# Introduction au Web

**Père d' Internet (pour faire simple) : Vint Cerf** (1943 - Américain)

**TCP-IP - années 70**

C'est le **protocole de communication bas niveau entre les ordis.**

Arpanet 1969 - DARPA en 1976 (Defense Advanced Research Projects Agency).

**Père du Web : Tim Bernes-Lee** (1955 - Anglais)

**HTML - 1991**

le **langage de base du WEB**

Créé au CERN en 1989-1992 (conseil Européen pour la Recherche Nucléaire).

# 1 : Qu’est-ce que le WEB

- **WEB** = World Wide Web = **WWW**
  - Le WEB : surfer, **naviguer** de page en page.
  - Le WEB : des **ordinateurs connectés entre eux, en réseau.** Pas de nœud central. Une toile d’araignée géante.
  - Le WEB : c’est **le HTML**, né en **1991**, au CERN (Tim Berners-Lee).
  - Le WEB : c’est **l’hypertexte** : des documents liés entre eux.
  - Quand le WEB arrive, Internet avait déjà 20 ans mais n’était pas généralisé.
  - Le WEB a tout changé. En quelques années, des millions de sites WEB ont été créés.

# Qu’est-ce que le WEB

- **Le WEB c’est : HTML + Navigateur + CSS + JavaScript**
  - **HTML** : Hyper Text Markup Language : Langage de balises hypertexte.
    - Le HTML perest le langage qui définit la structure des documents hypertextes.
    - Le HTML est simple : il permet de définir des titres, des paragraphes, des listes à points, des images et des liens hypertexte.
  - **Navigateurs** : ils permettent d’afficher les documents HTML. Ils proposent un style par défaut.
    - A l'aide de fichiers CSS, on peut personnaliser l'apparence des pages.
    - Au début : pas de CSS. Le style était dans la page HTML. Pas de JavaScript : on ne pouvait que créer et afficher des documents.
  - **CSS** : a permis de personnaliser l'apparence des pages HTML.
    - Le HTML s’occupe du contenu (paragraphe, titre, liste à points, image, lien hypertexte)
    - Le CSS s’occupe du style (mise en page, couleur, etc.).
  - **JavaScript** : a été créé chez Netscape (un navigateur) pour permettre de l'interactivité sur les pages Web. Il a été rapidement adopté par différents navigateurs.
    - 20 ans plus tard : JavaScript est un langage totalement différent, très puissant, que nous pouvons utiliser pour créer des applications.

# Qu’est-ce que le WEB

- **Le WEB : c’est le navigateur d’accès aux serveurs**
  - Les **3 tâches principales d'un navigateur** sont :
    - **afficher une page HTML**, qu’elle provienne du réseau ou de votre machine
    - **interagir avec l’utilisateur**, via l’écran ou via la barre d’adresse,
    - **envoyer des requêtes au serveur** et récupérer ses réponses
  - Les navigateurs offrent aux développeurs des outils de développement, c’est-à-dire un moyen de mieux contrôler et déboguer les pages Web et les applications Web.
    - Les outils de développement sont un outil essentiel pour les développeurs Web.

# Qu’est-ce que le WEB

- **Un navigateur WEB : c’est une plateforme d’exécution des programmes**

  - **A l'origine**, les navigateurs étaient de **simples outils de visualisation** de documents :

    - Ils affichaient du simple HTML.

  - **Aujourd’hui**, les navigateurs permettent de **faire tourner des applications Web** puissantes et complexes.

    - Google Maps et Gmail, entre autres, ont ouvert la voie.

  - **Historique** : en **1995, JavaScript** a été développé et ajouté au navigateur Netscape.

    - Avec le JavaScript, le navigateur est un « **runtime** » : une plateforme qui exécute d'autres programmes et permet de réaliser des applications Web.

# Qu’est-ce que le WEB

- **Un navigateur WEB : c’est une plateforme d’exécution des programmes**

  - Il faut que t**ous les navigateurs parlent le même langage** ! Donc, **ils partagent une norme** définit par des organisations :

    - Le World Wide Web Consortium (**W3C**) définit la **norme du HTML-CSS** pour tous les navigateurs.

    - **Ecma International** définit la **norme du JavaScript** pour tous les navigateurs.

  - Ces organisations sont chargées de définir et de faire progresser le HTML, le CSS, le JavaScript et tous les services (on dit API) fournies par les navigateurs.

    - Exemples de services standards :
      - Le HTML 5 permet la lecture d’audio et de video
      - DOM permet d’interagir avec la page HTML
      - Canvas permet de dessiner
      - Géolocalisation permet de localiser l’utilisateur
      - Fetch permet de faire des requêtes asynchrones
      - Web storage permet de stocker des données côté client
      - etc.

# Qu’est-ce que le WEB

- **Le WEB** : **c’est une partie d’ INTERNET**

  - **INTERNET** = **WEB** + **FTP** (échanges de fichiers sur le réseau) + **EMAIL** + **NEWS GROUP** (ancêtres des forums) + d’autres services.

  - Aujourd’hui, on accède aux services via une interface web : c’est le **Cloud**.

- **Historique**

  - **1969 : ARPAnet** (ancêtre d’internet), réseau militaire décentralisé.

  - 1972 : Email

  - 1974 : le réseau est militaire et universitaire.

  - **1991 : Le WEB : HTML 1.** Tim Berners-Lee. Il va fonder le **W3C** (World Wild Web) pour assurer une normalisation au WEB (normalisation des navigateurs).

# 2 : Organisation du WEB

## Le réseau

- **Le WEB : c'est le réseau**

  - Le WEB, ce sont **des ordinateurs reliés entre eux** dans un réseau.

  - **Lorsque qu'on ouvre une page** avec un navigateur comme Chrome, Firefox ou Safari, sur un smartphone ou un ordinateur, **une demande est adressée à un serveur.**

  - **Un serveur c'est un programme sur une machine « quelque part » sur le réseau**. Ce quelque part, c'est **le « cloud »**.

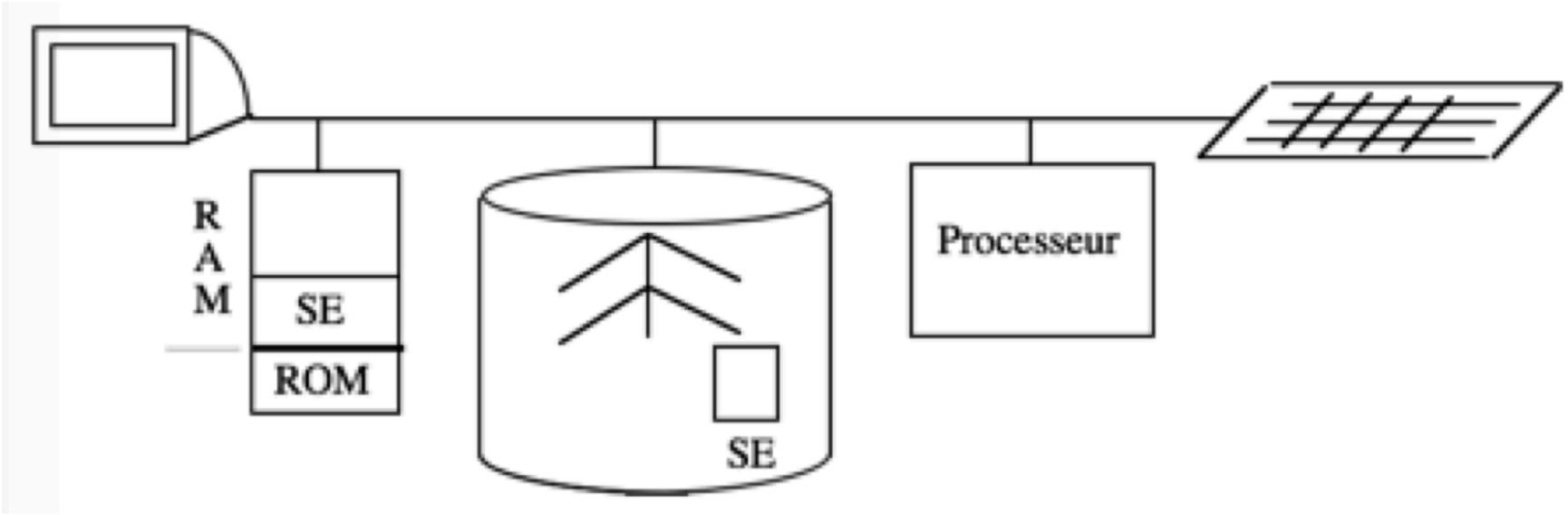
  - Pour aller de notre machine à la machine où se trouve ce qu'on veut, des protocoles de communication ont été définis : IP, TCP, HTTP, HTTPS

  - Le serveur répond à la demande en renvoyant (le plus souvent) un document texte au format HTML que tous les navigateurs savent interpréter.

  - Le document HTML peut contenir des liens vers d'autres ressources : des fichiers CSS, des polices, des fichiers JavaScript.

# Organisation du WEB
# architecture d'un ordinateur

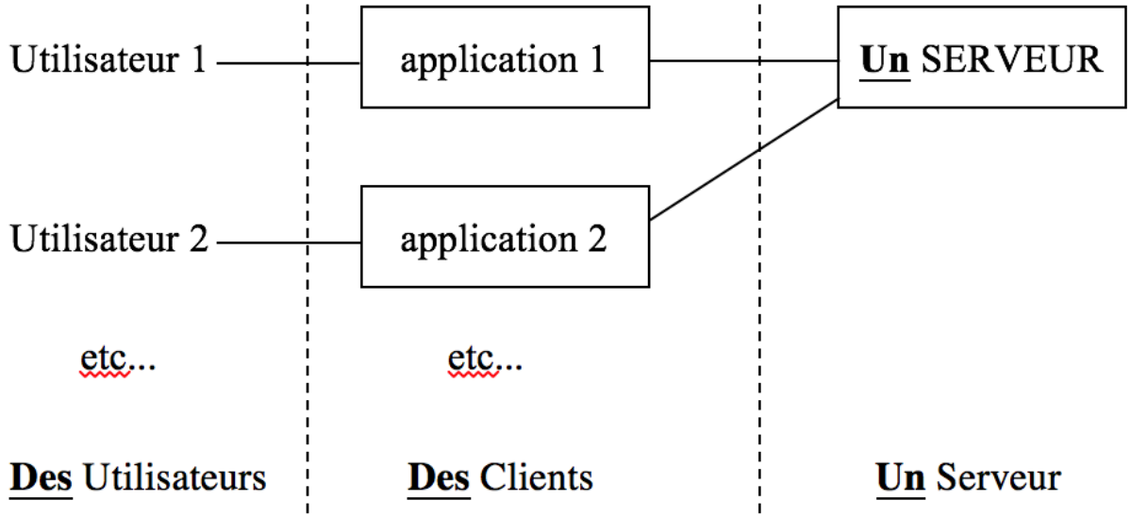
- **Architecture et fonctionnement d'un ordinateur :**




- ROM : mémoire « morte » : le bout de mémoire qui permet de démarrer l'ordi
- RAM : mémoire « vive » : la mémoire qui sert de « bureau » : là où on met les applications qu'on fait tourner sur notre machine. On peut y mettre Word, Excel, un navigateur, etc.
- SE : Système d'Exploitation = OS : Operating System. Le programme qui fait marcher notre machine.
- Disque dur : la mémoire de stockage, organisée avec une arborescence de dossier et de fichier. On y trouve le SE qui est copié sur la RAM quand l'ordi démarre.
- Processeur : le moteur qui fait tourner nos programme.

# Organisation du WEB
# architecture Client – Serveur

- **Sur le réseau :** certaines machines sont **des clients**, certaines machines sont **des serveurs**.
- **Le serveur** : un ordi puissant qui **sert les clients :** des PC en tout genre (ou d'autres serveurs en tant que client).
- **Architecture client serveur :**

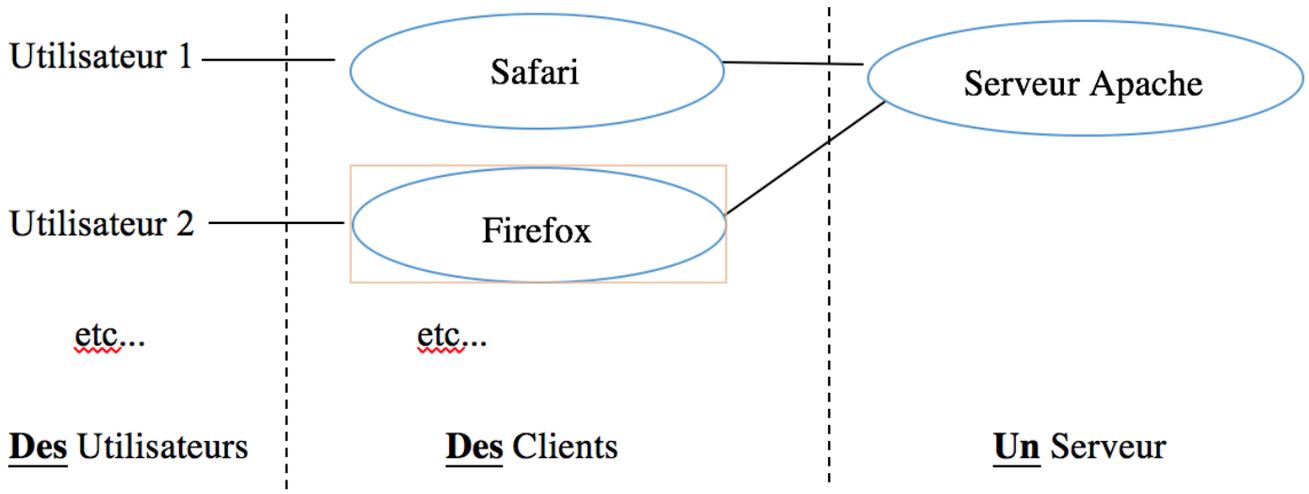



- **Exemple de serveur : le site bliaudet.free.fr client serveur :**
  - En vrai, ce n'est pas le site qui est le serveur, mais un programme, le **serveur « Apache »** qui permet d'accéder aux fichiers qui constituent le site.

# Organisation du WEB

# architecture Client – Serveur

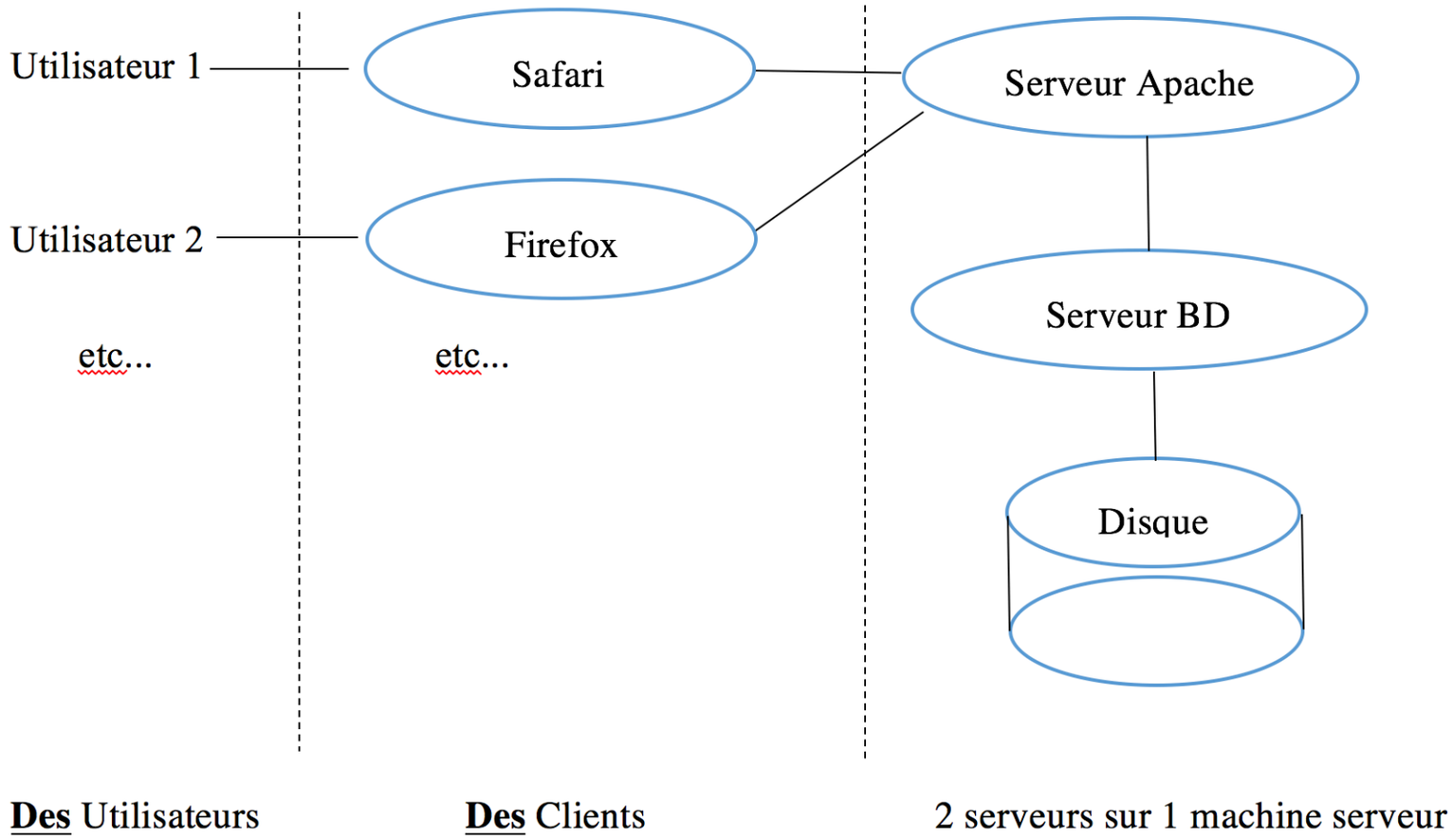
- **Architecture client-serveur WEB :**




- Il existe de nombreux types de serveurs. On va s'intéresse ici aux **serveurs WEB** (Apache est un serveur WEB)
  - Dans ce cas, le client est généralement un navigateur Web, qui appelle le serveur à la demande de l'utilisateur, soit via du HTML, soit via du JavaScript.
  - Mais ça peut aussi être un programme exécuté sur un ordinateur, à la recherche d'informations sur le WEB (par exemple une mise à jour).
- On peut créer son serveur en le programmant. C'est plus ou moins compliqué selon les langages. C'est facile en JavaScript à l'aide de Node.js ou en Python.

# Organisation du WEB
# architecture Client – Serveur

- **Architecture client-serveur avec BD :**




- Le serveur WEB (Apache ici) joue le rôle de client pour le Serveur de BD.
- Si **les deux serveurs sont sur 2 machines** différentes, on parle **d'architecture 3-tiers**.

# Organisation du WEB
# architecture Client – Serveur

- **PRECISIONS :**

- **Machine client, programme client**
  - **Un client est un programme** qui prend l'initiative de communiquer avec un serveur.
  - Une machine client est une machine sur laquelle « tourne » un programme client

- **Serveur, machine serveur, programme serveur :**
  - **Un serveur est un programme** qui n'a pas d'interface utilisateur et qui ne prend pas l'initiative de communiquer avec d'autres programmes.
  - Un serveur est un programme qui attend qu'un autre programme lui demande quelque chose : il attend les demandes d'un client.
  - Par extension, un serveur est une machine sur laquelle « tourne » un programme serveur.

- **Rôle du serveur web**: envoyer des données au client que le client restitue dans un navigateur.

.

# 3 : Les langages du WEB

## Les langages côté client

- **3 langages** connus par la machine du **client** : **HTML, CSS, JavaScript**
  - **HTML** : du texte et des balises. Les balises : le code donné au navigateur pour qu'il s'y retrouve et qu'il comprenne ce qu'il doit faire du texte (un paragraphe, un titre, du gras, etc.).
  - **CSS** : partie de HTML pour la mise en page
  - **JavaScript** : pour faire des calculs et de la programmation côté client

- **Rôle du navigateur :**
  - Le navigateur est un programme installé sur le PC qui récupère du HTML, CSS, JavaScript (HCJ) et l'affiche à l'écran.
  - 5 navigateurs principaux : Firefox, Safari, Chrome, Internet Explorer devenu Edge, Opera. **Privilégier Firefox et Chrome.**

# Les langages du WEB

# Les langages côté serveur

- **Présentation :**
  - La programmation côté client, avec le navigateur, est limitée. C'est le web 1.0 : on affiche une information fixe. Les mises à jour sont faites par programmation.
  - Pour avoir des sites dynamiques qui échangent de l'information avec les utilisateurs, il faut des langages serveur.
  - Le serveur s'occupe du comportement. Le client de la présentation.
  - Langages client : HTML, CSS, JavaScript : comment le site doit se présenter. Le JavaScript peut aussi s'occuper de comportement.
  - Langages serveur : PHP, java, python, ruby, C# : comment le site doit se comporter
- **Serveur WEB :**
  - Le client avec son navigateur ne comprend que le HTML-CSS et le javascript. Il ne comprend pas les langages serveur.
  - Sur la machine serveur, un programme qui tourne en permanence se charge de traduire les langages serveur en langage client HTML-CSS-Javascript. C'est le serveur WEB.
  - Le client demande la page web. Le serveur fabrique la page web. Le serveur envoie la page fabriquée au client.
- **Selon les langages serveur, il y a différents serveurs WEB :**
  - **PHP :** serveur apache - **Java :** serveur tomcat - **C# :** serveur IIS (2IS) - **JavaScript :** serveur node.js

# Les langages du WEB

# Framework et CMS

- **Framework :**
  - Un Framework est un outil (des bibliothèques, une architecture) pour faire plus rapidement un site web.
  - On trouve des Framework pour chaque langage.
  - Il faut être développeur pour utiliser un framework
- **Principaux Framework selon les langages côté client :**
  - CSS: bootstrap
  - JavaScript : JQuery
- **Principaux Framework selon les langages côté serveur :**
  - PHP : symfony, zend, etc.
  - Python : Django
  - C# : ASP .NET
  - Java : J2E
  - Ruby : ROR
- **CMS :**
  - Un CMS (Content Management System) est un site web clé en main qu'on n'a plus qu'à paramétrer.
  - On n'a pas besoin d'être développeur pour utiliser un CMS.
  - Quelques CMS : Wordpress, Dotclear, Joomla, SPIP, etc.
  - Pour utiliser Wordpress efficacement, mieux vaut connaître HTML, CSS et PHP.

# Les langages du WEB

# 5 - Base de données

- **Présentation :**
  - Les données (les utilisateurs, les produits, les achats, les mots de passe, etc.) sont rangées dans une base de données.
  - La base de données peut être vue comme un ensemble de tables excel.
- **SQL : le langage des base de données :**
  - L'accès à l'information se fait avec un langage : le SQL
  - SELECT id, name, login FROM users ORDER BY id DESC
  - SELECT * FROM visiteurs WHERE inscription='01-01-2015' and pays='BELGIQUE'
  - Ca peut être simple mais aussi très compliqué.
- **Le SGBD : le serveur d'accès aux bases de données :**
  - Un SGBD est un programme qui gère la base de donnée.
  - C'est un serveur : le serveur de base de données. Il fonctionne en mode client – serveur. Ses clients seront les serveurs web.
- **Les principaux SGBD :**
  - MySQL, PostGreSQL, MariaDB : pour les PME.
  - SQL Server : c#
  - ORACLE : pour les institutions
  - SQLite : Pour les petites BD. Toutes les infos sont dans un seul fichier. Performances faibles.

# Les applications du WEB

## Site responsive *versus* application mobile

- **Site web responsive :**
  - Adaptation de l'application WEB en fonction de la largeur de l'écran.
  - Pas toujours idéalement adapté aux smartphones.
  - Un site qui marche sur n'importe quel support avec un seul langage (un langage serveur) : moins cher, bien adapté aux besoins de l'entreprise
  - Les technologies responsives sont à maturité

- **Applications mobiles :**
  - Bien adaptés pour des applications faites pour les smartphones : meilleure expérience utilisateur, meilleure ergonomie.
  - Chaque application doit être développée avec le langage correspondant à l'OS du smartphone : iOS (swift), androïd (java), windowsPhone (c#), etc.
  - Technologies « cross-platform » pour permettre un développement unique pour toutes les plateformes : exemple : React-Native en JavaScript.

# Introduction au Web

**2**

## Les réseaux derrière le Web

# Les réseaux derrière le Web

# 1 - Des serveurs et des câbles

- **Le web :**
  - Le web est constitué de **tous les ordinateurs clients** (les PC) et de **tous les ordinateurs serveurs** qui hébergent les sites web.
- **Datacenter :**
  - Un data-center **héberge des centaines de serveurs** (machines serveur) qui tournent 24/24.
  - Il y a des data-centers **partout dans le monde**. Ce sont eux qui envoient les pages HTML aux clients.
  - Sécurisé, climatisé, fibre optique pour envoyer vite l'info
  - **Baie de serveurs**. On « racke » les serveurs dans la baie. Pas d'écran dans la baie, parfois un « KVM ».
- **Câbles :**
  - Les **clients** et les **serveurs** sont **reliés entre eux par des câbles**.
  - Des câbles sous-marins à travers l'atlantique. Exemple : l'Algérie avait 2 câbles qui la reliaient au web. Un câble a été coupé par une ancre. Plus d'internet, c'était le câble principal (80%).
- **FAI :**
  - **Les ordinateurs individuels passent par un FAI** (fournisseur d'accès à internet) qui les relie au web.

# Les réseaux derrière le Web

# 1 - Datacenter - serveur et baie de serveurs

Un serveur :

Plusieurs serveurs « rackés » dans une baie de serveur

# Les réseaux derrière le Web

# 1 - Datacenter - baies de serveurs et console KVM

Des baies de serveurs collées les unes à côté des autres

Une « console KVM » (écran clavier) pour accéder aux serveurs

# Les réseaux derrière le Web

# 1 - Datacenter - des serveurs connectés

Des serveurs connectés

# Les réseaux derrière le Web

# 1 - Datacenter : des bâtiments dédiés

Des datacenters :

*Une vue aérienne du Campus de Datacenter de DATA4, à Paris-Saclay, le plus puissant d'Europe (100 MW), relié à plus de 110 Data Destination comme AWF et Azure*

# Les réseaux derrière le Web

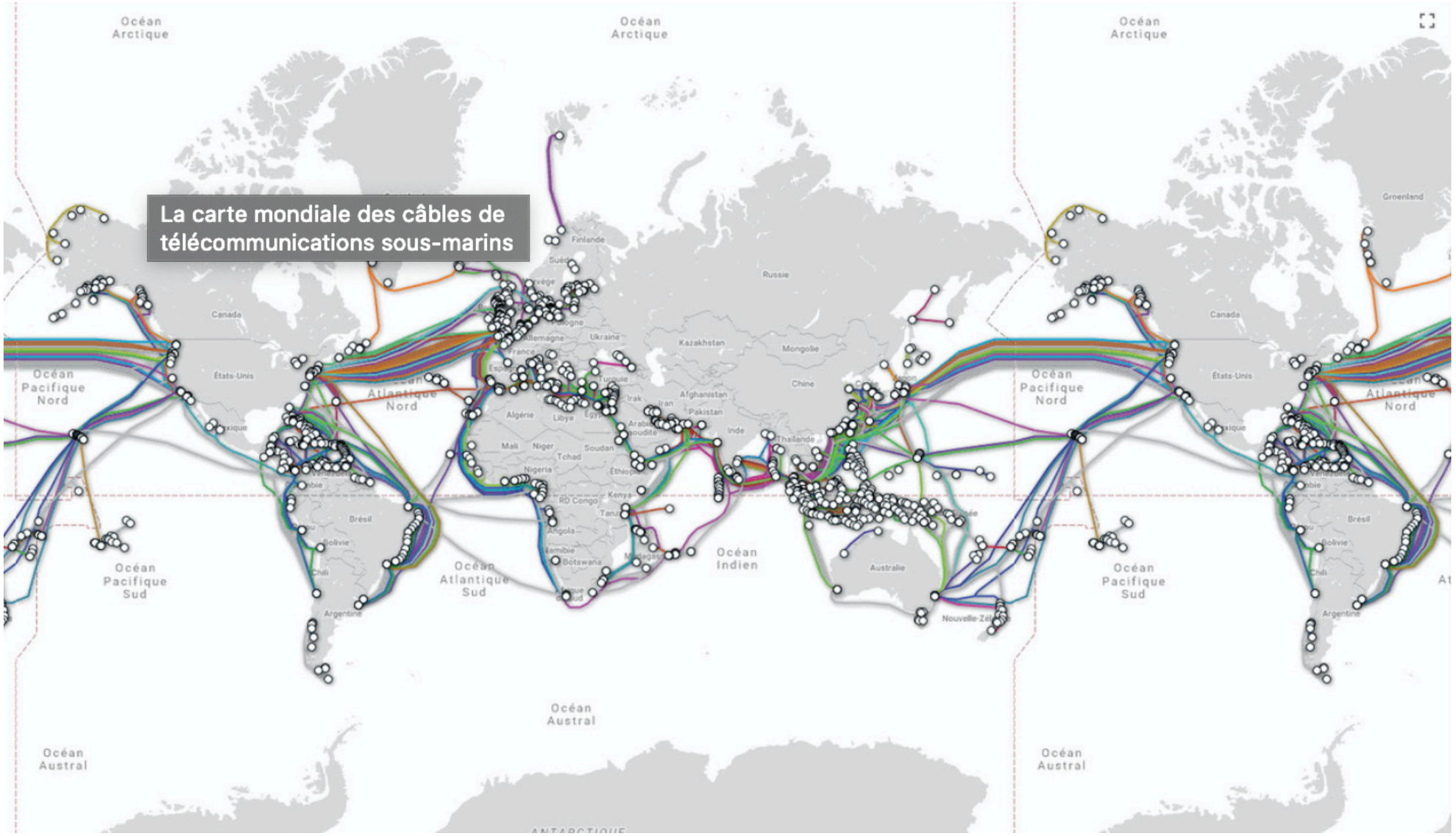
# 1 - Les datacenters dans le monde

https://www.datacentermap.com

# Les réseaux derrière le Web

## 1 - Les datacenters  - Les câbles sous-marins dans le monde

https://www.eurafibre.fr/la-carte-mondiale-des-cables-de-telecommunications-sous-marins/



# Les réseaux derrière le Web

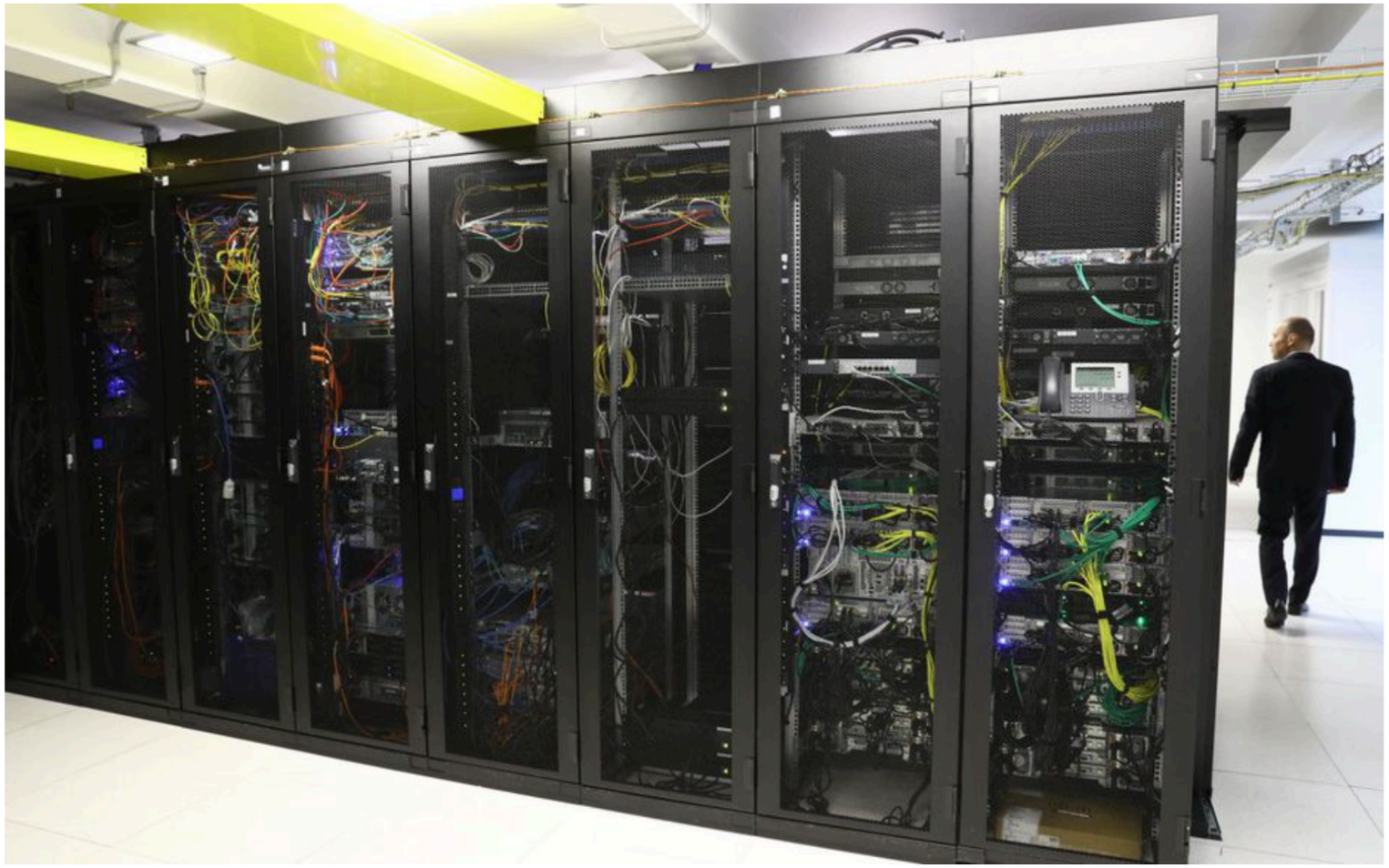
# 1 - Les datacenters  - les données sensibles

Sans data center, pas d'Internet.
Leur rôle majeur explique le secret qui entoure ces lieux, fermés au public.

https://www.leparisien.fr/

## Data centers : mais où se trouvent vos données ?

Les data centers hébergent  vos données les plus sensibles. Les connaissez-vous ?



# Les réseaux derrière le Web

## 2 – Les OS

- **Les serveurs** sont des ordinateurs avec leurs **système d'exploitation (OS, operating system) :**
    - **monde Unix-Linux** (équivalent MacOS)
    - **monde Microsoft**

- Pour microsoft, les OS s'appellent : **Windows Server** (avec l'année en cours).

- Pour le monde **Unix-Linux** : Ubuntu, Red-Hat, Suse, Debian, etc.

# Les réseaux derrière le Web

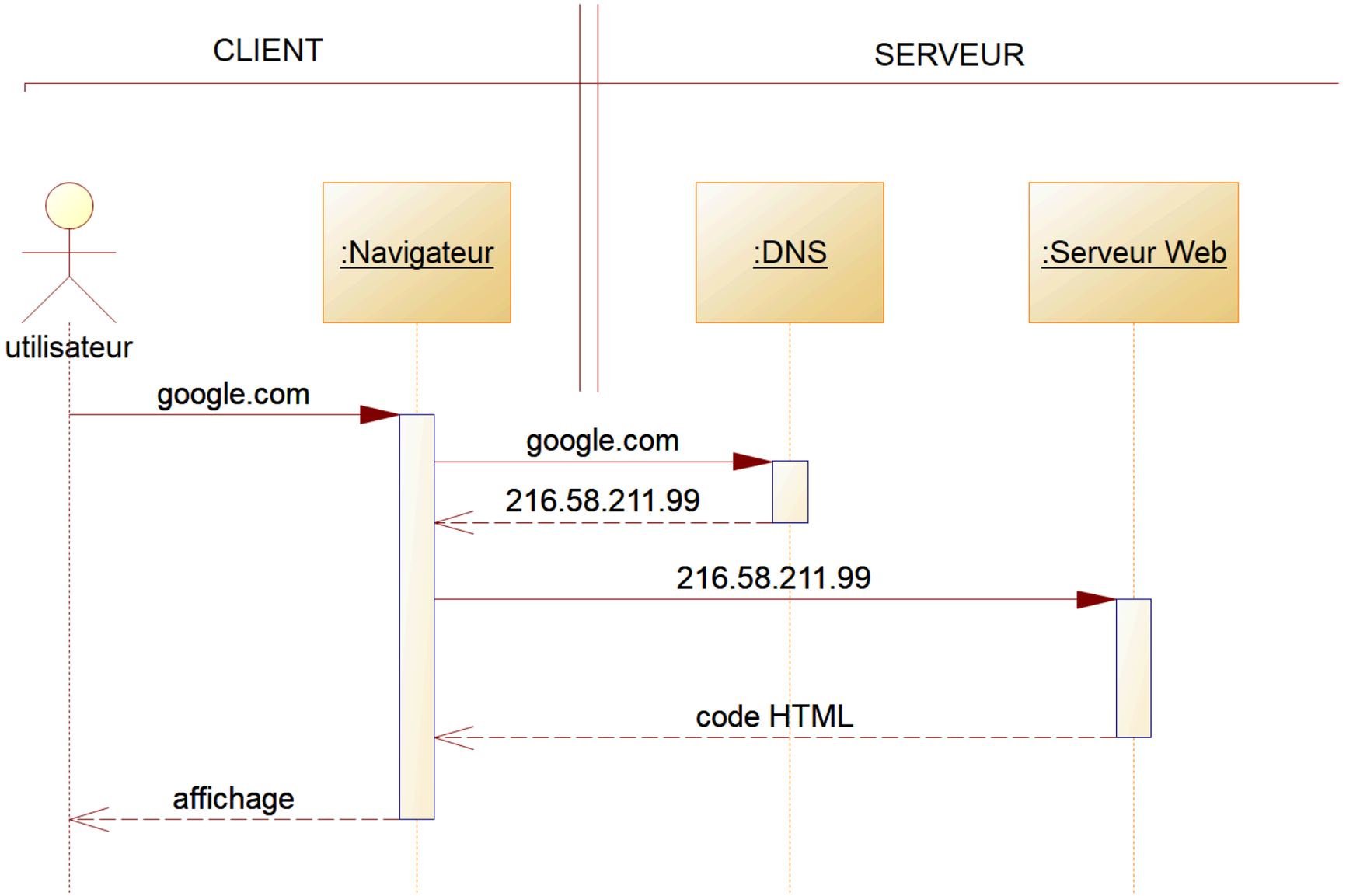
# 3 - IP, nom d'hôte, nom de domaine, DNS

- **Le problème :**
  - Comment on fait pour **retrouver les ordis** (il y en a des milliards) ?
- **Adresse IP :**
  - L'adresse IP identifie les machines : **plaque d'immatriculation**.
  - **216.58.211.99 : c'est google**. 157.240.21.35 : c'est facebook. Je peux mettre l'adresse IP ou le plus explicite comme https://www.google.com ou https/www.facebook.com
- **Nom de domaine, serveur DNS :**
  - Le serveur DNS **(Domain Name System)** associe un nom en langage courant à une adresse IP.
  - **Résolution de noms de domaines** : c'est la corrélation entre adresse IP et nom de domaine. C'est ce que fait le serveur DNS.
  - On récupère l'adresse IP avec la commande « ping google.com »

# Les réseaux derrière le Web

# 3 - IP, nom d'hôte, nom de domaine, DNS

- **Architecture Client - Serveur**
  - Comment on accède au site google.com à partir de son navigateur :



# Les réseaux derrière le Web

# 4 - Les protocoles de communication

- **Le problème : comment les ordis communiquent entre eux ?**
  - Les **protocoles de communication** servent à ça. Un protocole est une **langue commune** entre les machines

- **Protocole bas niveau : TCP/IP, UDP :**
  - **TCP/IP** : protocole de communication du Web pour faire transiter pages web, emails, news groupe, etc. Avec AR.
  - **UDP** : autre protocole, sans AR (Accusé de Réception) : j'envoie sans attendre la réponse.

- **9 protocoles de haut niveau, basés sur TCP, dont :**
  - **http** : pour faire transiter des pages web. http://bliaudet.free.fr utilise le protocole http. On envoie une requête avec un GET.
  - **https** : idem en crypté : https://www.lemonde.fr
  - **ftp** : pour l'envoi de fichiers. ftp://user:mdp@ftpperso.free.fr utilise le protocole ftp avec un « user » et un « mdp » (mot de passe).
  - **smtp** : pour l'envoi d'emails

- **Protocoles de très haut niveau :**
  - **API**, **REST**, etc. De la communication entre applications.

# Introduction au Web

**3**

## ARCHITECTURE LOGICIELLE

# Architecture

# Fonctionnement général d'un site web

## Fonctionnement Client/Serveur de base :

1. Le client demande sa page au serveur

2. Le serveur web traite la page demandée :
   - Si elle ne contient que du code client (HTML-CSS-JavaScript), elle n'est pas modifiée et sera envoyée telle quelle au client.
   - Si elle contient du code serveur (du PHP par exemple), ce code est traduit en code client pour construire la page à envoyer au client qui ne contiendra que du code client.
   - Si le code serveur contient du code BD, ce code est envoyé au serveur de BD et le serveur web récupère les réponses du serveur BD pour construire la page à envoyer au client.

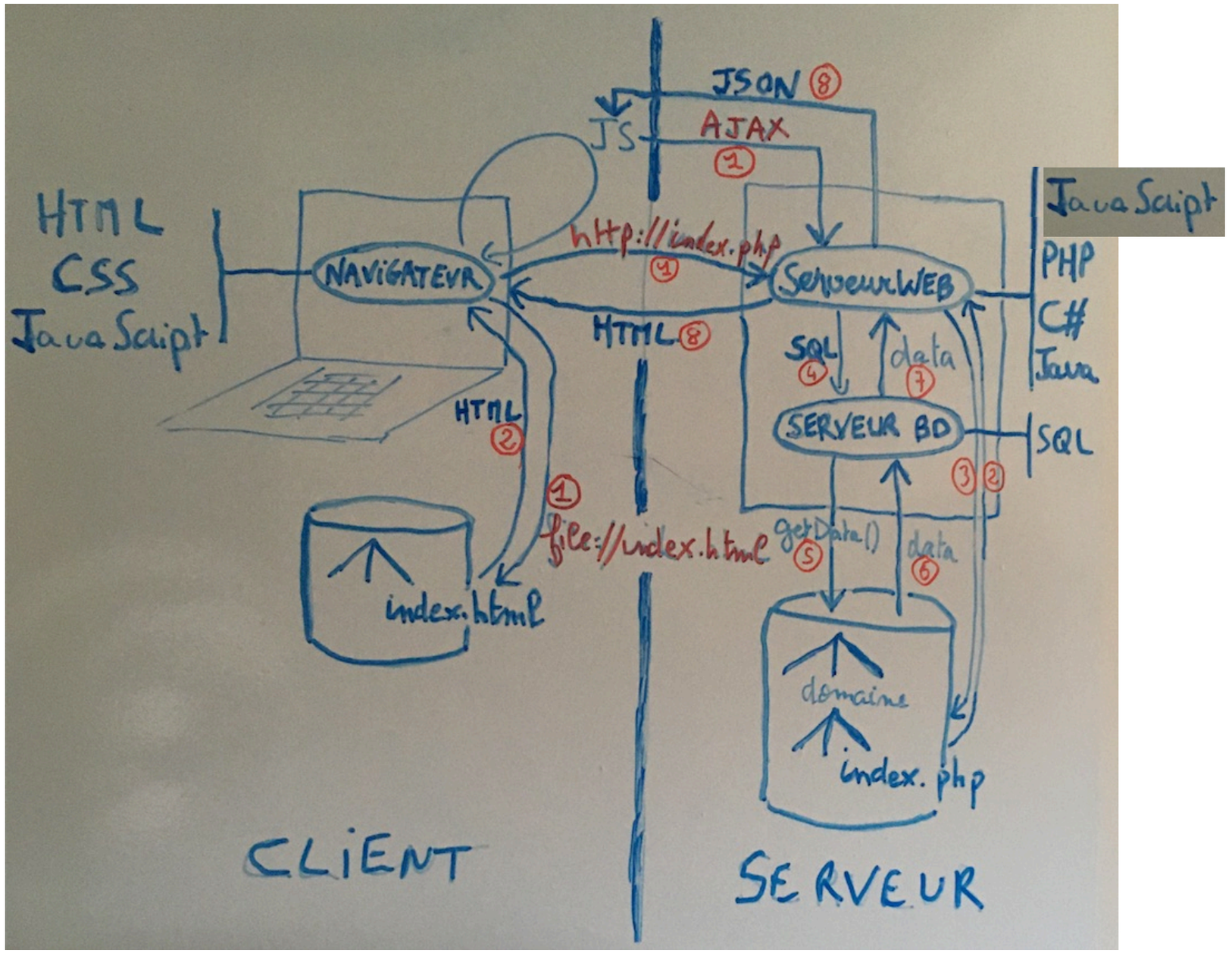
3. Quand le serveur a fini la génération de la page à envoyer au client (HTML-CSS-JavaScript), il l'envoie au client.

4. Le navigateur du client affiche le HTML et CSS. Le code JavaScript est traité par le navigateur à la demande.

# Architecture

# Fonctionnement d'un site web

## 3 cas possibles : local, langage serveur, AJAX



# Architecture

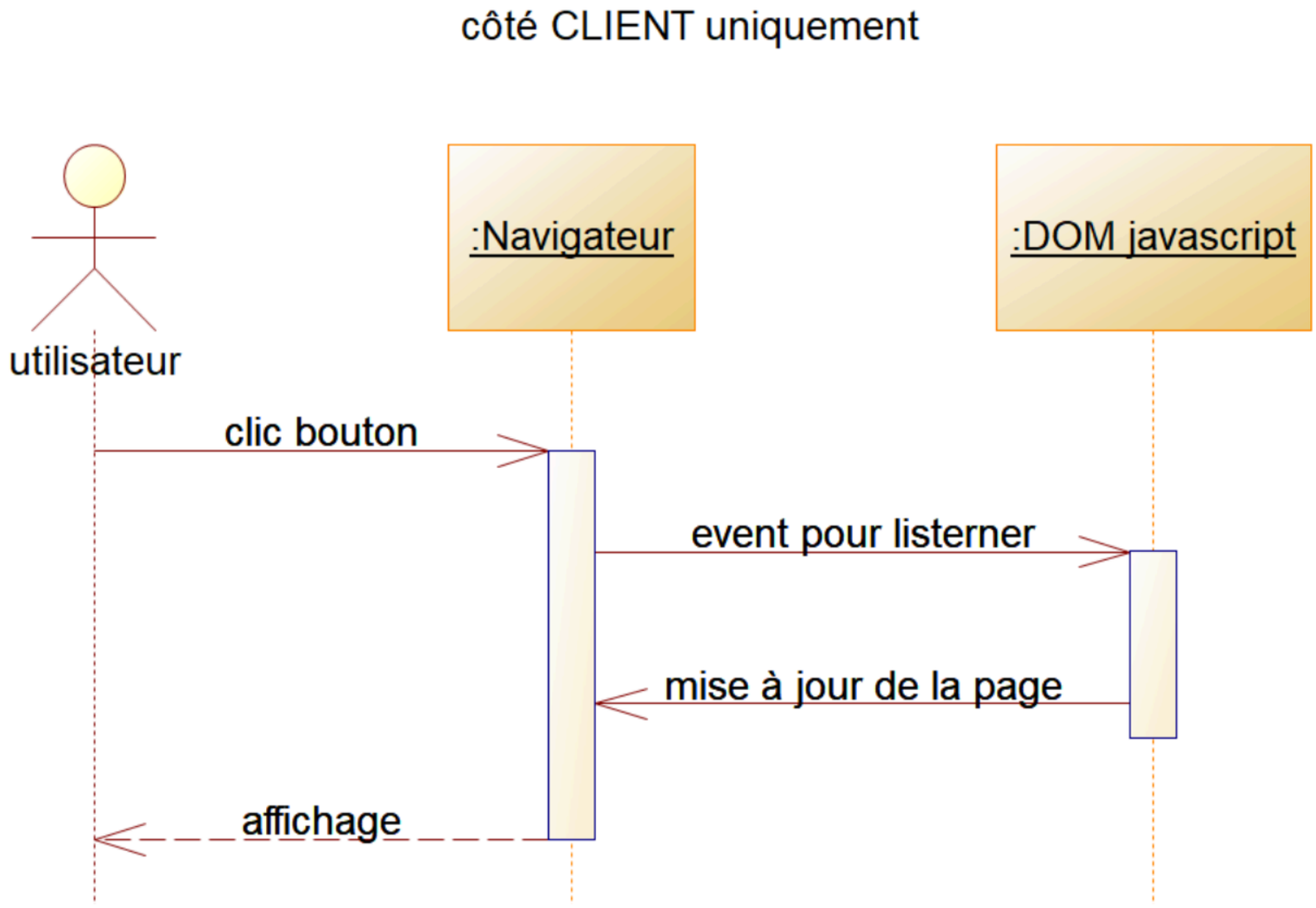
# Fonctionnement d'un site web

1. **Cas 1** : Le client ouvre une page « **en local** » : chemin **file://index.html** (1 à 2)
2. **Cas 2** : La page HTML client demande **une page au serveur** : chemin **http://index.php** (1 à 8)
   - Le serveur web construit la réponse HTML (2 à 7) :
     - Il récupère le fichier php
     - Il le traduit en HTML
     - Si nécessaire, il communique avec le serveur de BD pour récupérer de l'information dans la BD ou mettre à jour de l'information dans la BD.
   - Une fois la réponse HTML construite, le serveur l'envoie au client qui affiche une nouvelle page.
3. **Cas 3** : La page HTML du client sollicite son **code JavaScript qui va solliciter le serveur** (1 à 8)
   - Le code JavaScript demande une page au serveur en utilisant de l'AJAX : c'est asynchrone.
   - Le serveur va construire sa réponse selon le même principe que dans le cas 2.
     - La réponse est en général au format JSON.
     - La réponse est récupérée en asynchrone par le JavaScript.
   - Le JavaScript met à jour la page HTML dans changer de page.

# Architecture

# Cas 1 : HTML côté client, sans réseau (théorique)

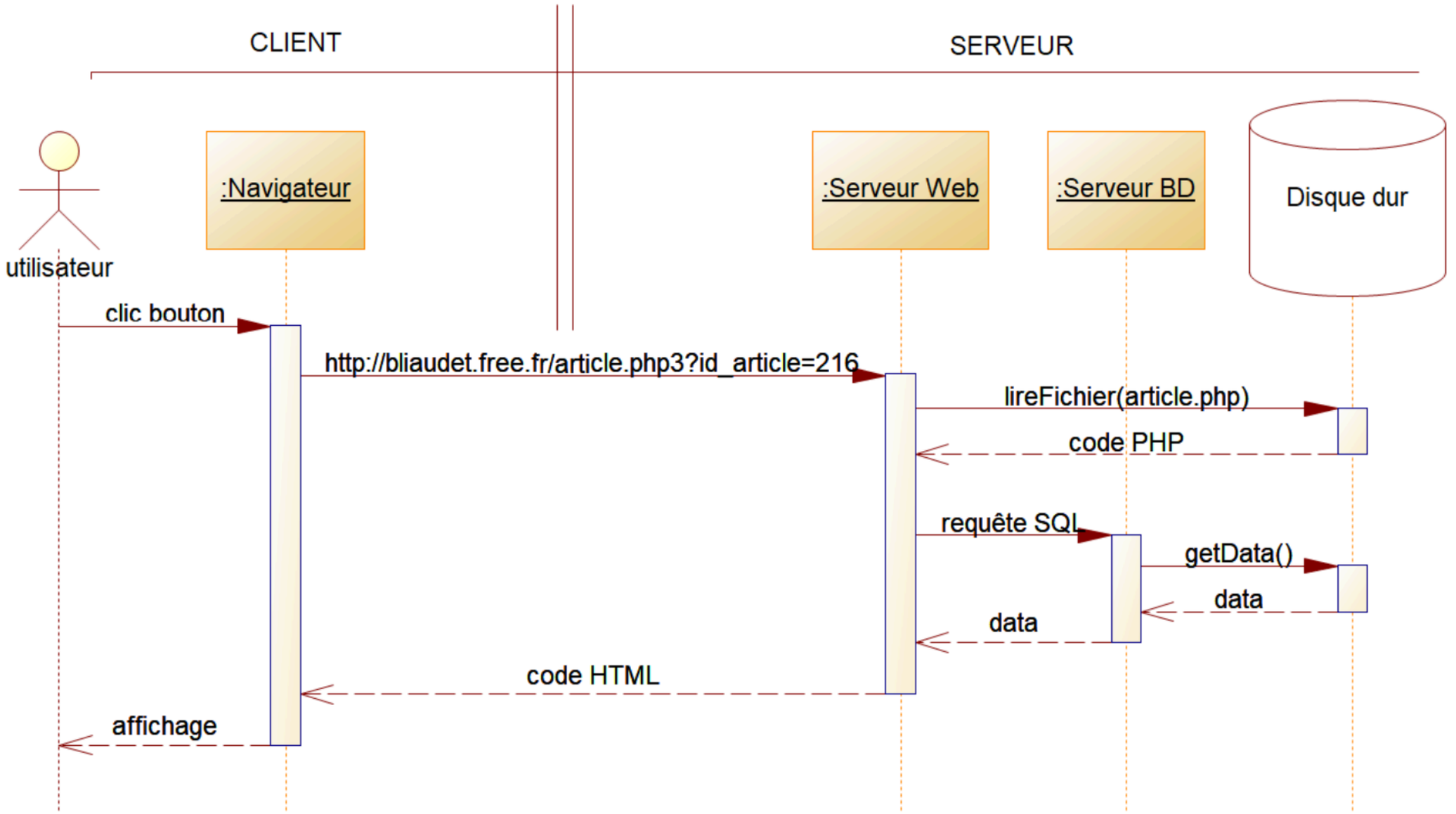
- **Cas 1 - HTML et JavaScript - Uniquement côté client, sans réseau (théorique)**
  - Comment se fait une modification JavaScript côté client ?



# Architecture

# Cas 2 : architecture client - serveur synchrone

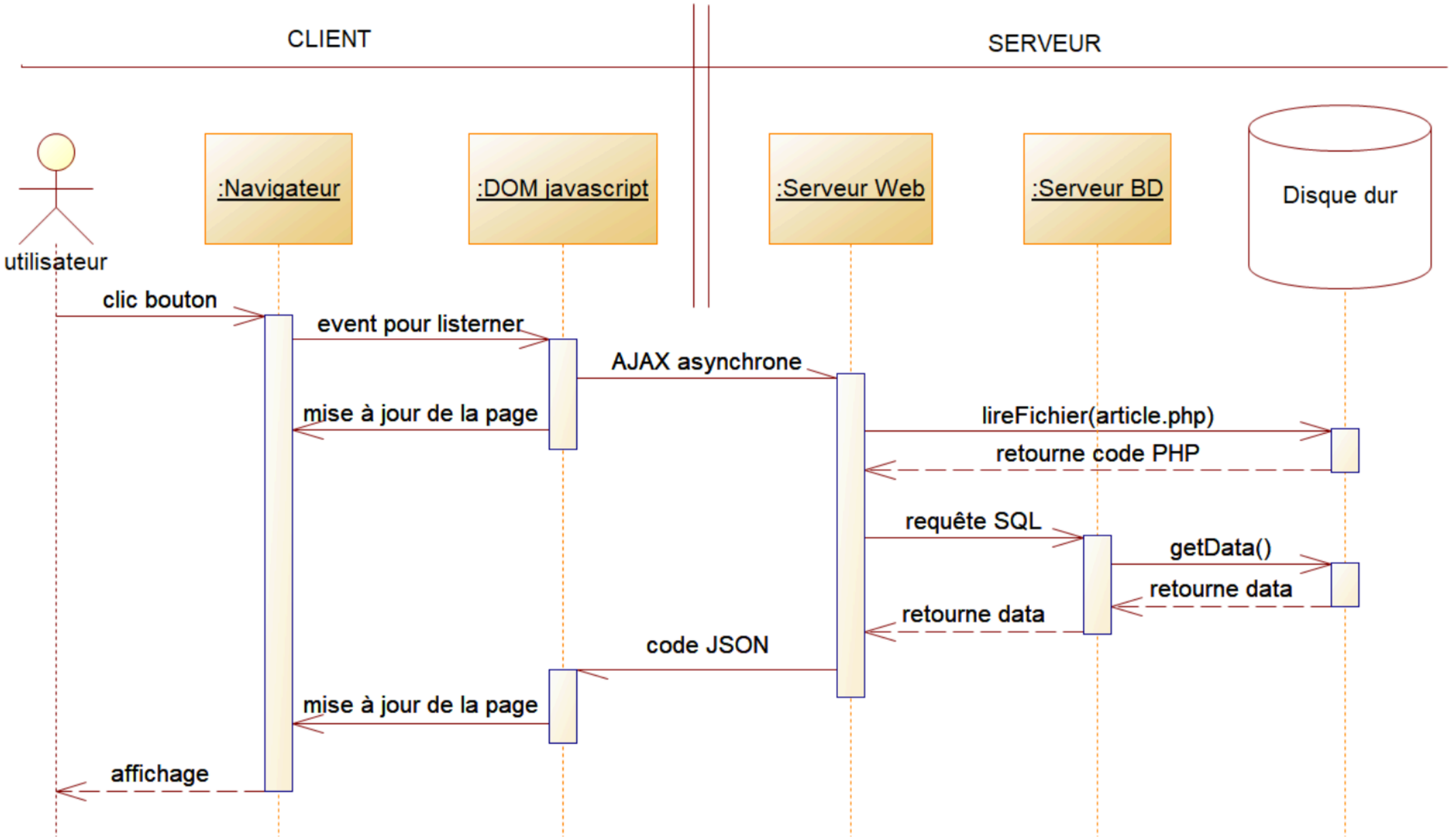
- **Cas 2 : Architecture Client - Serveur**
  - Comment on accède à la page « introduction web » à partir du site bliaudet.free.fr



# Architecture

# Cas 3 : architecture AJAX : asynchrone

- **Cas 3 : JavaScript + AJAX : architecture Client-Serveur asynchrone**
  - Comment on fait une modification de la page par une requête AJAX ?



# Introduction au Web

**4**

**Conclusion : les développeurs web**

# Les développeurs

- **Développeur front-end :** langage client. Travaille sur les pages web. HTML, CSS (plutôt intégrateur) mais surtout javascript, animation de page web (à distinguer du front-office).

- **Intégrateur :** prendre une maquette de graphiste et la traduire en html - CSS

- **Développeur back-end :** langage serveur. Le soldat : la grosse artillerie. Le communique l'info au dev front (à distinguer du back-office)

- **Responsable qualité** : vérifie que tout marche. Joue le rôle de l'utilisateur.

- **Administrateur Système :** fait marcher les serveurs.

- La ou le **développeur Full Stack** : il fait tout !

# Devenir une ou un développeur Full Stack !

1. Avoir une vison d'ensemble (cours d'introduction)
2. HTML 5 - CSS 3
3. Apprendre un langage serveur : PHP, ou Java, ou Ruby, ou Python, ou Node, etc.
4. Apprendre le SQL et la modélisation, pour maîtriser les données
5. Apprendre JavaScript côté client : pour avoir une page web plus dynamique côté client
6. Apprendre un framework : Laravel pour le PHP (par exemple), JEE pour le Java, etc.
7. Apprendre la ligne de commande Linux pour maitriser les serveurs.
8. Connaître les bases en réseaux
9. Faire de la veille techno
10. Pratiquer
11. Aimer ça !

# Classification des langages :

**Classification : langages, types de langages, type d'application**

| | | | | | |
|---|---|---|---|---|---|
| | Développeur Full-Stack – Dev Ops | | | | |
| | Front = Front-end | Back = Back-end | | | |
| Calcul - IA | WEB - Client | WEB - Serveur | Datas | Système | Mobile |
| C<br>C++, Java, C#<br>Python<br>Etc. | HTML<br>CSS<br>JS | PHP<br>Java, C#<br>JS, Python, Ruby<br>Etc. | SQL<br>Python, R<br>JSON | Shell<br>PowerShell<br>DOS | Java, Kotlin<br>Swift<br>JS |
| Client lourd<br>Desktop | Client léger | | | | Mobile |

# 5 acteurs web majeurs sur le marché (2024) :

1. Oracle : Java - JEE - Oracle
   - ⇨ Langage : **Java** (JSE : Java Standard Edition), **JEE** (Java Entreprise Edition) ou **Spring**
   - ⇨ BD : plutôt **Oracle**

2. Node.js : JavaScript - Angular, Express, …
   - ⇨ **Nodes.js :** du JavaScript côté serveur. Framework : **Angular** ou **Express**
   - ⇨ BD : au choix.

3. Libres -1 : Python - Django - PostgreSQL
   - ⇨ Langage: **Python** et les packages associés. Framework : **Django** ou **Flask**
   - ⇨ BD : plutôt **PostgreSQL**

4. Libres -2 : PHP - Laravel - MySQL
   - ⇨ Langage: PHP. Framework : Laravel
   - ⇨ BD : plutôt **MySQL** ou **MariaDB**.

5. Microsoft : C# - .NET - SQL Server
   - ⇨ Langage : **C#** (C sharp). Framework : .**NET** (DotNET)
   - ⇨ BD : plutôt **SQL Server**

# Usages pro du PC

**5**

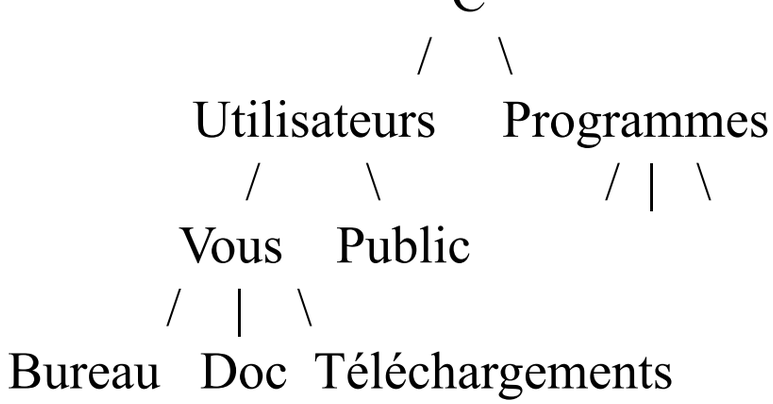
# Usages Pro de son PC

## 2 usages du PC

- ⇨ Grand public : on oublie !
- ⇨ Pro : c'est le but !

## Organisation du PC : l'arborescence des dossiers et des fichiers

- ⇨ Des dossiers et des fichiers organisés en arborescence
- ⇨




## 2 usages Pro : GUI vs CLI

- ⇨ Graphic User Interface : fenêtre graphique, souris, images, etc.
  Usage Grand public et pro
- ⇨ Command Line Interface : écran noir, clavier, texte.
  Usage uniquement pro

# Usages Pro de son PC

## OS : Windows10 ou Windows11

- ⇨ OS windows : 1 sur 2 de bien :
- ⇨ XP - Vista (nul) - 7 - 8 (nul) - 10 - 11 (pas terrible, mais ça va)

## Bon usage : désinstallation, installation

- ⇨ Désinstaller One Drive !
- ⇨ Désinstaller Edge
- ⇨ Installer Chrome ou Firefox
- ⇨ Installer Visual Studio Code
- ⇨ Installer SublimeText

## Bon usage : GUI et CLI

- ⇨ Graphic User Interface : fenêtre graphique, souris, images, etc.
  Usage grand public et pro
- ⇨ Command Line Interface : écran noir, clavier, texte.
  Usage uniquement pro
- ⇨ Usages pro : il faut maitriser les 2 !

# Usages Pro de son PC

## Commandes OS - CLI de base :

- ⇨ cd .. : pour remonter d'un niveau dans l'arborescence
- ⇨ cd ../.. : pour remonter de 2 niveaux dans l'arborescence
- ⇨ cd exo1 : pour aller dans le dossier exo1 à partir du dossier courant
- ⇨ cd ./exo1 : idem
- ⇨ cd html/exo1 : pour aller dans le dossier html/exo1 à partir du dossier courant
- ⇨ cd ./html/exo1 : idem
- ⇨ cd De* : pour aller dans Desktop s'il existe à partir du dossier courant
- ⇨ cd : pour afficher le dossier courant (pwd sur MAC)
- ⇨ dir : pour lister le contenu du dossier courant (ls -al sur MAC)
- ⇨ copy fic1.txt fic2.txt : fait une copie de fic1.txt appelé fic2.txt (cp sur MAC)

## Arborescence d'une page HTML de base :