Cours Javascript-PWA
V1.0 08/04/2022

Y. Stroppa 2022

Introduction

Première partie :

Rappel

Exercice et mise en œuvre

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

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

Deuxième partie : Javascript avancé

PWA : explication et mise en œuvre

Rendu/projets

Définition des conditions du projet

- groupe (2,3 étudiants)

- Etape 1 : définition et validation : définir son projet / **valider** (expliquer objectifs/détailler si besoin)

- Etape 2 réalisation : réaliser le site + ajouter les fonctionnalités avancées de PWA ....

-- rendre en Juin (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 ??????

Présentation ???? de mettre en accès les rapports pour PSB(étudiants M2) et .... mettre en ligne vos réalisations (sous domaine) et en accès limité (étudiants M2) ...

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)

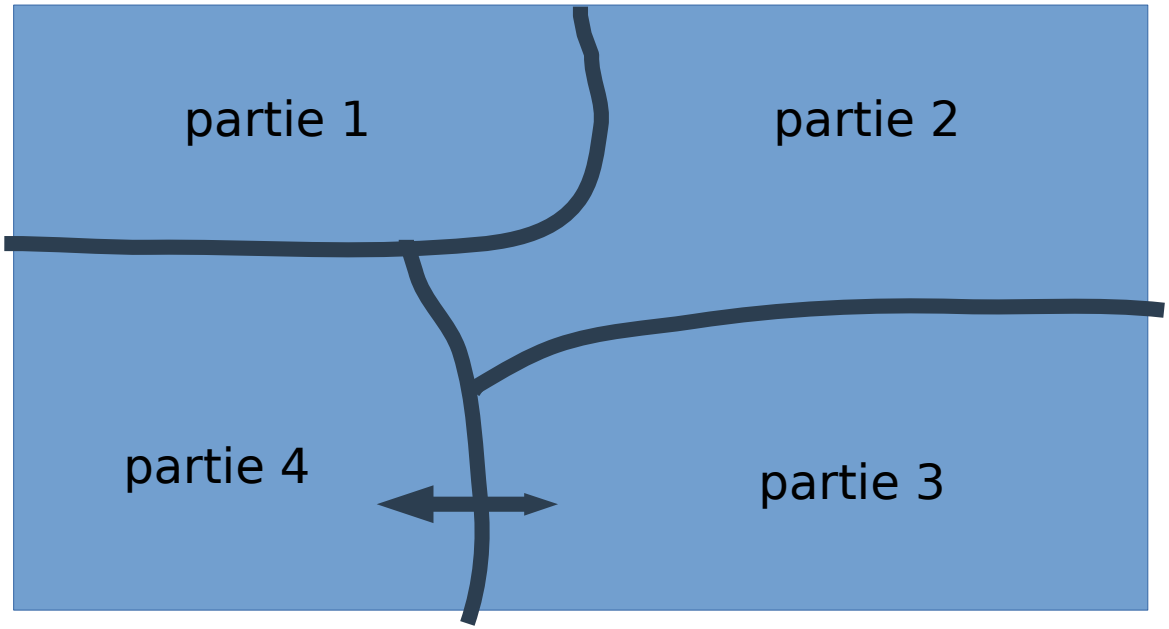
  (hors de question de modifier une librairie/solution importée valide pour la partie JS / PHP)

Epuisée les solutions existantes

- Développe from scratch (on a la main)

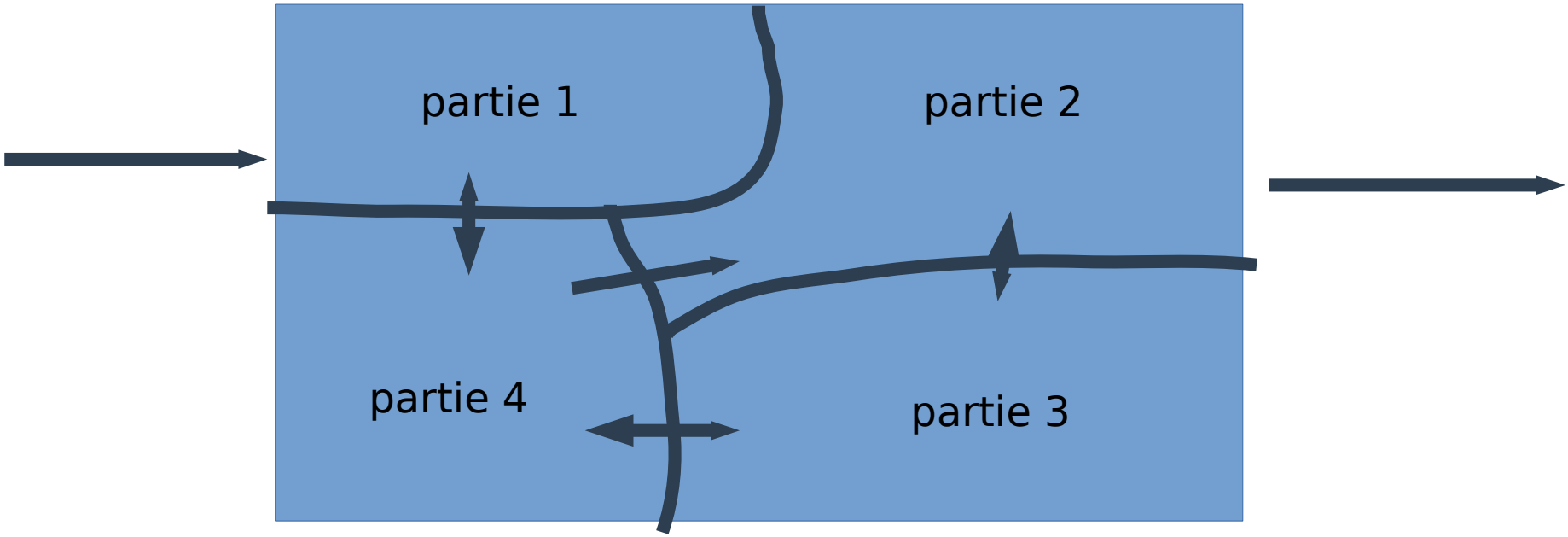
  démarche à prendre en compte dans le développement (tests ……)

découpage



tests unitaires

Intégration des différentes parties



Rappels

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

Partie infrastructure Web :

Service Web

Protocoles http(local) et https :

Ports : 80 et 443

http: contexte portable

https : contexte public/entreprise

scripts
- php
- js
- python
- java

Bases de données
- Solutions possibles :
  - Relationnelles :
    - Mysql,
    - Mariadb,
    - Postgresql,
    - Oracle,
    - Sqlserver ...
  - noSQL :
    - Mongodb
    - Cassandra
    - Couchbase
    - Neo4j
    - ....

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)

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 : Mongodb
- couchBase
- Neo4j ....

Relationnelles — Langage SQL

NoSQL — Langages Javascript N1QL

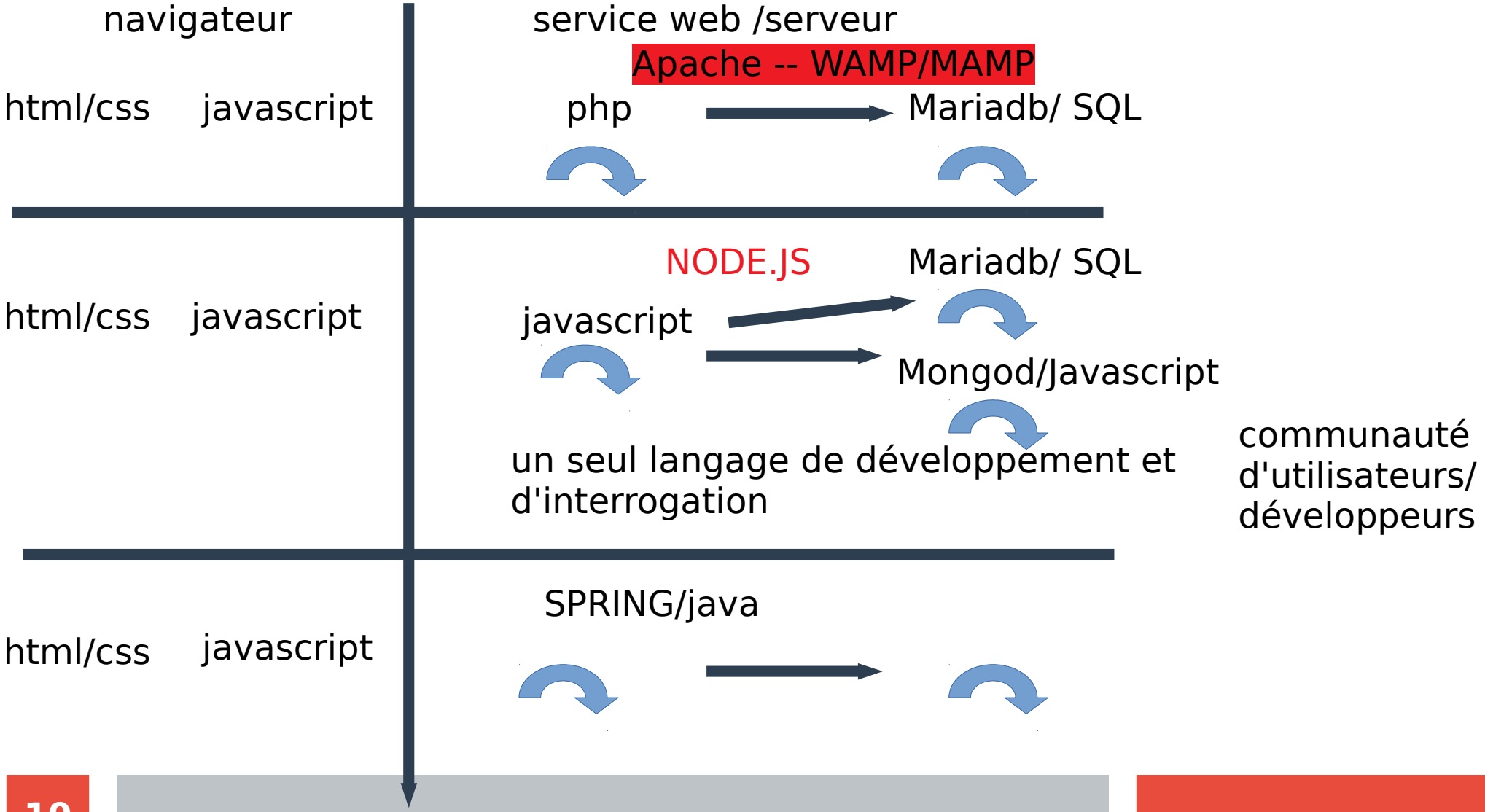
Traitements internes

Structures de données sous une forme particulière

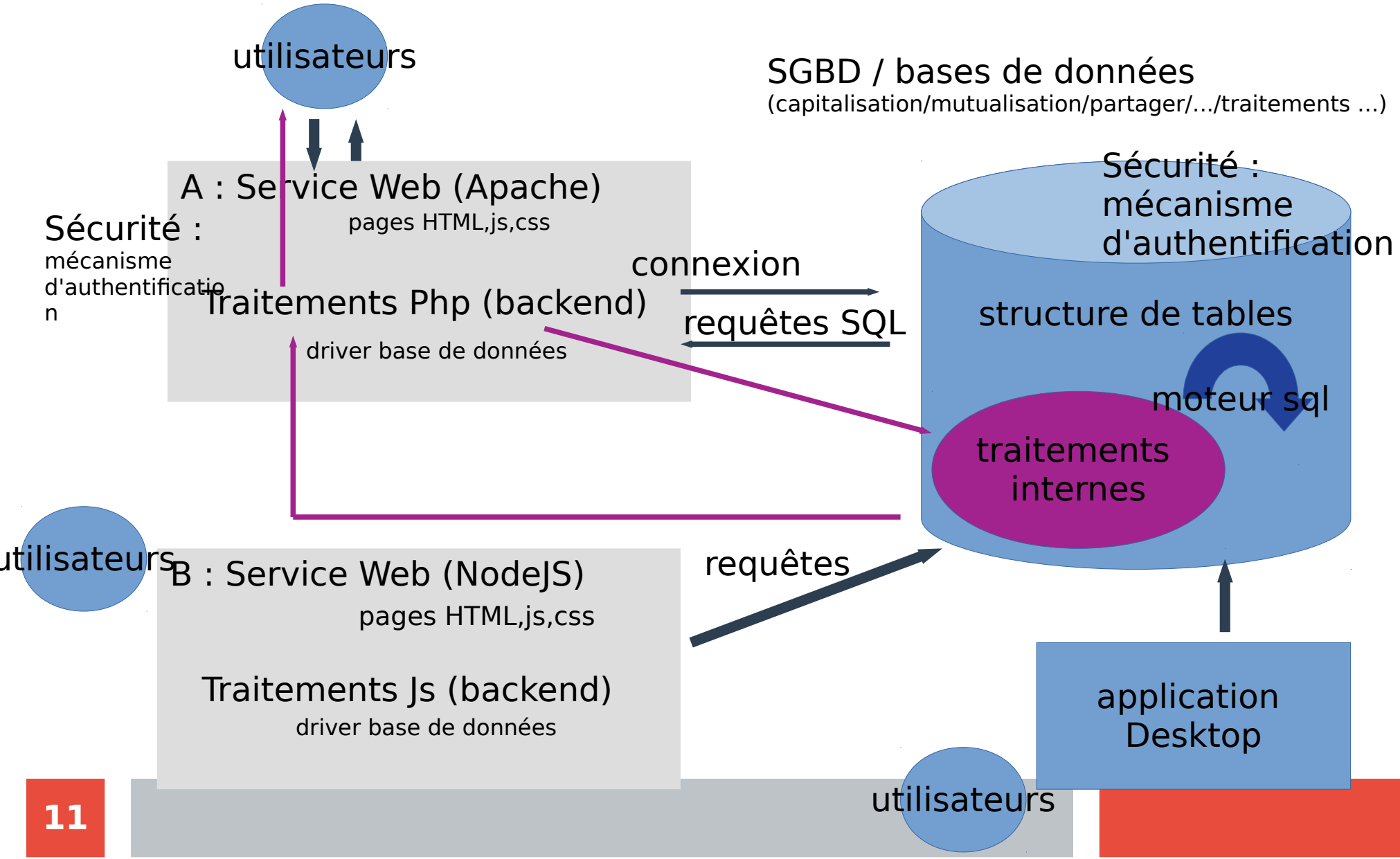
- Relationnelle : tables == relations .... langage d'interrogation (SQL)
- NoSQL : structure de type document/ key-value /graphe -- langage spécifique

Procédures stockées, trigger ....langages spécifiques

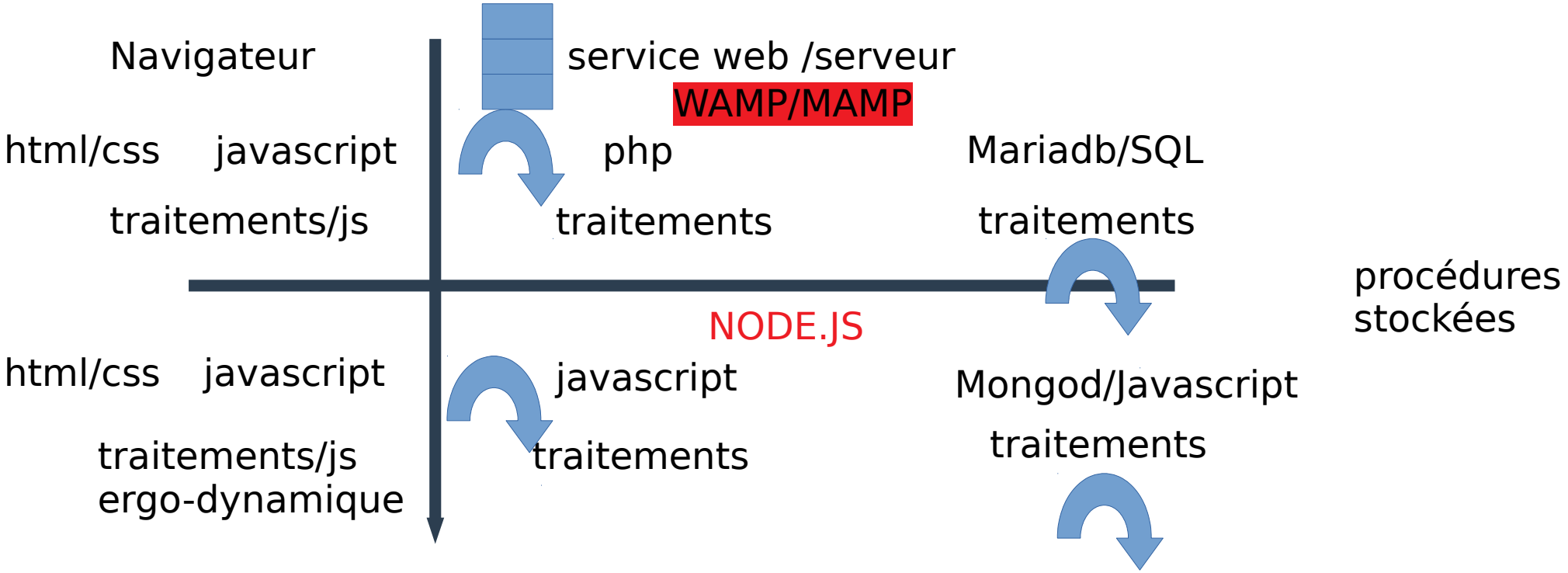
cohérence des langages

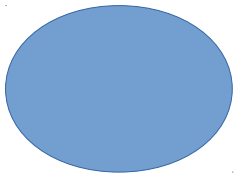


Localisation des traitements/notion multi applications/multi utilisateurs



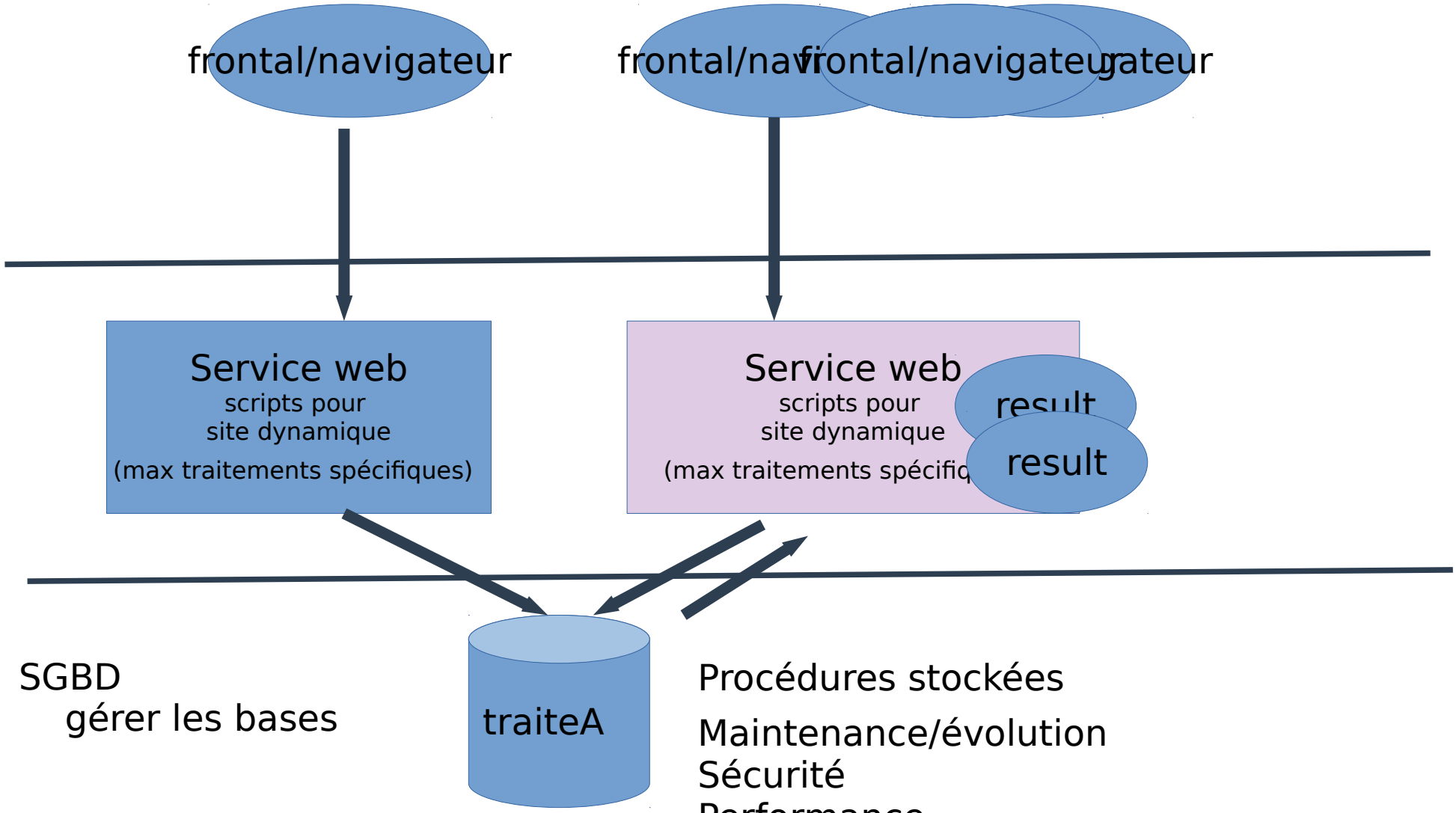
Positionnement des traitements dans le processus global de notre site web



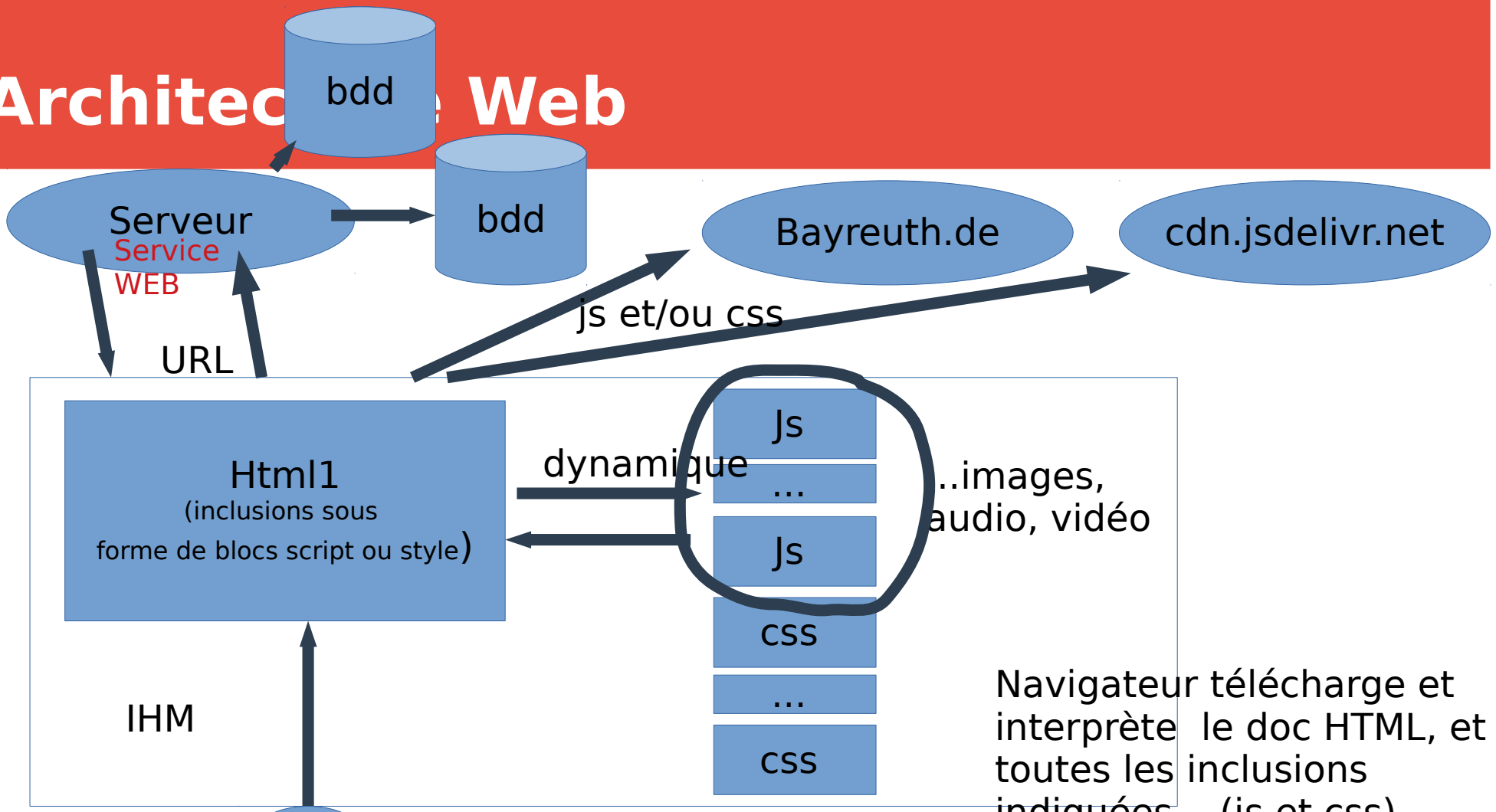


!!!!! attention à l'ouverture d'une ressource pensez à la terminer. Parce que les processus fonctionnent en permanence.

Architecture multi applicatives



Architec e Web



Navigateur télécharge et interprète le doc HTML, et toutes les inclusions indiquées ...(js et css)

event est gèré ?
action/event ==> réaction (js)

IHM:interface homme machine

Archecture 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 ?

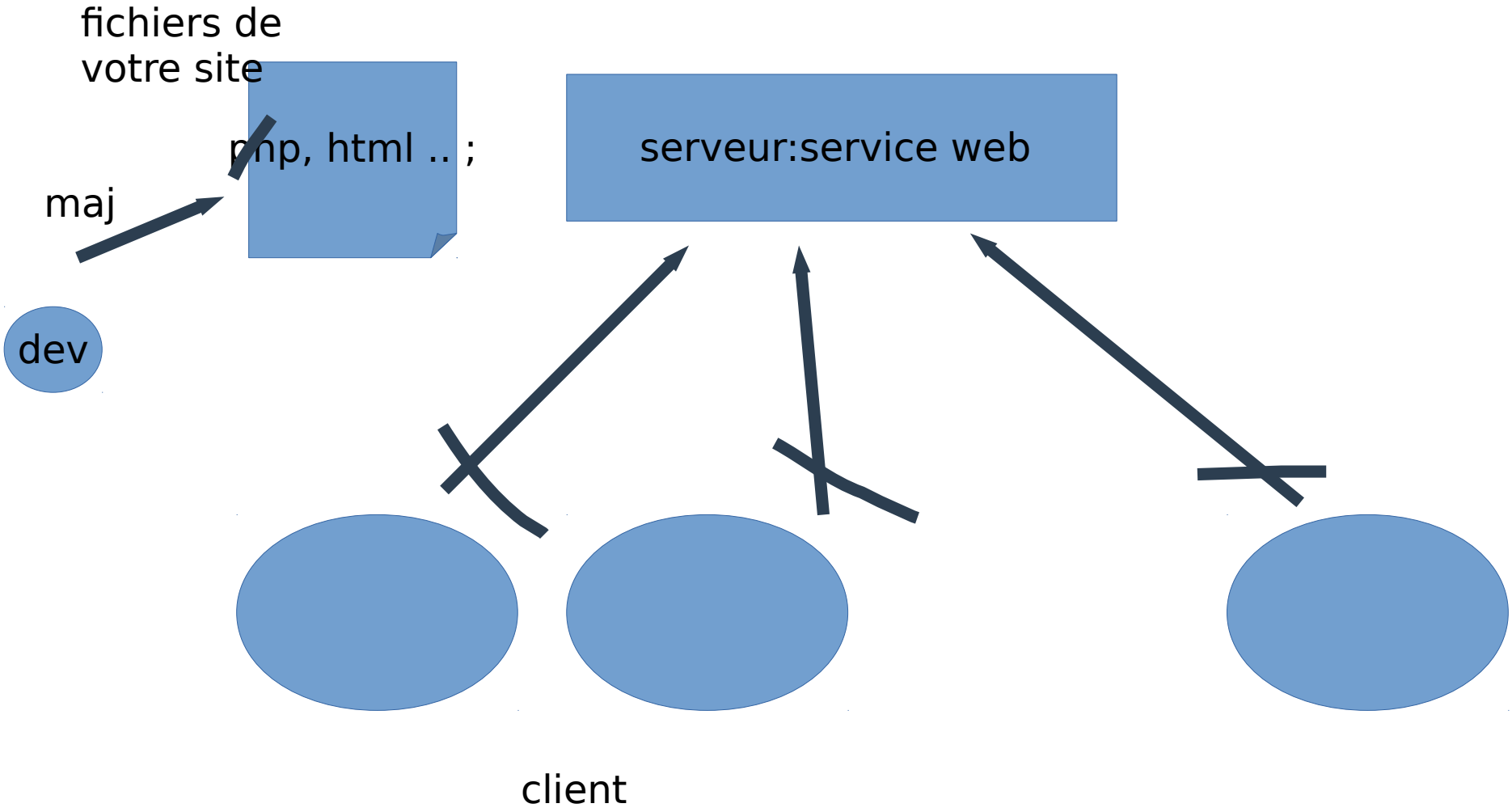
pas la maitrise du frontal ...

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 fait 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)

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

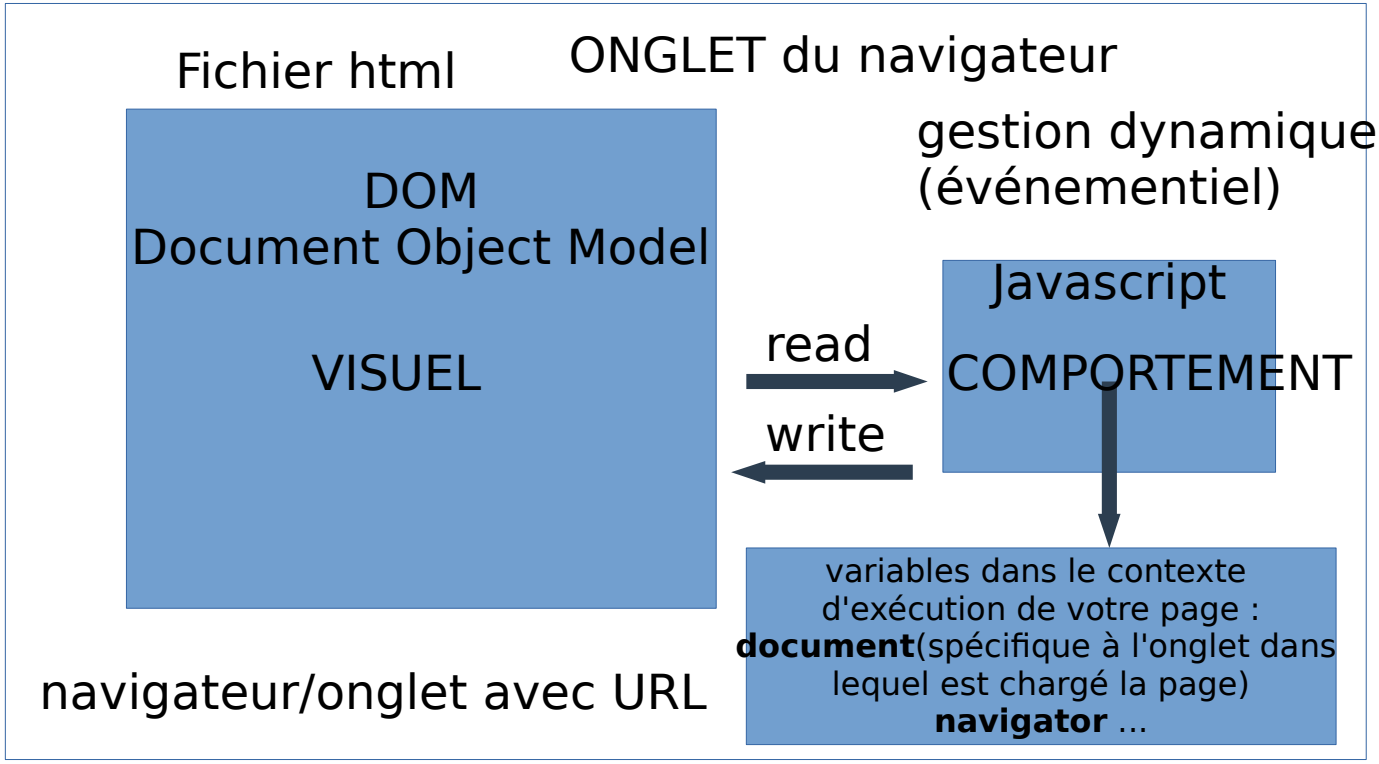
--- Php symfony

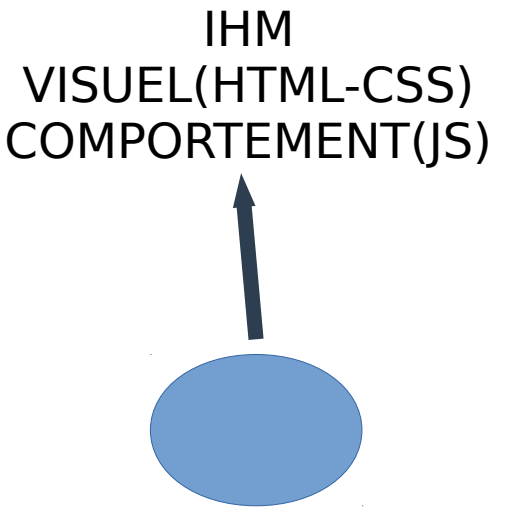
--- Javascript --- angular.js ..... vue.js .... meteor.js

Coût d'appropriation

Rappel sur le fonctionnement

Communication entre la DOM et Javascript





Lecture de la structure à l'aide de la variable document.
Possibilité de récupérer toutes les informations et de modifier la structure.

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.

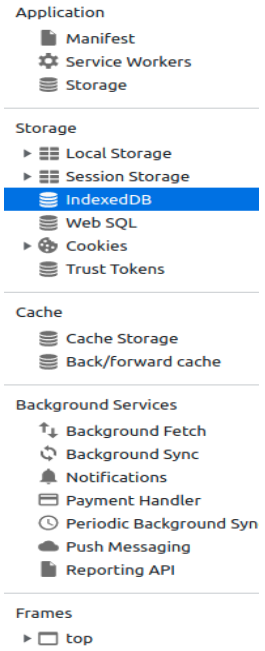
C'est dans ces cas que l'on va chercher la variable **document** (définie par le navigateur) et que l'on explore via les méthodes document.getElementBy....ou document.querySelector

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

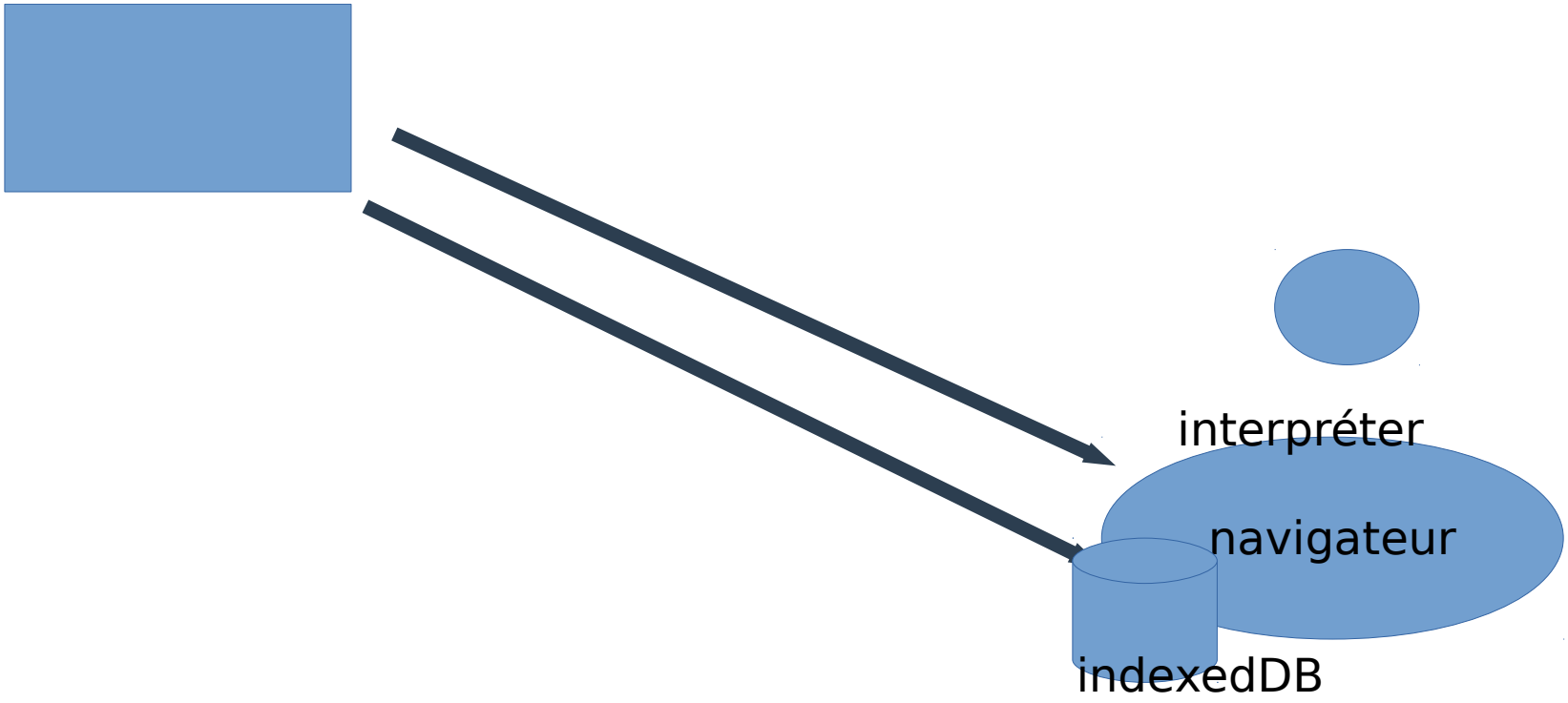
Navigateur

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 — cookieStore
    - Cache local — localStorage
      - key, valeur
    - Base de données : indexedDB — indexedDB
      - key, valeur
- 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



Rappels :

Variables

Types de variable

(Number, String, Array, Date, Object)

Déclaration implicite

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.

Analogie avec les bases de données

Modèle Relationnel :

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)

Modèle NoSQL

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

On définit des structures sans typage

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.err, console.table

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

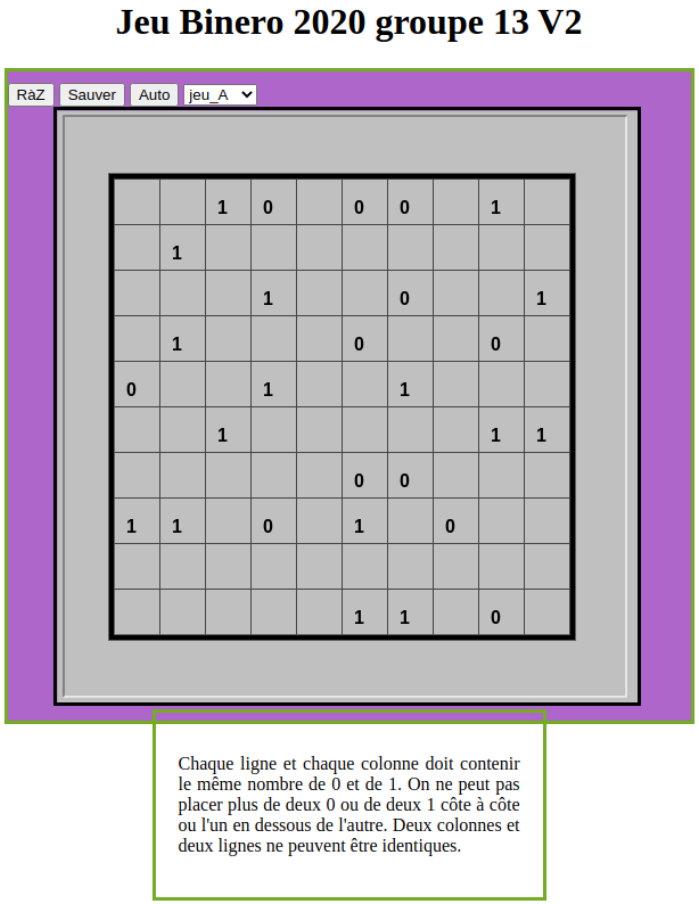
Description sous un format JSON

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,-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."
    },
.....
```



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

Exemple de structure complexe

On peut définir des structures complexes de données au format json (key, value) simplement

dans lesquelles la value peut être :

- types simples (String, Number, Date)
- types complexes (List, Array, Object)
- types fonctions
- Avec une profondeur adaptable au contexte

le tout est de pouvoir naviguer

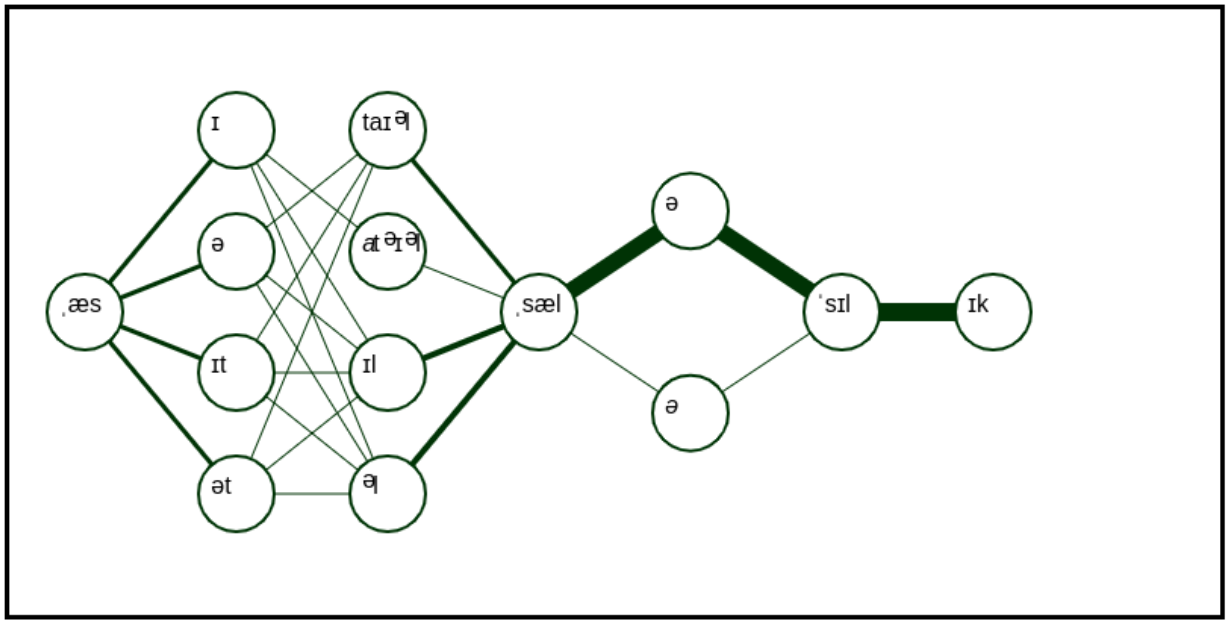
- trouver l'élément que l'on veut, parcourir
- pouvoir le modifier, en rajouter , en supprimer

Ce que l'on souhaite : afficher l'arbre des syllabes

Tracé graphique des syllabes

acetylsalicylic

ˌæs ɪ ɑtˈəɪˈəl ˌsæl ə ˈsɪl ɪk
ˌæs ɪ ɪl ˌsæl ə ˈsɪl ɪk
ˌæs ɪ ˈəl ˌsæl ə ˈsɪl ɪk
ˌæs ə taɪˈəl ˌsæl ə ˈsɪl ɪk
ˌæs ə ɪl ˌsæl ə ˈsɪl ɪk
ˌæs ə ˈəl ˌsæl ə ˈsɪl ɪk
ˌæs ɪt taɪˈəl ˌsæl ə ˈsɪl ɪk
ˌæs ɪt ɪl ˌsæl ə ˈsɪl ɪk
ˌæs ɪt ˈəl ˌsæl ə ˈsɪl ɪk
ˌæs ət taɪˈəl ˌsæl ə ˈsɪl ɪk
ˌæs ət ɪl ˌsæl ə ˈsɪl ɪk
ˌæs ət ˈəl ˌsæl ə ˈsɪl ɪk



100

Plusieurs parties : un sélecteur de mot, la liste des prononciations associés et le graphe.

Comment ça fonctionne ?

Quand l'utilisateur sélectionne un mot l'application lui affiche la liste des prononciations possibles et le graphe. Lorsqu'il se déplace dans la liste des prononciations on lui affiche le chemin dans le graphe correspondant.

Elaborer la solution pour effectuer ce type de graphe

Plusieurs possibilités pour effectuer ce type de tracé :

- soit on utilise un canvas
- soit on utilise le SVG
- https://js.cytoscape.org/
  - https://cytoscape.org/cytoscape.js-tutorial-demo/
  - (demo
- d3.js
- …

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

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":{'fɪʃ: 4},"syllabe4":{ri: 1, ər: 1, jər: 1, i: 1,"i":2},"syllabe5":{i: 1, er: 2,ər:2},"syllabe6":{i: 2}};
const syllabes={"syllabe1":{",æs": 2},"syllabe2":{"ɪ": 2,"ə": 2,"ɪt": 2,"ət": 2,},"syllabe3":{"taɪ^əl": 2,"at^əɪ^əl": 2,"ɪl": 3, "^əl": NaN},"syllabe4":{",sæl": 3},"syllabe5":{"ə": 2,"ə":3},"syllabe6":{"'sɪl": 3},"syllabe7":{"ɪk": 3}};

Liste des prononciations :
const tableau_prononciations1=[",ben ɪ 'fɪʃ ri",",ben ə 'fɪʃ ri",",ben ɪ 'fɪʃ ər i",",ben ɪ 'fɪʃ jər i",",ben ə 'fɪʃ ər i",",ben ə 'fɪʃ jər i", ",ben ɪ 'fɪʃ i ər i",",ben ə 'fɪʃ i ər i",",ben ɪ 'fɪʃ ri",",ben ə 'fɪʃ ri",",ben ɪ 'fɪʃ ər i",",ben ə 'fɪʃ ər i",",ben ɪ 'fɪʃ i er i",",ben ə 'fɪʃ i er i",",ben ɪ 'fɪʃ i er i",",ben ə 'fɪʃ i er i"];
const tableau_prononciations=[
",æs ɪ at^əɪ^əl ˌsæl ə 'sɪl ɪk",",æs ɪ ɪl ˌsæl ə 'sɪl ɪk",",æs ɪ ^əl ˌsæl ə 'sɪl ɪk",",æs ə taɪ^əl ˌsæl ə 'sɪl ɪk",",æs ə ɪl ˌsæl ə 'sɪl ɪk",",æs ə ^əl ˌsæl ə 'sɪl ɪk",",æs ɪt taɪ^əl ˌsæl ə 'sɪl ɪk",",æs ɪt ɪl ˌsæl ə 'sɪl ɪk",",æs ɪt ^əl ˌsæl ə 'sɪl ɪk",",æs ət taɪ^əl ˌsæl ə 'sɪl ɪk",",æs ət ɪl ˌsæl ə 'sɪl ɪk",",æs ət ^əl ˌsæl 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 besoinn 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 créé 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+="<ul>";
        for (let mot of tab_prononciations) {
            chaine+="<li onmouseover='debut_eclairage(this);' onmouseleave='fin_eclairage(this,"+nb+");'>";
            chaine+=mot;
            chaine+="</li>";
        }
        chaine+="</ul>";
        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

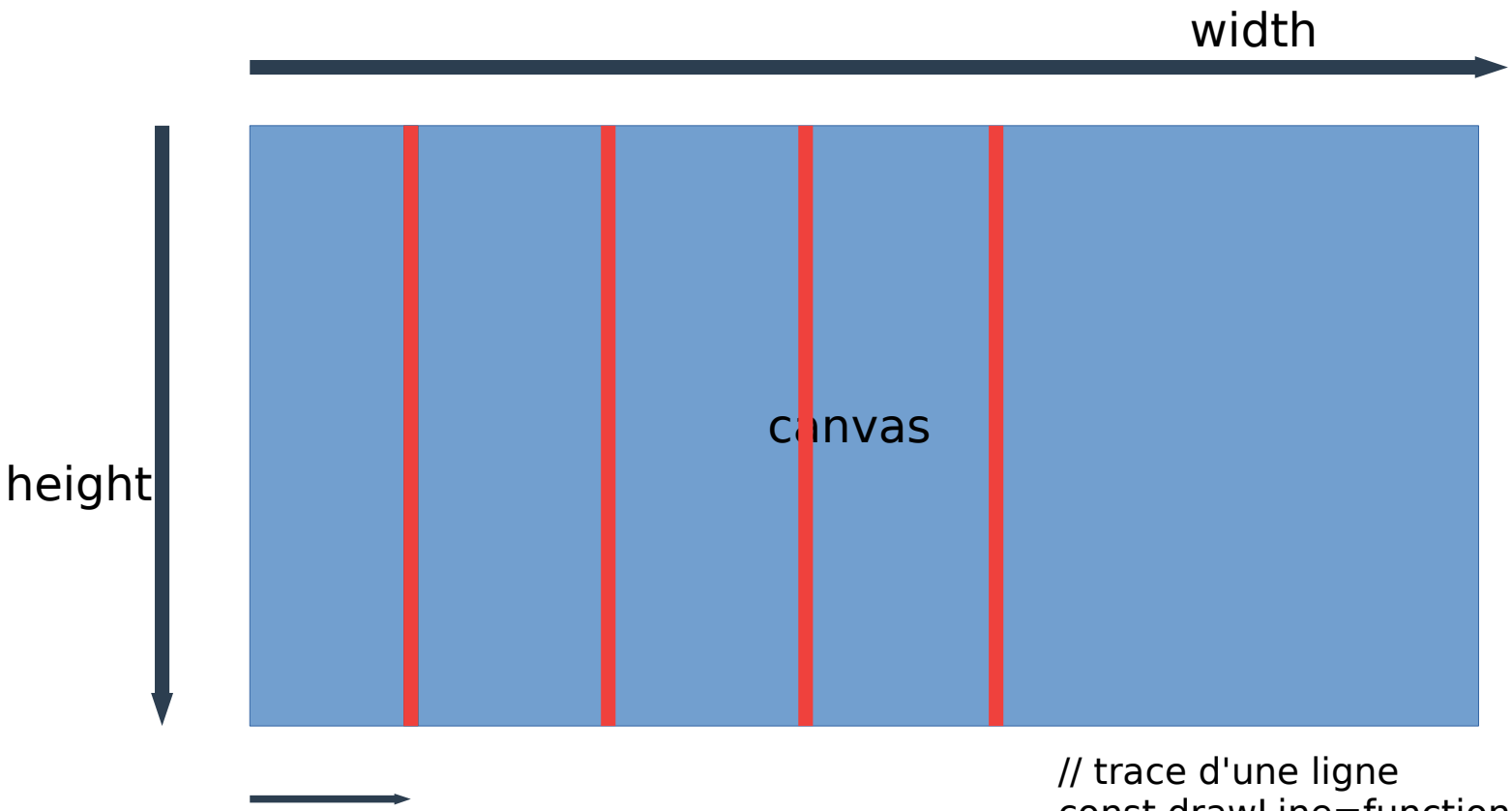
Exemple de l'approche : étape 1

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

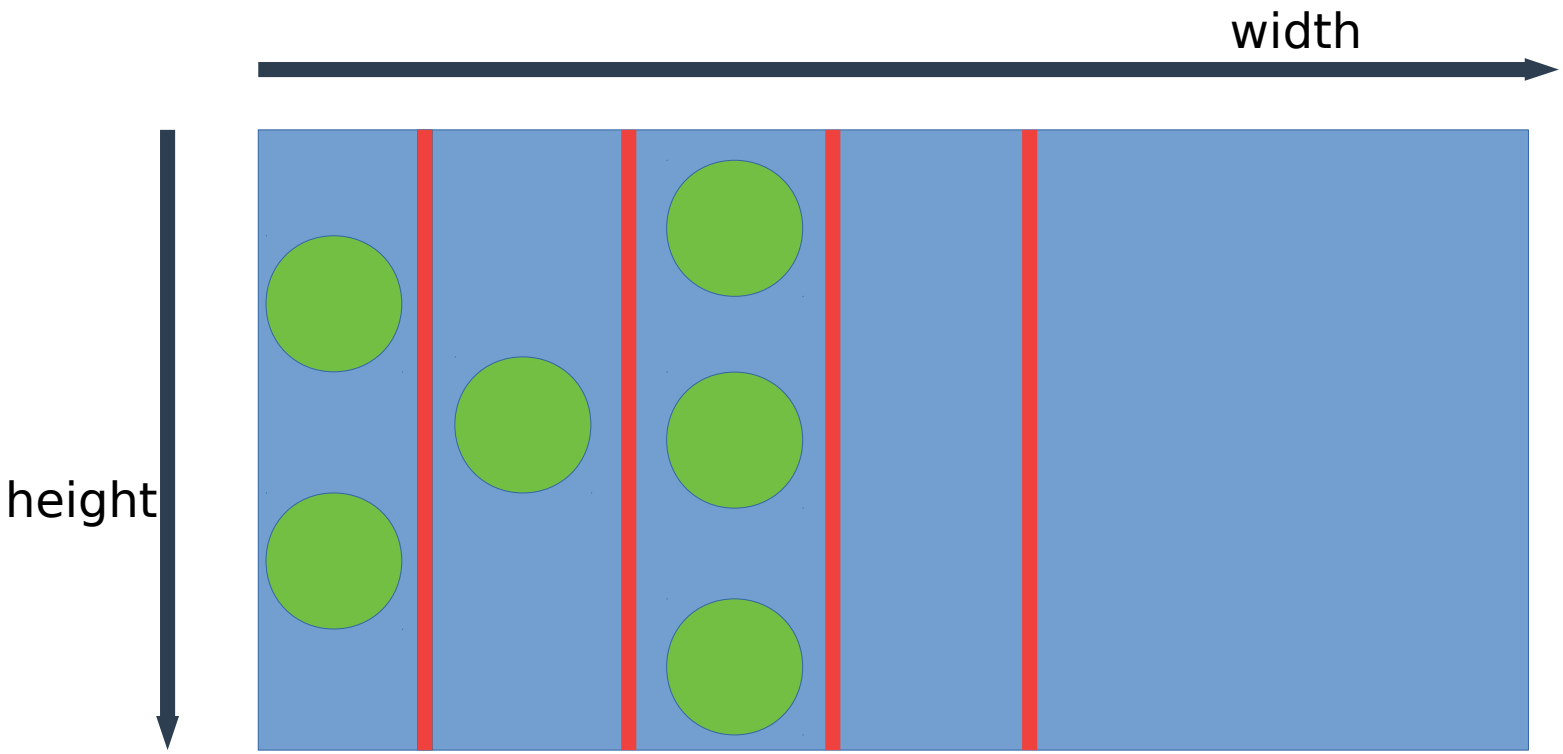
découper la zone par syllabe



Tracé d'une ligne verticale
de position sur x = num_colonne* width/nb_colonnes
(nb_colonnes== 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 tracé)

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 créé 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

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.

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