

PSB 2023 v1.0

Contexte de développement Web

Cours Javascript :

dépôt : [https://github.com/ystroppa/PSB2020\(historique\)](https://github.com/ystroppa/PSB2020(historique))

<https://github.com/ystroppa/PSB2023-M1>

Sujet à valider ensemble

Yvan Stroppa

IR CNRS cursus informatique dans un contexte Linguistique

Yvan.stroppa1@univ-orleans.fr (préciser PSB - M1 - groupe avec un objet)

Projet : 2 à 3 étudiants --- à rendre fin juin 2023 (date à préciser)

- 1 / Rapport (pdf)

Cahier de charges : expression de besoins

Analyse et conception

Explications et illustrations

Phase des Tests (vérif sur tous les navigateurs)

Conclusion

- 2/ Livrable : Programme à livrer (tous les fichiers)

Accompagnement
technique si besoin
par mail, bb, autres

jusqu'à fin juin

Dépôt de doc.... ou envoi par mail

(avec accusé de réception)

Evaluation

Evaluation des rendus : sous un contexte ordinateur portable Linux - navigateur (chrome, firefox ...)

Développeur

Pour résoudre une problématique (échéance, budget)

Intégration - Récupérer du code / ou de produit licence :

- approprier le code (évaluer et vérifier l'intégrité)
- intégrer le code dans votre contexte
- tester

- indiquer les sources

Développement - Produit du code : (documenter)

- algorithme
- traduire dans le langage cible (javascript)
- Mise au point et tester (cas de tests pour valider le code)

Contexte du projet : technique et pédagogique

Le contexte de développement de vos projets va s'inscrire dans un objectif de fabrication de CDS "Cahier Didactique du Savoir".

L'idée est de présenter et d'expliquer un concept (algorithme, théorème ... au autre) dans une SPA (Single Page Applicatif)

Dans une page (html) : public visé : primaire, collège, lycée, supérieur

- Page de présentation : logo "CDS PSB" + concept
- Introduction du concept - historique (voir les infos wikipedia et autres)
- Explications et illustrations du concept (avec des parties **dynamiques, interactives et sonores(mp3, wav ...)**)
 - 1 ou 2 Exemples (interactifs, sonores)
- 1 ou 2 Exercices avec correction (optionnelle)
- Quizz de 5 questions aléatoires sur une liste de 20 à 25
(pseudo évaluation) HTML, CSS, JS (images, graphiques, sonores,)
- Conclusion et liens public visé : primaire, collège, lycée, supérieur ...
Thématiques : Economie, Informatique, Mathématiques, autres(Langage)

CDS : sujets

Parcours d'arbre

1/ algorithme de Kruskal

2/ algorithme de Dijkstra

3/ algorithme de Bellamn-Ford

4/ algorithme de Floyd-Warshall

Sur les automates : expressions régulières et langage

5/ algorithme de McNaughton et Yamada

6/ algorithme de Brzowski et Mc cluskey

Vous pouvez définir votre propre sujet --- à valider avant de commencer

Intérêt : composants logiciels (module)

Regardez les bibliothèques js de type impress.js et autres dans le même contexte.

7) Introduction à la théorie du chaos à une dimension

- construction des graphiques
- proposition des fonctions
- graphique
- diagramme à une dimension

8) Dilemme du prisonnier : tournoi d'Axelrod

https://axelrod.readthedocs.io/en/fix-documentation/reference/overview_of_strategies.html

9) Dilemme du prisonnier spatial

10) Théorème de Pythagore (démonstrations graphiques épaulées par l'algèbre si nécessaire)

11) Diagramme d'Edgeworth animé

12) Pi par la démonstration des camemberts d'Archimède

13) Calcul d'aires et d'intégrales (calculs par les aires des rectangles par défaut et par excès)

14) Percolation

15) Planche de Galton (possibilité d'ajouter des obstacles et des frontières)

16) Systèmes de vote

17) Evaluation d'une fonction d'utilité en incertain (j'expliquerai)

18) Application du théorème de Thales (calcul de hauteurs inaccessibles)

19) Diagramme des champs d'un système d'équations différentielle à 2 dimensions $dx/dt = f(x(t), y(t))$ $dy/dt = g(x(t), y(t))$

20) Fonctions 3D et projections de lignes de niveau sur les trois plans

21) Affichage d'une série chronologique et données statistiques associées

22) Animation de la convergence d'une série de Taylor

<http://www.physics.miami.edu/~nearing/mathmethods/power-animations.html>

23) Théorème d'Euler-Descartes

http://serge.mehl.free.fr/anx/Th_Euler-Descartes.html

24) Modéliser le jeu de HEX

25) Les triangles de Sperner

Mise en oeuvre d'un prototype

proposition d'analyse et de réalisation :

réaliser une présentation sous Powerpoint (pour un prototypage rapide)

page/page

description du contenu de chaque page

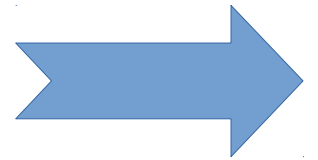
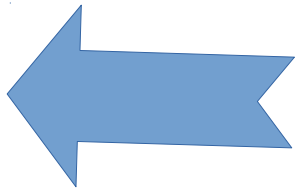
- on réalise son squelette de l'application**
- réfléchir sur les interactions que l'on veut ajouter et les définir**

Idée est de se mettre d'accord sur le contenu et sur ses interactions (échange avec l'utilisateur)

On peut commencer la réalisation dans l'environnement cible et de voir quelles sont les librairies nécessaires

Exemple : CDS Pythagore

Présentation du concept



Bibliothèques JS

impress.js
reveal.js

<https://digital-cover.fr/20-frameworks-et-libs-javascript-a-connaître-en-2022/>

highcharts.js
charts.js
d3.js
jsxgraph.js

datatable.js **bibliothèque 3D Three.js**

Langages

éditeur - IDE (notepad++, Sublime, Atom, VCS ... Eclipse, Netbeans, IntelliJ, Visual studio)

C
C++
Fortran
Java
C#

compilés (on passe par un compilateur qui va traduire les instructions en code machine)

Application Standalone



code machine - Binaire

Python
Perl
Shell
ruby
...

interprétés : on utilise un interpréteur

Javascript

langages de description
HTML : langage tagué
CSS :

JSON : description de données
XML : Extended Markup Language

Pages Web
navigateur

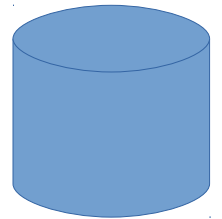
Persistance

Langage SQL

Bases de données
Mysql
MariaDB
Postgresql

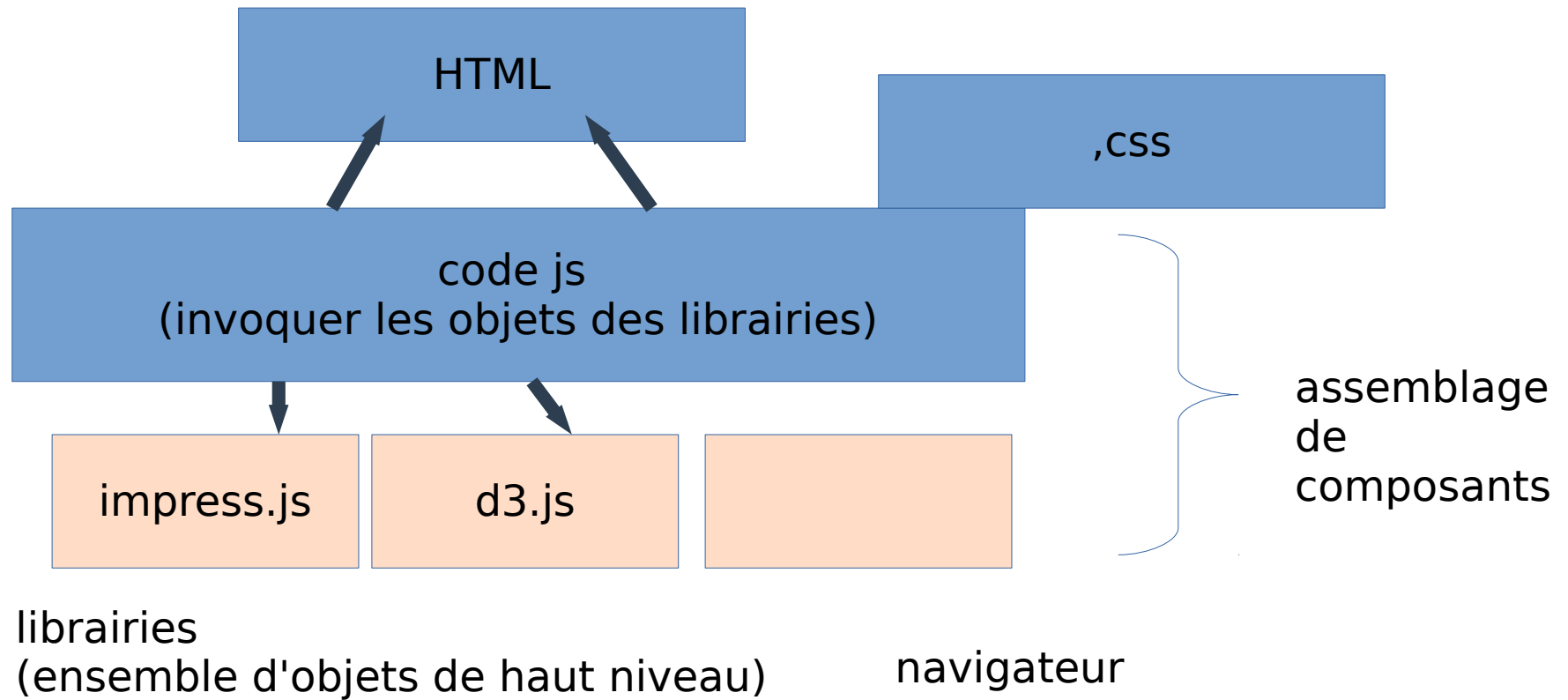
Oracle
DB2
Sql server

// NoSQL
Mongodb
Couchbase

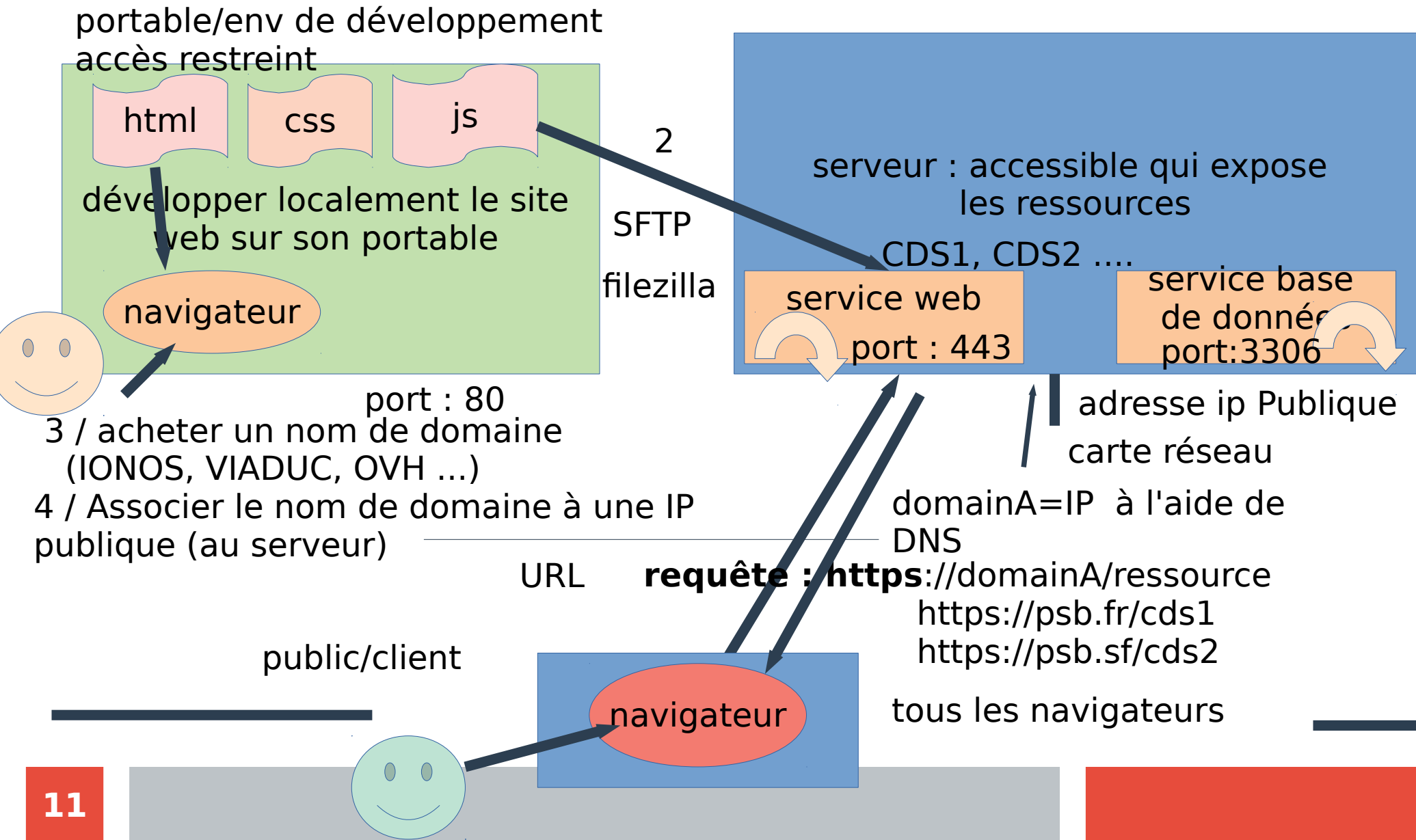




utilisateurs

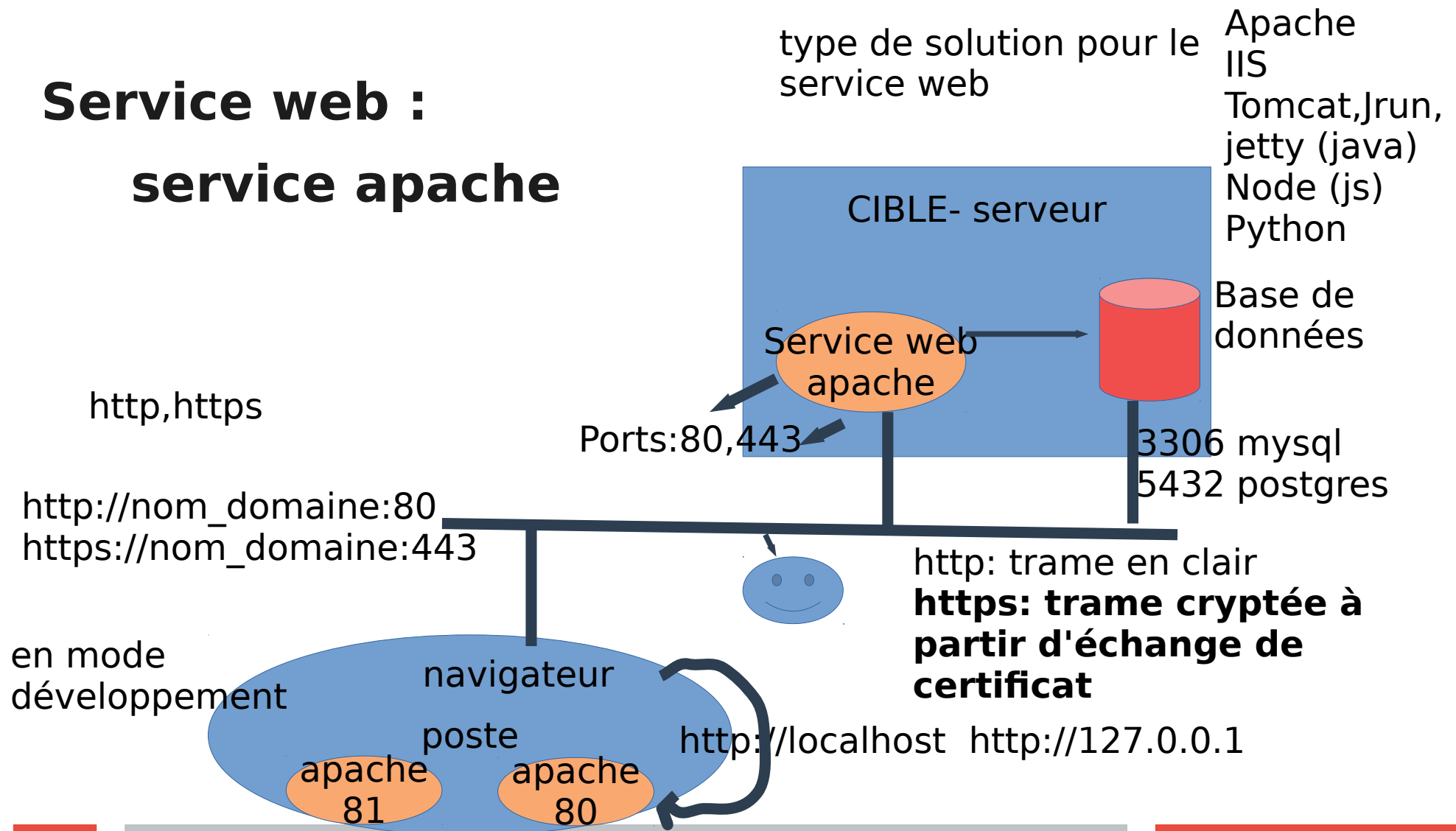


Principe de base de dév d'un site web



Services web / Architecture

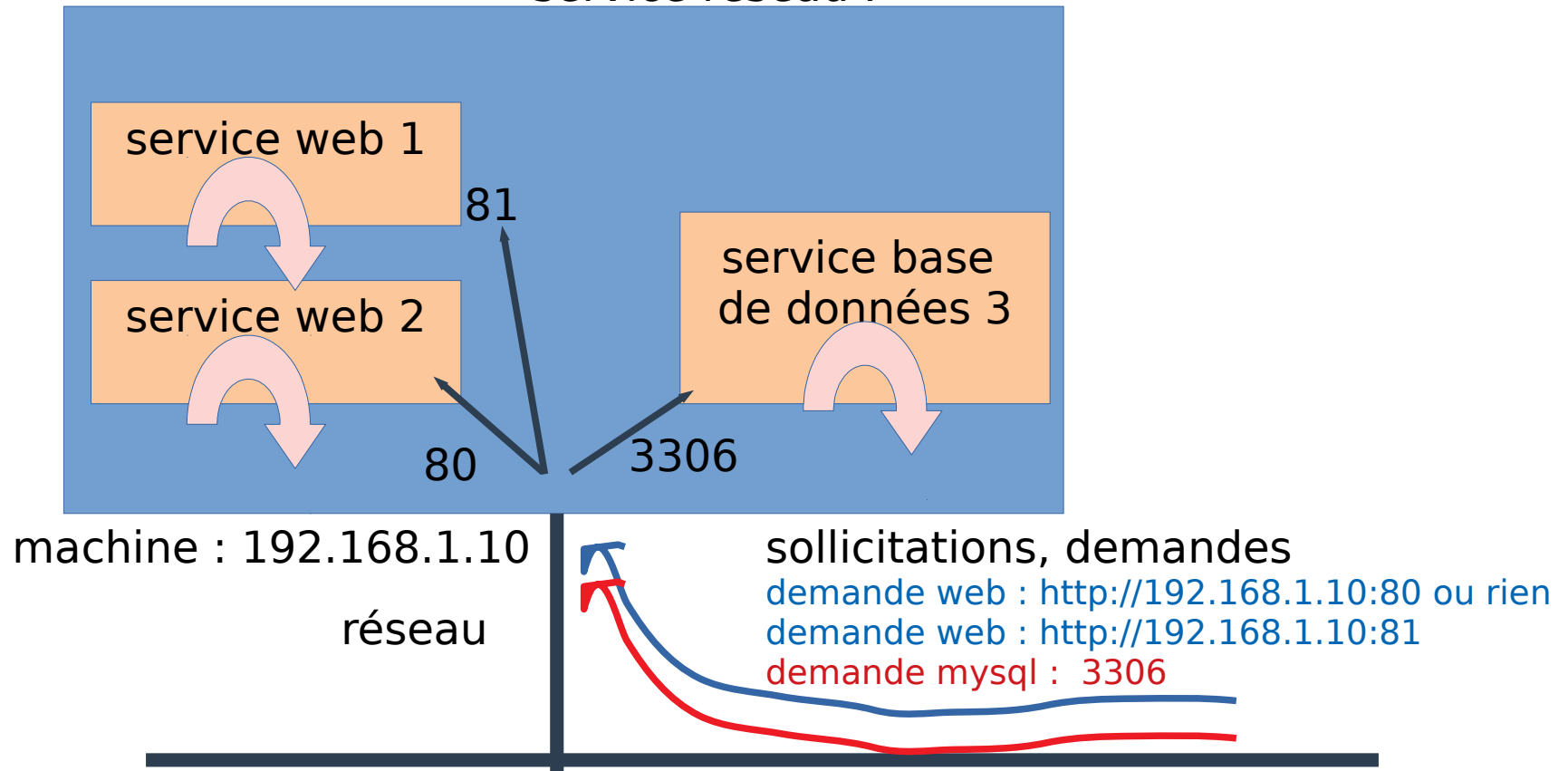
Service web : service apache



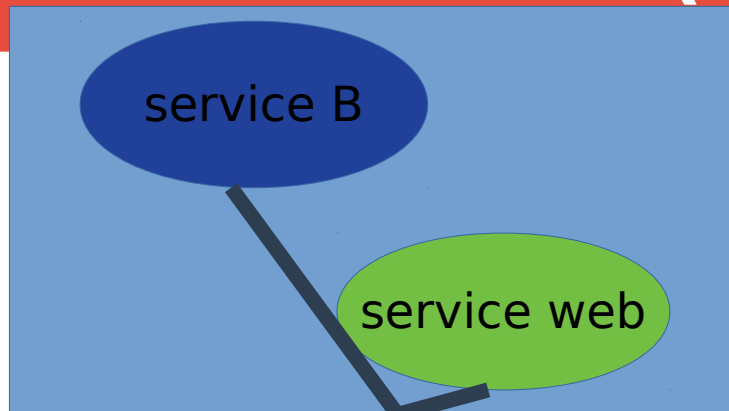
Numéro port pour les services

service = application qui fonctionne tout le temps

service réseau :



Architecture (suite)



serveur/machine distante chez un hébergeur

Service web :(http ou https)

service réseau : écoute des sollicitations sur le réseau à partir des pots 80 ou 443

contenir un ensemble d'éléments

développeur ==>(fichiers --- html --- css ---js -- autres)

network

trame/
demande

psbedu.org

portable

Env de développement
console

1/ requête : url

Navigateur qui interprète : important

- pas la maîtrise de son évolution
- plusieurs versions de navigateur (Chrome, Firefox, Safari, Opera, Edge)
- pas de maîtrise de la mise à jour (nouvelle version diffusée par l'éditeur) peut nécessiter des adaptations



2/ retour du service web

index.html + css (visuel)

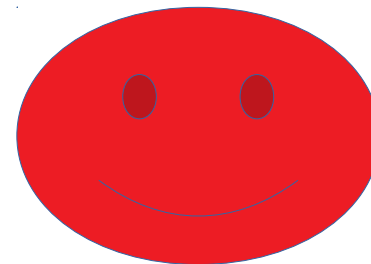
+ js (dynamique /comportement)

+png + audio + vidéo

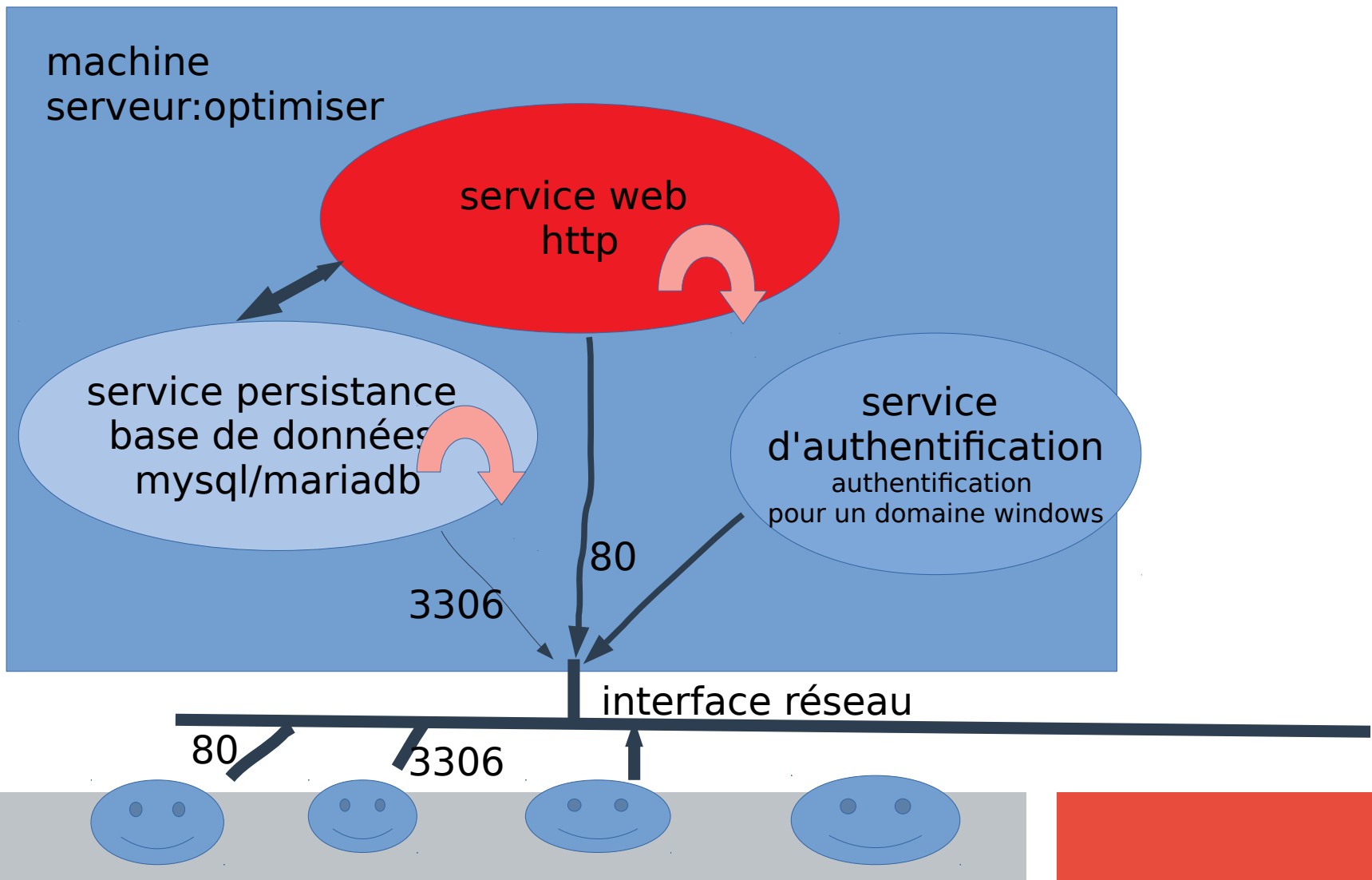


protocole

Dans le cadre d'une
communication entre
deux parties :
Protocole d'échange et
communication
http/https



services



Application WEB : hébergée sur une machine (serveur ou en dev sur votre portable) accessible par la protocole http

Elle est composée :

HTML (documents ou fichiers html) inclusions : script et css
structure de vos pages web

CSS (fichiers de style)

JS (fichiers de scripting en javascript)

la gestion des interactions et événementiel (dynamique sur le poste client (dans le navigateur))

autres ingrédients :

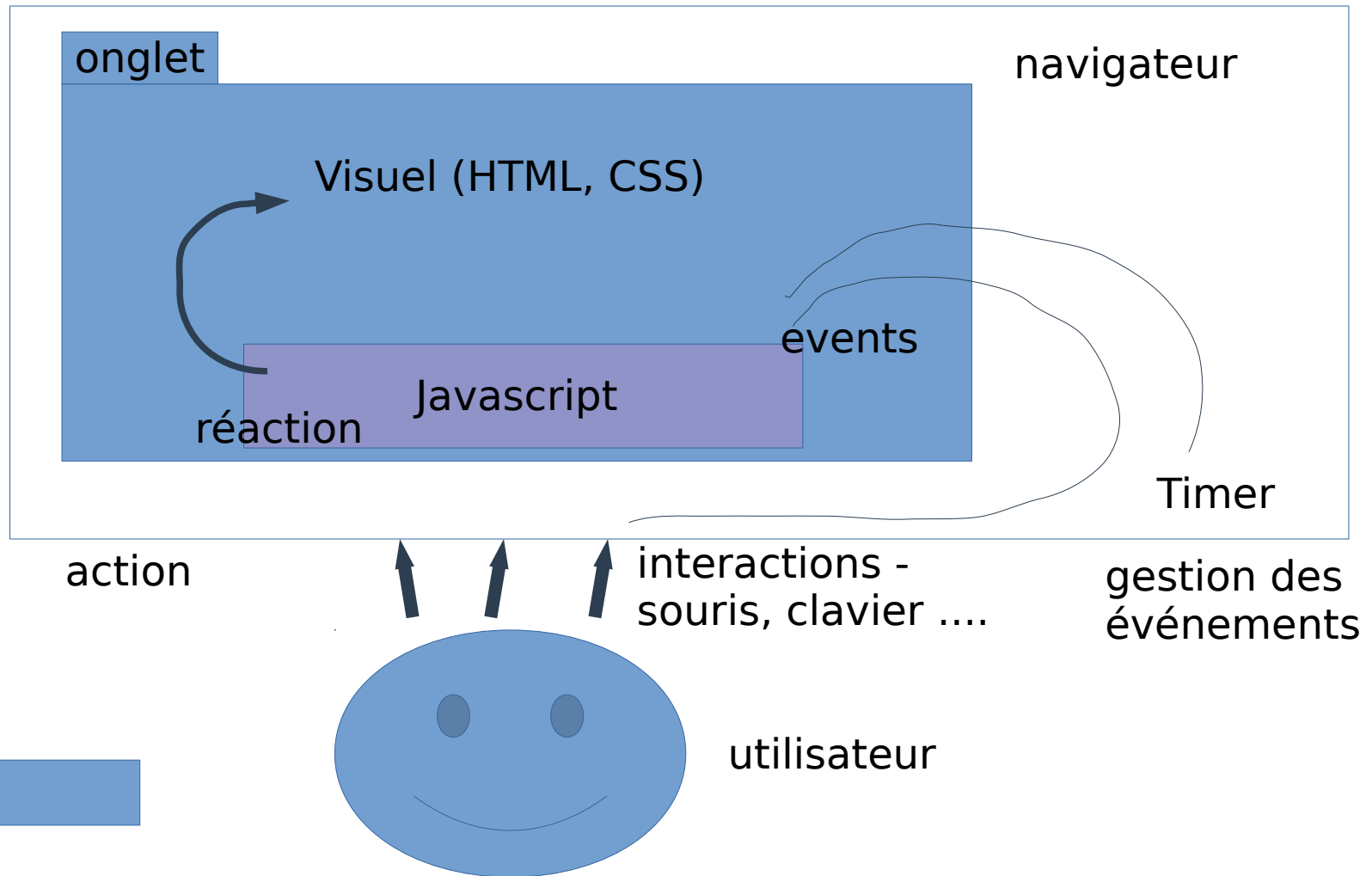
fichiers images

fichiers audios

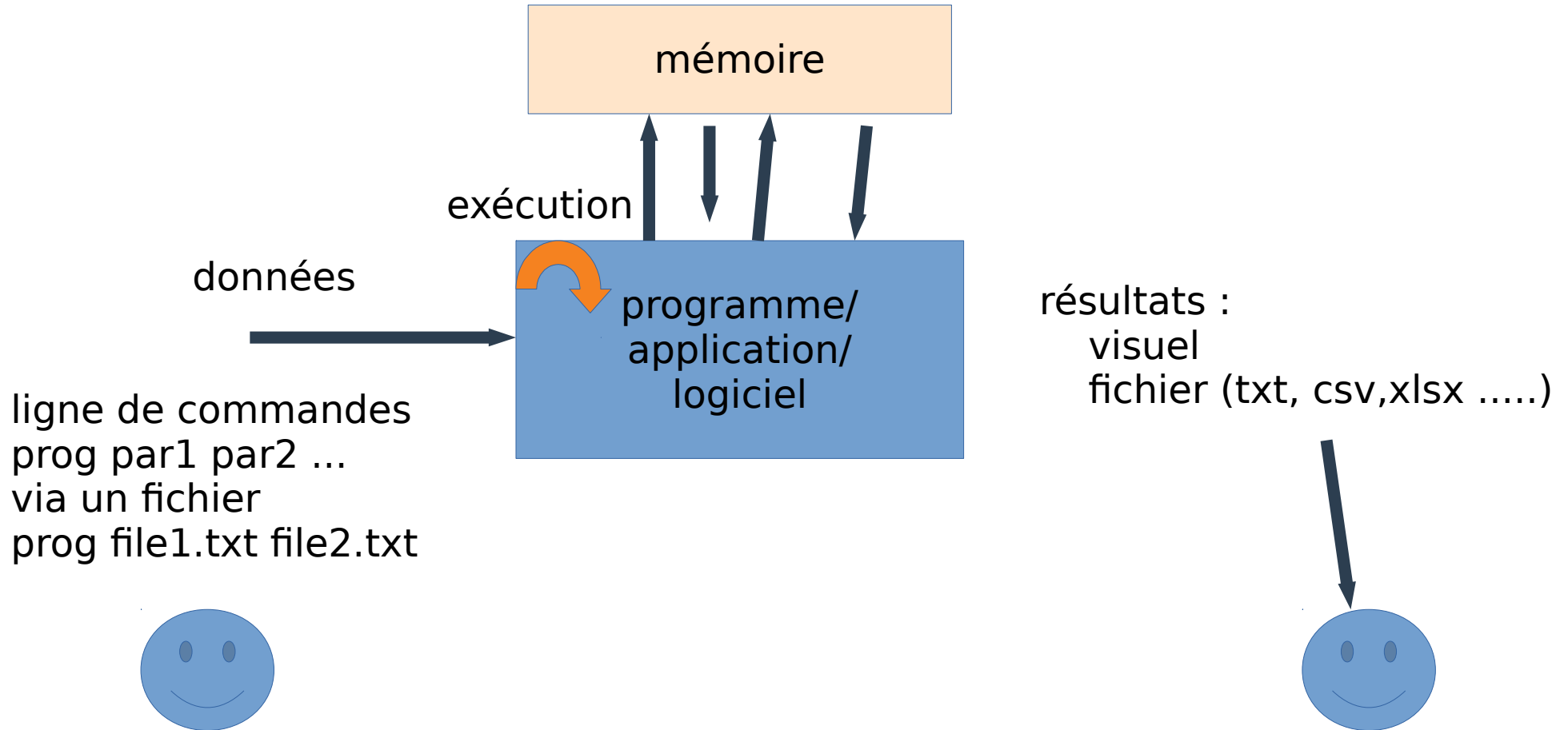
fichiers vidéos

navigateur qui interprète :
- les tags HTML
- Les styles
- les scripts JS

Application - web



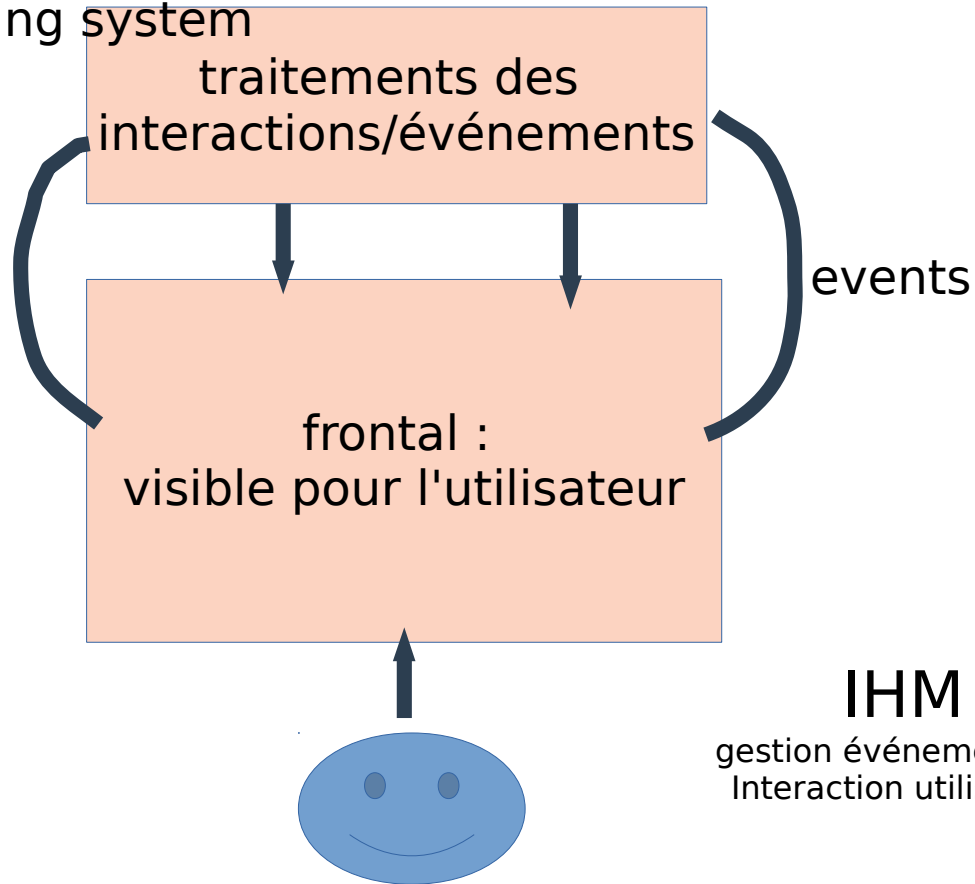
Passage de données : sans IHM (interface homme-machine)



IHM : interface homme/machine

un seul langage Java ou Python
application standalone

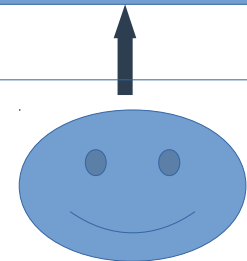
operating system



navigateur/onglet

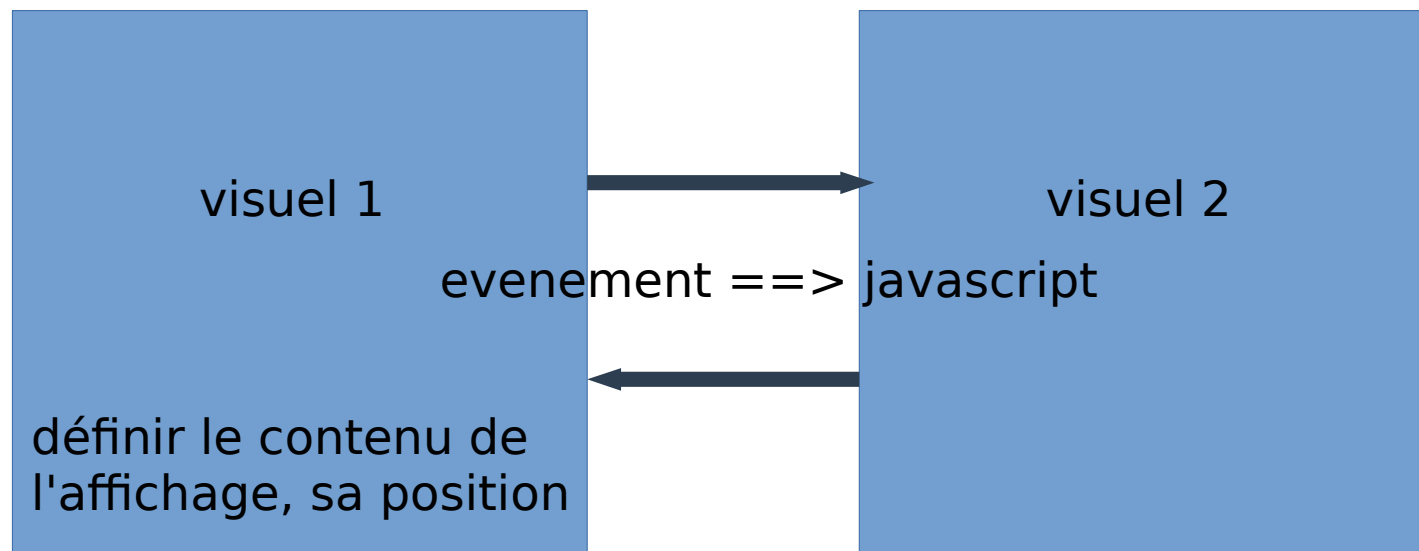
traitements des
interactions/événements
Javascript

frontal :
visible pour l'utilisateur
HTML/CSS



Pour la construction de votre projet :IHM

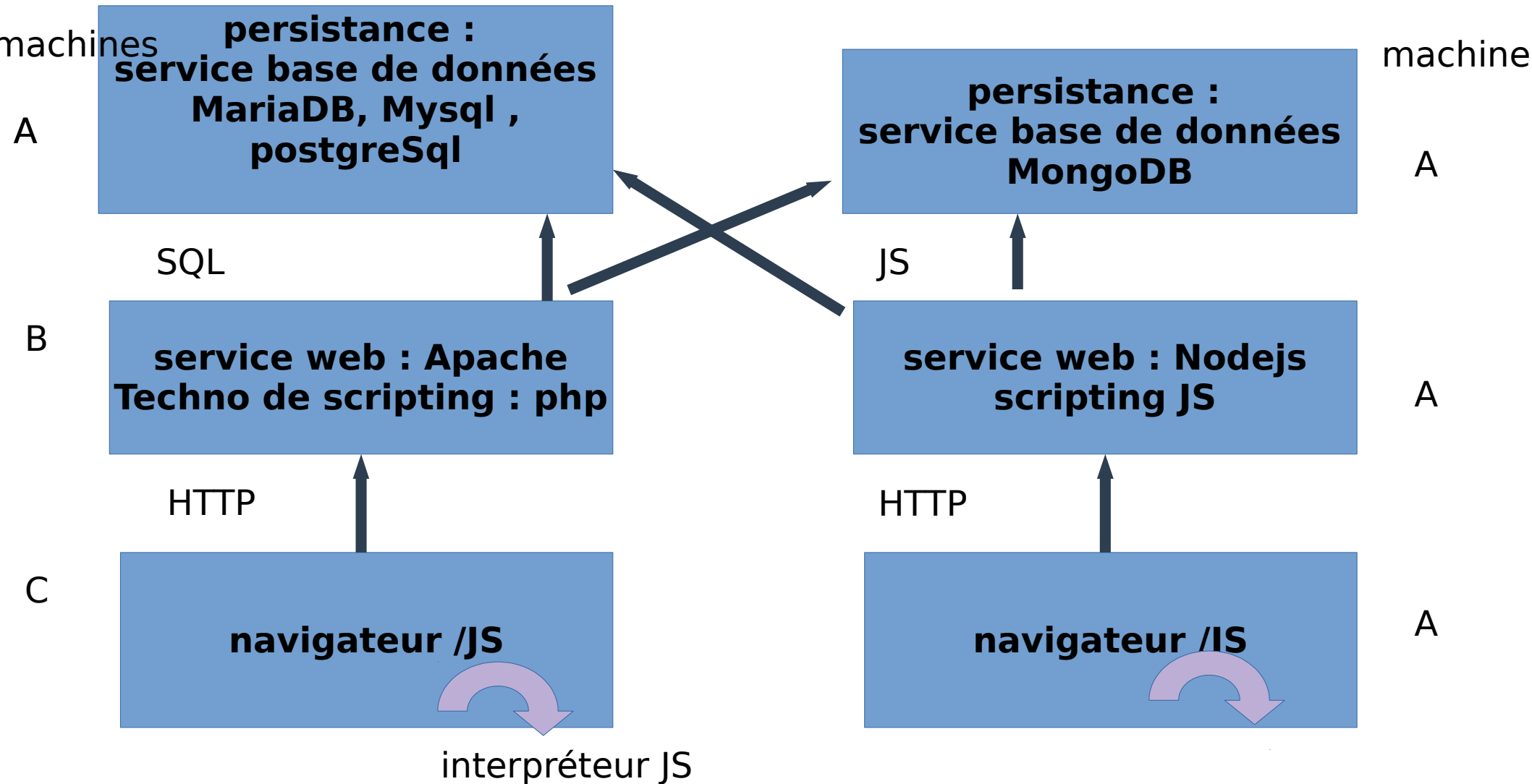
Peut se construire par un prototype sous Powerpoint : c'est une description et on se met d'accord



interaction - échange avec l'utilisateur

objet graphique -- événements -- réaction

Utilisation de Javascript fullstack



À installer sur vos machines

Nodejs (sur tous les systèmes)

interpréteur de javascript

Git (dispo sur toutes les machines)

Installer plusieurs navigateurs pour des tests

- chrome, safari, firefox
- chrome, edge, firefox

Langage informatique

A quoi sert un langage de programmation ?

Programmer des traitements en utilisant des séquences d'instructions du langage.

Que peut-on trouver dans un langage et de quoi a t-on besoin ?

Variables (de différentes natures)

jeu d'instructions

- Blocs d'instructions
- Modulaires

syntaxe à respecter
mots clés (spécifique au langage)

Environnement pour faire du javascript

Pour mise au point et interprétation

Navigateur : console

nodejs : interpréteur de commandes

<https://www.programiz.com/javascript/online-compiler/>

jsfiddle

codeopen

....

Fichier js

Algorithmie

Permet de décrire les étapes de son traitement :
utiliser les commentaires pour décrire les étapes
// première étape : récupérer les valeurs saisies
// deuxième étape : vérifier leur contenu
//

données en
entrée

décrire le contenu
de cette boîte

sorties

Langage informatique : Variable

Notion de variable :

Permet de stocker une valeur au cours de l'exécution du programme dans le but de pouvoir la reprendre, la modifier ou la détruire. La variable n'existe que lorsque le programme s'exécute.

Utilisation d'une variable :

Allocation : déclaration explicite ou implicite

Utilisation : `variable1="voiture"`

Désallocation : nécessaire dans certains langages de type c,c++, en Java il y a un mécanisme de libération des variables mémoires (Garbage Collector), en Javascript notion de delete, elles sont généralement déclarée en local.

affectation des types/classes

Dans Javascript tout est objet

un ensemble d'attributs et de méthodes(fonctions)

Le système a défini des objets de bases

String, Number, Date, Array, Object

String

attributs
length

bibliothèque
fonctionnelle

méthodes
indexOf
substr

Chaque objet possède des propriétés et des méthodes associées

Lors de l'affectation de vos variables, il va associer la variable à un objet déjà défini.

var psb1="voiture" ; ==> associe la variable psb1 à la classe/type String, du coup la variable psb1 peut utiliser toutes les méthodes définies dans le type String

psb1.substr(0,2) ; prend les deux premiers caractères de la variable

psb1.indexOf() ; chercher l'occurrence d'un caractère dans la contenu de la variable

var num1=12.12 ; ==> associé la variable num1 à la classe/type Number du coup la variable num1 peut utiliser toutes les méthodes définies dans le type Number

num1.toPrecision(2)

Number

attributs
méthodes
toPrecision
toString

Pour la date il faut utiliser new

new : opérateur qui crée l'objet

var d1= new Date() ;

Type Date
Attributs

Méthodes
de manipulation,
conversion de date

Nuance entre les déclarations à l'aide de var ou de let

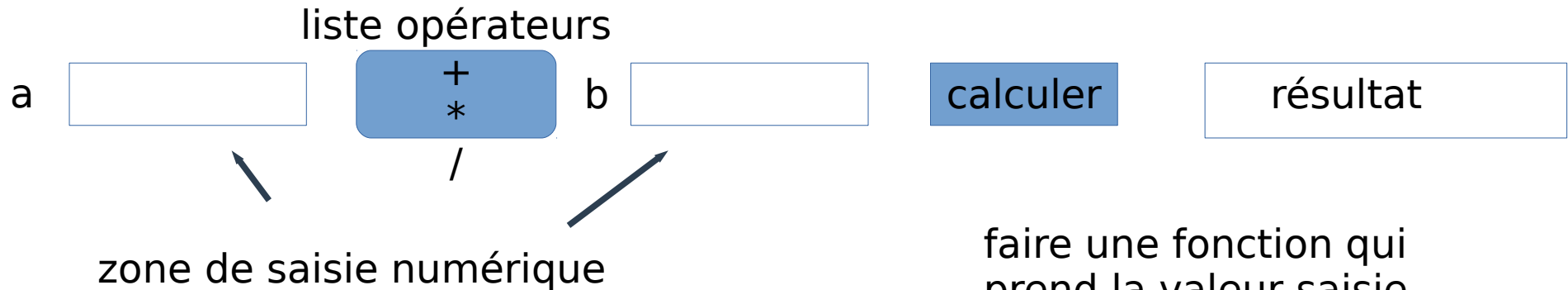
```
var topic="stroppa" ;  
if (topic) {  
    var topic="yvan" ;  
    console.log(topic) ;    yvan  
}  
console.log(topic) ;        yvan
```

```
var topic="stroppa" ;  
if (topic) {  
    let topic="yvan" ;  
    console.log(topic) ;    yvan  
}  
console.log(topic) ;        stroppa
```

scope de la variable

Exercice pour groupe 36 dans codeopen ou jsfiddle

calcul d'une expression



faire une fonction qui
prend la valeur saisie
dans la zone a et la
zone b et qui effectue le
calcul $a+b$

1 partie : juste faire la somme

2 partie : possibilité de sélectionner
l'opérateur

recupérer un descripteur sur un
objet de la page se fait à l'aide de
`document.getElementById("")`

Equation du second degré grp26

-- présentation du calcul : equa du 2nd degré $y=a*x^2 + b*x + c$

-- a<-lire('a') , b<-lire('b'), c<-lire('c')

-- discriminant $\leq b^2 - 4*a*c$

-- si discriminant=0 alors

--faire le calcul solution unique

-- $x \leq -b/(2*a)$

fin si

-- si discriminant<0 alors

--faire le calcul de deux solutions imaginaires

fin si

-- si discriminant>0 alors

--faire le calcul de deux solutions réelles

fin si

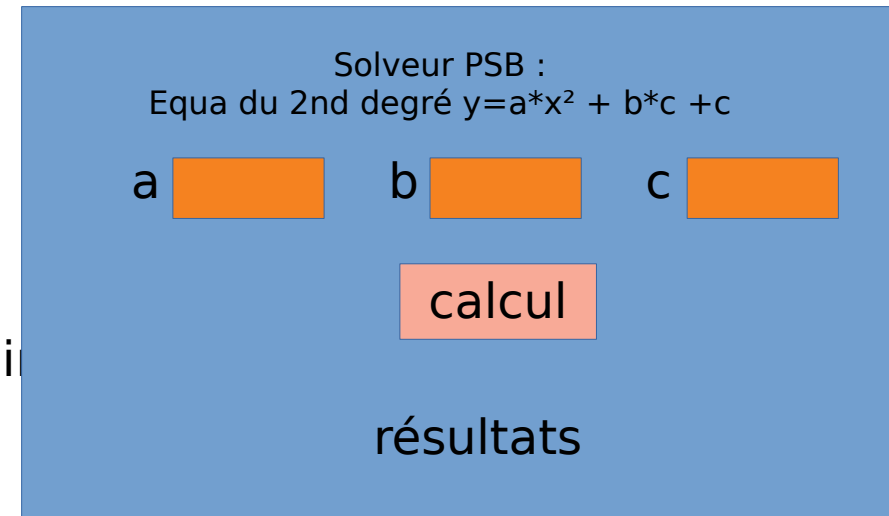
-- affiche le résultat

function solve() {

-- calcul le discriminant

-- en fonction de sa valeur elle fournisse le ou les résultats

}



Blocs d'instructions

Bloc d'instruction conditionnel

```
if <condition> {  
  
} else {  
  
}
```

Bloc d'instruction : fonction

```
function f1_() {  
  
}  
f1_()
```

Bloc d'instruction itératif

```
for (let i=0;i<val;i++) {  
  -- bloc répété  
  
  --variable i est incrémentée de +1  
}
```

navigateur

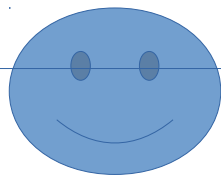
fichier html

visuel

tags html
(part de description visuel,
style, identification, comportement)

fichier ou plusieurs fichiers js

traitement de l'événement



Attention au changement de typage lors de nouvelle affectation

déclaration de type
implicite

var etudiant="stroppa" ; ==> String

etudiant.indexOf("ld"); ==> -1 (rien trouvé)

etudiant.indexOf("ro"); ==> 2

etudiant=54 ; ==> Number

etudiant.indexOf("ld"); ==> function not found

Classique des environnements sans typage explicite
Javascript, Python, Matlab, Mathematica, R

Création d'un nouvel objet

A éviter

```
var etud_nom="stroppa" ;  
var etud_prenom="yvan" ;  
var etud_adresse="rue de la " ;  
  
// problème si on doit créer un autre  
étudiant  
var etud1_nom="ly" ;  
var etud1_prenom="a-phat" ;  
var etud1_adresse="rue de la " ;  
  
.....
```

Problème : on va multiplier les variables, et les variables ne sont pas reliées

object= {key:value, ...} key:unique



```
var etudiant = {  
  "nom" : "stroppa",  
  "prenom" : "yvan",  
  "adresse" : "rue de la .. ;" } ;
```

```
var etudiant1 = {  
  "nom" : "ly",  
  "prenom" : "a-phat",  
  "adresse" : "rue de la .. ;" } ;
```

```
console.log(etudiant.nom  
+etudiant.prenom)
```

value : scalaire, liste (array),
object

structuration des variables complexes

```
etudiant={nom : "stroppa", prenom : "yvan"....}
```

etudiant ➡ nom : stroppa

notes par matières

➡ prenom : yvan

➡ adresse : rue Idlld

➡ Matieres :

etudiant.matieres={} ;
key : nom de la matière
value : les notes

informatique:
[12.3,15.2]

```
{ "Informatique" : [12.3, 15.2] ,  
  "Economie" : [14.5, 12.3] ,  
}
```

Pour ajouter une matière et des notes :
etudiant["Matieres"]["Informatique"]["nouv_mat"]=[] ;
on peut ajouter des notes dans la liste
etudiant["Matieres"]["Informatique"]["nouv_mat"].push(12.3)

Pour extraire les infos :
il suffit de donner le chemin à la donnée
pour avoir la liste des notes de la matière :
etudiant["Matieres"]["Informatique"]["nouv_mat"]
pour avoir la première note
etudiant["Matieres"]["Informatique"]["nouv_mat"][0]

Pour aller plus loin

On vient de définir une structure objet qui nous permet de stocker différentes informations ensemble. Maintenant, si on souhaite ajouter des traitements (fonctions/méthodes) dans cette même structure comment faire ?

Quelques fonctions

fonction etat_civil : on souhaite avoir juste l'affichage du nom associé au prénom

```
etudiant["etat_civil"]=function () { return this.nom + " " + this.prenom;}
```

Pour l'utiliser :

etudiant.etat_civil()

le mot clé this permet d'indiquer une référence sur l'objet lui même.

On souhaite avoir la liste des matières (uniquement les noms des matières) :

Object.keys(etudiant.matiere)

On souhaite la moyenne pour chaque matière :

etudiant.moy_matiere() doit retourner la liste des matières et la moyenne associée

Calcul de la moyenne par matière

**On réalise la fonction à l'extérieur de l'objet :
il faut parcourir l'ensemble des matières de l'étudiant et
faire la somme que l'on divise par le nombre de notes ?**

```
var result={} ;  
Object.keys(etudiant.matieres).forEach(  
  e=>{  
    result[e]=etudiant.matieres[e].reduce((p,ee)=>p+=ee,0)/  
    etudiant.matieres[e].length;  
  }  
);
```


Calcul de la moyenne par matière

A l'intérieur de l'objet

```
etudiant.moy_mat=function() {  
  var result={} ;  
  Object.keys(this.matieres).forEach(e=>{  
    result[e]=this.matieres[e].reduce((p,ee)=>p+=ee,0)/this.matieres[e].length;  
  });  
  return result ;  
}
```

attention aux affectations

quand c'est des types de base (Number, string) une copie

quand c'est des objets : attention javascript passe la référence

Et si on souhaite étendre ce concept et créer son propre modèle d'objet

==> CLASS

Notion de classe

On peut définir la classe étudiant à partir de laquelle on pourra générer des objets ayant les mêmes attributs et méthodes

Comment ?

A l'aide du mot clé : class

Class (suite)

// getter et setter
// pas nécessaire car les attributs sont directement accessibles

```
class etudiant {  
    constructor(Nom, Prenom, DateNaiss) {  
        this.nom=Nom ;  
        this.prenom=Prenom ;  
        this.date_naissance=DateNaiss ;  
        this.matieres={ } ;  
    }  
    // autres méthodes  
    age() {  
        return new Date()-this.date_naissance)/1000/60/60/24/365 ;  
    }  
    calcul_moy_matieres() {  
  
    }  
    ajoute_matiere(mat) {  
        this.matieres[mat]=[] ;  
    }  
    ajoute_note_matiere(note,mat) {  
        this.matieres[mat].push(note) ;  
    }  
}
```

Pour utiliser dans votre code :
var etu1=new etudiant("Stroppa","yvan") ;

Class avec encapsulation

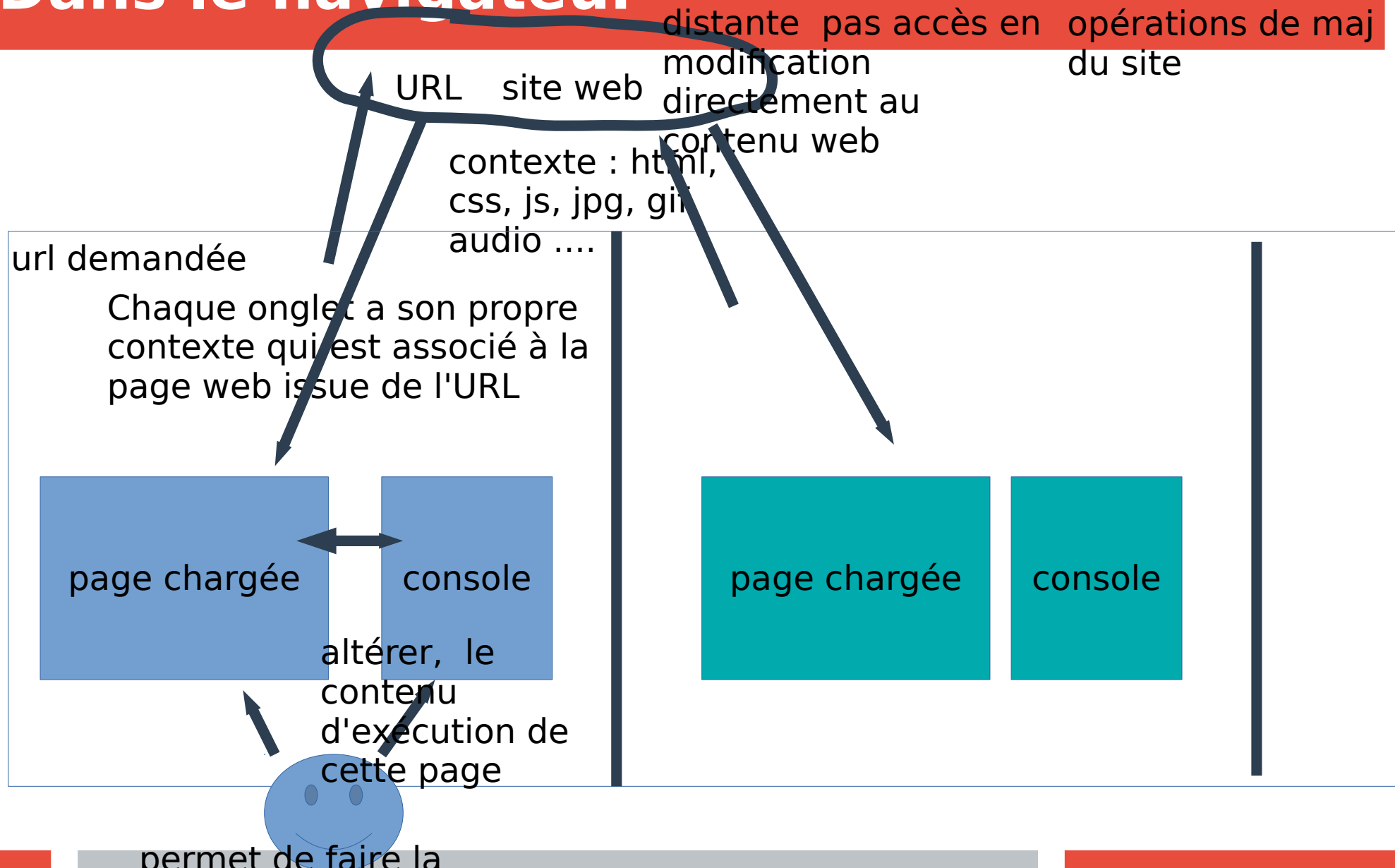
```
class etudiant {  
    #nom=Nom ;  
    #prenom=Prenom ;  
    #date_naissance=DateNaiss ;  
    #matieres={ } ;
```

Pour utiliser dans votre code :
var etu1=new etudiant("Stroppa","yvan") ;

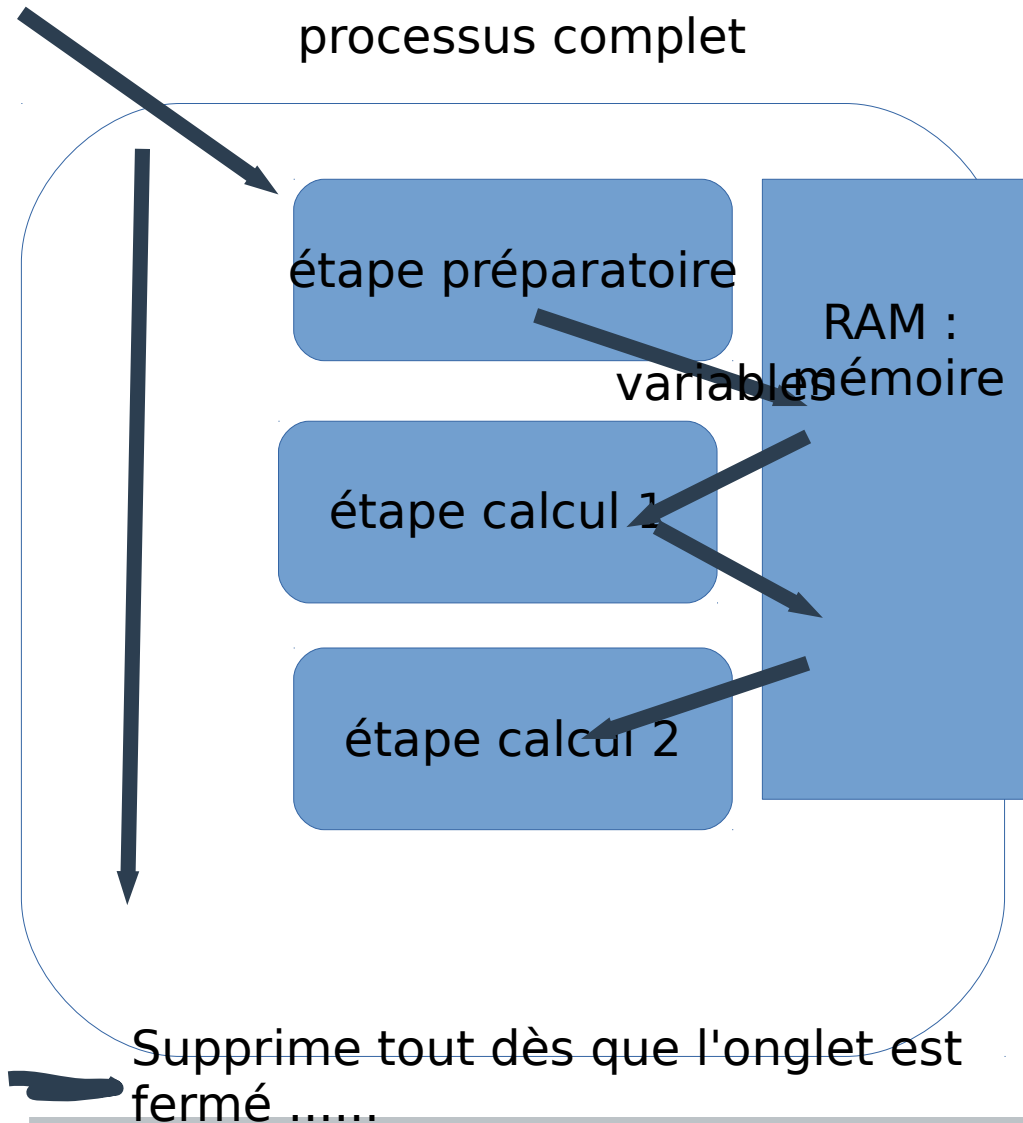
```
    constructor(Nom, Prenom, DateNaiss) {  
        this.#nom=Nom ;  
        this.#prenom=Prenom ;  
        this.#date_naissance=DateNaiss ;  
        this.#matieres={ } ;  
    }  
    ajoute_matiere(mat) {  
        this.#matieres[mat]=[] ;  
    }  
    ajoute_note_matiere(note,mat) {  
        this.#matieres[mat].push(note) ;  
    }  
}
```

Dans le navigateur

nécessite un accès
particulier pour
effectuer des
opérations de maj
du site



traitement en étape



Définition et controle de type

Dans Javascript comme dans Python

Un déclaration implicite du type des variables

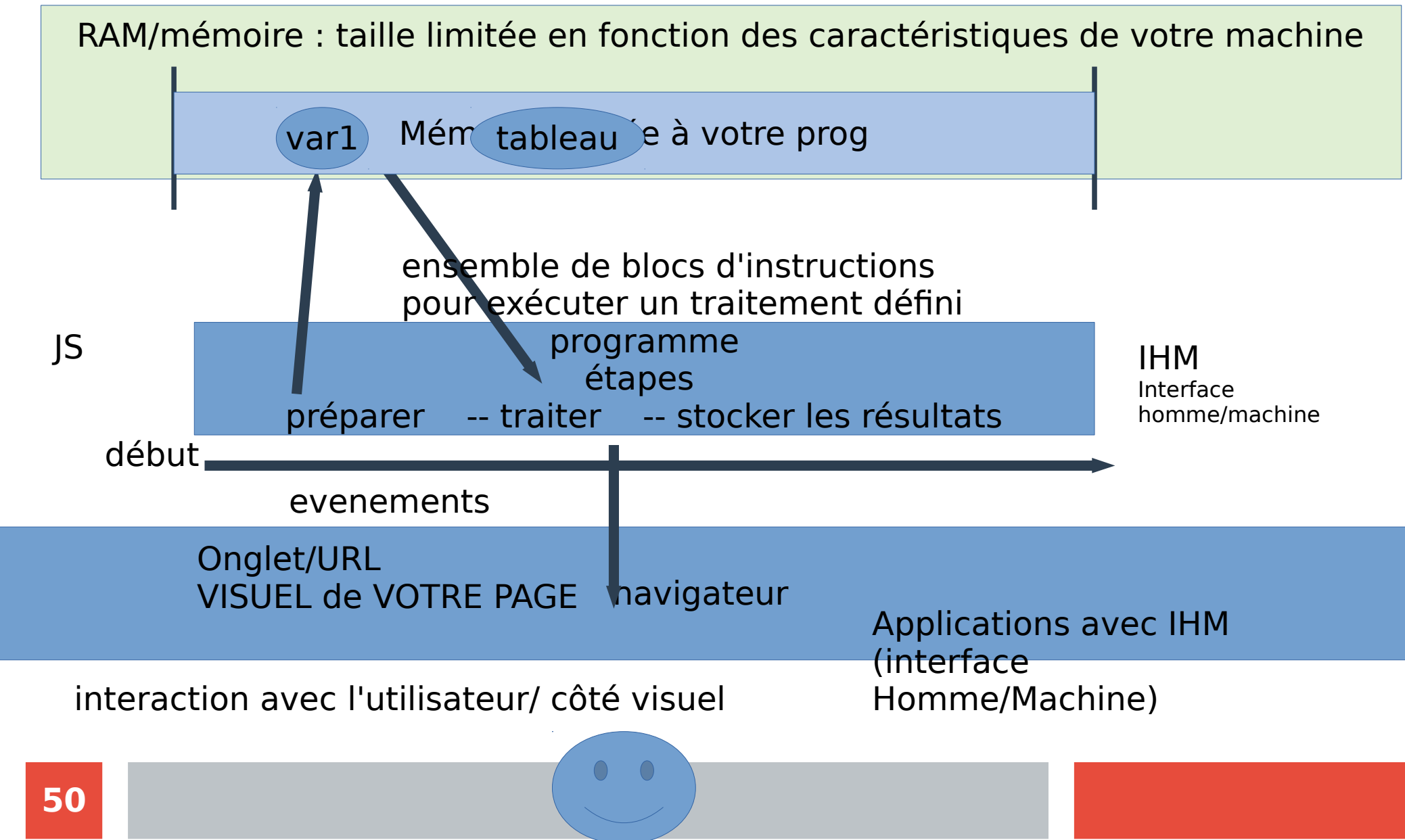
Il n'y a pas de contrôle implicite des types

Dans d'autres langages de type Java, C, C++, typescript

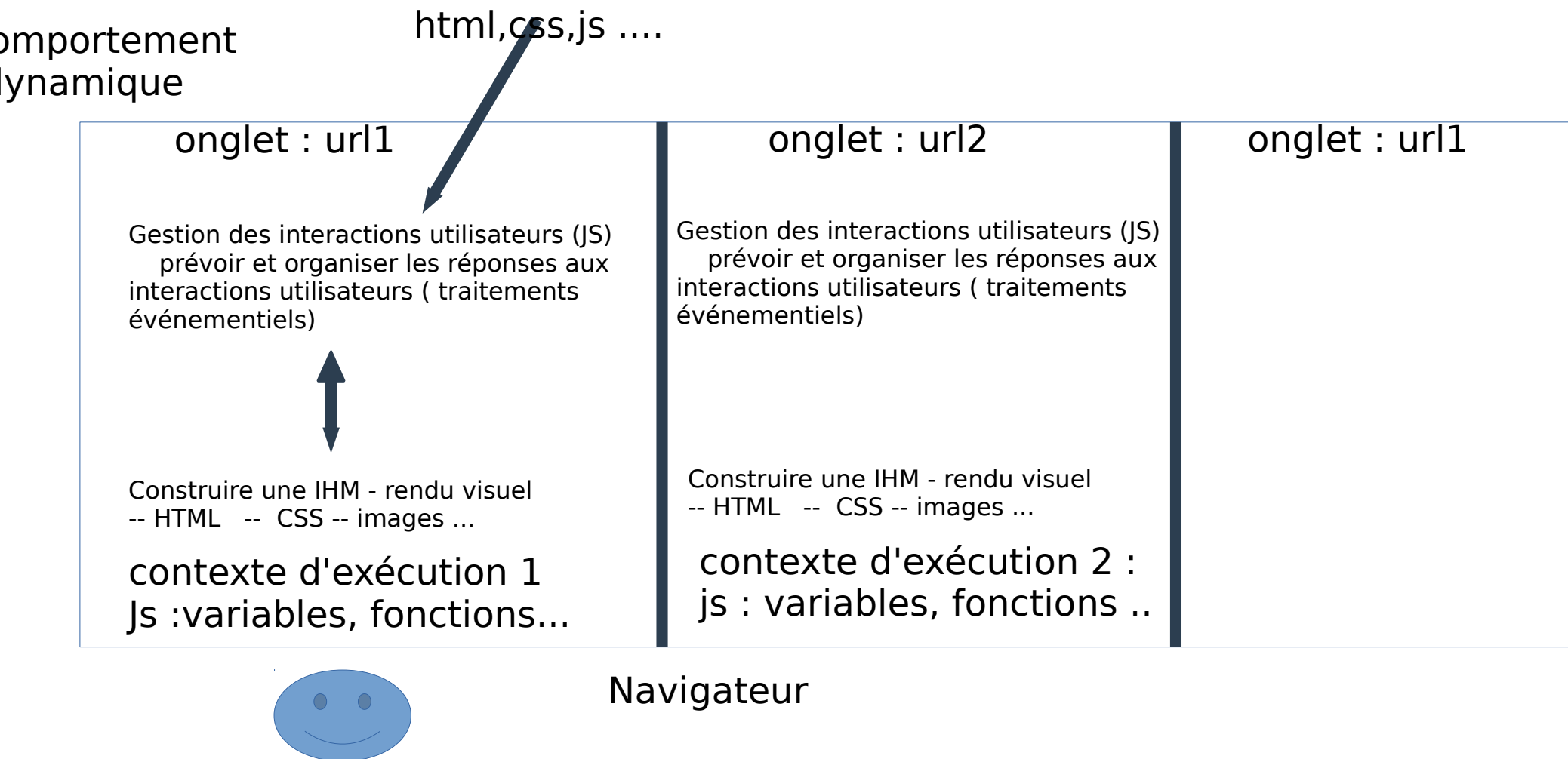
Déclaration explicite des types `public int age ;`

Contrôle des types au moment des affectations

Langage informatique : Variable



contexte technique de votre application web



structure d'une page HTML

<!DOCTYPE html>

<html>

<head>

<!-- déclarations de meta-données utilisés par les moteurs de recherche -->

<!-- inclusions de fichiers css, js --> <script src="http..."></script>

<!-- blocs style ou js -->

<style> </style>

<script> </script>

</head>

<body>

<h1> salut à tous </h1>

</body>

</html>

permet d'indiquer au navigateur la version html utilisée (version 5)

description des tags issue de XML

Mécanisme de type

**Déclaration d'une variable : pas de type explicite
à l'aide de trois mots clés
var ou let ou const**

**Lors de l'affectation : définit un contenu
le contexte associe à la variable un type
(String, Number, Date, Array , Object)**

**Chaque type == définit un Objet (une classe)
qui possède des attributs et des fonctions
(méthodes prédéfinies)**

Différents types d'application

application en mode batch (lot)

input -- exécute -- output

fichier -- prog -- fichier résultat

Applications avec IHM

+ à gérer l'interaction avec l'utilisateur (prévoir anticiper orienter le comportement de l'utilisateur)

applications en standalone (installer sous système d'exploitation)

applications légères/ en ligne : elles s'appuient sur le navigateur (les applications Online qui sont les plus utilisées)

Déclaration des variables et types de variables

Dans certains langages vous avez obligation de déclarer et de typer vos variables (c,c++,Fortran, Java ...) ce sont les langages fortement typés. (voir des exemples)

Dans d'autres, il y a un typage implicite, ce qui veut dire c'est en fonction de la valeur que l'on stocke que le système associe le type à votre variable (e : python, javascript, R, Matlab ...)

Structure de variables

Construction

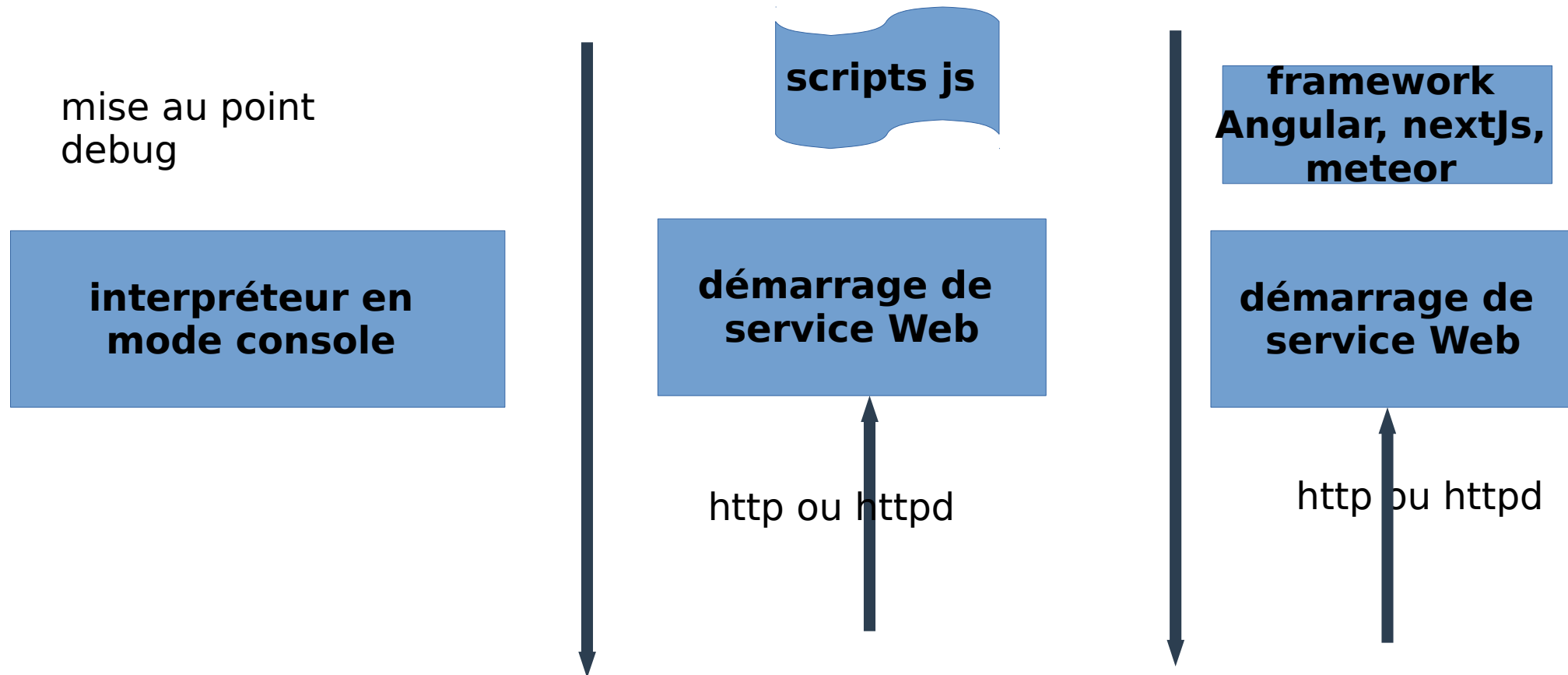
Etre capable de les définir (de les construire)

Manipulation

Etre capable d'extraire les informations

Etre capable de les modifier ou de la compléter

NodeJs : environnement tout JS



Blocs d'instructions

affichage dans la console (afficher différentes valeurs dans la console)

blocs d'instructions conditionnels

condition : si valeur faire

```
if (condition) {
```

```
    // vraie
```

```
} else {
```

compléments !

```
    // fausse
```

```
}
```

html : attention aux contrôles des service
web

1^{er} congrès

js : petite erreur -- effet important

```
if ( true ) {
```

```
}
```

```
if (condition1 && condition2) {  
    // deux conditions sont vraies
```

ET

```
}
```

```
if (condition1 || condition2) {
```

```
    // si une des deux
```

OU

```
}
```

blocs conditionnels

notion de switch (contenu)

```
switch (contenu) {  
    case "Oranges" :  
        //instructions pour cas 1 ;  
        break ;  
    case "Pommes" :  
        //instructions pour cas 2 ;  
        break ;  
    case "Poire":  
        //instructions pour cas 3 ;  
        break ;  
  
    default :  
        break ;  
  
}
```

```
if (contenu==1) {  
    // bloc instruction 1  
  
} else if (contenu==2) {  
    // bloc instruction 2  
  
} else if (contenu==3) {  
    // bloc instruction 3  
  
} else {  
    // bloc instruction autre  
}
```

blocs itératifs : faire des boucles

bloc de type for

on commence à début et on finit à fin (itération explicite)

```
for (let i=0 ; i < 10 ; i++) {
```

// blocs d'instructions

```
}
```

condition d'exécution de l'itération

```
for (let i=0 ; i<10;i++)  
  console.log(i) ;
```

i++ ==> i=i+1

```
i=0 ;  
while (i<100) {  
  // instructions  
  console.log(i) ;  
  i=i+1 ;  
}
```

```
while (condition) {
```

```
}
```


ne pas omettre la condition de sortie sinon boucle infinie

définition de fonction

Factoriser et réutiliser des blocs d'instructions

Evite d'avoir de la redondance/duplication de blocs d'instructions

difficilement réutilisation
et modifiable



Instructions
bloc 1
instruction

instruction ...
bloc1
instruction

"factoriser"

instructions
bloc1() ;
instruction ...
bloc1() ;

Définition d'une fonction
function bloc1() {
instruction 1
.....
}

réutilisation -- test unitaire
(dans le cadre de la
validation)

Complément fonction

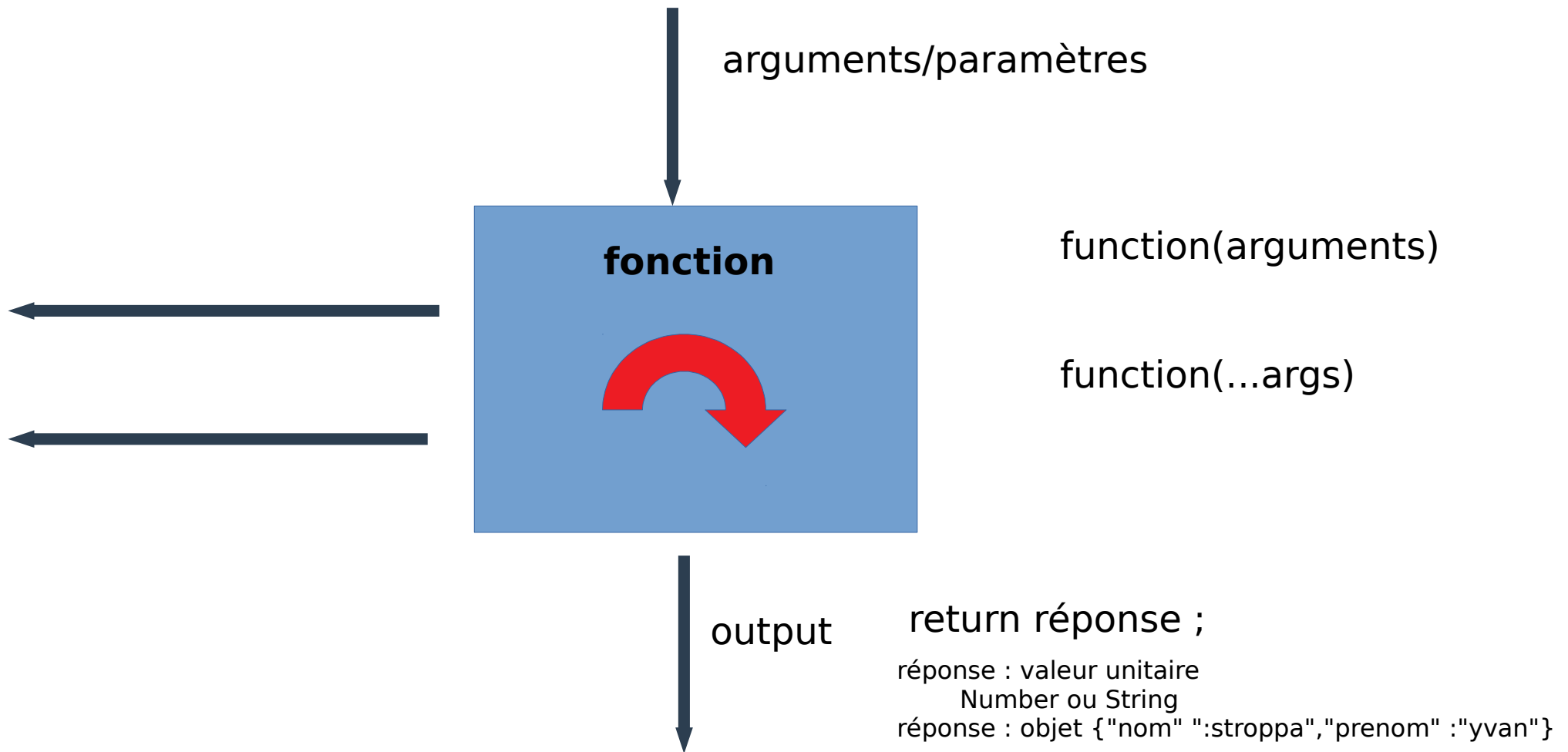
```
// commentaires sur la fonction et les paramètres
// a : représente telle valeur
// b :
// c :
```

```
function g(a,b,c) {
  if (c===undefined) {
    console.error("manque une valeur c");
    c=1 ;
  }
  console.log("je suis dans la fonction g",a,b,c);
}
```

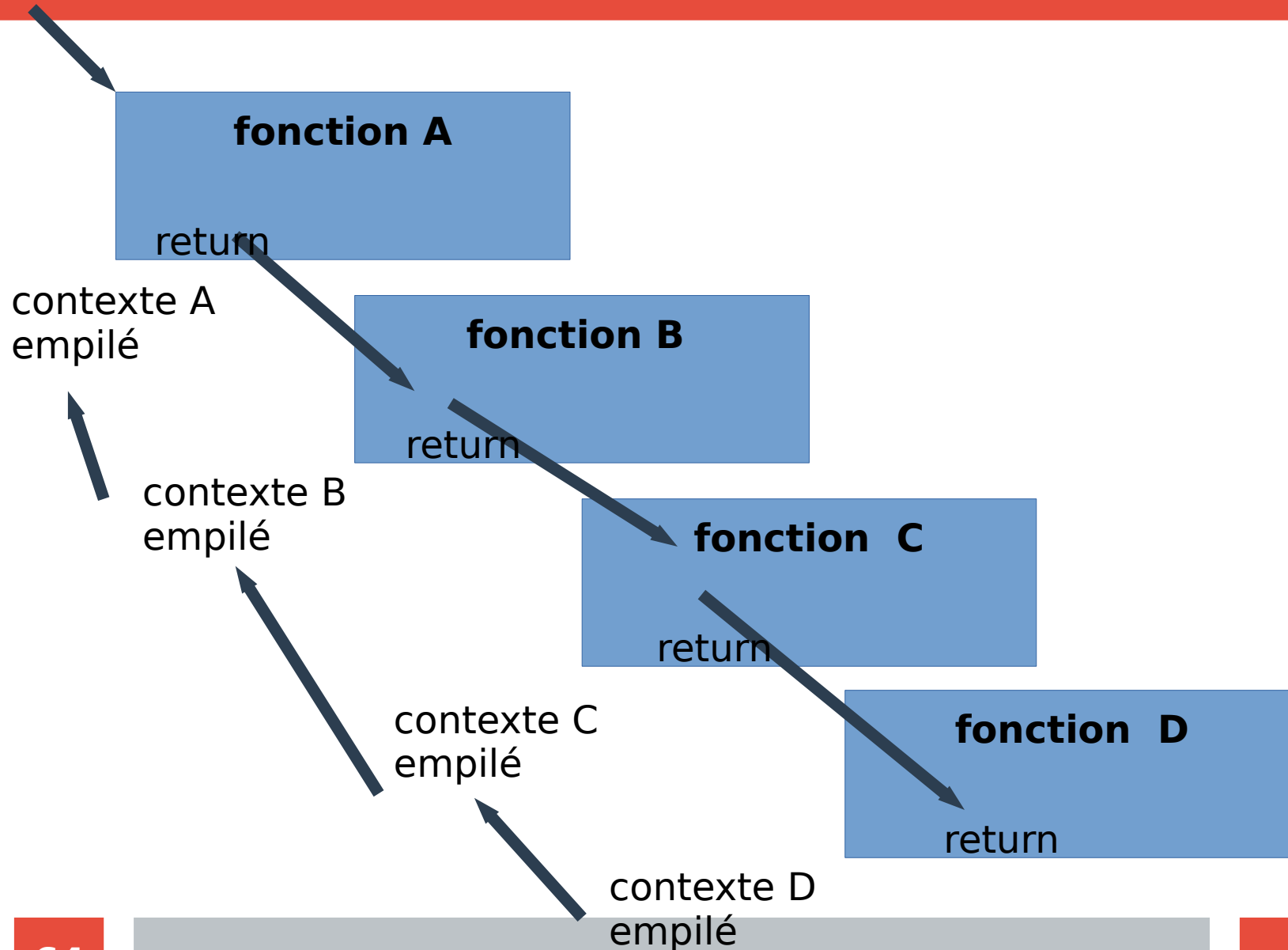
return : retour de la
fonction
sort de la fonction

```
function g(a=0,b=0,c=0) { //valeur par défaut
  console.log("je suis dans la fonction g",a,b,c);
}
```

Définition fonction



empilement des appels de fonctions (en mode classique)



Différentes formes de fonction

Forme classique d'une fonction

```
function f(a,b,c) { }
```

fonction sous forme d'expression

```
const f=function(a,b,c) { }
```

fonction fléchée

```
f=function(a) { }
```

```
f=a=>corps de la fonction ;
```

```
f=(a,b) => corps de la fonction ;
```

```
f=(a,b) => { corps de la fonction multi lignes ; }
```

Javascript et Html

Enjeux et interactions

Comment se passe les échanges entre les deux parties d'une même application

la partie HTML : permet de décrire le visuel (IHM) par l'intermédiaire de tag

`<div>`, `<p>`, `<li,ul,ol>` `<article>` `` `<form>` ``

Chaque tag peut posséder des attributs qui vont permettre de lui apporter des caractéristiques supplémentaires :

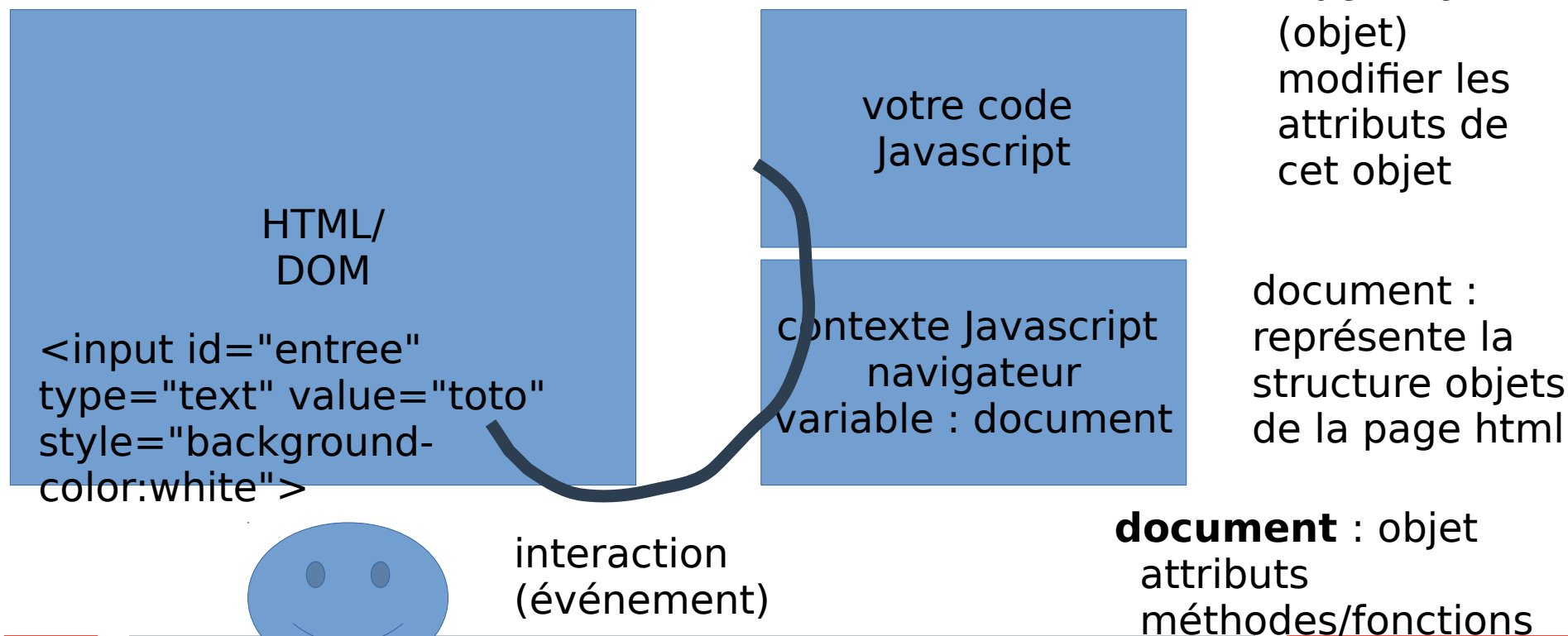
de type style, contenu, identification, ... pour récupérer ensuite à partir de javascript les valeurs on pourra utiliser les méthodes associées à document de type `document.getElementById("identification")`.

structure HTML

HTML est associé une DOM (Document Object Model) -- s'appuie sur des tags

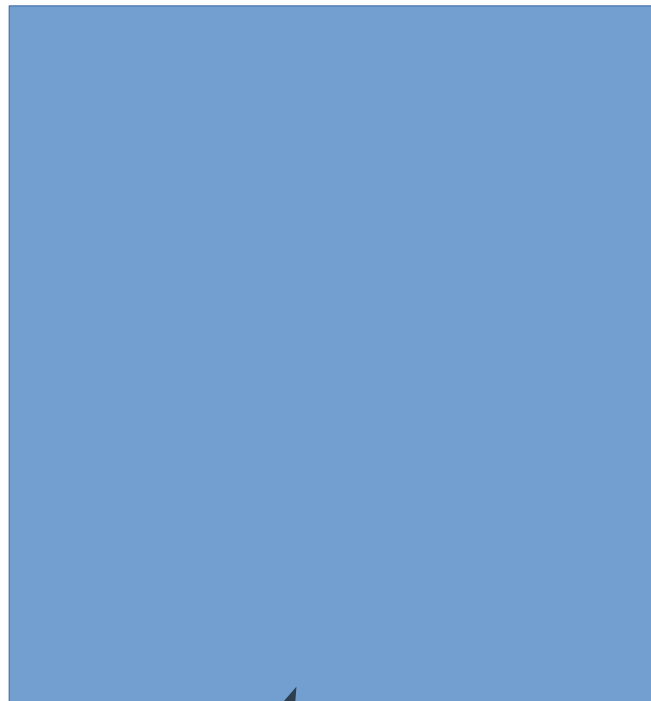
Les tags `<body>` `<div ...>` `<p>` `<input ...>`

Structure l'information à partir des tags



structure DOM

Javascript



identifier l'objet sur lequel on veut travailler
effectuer les opérations sur cet objet

- extraction
- modification
- suppression



- ajout d'enfants

action utilisateur --> réaction de l'application



http:// URL
html,css,js,png,gif

Dans un onglet

Affichage d'un contenu d'un site

HTML + CSS

contexte d'exécution
javascript de l'onglet

objet : **document**
(contient la structure
complète de la page
html en mémoire)

javascript

Structure de vos pages HTML

<!DOCTYPE html>

<html>

<head>

<style></style>

<link rel="stylesheet" href=".....css" />

<script src="https://.....js"></script>

<script>

// javascript :

// déclaration de variables

// déclaration de fonctions

function toto() {

**// blocs d'instructions cond ...
itératif**

}

</script>

</head>

<body>

</body>

</html>

Téléchargement des
ressources js et css

Organisation de vos fichiers
/index.html ou /index.php

Arborescence de votre site
/index.html

scripts/

fichiers js

styles/

fichiers css

Gestion événementielle du côté HTML

contexte HTML

```
<input class="" type="text" value="dldl" style="... ;" ... onclick=""  
ou onchange="" />
```

```
<input type="button" value="dldl" style="... ;" ...  
onclick="appel_fonction() ;" ou onchange="appel_fonction_1() ;" />
```

```
<script>
```

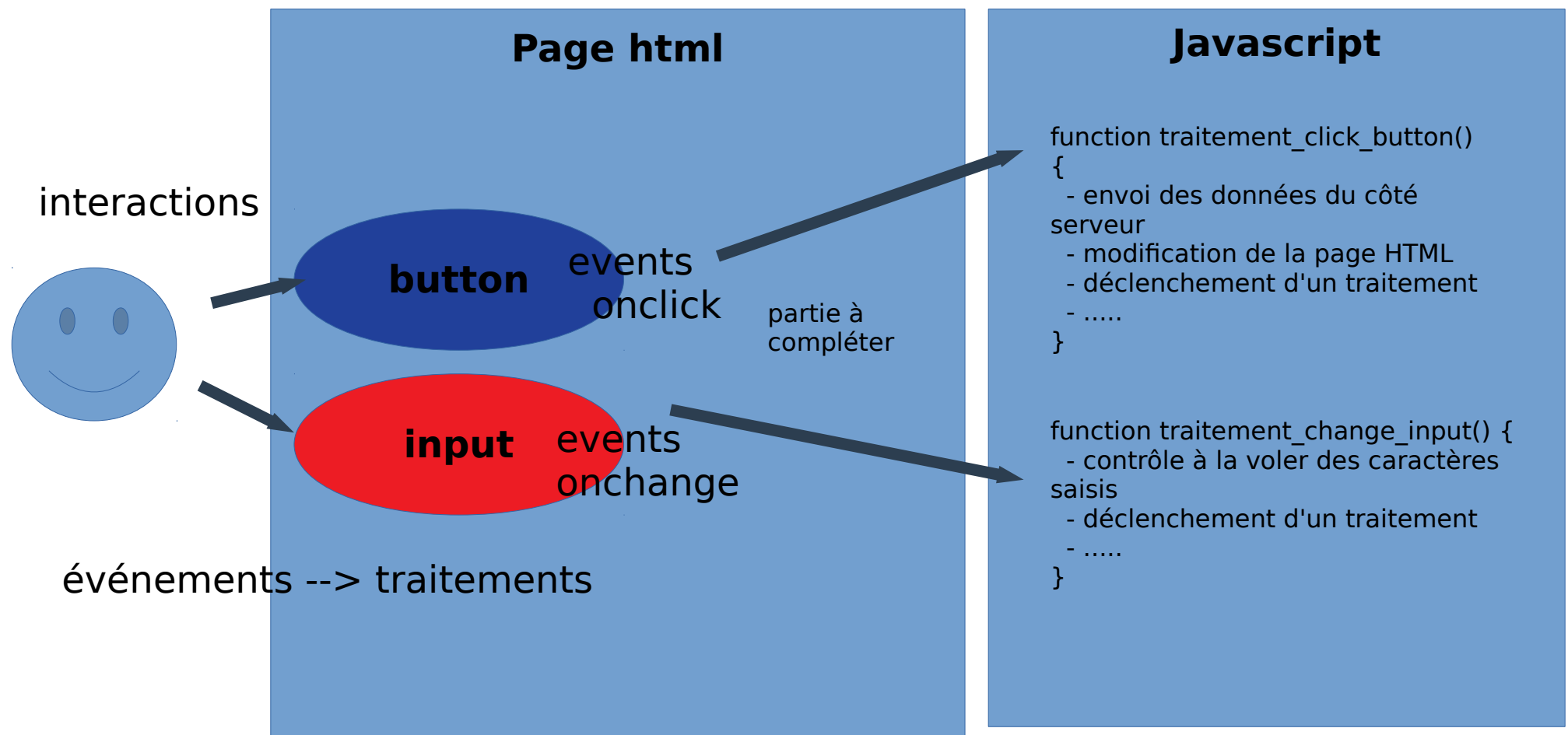
```
function appel_fonction() {
```

```
}
```

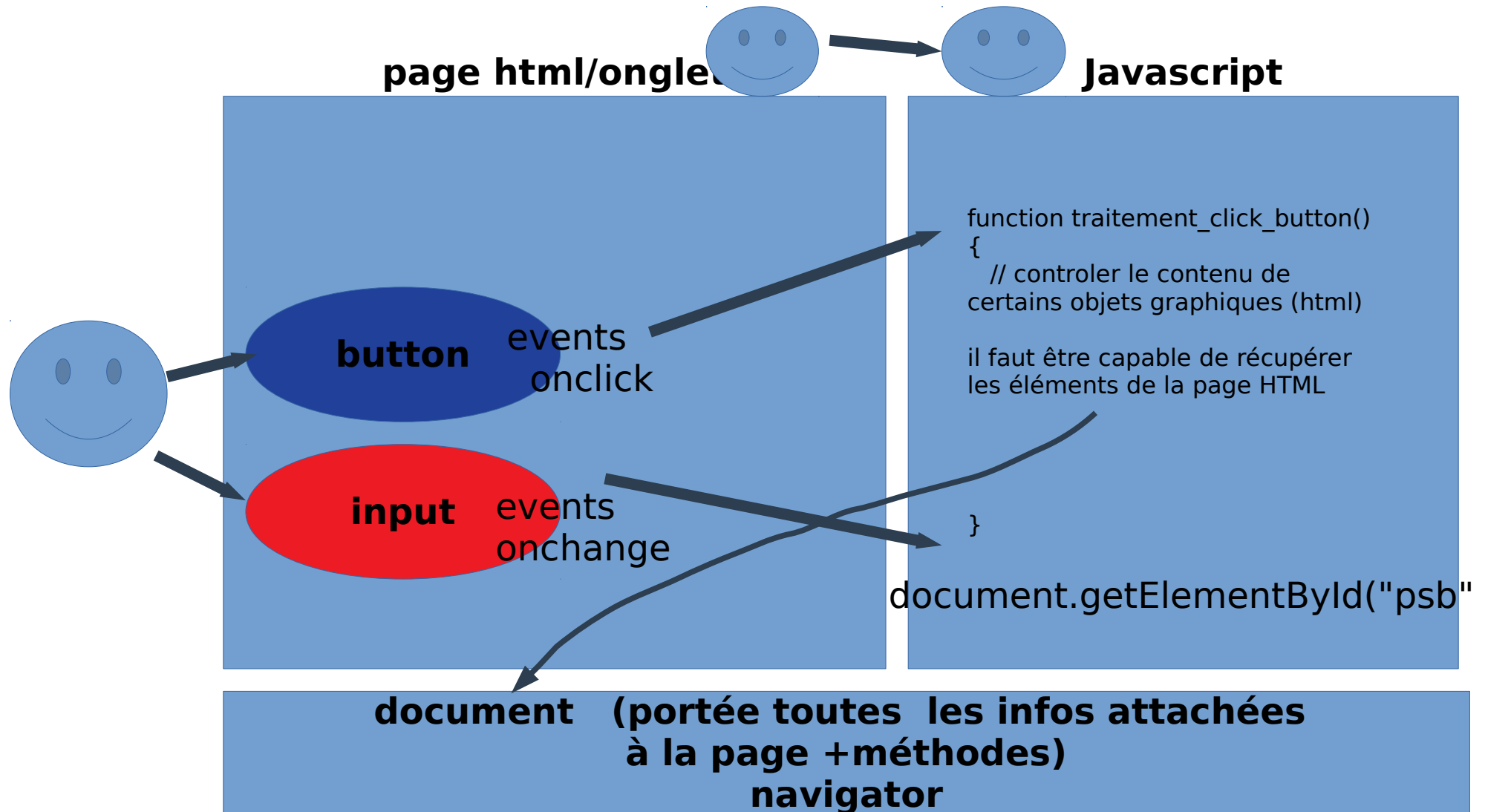
```
</script>
```

Construction de l'IHM (interface Homme-machine) : tout prévoir

Partie visible de l'application



Construction de l'IHM : tout prévoir



Exemple autour d'étudiant/article/produits/entreprise

**Monter une structure qui contient les attributs
d'étudiant et les méthodes associées :**

Exemple autour d'étudiant

Monter une structure qui contient les attributs d'étudiant et les méthodes

calcul de l'age :

différence entre deux dates : retour en ms à convertir en années 1000/60/60/24/365

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

Notion de classe

Définir une classe

Intérêt dans nos programmes

notion d'attributs associés à la classe

notion de **static**

attribut

méthode

notion d'attribut protégé

avec le caractère **#**

Notion d'héritage avec le mot clé **extends**

Opérateur new pour instancier un objet issue d'une classe

A travailler sur le document

https://github.com/ystroppa/PSB2020/blob/master/explications_complementaires_javascript.pdf

Annexes

Evaluation des projets

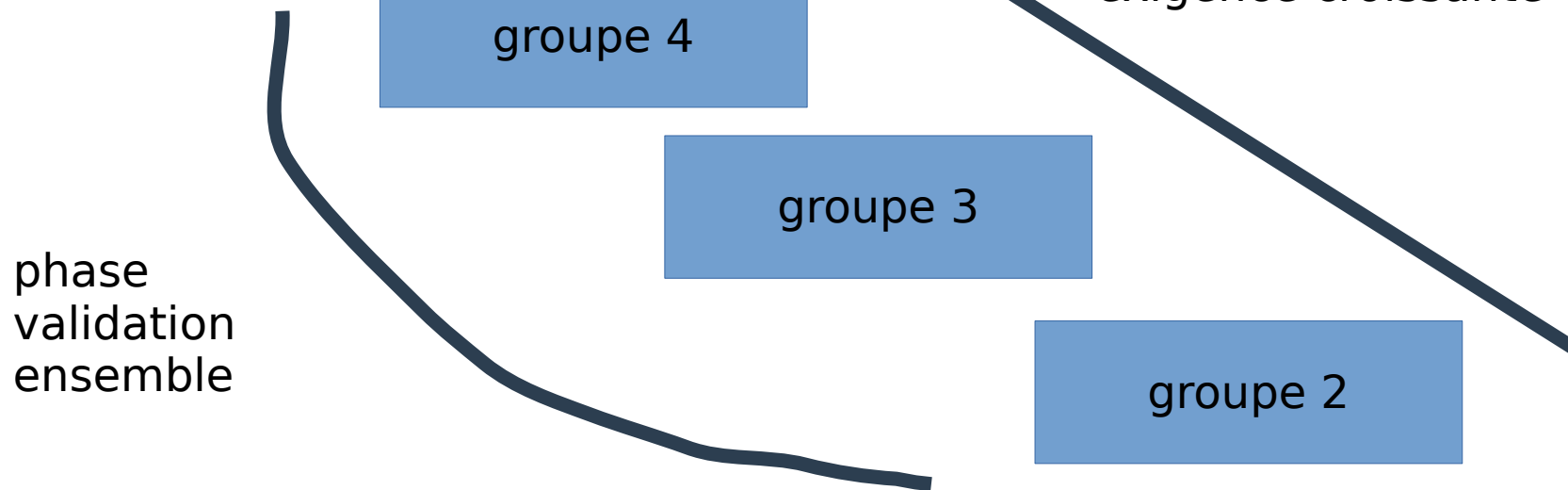
deux phases :

Rapport

Test de la réalisation

quantité / complexité

exigence croissante



Réalisation d'un projet

Vue globale -- description la plus précise possible des fonctionnalités

Découper en lots : organisation/répartition

Description fonctionnelle et ses interfaces avec les autres

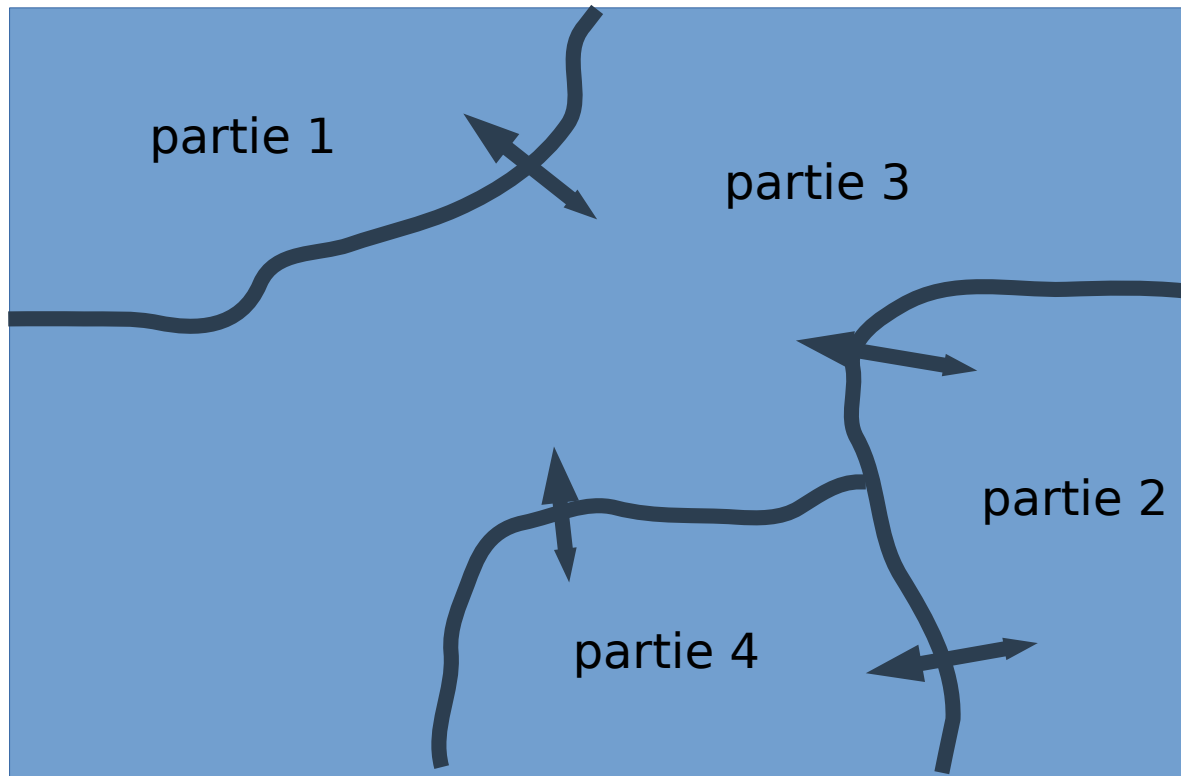
Développer les lots /mettre au point /tester

de façon unitaire (javascript -- modulaire -- fichiers dédiés js) --
vérifier compatibilité pour toutes les navigateurs

Phase d'intégration qui consiste à ramener les lots dans leur contexte cible d'exécution

Tests globaux

problème



Déléguer et répartir
travail en groupe

décomposition en partie /lot
décrire les interfaces entre les
différentes parties

PSB 2023 v1.0

Contexte de développement Web

Cours Javascript :

dépôt : [https://github.com/ystroppa/PSB2020\(historique\)](https://github.com/ystroppa/PSB2020(historique))

<https://github.com/ystroppa/PSB2023-M1>

Sujet à valider ensemble

Yvan Stroppa

IR CNRS cursus informatique dans un contexte Linguistique

Yvan.stroppa1@univ-orleans.fr (préciser PSB - M1 - groupe avec un objet)

Projet : 2 à 3 étudiants --- à rendre fin juin 2023 (date à préciser)

- 1 / Rapport (pdf)

Cahier de charges : expression de besoins

Analyse et conception

Explications et illustrations

Phase des Tests (vérif sur tous les navigateurs)

Conclusion

- 2/ Livrable : Programme à livrer (tous les fichiers)

Accompagnement
technique si besoin
par mail, bb, autres

jusqu'à fin juin

Dépôt de doc.... ou envoi par mail

(avec accusé de réception)

Evaluation

Evaluation des rendus : sous un contexte ordinateur portable Linux - navigateur (chrome, firefox ...)

Développeur

Pour résoudre une problématique (échéance, budget)

Intégration - Récupérer du code / ou de produit licence :

- approprier le code (évaluer et vérifier l'intégrité)
- intégrer le code dans votre contexte
- tester

- indiquer les sources

Développement - Produit du code : (documenter)

- algorithme
- traduire dans le langage cible (javascript)
- Mise au point et tester (cas de tests pour valider le code)

Contexte du projet : technique et pédagogique

Le contexte de développement de vos projets va s'inscrire dans un objectif de fabrication de CDS "Cahier Didactique du Savoir".

L'idée est de présenter et d'expliquer un concept (algorithme, théorème ... au autre) dans une SPA (Single Page Applicatif)

Dans une page (html) : public visé : primaire, collège, lycée, supérieur

- Page de présentation : logo "CDS PSB" + concept
- Introduction du concept - historique (voir les infos wikipedia et autres)
- Explications et illustrations du concept (avec des parties **dynamiques, interactives et sonores(mp3, wav ...)**)
 - 1 ou 2 Exemples (interactifs, sonores)
- 1 ou 2 Exercices avec correction (optionnelle)
- Quizz de 5 questions aléatoires sur une liste de 20 à 25 (pseudo évaluation) HTML, CSS, JS (images, graphiques, sonores,)
- Conclusion et liens public visé : primaire, collège, lycée, supérieur ...
Thématiques : Economie, Informatique, Mathématiques, autres(Langage)

CDS : sujets

Parcours d'arbre

1/ algorithme de Kruskal

2/ algorithme de Dijkstra

3/ algorithme de Bellamn-Ford

4/ algorithme de Floyd-Warshall

Sur les automates : expressions régulières et langage

5/ algorithme de McNaughton et Yamada

6/ algorithme de Brzowski et Mc cluskey

Vous pouvez définir votre propre sujet --- à valider avant de commencer

Intérêt : composants logiciels (module)

Regardez les bibliothèques js de type impress.js et autres dans le même contexte.

7) Introduction à la théorie du chaos à une dimension

- construction des graphiques
- proposition des fonctions
- graphique
- diagramme à une dimension

8) Dilemme du prisonnier : tournoi d'Axelrod

https://axelrod.readthedocs.io/en/fix-documentation/reference/overview_of_strategies.html

9) Dilemme du prisonnier spatial

10) Théorème de Pythagore (démonstrations graphiques épaulées par l'algèbre si nécessaire)

11) Diagramme d'Edgeworth animé

12) Pi par la démonstration des camemberts d'Archimède

13) Calcul d'aires et d'intégrales (calculs par les aires des rectangles par défaut et par excès)

14) Percolation

15) Planche de Galton (possibilité d'ajouter des obstacles et des frontières)

16) Systèmes de vote

17) Evaluation d'une fonction d'utilité en incertain (j'expliquerai)

18) Application du théorème de Thales (calcul de hauteurs inaccessibles)

19) Diagramme des champs d'un système d'équations différentielle à 2 dimensions $dx/dt = f(x(t), y(t))$ $dy/dt = g(x(t), y(t))$

20) Fonctions 3D et projections de lignes de niveau sur les trois plans

21) Affichage d'une série chronologique et données statistiques associées

22) Animation de la convergence d'une série de Taylor

<http://www.physics.miami.edu/~nearing/mathmethods/power-animations.html>

23) Théorème d'Euler-Descartes

http://serge.mehl.free.fr/anx/Th_Euler-Descartes.html

24) Modéliser le jeu de HEX

25) Les triangles de Sperner

Mise en oeuvre d'un prototype

proposition d'analyse et de réalisation :

réaliser une présentation sous Powerpoint (pour un prototypage rapide)

page/page

description du contenu de chaque page

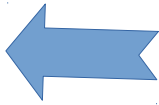
- on réalise son squelette de l'application
- réfléchir sur les interactions que l'on veut ajouter et les définir

Idée est de se mettre d'accord sur le contenu et sur ses interactions (échange avec l'utilisateur)

On peut commencer la réalisation dans l'environnement cible et de voir quelles sont les librairies nécessaires

Exemple : CDS Pythagore

Présentation du concept



Bibliothèques JS

impress.js
reveal.js

<https://digital-cover.fr/20-frameworks-et-libs-javascript-a-connaître-en-2022/>

highcharts.js
charts.js
d3.js
jsxgraph.js

datatable.js **bibliothèque 3D Three.js**

Langages

éditeur - IDE (notepad++ , Sublime, Atom, VCS ... Eclipse, Netbeans, IntelliJ, Visual studio)

C
C++
Fortran
Java
C#

compilés (on passe par un compilateur qui va traduire les instructions en code machine)

Application Standalone
code machine - Binaire

Python
Perl
Shell
ruby
...

interprétés : on utilise un interpréteur

Javascript

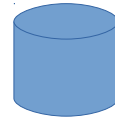
langages de description
HTML : langage tagué
CSS :

Pages Web
navigateur

Persistance

Language SQL

Bases de données
Mysql
MariaDB
Postgresql

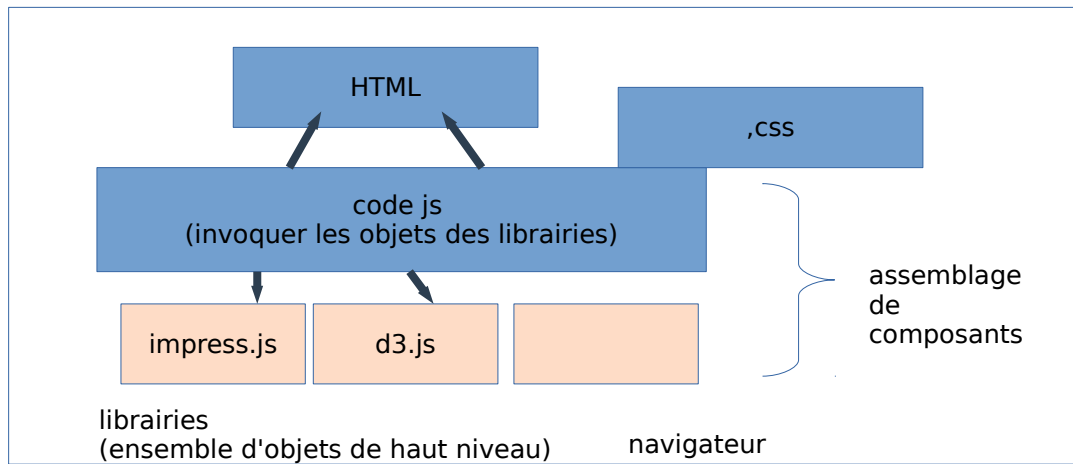


Oracle
DB2
Sql server

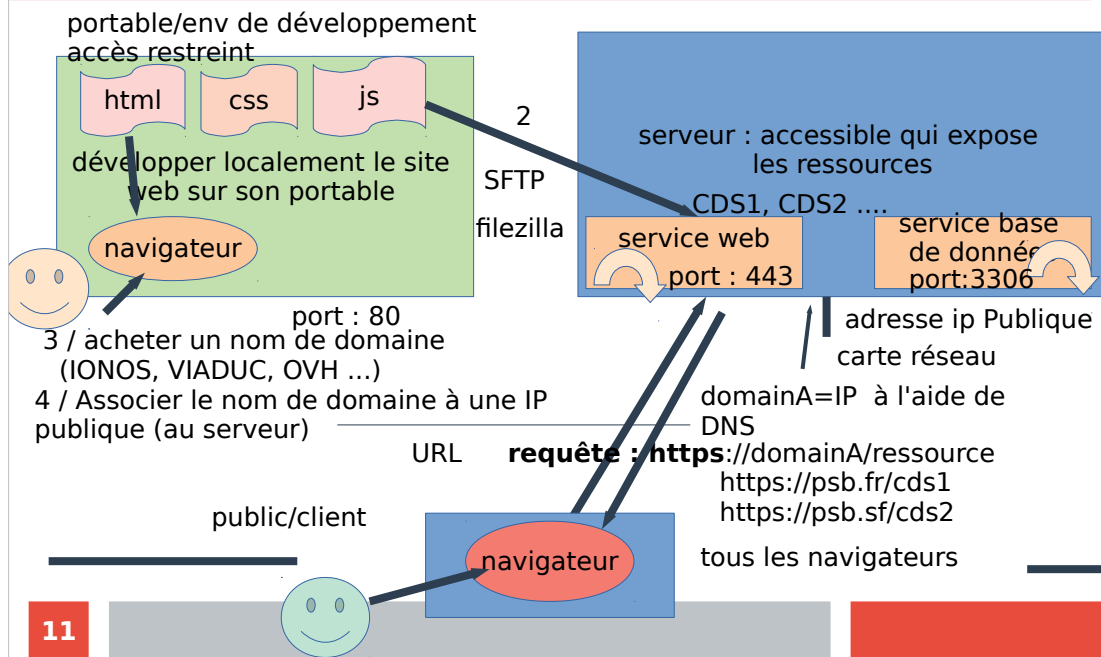
// NoSQL
Mongodb
Couchbase



utilisateurs

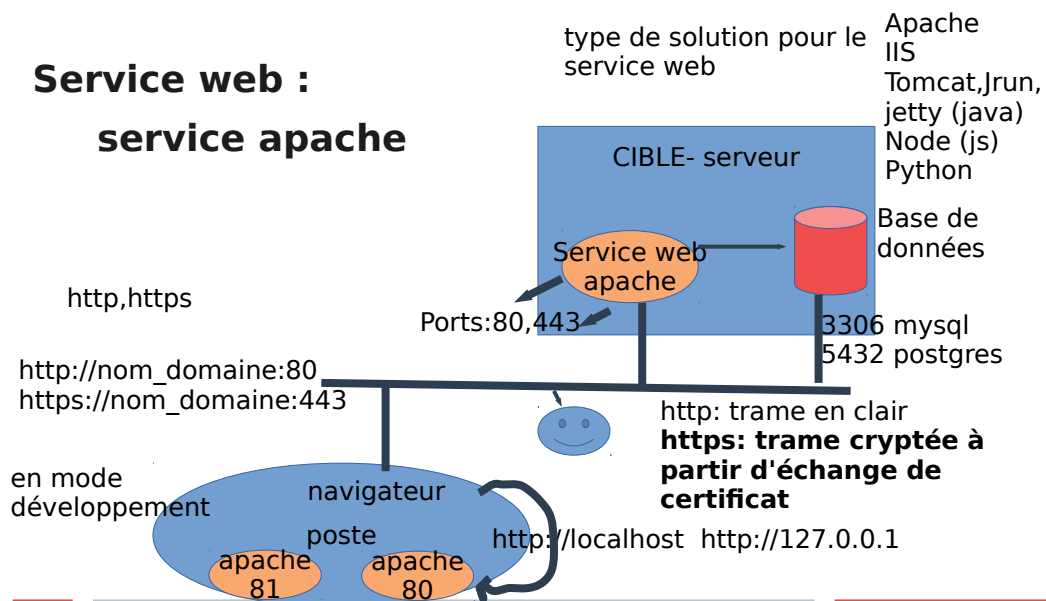


Principe de base de dév d'un site web



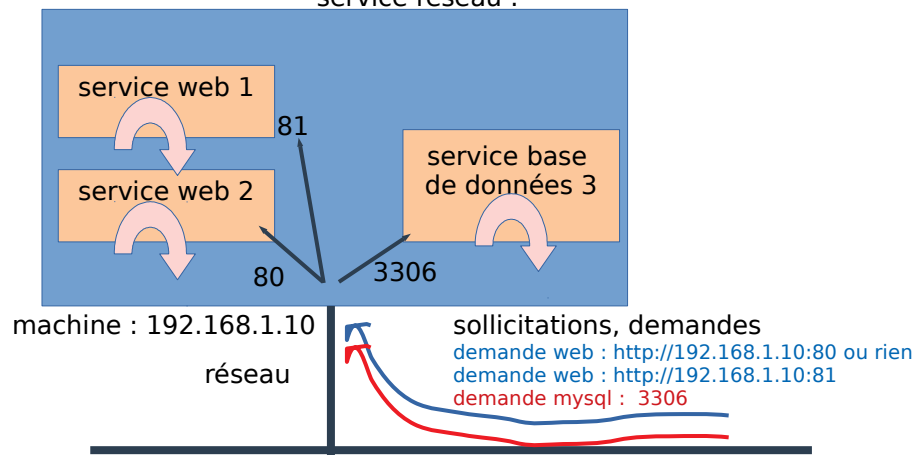
Services web / Architecture

Service web : service apache

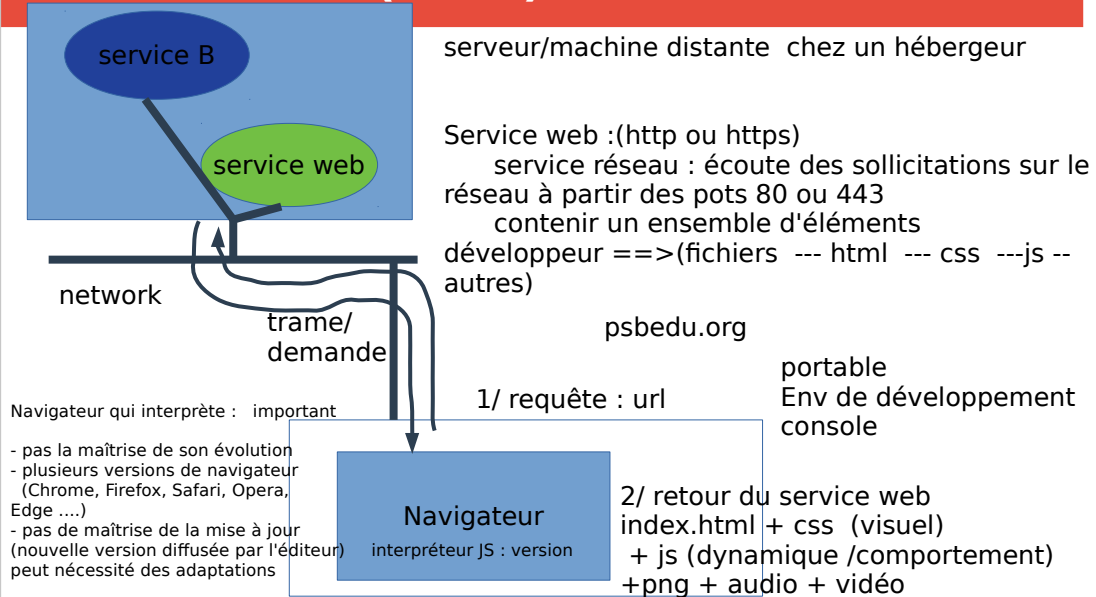


Numéro port pour les services

service = application qui fonctionne tout le temps
service réseau :

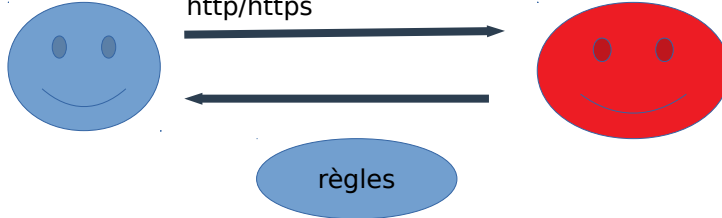


Architecture (suite)

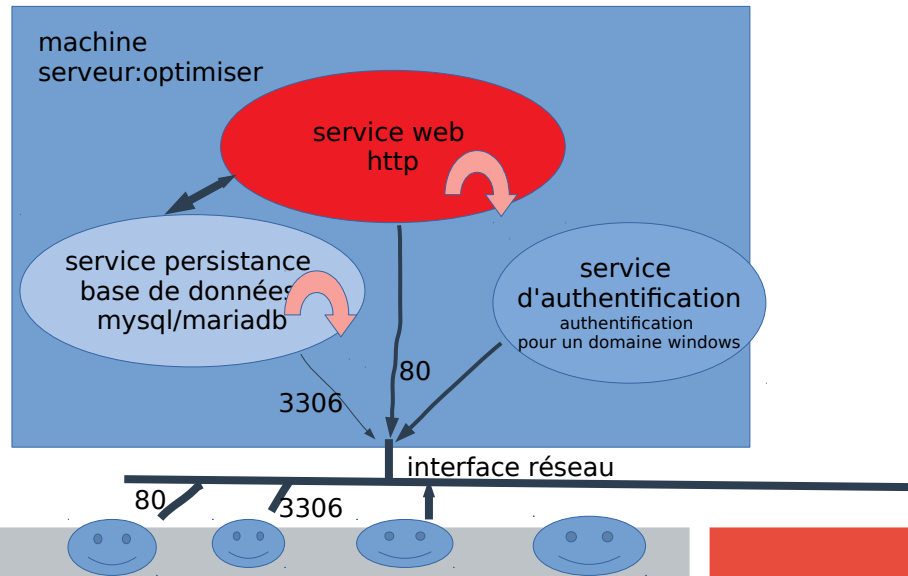


protocole

Dans le cadre d'une
communication entre
deux parties :
Protocole d'échange et
communication
http/https



services



Application WEB : hébergée sur une machine (serveur ou en dev sur votre portable) accessible par la protocole http

Elle est composée :

HTML (documents ou fichiers html) inclusions : script et css
structure de vos pages web

CSS (fichiers de style)

JS (fichiers de scripting en javascript)

la gestion des interactions et événementiel (dynamique sur le poste client (dans le navigateur))

autres ingrédients :

fichiers images

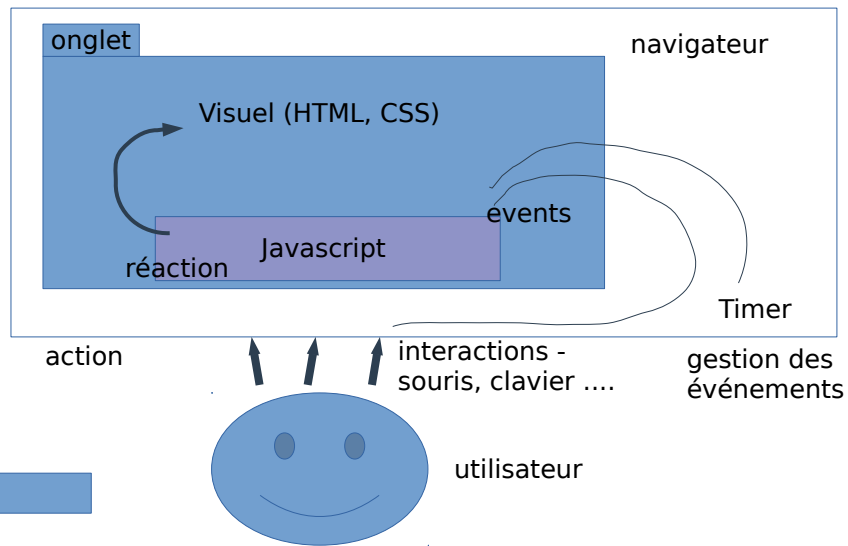
fichiers audios

fichiers vidéos

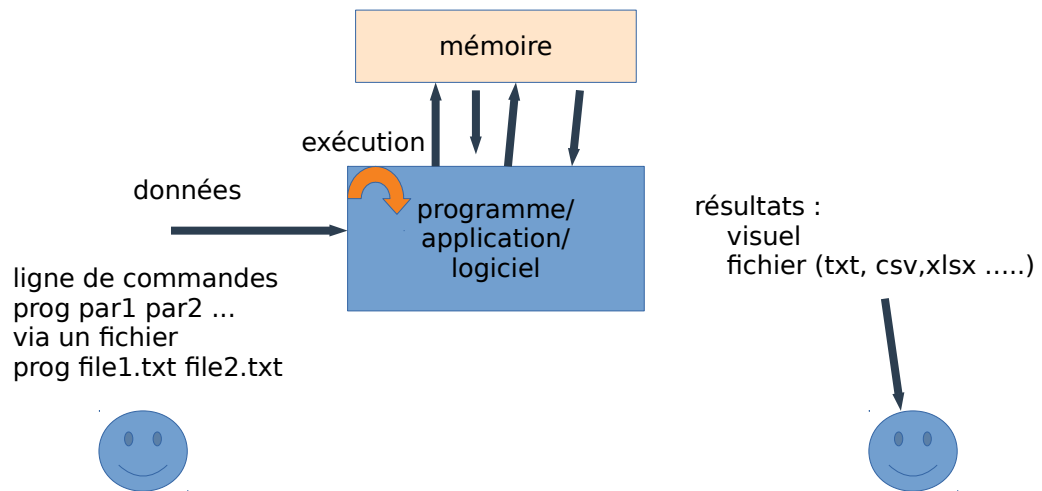
navigateur qui interprète :

- les tags HTML
- Les styles
- les scripts JS

Application - web



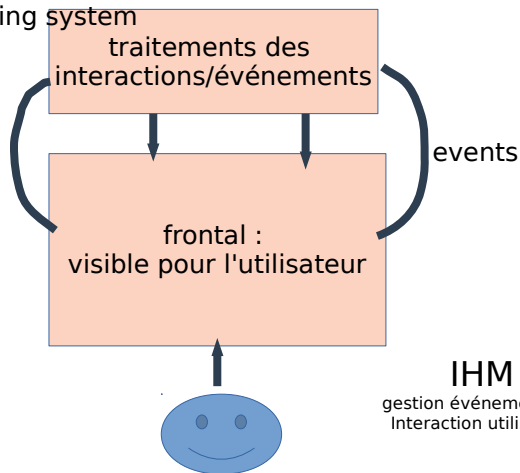
Passage de données : sans IHM (interface homme-machine)



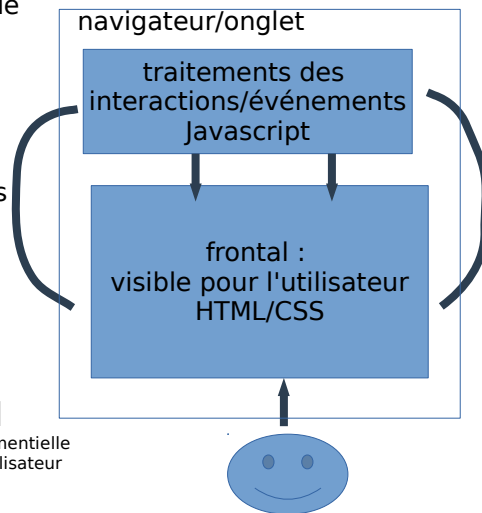
IHM : interface homme/machine

un seul langage Java ou Python
application standalone

operating system

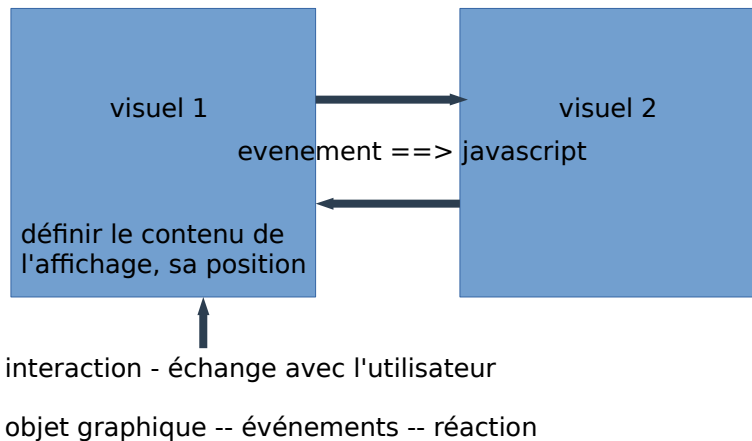


Analogie

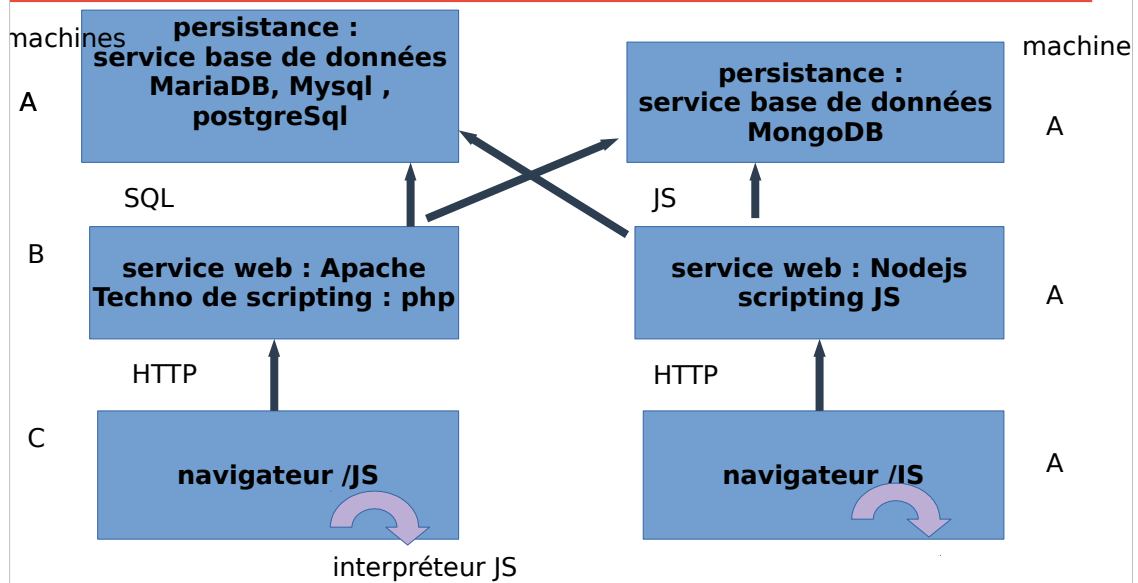


Pour la construction de votre projet :IHM

Peut se construire par un prototype sous Powerpoint : c'est une description et on se met d'accord



Utilisation de Javascript fullstack



À installer sur vos machines

Nodejs (sur tous les systèmes)

interpréteur de javascript

Git (dispo sur toutes les machines)

Installer plusieurs navigateurs pour des tests

- chrome, safari, firefox

- chrome, edge, firefox

Langage informatique

A quoi sert un langage de programmation ?

Programmer des traitements en utilisant des séquences d'instructions du langage.

Que peut-on trouver dans un langage et de quoi a t-on besoin ?

Variables (de différentes natures)

jeu d'instructions

-- Blocs d'instructions

-- Modulaires

syntaxe à respecter
mots clés (spécifique au langage)

Environnement pour faire du javascript

Pour mise au point et interprétation

Navigateur : console

nodejs : interpréteur de commandes

<https://www.programiz.com/javascript/online-compiler/>

jsfiddle

codeopen

....

Fichier js

Algorithmie

Permet de décrire les étapes de son traitement :
utiliser les commentaires pour décrire les étapes
// première étape : récupérer les valeurs saisies
// deuxième étape : vérifier leur contenu
//

données en
entrée

décrire le contenu
de cette boîte

sorties

Langage informatique : Variable

Notion de variable :

Permet de stocker une valeur au cours de l'exécution du programme dans le but de pouvoir la reprendre, la modifier ou la détruire. La variable n'existe que lorsque le programme s'exécute.

Utilisation d'une variable :

Allocation : déclaration explicite ou implicite

Utilisation : `variable1="voiture"`

Désallocation : nécessaire dans certains langages de type c,c++, en Java il y a un mécanisme de libération des variables mémoires (Garbage Collector), en Javascript notion de delete, elles sont généralement déclarée en local.

affectation des types/classes

Dans Javascript tout est objet

un ensemble d'attributs et de méthodes(fonctions)

Le système a défini des objets de bases

String, Number, Date, Array, Object

String

attributs
length

bibliothèque
fonctionnelle

méthodes
indexOf
substr

Chaque objet possède des propriétés et des méthodes associées

Lors de l'affectation de vos variables, il va associer la variable à un objet déjà défini.

var psb1="voiture" ; ==> associe la variable psb1 à la classe/type String, du coup la variable psb1 peut utiliser toutes les méthodes définies dans le type String

psb1.substr(0,2) ; prend les deux premiers caractères de la variable

psb1.indexOf() ; chercher l'occurrence d'un caractère dans la contenu de la variable

var num1=12.12 ; ==> associé la variable num1 à la classe/type Number du coup la variable num1 peut utiliser toutes les méthodes définies dans le type Number

num1.toPrecision(2)

Number

attributs
méthodes
toPrecision
toString

Pour la date il faut utiliser new

new : opérateur qui crée l'objet

var d1= new Date() ;

Type Date
Attributs

Méthodes
de manipulation,
conversion de date

Nuance entre les déclarations à l'aide de var ou de let

```
var topic="stroppa" ;  
if (topic) {  
  var topic="yvan" ;  
  console.log(topic) ;  
}  
console.log(topic) ;
```

yvan

yvan

```
var topic="stroppa" ;  
if (topic) {  
  let topic="yvan" ;  
  console.log(topic) ;  
}  
console.log(topic) ;
```

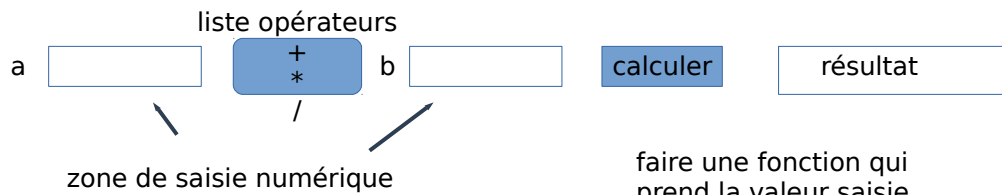
yvan

stroppa

scope de la variable

Exercice pour groupe 36 dans codeopen ou jsfiddle

calcul d'une expression



1 partie : juste faire la somme

faire une fonction qui
prend la valeur saisie
dans la zone a et la
zone b et qui effectue le
calcul a+b

2 partie : possibilité de sélectionner
l'opérateur

recupérer un descripteur sur un
objet de la page se fait à l'aide de
`document.getElementById("")`

Equation du second degré grp26

-- présentation du calcul : equa du 2nd degré $y=ax^2 + b*x + c$

-- a<-lire('a') , b<-lire('b'), c<-lire('c')

-- discriminant $\leq b^2 - 4*a*c$

-- si discriminant=0 alors

--faire le calcul solution unique

-- $x = -b/(2*a)$

fin si

-- si discriminant<0 alors

--faire le calcul de deux solutions imaginaires

fin si

-- si discriminant>0 alors

--faire le calcul de deux solutions réelles

fin si

-- affiche le résultat

function solve() {

-- calcul le discriminant

-- en fonction de sa valeur elle fournisse le ou les résultats

Solveur PSB :
Equa du 2nd degré $y=ax^2 + b*x + c$

a b c

résultats

Blocs d'instructions

Bloc d'instruction conditionnel

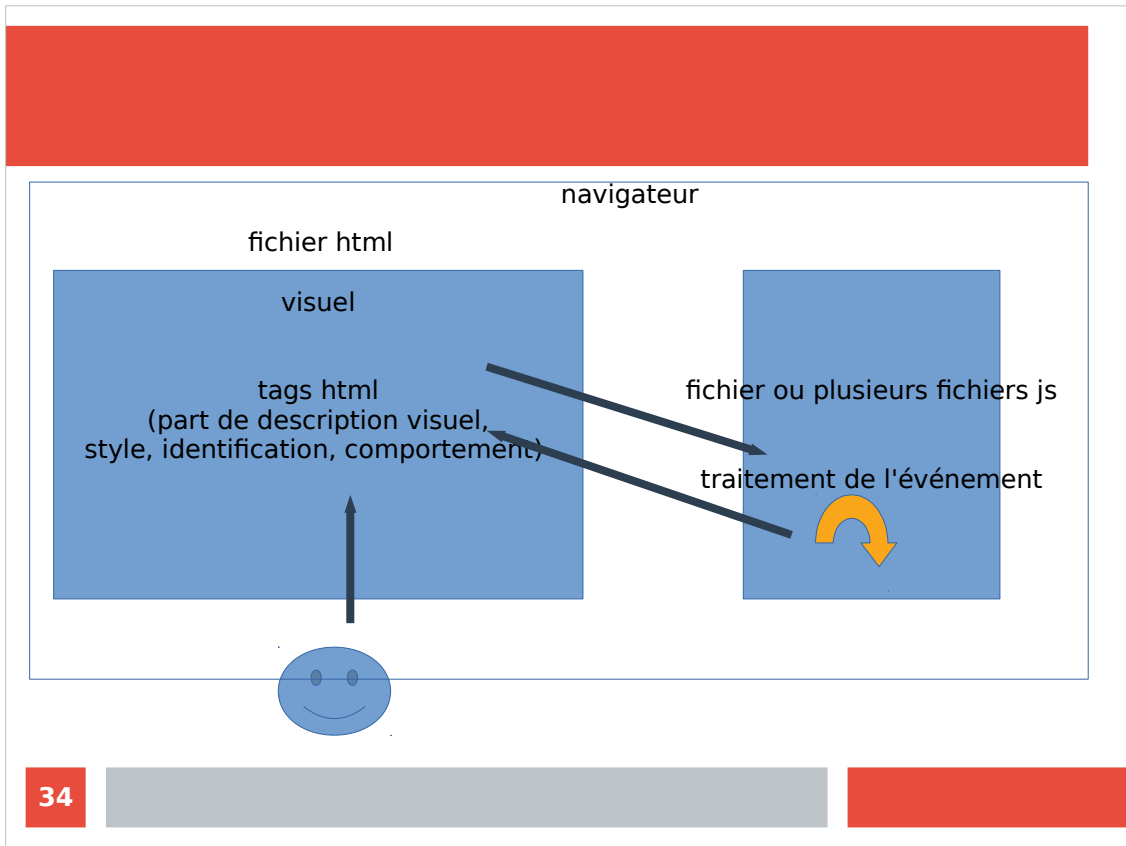
```
if <condition> {  
  } else {  
  
  }
```

Bloc d'instruction : fonction

```
function f1_() {  
  
  }  
f1_()
```

Bloc d'instruction itératif

```
for (let i=0;i<val;i++) {  
  -- bloc répété  
  
  --variable i est incrémentée de +1  
}
```



Attention au changement de typage lors de nouvelle affectation

déclaration de type
implicite

```
var etudiant="stroppa" ; ==> String  
etudiant.indexOf("ld" ); ==> -1 (rien trouvé)  
etudiant.indexOf("ro" ); ==> 2  
etudiant=54 ; ==> Number  
etudiant.indexOf("ld" ); ==> function not found
```

Classique des environnements sans typage explicite
Javascript, Python, Matlab, Mathematica, R

Création d'un nouvel objet

A éviter

```
var etud_nom="stroppa" ;  
var etud_prenom="yvan" ;  
var etud_adresse="rue de la " ;
```

```
// problème si on doit créer un autre  
étudiant  
var etud1_nom="ly" ;  
var etud1_prenom="a-phat" ;  
var etud1_adresse="rue de la " ;
```

.....

Problème : on va multiplier les
variables, et les variables ne sont
pas reliées

object= {key:value, ...} key:unique



```
var etudiant = {  
  "nom" : "stroppa",  
  "prenom" : "yvan",  
  "adresse" : "rue de la .." ;  
};
```

```
var etudiant1 = {  
  "nom" : "ly",  
  "prenom" : "a-phat",  
  "adresse" : "rue de la .." ;  
};
```

```
console.log(etudiant.nom  
+etudiant.prenom)
```

value : scalaire, liste (array),
object

structuration des variables complexes

```
etudiant={nom : "stroppa", prenom : "yvan"....}
```

etudiant ➡ nom : stroppa

notes par matières

➡ prenom : yvan

➡ adresse : rue Idlld

➡ Matieres :

etudiant.matiere={ } ;
key : nom de la matière
value : les notes

informatique:
[12.3,15.2]

```
{ "Informatique" : [12.3, 15.2] ,  
  "Economie" : [14.5, 12.3] ,  
}
```

Pour ajouter une matière et des notes :

```
etudiant["Matiere"]["Informatique"]["nouv_mat"]=[] ;  
on peut ajouter des notes dans la liste  
etudiant["Matiere"]["Informatique"]["nouv_mat"].push(12.3)
```

Pour extraire les infos :

il suffit de donner le chemin à la donnée

pour avoir la liste des notes de la matière :

```
etudiant["Matiere"]["Informatique"]["nouv_mat"]
```

pour avoir la première note

```
etudiant["Matiere"]["Informatique"]["nouv_mat"][0]
```

Pour aller plus loin

On vient de définir une structure objet qui nous permet de stocker différentes informations ensemble. Maintenant, si on souhaite ajouter des traitements (fonctions/méthodes) dans cette même structure comment faire ?

Quelques fonctions

fonction etat_civil : on souhaite avoir juste l'affichage du nom associé au prénom

```
etudiant["etat_civil"]=function () { return this.nom + " " + this.prenom;}
```

Pour l'utiliser :

etudiant.etat_civil()

le mot clé this permet
d'indiquer une référence sur
l'objet lui même.

On souhaite avoir la liste des matières (uniquement les noms des matières) :

Object.keys(etudiant.matieres)

On souhaite la moyenne pour chaque matière :

etudiant.moy_matieres() doit retourner la liste des matières et la moyenne associée

Calcul de la moyenne par matière


**On réalise la fonction à l'extérieur de l'objet :
il faut parcourir l'ensemble des matières de l'étudiant et
faire la somme que l'on divise par le nombre de notes ?**

```
var result={} ;  
Object.keys(etudiant.matieres).forEach(  
  e=>{  
    result[e]=etudiant.matieres[e].reduce((p,ee)=>p+=ee,0)/  
    etudiant.matieres[e].length;  
  }  
);
```

Calcul de la moyenne par matière

A l'intérieur de l'objet


```
etudiant.moy_mat=function() {  
  var result={} ;  
  Object.keys(this.matieres).forEach(e=>{  
    result[e]=this.matieres[e].reduce((p,ee)=>p+=ee,0)/this.matieres[e].length;  
  });  
  return result ;  
}
```



attention aux affectations

quand c'est des types de base (Number, string) une copie

quand c'est des objets : attention javascript passe la référence



Et si on souhaite étendre ce concept et créer son propre modèle d'objet

==> CLASS

Notion de classe

On peut définir la classe étudiant à partir de laquelle on pourra générer des objets ayant les mêmes attributs et méthodes

Comment ?

A l'aide du mot clé : class

Class (suite)

// getter et setter
// pas nécessaire car les attributs sont directement accessibles

```
class etudiant {
    constructor(Nom, Prenom, DateNaiss) {
        this.nom=Nom ;
        this.prenom=Prenom ;
        this.date_naissance=DateNaiss ;
        this.matières={} ;
    }
    // autres méthodes
    age() {
        return new Date()-this.date_naissance)/1000/60/60/24/365 ;
    }
    calcul_moy_matières() {

    }
    ajoute_matiere(mat) {
        this.matières[mat]=[] ;
    }
    ajoute_note_matiere(note,mat) {
        this.matières[mat].push(note) ;
    }
}
```

Pour utiliser dans votre code :
var etu1=new etudiant("Stroppa","yvan") ;

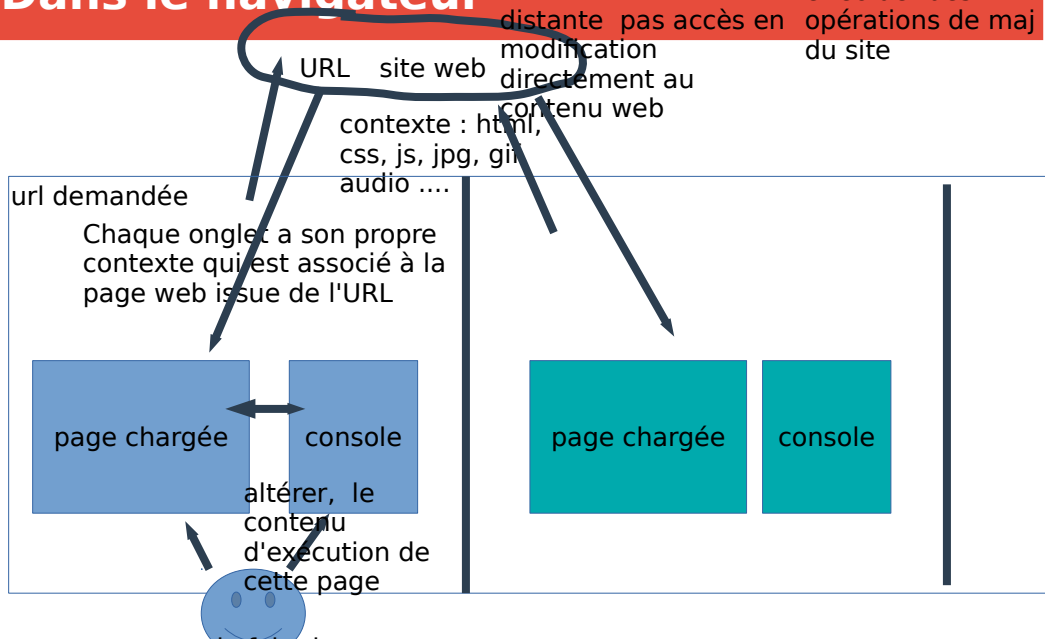
Class avec encapsulation

```
class etudiant {  
    #nom=Nom ;  
    #prenom=Prenom ;  
    #date_naissance=DateNaiss ;  
    #matieres={} ;  
  
    constructor(Nom, Prenom, DateNaiss) {  
        this.#nom=Nom ;  
        this.#prenom=Prenom ;  
        this.#date_naissance=DateNaiss ;  
        this.#matieres={} ;  
    }  
    ajoute_matiere(mat) {  
        this.#matieres[mat]=[] ;  
    }  
    ajoute_note_matiere(note,mat) {  
        this.#matieres[mat].push(note) ;  
    }  
}
```

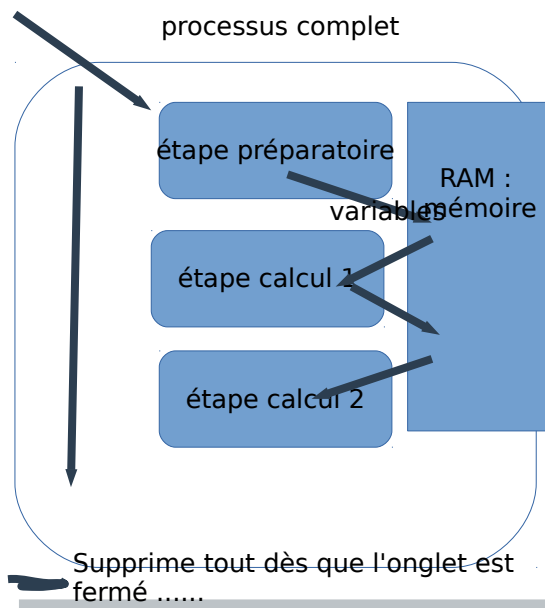
Pour utiliser dans votre code :
var etu1=new etudiant("Stroppa","yvan") ;

Dans le navigateur

nécessite un accès
particulier pour
effectuer des
opérations de maj
du site



traitement en étape



Définition et controle de type

Dans Javascript comme dans Python

Un déclaration implicite du type des variables

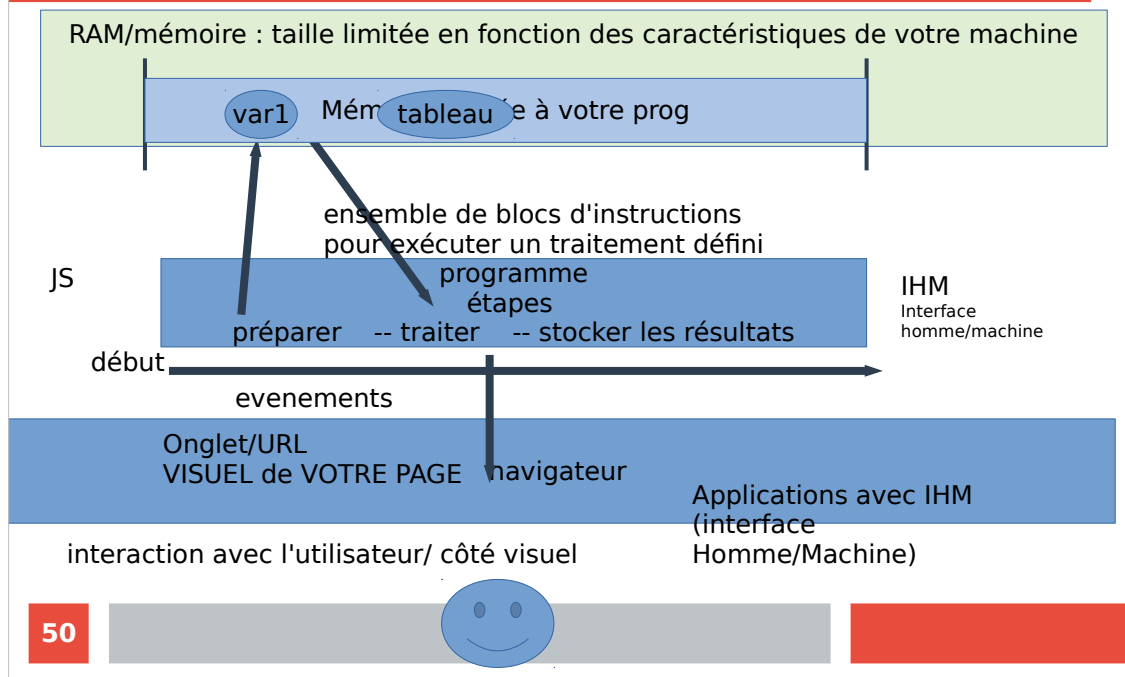
Il n'y a pas de contrôle implicite des types

Dans d'autres langages de type Java, C, C++, typescript

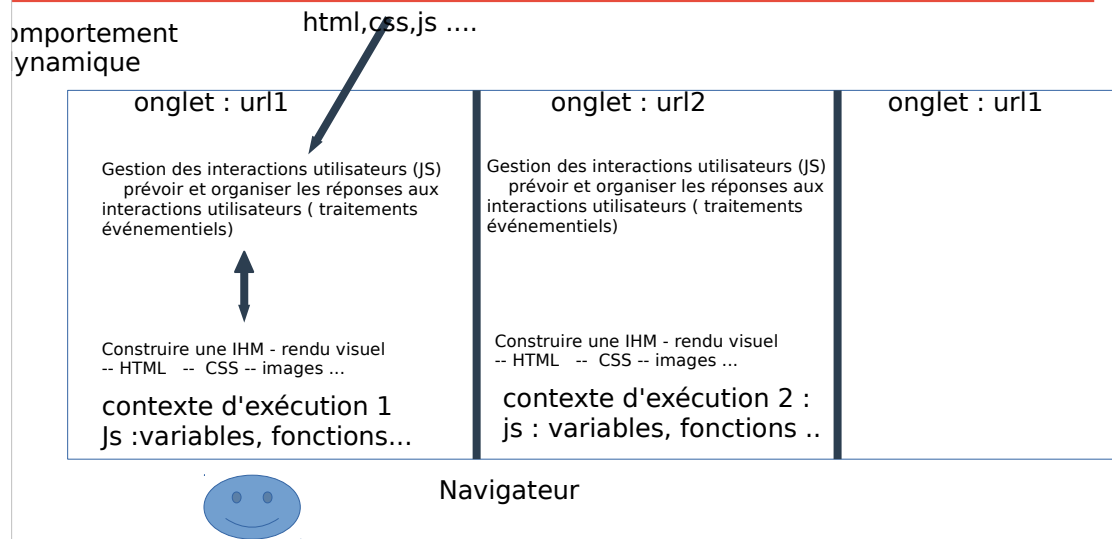
Déclaration explicite des types `public int age ;`

Contrôle des types au moment des affectations

Langage informatique : Variable



contexte technique de votre application web



structure d'une page HTML

<!DOCTYPE html>

permet d'indiquer au navigateur la version html utilisée (version 5)

<html>

<head>

<!-- déclarations de meta-données utilisés par les moteurs de recherche -->

<!-- inclusions de fichiers css, js --> <script src="http..."></script>

<!-- blocs style ou js -->

<style> </style>

<script> </script>

description des tags issue de XML

</head>

<body>

<h1> salut à tous </h1>

</body>

</html>

Mécanisme de type


**Déclaration d'une variable : pas de type explicite
à l'aide de trois mots clés
var ou let ou const**

**Lors de l'affectation : définit un contenu
le contexte associe à la variable un type
(String, Number, Date, Array , Object)
Chaque type == définit un Objet (une classe)
qui possède des attributs et des fonctions
(méthodes prédéfinies)**

Différents types d'application

application en mode batch (lot)

input -- exécute -- output
fichier -- prog -- fichier résultat



Applications avec IHM

+ à gérer l'interaction avec l'utilisateur (prévoir anticiper orienter le comportement de l'utilisateur)

applications en standalone (installer sous système d'exploitation)

applications légères/ en ligne : elles s'appuient sur le navigateur (les applications Online qui sont les plus utilisées)

Déclaration des variables et types de variables

Dans certains langages vous avez obligation de déclarer et de typer vos variables (c,c++,Fortran, Java ...) ce sont les langages fortement typés. (voir des exemples)

Dans d'autres, il y a un typage implicite, ce qui veut dire c'est en fonction de la valeur que l'on stocke que le système associe le type à votre variable (e : python, javascript, R, Matlab ...)

Structure de variables

Construction

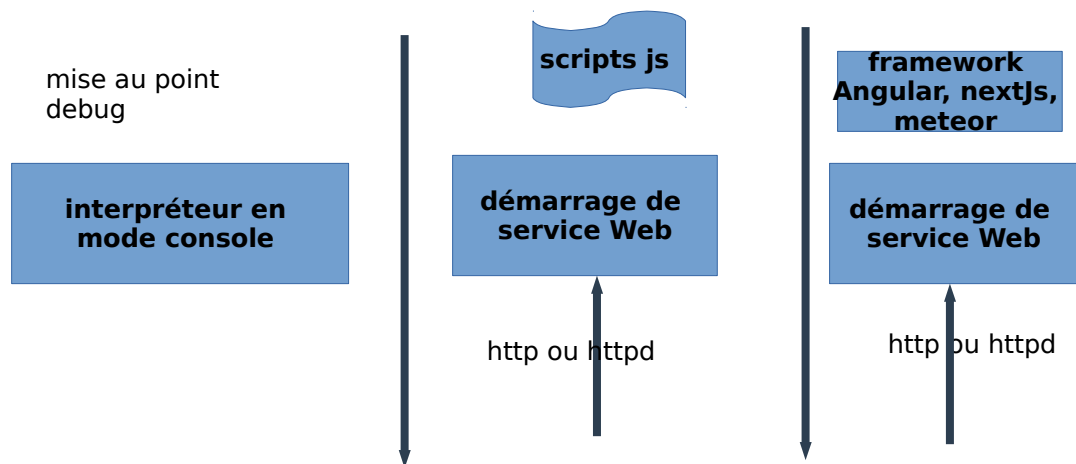
Etre capable de les définir (de les construire)

Manipulation

Etre capable d'extraire les informations

Etre capable de les modifier ou de la compléter

NodeJs : environnement tout JS



Blocs d'instructions

affichage dans la console (afficher différentes valeurs dans la console)

blocs d'instructions conditionnels

condition : si valeur faire	if (condition1 && condition2) {	
	// deux conditions sont vraies	ET
if (condition) {	}	
// vraie		
} else {		
// fausse	if (condition1 condition2) {	
	// si une des deux	OU
}	}	
html : attention aux contrôles des service web		
1^{er} congrès		
js : petite erreur -- effet important		
if (IIII) {		
}		

blocs conditionnels

notion de switch (contenu)

```
switch (contenu) {  
    case "Oranges" :  
        //instructions pour cas 1 ;  
        break ;  
    case "Pommes" :  
        //instructions pour cas 2 ;  
        break ;  
    case "Poire":  
        //instructions pour cas 3 ;  
        break ;  
    default :  
        break ;  
}
```

```
if (contenu==1) {  
    // bloc instruction 1  
}  
else if (contenu==2) {  
    // bloc instruction 2  
}  
else if (contenu==3) {  
    // bloc instruction 3  
}  
else {  
    // bloc instruction autre  
}
```

blocs itératifs : faire des boucles

bloc de type for

on commence à début et on finit à fin (itération explicite)

```
for (let i=0 ; i < 10 ; i++) {
```

// blocs d'instructions

```
}
```

while (condition) {

```
}
```

condition d'exécution de l'itération

```
for (let i=0 ; i<10;i++)  
  console.log(i) ;
```

i++ ==> i=i+1

```
i=0 ;  
while (i<100) {  
  // instructions  
  console.log(i) ;  
  i=i+1 ;  
}
```

ne pas omettre la condition de sortie sinon boucle infinie

définition de fonction

Factoriser et réutiliser des blocs d'instructions Evite d'avoir de la redondance/duplication de blocs d'instructions

difficilement réutilisation
et modifiable

Instructions
bloc 1
instruction

instruction ...
bloc1
instruction



instructions
bloc1() ;
instruction ...
bloc1() ;

"factoriser"

Définition d'une fonction
function bloc1() {
instruction 1
.....
}

réutilisation -- test unitaire
(dans le cadre de la
validation)

Complément fonction

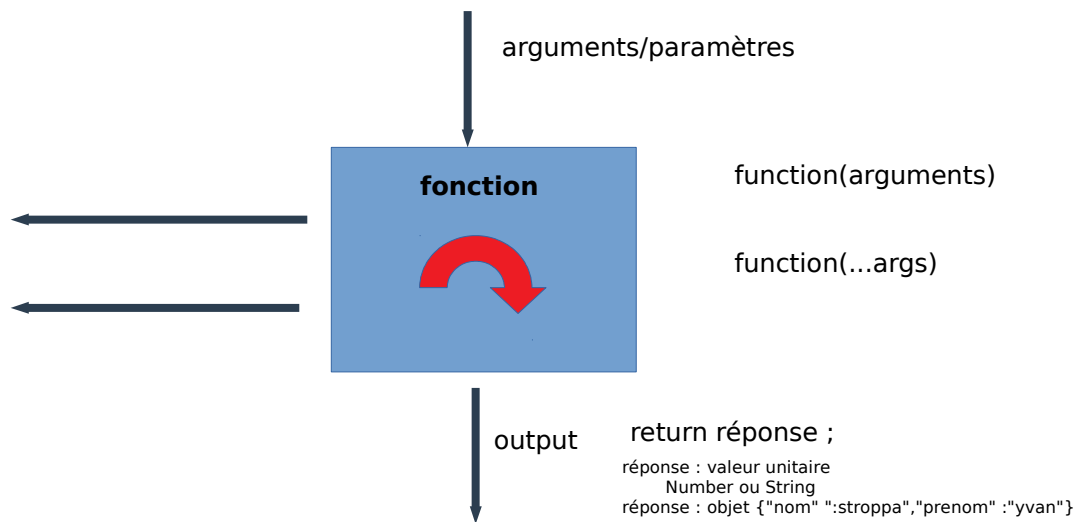
```
// commentaires sur la fonction et les paramètres  
// a : représente telle valeur  
// b :  
// c :
```

```
function g(a,b,c) {  
  if (c==undefined) {  
    console.error("manque une valeur c");  
    c=1 ;  
  }  
  console.log("je suis dans la fonction g",a,b,c);  
}
```

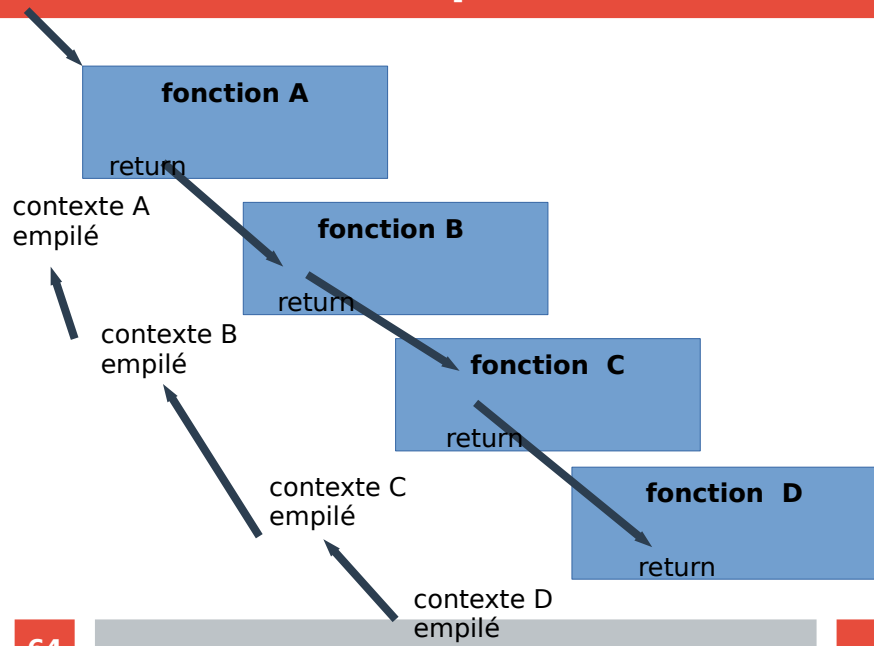
return : retour de la
fonction
sort de la fonction

```
function g(a=0,b=0,c=0) { //valeur par défaut  
  console.log("je suis dans la fonction g",a,b,c);  
}
```

Définition fonction



empilement des appels de fonctions (en mode classique)



Différentes formes de fonction

Forme classique d'une fonction

```
function f(a,b,c) { }
```

fonction sous forme d'expression

```
const f=function(a,b,c) { }
```

fonction fléchée

```
f=function(a) { }
```

```
f=a=>corps de la fonction ;
```

```
f=(a,b) => corps de la fonction ;
```

```
f=(a,b) => { corps de la fonction multi lignes ;}
```

JavaScript et Html

Enjeux et interactions

Comment se passe les échanges entre les deux parties d'une même application

la partie HTML : permet de décrire le visuel (IHM) par l'intermédiaire de tag

`<div>`, `<p>`, ``, ``, `` `<article>` `` `<form>` ``

Chaque tag peut posséder des attributs qui vont permettre de lui apporter des caractéristiques supplémentaires :

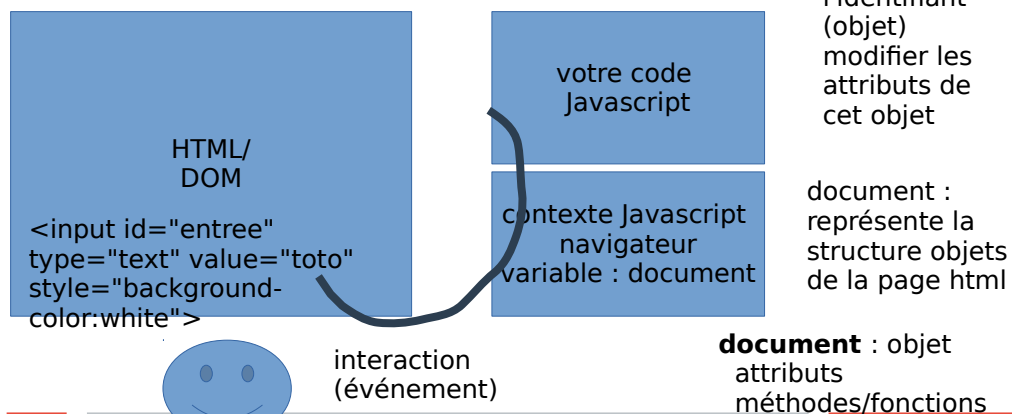
de type style, contenu, identification, ... pour récupérer ensuite à partir de javascript les valeurs on pourra utiliser les méthodes associées à document de type `document.getElementById("identification")`.

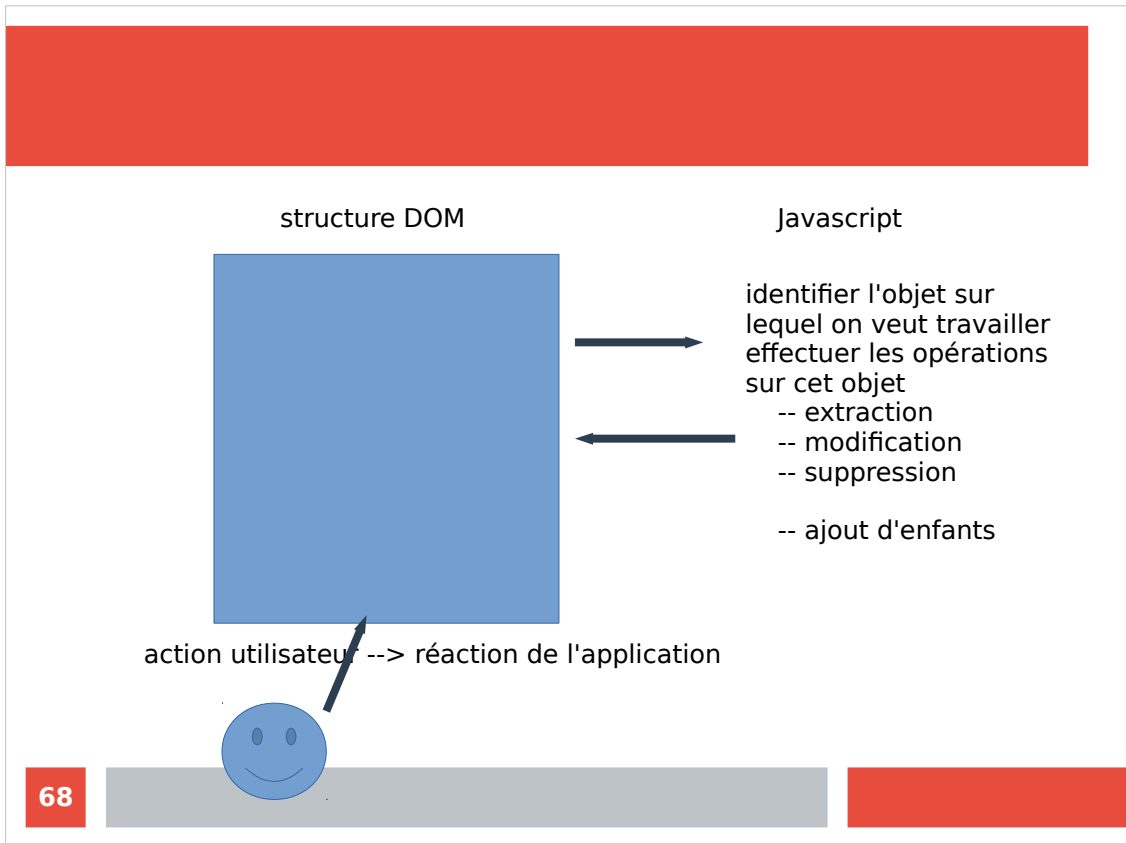
structure HTML

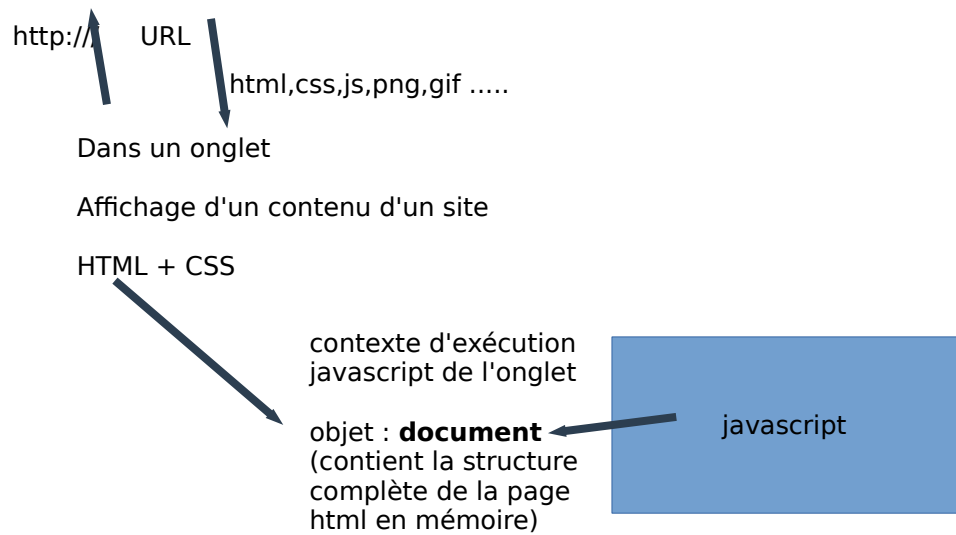
HTML est associé une DOM (Document Object Model) -- s'appuie sur des tags

Les tags <body> <div ...> <p > <input ...>

Structure l'information à partir des tags







Structure de vos pages HTML

<!DOCTYPE html>

<html>

<head>

<style></style>

<link rel="stylesheet" href=".....css" />

<script src="https://.....js"></script>

<script>

// javascript :

// déclaration de variables

// déclaration de fonctions

function toto() {

// blocs d'instructions cond ...

itératif

}

</script>

</head>

<body>

</body>

70

</html>

Téléchargement des
ressources js et css

Organisation de vos fichiers
/index.html ou /index.php

Arborescence de votre site

/index.html

scripts/

fichiers js

styles/

fichiers css

Gestion événementielle du côté HTML

contexte HTML

```
<input class="" type="text" value="dldl" style="... ;" ... onclick=""  
ou onchange="" />
```

```
<input type="button" value="dldl" style="... ;" ...  
onclick="appel_fonction() ;" ou onchange="appel_fonction_1() ;" />
```

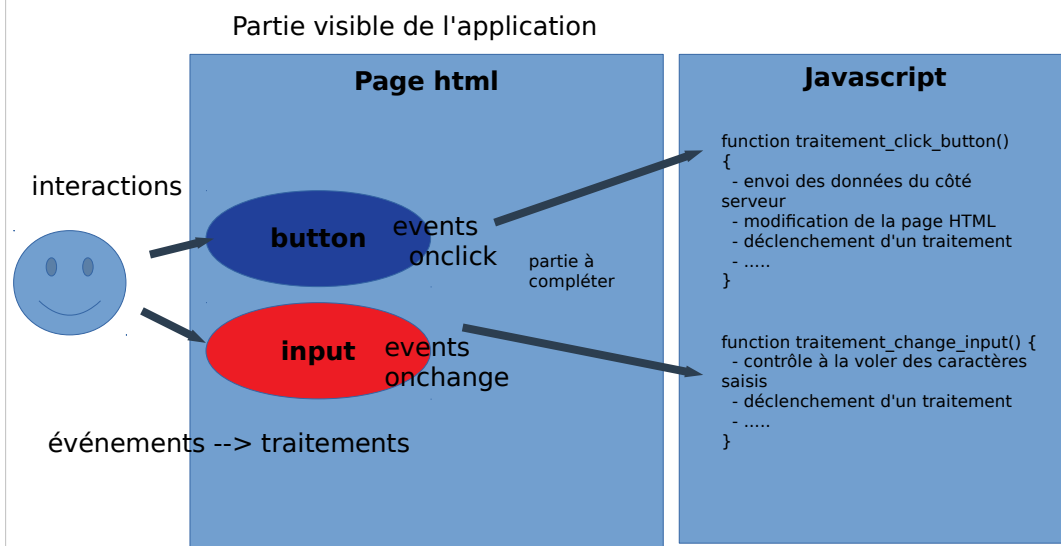
```
<script>
```

```
    function appel_fonction() {
```

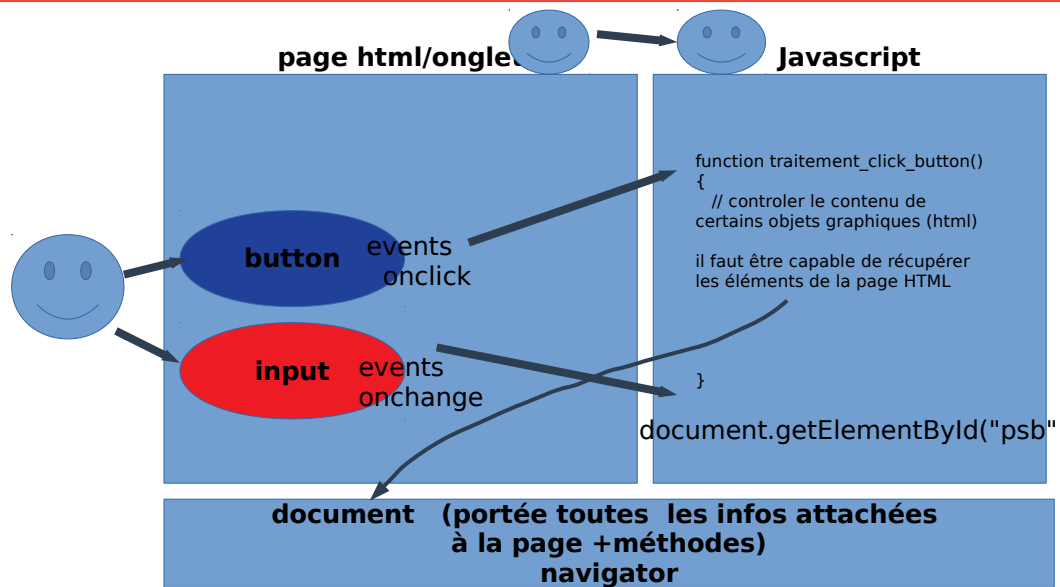
```
    }
```

```
</script>
```


Construction de l'IHM (interface Homme-machine) : tout prévoir



Construction de l'IHM : tout prévoir



Exemple autour d'étudiant/article/produits/entreprise

**Monter une structure qui contient les attributs
d'étudiant et les méthodes associées :**

Exemple autour d'étudiant

Monter une structure qui contient les attributs d'étudiant et les méthodes

calcul de l'age :

différence entre deux dates : retour en ms à convertir en années 1000/60/60/24/365

```
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)}}}
```

Notion de classe

Définir une classe

Intérêt dans nos programmes

notion d'attributs associés à la classe

notion de **static**

attribut

méthode

notion d'attribut protégé

avec le caractère **#**

Notion d'héritage avec le mot clé **extends**

Opérateur new pour instancier un objet issue d'une classe

A travailler sur le document

https://github.com/ystroppa/PSB2020/blob/master/explications_complementaires_javascript.pdf

Annexes

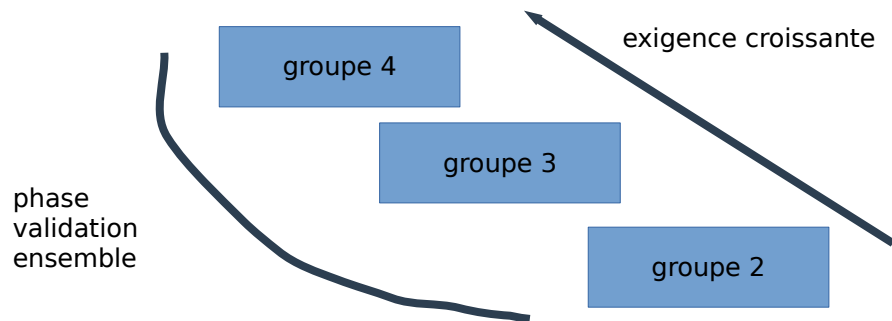
Evaluation des projets

deux phases :

Rapport

Test de la réalisation

quantité / complexité



Réalisation d'un projet

Vue globale -- description la plus précise possible des fonctionnalités

Découper en lots : organisation/répartition

Description fonctionnelle et ses interfaces avec les autres

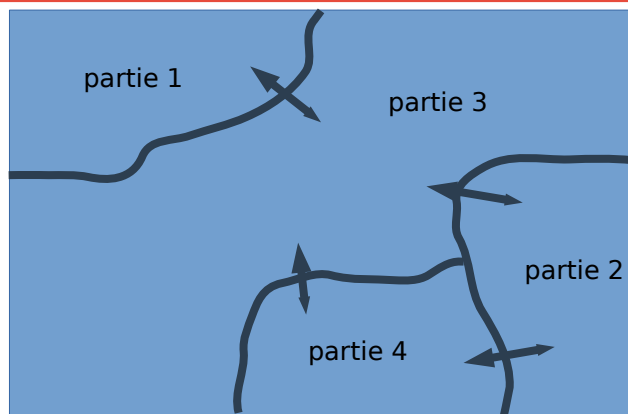
Développer les lots /mettre au point /tester

de façon unitaire (javascript -- modulaire -- fichiers dédiés js) --
vérifier compatibilité pour toutes les navigateurs

Phase d'intégration qui consiste à ramener les lots dans leur contexte cible d'exécution

Tests globaux

problème



Déléguer et répartir
travail en groupe

décomposition en partie /lot
décrire les interfaces entre les
différentes parties