

Web back-end

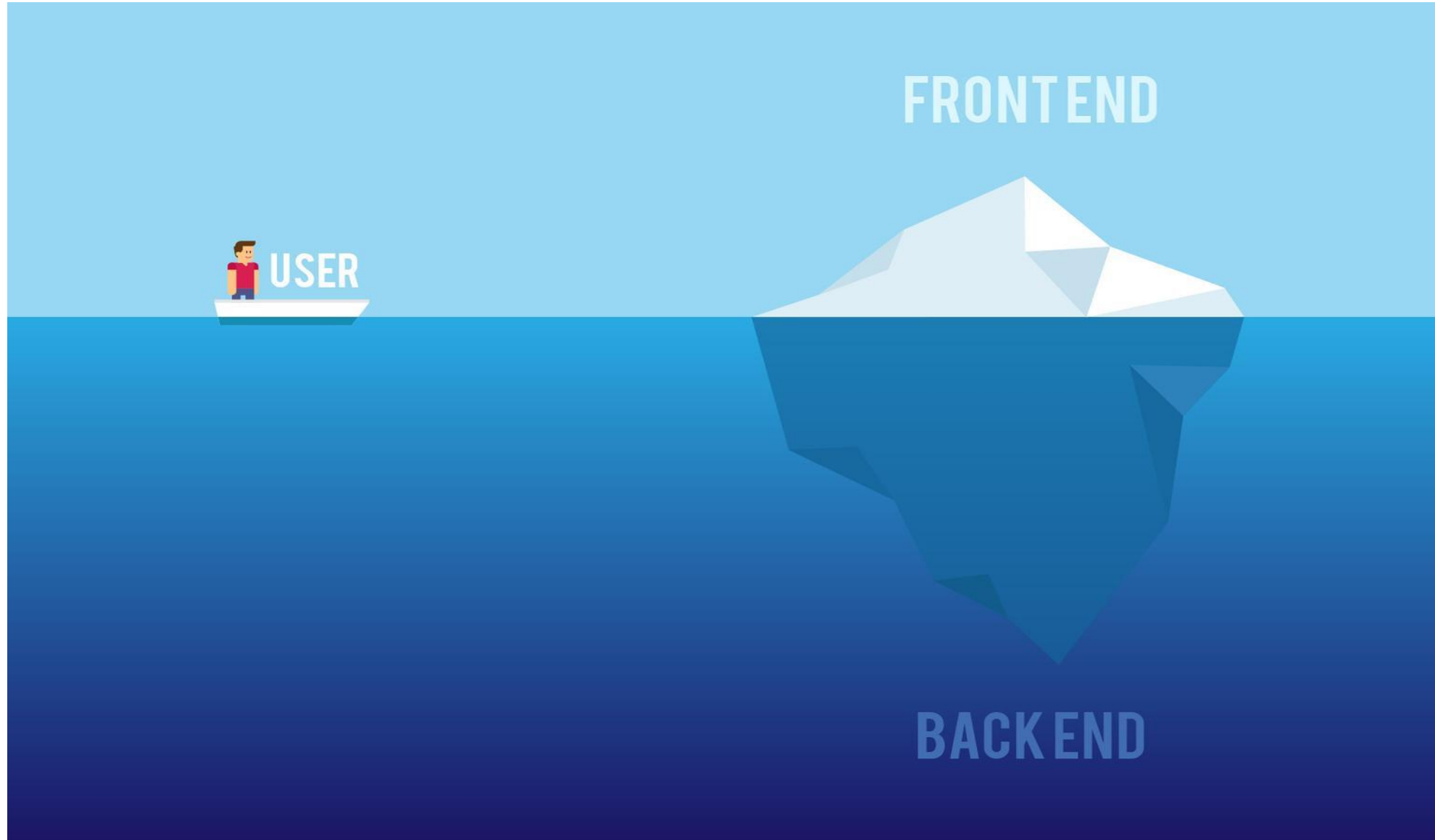
Kaouther TABIA

kaouther.tabia@ensicaen.fr



L'École des INGÉNIEURS Scientifiques

WEB BACK-END VS FRONT-END



WEB BACK-END VS FRONT-END

Le web Front-end:

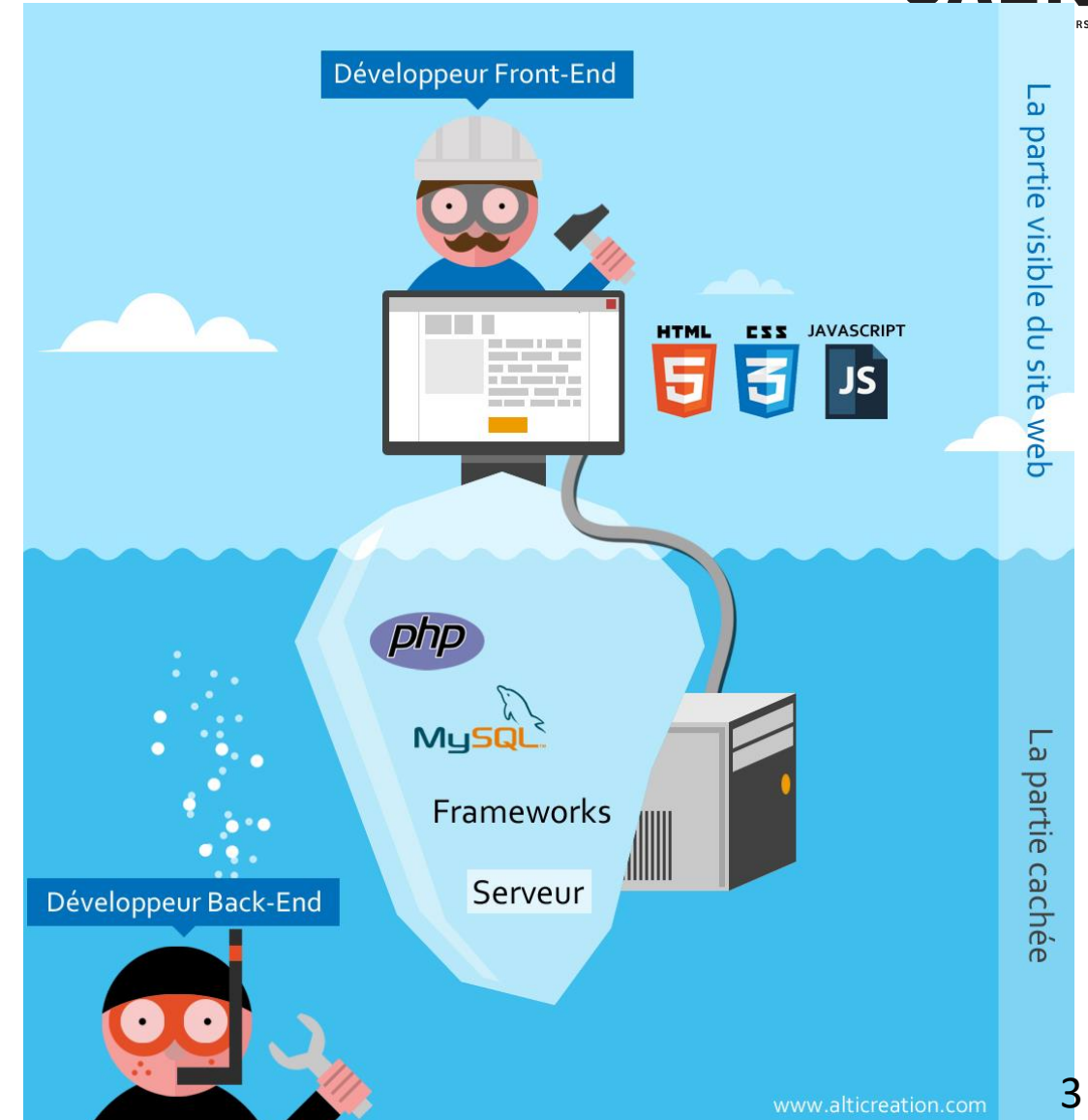
- Les éléments du site que l'on voit à l'écran et avec lesquels on peut interagir.
- Ces éléments sont composés de HTML, CSS et de Javascript, contrôlés par le navigateur web de l'utilisateur.

Le web Back-end: (la partie immergée de l'iceberg)

- Elle est invisible pour les visiteurs
- Sans elle, le site web reste une coquille vide
- On peut décomposer le Back-End en trois parties essentielles :
 - Un **serveur** (ou hébergement web)
 - Une **application** (en l'occurrence le site web)
 - Une **base de données** (ou l'on stocke les données de l'application)

Le **serveur** est comme un disque dur accessible 24 heures sur 24, sur lequel les pages du site web sont enregistrées.

2021/2022



- Comment modifier le contenu d'une page HTML automatiquement (dynamique) ?

- En HTML :

```
<HTML>
```

```
<HEAD><TITLE> page statique </TITLE></HEAD>
```

```
<BODY>
```

```
Nous sommes le 17/01/2022
```

```
</BODY>
```

```
</HTML>
```

- Solution :
 - PHP
- *PHP: Hypertext Preprocessor*
- Langage de script généraliste et Open Source
- Conçu pour le développement web
- Avantages:
 - Simple
 - Fonctionnalités avancées
 - Utilisable sur la majorité des OS

- Compter le nombre de visiteurs
 - Connectés sur votre site
 - Connectés sur votre site actuellement
 - ...
- Gestion des membres
 - Vérification des mots de passe
 - Gestion des accès
 - Forum
 - Envoyer des mails
 - ...

- Gestion des bases de données
 - MySQL
 - Oracle
 - SQLite
 - ...
- Gestion d'annuaires
- Manipulations des URL
- Génération et « parsing » de fichiers XML
- Exécution de commandes Shell
- ...

- Déroulement:
 - Demande une page PHP
 - Serveur web exécute le code de la page
 - Interpréteur
 - Exécution du code
 - Le serveur renvoie le résultat de l'exécution
 - Affichage du résultat



Client	Serveur
<ul style="list-style-type: none">• Ne voit pas le code PHP• Seul le résultat de l'exécution est visible	<ul style="list-style-type: none">• Exécute le code• Renvoie le résultat

- Exemple:

*.php

```
<HTML>
<HEAD><TITLE>Page dynamique</TITLE></HEAD>
<BODY>
<?php echo ( "Nous sommes le " ); echo ( date ( " d/m/Y" ) ); ?>
</BODY>
</HTML>
```

Résultat html

```
<HTML>
<HEAD><TITLE>Page dynamique</TITLE></HEAD>
<BODY>
→ Nous sommes le 12 janvier 2023
</BODY>
</HTML>
```

- Comment intégrer du PHP?

<?php **/* code */** **?>**

- Où intégrer le code PHP ?
 - Dans le code HTML (même dans l'en-tête)

```
<body>
  <h2> Page de test </h2>
  <p> Cette page contient du code HTML avec des balises PHP.  <br />
    <?php /* Insérer du code PHP ici */ ?>
  </p>
</body>
```

- Mon premier programme
 - Afficher un message

`<?php echo "Ceci est du texte"; ?>`

```
<html>
  <head>
    <title>Notre première instruction : echo</title>
  </head>
  <body>
    <h2>Affichage de texte avec PHP</h2>
    <p>message HTML.<br />
    <?php echo " message PHP."; ?>
  </p>
  </body>
</html>
```

- Les variables:
 - Les différents types
 - **Les chaînes de caractères (string)**
 - Entre guillemets (interprété) ou apostrophes (non interprété)
 - **Les nombres entiers (int)**
 - **Les nombres décimaux (float)**
 - **Les booléens (bool)**

```
$traitees = '****'  
echo 'Les variables ne seront pas $traitees $ici';  
// Affiche : Les variables ne seront pas $traitees $ici
```

- Variables :
 - Préfixés par un \$
 - Sensible au majuscule
 - **Pas de déclaration, typage implicite**
 - \$var=54 → entier
 - \$var="toto" → chaîne de caractères

```
<?php  
$age= 17; // int  
$nom="toto"; // string  
$poids = 57.3; // float  
?>
```

- Variables
 - Affichage:



- Variables
 - Calculs simples

```
<?php
    $nombre = 2 + 4;
    $nombre = 5 - 1;
    $nombre = 3 * 5;
    $nombre = 10 / 2;
    $nombre = 3 * 5 + 1;
    $nombre = (1 + 2) * 2;
    echo "$nombre";
    $resultat = ($nombre + 5) * $nombre;
    echo "$resultat";
    $modulo= 10 % 5;
?>
```

6

66

- Variables

- fonctions de test:

- | | | |
|-----------------------------|---|-------------------------------------|
| • isset (\$var) | → | renvoie true si \$var existe |
| • unset (\$var) | → | détruit \$var |
| • is_integer (\$var) | → | renvoie true si \$var est un entier |
| • is_string (\$var) | → | renvoie true si \$var est un chaîne |
| • ... | | |

- Boucles et structures conditionnelles

```
<?php
    $monsieur= "toto";
    if ($monsieur == "toto"){
        echo "Bonjour monsieur<br />";
        $entrer = "Oui";
    }
    else {
        echo "Vous n'êtes pas autorisé à entrer<br />";
        $entrer = "Non";
    }
    echo "Autorisation: $entrer";
?>
```

- Boucles et conditionnelles

```
<?php
$monsieur= "toto";
switch ($monsieur) {
    case "toto":
        echo "OK";
        break;
    case "tata":
        echo "Ok";
        break;
    default:
        echo "NO";
}
?>
```

- Boucles et conditionnelles

```
<?php
$nb_lignes = 1;
while ($nb_lignes <= 10){
    echo 'PHP' . $nb_lignes '<br />';
    $nb_lignes++;
}
?>
```

```
<?php
for ($nb_lignes = 1; $nb_lignes <= 100; $nb_lignes++){
    echo 'PHP' . $nb_lignes '<br />';
}
?>
```

- Fonctions

```
<?php
    function addition($op1,$op2){
        $somme=$op1+$op2;
        return $somme
    }
    $rest= addition(4,2);
?>
```

- Fonctions prédéfinies
 - « **strlen** » longueur d'une chaîne
 - « **str_replace** » rechercher et remplacer
 - « **strtolower** » écrire en minuscules
 - « **date** » récupérer la date

```
<?php
```

```
$mot= 'Bonjour ';  
$longueur = strlen($mot);  
$remplace= str_replace('t', 'm', 'tata');  
echo $remplace;
```

mama

```
$chaine = strtolower($mot);  
echo $chaine;
```

bonjour

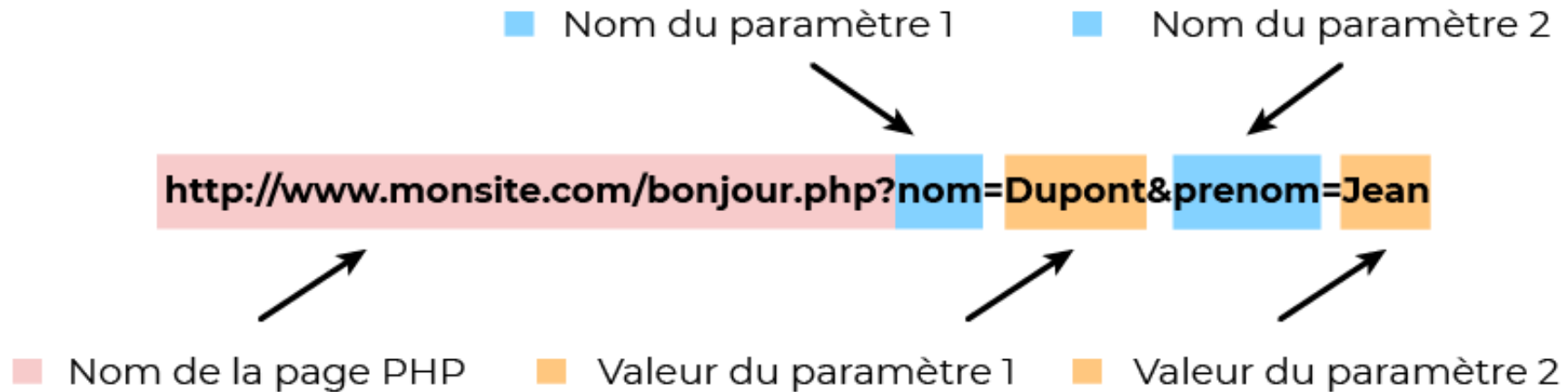
```
$jour = date('d');  
$mois = date('m');  
$annee = date('Y');  
echo $jour;
```

12

```
?>
```

- **Transmission de données**
 - Via l'URL
 - Via un formulaire

- Transmission de données
 - Via l'URL



- Transmission de données
 - Via l'URL

Envoie des données



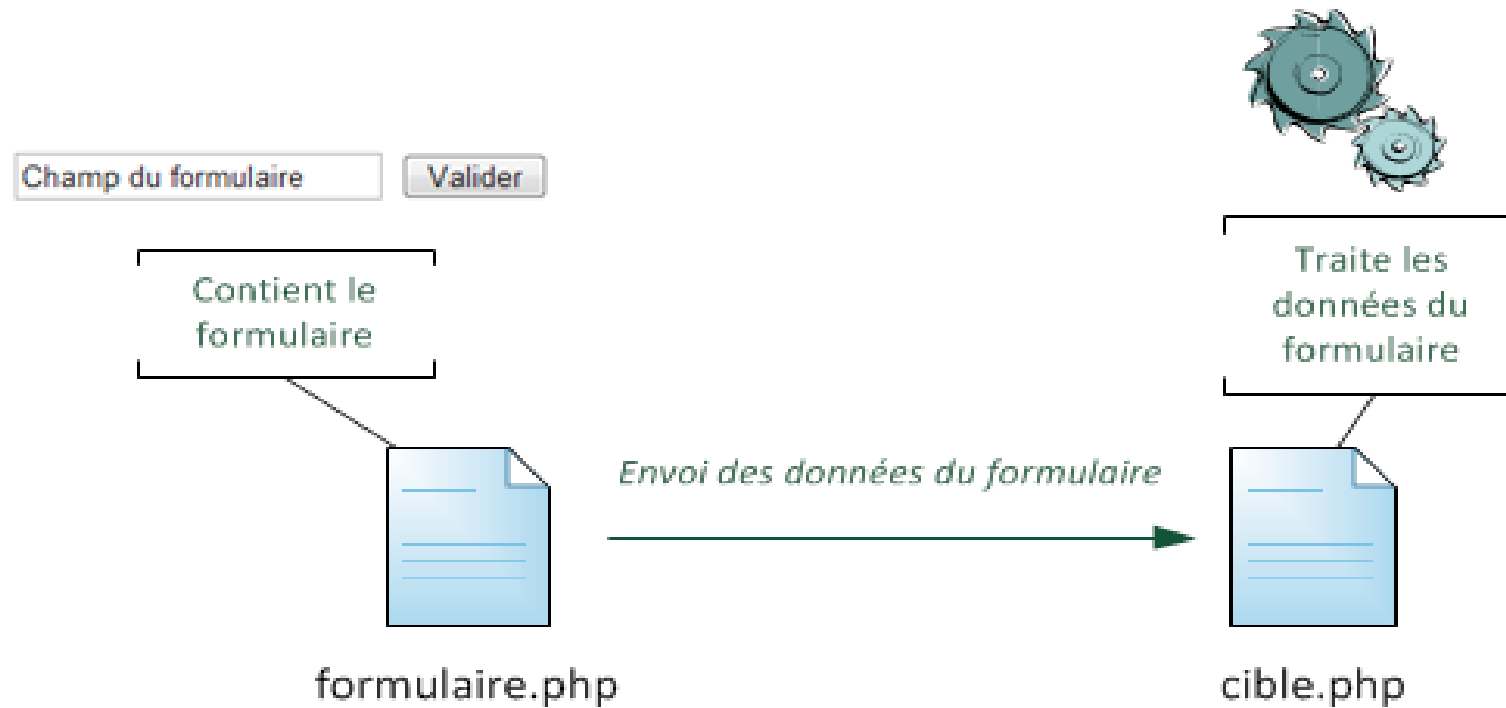
```
<a href="bonjour.php?nom=Dupont&prenom=Jean">Hello</a>
```

Réception

```
<p>Bonjour <?php echo $_GET['prenom'] . ' ' . $_GET['nom']; ?> !</p>
```

Mais attention à la sécurisation des données

- Transmission de données
 - Via les formulaires



- HTML: création d'un formulaire
 - Balise **form**: balise principale du formulaire
 - Récupération des données et envoi
- Deux méthodes
 - **get**: limitée en nombre de caractères (256 caractères) . Transmission via l'URL. Par défaut si la méthode d'envoi n'est pas spécifiée
 - **post**: permet un envoi avec un grand nombre de données (la plus utilisée). Ne transmet pas les données via l'URL.

- Récupération des données et envoi (*get* ou *post*)
 - **action**: indique le fichier qui va recevoir les données extraites.

```
<p>Début du formulaire</p>
<form method="post" action="traitement.php">
    <p>Formulaire</p>
    <input type="text" name="prenom" />
</form>
<p>Fin du formulaire</p>
```

- Réception des données
 - **action**: indique le fichier qui va recevoir les données extraites.
 - \$_GET
 - \$_POST

```
<?php  
    echo $_GET['prenom'];  
?>
```

```
<?php  
    echo $_POST['prenom'];  
?>
```

LECTURE D'UN FICHIER AVEC PHP

- `file_exists("url fichier")` pour vérifier si le fichier texte existe ou pas
- `fopen("url fichier", "Mode ouverture");`
- Les différents modes d'ouverture: (`r`, `r+`, `w`, `w+`,...))
- `fread(file,size)` pour lire d'un fichier

```
<?php
if(file_exists("fichier.txt"))
{
    $file = fopen( "fichier.txt", "r" );
    $content =fread($file, filesize('fichier.txt')) ;
    fclose($file);
}
?>
```

ECRIRE DANS UN FICHIER AVEC PHP

Ecrire dans un fichier

```
<?php  
  
$file = fopen("fichier.txt", "w");  
  
$contenu="Hello";  
  
fwrite($file,$contenu);  
  
fclose($file);  
  
?>
```

Tout le contenu initialement présent est écrasé et remplacé par le nouveau

```
<?php  
  
$contenu="Hello";  
  
file_put_contents("fichier.txt",  
$contenu);  
  
?>
```

HTML DANS DU CODE PHP

- Possibilité de mettre du code HTML
 - dans `<?php ?>`
 - À afficher dans un echo

- Exemple

```
<?php  
    $var='test.html';  
    echo "<a href= ' $var ' > lien </a>";  
?>
```

- Besoin d'un éditeur de texte (Visual code, ATOM,)
- Besoin d'un navigateur
- Besoin
 - d'un serveur web comme Apache,
 - PHP (interpréteur)
- **Utiliser easyPHP** : (un pack sous windows contient les 3 produits incontournables de la scène PHP)
 - Le serveur Web Apache
 - Le moteur de scripts PHP4
 - La base de données MySQL
 - Un outil de gestion de base de donnée graphique, Phpmyadmin

COOKIES : DÉFINITION

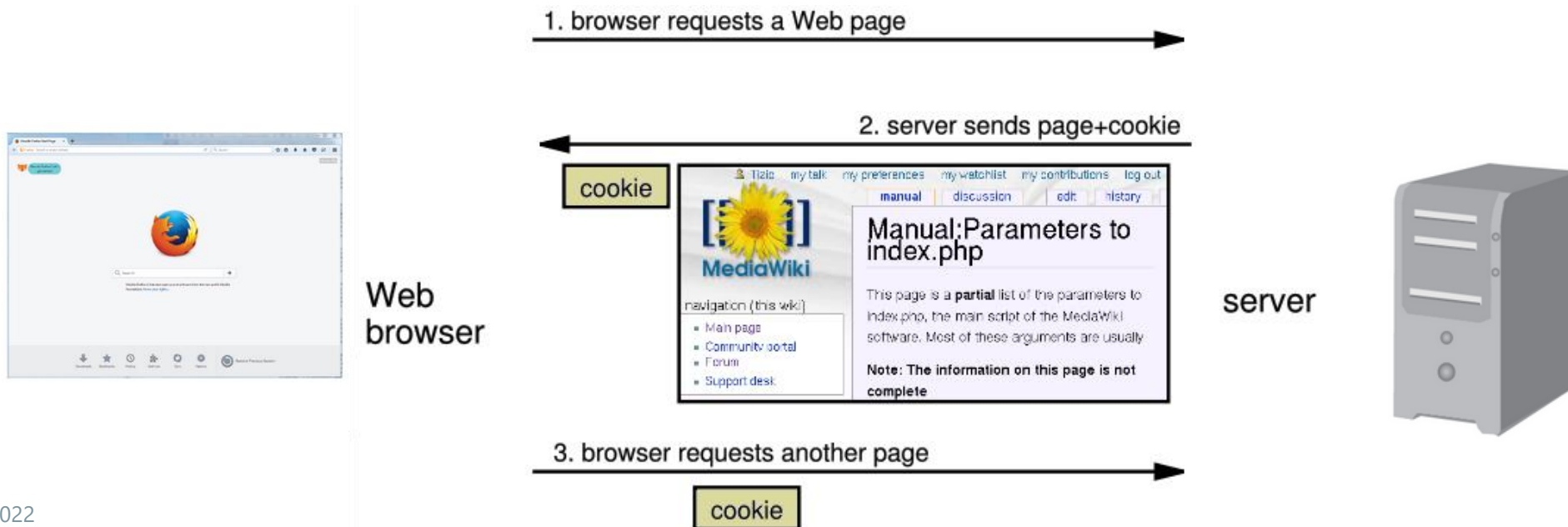
- Ce ne sont pas des virus, juste de petits fichiers texte qui permettent de retenir des informations.
- déposé sur le disque dur de l'internaute par le serveur du site visité ou par un serveur tiers (régie publicitaire, service de web analytique, etc.)

Le cookie est une suite d'informations, généralement de petite taille et identifié par un **nom**, qui peut être transmis à votre navigateur par un site web sur lequel vous vous connectez. Votre navigateur web le conservera pendant une certaine **durée**, et le renverra au serveur web chaque fois que vous vous y reconnecterez.

CNIL

COOKIES

- Permettent de stocker une information **chez le client**
 1. Le client demande une page
 2. Le serveur renvoi la page+ un (ou plusieurs)cookies
 3. Le client revoie automatiquement les cookies au serveur lors de connexions futures



UTILISATION DES COOKIES

- A quoi cela sert il ?
 1. Identifier une personne
 2. Implémenter la notion de panier
 3. personnaliser des applications (igoogle, Wikipedia,...)
 4. Calculer des profils d'utilisateurs (pub ciblés) ou calculer des statistiques sur des pages

Exemple:

Vous allez fêter votre anniversaire cet été et, voilà que sur votre fil d'actualité Facebook, on vous propose d'acheter un tee-shirt où est inscrit «Les meilleurs sont nés en juillet»

« Une bonne publicité, c'est le bon message livré à la bonne personne au bon moment.»

COOKIES

- Un cookie possède :
 1. Un **nom** (name)
 2. Une **valeur** (value)
 3. Une date **d'expiration** (date) au delà de laquelle il sera effacé chez le client. Une date dans le passé efface un cookie
 4. Un **domaine**(domain) auquel il sera transmis
 5. Un **chemin** (path) indiquant à partir d'où il pourra être lue

Set-Cookie: name=value; expires=date; path=/; domain=www.greyc.ensicaen.fr

COOKIES

- Par défaut la date d'expiration correspond à la fin de la session.
- Le format de la date est :

Wdy, DD-Mon-YYYY HH:MM:SS GMT

Exemple: Mercredi 20 août 2008 à 21h05:

Wed, 20-08-2008 21:05:00 GMT

- Par défaut le nom de domaine & chemin sont tirés de l'url. Par sécurité un cookie est refusé si le nom du serveur n'apparaît pas dans l'url.

COOKIES

- Positionné par la fonction **setcookie**

Syntaxe:

`setcookie(name, value, expire, path, domain);`

- Le positionnement du cookie doit apparaître dans l'entête donc **avant** toute balise html (même avant tout espace blanc et/ou ligne blanche).

COOKIES EN PHP: EXEMPLE

```
<?php
nom
valeur
Temps avant expiration
Le temps à cet instant (time()) + un an (3600 seconde *24 heures* 365 Jours)
setcookie("seen", "yes", time()+3600*24*365);
?>

<html>
  <head><title>Hello</title></head>
  <body><h1>
    <?
    if(isset($_COOKIE["seen"]))
      echo "Je vous ai déjà vu quelque part";
    else
      echo "Hello ";
    ?>
  </h1>
</body>
</html>
```

CONNEXION À UNE BASE DE DONNÉES

PDO (PHP Data Objects)

- Définit une excellente interface pour accéder à une base de données depuis PHP
- Une Interface qui va nous permettre de travailler, de manipuler nos bases de données quel que soit le système de gestion de base de données que vous utilisez
- Les connexions sont établies en créant des instances de la classe de base de PDO.
- Le constructeur accepte des paramètres pour spécifier la source de la base de données et optionnellement, le nom d'utilisateur et le mot de passe (s'il y en a un)

EXEMPLE : CONNEXION À POSTEGRESQL

```
<?php
```

```
$dbh = new PDO('pgsql:host=postgres;dbname=test', $user, $pass);
```

```
?>
```

Objet de connexion
(une instance de la classe PDO)

Nom de moteur de *base de données*
Mysql / Postgresql / Oracl / SQLite

Nom de serveur

Nom de la base de données

Nom d'utilisateur

Mot de passe



S'il y a des erreurs de connexion, un objet PDO Exception est lancé.
Vous pouvez attraper cette exception si vous voulez gérer cette erreur

EXEMPLE : CONNEXION À POSTEGRESQL AVEC GESTION DES ERREURS DE CONNEXION

```
<?php
try
{
    $dbh = new PDO('pgsql:host= postgres;dbname=$db',$user, $pass);
}
catch(PDOException $e)
{
    echo $e->getMessage();
}
?>
```

- Capable de gérer n'importe quelles commandes SQL (INSERT, UPDATE, DELETE, SELECT)
- PDO utilise deux fonctions différentes:
 - La fonction **exec**, pour exécuter les commandes SQL (INSERT, UPDATE, DELETE)
 - La fonction **query** pour exécuter la commande SQL SELECT

EXEMPLE : CONNEXION À POSTGRESQL AVEC GESTION DES ERREURS DE CONNEXION

<?php

try

```
{  
    $dbh = new PDO('pgsql:host=localhost;dbname=test', $user, $pass);  
  
    foreach($dbh->query('SELECT * from FOO') as $row)  
    {  
        print_r($row);  
    }  
  
    $dbh = null;  
  
}  
  
catch (PDOException $e)  
{  
    print "Erreur !: " . $e->getMessage() . "<br/>";  
  
    die();  
}
```

RÉSULTAT

| ID | nom | prenom |
|----|-----|--------|
| 1 | aa | bb |
| 2 | cc | dd |

Array ([ID] => 1 [0] => 1 [nom] => aa [1] => aa [prenom] => bb [2] => bb)

Array ([ID] => 2 [0] => 2 [nom] => cc [1] => cc [prenom] => dd [2] => dd)

EXEMPLE D'UTILISATION DE LA FONCTION EXEC AVEC LA COMMANDE INSERT

<?php

```
$dbh = new PDO('pgsql:host=localhost;dbname=test', $user, $pass);
```

```
$dbh -> exec("INSERT INTO membres(champ_login,champ_mdp) VALUES ('login','mot_de_passe')");
```

?>

EXEMPLE D'UTILISATION DE LA FONCTION EXEC AVEC LA COMMANDE DELETE

<?php

```
$dbh = new PDO('pgsql:host=localhost;dbname=test', $user, $pass);
```

```
$dbh -> exec("DELETE FROM membres WHERE champ_id_membre='id_membre' ");
```

?>

EXEMPLE D'UTILISATION DE LA FONCTION QUERY AVEC LA COMMANDE SELECT

```
<?php
```

```
$dbh = new PDO('pgsql:host=localhost;dbname=test', $user, $pass);
```

```
$dbh -> query("SELECT membre FROM membres WHERE champ_id_membre=  
'id_membre'");
```

```
?>
```


- Le cas de la fonction **query** est un peu particulier, car la commande retourne un résultat,
- Cette fonction s'accompagne de la fonction **fetch** permettant de parcourir les lignes du résultat.
- En effet la fonction **fetch** lit les résultats **ligne à ligne**, donc en l'utilisant dans une boucle on peut parcourir l'ensemble des lignes résultats.
- On peut spécifier la méthode de récupération des données résultat, ainsi on peut indiquer si l'on souhaite récupérer les données sous forme d'objet, ou bien de tableaux soit à indice numérique soit à indice associatif, en utilisant la fonction **setFetchMode**

- La fonction **setFetchMode** prend un paramètre pouvant prendre les valeurs suivantes:
 - **PDO::FETCH_OBJ** la ligne résultat est récupérée sous forme d'un objet
 - **PDO::FETCH_ASSOC** la ligne résultat est récupérée sous forme d'un tableau à indice associatif
 - **PDO::FETCH_NUM** la ligne résultat est récupérée sous forme d'un tableau à indice numérique
 - **PDO::FETCH_BOTH** la ligne résultat est récupérée sous forme d'un tableau à indice numérique soit associatif


EXEMPLE UTILISATION DE LA FONCTION FETCH ET DE LA FONCTION SETFETCHMODE

```
<?php
try {
    $dbh = new PDO('pgsql:host=localhost;dbname=test', $user, $pass);

    $stmt = $dbh->query("SELECT nom, prenom FROM personne where nom LIKE'%a%' ");

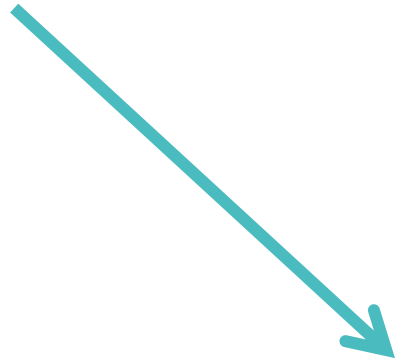
    $stmt->setFetchMode(PDO::FETCH_NUM);

    while ($row = $stmt->fetch())
    {
        print $row[0]."\t".$row[1] . "\n";
    }
}
catch (PDOException $e)
{
    print $e->getMessage();
}
```



aa bb

```
while ($row = $stmt->fetch())  
{  
    print_r ($row);  
}
```



Array ([0] => aa [1] => bb)


SANS FETCH: C'EST POSSIBLE AUSSI

```
<?php
try {
    $dbh = new PDO('pgsql:host=localhost;dbname=test', $user, $pass);

    $stmt = $dbh->query("SELECT nom, prenom FROM personne where nom LIKE'%a%' ");

    $stmt->setFetchMode(PDO::FETCH_NUM);

    foreach ($stmt as $row)
    {
        print $row[0]."\t".$row[1] . "\n";
    }
} catch (PDOException $e)
{
    print $e->getMessage();
}
?>
```



aa bb

CREATION D'UN TABLE DANS UNE BASE DE DONNÉES

```
try {  
  
    $dbh = new PDO ('pgsql:host=localhost;dbname=test', $user, $pass);  
  
    $res = $dbh->exec('CREATE TABLE Users ( id INT(3) UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
    firstname VARCHAR(50) NOT NULL, lastname VARCHAR(50) NOT NULL)');  
  
}  
  
catch(PDOException $e)  
  
{  
  
    echo $e->getMessage();  
  
}
```

- Lorsque la connexion à la base de données a réussi, une instance de la classe PDO est créée.
- Pour clore la connexion, vous devez détruire l'objet en vous assurant que toutes ses références sont effacées.
- Vous pouvez faire cela en assignant **NULL** à la variable gérant l'objet.
- Si vous ne le faites pas explicitement, PHP fermera automatiquement la connexion lorsque le script arrivera à la fin.

EXEMPLE DE FERMETURE D'UNE CONNEXION

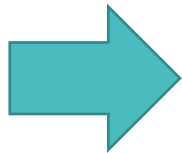
```
<?php
$dbh = new PDO('pgsql:host=localhost;dbname=test', $user, $pass);

// utiliser la connexion ici
$stmt = $dbh->query('SELECT * FROM foo');

// et maintenant, on ferme la connexion
$stmt = null;
$dbh = null;
?>
```


COMMENT PASSER UNE VARIABLE DE PHP À JAVASCRIPT ?

- PHP et JavaScript sont des langages couramment utilisés en même temps sur plusieurs sites web.
- PHP est utilisé pour effectuer des traitements serveurs, et JavaScript des interactions avec l'utilisateur.
- Il arrive donc que vous deviez passer une variable de PHP vers JavaScript



Pour réaliser cela, il existe plusieurs méthodes.

COMMENT PASSER UNE VARIABLE DE PHP À JAVASCRIPT ?

- La méthode la plus simple consiste à afficher grâce à PHP la valeur de la variable directement dans le code JavaScript.
- Il s'agit de la méthode la plus directe.

Exemple

```
<?php
```

```
$variableAPasser="nom";
```

```
?>
```

```
<script>
```

```
var variableRecuperee = <?php echo json_encode($variableAPasser); ?>;
```

```
</script>
```

code des objets ou des tableaux pour le Javascript



COMMENT PASSER UNE VARIABLE DE PHP À JAVASCRIPT ?

- Deuxième méthode consiste à afficher la variable PHP dans le HTML.
- Par exemple, dans un `<input>` caché ou dans une `<div>` puis dans le code JavaScript servant à récupérer la valeur en utilisant le DOM (*Document Object Model*).

Exemple

`<!-- HTML -->`

```
<input type=hidden id=variableAPasser value= <?php echo $variableAPasser; ?>/>
```

`//JavaScript`

```
var variableRecuperee = document.getElementById(variableAPasser).value;
```

COMMENT PASSER UNE VARIABLE DE PHP À JAVASCRIPT ?

- Une autre solution est d'utiliser AJAX.
- Le code JavaScript va envoyer une requête AJAX à un script PHP qui affichera la valeur.
- Grâce à l'événement onLoad qui est appelé lors du chargement du script PHP, on peut récupérer la valeur dans le code JavaScript.
- C'est la méthode la plus propre, car les langages sont alors bien séparés.
- Elle permet également de charger plus vite la page, car l'appel du script PHP peut être effectué dans un deuxième temps (par exemple, lors d'un clic sur un bouton).

- AJAX (**Asynchronous Javascript and Xml**)
 - AJAX n'est pas Javascript.
- est une méthode permettant d'interroger un serveur http a partir d'un navigateur et du langage Javascript.
(back-end)
- modifier partiellement la page web affichée sur le poste client sans avoir à afficher une nouvelle page complète. **(front-end)**

Le but étant de **transmettre** des **données** à un **serveur** afin d'en **recupérer d'autres** de manière asynchrone, sans avoir à recharger l'intégralité de la page, et donc d'augmenter l'interactivité de cette dernière.

- **Avantage**
 - permettre une plus grande réactivité de l'interface par rapport au web classique

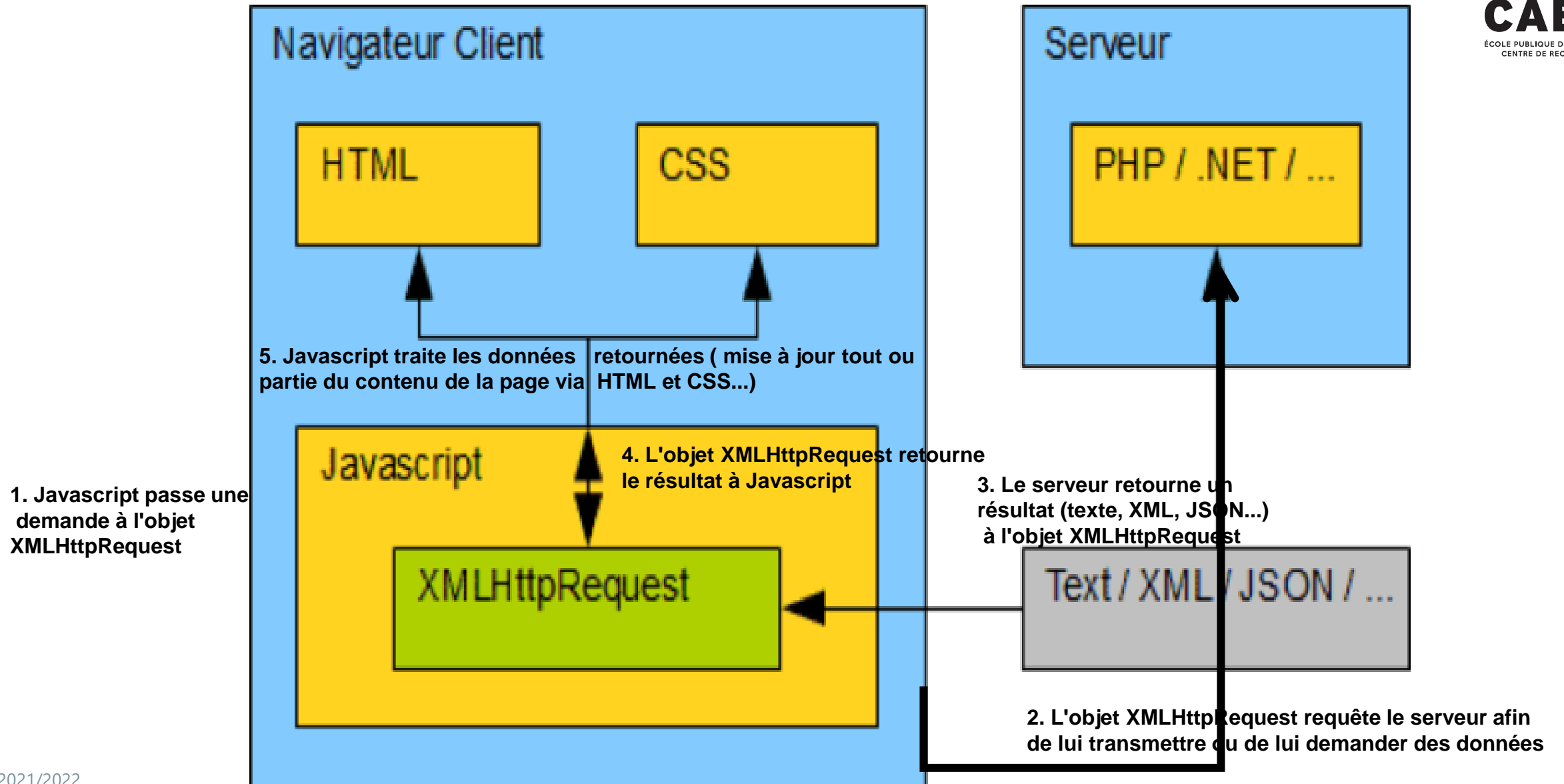
Comme son nom l'indique, la mise en place d'AJAX requiert plusieurs langages :

- **Javascript**, pour les traitements côté client ;
- L'objet ***XMLHttpRequest*** de Javascript, pour dialoguer avec le serveur ;
- **PHP**, ou tout autre langage "*server-side*", pour les traitements côté serveur (notez qu'il est également tout à fait possible d'appeler des fichiers statiques) ;
- **XML et JSON**, qui peuvent servir de formats d'échanges standardisés entre le client (Javascript) et le serveur (PHP...)
 - XML et JSON : sont deux formats de données textuelles, permettant de représenter de l'information structurée.

AJAX est constitué d'une combinaison de technologies existantes. Un schéma classique d'utilisation est le suivant:

1. Un utilisateur exécute une action
2. l'action appelle une fonction Javascript qui envoie une requête à un serveur
3. le serveur traite la requête et renvoie une réponse au format texte ou XML
4. la réponse est récupérée par une fonction Javascript qui va modifier le document HTML en jouant sur les propriétés CSS ou sur le contenu HTML du site en utilisant DOM (**document.getElementById("...").innerHTML**, **document.getElementById("...").value**,...).

AJAX



XMLHTTPREQUEST

- AJAX est basé sur la classe javascript *XMLHttpRequest*.
- Un objet XMLHttpRequest:
 - ouvre une connexion HTTP
 - permet de communiquer via HTTP en GET ou POST
 - gère la réception des réponses du serveur en XML ou texte....

CRÉATION D'UN OBJET XMLHTTPREQUEST

- Il suffit de taper :

<script>

```
var requeteHTTP = new XMLHttpRequest() ;
```

</script>

PRÉPARATION DE LA REQUÊTE

initialisation de la requête

open(String method , String url, asynFlag)

method : GET ou POST

url : url à déclencher (le nom de fichier php)

asynFlag : synchrone ou asynchrone (true ou false)

Exemple:

<script>

```
var requeteHTTP = new XMLHttpRequest() ;
```

```
requeteHTTP.open( "GET" , "data.php" ,true) ;
```

</script>

REMARQUES

- La méthode GET/POST est indiquée par le premier paramètre de `requeteHttp.open('GET',url,false);`
- En mode GET les paramètres sont passés dans l'URL
- Enfin si le dernier champs est à `false`, il indique que l'on est en mode synchrone (le client attend la réponse du serveur avant de continuer)

EXÉCUTION / ENVOI DE LA REQUÊTE

send(Variant body) : Envoi de la requête (null en get ou données en post)

Exemple

<script>

```
var requeteHTTP = new XMLHttpRequest() ;
```

```
requeteHTTP.open ( "GET" , "data.php" ,true) ;
```

```
requeteHTTP.send(); // requeteHTTP.send(null);
```

</script>

EXÉCUTION / ENVOI DE LA REQUÊTE

send(Variant body) : Envoi de la requête (null en get ou données en post)

Exemple avec **get**

```
<script>  
    var requeteHTTP = new XMLHttpRequest() ;  
    requeteHTTP.open ( "GET" , "data.php" ,true) ;  
    requeteHTTP.send(); // requeteHTTP.send(null);  
  
</script>
```

Exemple avec **post**

```
<script>  
    var nom="toto";  
    var prenom="tata";  
    var requeteHTTP = new XMLHttpRequest() ;  
    requeteHTTP.open ( "POST" , "data.php" ,true) ;  
    requeteHTTP.send("nom="+nom+"&prenom="+prenom);  
  
</script>
```

RÉCUPÉRATION DE L'OBJET XMLHTTPREQUEST

On assigne une fonction qui, lorsque l'état de la requête change, va traiter le résultat:

l'objet **onreadystatechange** : nom de la **fonction callback** à exécuter lorsqu'il y a un changement d'état de

Exemple

<script>

```
var requeteHTTP = new XMLHttpRequest() ;  
requeteHTTP.open ( "GET" , "data.php" ,true) ;  
requeteHTTP.send(); // requeteHTTP.send(null);  
requeteHTTP.onreadystatechange =function()  
    { .....traitement de résultat retourné.....  
    }
```

</script>

PROPRIÉTÉS DE XMLHttpRequest

XMLHttpRequest. *Nompropriétés* ou **NomObjetXMLHttpRequest.***Nompropriétés* ou **this.***Nompropriétés*

readyState : retourne l'état de la requête (2 = résultat partiellement reçu, 3 = chargement, 4 = OK)

status : le code de statut HTTP renvoyé par la requête (200 = OK, 404 = page non trouvée, 500 = erreur du serveur)

Exemple:

```
.....  
requeteHTTP.onreadystatechange=function()  
{  
    if (this.readyState == 4 && this.status == 200)  
    {  
        .....  
    }  
}
```

PROPRIÉTÉS DE XMLHttpRequest

responseText : réponse au format texte

responseXML : réponse au format XML

Exemple:

```
requeteHTTP.onreadystatechange =function()  
{  
    if (this.readyState==4 && this.status==200)  
    {  
        document.getElementById("myDiv").innerHTML=this.responseText;  
    }  
}
```

UN PREMIER EXEMPLE

```
<div id="demo">
  <h1> The XMLHttpRequest Object </h1>
  <button onclick="loadDoc()"> Change Content </button>
</div>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.open("GET", "data.txt", true);
  xhttp.send();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText; }
    }
  }
</script>
```

UN PREMIER EXEMPLE

data.txt

<h1>AJAX </h1>

<p>AJAX is not a programming language.</p>

<p>AJAX is a technique for accessing web servers from a web page.</p>

<p>AJAX stands for Asynchronous JavaScript And XML.</p>

JSON

JSON signifie Java**Script** Object **Notation**

JSON est un **format texte** pour stocker et transporter des données

Exemple

```
[{"id": "1", "name": "John", "age": 30}, ...]
```

JavaScript a une fonction intégrée pour convertir les chaînes JSON en objets JavaScript :

JSON.parse()

RECEVOIR DES DONNÉES EN FORMAT JSON

```
var html= "";  
requeteHTTP.onreadystatechange=function()  
{  
  if (this.readyState==4 && this.status==200)  
  {  
    var resp = this.responseText;  
    var data = JSON.parse(resp);  
    for(var i=0; i < data.length; i++)  
    {  
      var firstName=data[i].prenom;  
      var lastName=data[i].nom;  
      html+= firstName+ " " + lastName;  
    }  
    document.getElementById("data").innerHTML=html;  
  }  
}
```

ENVOYER DES DONNÉES EN FORMAT JSON

```
$conn=new PDO(pgsql:host=postgres;dbname=repertoire','root', "");

$stmt= $conn->query("SELECT * FROM personne where nom like 'A%' ");

$stmt->setFetchMode(PDO::FETCH_ASSOC);

//storing in array
$data=array();

while ($row = $stmt->fetch())
{
    $data[ ]=$row;
}
//returning response in JSON format
echo json_encode($data);
}
```

ÉVÉNEMENT KEYUP (ONKEYUP)

L'événement keyup permet d'enregistrer un gestionnaire d'événements qui se déclenche dès qu'une touche est relâchée.

```
<input type="text" onkeyup="NomFonction(this.value)" name="name" id="nameld">
```

```
<script>
```

```
function NomFonction(str)
{
  ...//call ajax
}
```

```
</script>
```


- Les sessions sont un moyen simple pour **stocker des données individuelles** pour **chaque utilisateur** en utilisant un **identifiant de session unique**.
- Cet **identifiant** est transmis de page en page par un **cookie** ou en le plaçant dans l'**url**
- Chaque client qui se connecte sur une page du site envoie son identifiant
- Le serveur PHP, retrouve les informations relatives à l'utilisateur à partir de son identifiant.

```
echo session_id(); //fhj5mts2h1jnf0so0j7hmit685
```

- Les sessions permettent de **conserver sur le serveur**, dans un fichier temporaire, des informations relatives à un internaute.
- Elles peuvent être utilisées pour faire **persister des informations entre plusieurs pages**.
- Les sessions permettent de **sauvegarder des variables de page en page** pendant une **certaine durée prédéfinie** (modifiable avec la fonction `session_set_cookie_params`).

Exemple pour un panier :

- Nous ferons le panier d'un site e-commerce avec un système de session.
- Nous n'enregistrerons pas les produits ajoutés au panier dans une base de données puisque la plupart du temps les paniers ne sont pas payés (les internautes ne finalisent pas toujours la commande).
- On utilise alors des sessions, cela évitera de "polluer" la base de données pour rien.
- En revanche, (si le panier et le paiement sont validés) la commande sera enregistrée dans une base de données.

DIFFÉRENCE ENTRE COOKIES ET SESSION EN PHP

Sécurité

- Les **sessions** sont stockées sur le serveur et les **cookies** ne sont conservés qu'au niveau du navigateur côté client.
 - Cookies: l'utilisateur peut voir et/ou modifier
 - Sessions: l'utilisateur ne peut pas accéder directement (car les données sont stockées sur le côté serveur)

DÉMARRAGE D'UNE SESSION PHP

- Avant d'utiliser les sessions sur une page, on doit toujours appeler la fonction **session_start()**
- **session_start** doit être placée avant toute balise HTML, et donc généralement tout en haut de votre page PHP

```
<?php
```

```
    session_start()
```

```
?>
```



Crée une nouvelle session ou récupère les données d'une session précédente

- Si aucune session n'existe : génération d'un identifiant, création d'un fichier de données sur le serveur
- Si on a déjà une session : les variables enregistrées sont chargées en mémoire dans le tableau **\$_SESSION**
- À la fin du script, le contenu du tableau **\$_SESSION** est sauvegardé sur le serveur

POSITIONNEMENT DES VARIABLES : \$_SESSION.

- Un tableau associatif contenant les variables du script courant.

Exemple:

```
<?php
```

```
    session_start();
```

```
    $_SESSION["langage"] = "PHP";
```

```
    echo $_SESSION["langage"];
```

```
?>
```

LES FONCTIONS ISSET, UNSET

- **isset** et **unset** permettent respectivement de déterminer si une variable est positionnée et de la supprimer.
- Dans le cadre des sessions :

isset(\$_SESSION['langage']) : permet de savoir si la variable langage est positionnée dans la session.

unset(\$_SESSION['langage']) : permet de supprimer la variable langage de la liste des variables de la session.

`session_destroy()`

- Détruit toutes les variables de la session mais pas la session même qui peut être récupéré par un nouveau `session_start()`.
- Pour tuer également la session, il faut détruire l'id de celle-ci. Si l'id est passé à l'aide de cookies, cela peut être effectué en détruisant le cookie associé (la fonction `setcookie()` peut être utilisée pour cela)
- appelle implicite à cette fonction lorsque on ferme le navigateur


```
CREATE OR REPLACE FUNCTION function_name(arguments)
RETURNS return_datatype AS $variable_name$
DECLARE
<declaration>
[...]
BEGIN
< function_body>
[...]
RETURN {variable_name | value };
END;
LANGUAGE plpgsql;
```

PL/SQL: EXEMPLE FONCTION SANS PARAMÈTRE

```
CREATE OR REPLACE FUNCTION total_num()  
RETURNS integer AS $totalnumbofpersonne$  
DECLARE  
totalnumbofpersonne integer;  
BEGIN  
SELECT COUNT(*) INTO totalnumbofpersonne FROM "Personne";  
RETURN totalnumbofpersonne;  
END;  
$totalnumbofpersonne$ LANGUAGE plpgsql;
```

PL/SQL: EXEMPLE FONCTION SANS PARAMÈTRES

Appel de fonction

```
SELECT total_num()
```

PL/SQL: EXEMPLE FONCTION AVEC PARAMÈTRES

```
CREATE OR REPLACE FUNCTION insert_personne(integer,text,text)
RETURNS integer AS $code$
DECLARE
code integer;
BEGIN
IF $1 NOT IN (Select "ID" from "Personne") THEN
INSERT INTO "Personne" ("ID","Nom","Prenom") VALUES ($1,$3,$3) ;
code:=$1;

ELSE
code:=0;

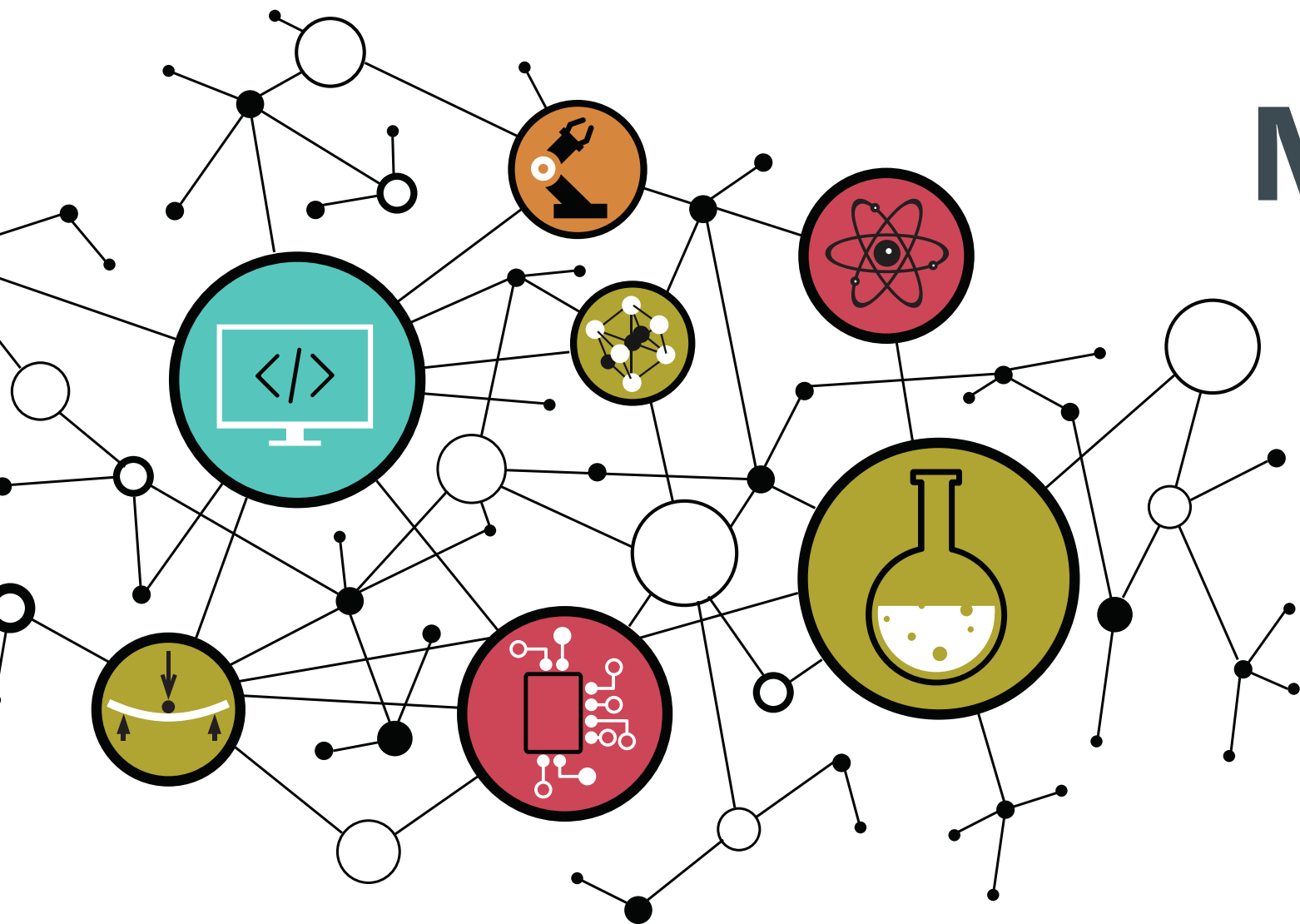
END IF;

RETURN code;
END;|
$code$ LANGUAGE plpgsql;
```

PL/SQL: EXEMPLE FONCTION AVEC PARAMÈTRES

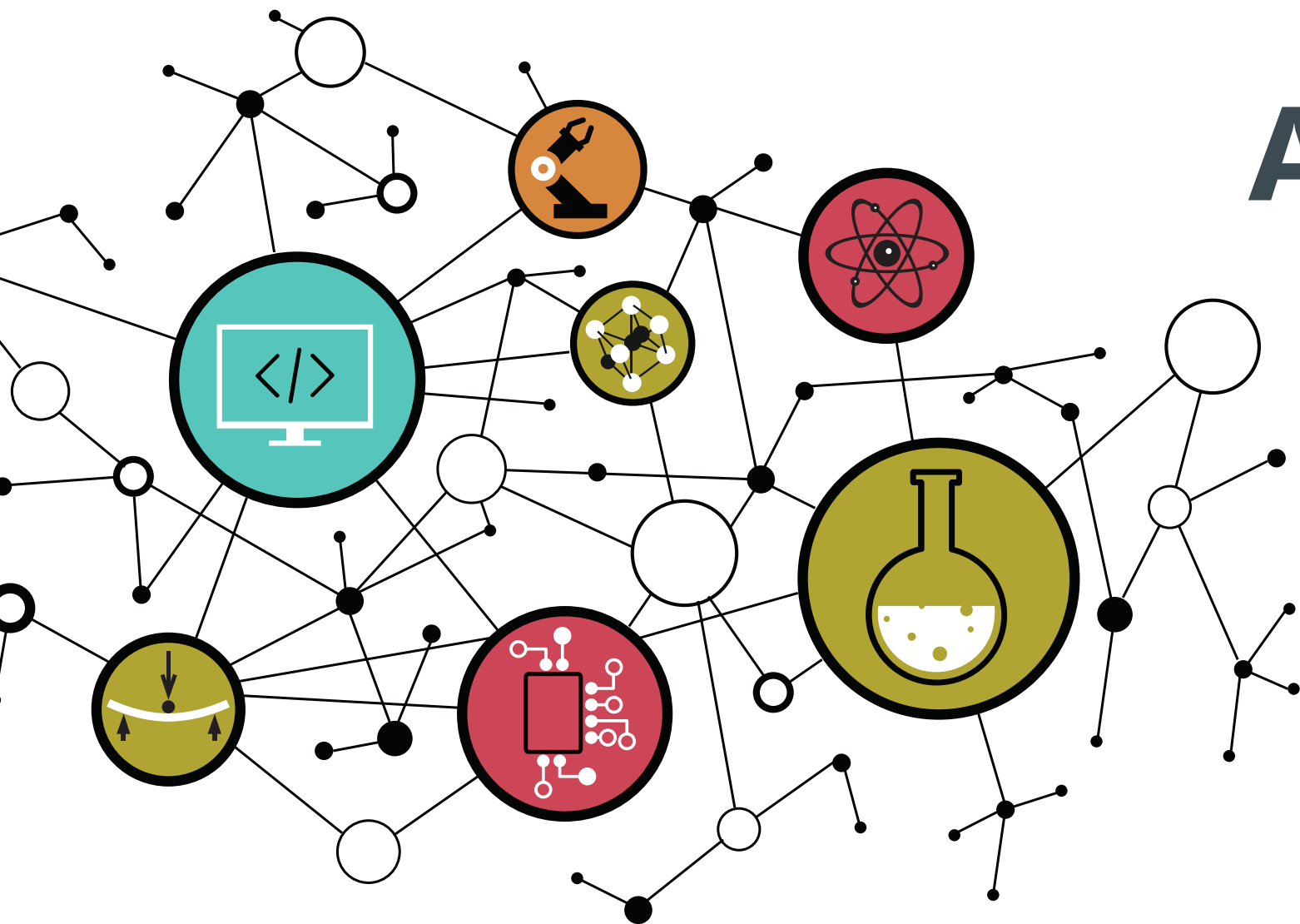
Appel de fonction

```
SELECT public.insert_personne(4, 'aa', 'bb')
```



Merci





Annexe

psql -h postgres -d livres -U nom

\dt pour lister les tableaux de la base de donnée

PDO

```
$conn = new PDO('pgsql:host=hostname;port=5432;dbname=db', 'user',  
'pass');
```

db: livres

hostname: 'postgres'

Pas de port

User: nome

Mot de passe: ensicaen

Dans le répertoire personnel de chaque utilisateur il y a un répertoire "public_html" pour la création des pages web.

Les sites sont ensuite consultables à partir de

<https://www.ecole.ensicaen.fr/~login>