit les échanges entre le serveur WEB et le navigateur internet.

Voici le contenu d'une requête HTTP.

- A gauche, la couche a transport.
- A droite, le contenu d'une requête HTTP.

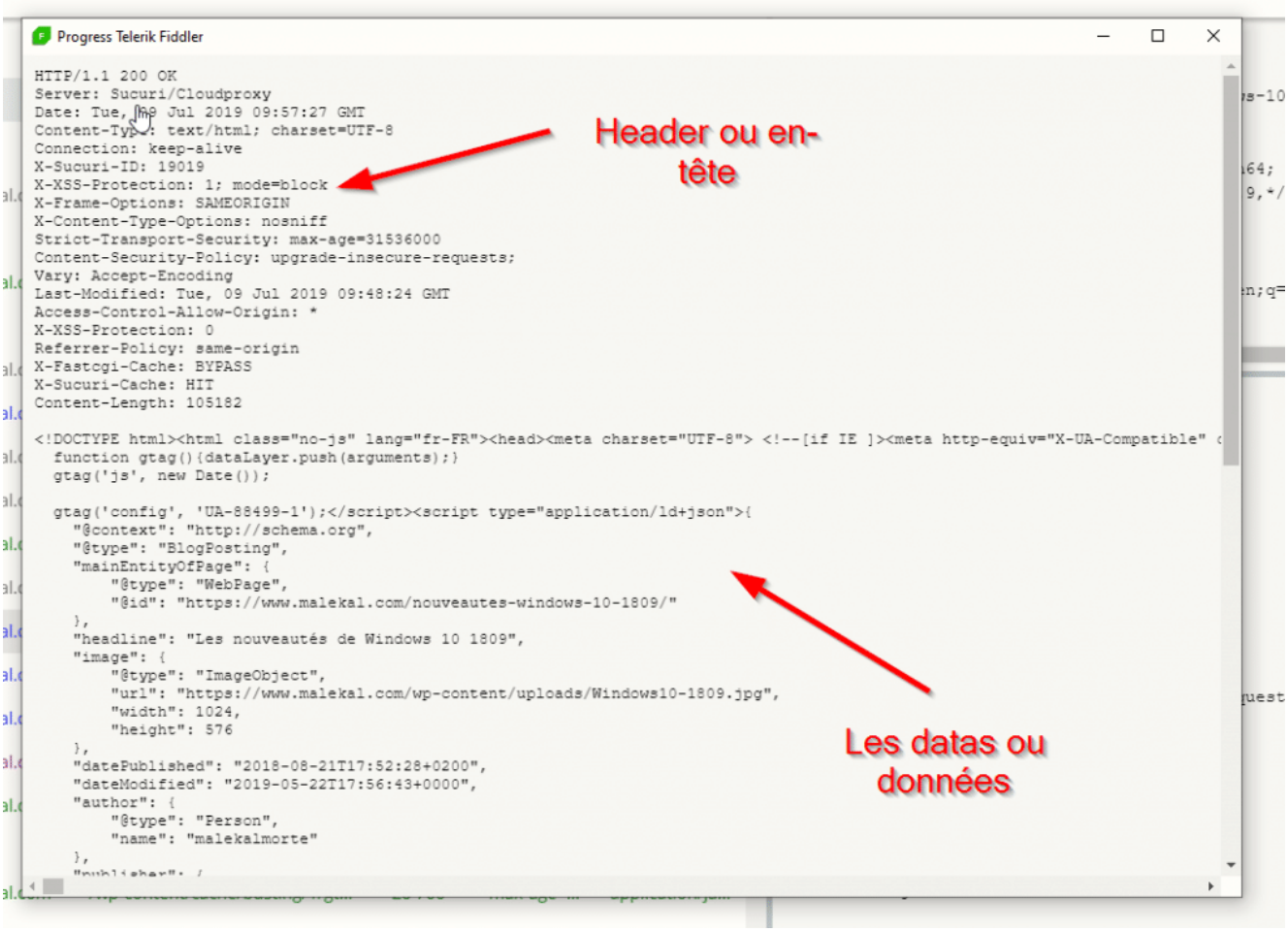


source : https://developers.google.com/web/fundamentals/performance/http2/

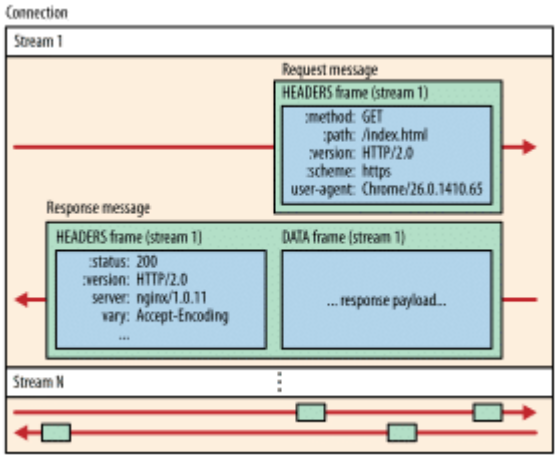
Celle-ci se divise en deux parties :

- **L'en-tête HTTP ou HTTP Header** avec des données sur la méthode, l'host et la longueur du contenu.
- **Les données ou Data**. En général, le contenu en langage de programmation de la page WEB (HTML, CSS, JavaScript, etc).

Ci-dessous un exemple de requête HTTP en retour par le serveur WEB.



Comprenez que ce protocole HTTP s'utilise dans les deux sens. Lorsque le client demande une page et lorsque le serveur WEB lui donne en retour.
Ainsi le client envoie une requête HTTP et le serveur WEB en réponse.

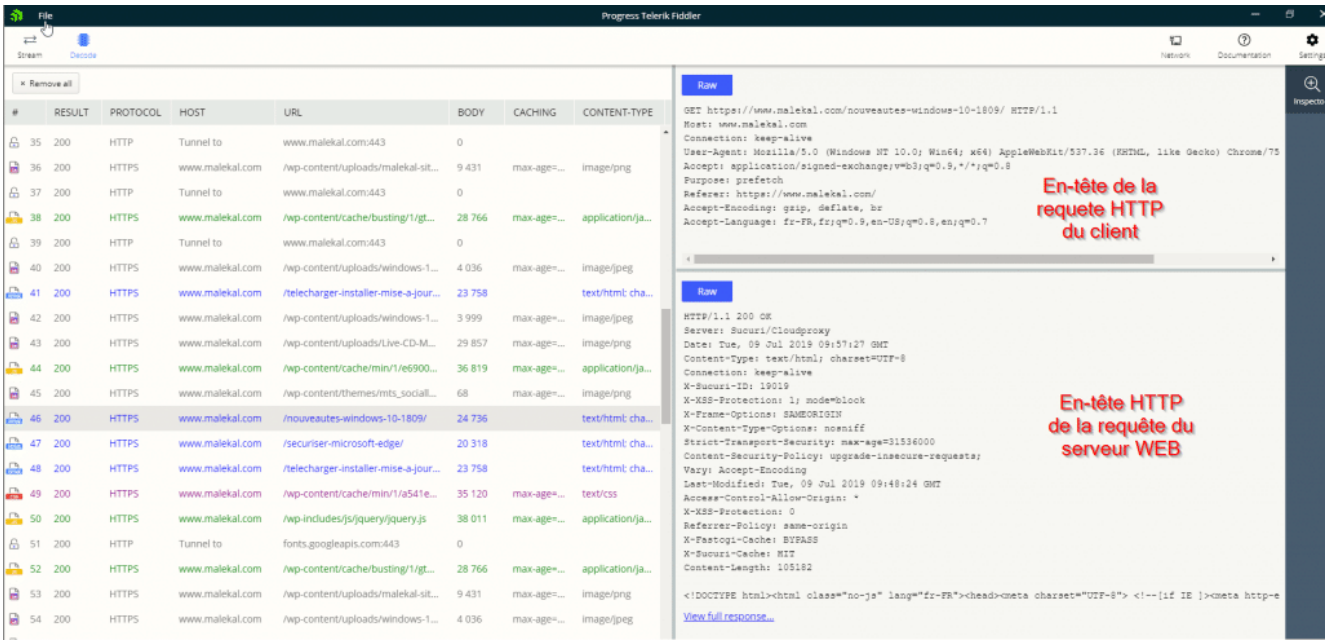


source : https://developers.google.com/web/fundamentals/performance/http2/

Si vous regardez bien, la première ligne indique la version utilisée, ici **HTTP/1.1**.
En effet, comme tous les protocoles des versions existent car ce dernier évolue.
La dernière version est le HTTP 2 mais tous les serveurs WEB ne la gère pas.
La principale différente entre les deux est sur le mode de connexion.
HTTP 1 effectue plusieurs connexions TCP alors que HTTP 2 permet d'envoyer plusieurs stream dans une même connexion TCP.

Header HTTP

Ci-dessous, les deux headers des deux côtés.
Par exemple, on peut voir que le client WEB, indique la méthode ici GET ainsi que le user agent.



Voici un exemple d'en-tête HTTP de réponse du serveur WEB.
Comme vous pouvez le constater, on trouve beaucoup d'informations.
Ainsi le serveur WEB indique la date de la requête, le type de contenu, la taille, des informations sur le cache, etc.

```
root@ns3038035:~# curl -I -s https://www.malekal.com
HTTP/2 200
server: nginx
date: Tue, 09 Jul 2019 09:54:31 GMT
content-type: text/html; charset=UTF-8
x-sucuri-id: 19019
x-xss-protection: 1; mode=block
x-frame-options: SAMEORIGIN
x-content-type-options: nosniff
strict-transport-security: max-age=31536000
content-security-policy: upgrade-insecure-requests;
vary: Accept-Encoding
link: <https://www.malekal.com/wp-json/>; rel="https://api.w.org/"
last-modified: Tue, 09 Jul 2019 09:50:20 GMT
access-control-allow-origin: *
x-xss-protection: 0
referrer-policy: same-origin
x-fastcgi-cache: BYPASS
content-encoding: gzip
x-sucuri-cache: HIT

root@ns3038035:~#
```

Les méthodes

Le protocole HTTP fonctionne un ensemble de méthode.
Lors d'une requête HTTP par le navigateur WEB, ce dernier indique la méthode à utiliser dans l'en-tête.

Il en existe beaucoup mais voici les trois principales :

- **GET :** on demande une ressource au serveur WEB. En général, il s'agit du contenu d'une page WEB.
- **HEAD** : On demande QUE le contenu d'une en-tête.
- **POST : permet d'insérer ou mettre à jour une donnée.**

Le site de Mozilla donne les définitions de tout les méthodes HTTP : [Méthodes de requête HTTP]

Les données

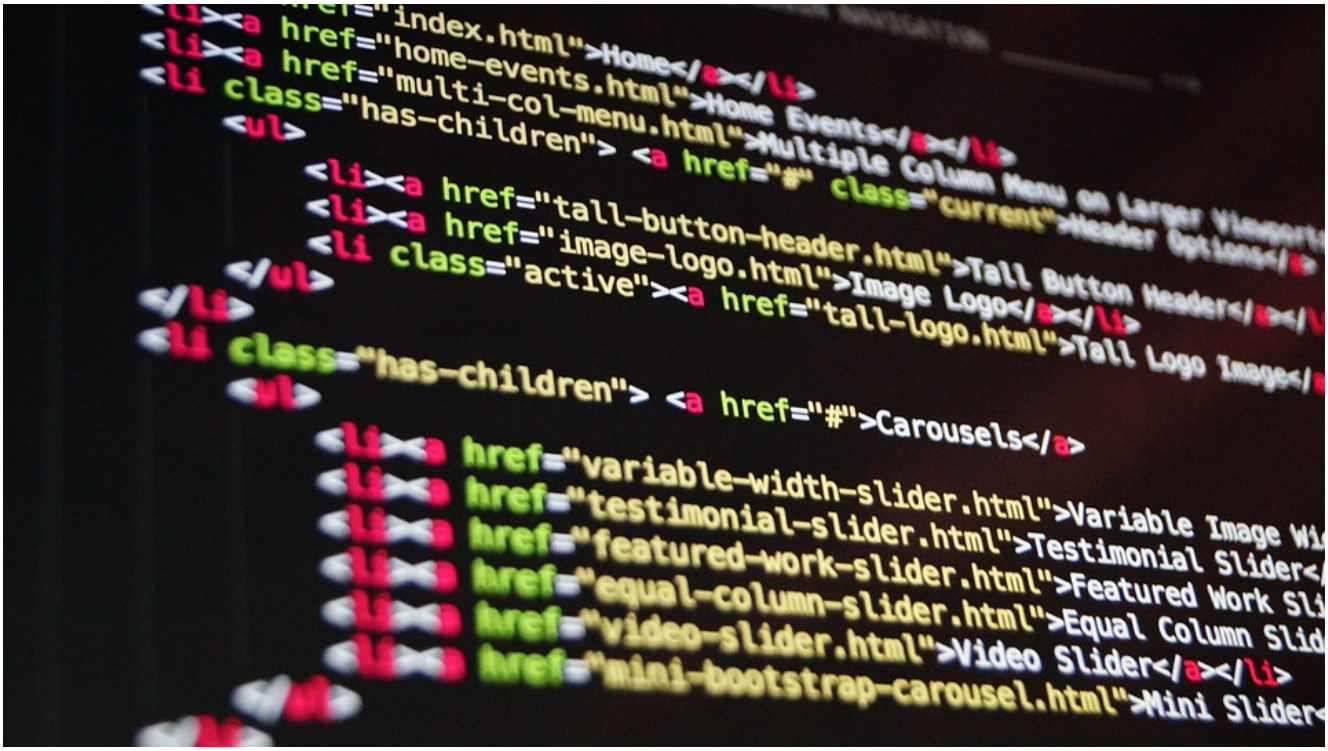
Rien d'extraordinaire là dessus puisqu'il s'agit du code de la page WEB.
D'ailleurs n'importe quel navigateur WEB peut l'afficher.

Les langages de programmation

On sait maintenant comment le client et le serveur communique.
Il faut maintenant qu'ils se comprennent en utilisant le même langage.

Plusieurs langages existent pour afficher des pages internet.

Un serveur WEB peut accueillir plusieurs langages de programmation.
Certains nécessite l'installation de serveur annexe qui se greffe alors au serveur WEB.
Bien sûr chaque langage a ses spécifications, versions et évoluent dans le temps.



Les langages standards

Les langages de programmes standard qu'un serveur WEB sait interpréter.

- **HTML (HyperText Markup Language)** : c'est le langage de base pour créer une page WEB. Il fonctionne avec des balises et peut appeler des images, créer des liens hypertextes, etc. Plusieurs versions existent depuis sa création. Aujourd'hui nous sommes à la version 5.
- **XHTML** (***Extensible HyperText Markup Language***) : Il se voulait le successeur de HTML basé sur le XML.
- **CSS (Cascading Style Sheets)** : Langage de présentation qui permet de donner la mise en forme d'un site WEB. Par exemple les couleurs, les tailles de polices, cadres etc. Le CSS se charge en début de page WEB dans l'en-tête.
- **JavaScript et Node.js** : Il s'agit d'un langage de script qui permet de créer des pages WEB interactives. A ne pas confondre avec Java.

Ainsi de nos jours, un site WEB utilisent au moins HTML, CSS et JavaScript.

Les langages supplémentaires qui peuvent se greffer à un serveur WEB.
Pour l'activer, l'administrateur doit installer un serveur supplémentaire.
Le serveur WEB se connectent alors à ce dernier pour interpréter ce langage.

Les langages supplémentaires

Il existe énormément de langage, voici les principaux :

- **ASP**
- **PHP**
- **Java**
- **Ruby**
- **Perl**
- **Python**

Le serveur utilise ces langages pour au final délivrer du contenu HTML.

Du côté du navigateur internet, cela peut nécessiter l'installation d'application particulière.
Par exemple pour Java, cela peut nécessiter d'installer une application Java sous la forme d'un plugin.

Hébergement mutualisé et CMS

Tout cela paraît compliqué mais des applications clés en main existent.
En effet pour créer un site internet, il faut :

- Enregistrer un domaine auprès d'**un registrar**. Par exemple malekal.com
- On déclare **les entrées DNS** du domaine, par exemple www.malekal.com. Le but est de faire correspondre

l'adresse IP du serveur WEB à l'adresse littérale.
- Il faudrait ensuite prendre un serveur dédié pour y installer le serveur WEB.
- Ensuite on configure ce dernier.

L'article détaille ces termes.

Comment fonctionnent les sites internet

Tout cela est donc compliqué.

Afin d'éviter de devoir gérer un serveur entier, les hébergeurs proposent de l'hébergement mutualisé.
Il s'agit d'une proposer souvent à travers une interface de gestion la gestion d'un serveur WEB.
L'utilisateur déclare le site et envoie les fichiers sur ce dernier.
Le site est alors mis en ligne de manière automatique.

De même pour les bloggeurs, on trouve les **CMS (content management system)** pour système de contenu.
Ces derniers permettent de mettre en ligne du contenu de manière très simple.
En effet, il s'agit d'une application qui permet de rédiger des articles.
WordPress et **Joomla** sont les principaux CMS.
Par exemple, le site malekal.com utilise WordPress.