

# **Cours Javascript-PWA**

## **V1.0 Groupes 06/07/08**

Y. Stroppa 2023

[https://github.com/ystroppa/M2\\_PWA](https://github.com/ystroppa/M2_PWA)

# Introduction

complexité

## Première partie :

Rappel

Exercice/tuto et mise en œuvre (en contexte DEV)

Pré-requis : virtualisation, conteneurisation

complexité

## Utilisation de CMS : Wordpress / Joomla / Drupal ...

Illustrations de sites sous Wordpress (html/css .. Js ... Php, Mysql/Postgres-sql) et exemples (mise en œuvre)

## Deuxième partie : Javascript avancé

Concepts asynchrones

PWA : explication et mise en œuvre

(similitude entre d'expérience utilisateur standalone-online)

# Rendu/projets

## Définition des conditions du projet

- groupe (2,3 étudiants)
- Etape 1 : définition : définir son projet / **valider** (expliquer objectifs/détailler si besoin -- **uniquement en PDF pas de docx** )
- Etape 2 : réalisation : réaliser le site web + ajouter les fonctionnalités avancées de PWA ....
- rendre en Juin-Juillet (date à définir ???) -- (date du jury - 15j)

Deux choses : dossier/rapport d'élaboration (PDF) et le livrable (partie site web)

Orientation libre du sujet : jeux ..... ou autres ???????

Rendre accessible vos rapports pour PSB (étudiants M2) et .... mettre en ligne vos réalisations (sous domaine) et en accès limité (étudiants M2) ...

# Contexte d'élaboration du projet

## Première solution :

### MAMP/WAMP/XAMP

Node.js tout en JS (prendre le squelette du jeu sur github et bâtir votre solution)

```
git clone https://github.com/ystroppa/....
```

```
npm install
```

```
npm start
```

à compléter et à personnaliser en fonction de vos objectifs

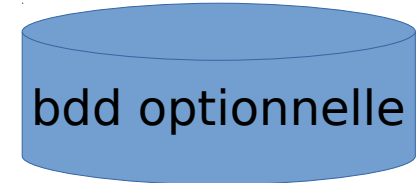
## Deuxième solution :

Travailler dans un contexte CMS (Wordpress)

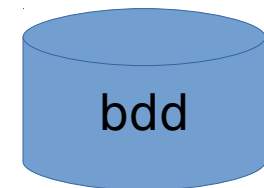
Adapter et utiliser les plugins nécessaires à votre projet

Contexte de traitement dynamique entre la partie navigateur et serveur via des appels fetch

mongodb



mysql



# A faire

## **Idée est de développer votre projet sous Wordpress**

(45 % des sites mondiaux)

environ 35 000 plugins (bibliothèque : composants logiciels-fonctionnels  
gratuit ou payant )

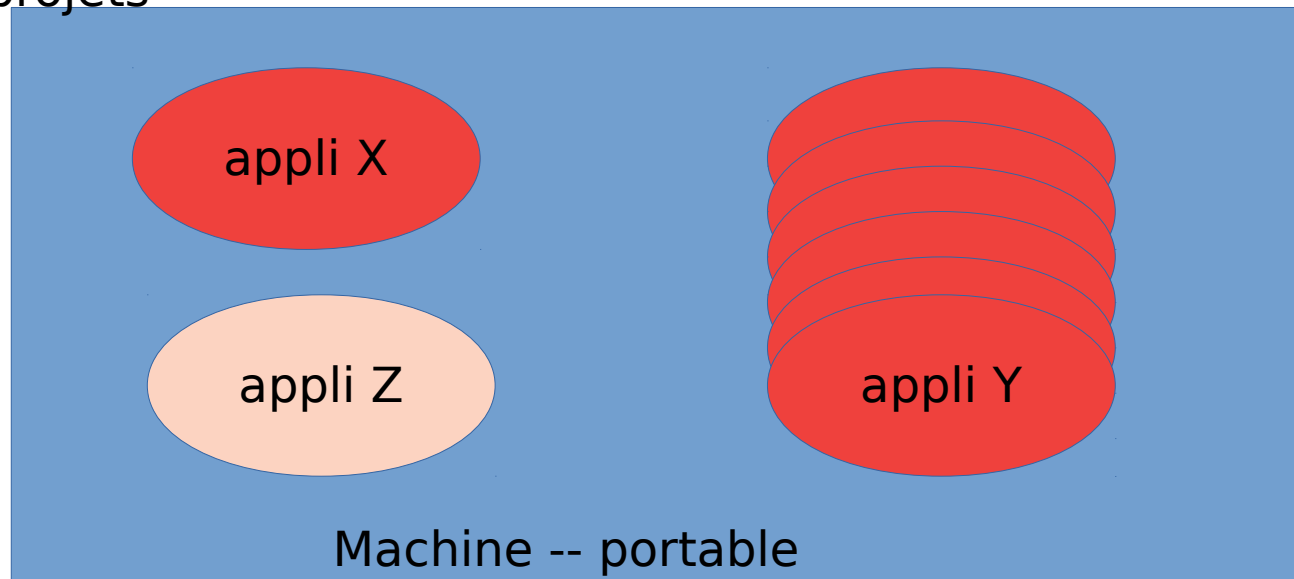
**Développer sous Wordpress un ensemble de gestion d'étudiants/produits/articles/.... sur le même principe avec la partie backend ....en php**

**Ou reprendre votre développement de l'année dernière (essayer d'innover) et venir étendre les fonctionnalités autour de la gestion des comptes, dashboard représentant les stats (graphiques) autour des parties effectuées ...top 10 ....**

**Installation de Wordpress (approche globale de la problématique web )**

# Sans virtualisation

Au fur et à mesure, on installe à la volée sur notre environnement les différents outils nécessaires aux projets



Problème d'accumulation d'environnement et d'applicatifs de différentes versions ...

Lors de la **saturation du disque** ou lors d'un début de **défaillance/latence** ou de **comptabilité** entre les librairies installées, on se pose la question comment je peux faire pour nettoyer ???? commence les misères

# Contexte portable

## Virtualisation

Objectif :

Préserve au maximum l'environnement Hôte  
(parce qu'il est complexe)

Solutions :

Windows : Hyper V, VirtualBox (Oracle), VmWare, ....

Mac os : Parallels (payant), VirtualBox (attention à la puce), VmWare Fusion

Linux : VirtualBox, VmWare ....

Définition :

installation et virtualisation d'un operating system

( Installer un système sur un autre -- cohabitation de plusieurs systèmes sur une même machine )

Exemple : une machine windows 10 : je peux installer un Linux, un autre windows 11.....

Une machine Mac os : installation d'un Linux, d'un Windows

Par la pratique

Système (OS) Hôte : sur lequel je viens superposer un autre OS

on va pouvoir travailler dans  
un environnement homogène  
à l'équipe partagé à partir de  
l'image de la VM

une fois votre projet terminé,  
on peut supprimer la VM, et  
votre Hôte reste propre.

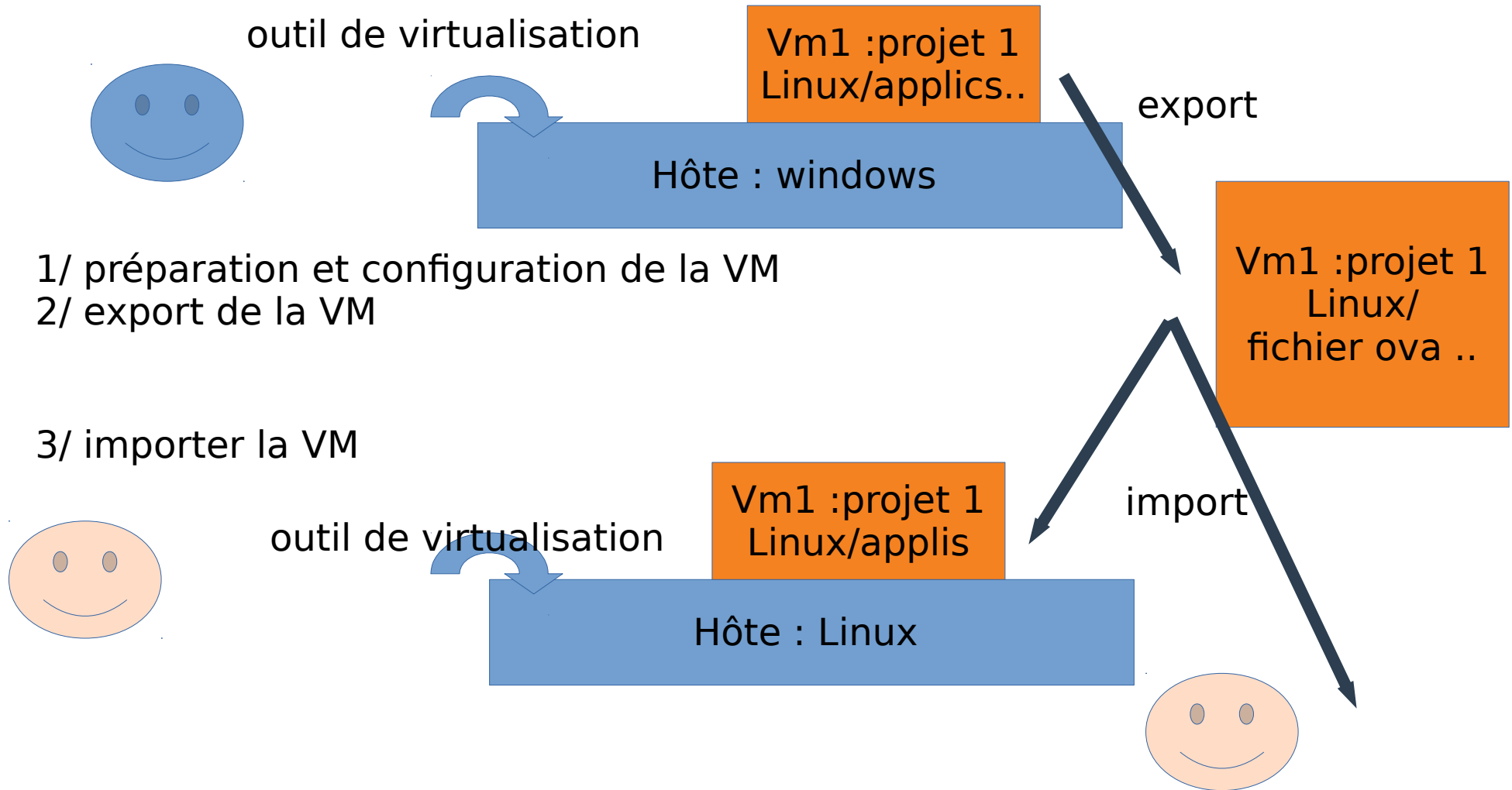
Hyperviseur  
outil de virtualisation

Vm1 : projet 1  
Linux/applics..

Vm2 : projet 2  
Windows

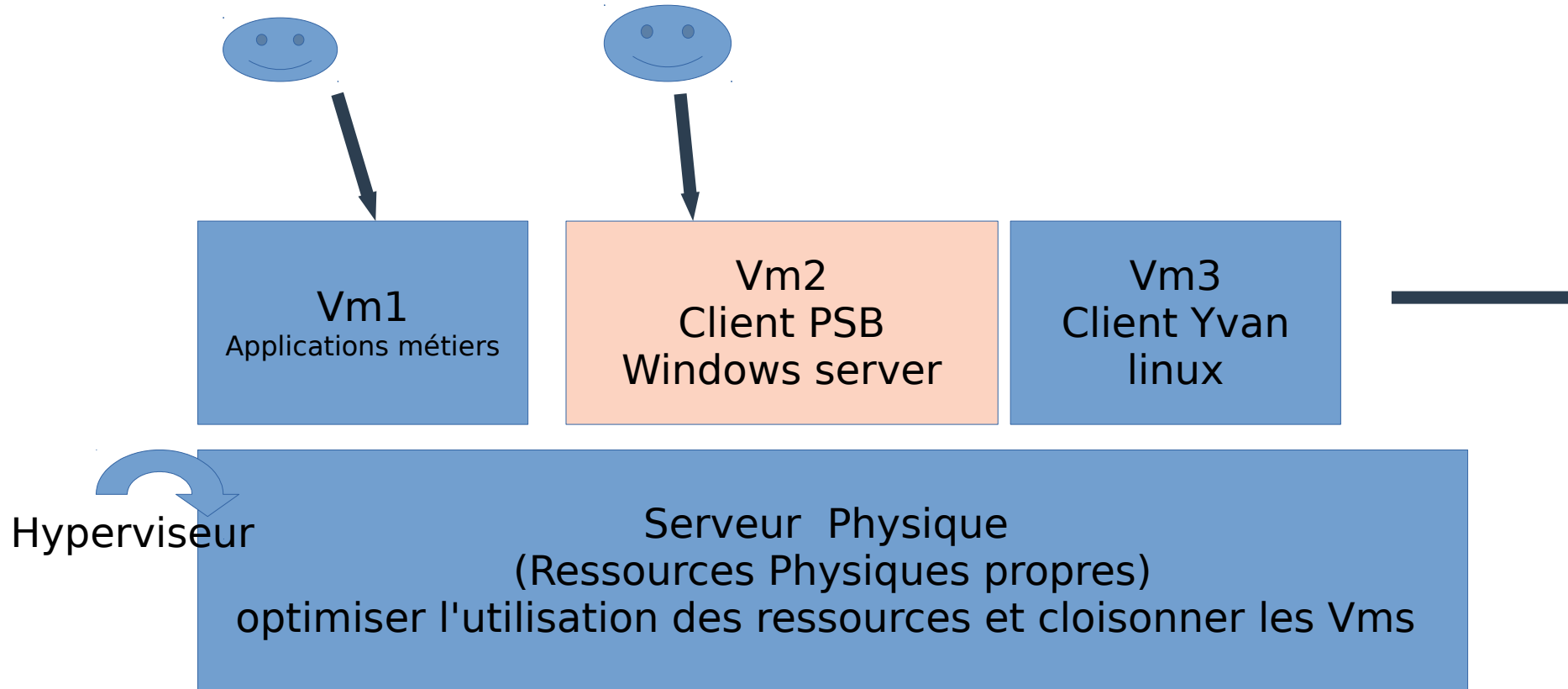
Hôte : operating system : système d'exploitation

# Echange de Vm





# Système en production /sur le CLOUD OVH, Amazone, Google .... entreprise



Vm => VPS : virtual Private Server

# Autre avantage de cette virtualisation

## Permet de se rapprocher des environnements en PRODUCTION

Avoir les mêmes concepts entre poste en dev et machine en prod (VmWare ESX, HyperV, Proxmox, KVM(Openstack) ....)

## Procédé de création d'une machine virtuelle

Hyperviseur (VirtualBox ou VmWare fusion, Workstation, player.....)

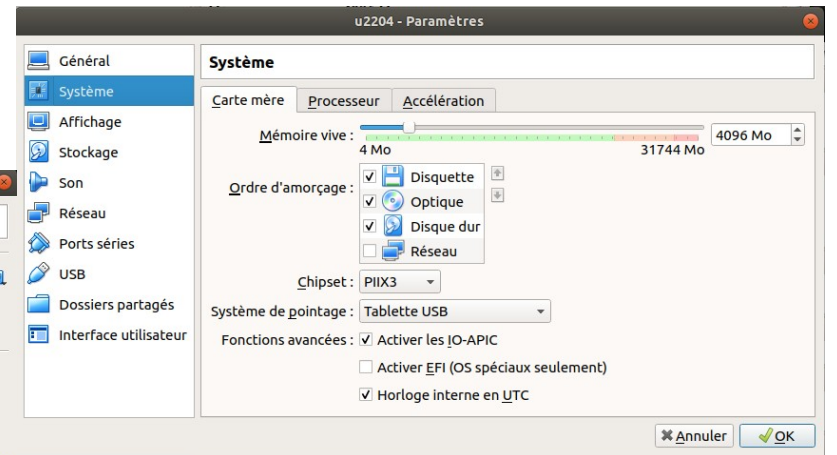
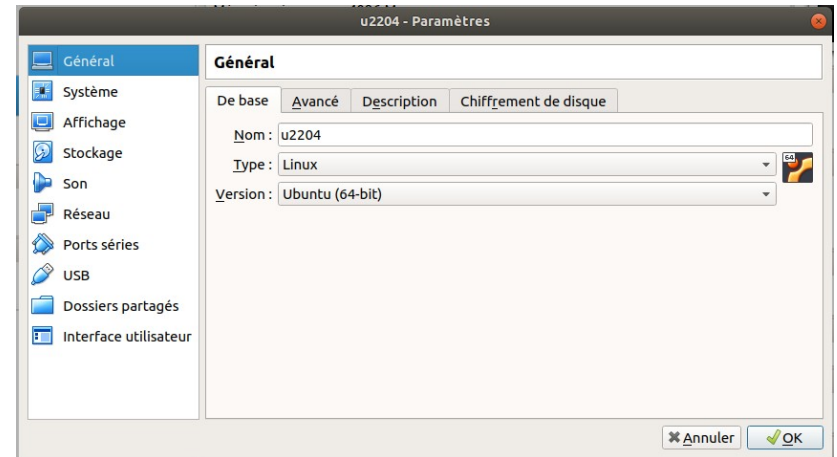
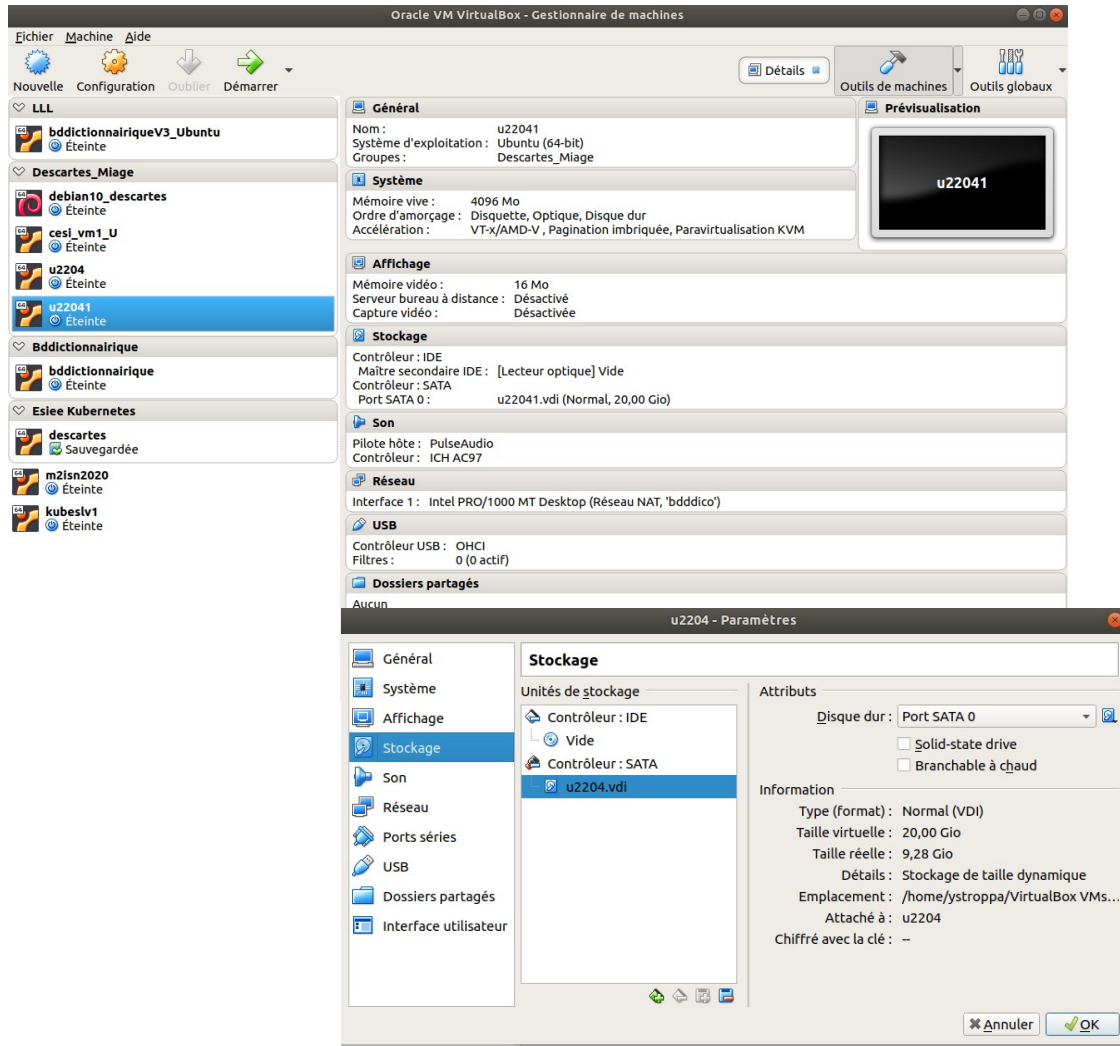
Création d'une nouvelle machine VM

Caractériser la VM : définir le type d'OS, la taille Mémoire, l'espace disque, le nombre cpu ....(définition des ressources allouées à cette VM)

On récupère l'image ISO du système que l'on veut installer

Et on démarre et on installe

# Exemples : sous virtualBox



# Deuxième pré-requis

## Conteneurisation

Fournir un ensemble applicatif avec toutes ses dépendances dans un conteneur ( un espace restreint - délimité)

Solutions actuelles : docker (proposer)

- constituer une structure de fichiers sous forme d'image (arborescence de fichiers -- répertoires )
- dans laquelle on va trouver :
  - binaire (programme -application )
  - toutes les dépendances associées à cette

application

**image : système de fichiers  
application  
+ toutes ses dépendances**

# **docker**

## **1/ visiter le site docker hub**

**inventaire des images disponibles et informations sur les images**

**<https://hub.docker.com/>**

## **2/ Intérêts**

**va permettre d'effectuer des installations simplifiées et cumulatives**

## **3/ Pré-requis pour ces installations**

**Installer docker dans votre environnement ( Mac Os, Windows, linux ..... ) ... un service docker qui fonctionne**

**4/ Installer et l'utiliser les images à partir de docker-hub et démarrer les services en question**

# Illustration classique sans docker

Installation classique (sans docker) d'un service (httpd, mysql, mariadb, mongodb .....)

- trouver le package d'installation du service
  - windows : .msi
  - mac os : .deb
  - linux : .tar.gz .deb (ubuntu)
- installation sur le système d'exploitation
- paramétrage du service
- difficultés :
  - droit administrateur
  - souhaite installer ce service sous différentes versions
  - souhaite installer des services qui utilisent des ressources communes (port : exemple mysql --> 3306 mariadb --> 3306 )
  - souhaite démarrer plusieurs fois le même service
  - souhaite également partager l'ensemble à d'autres

# Première approche pratique avec docker

**Préalable : installation de docker dans votre environnement (service doit être activé)**

**Pour vérifier : dans la console => docker ps**

**installation d'un serveur apache**

**vérifier l'existence d'une image pour apache**

**=> docker search httpd**

**Rapatrier l'image sur votre machine**

**=> docker pull httpd (l'image est rapatriée dans votre dépôt)**

**Exécuter (démarrer le conteneur à partir de l'image)**

**Démarrer le service (application - programme)**

**=> docker run httpd**

**Installation d'un serveur mysql**

**=> docker search mysql**

**=> docker pull mysql**

**=> docker run mysql (attention besoin de paramétrages supplémentaires password de root)**

# compléments commandes

**docker run -d httpd ==> (-d permet de détacher la sortie standard de l'application de la console de lancement)**

**docker ps ==> permet de lister les conteneurs actifs sous votre environnement**

**docker ps -a => permet de liste l'ensemble des conteneurs actifs ou pas**

**docker logs [PID] => permet de consulter les logs associés au conteneur avec le PID**

**docker inspect [PID] => permet de connaître les caractéristiques de votre conteneur, son paramétrage et son adresse réseau.**

**docker logs [pid] => permet de voir le contenu des logs associés au conteneur.**



# Petit exemple avec un service web

## Première exemple d'accès au service

`docker run -d httpd`

`docker ps =>` récupérer l'ID du conteneur

`docker inspect [PID] =>` récupérer l'adresse IP du conteneur

se connecter à partir d'un navigateur et se connecter sous l'URL  
`http://adresse_ip`

## Deuxième exemple d'accès au service

`docker run -p 8081:80 -d httpd` => permet d'attacher le port  
8081 de l'hôte au port 80 du conteneur

se connecter à partir d'un navigateur et se connecter sous l'URL  
`http://localhost:8081`

# Rappel des commandes

<code>docker build -t friendlyname .</code>	# Create image using this directory's Dockerfile
<code>docker run -p 4000 : 80 friendlyname</code>	# Run "friendlyname" mapping port 4000 to 80
<code>docker run -d -p 4000 : 80 friendlyname</code>	# Same thing, but in detached mode
<code>docker exec -it [container-id] bash</code>	# Enter a running container
<code>docker ps</code>	# See a list of all running containers
<code>docker stop &lt;hash&gt;</code>	# Gracefully stop the specified container
<code>docker ps -a</code>	# See a list of all containers, even the ones not running
<code>docker kill &lt;hash&gt;</code>	# Force shutdown of the specified container
<code>docker rm &lt;hash&gt;</code>	# Remove the specified container from this machine
<code>docker rm -f &lt;hash&gt;</code>	# Remove force specified container from this machine
<code>docker rm \$(docker ps -a -q)</code>	# Remove all containers from this machine
<code>docker images -a</code>	# Show all images on this machine
<code>docker rmi &lt;imagename&gt;</code>	# Remove the specified image from this machine
<code>docker rmi \$(docker images -q)</code>	# Remove all images from this machine
<code>docker logs &lt;container-id&gt; -f</code>	# Live tail a container's logs
<code>docker login</code>	# Log in this CLI session using your Docker credentials
<code>docker tag &lt;image&gt; username/repository:tag</code>	# Tag <image> for upload to registry
<code>docker push username/repository:tag</code>	# Upload tagged image to registry
<code>docker run username/repository:tag</code>	# Run image from a registry
<code>docker system prune</code>	# Remove all unused containers, networks, images (both dangling and unreferenced), and optionally, volumes. (Docker 17.06.1-ce and superior)
<code>docker system prune -a</code>	# Remove all unused containers, networks, images not just dangling ones (Docker 17.06.1-ce and superior)
<code>docker volume prune</code>	# Remove all unused local volumes
<code>docker network prune</code>	# Remove all unused networks

# Les volumes :

**Docker peut travailler avec des volumes locaux afin de conserver et de garantir la persistance des données une fois le conteneur arrêté et supprimé. Exemple avec httpd ... on souhaite conserver et modifier le contenu du répertoire par défaut de httpd. /var/local/apache/htdocs**

**Démarrer le conteneur et modifier le contenu index.html**

**Rendre le contenu permanent et directement modifiable à partir du host**

# Comment créer une image

**Construire un fichier Dockerfile et préciser dans ce fichier les différents opérations nécessaires pour construire une image avec les fichiers souhaités :**

Fichier Dockerfile

```
FROM httpd  
workdir /usr/local/apache2/htdocs  
copy index.html index.html
```

Pas besoin de préciser le programme à démarrer et le port d'écoute : prendra les valeurs par défaut définis dans l'image de départ.

`docker build -t image1 .`

génération de l'image

# **docker-compose**

**Wordpress est composé de deux éléments qui ont besoin de se connaître et de dialoguer ensemble : plusieurs solutions pour effectuer cette opération**

- créer un réseau dédié à l'aide de la commande `docker create network networkwordpress`**
- les affecter sur ce réseau à l'aide de l'option `--network networkwordpress`**
- et faire en sorte que le conteneur wordpress connaisse le conteneur mysql**

# Démarrage de wordpress : wordpress.yml

version: "3.3"

services:

db:

image: mysql:5.7

volumes:

- /home/ystroppa/LLL/ravioli/site/db\_data:/var/lib/mysql

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: somewordpress

MYSQL\_DATABASE: wordpress

MYSQL\_USER: wordpress

MYSQL\_PASSWORD: wordpress

wordpress:

depends\_on:

- db

image: wordpress:latest

volumes:

- /home/ystroppa/LLL/ravioli/site/wordpress\_data:/var/www/html

ports:

- "10000:80"

restart: always

environment:

WORDPRESS\_DB\_HOST: db:3306

WORDPRESS\_DB\_USER: wordpress

WORDPRESS\_DB\_PASSWORD: wordpress

WORDPRESS\_DB\_NAME: wordpress

services:

db:

# We use a mariadb image which supports both amd64 & arm64 architecture

image: mariadb:10.6.4-focal

# If you really want to use MySQL, uncomment the following line

#image: mysql:8.0.27

command: '--default-authentication-plugin=mysql\_native\_password'

volumes:

- db\_data:/var/lib/mysql

restart: always

environment:

- MYSQL\_ROOT\_PASSWORD=somewordpress

- MYSQL\_DATABASE=wordpress

- MYSQL\_USER=wordpress

- MYSQL\_PASSWORD=wordpress

expose:

- 3306

- 33060

wordpress:

image: wordpress:latest

volumes:

- wp\_data:/var/www/html

ports:

- 80:80

restart: always

environment:

- WORDPRESS\_DB\_HOST=db

- WORDPRESS\_DB\_USER=wordpress

- WORDPRESS\_DB\_PASSWORD=wordpress

- WORDPRESS\_DB\_NAME=wordpress

volumes:

db\_data:

wp\_data:

# Commandes docker-compose

docker-compose up	# Create and start containers
docker-compose up -d	# Create and start containers in detached mode
docker-compose down	# Stop and remove containers, networks, images, and volumes
docker-compose logs	# View output from containers
docker-compose restart	# Restart all service
docker-compose pull	# Pull all image service
docker-compose build	# Build all image service
docker-compose config	# Validate and view the Compose file
docker-compose scale <service_name>=<replica>	# Scale special service(s)
docker-compose top	# Display the running processes
docker-compose run -rm -p 2022:22 web bash	# Start web service and runs bash as its command, remove old container.



# Wordpress

## installation

MAMP/WAMP/XAMP (Apache et mysql)

à compléter avec les fichiers archives de Wordpress sous la racine de votre serveur web (création d'un site virtuel sous Apache )

à créer un utilisateur (dédié pas root) au niveau de votre base de données

## **Se connecter sous Wordpress et initialiser l'instance ( indiquer les comptes pour l'accès à la base de données et le compte principal de WP)**

## **Mettre en place les plugins**

Formulaires classiques : Everest forms (site avec inscriptions et formulaires divers et variés)

**Snippets woody** ( permet de fabriquer des fichiers de type html, js, css, php, ...scripts)  
et le thème vdperanto ....

## **Fabrication de votre projet dans ce contexte**

# contexte CMS



serveur apache + base de données + plugins +  
développement et adaptation php et js

# Elaboration d'un site de présentation/exposition M2 2022

**Proposition pour cette année ???**

**Wordpress dédié pour la promotion**

liste des sujets (indexation)

recherche par mots clés

accès aux rapports

et aux sites élaborés

**Sous un domaine ....**

**Recueil complet des projets et des solutions  
utilisées ....**

# Démarche projet

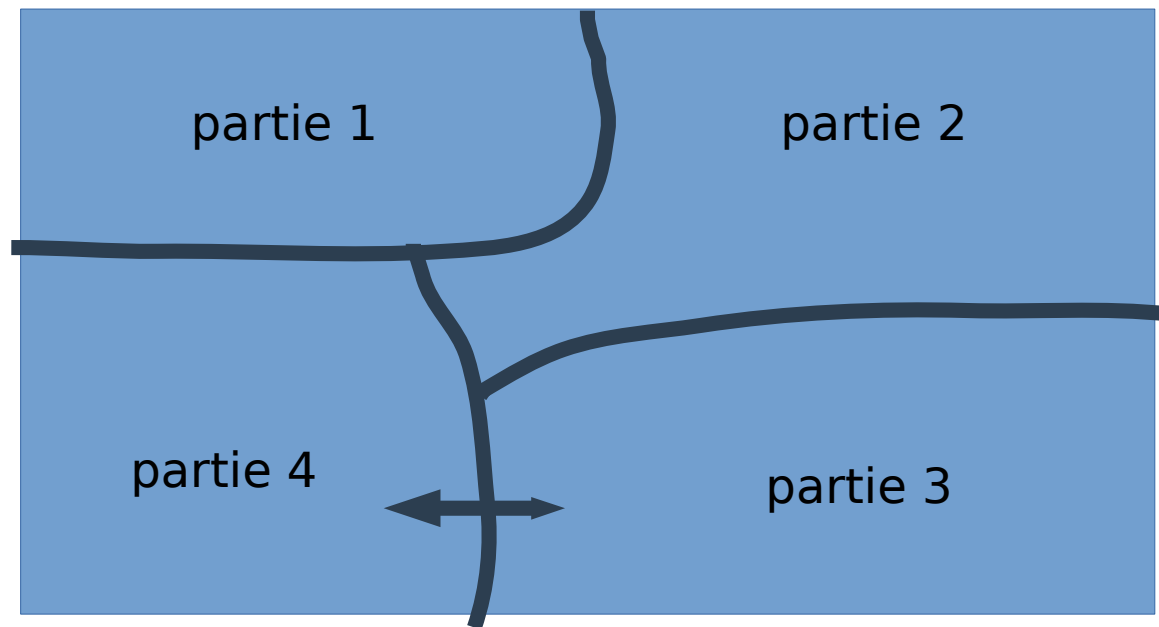
## **Vision globale : description la plus détaillée possible**

Découpage en lot pour une répartition au niveau de l'équipe  
(méthode AGILE développement par partie (priorisation))

## **Analyse et développement(temps/coûts )**

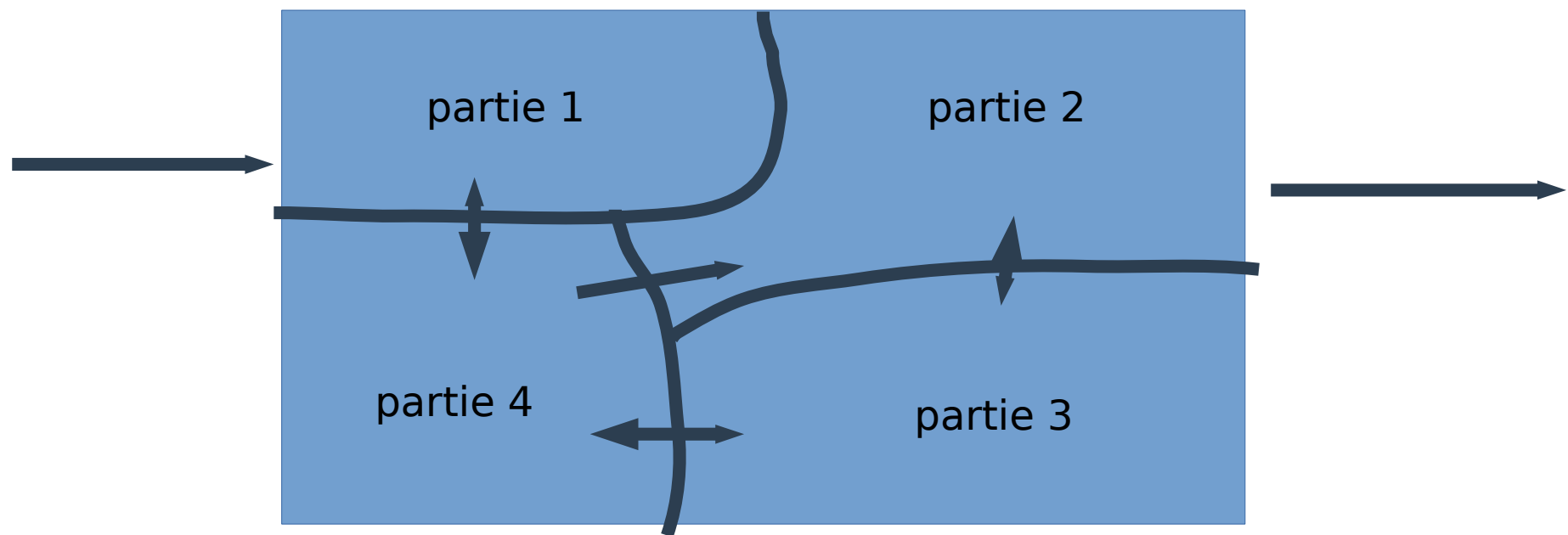
- Intègre (exploration de solutions existantes payantes ou non)  
levée l'ambiguïté rapidement ... par la construction d'un prototype (vérifier la faisabilité)  
( hors de question de modifier une librairie/solution importée valide pour la partie JS / PHP )  
Epuisée les solutions existantes
- Développe from scratch ( on a la main)  
démarche à prendre en compte dans le développement (tests .....)

# découpage



tests unitaires

# Intégration des différentes parties



# Rappels

## Partie infrastructure Web :

### Service Web

Protocoles http(local) et https :

Ports : 80 et 443

http: contexte portable

https : contexte public/entreprise

Stateless ou Statefull

(Notion de conservation d'état/données)

au niveau requête, session, application

Solutions possibles :

- Service d'hébergement (OVH, IONOS, AWS ... Google...Azure...) solutions packagées (CMS disponibles .....
- Solutions virtuelles : VPS
  - Apache**, MAMP, WAMP, LAMP (package : assemblages de services)
  - Tomcat, IIS, Nodejs ....python(Django -- flask)**
- Serveurs dédiés (matériel -- exclusif)

Besoin de **persistance longue durée**  
: dans des systèmes dédiés et structurés

Bases de données

Solutions possibles :

Relationnelles :

Mysql,  
Mariadb,  
Postgresql,  
Oracle,  
Sqlserver ...

noSQL :

Mongodb  
Cassandra  
Couchbase  
Neo4j  
....

scripts

php

js

python

java

# Rappels bases de données (persistance -- conservation)

**Service réseau (accessible via un port sur une machine)**

**Port d'écoute spécifique**

3306 : Mariadb ou mysql

5432 : postgresQL

1521 : Oracle

XXXX : Sql Server(Microsoft)

.....

27017 : Mongoddb

couchBase

Neo4j .....

Relationnelles

Langage SQL

NoSQL

Langages  
Javascript  
N1QL

**Traitements internes**

Structures de données sous une forme particulière

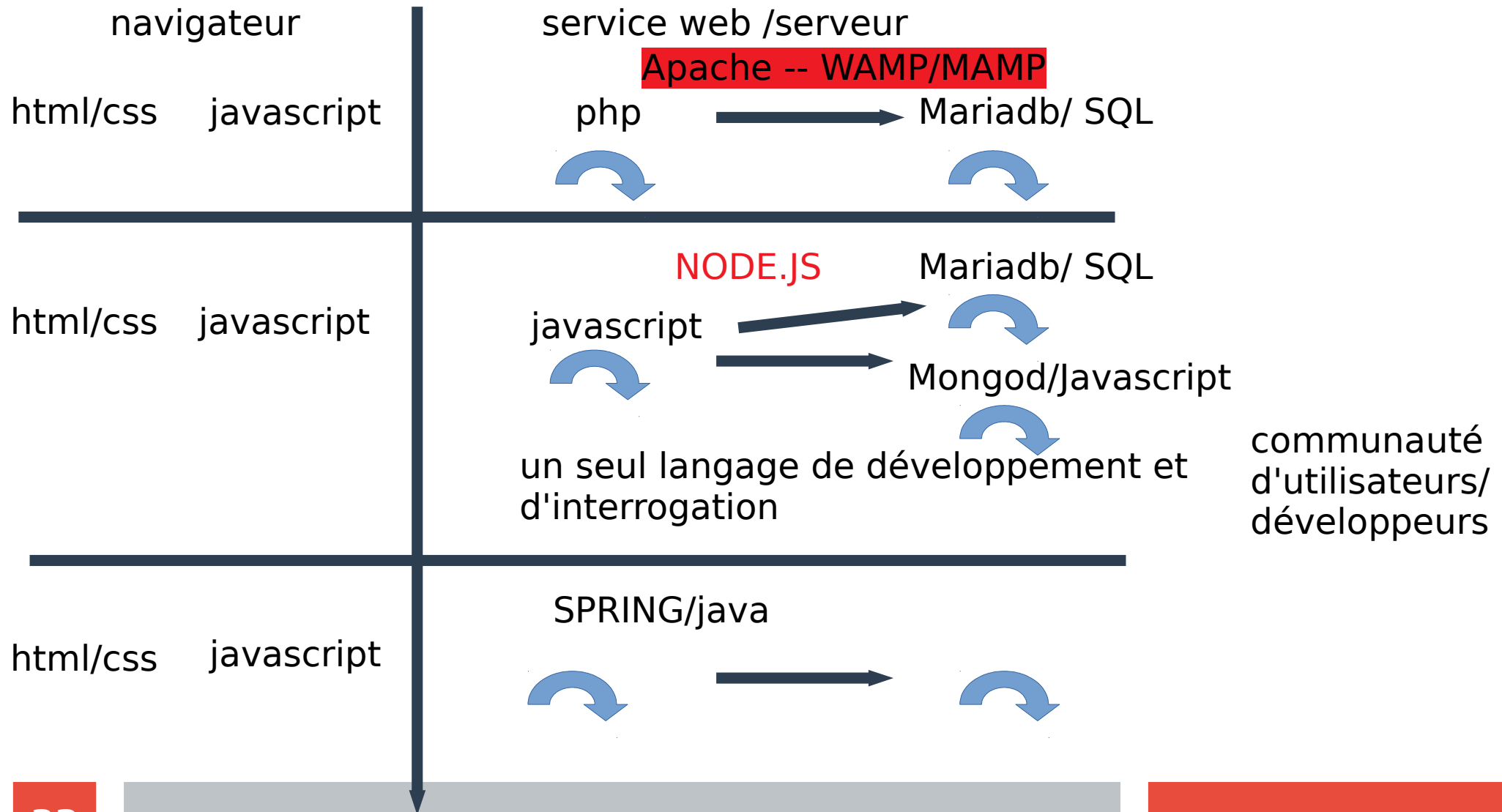
Relationnelle : tables == relations .... langage d'interrogation (SQL)

NoSQL : structure de type document/ key-value /graphe -- langage spécifique

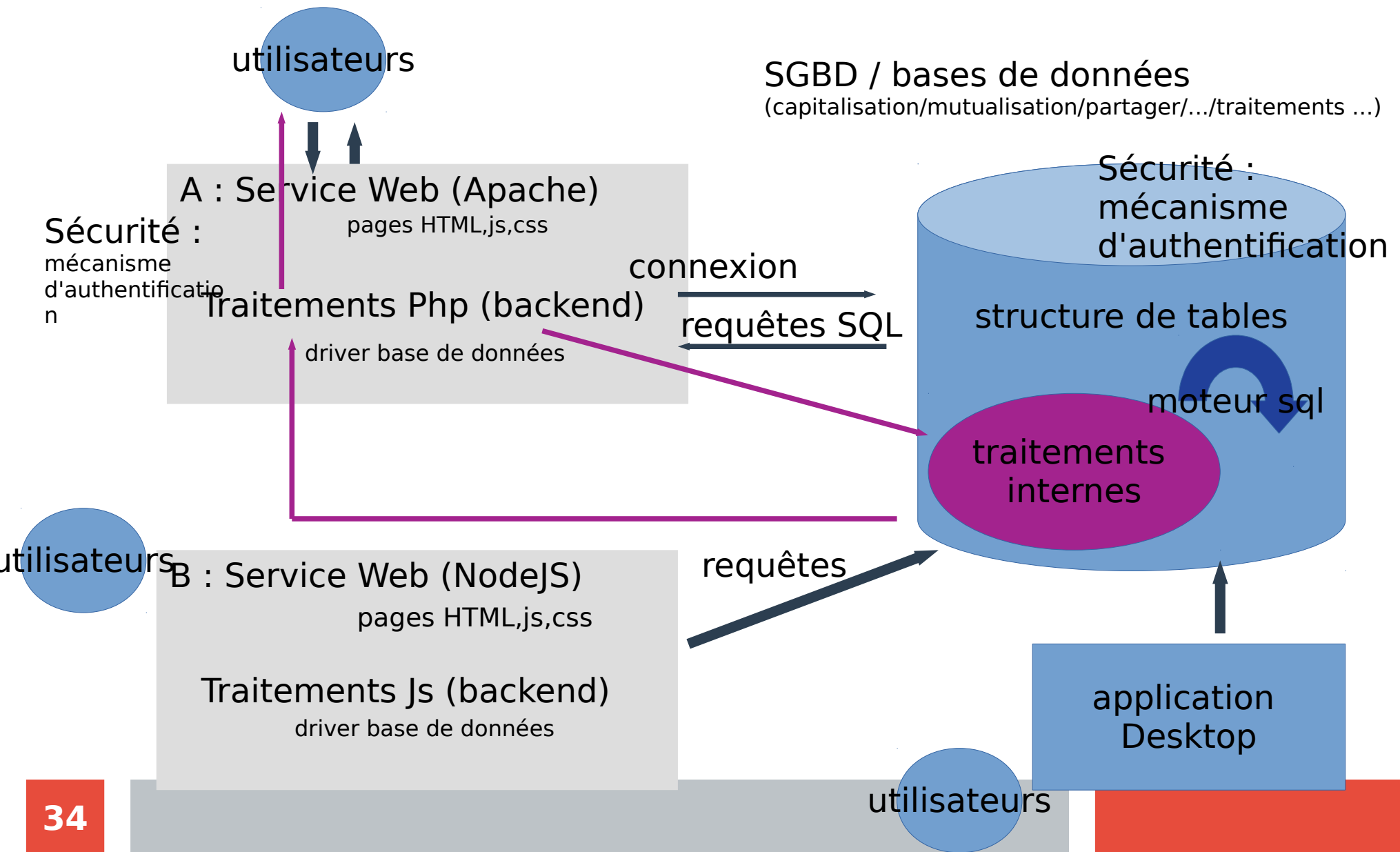
Procédures stockées, trigger ....langages spécifiques



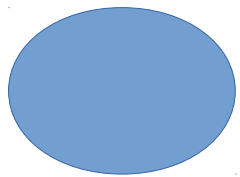
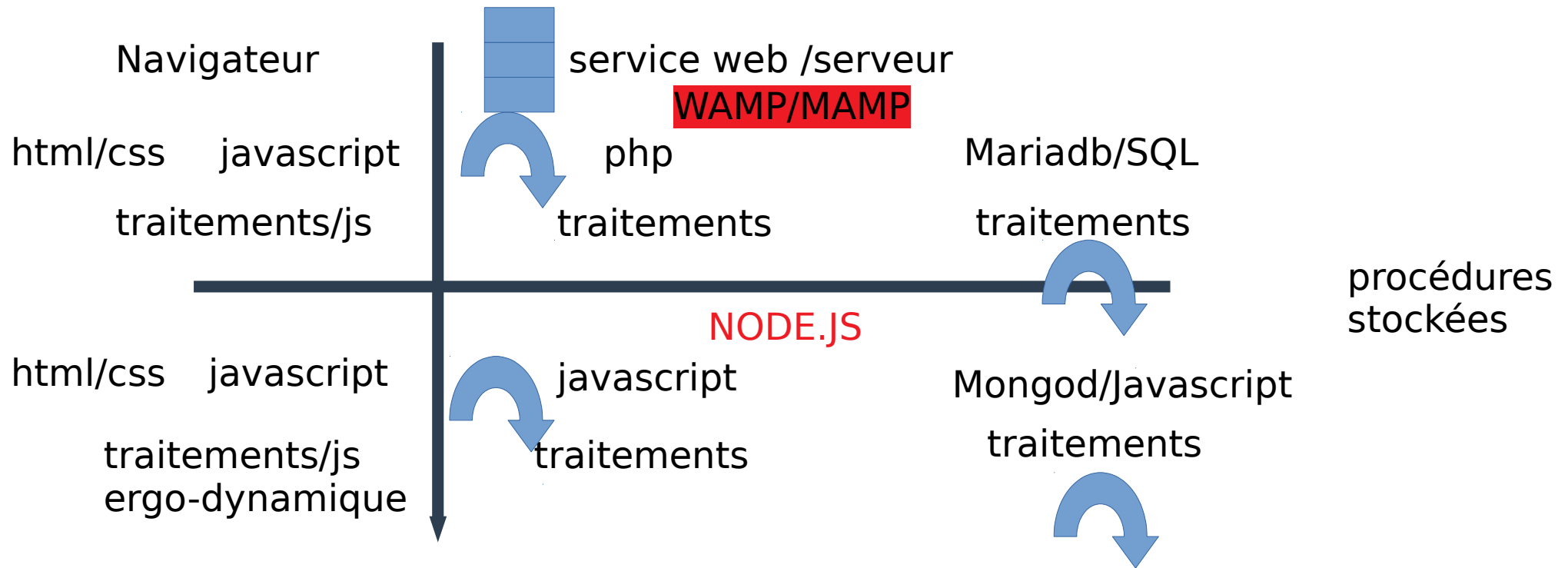
# cohérence des langages



# Localisation des traitements/notion multi applications/multi utilisateurs

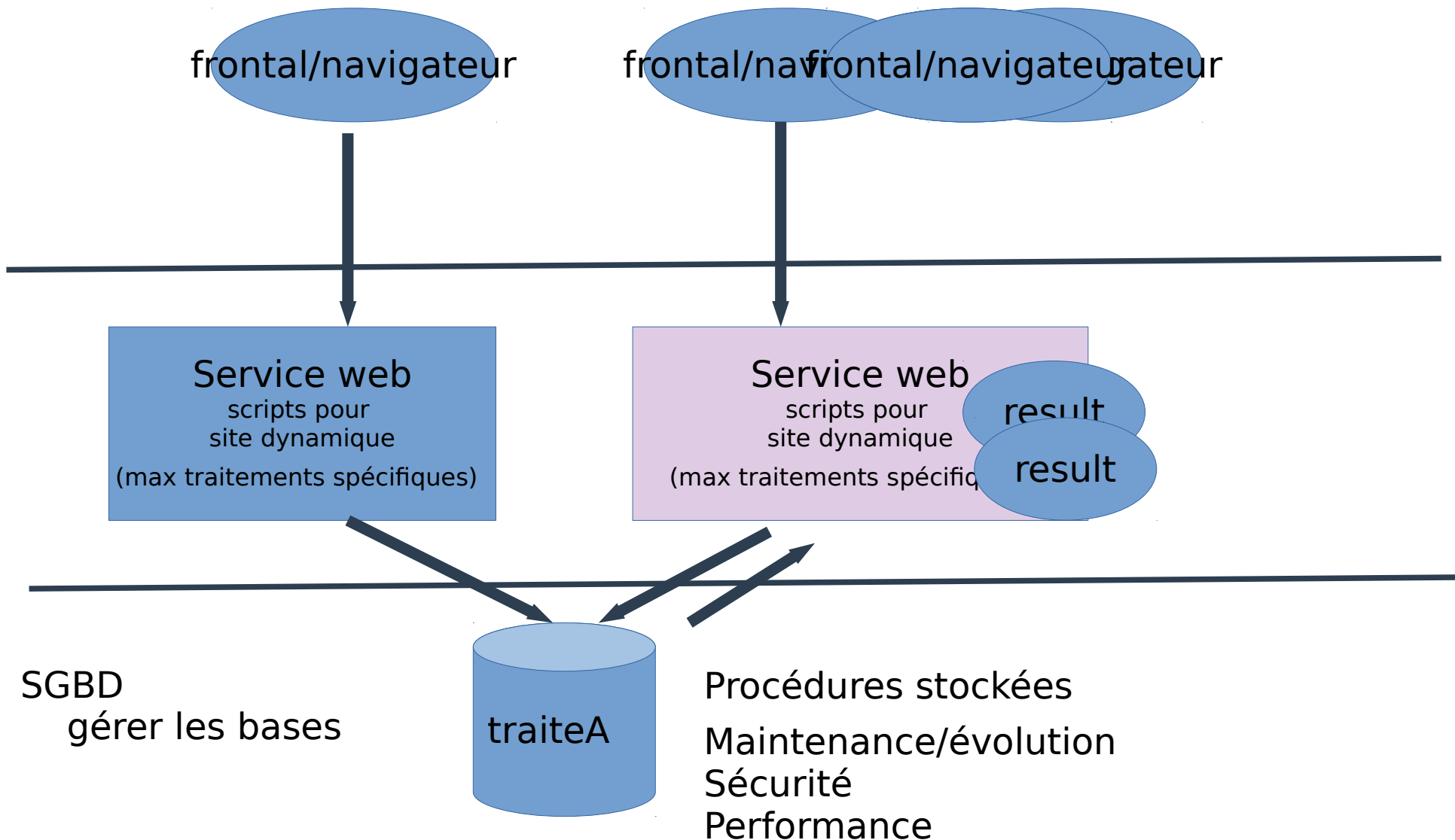


# Positionnement des traitements dans le processus global de notre site web

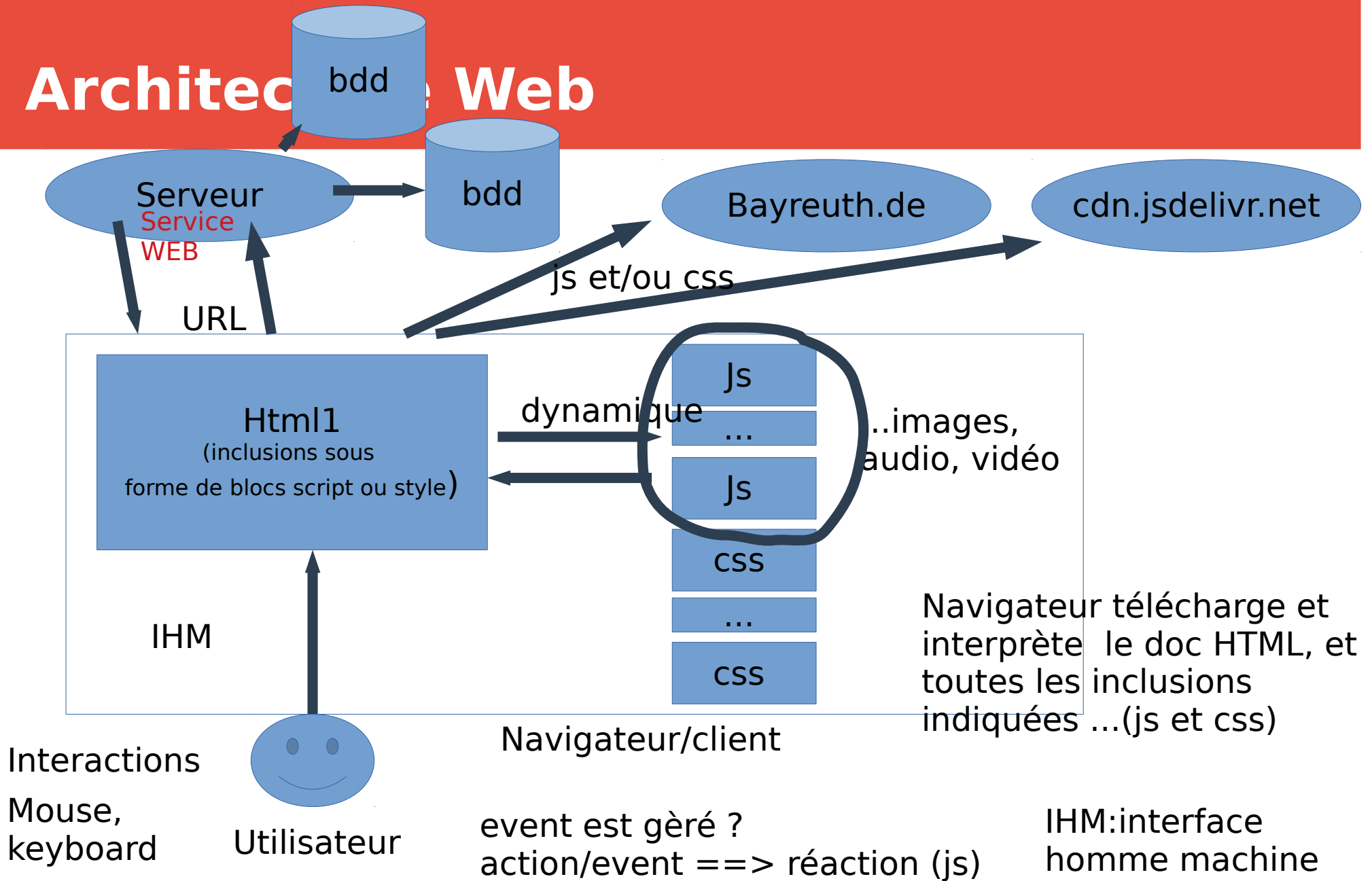


!!!!!! attention à l'ouverture d'une ressource pensez à la terminer. Parce que les processus fonctionnent en permanence.

# Architecture multi applicatives



# Architecture Web



# Architecture client-léger

**On développe une application multi-client et multi-machine (ordi, portable, tablette )**

**Pas à se préoccuper de l'installation du frontal de notre application ==> utilise le navigateur**

**Avantages :**

Lors des mises à jour ==> uniquement du côté du serveur (service web)

**Quelles sont les contraintes ?**

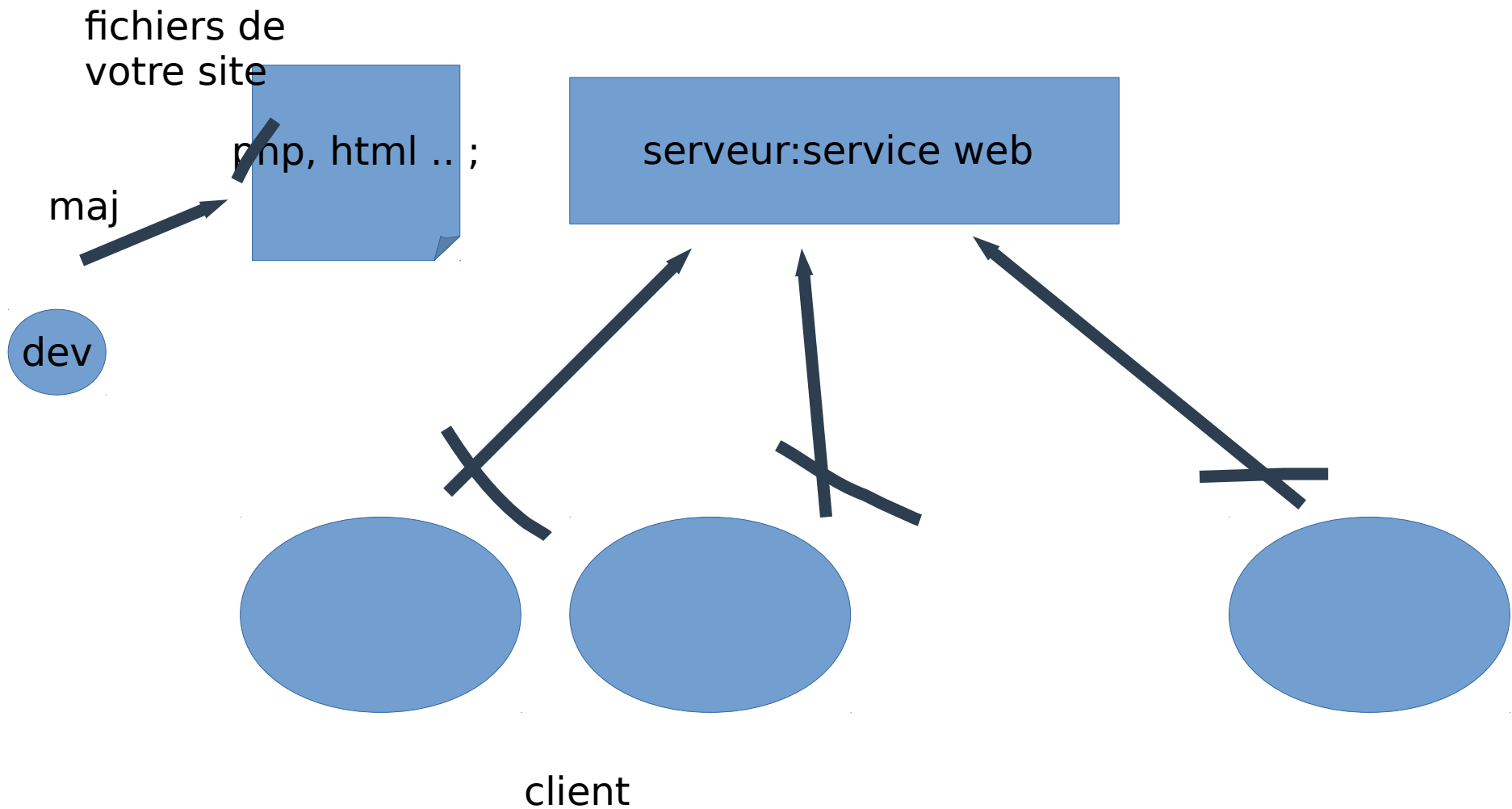
pas la maîtrise du frontal ...

prévoir des tests sur les différents modèles avant déploiement (incompatibilité ou dysfonctionnement liés à votre code ou à un module )

le frontal évolue en fonction de l'éditeur (tenir au courant des évolutions et les anticiper ... si votre application à un cycle de vie )

Edge/Safari/Chrome/Firefox/Opera ..... différentes versions

# Impacte d'une mauvaise manipulation sur la partie centrale (effet immédiat)



# **avant les maj**

**étape de test en local (simuler sur sa machine le contexte cible)**

**étape pre-prod : système de tests ressemblant à la cible ouvre une campagne de tests multi-utilisateurs**

**étape en prod : on déploie que si c'est OK**

**Attention on s'appuie sur des langages interprétés ==> c'est au moment où l'interpréteur voit le code qu'il lève une erreur**



# Rappels

## Partie javascript :

Langage de développement objets (version ES6 : voir pour détail du langage <https://262.ecma-international.org/6.0/> )

Utilisé principalement dans le cadre de développement Web

Partie FrontOffice et BackOffice (node.js, base de données dédiées MongoDB)

Pour ne pas se tromper :

L'exécution du Javascript est du côté du client à partir du navigateur.

# Navigateurs

## Outils côté clients :

Partie élaborée et fournie par l'éditeur

Firefox Chrome Opera Opera GX Safari Edge ...

cycle de vie spécifique à chaque solution et éditeur

L'éditeur qui définit sa compatibilité, ses spécificités ... ses montées versions

**Ce qui peut provoquer des dysfonctionnements liés à la compatibilité/l'interprétation du navigateur à instant donné.**

**D'où la nécessité de vérifier que l'application fonctionne correctement sur les différents types de navigateur. S'assurer également que le développement réalisé + les bibliothèques soient bien utilisés. Tout au long du cycle de vie de l'application il faut être vigilant au modification de version des navigateurs ...**

**==>Maintenance du site /**

Sécurité (faille de sécurité)

Coûts induits ou effectuer de façon systématique des contrôles du client (navigateur)

# Sécurité et précaution

## Exposition permanente

via des robots

## Attention au respect de l'état de l'art

Attention si on développe un site from scratch (appliquer un certain nombre de règles dans le développement)

## Pour limiter ces tracas

Partir soit d'un CMS (coût d'appropriation plus faible)

Partir d'un framework (à choisir en fonction de la technologie souhaitée )

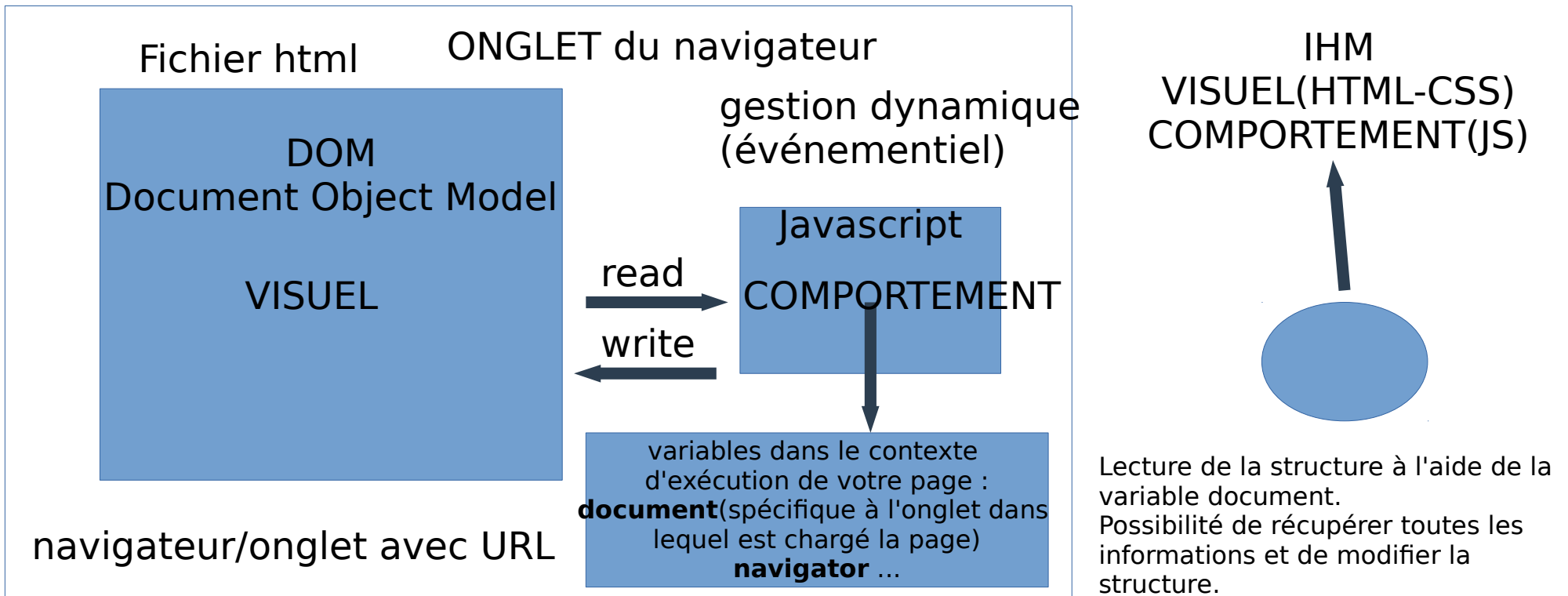
--- Php symfony

--- Javascript --- angular.js ..... vue.js .... meteor.js

Coût d'appropriation

# Rappel sur le fonctionnement

## Communication entre la DOM et Javascript



Onglet : représente un contexte d'exécution d'une page/application web

# Les actions possibles entre le HTML et Javascript

## On doit pouvoir interagir avec la DOM

être capable d'accéder à un élément de la DOM

pour récupérer ses propriétés visuelles et son contenu

pour modifier ses propriétés et son contenu

être capable de modifier la DOM

en supprimant des éléments (objets) : `remove`

en ajoutant des éléments : `createElement`

**La DOM peut se représenter sous forme d'arbre (structure hiérarchique). Le tout est de pouvoir naviguer dans cet arbre.**

# Structure hiérarchique

**<body>**

<DIV>

<DIV>

<input ..../>

<input />

</DIV>

</DIV>

<DIV>

.....

</DIV>

**</body>**

Javascript

Le traitement consiste à trouver les éléments dans cette arborescence et d'interagir avec leurs propriétés.

C'est dans ces cas que l'on va chercher la variable **document** (définie par le navigateur) et que l'on explore via les méthodes **document.getElementById....** ou **document.querySelector**

Attention aux différentes natures du retour de ces fonctions : soit un objet soit une liste d'objets

# Bâtir une IHM

Attribut des tabs HTML5

onclick

onmouseover

onmouseleave

onkeypressed

....

HTML/CSS

visuel

href

obj1

events

obj3

events

obj2

events

intervenir  
dans la  
structure de  
la DOM

serveur

ajouter l'interaction  
utilisateur

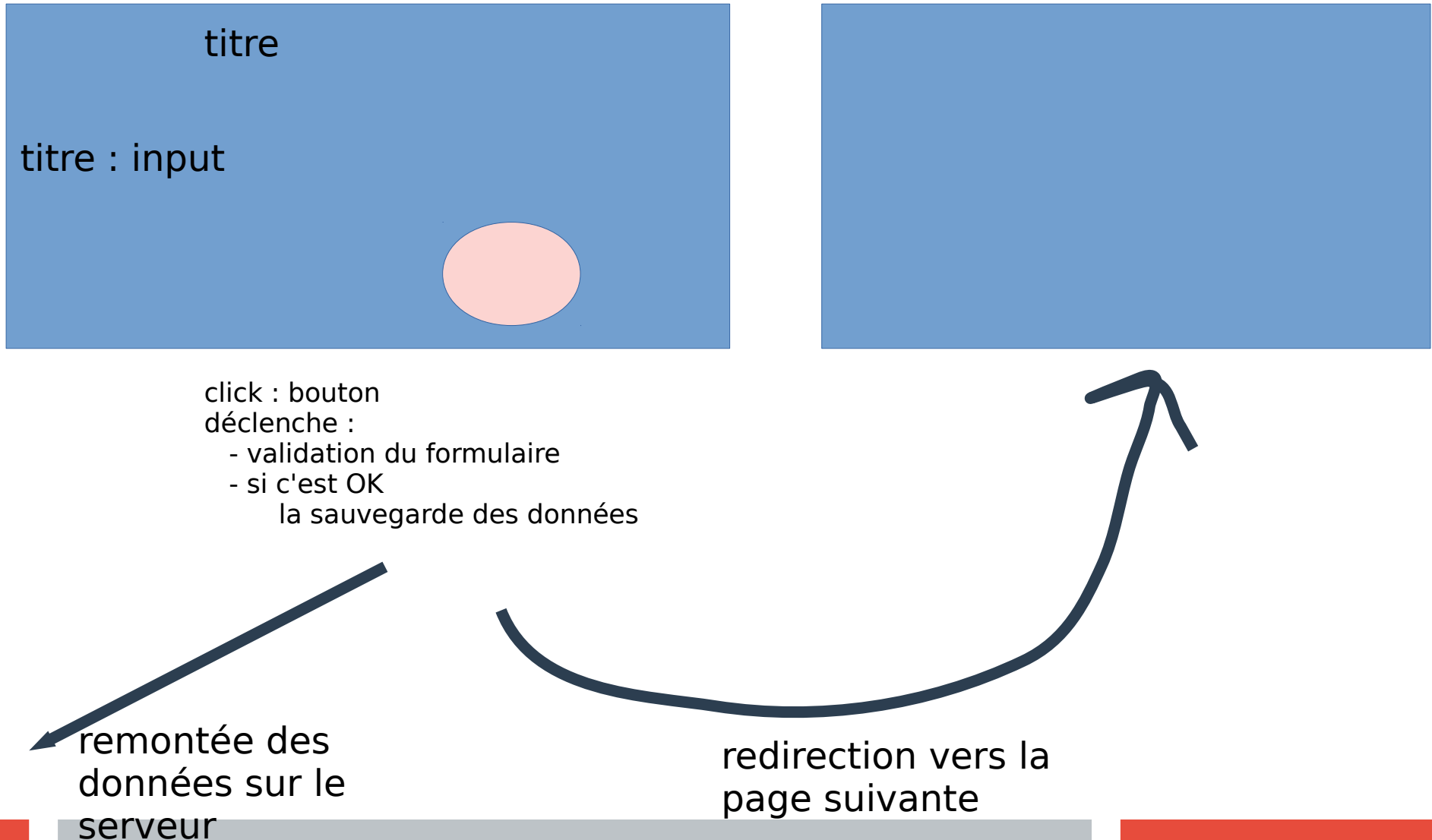
javascript

Traitements des  
events

objet : propriétés/  
contenu/  
events == javascript

navigateur

# esquisses modèles de vitre Application (IHM + comportement )





# Navigateur

## De quoi dispose t-on sur le client :

Langage (javascript)

Déboggeur et **console**

Editeur de sources, de styles

Analyseur de réseau

Permet de suivre les échanges et les temps de transfert

Données locales (cache)

Différents niveaux de persistance

Cookies

Cache local

key, valeur

Base de données : indexedDB

key, valeur

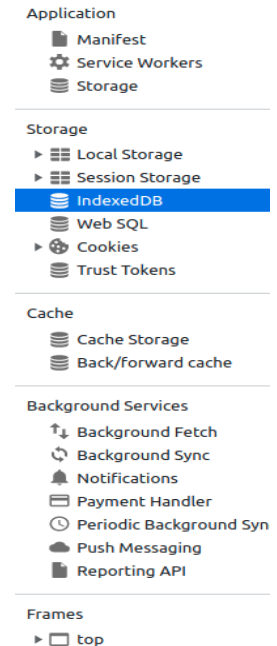
**cookieStore**

**localStorage**

**indexedDB**

Security : visualisation de la validité des certificats uniquement dans le cas de https

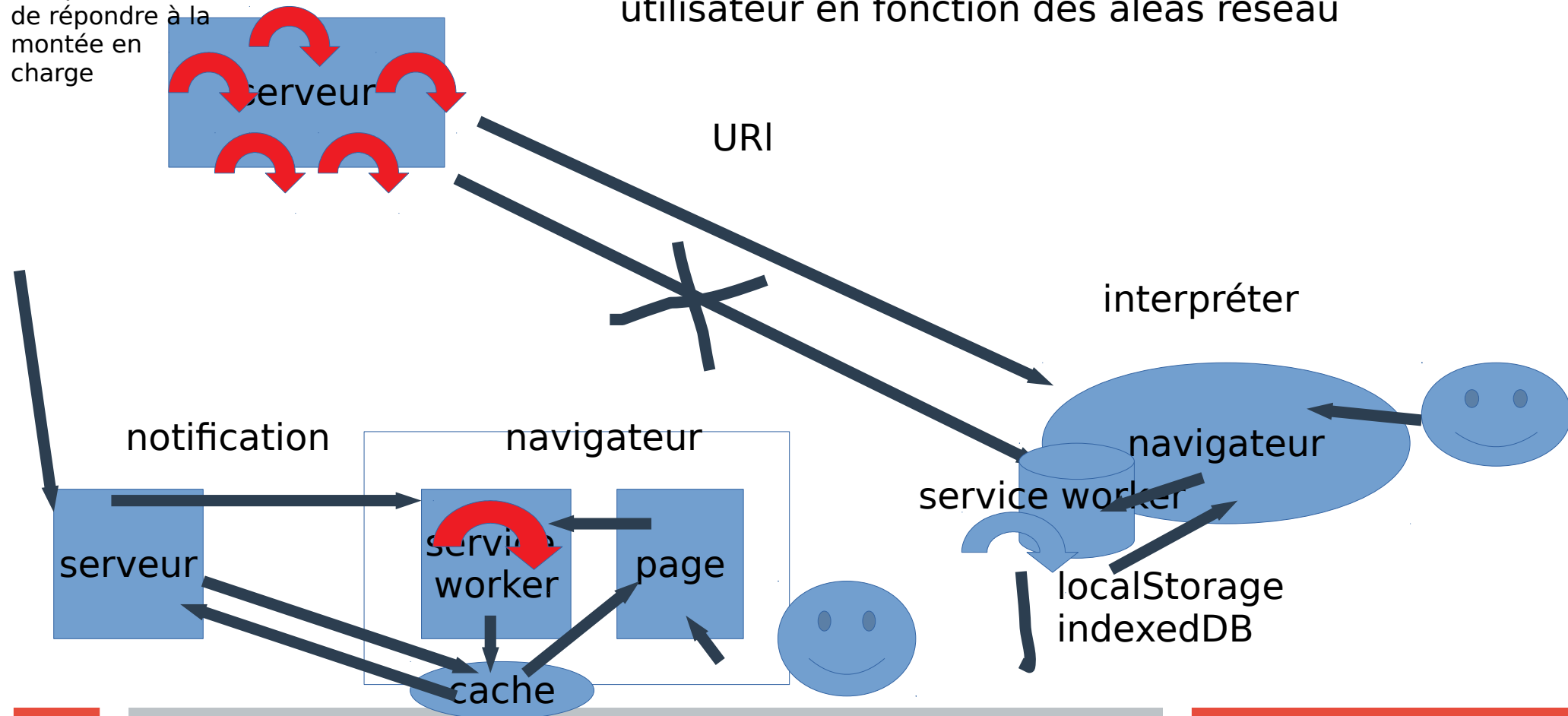
Lighthouse : sur chrome évaluation conformité du site aux concepts PWA



# Accélération des échanges - transfert en temps masqué

scalabilité qui doit permettre de répondre à la montée en charge

utilisation au mieux des ressources du poste client -- amélioration du confort utilisateur en fonction des aléas réseau



# Rappels :

## Variables

Types de variable

(Number, String, Array, Date, Object)

Affectation de type implicite, pas de déclaration

c'est le contenu qui affecte le type à la variable (analogue au langage Python, Matlab , R ...)

Contrairement aux langages avec **déclarations explicites** de type :

Java, C, C++, Typescript ...

## Retour d'expérience

Si pas de typage explicite , pas de contrôle sauf implémentation explicite  
(c'est au développeur de contrôler le contenu de ses variables)

Instruction instanceof ou typeof qui vont permettre de contrôler le type de la variable. (attention instanceof fonctionne que si on utilise le constructeur via l'appel new : t= new string )

Une variable peut contenir du contenu de différentes natures.

# typage: (action de typé ) on définit le type de la variable

typage implicite à partir du contenu de la variable

```
var variable="voiture" ;
```

de façon implicite le système associé l'objet variable au type String

variable.on peut utiliser les fonctions suivantes sur l'objet variable

```
if (variable.indexOf("voit")!= -1) {}
```

```
variable=12;
```

```
if (variable.indexOf("voit")!= -1) {}
```

indexOf function Not found

Notion de classe  
type String

.indexOf()  
.substr()  
.split()  
.replace()

.....  
.length  
.

Notion de classe  
type Number

.toPrecision()  
.toString()  
.toExponential()

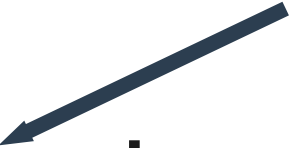
.....

# Autre exemple avec typage explicite

Java

**private int nombre ;**  
**private String chaine ;**  
**private double reel ; // réel nombre flottant**

type == classe



**reel=12.3 ;**

.....

.....

**reel="trouve" ;**

Erreur d'affectation  
pas possible



# Analogie avec les bases de données

## Modèle Relationnel : mysql, mariadb

Attribut appartient à un **domaine** de validité (type)

attribut NOM : VARCHAR(32)

attribut poids : float()

vérifie avant chaque validation de données si les attributs sont bien définis dans leur domaine (sont bien valides)

on peut s'appuyer sur le contrôle final du SGBD

Avant toute validation d'enregistrement  
insert  
update  
contrôle l'appartenance de tous les attributs à leur domaine de validation

## Modèle NoSQL : mongodb, couchBase

pas de contrôle d'intégrité/ .... / au autres

On définit des structures sans typage

Flexibilité  
structure libre  
pas de contrôle de conformité, d'intégrité du côté de la base

à ajouter du côté de l'application



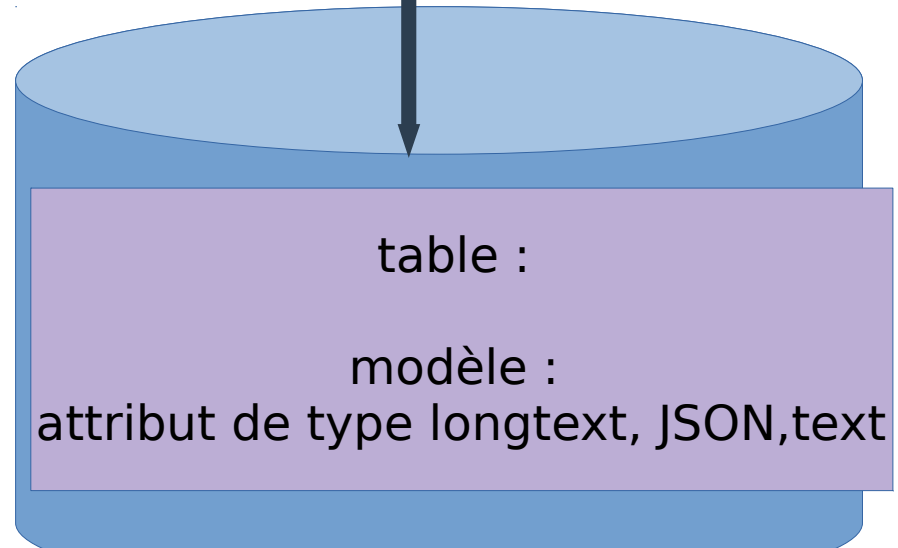
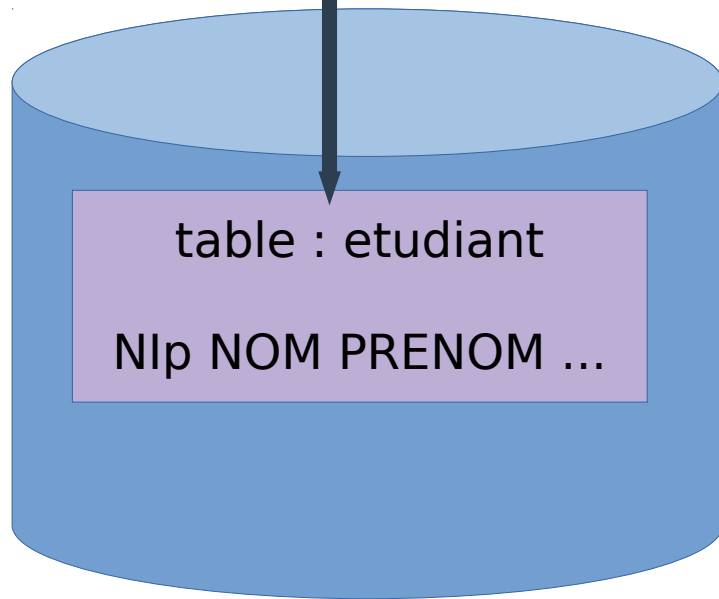
navigateur

ex : contexte CMS

application php  
data : etudiant

application web  
data : struct1 ... struct2 ... struct x  
contrôle au niveau de  
l'application

relationnelle



1FN : normalisation, le schéma ne doit pas  
contenir d'attribut multivalué

Contexte n'est plus normalisé....  
car valeurs multivaluées  
ex : format de type JSON

# Toujours sur le langage

## Langage modulaire

De nombreuses librairies sont disponibles

Payantes ou gratuites

jsxgraph, hightcharts, charts.js, p5.js d3.js ..... go.js

Intérêt des librairies  
permet de construire des  
éléments réutilisables.

Sous forme de fichier js ou minifié

minifié : réduction et suppression des espaces, et éventuellement  
changement des noms de variables ;

inclusion de fichier js directement dans le source

Utilisation des instructions import et export pour insérer des  
fonctionnalités ....



# Démarche de développement

## Utiliser un éditeur (vcs, notepad++ ...) (avec le contrôle syntaxique)

Besoin de l'interpréteur

Console sous le navigateur

ou utilisation de node.js

## Conseils et utilisation

Aspect du code : tabulation (**indentation** -- lisibilité du code) à effectuer de façon systématique : importante pour faciliter l'interprétation et le débogage,

Documenter les parties complexes // commentaires

Mettre des marqueurs dans les différentes étapes du traitement que l'on enlèvera par la suite

**Toujours dans le cadre du débogage : utilisation de la console et des instructions `console.log`, `console.error`, `console.table` pour faciliter la mise au point et la mise en évidence d'événements importants**

# Deux éléments clés

## Définir des structures de données et savoir les utiliser

rappel dans Javascript tout est objet

## Définir des structures de données

Intérêt : piloter un ensemble lié de données plutôt que de piloter chaque donnée de façon individuelle (trop complexe)

Les tableaux et les objets

Les tableaux : liste d'objets de n'importe quelle nature

Les objets : tout type de description

Description sous un format JSON

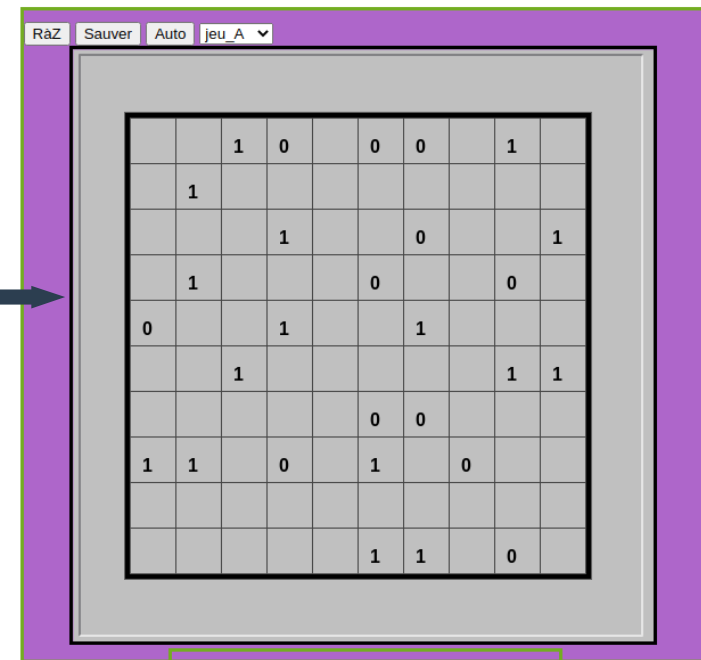
Key, value

On a deux descriptions internes : les données et les comportements (description sous forme de fonction )

# Exemple de structure

```
var configs={ // définition d'un objet   key:value
  "jeu_A":{
    "nbcases":100,
    "name":"A",
    "indications":"","
    "couleur":"lime",
    "positions":[
      [-1, -1, 1,0,-1, 0, 0,-1, 1,-1],
      [-1, 1,-1,-1,-1,-1,-1,-1,-1,-1],
      [-1,-1,-1, 1,-1,-1, 0,-1,-1, 1],
      [-1, 1,-1,-1,-1, 0,-1,-1, 0,-1],
      [ 0,-1,-1, 1,-1,-1, 1,-1,-1,-1],
      [-1,-1, 1,-1,-1,-1,-1, 1, 1],
      [-1,-1,-1,-1,-1, 0, 0,-1,-1,-1],
      [ 1, 1,-1, 0,-1, 1,-1, 0,-1,-1],
      [-1,-1,-1,-1,-1,-1,-1,-1,-1,-1],
      [-1,-1,-1,-1,-1, 1, 1,-1, 0,-1]
    ],
    "descriptif":"Chaque ligne et chaque
    colonne doit contenir le même nombre de 0 et de 1."
  },
  .....
}
```

## Jeu Binero 2020 groupe 13 V2



Chaque ligne et chaque colonne doit contenir le même nombre de 0 et de 1. On ne peut pas placer plus de deux 0 ou de deux 1 côte à côte ou l'un en dessous de l'autre. Deux colonnes et deux lignes ne peuvent être identiques.

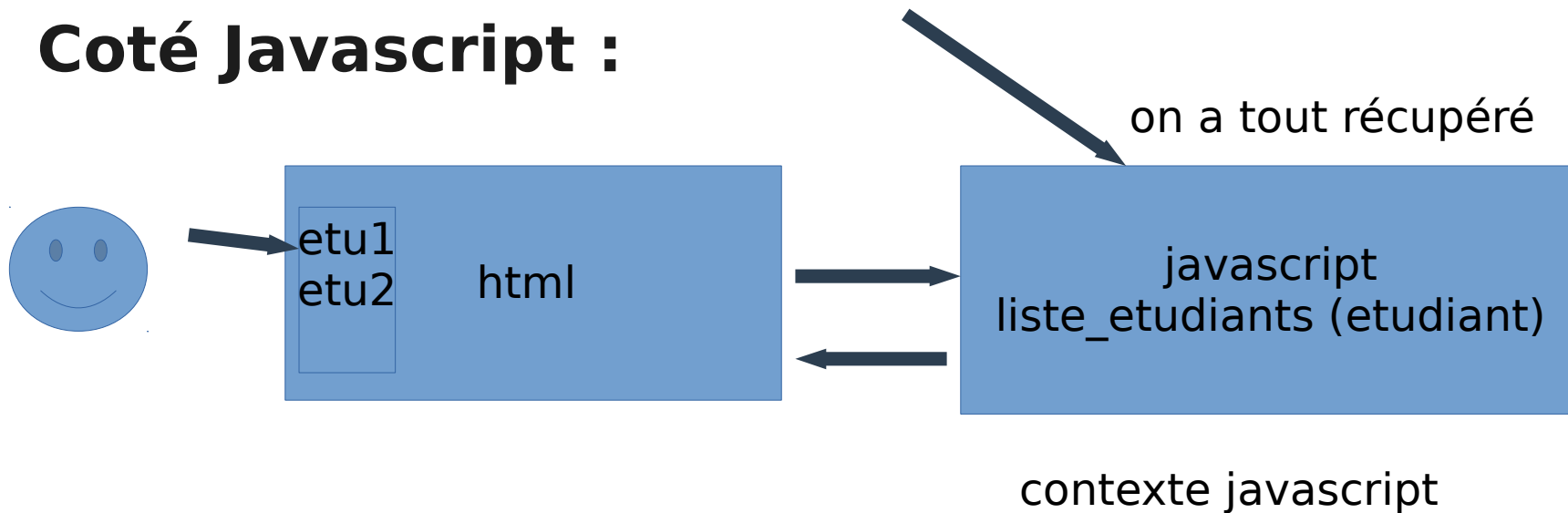
# Exercice de structuration et exploitation de données

Entreprise, Produit, Client

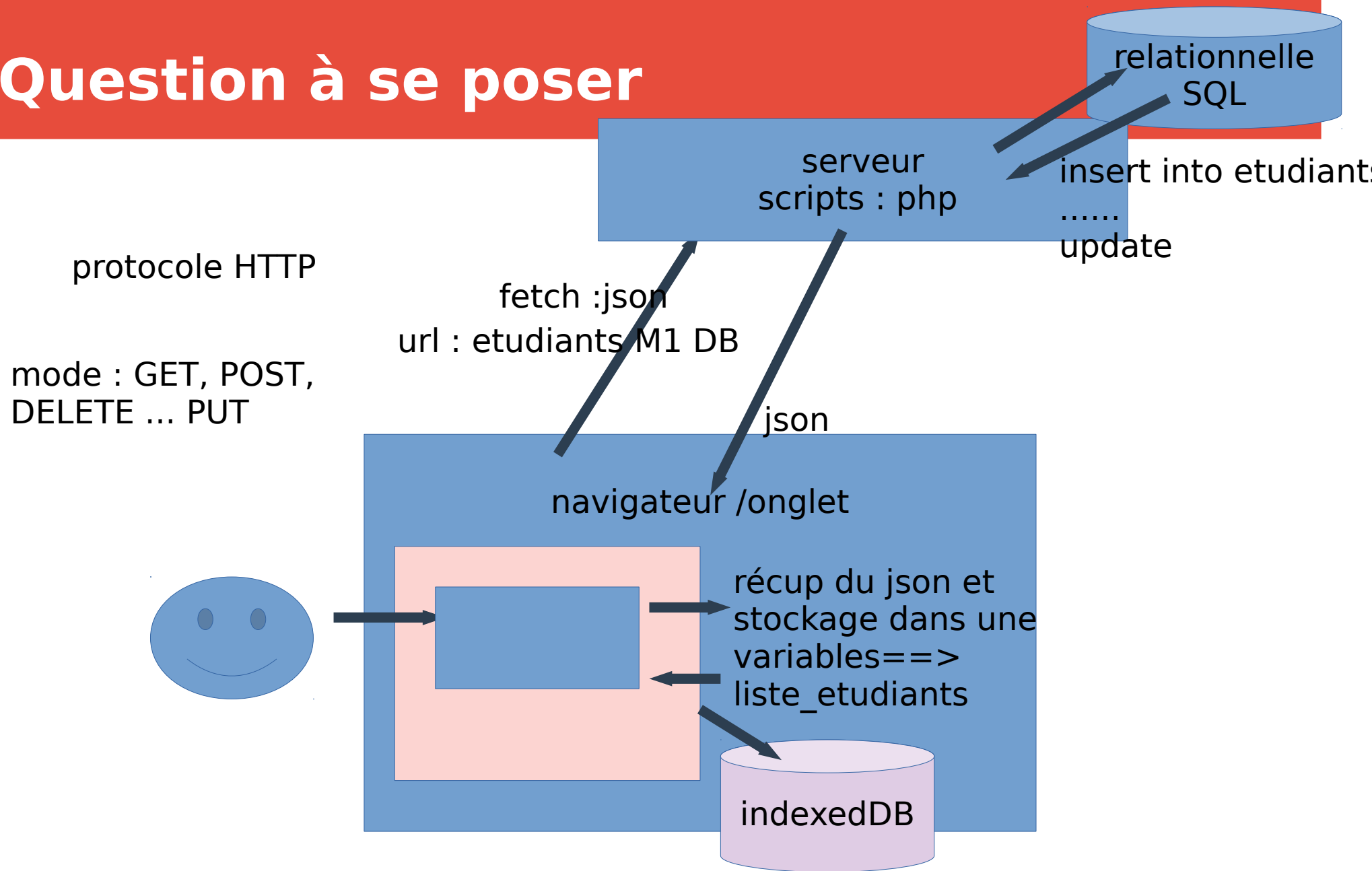
**Etudiant : (NIP, Nom, Prenom, Adresse, email, sexe, notes dans les matières (informatique, économie, mathématiques .....))**

**Côté visuel : input, select, textarea ....**

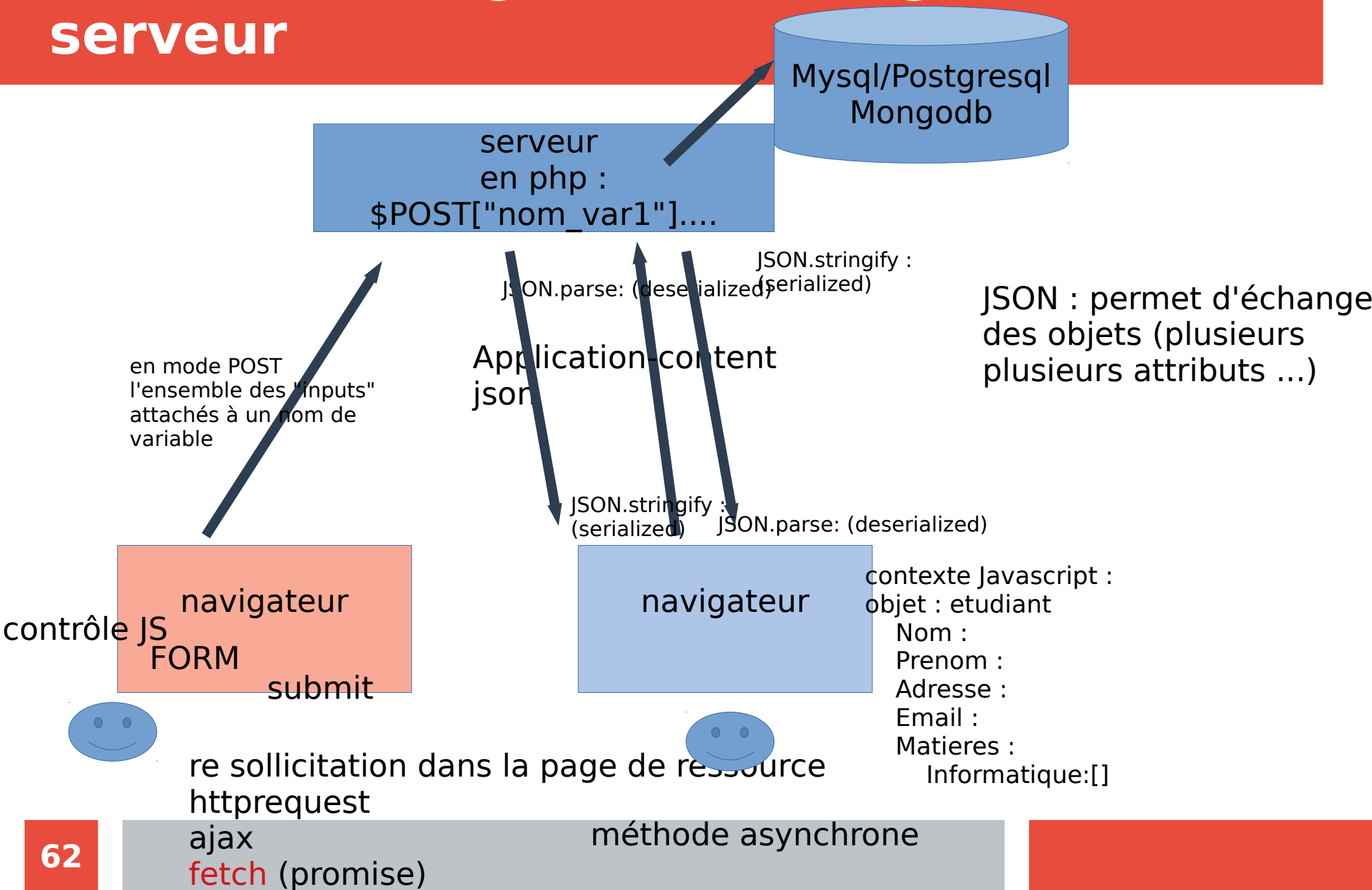
**Coté Javascript :**



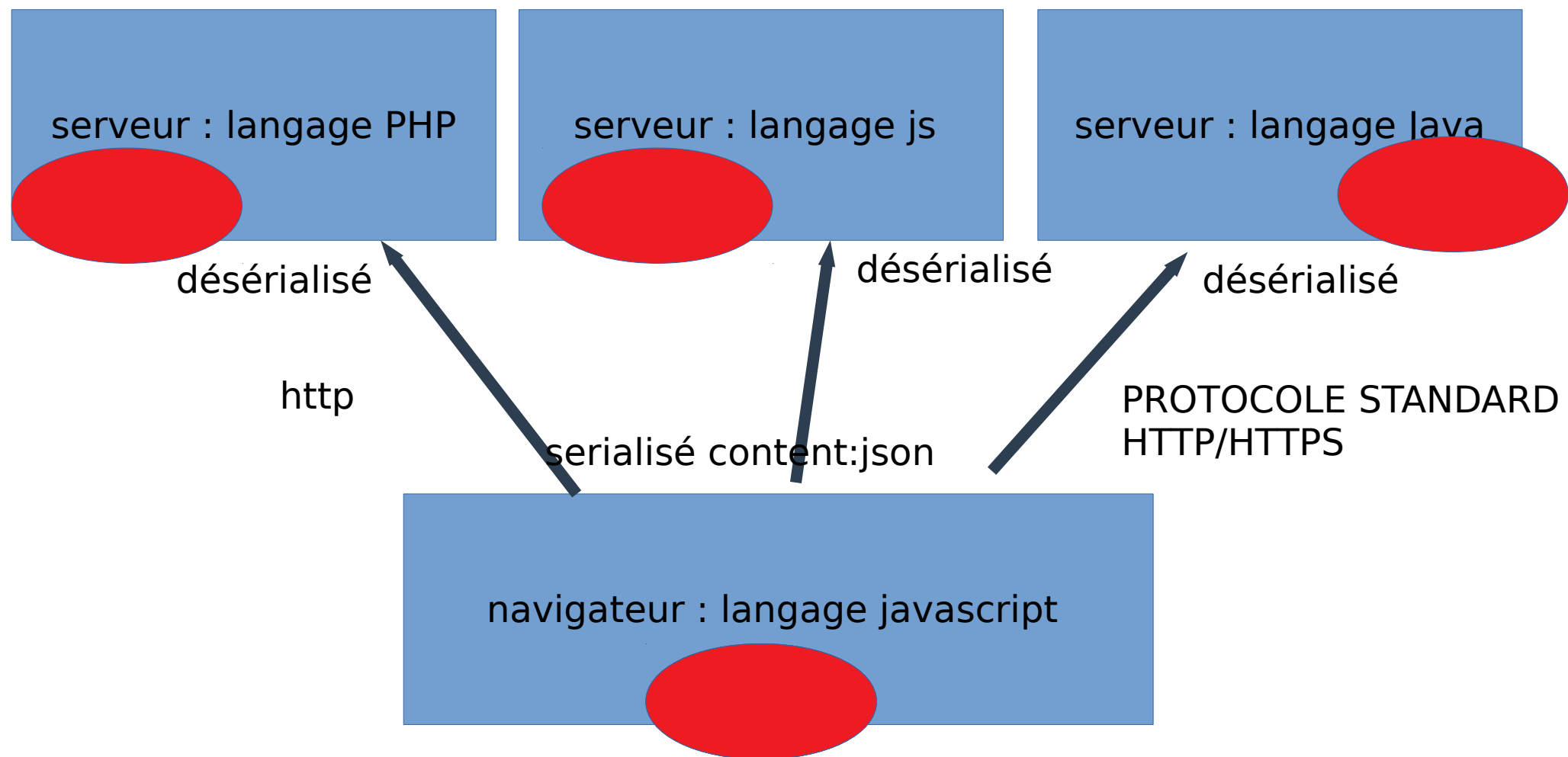
# Question à se poser



# Mode d'échange entre navigateur et serveur



# échange entre contexte et langage différent



# Définition de la structure

key : rouge

```
var etudiant={ } ; /// {key :value }  
etudiant["Nom"]="Stroppa" ;  
etudiant["Prenom"]="Yvan" ;  
etudiant["Email"]="yvan.stroppa@gmail.com" ;  
etudiant["date_naissance"]=new Date("01/01/2000") ;  
etudiant["Matières"]={ } ;  
etudiant["Matières"]["Informatique"]=[12.2,15.5] ;  
etudiant["Matières"]["Economie"]=[12.2,15.5] ;  
etudiant["Matières"]["Mathématiques"]=[12.2,15.8] ;
```

type Array  
[]  
.push()  
.pop()  
.splice()  
.find .....

Ajouter une note :

```
etudiant["Matières"]["Mathématiques"].push(18.2) ;  
etudiant["Matières"]["Economie"].push(16.2) ;
```

Retirer la première note : (on s'appuie sur les fonctions propres à un tableau)

```
etudiant["Matières"]["Mathématiques"].splice(0,1) ;
```

retirer la dernière valeur :

```
etudiant["Matières"]["Mathématiques"].pop() ;
```

Opérations CRUD  
Create  
Read  
Update  
Delete



# Ajout de traitement à l'objet (méthode)

Calcul de moyenne par matière :

```
etudiant.calcul_moy=function()  
{Object.keys(this.Matieres).forEach((e)=>{v=this.Matieres[e].reduce((p,u)=>p+=u,0);console.log("moyenne",e,v/this.Matieres[e].length)}}}
```

Calcul de l'age

```
etudiant["age"]=function() {return ((new Date()-this.date_naissance)/1000/60/60/24/365).toFixed(0);}
```

# Définition de la structure : autre solution

key : rouge

```
var etudiant={} ; /// {key :value }  
etudiant["Nom"]="Stroppa" ;  
etudiant["Prenom"]="Yvan" ;  
etudiant["Email"]="yvan.stroppa@gmail.com" ;  
etudiant["Matières"]={ } ;  
etudiant["Matières"]["Informatique"]["notes"]=[12.2,15.5] ;  
etudiant["Matières"]["Informatique"]["moyenne"]=13,35 ;  
etudiant["Matières"]["Economie"]=[12.2,15.5] ;  
etudiant["Matières"]["Economie"]["moyenne"]=13,35 ;  
etudiant["Matières"]["Mathématiques"]=[12.2,15.8] ;  
etudiant["Matières"]["Mathématiques"]["moyenne"]=13,35 ;
```

type Array  
[]  
.push()  
.pop()  
.splice()  
.find .....

Ajouter une note :

```
etudiant["Matières"]["Mathématiques"].push(18.2) ;
```

```
etudiant["Matières"]["Economie"].push(16.2) ;
```

Retirer la première note : (on s'appuie sur les fonctions propres à un tableau)

```
etudiant["Matières"]["Mathématiques"].splice(0,1) ;
```

retirer la dernière valeur :

```
etudiant["Matières"]["Mathématiques"].pop() ;
```

```
etudiant.ajoute_note=function(no) {  
  // calculer la moyenne et la poussée  
  dans l'attribut moyenne  
  // mettre à jour le tableau des notes
```

# Problématique de mapping objet-relationnel

langages Objets

objet1  
attributs.....

objet2  
attributs.....

attributs  
multivalués

Script php  
parser et définir  
des règles pour  
entreposer les attributs  
au bon endroit

base de données  
Relationnelle

ORM : Object Relationnal Mapping  
Automatique / Manuel

liste  
étudia  
nts

Etudiant : trucmuche

select matières

informatique  
mathématiques

var : etudiant\_encours

Matière courante : informatique

Liste des notes  
12,3  
15,2

+

-

nouvelle note :

15,6

sauvegarde

sélectionne  
l'étudiant

sélectionne la  
matière

sélectionne  
l'opération (add ou  
update/remove)

validation

# Modèle relationnel classique (relation==table)

plusieurs vers plusieurs

Etudiant  
Varchar(10) : NIP  
Varchar(32) : Nom  
Varchar(32) : Prenom  
Email  
Date\_nais

notes  
id\_mat  
NIP  
note

Matiere  
ID\_mat  
Intitule  
Description

contexte Javascript /Objets

```
var etudiant={ } ; /// {key :value }  
etudiant["Nom"]="Stroppa" ;  
etudiant["Prenom"]="Yvan" ;  
etudiant["Email"]="yvan.stroppa@gmail.com" ;  
etudiant["date_naissance"]=new Date("01/01/2000") ;  
etudiant["Matiere"]={ } ;  
etudiant["Matiere"]["Informatique"]=[12.2,15.5] ;  
etudiant["Matiere"]["Economie"]=[12.2,15.5] ;  
etudiant["Matiere"]["Mathématiques"]=[12.2,15.8] ;
```

mapping manuel

# **Adapter ou changer de support**

## **Relationnel**

**On utilise des attributs avec des types :**

**longtext :**

**Json :**

**de stocker des valeurs multivaluées**

**Ou**

**d'utiliser des structures NOSQL :**

**mongodb/couchbase/cassandra/....**

# Exemple de bibliothèque utilitaire

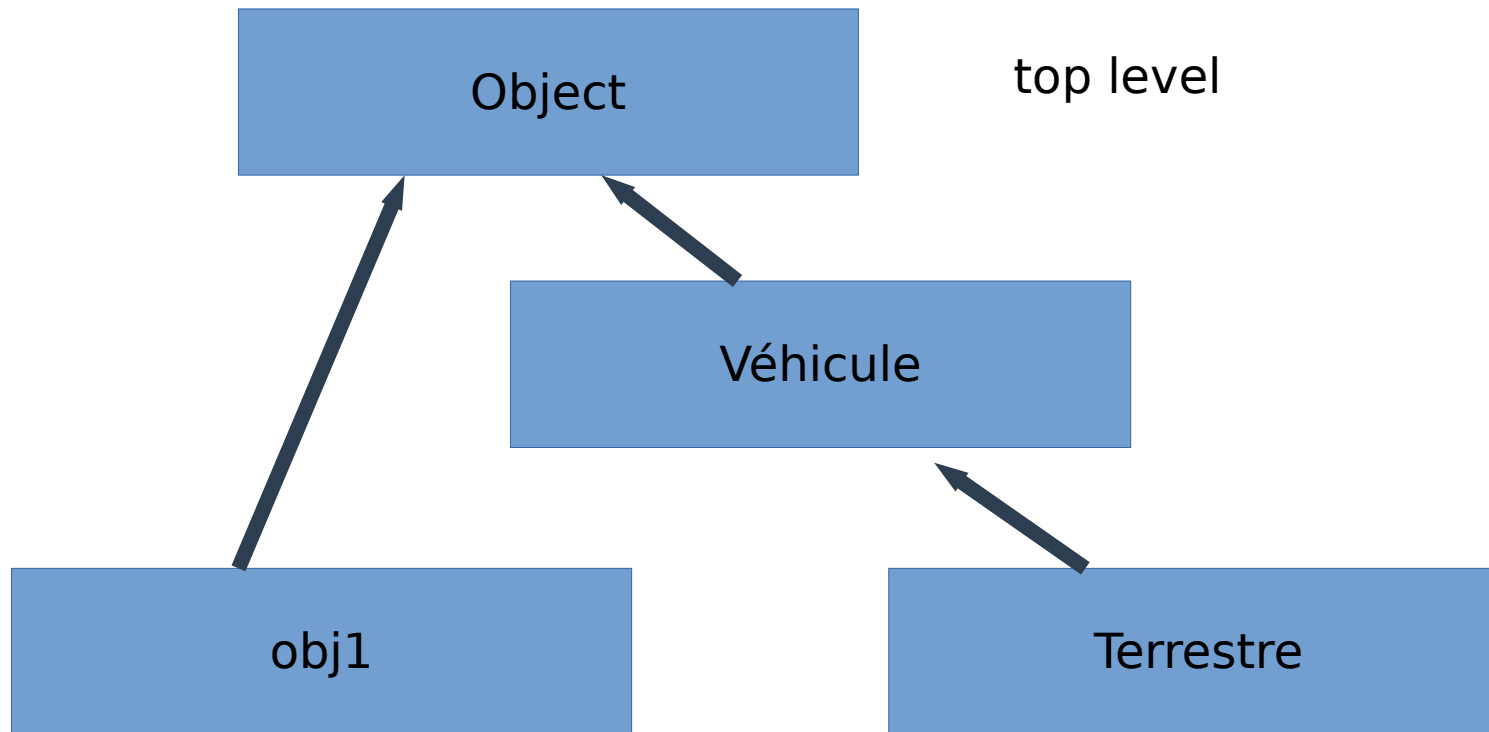
**lodash voir <https://lodash.com/>**

**Permet de disposer d'un ensemble de fonctions pour les manipulations des différentes structures de Javascript**

**à intégrer si besoin dans vos projets et voir la documentation pour différents exemples**

**<https://lodash.com/docs/4.17.15>**

# Langage objet : concepts





# première approche : Fonction constructeur

```
function Voiture(marque, type) {
```

```
  this.type=type ;
```

```
  this.marque=marque ;
```

```
}
```

```
var mavoit=new Voiture("Citroën","c3") ;
```

```
var mavoit1=new Voiture("Citroën","c4") ;
```

```
var mavoit2=new Voiture("Citroën","ds") ;
```

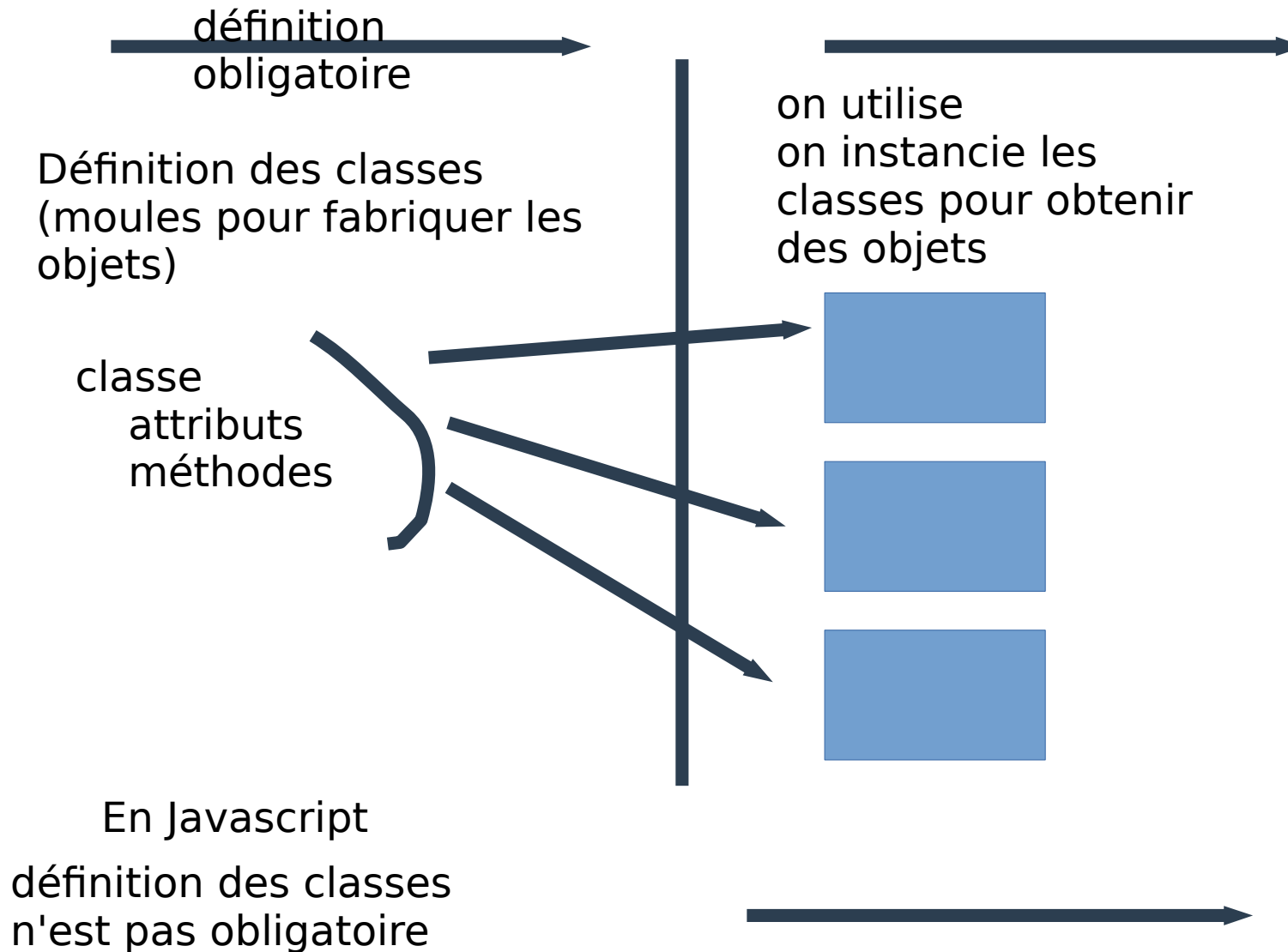
new : opérateur d'instanciation  
création d'un objet issu d'une  
classe

**Redéfinition au niveau de la classe**

```
Voiture.prototype.demarre=function() { console("je  
demarre");}
```

```
Voiture.prototype.puissance=0 ;
```

# modèle classique de langage objets (Java, C++ ...C#)



# Définition de classe == données et traitements

Notion de classe :

```
class etudiant {  
    constructor(a,b ....) {  
        // fonction de construction  
        this.nom=a ;  
        this.prenom=b ;  
        ....  
    }  
    // fonctions associées à cette classe  
    calcul_age() {  
        .....  
        return age;  
    }  
    calcul_moy() {  
        .....  
    }  
}
```

```
Class Name {  
    Attributs  
    .....  
    Méthodes  
    .....  
}
```

Utilisation :

// création d'un objet etud1

var etud1=new etudiant("stroppa","yvan") ;

// appel de la méthode calcul\_age de cet objet  
etud1.calcul\_age() ;

# Concept Objet : static

**Variable ou méthode statique : dites de classe**

**Est attaché directement à la classe et non pas à l'objet**

**Se définit avec le mot clé static dans la description de la classe**

```
class B {  
    static nb_objets=0 ;  
    constructor(hauteur, largeur) {  
        this.attrib1=hauteur ;  
        this.attrib2=hauteur ;  
        B.nb_objets++ ;  
    }  
    method1() {  
        return "methode1" ;  
    }  
    method2() {  
        return "methode2" ;  
    }  
}
```

permet de compter le  
nombre d'objets créés.

# Concept objet : accessibilité privée

Avec le caractère #

On se rapproche un peu plus des langages conventionnels Objets où l'on définir les attributs avant de les utiliser

```
class A {  
    #attrib1=0 ;  
    #attrib2=0 ;  
    constructor(hauteur, largeur) {  
        this.#attrib1=hauteur ;  
        this.#attrib2=hauteur ;  
    }  
    _attrib1() {  
        return this.#attrib1 ;  
    }  
    _attrib2() {  
        return this.#attrib2 ;  
    }  
}
```

les attributs sont  
déclarés en privés  
donc pas accessibles  
directement :

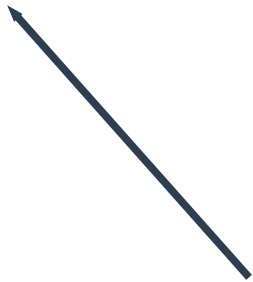
**Encapsulation**

getters

# Concept objet : Héritage

```
Class A {  
    constructor() {  
    }  
}
```

```
var A=new A() ;  
var B=new B() ;
```



```
Class B extends A {  
    constructor() {  
        super() ;  
    }  
}
```

super appel du  
constructeur de la  
classe Mère.

# Illustration héritage

```
class A {  
    constructor(_a,_b) {  
        this.a=_a ;  
        this.b=_b ;  
    }  
}
```

```
class B extends A {  
    constructor(_c) {  
        super() ;  
        this.c=_c ;  
    }  
}
```

# Notion de polymorphisme

**Dans Javascript il n'y a pas possibilité de définir plusieurs méthodes ayant le même nom mais avec des signatures différentes, car il n'y a pas de contrôle spécifique dans l'appel.**

**Soit deux fonctions methode1(a) et methode1(a,b)**

**L'interpréteur prendra la dernière définition :**

```
function methode1(a) {
```

```
}
```

```
const methode1=function() {
```

```
}
```

```
function methode1(a,b) {
```

```
}
```



# Plomorphisme : suite

**d'où l'idée de déclarer des expressions constantes :**

**exemple :**

```
const methode1=function(a) {  
}
```

**Si vous tentez de la définir, il y aura une erreur de l'interpréteur**

# Ecriture des fonctions fléchées

```
function f(x) {  
  x+=1 ;  
  return x;  
}
```

```
const f=function(x) {  
  x+=1 ;  
  return x;  
}
```

```
const f=(x){x+=1; return x;}
```

```
const f=x=>x+=1
```

Si plusieurs d'arguments :

```
const g=(a,b) =>a+=b
```

# Exemple de tableau de fonctions

**Soit const f=function() {...}**

**Soit const g=function() {...}**

const : afin d'éviter toute modification possible du coté client.

**t=[] ;**

**t.push(f)**

**t.push(g)**

**Comment exécuter la bonne fonction ?**

**t[0]() ==> exécute la fonction f**

**t[1]() ==> exécute la fonction g**

# Exemple de tableau de fonctions fléchées

**Soit `const f={()=>{...}}`    ex : `console.log(" fonction ff") ;`**

**Soit `const g={()=>{...}}`    ex : `console.log("fonction gg") ;`**

**`t=[] ;`**

**`t.push(f)`**

**`t.push(g)`**

**Comment exécuter la bonne fonction ?**

**`t[0]() ==> exécute la fonction f      ==> fonction ff`**

**`t[1]() ==> exécute la fonction g      ==> fonction gg`**



# Elaborer la solution pour effectuer ce type de graphe

## Plusieurs possibilités pour effectuer ce type de tracé : exploration solutions existantes et tests dans votre contexte

<https://js.cytoscape.org/>

<https://cytoscape.org/cytoscape.js-tutorial-demo/>  
(demo)

d3.js

Go.js

Choix à faire en fonction du temps (échéance),  
de l'ergonomie recherchée et budget

soit on utilise un canvas

soit on utilise le SVG

...

# Structure HTML : avec un Canvas

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      canvas {display:inline;}
      ul {display:inline;}
      #liste {position:absolute;top: 100px;}
      #trace_graphe {position:absolute; left: 200px;}
    </style>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/js/bootstrap.bundle.min.js"></script>
  </head>
  <body>
    <h1> Tracé graphique des syllabes </h1>
    <select onchange="changement();" id="selecteur">
      <option value="acetylsalicylic">acetylsalicylic</option>
      <option value="beneficiary">beneficiary</option>
    </select>
    <div id="trace_graphe">
      <canvas id="canvas" width="800px" height="400px"></canvas>
    </div>
    <div id="liste" style="width:200px;">
    </div>
  </body>
</html>
```

# Analyse de la problématique

## Que doit on faire pour effectuer ce type de traitement ?

Charger la liste des mots dans le sélecteur

Charger la liste des prononciations associées à la sélection

Afficher le graphe correspondant

## Comment procéder ?



# Définir les différents structures

## Mots à charger :

acetylsalicylic, beneficiar |y

## Détail des compositions possibles :

```
const syllabes1={"syllabe1":{"ben": 4},"syllabe2":{"ɪ": 4, "ə": 4},"syllabe3":{"fɪ": 4},"syllabe4":{"ri": 1, ə: 1, jər: 1, i: 1,"<i>i</i>":2},"syllabe5":{"i: 1, er: 2,ər:2},"syllabe6":{"i: 2}};  
const syllabes={"syllabe1":{"æ": 2},"syllabe2":{"ɪ": 2,"ə": 2,"ɪt": 2,"ət": 2,},"syllabe3":{"taɪ<sup>ə</sup>l": 2,"<i>a</i>t<sup>ə</sup>ɪ<sup>ə</sup>l": 2,"ɪl": 3, "<sup>ə</sup>l": NaN},"syllabe4":{"sæl": 3},"syllabe5":{"ə": 2,"<i>ə</i>":3},"syllabe6":{"sɪl": 3},"syllabe7":{"ɪk": 3}};
```

## Liste des prononciations :

```
const tableau_prononciations1=[ ",ben ɪ 'fɪj ri",",ben ə 'fɪj ri",",ben ɪ 'fɪj əɾ i",",ben ɪ 'fɪj jər i",",ben ə 'fɪj əɾ i",",ben ə 'fɪj jər i", ",ben ɪ 'fɪj <i>i</i> əɾ i",",ben ə 'fɪj i əɾ i",",ben ɪ 'fɪj ri",",ben ə 'fɪj ri",",ben ɪ 'fɪj əɾ i",",ben ə 'fɪj əɾ i",",ben ɪ 'fɪj <i>i</i> er i",",ben ə 'fɪj i er i",",ben ɪ 'fɪj i er i",",ben ə 'fɪj i er i"];  
const tableau_prononciations=[  
",æ ɪ <i>a</i>t<sup>ə</sup>ɪ<sup>ə</sup>l ,sæl <i>ə</i> 'sɪl ɪk",",æ ɪ ɪl ,sæl ə 'sɪl ɪk",",æ ɪ <sup>ə</sup>l ,sæl ə 'sɪl ɪk",",æ ə taɪ<sup>ə</sup>l ,sæl ə 'sɪl ɪk",",æ ə ɪl ,sæl ə 'sɪl ɪk",",æ ə <sup>ə</sup>l ,sæl ə 'sɪl ɪk",",æ ɪt taɪ<sup>ə</sup>l ,sæl ə 'sɪl ɪk",",æ ɪt ɪl ,sæl ə 'sɪl ɪk",",æ ɪt <sup>ə</sup>l ,sæl ə 'sɪl ɪk",",æ ət taɪ<sup>ə</sup>l ,sæl ə 'sɪl ɪk",",æ ət ɪl ,sæl ə 'sɪl ɪk",",æ ət <sup>ə</sup>l ,sæl ə 'sɪl ɪk";
```

# Pour démarrer

**Créer le fichier html**

**Copier les structures de données**

**On définit le fonctionnement de l'application :**

Lorsque l'utilisateur sélectionne un mot il faut pouvoir effacer la liste des prononciations et lui afficher la bonne. Ensuite lui faire le graphe

**On va avoir besoin de petites fonctions utilitaires**

De type :

Efface une liste

Chargement de la liste

# Fonctions utilitaires

**Effacer un objet chose de la DOM c'est :**

**- soit modifier sa propriété visible à l'aide de l'attribut display défini dans le style de l'objet :**

display:none ou display:block

**- soit supprimer tous ses enfants (attention à ne pas le supprimer car on va en avoir besoin de lui pour accrocher des descendances à la nouvelle sélection du mot**

pour supprimer ses enfants il faut passer en revue si il a des descendants et les supprimer

```
const raz_liste=function(Liste) {  
  let liste=document.getElementById(Liste);  
  if (liste != null )  
    while (liste.firstChild) {  
      liste.removeChild(liste.lastChild);  
    }  
}
```

Utilisation de la fonction  
raz\_liste("liste")

# Fonctions utilitaires

**On a besoin d'une fonction pour ajouter des éléments de type div, p, li ... dans la DOM**

```
const addElement=function(parentId, elementType, elementId, contenu,display="display:block") {  
    var p = document.getElementById(parentId);  
    var newElement = document.createElement(elementType);  
    newElement.setAttribute("id", elementId);  
    newElement.innerHTML = contenu;  
    newElement.setAttribute("style", display);  
    p.appendChild(newElement);  
}
```

# Exemple d'utilisation et d'appel

Illustration avec le chargement de la liste dans la zone prévue à cet effet. On crée une variable de type String avec l'ensemble des éléments de visualisation que l'on passe à la fonction.

```
// -----  
// Chargement des prononciations et de leur animations  
// -----  
const chargement_liste=function(tab_prononciations, nb) {  
    let chaine="";  
    chaine+="    for (let mot of tab_prononciations) {  
        chaine+="- 
        chaine+=mot;  
        chaine+="    }  
    chaine+="    addElement("liste", "p","id_liste",chaine);  
}
```

Pour les fonctions `debut_eclairage` et `fin_eclairage` on pourra dans un premier les définir vide ....

# Etape 1

**Prendre l'ensemble des éléments pour constituer votre projet afin d'expérimenter, tester et d'adapter les différents illustrations qui vous sont fournies.**

**On pourra s'appuyer sur la console pour utiliser et tester les différents codes afin de comprendre le comportement.**

**A réaliser : un fichier principal html index.html et un fichier comportement.js dans lequel on va retrouver les définitions et la gestion du comportement.**

## Etape 2 : construction du graphe

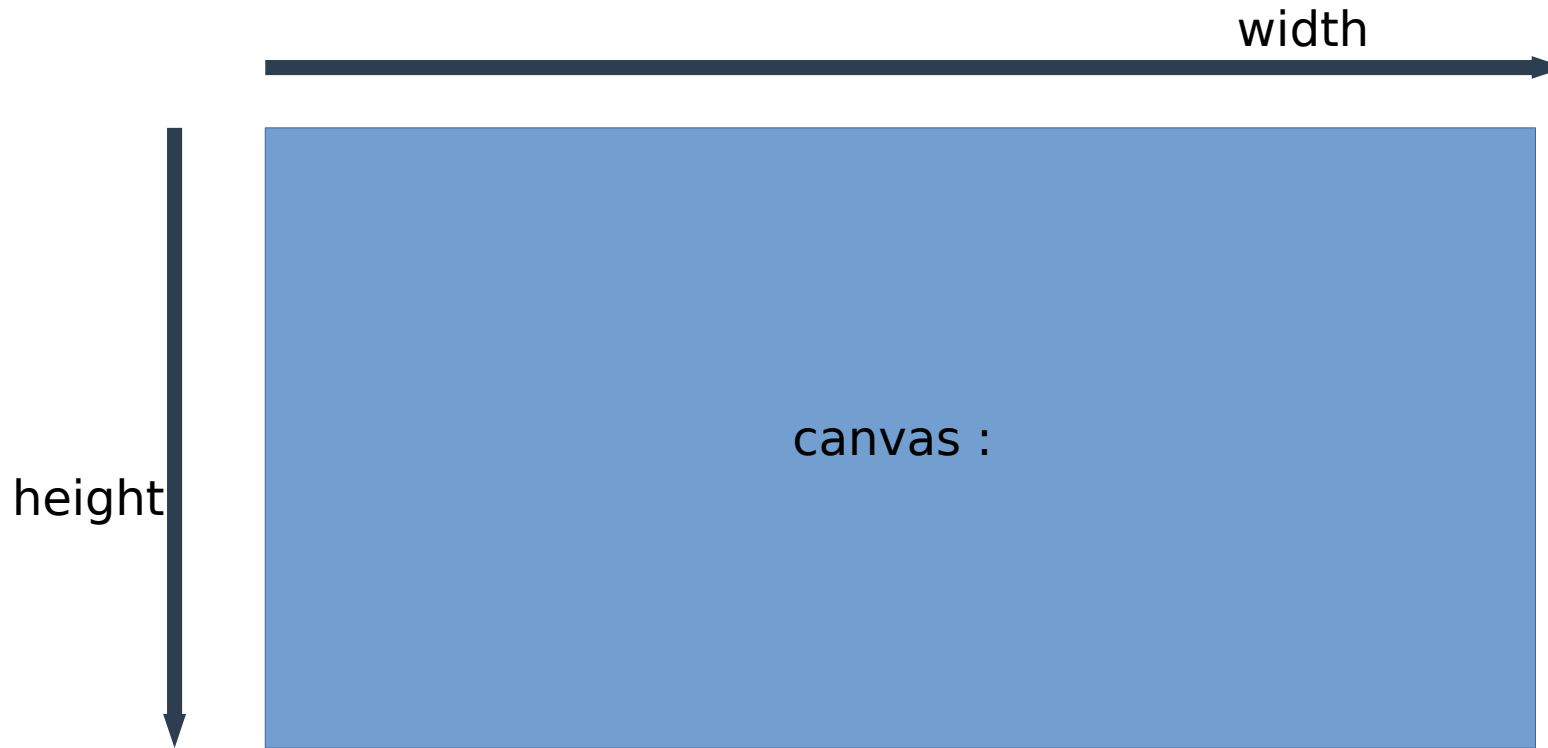
### **On peut analyser ce problème de la manière suivante :**

On doit construire un graphe contenant des nœuds.

La structure du graphe peut-être représentée par une zone rectangulaire que l'on va découper en segment verticaux dans lesquels on va positionner les nœuds.

On va fournir une liste de prononciations associée à un mot (tableau de prononciations) dont chacune est composée de syllabes qu'il faut représenter sous forme de cercle à une position précise sur X (qui va représenter le numéro de la syllabe ) et la position y

# Exemple de l'approche : étape 1



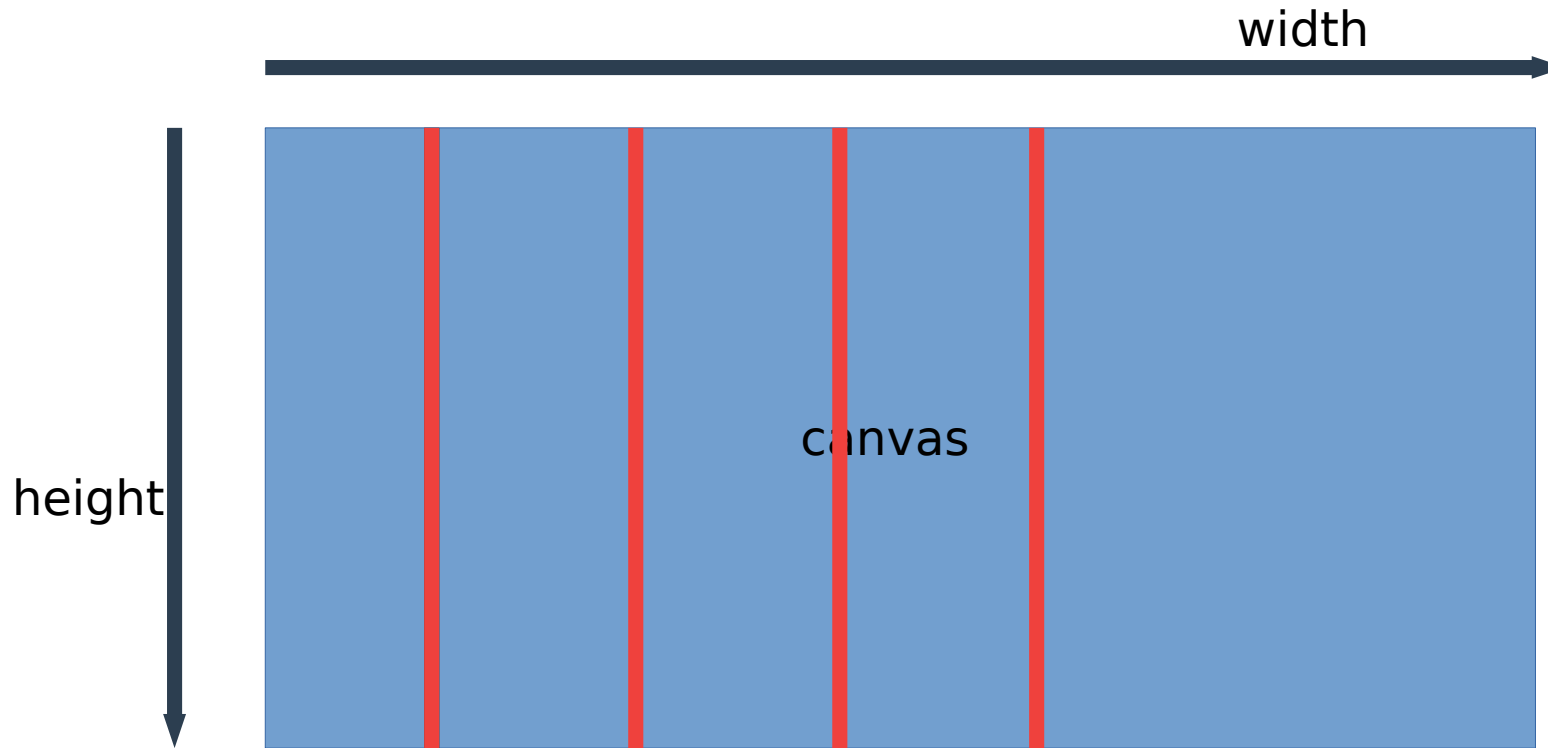
le canvas est repéré par son ID et possède les dimensions width et height (voir dans la déclarations html)

```
<div id="trace_graphe">  
  <canvas id="canvas" width="800px" height="400px"></canvas>  
</div>
```



# Exemple de l'approche : étape 2

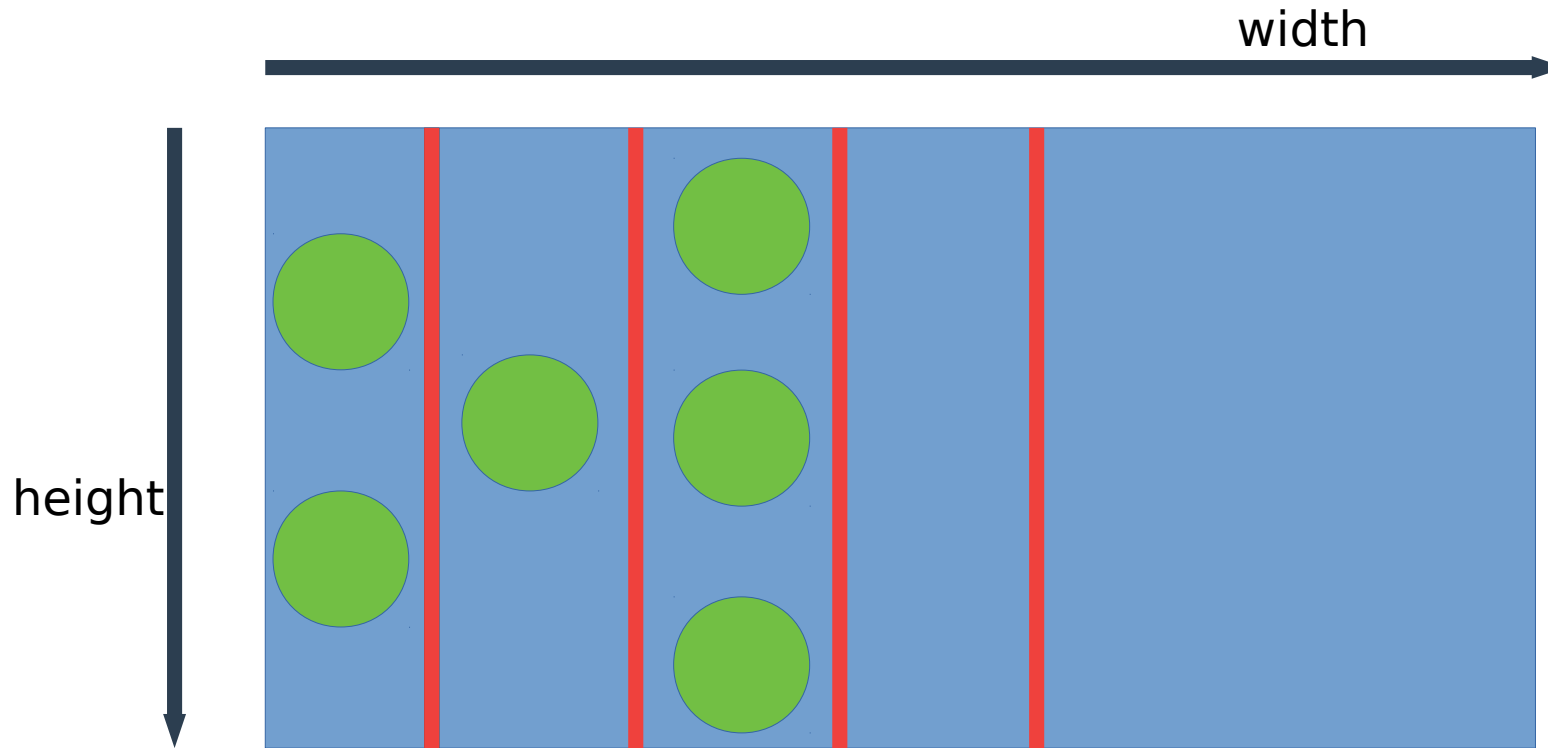
## découper la zone par syllabe



Tracé d'une ligne verticale  
de position sur  $x = \text{num\_colonne} * \text{width} / \text{nb\_colonnes}$   
( $\text{nb\_colonnes} == \text{nb\_syllabes}$ )

```
// trace d'une ligne
const drawLine=function(h,xi,yi,xf,yf,epaisseur){
  h.lineWidth = epaisseur;
  h.beginPath();
  h.moveTo(xi,yi);
  h.lineTo(xf,yf);
  h.stroke();
}
```

# Exemple de l'approche : étape 3 positionner les noeuds



Pour positionner les nœuds, on pourra évaluer le nombre de nœuds par colonne et les positionner les uns au dessus des autres en respectant un intervalle fixe.

# Analyse sous forme objet

**On va définir une classe nommée Cgraphe qui va représenter l'ensemble du graphe. Et une classe Cnoeud qui va représenter un nœud**

**Pour construire le graphe il nous faut l'id du canvas pour le tracer, les prononciations à représenter.**

**Ensuite il faut que l'objet Cgraphe instancié à l'aide de l'instruction suivante `new Cgraphe(id_canvas, 'mot',echelle)` définisse ses structures internes en fonction du mot. Qu'il puisse générer des objets Cnoeud pour chaque syllabe appartenant aux prononciations dans la bonne colonne et à la bonne position. Une syllabe dans une colonne ne peut être représentée qu'une seule fois.**

# Création d'une classe

```
class nom_classe {  
}
```

**La classe définit des attributs et des méthodes.**

**Les attributs de la classe se définissent à l'aide du mot clé this**

**Parmi les méthodes on distingue la méthode constructor qui permet d'instancier un objet à partir de cette classe à l'aide de l'opérateur new**

```
class nom_classe {  
    constructor(A,B) {  
        this.a=A ;  
        this.b=B ;  
    }  
    methode1() {  
    }  
    ....  
}
```

c'est la fabrique  
des objets

Instructions pour créer des  
objets issus de cette classe

```
var cl=new nom_classe(a,b)
```

Autre exemple : l'utilisation de date  
let d=new Date()

## **intérêt :**

**C'est de définir une structure de données associées à son propre comportement et de construire un assemblage d'objets qui possèdent et gèrent leur propre affichage. De plus l'ensemble est réutilisable ....**

# Création classe

**On va définir deux classes :**

## **Cgraphe :**

Définir un constructeur qui va recevoir les arguments : identifiant , l'id du canvas, l'expression à afficher, echelle

Identifiant : pour repérer le graphe et le manipuler à partir de cet identifiant. Si on souhaite cumuler plusieurs graphes à partir d'une liste ....

Id du canvas : référence du canvas où l'on va tracer le graphe

expression : le mot à afficher dans ce graphe

echelle : effet de zoom

## **Cnoeud :**

Définir un constructeur : qui va recevoir les arguments suivants :

id : identifiant du noeud

graphe : parent dans lequel le noeud va être tracé

zone : colonne dans laquelle le noeud sera tracé

position : position dans la colonne

id canvas : identification du canvas dans lequel sera tracé le noeud

graphe : parent permet d'avoir le lien entre le noeud et son parent

# Analyse de la classe Cnoeud :

## Cnoeud :

constructor : initialisation des propriétés d'un noeud

trace() : affiche le noeud dans le canvas : cercle avec le texte

trace\_surbrillance() : change les propriétés visuelles du texte et du cercle

trace\_lien\_avant() : trace les liens vers les voisins (on choisira un sens)

## Un noeud pour ce tracé a besoin :

du canvas dans lequel il doit être tracé

de la colonne et de sa position verticale (pixels)

du texte (contenu de la syllabe qu'il doit tracer)

# Définition de la première classe

**On va définir la première classe Cnoeud qui va permettre d'afficher les syllabes dans une colonne et à une position Y donnée.**

```
class Cnoeud {
    constructor(id,contenu,colonne,position, IdCanvas) {
        this.id=id+contenu;
        this.colonne=colonne;
        this.position=position;
        this.contenu=contenu;
        this.couleur="black";
        this.canvas=IdCanvas;
        this.hh=document.getElementById(IdCanvas).getContext('2d');
    }
    // trace le noeud et son contenu
    trace() {
        this.hh.canvas.style.border="3px solid #000";
        this.hh.lineWidth = 2;
        this.hh.fillStyle="white";
        this.hh.beginPath();
        this.hh.arc((this.colonne-1)*L+L/2, this.position, L/4, 0, 2 * Math.PI);
        this.hh.fill();
        this.hh.stroke();
        this.hh.font = 'bold 16px serif';
        this.hh.strokeStyle="#003300";
        this.hh.fillStyle = "black";
        this.hh.fillText(this.contenu, (this.colonne-1)*L+L/2-L/6, this.position);
    }
}
```

L : représente  
la largeur  
d'une colonne

Utilisation du this  
permet d'indiquer  
que l'on utilise  
l'objet lui même



# Utilisation et Tests

**A partir de la console de votre navigateur, vous pouvez instancier dans le contexte de votre application un objet de type Cnoeud à l'aide**

```
var t=new Cnoeud(21,"kd",1,200,"canvas") ;  
// première colonne et à une hauteur de 200  
pixels dans le canvas  
t.trace() ;
```

# La classe Cgraphe

**A implémenter et doit permettre de construire un graphe constitué de noeuds pour une représentation des chemins que constituent les différentes prononciations**

**A vous de jouer ?**

# Le tracé des chemins

**Une question à résoudre est comment relier les noeuds qui constituent les différents chemins de prononciations d'un mot.**

# Comment trouver les voisins pour tracer les liens entre les noeuds

**Il faut trouver les voisins des noeuds, pour se faire on va essayer de composer la matrice d'incidence (tableau) elle permet d'indiquer qui est voisin de qui.**

**Dans notre cas, on va la composer de la manière suivante : chaque sommet est nommé par son contenu .... et contiendra la liste des autres sommets à l'aide d'une clé. Ensuite lorsqu'on parcourt les prononciations on complétera chaque liste en fonction du noeud qui suit.**

# Illustration pour la liste des prononciations : acetylsalicylic

On liste toutes les syllabes et on crée une structure lignes de la manière suivante :

`ligne["syll1"]={}`

Pour toutes les syllabes on note :

`ligne["syll1"]["syll"+i]=0 ;`

syll1	syll1:0	syll2:0	syll3:0	syll4:0	..	sylln:0
syll2	syll1:0	syll2:0	syll3:0	syll4:0	..	sylln:0
syll3	syll1:0	syll2:0	syll3:0	syll4:0	..	sylln:0
syll4	syll1:0	syll2:0	syll3:0	syll4:0	..	sylln:0
syll5	syll1:0	syll2:0	syll3:0	syll4:0	..	sylln:0

sylln	syll1:0	syll2:0	syll3:0	syll4:0	..	sylln:0
-------	---------	---------	---------	---------	----	---------

# Remplissage de la structure

**A la lecture des prononciations, on complète le tableau pour indiquer qui est à côté de qui on ajoutant par ex si syll1 précède syll2 ligne[syll1][syll2]=1**

# Compléments :

**Virtualisation**

**Conteneurisation**

# Outils de virtualisation/conteneurisation

**Plusieurs outils peuvent nous accompagner dans nos développements et nous permettent de prototyper et développer en toute autonomie.**

**Ce sont dans un premier temps les hyperviseurs de type VMware, VirtualBox, Parallel, Hyper V ...que l'on peut installer sur son poste de travail de façon assez simple.**

**On distingue dans**

**L'objectif est de travailler dans un contexte cloisonner et de conserver intacte l'environnement de votre Hôte.**



# Concept PWA

## **Notion de promise**

Api Fetch

## **Rôle et installation d'un service worker**

## **Rôle et installation d'un manifest**

## **Fonctionnement Offline**

## **Principe des caches locaux**

Cookies

Local storage

IndexedDB

# Promises fetch

## Principe d'une promesse :

exécution asynchrone

gestion du retour de l'exécution

.then

## Possibilité d'exécution en cascades des promises

## Différents du fonctionnement des callbacks

# Exemple d'utilisation

```
const appel_get_enregistrement_locuteur=function(corpus,locuteur) {  
  const critere = {"Corpus":corpus,"Locuteur":locuteur };  
  fetch("/personnalisation/get_enregistrement_locuteur.php", {  
    method: "POST",  
    mode: "same-origin",  
    credentials: "same-origin",  
    headers: {  
      "Content-Type": "application/json"  
    },  
    body: JSON.stringify(critere)  
  })  
  .then((res) => res.json())  
  .then((data) => {  
    console.log(corpus,data);  
    affichage_enregistrements_locuteurs(data.details);  
  })  
  .catch((error) => console.log(error))  
}
```

# Méthodes Ajax

(ex : <https://www.pierre-giraud.com/jquery-apprendre-cours/creation-requete-ajax/>)

```
$(document).ready(function(){
    $.ajax({
        //L'URL de la requête
        url: "une/url/au/choix",
        //La méthode d'envoi (type de requête)
        method: "GET",
        //Le format de réponse attendu
        dataType : "json",
    })
    //Ce code sera exécuté en cas de succès - La réponse du serveur est passée à done()
    /*On peut par exemple convertir cette réponse en chaîne JSON et insérer
    * cette chaîne dans un div id="res"*/
    .done(function(response){
        let data = JSON.stringify(response);
        $("#div#res").append(data);
    })

    //Ce code sera exécuté en cas d'échec - L'erreur est passée à fail()
    //On peut afficher les informations relatives à la requête et à l'erreur
    .fail(function(error){
        alert("La requête s'est terminée en échec. Infos : " + JSON.stringify(error));
    })

    //Ce code sera exécuté que la requête soit un succès ou un échec
    .always(function(){
        alert("Requête effectuée");
    });
});
```