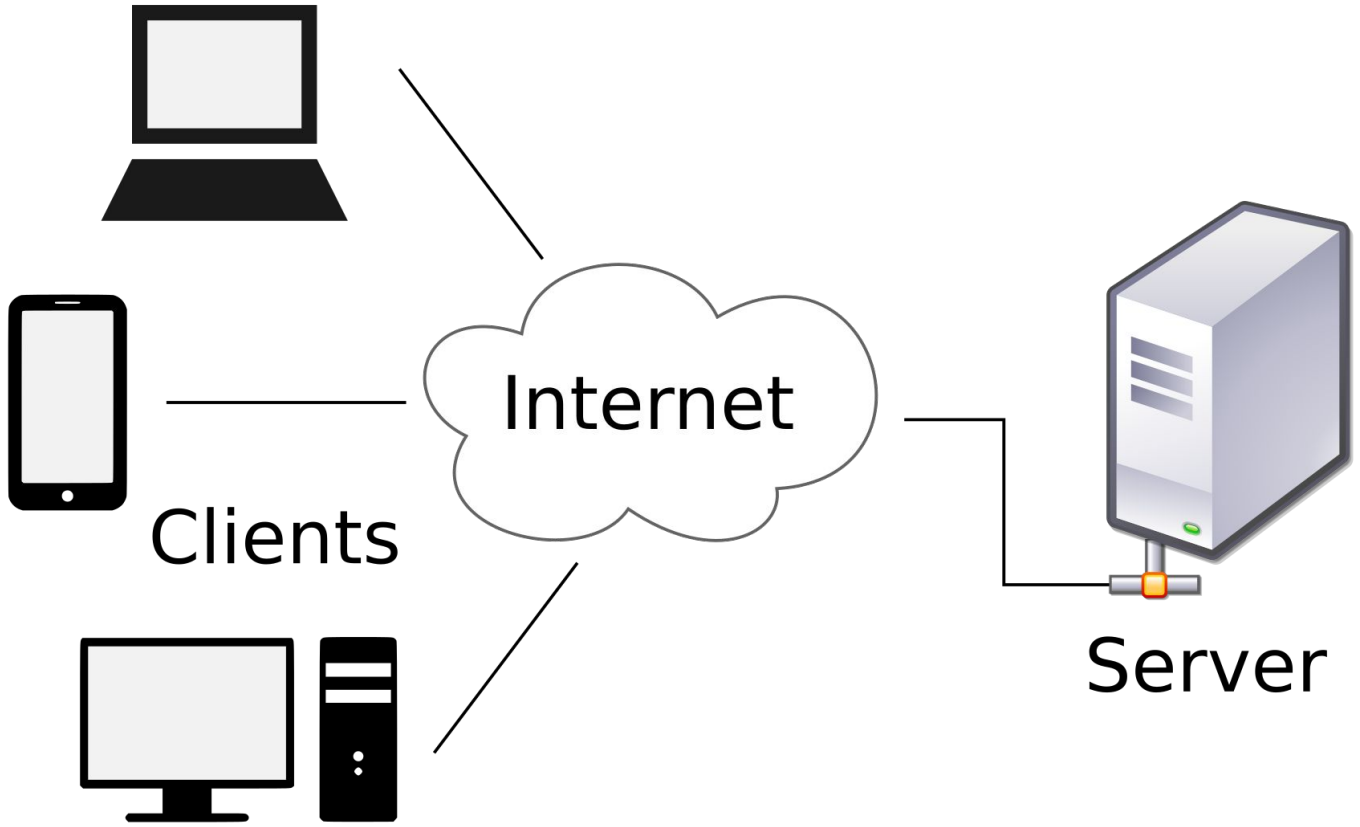
Développement Web: Rappels

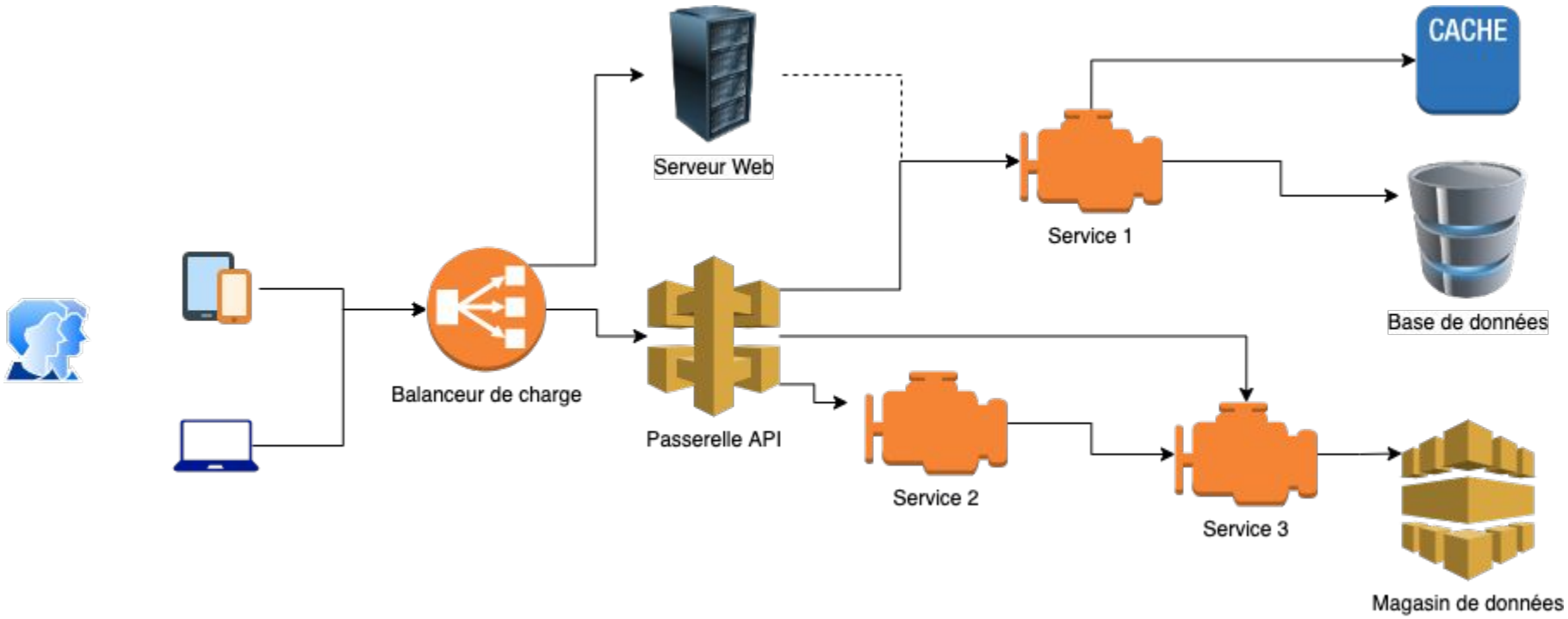
Plan et objectifs

- Réviser les bases du web
- Réviser le protocol HTTP
- Rappel des technologies de base en programmation web
- Différencier les différents types de site web
- Comprendre l’architecture en 3 couches

Architecture Client-Serveur



Architecture Web



Protocol HTTP

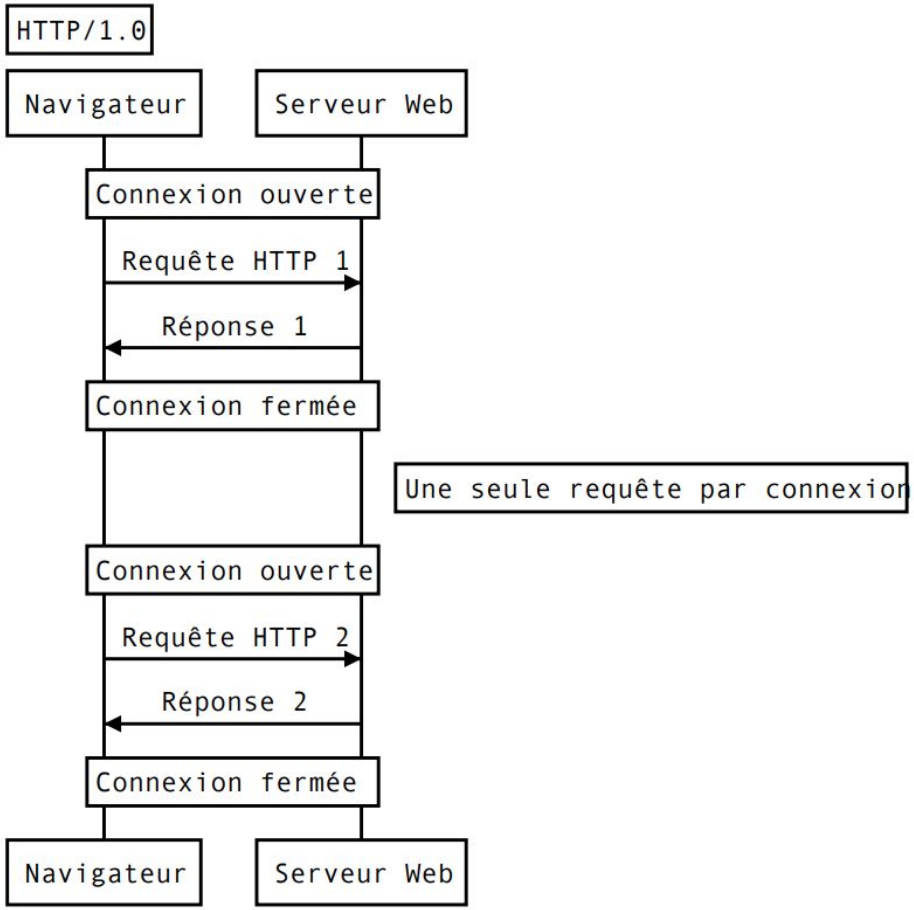
- Transfert de contenu (HTML, CSS, javascript)
- Protocol de type requête - réponse
- Port par défaut: 80
- Une requête par ressource
- Principalement utilisé par les navigateurs web, mais aussi pour les **API**s

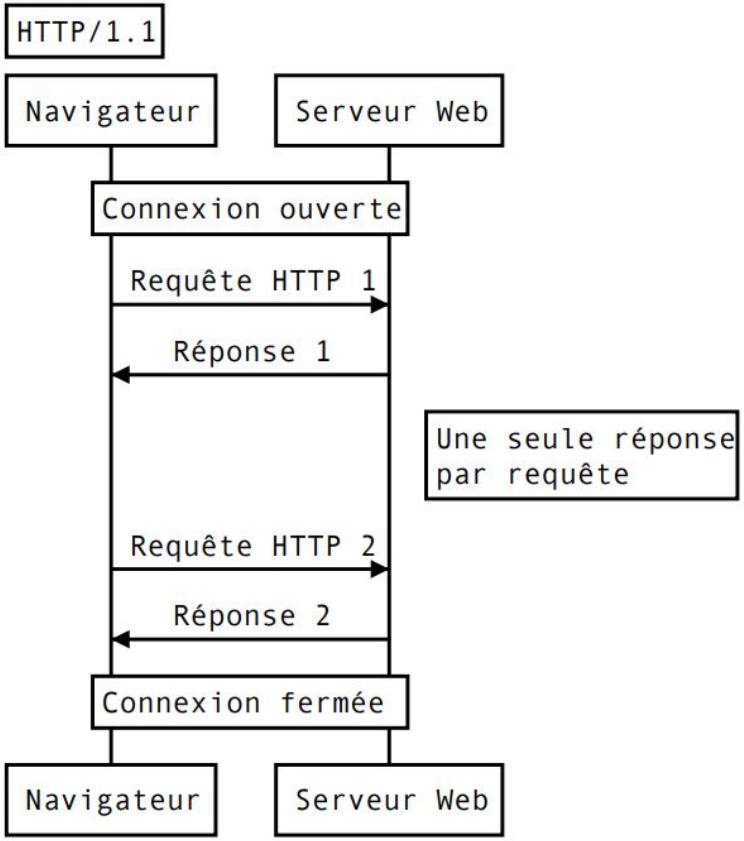
HTTPS est la version sécurisée

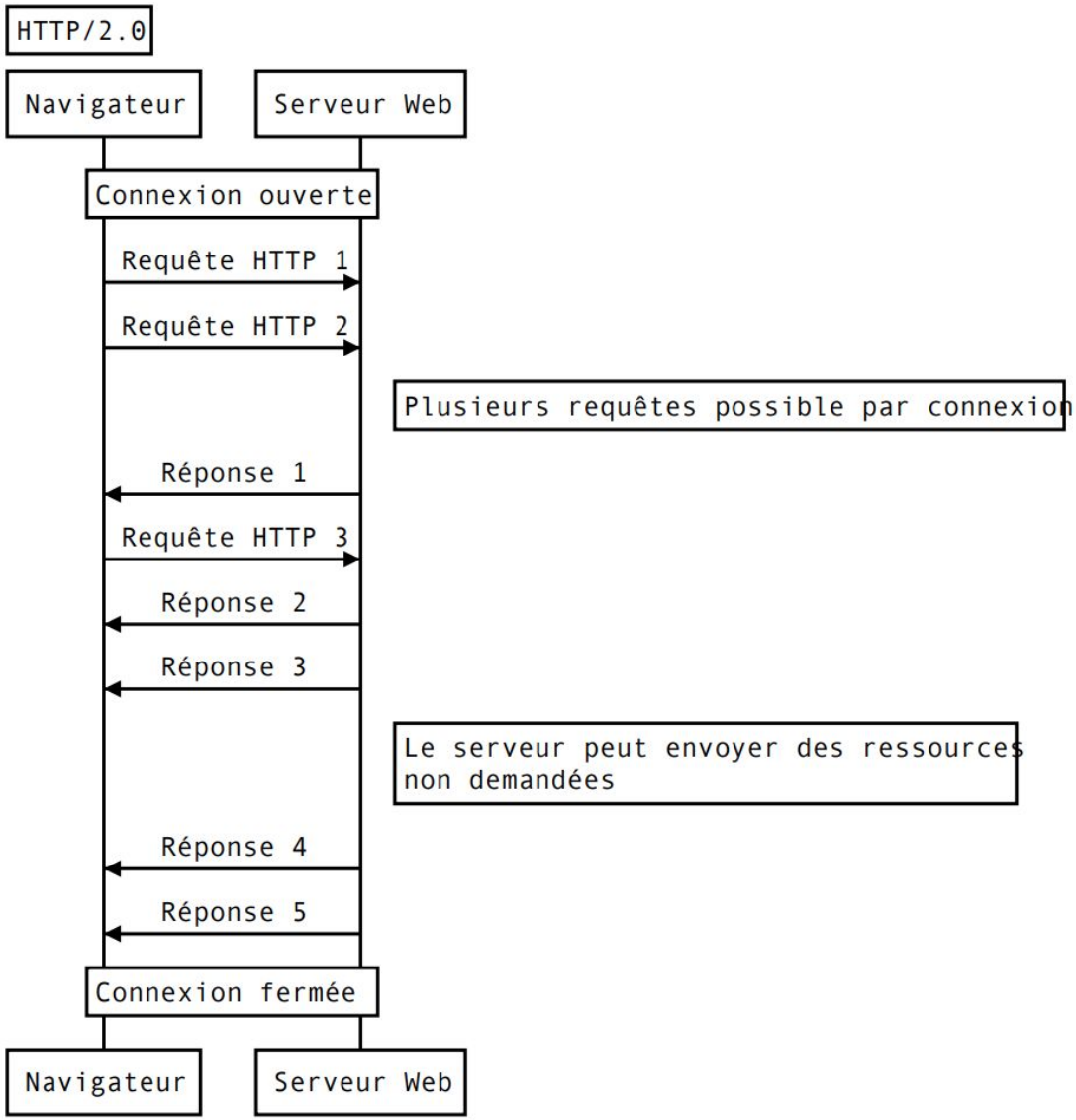
- HTTP avec une couche de chiffrement et d'authentification
- Port par défaut: 443

Protocol HTTP(S)

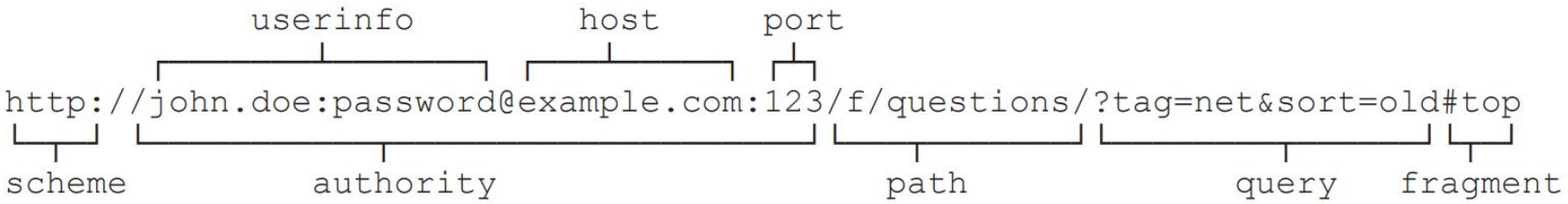
- HTTP 1.0
  - connexions non persistantes
  - pas de réponses par morceau (chunk transfer)
- HTTP 1.1
  - réutilisation de connexions (keep-alive)
  - réponses par morceau (chunk transfer)
  - réponses par plage (range transfer)
- HTTP 2.0
  - compression des en-têtes
  - multiplexage de requêtes
  - Server Push
- HTTP 3.0
  - utilisation du protocol QUIC au lieu de TCP


HTTP/1.0
Navigateur
Serveur Web
Connexion ouverte
Requête HTTP 1
Réponse 1
Connexion fermée
Une seule requête par connexion
Connexion ouverte
Requête HTTP 2
Réponse 2
Connexion fermée
Navigateur
Serveur Web


HTTP/1.1
Navigateur
Serveur Web
Connexion ouverte
Requête HTTP 1
Réponse 1
Une seule réponse par requête
Requête HTTP 2
Réponse 2
Connexion fermée
Navigateur
Serveur Web


HTTP/2.0
Navigateur
Serveur Web
Connexion ouverte
Requête HTTP 1
Requête HTTP 2
Plusieurs requêtes possible par connexion
Réponse 1
Requête HTTP 3
Réponse 2
Réponse 3
Le serveur peut envoyer des ressources non demandées
Réponse 4
Réponse 5
Connexion fermée
Navigateur
Serveur Web

Anatomie du chemin d’accès d’une requête HTTP



Protocol de communication: HTTP(S)

7

```
<!DOCTYPE html>… (here come the 29769 bytes of the requested web page)
```

source: https://developer.mozilla.org/fr/docs/Web/HTTP/Overview

Méthodes

Nom	Description
GET	Obtenir une ressource. Ne devrait pas modifier une ressource.
HEAD	Similaire à GET, mais sans la réponse du serveur. Sert à récupérer les en-têtes (*headers*).
POST	Soumettre une entité à une ressource (ex: envoi de formulaire). Changement d'état possible.
PUT	Remplace la ressource existante avec le *payload* envoyé.
DELETE	Supprime une ressource spécifique.

Voir https://developer.mozilla.org/fr/docs/Web/HTTP/Methods

En-têtes

*Les **en-têtes** HTTP permettent au client et au serveur de **transmettre des informations supplémentaires** (métadonnées) avec la requête ou la réponse.*

Voir https://developer.mozilla.org/fr/docs/Web/HTTP/Headers

En-têtes - exemple

Requête

accept: text/html,application/xhtml+xml

accept-encoding: gzip

accept-language: fr-CA,fr;q=0.9,en-US;q=0.8,en;q=0.7

cookie: …

user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36

Réponse

content-encoding: gzip

content-language: en

content-type: text/html; charset=utf-8

set-cookie: ...

Cookie

Le protocole HTTP est *sans état.*

Comment peut-on passer un contexte au serveur pour qu'il puisse:

- savoir qui envoie la requête (session)
- conserver des préférences
- faire le suivi de l'usager (métriques, parcours)

Le cookie est une information (max 4K, max 20 par domaine) générée par le serveur et passée au client via l'en-tête **set-cookie**.

Pour les requêtes subséquentes, le client retourne les cookies dans les en-têtes de la requête via **cookie**.

Cookie - Exemple

Réponse

```
HTTP/1.0 200 OK

Content-type: text/html

Set-Cookie: yummy_cookie=choco

Set-Cookie: tasty_cookie=strawberry
```

Requête

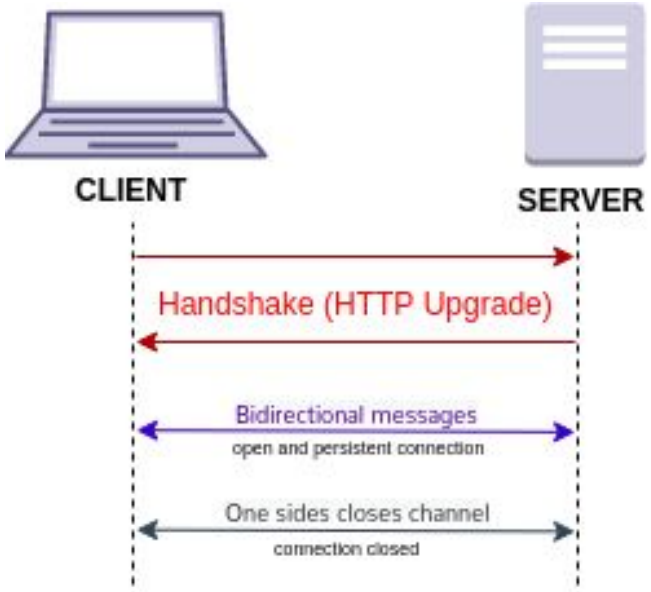
```
GET /sample_page.html HTTP/1.1

Host: www.example.org

Cookie: yummy_cookie=choco; tasty_cookie=strawberry
```

WebSocket

WebSocket *est un standard du Web désignant un protocole réseau visant à créer des canaux de communication* ***full-duplex*** *par-dessus une connexion* ***TCP*** *pour les navigateurs web.*

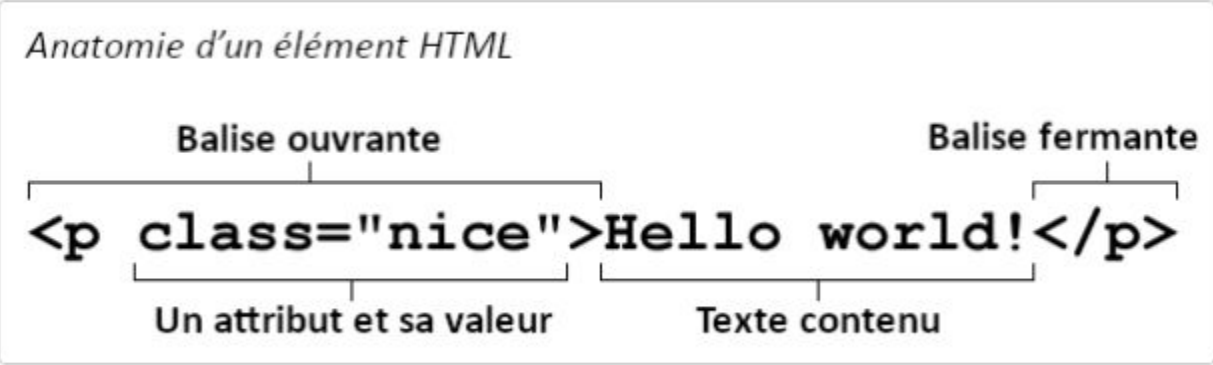
Utilise le schéma **ws** ou **wss** (e.g. ws://www.example.com)

Utile pour la communication temps réel entre le client et le serveur.



HTML

HTML *(HyperText Markup Language) est un langage descriptif qui* ***définit la structure*** *d'une page web.*



Voir https://developer.mozilla.org/fr/docs/Glossary/HTML

CSS

CSS *(Cascading Style Sheets ou en français "Feuilles de style en cascade") est utilisé pour* ***styliser*** *et* ***mettre en page*** *des pages Web.*

```
/* Le sélecteur "p" indique que tous les paragraphes dans le document
seront affectés par la règle */
p {
  /* La propriété "color" (couleur) définit la couleur du texte. */
  /* Ici, cette couleur sera le jaune (yellow en anglais) */
  color: yellow;

  /* La propriété "background-color" (couleur d'arrière-plan) définit la
couleur d'arrière-plan */
  /* Ici, cette couleur sera le noir (black) */
  background-color: black;
}
```

Voir https://developer.mozilla.org/fr/docs/Glossary/CSS

Javascript

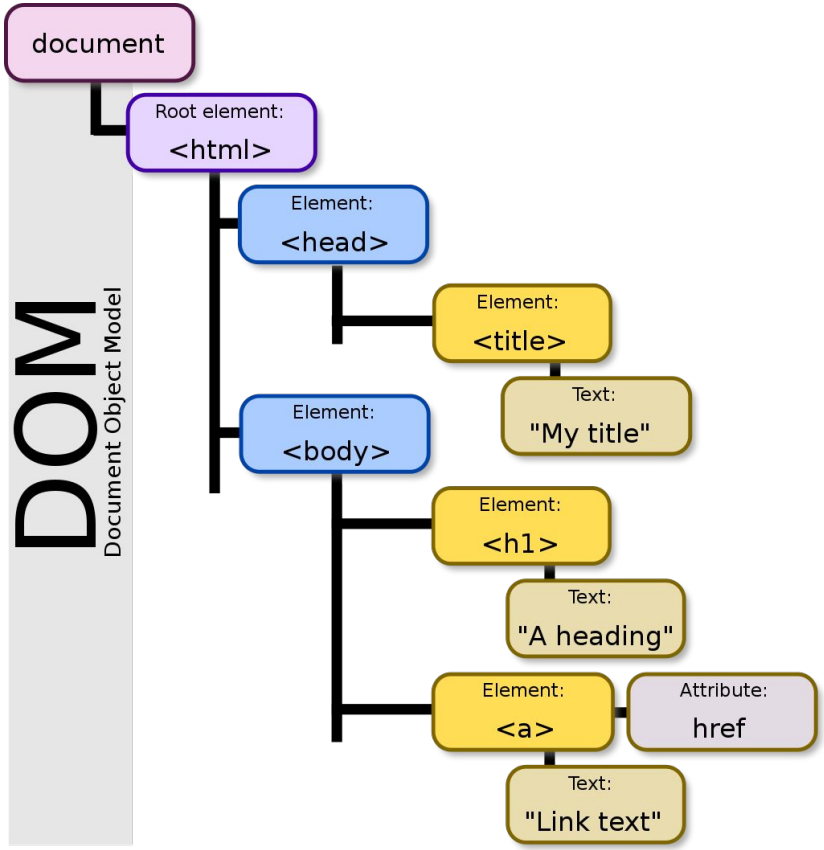
JavaScript *(souvent abrégé en « JS ») est un langage de script léger, orienté objet, principalement connu comme le langage de script des pages web.*

- Implémentation de la norme ECMAScript.
- Initialement conçu comme langage pour la programmation côté client, mais maintenant aussi utilisé côté serveur grâce à certains cadriciels (Node.js).
- Les dernières fonctionnalités ne sont pas nécessairement supportées par tous les navigateurs, ce qui peut demander une transpilation pour générer le code final.

Voir https://developer.mozilla.org/fr/docs/Web/JavaScript

Le DOM (Document Object Model)

- Interface de programmation pour les documents HTML
- Groupe structuré (arbre) de nœuds et d'objets possédant différentes propriétés et méthodes
- Lien avec le javascript qui peut modifier le DOM ou réagir aux événements et ainsi rendre la page dynamique



Outils de développement

Plusieurs navigateurs offrent des outils de développement très utiles pour déboguer et apprendre.

Démo - https://www.perdu.com/

- Voir les requêtes HTTP - onglet Network
- Voir les éléments du DOM et modifier leur contenu - onglet Elements
- Voir les classes CSS et modifier leur contenu - onglet Elements
- Jouer sur le site avec du javascript - onglet Console
  - https://gist.github.com/coderunner/9e4d9a12c966b1200a626bc1f56c9f38

Démo - Cookie

- https://github.com/coderunner/vigilant-octo-potato

Site Web vs Application Web

Un **site web** offre du contenu aux utilisateurs.

- même contenu pour tous les visiteurs
- simple à construire et maintenir
- plusieurs outils disponibles (wordpress, joomla, wix, squarespace, etc)
- en général, la création demande peu de connaissance en programmation

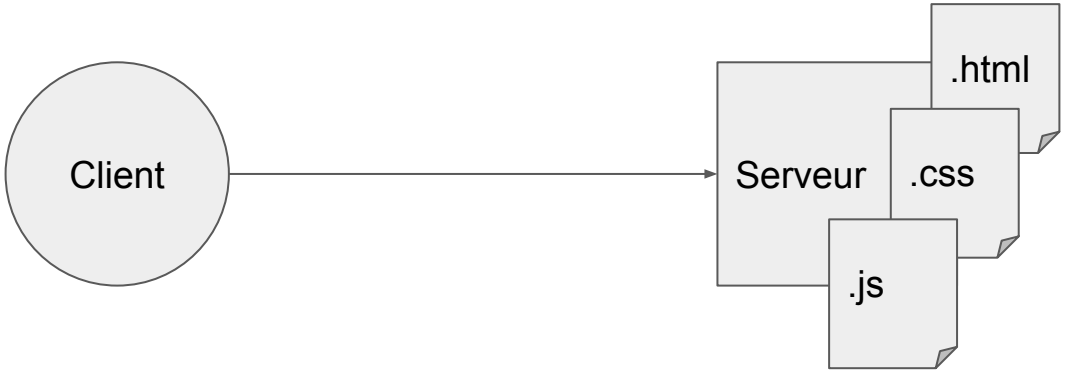
Une **application web** offre aux utilisateurs des outils pour accomplir une ou plusieurs tâches spécifiques.

- logiciel qui s'exécute dans un navigateur
- axé sur les intéractions avec l'utilisateur
- plusieurs cadriciels disponibles (django, angular, react, laravel, etc)
- l'application est parfois un produit (SaaS)
- en général, la création demande beaucoup de connaissance en programmation

Sites statiques

Un site statique sert toujours les mêmes fichiers (HTML, CSS, js)

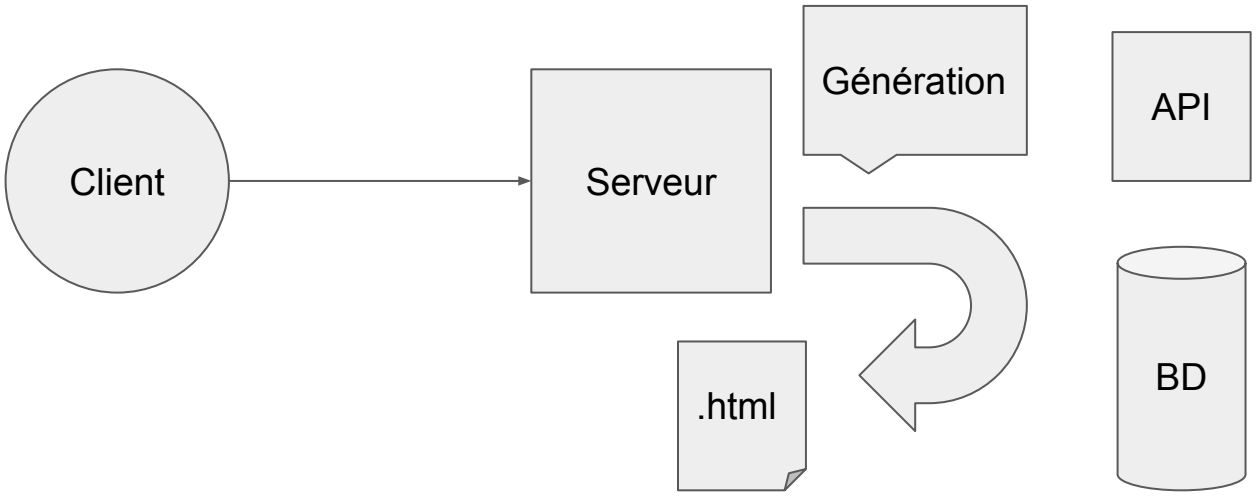
- simple à bâtir
- simple à déployer
- le contenu de la page peut être intéractif (avec du code js)
- rapide (le contenu est pré-généré)
- excellent pour le SEO
- outils de génération disponibles (jekyll, hugo, etc)



Sites dynamiques

Le **serveur** génère les pages web dynamiquement en réponse aux interactions de l'usager.

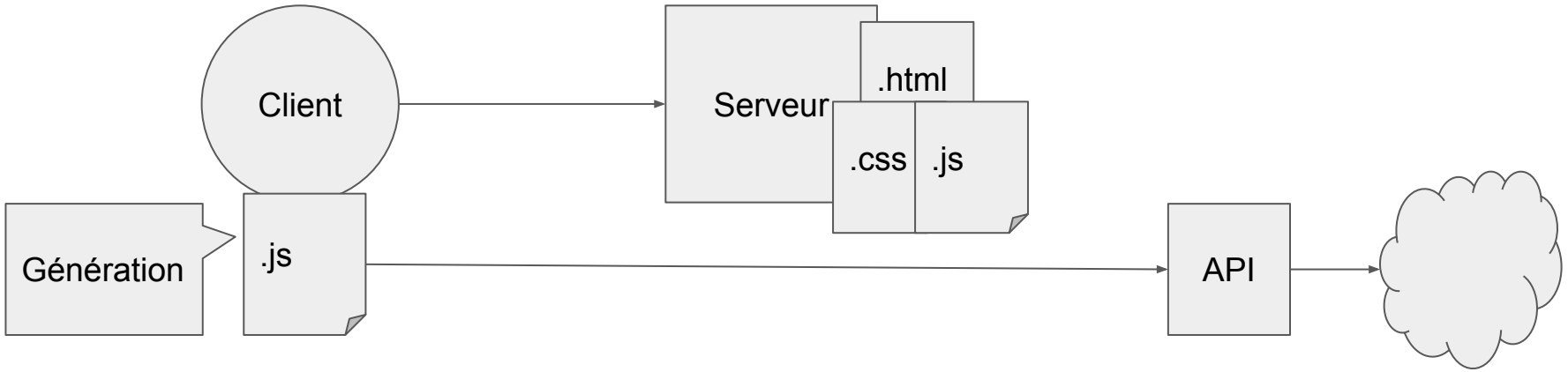
- logique de présentation gérée par le serveur
- requiert l'utilisation d'une cadriciel côté serveur (laravel, django, etc)
- bon pour le SEO (contenu final retourné aux clients)
- moins rapide (aller-retour vers le serveur pour la génération)



Application Monopage - SPA

Le client télécharge un ensemble de fichiers statiques depuis le serveur, puis le code javascript modifie l'apparence de la page dynamiquement et effectue des appels aux APIs pour charger (et modifier) les données.

- logique de présentation gérée par le client (manipulation du DOM)
- requiert souvent l'utilisation d'une cadriciel côté client (angular, react.js, etc)
- difficile pour le SEO
- rapide (changement de l'IU sans recharger la page)



Architecture en 3 couches (3-tier architecture)

L'architecture logique du système est divisée en trois couches

- couche de *présentation - UI, navigateur (e.g. js, Angular, bootstrap)*
- couche de *traitement - logique, serveur (e.g. php, spring, flask)*
- couche de *données - état persistant, base de données (e.g. mysql, mongo db)*

L’objectif est de diviser les responsabilités pour structurer le système et faciliter les modifications (principe de responsabilité unique).

Chaque couche utilise des technologies qui lui sont parfois spécifiques (e.g. les bases de données).

Nous discuterons de chacune de ces couches dans les prochains cours.

Architecture en 3 couches (3-tier architecture)