

Web back-end

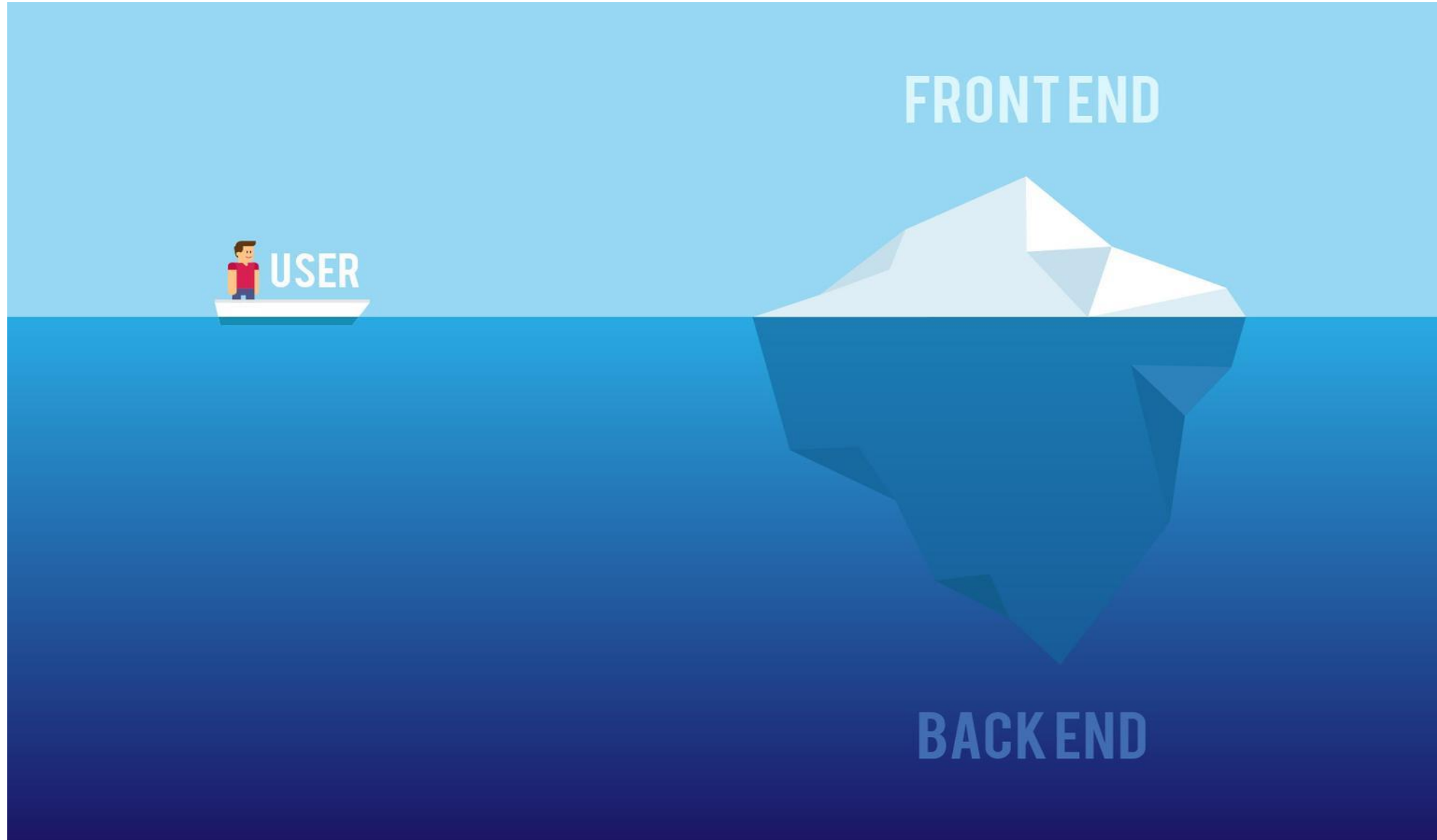
Kaouther TABIA

kaouther.tabia@ensicaen.fr



L'École des INGÉNIEURS Scientifiques

WEB BACK-END VS FRONT-END



WEB BACK-END VS FRONT-END

Développement Front-End :

Le front-end concerne tout ce avec quoi les utilisateurs interagissent directement sur un site web ou une application web.

Il est axé sur les aspects visuels et interactifs. Il est responsable à:

- Concevoir et développer des interfaces utilisateur (UI).
- Assurer la compatibilité avec différents appareils (ordinateurs, tablettes, mobiles).
- Gérer les fonctionnalités côté client (ex. : animations, menus de navigation).

Technologies principales :

1. **HTML** (HyperText Markup Language) : Structure le contenu.
2. **CSS** (Cascading Style Sheets) : Stylise le contenu (couleurs, mise en page, polices).
3. **JavaScript** : Ajoute de l'interactivité (contenu dynamique, interactions utilisateur).

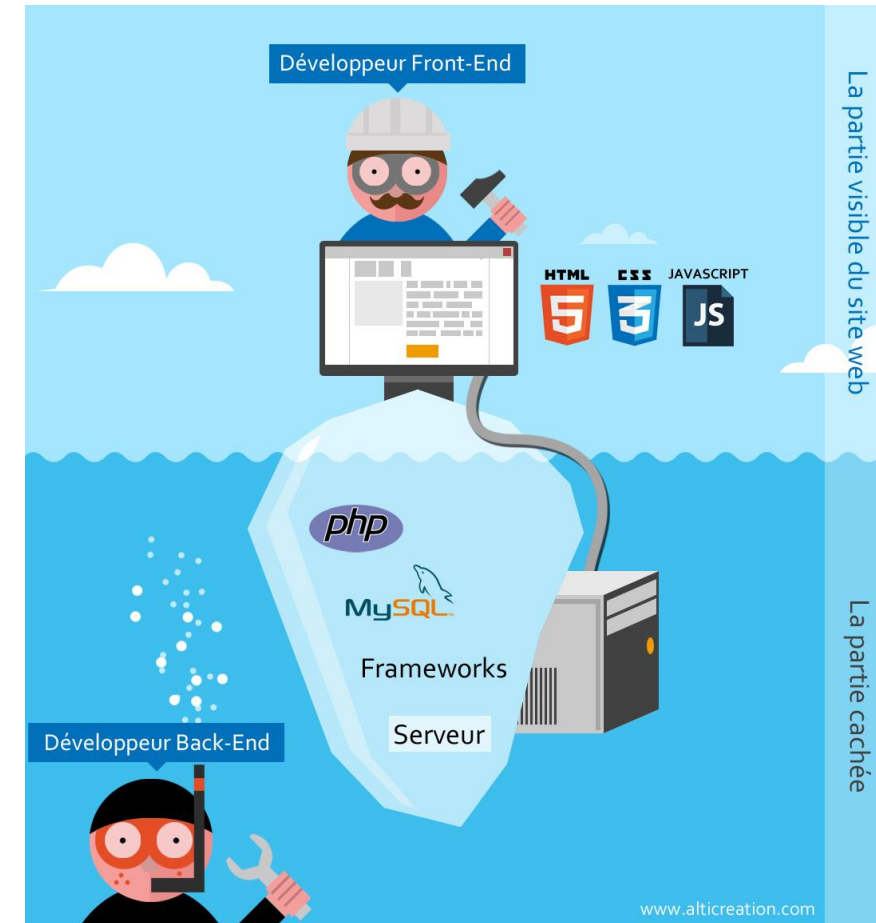
Développement Back-End :

Le back-end concerne la partie serveur, où se trouvent la logique du site, la gestion des bases de données et les interactions côté serveur. Il est responsable à:

- Gérer la logique côté serveur.
- Stocker, récupérer et traiter les données dans des bases de données.
- Assurer la sécurité et la scalabilité de l'application web.

Technologies principales :

1. **Langages côté serveur** : Node.js (JavaScript), Python, PHP, Ruby, Java, etc.
2. **Bases de données** : MySQL, PostgreSQL, MongoDB, etc.



WEB BACK-END

On peut décomposer le Back-End en trois parties essentielles :

1. Le Serveur (ou Hébergement Web) : est l'endroit où le site web et son application sont hébergés.

- Il s'agit d'une machine (physique ou virtuelle) qui exécute l'application back-end et gère les requêtes des utilisateurs.
- Les serveurs populaires incluent **Apache**, **NGINX**, et les plateformes cloud comme **AWS**, **Google Cloud**, ou **Azure**.

Fonction principale :

- Répondre aux requêtes des navigateurs et acheminer les données correspondantes (HTML, JSON, etc.).
- Gérer les connexions réseau et assurer la disponibilité du site.

2. L'Application (Logique métier) : C'est ici que réside la logique métier, qui détermine **comment les données sont traitées** et **comment les fonctionnalités du site fonctionnent**.

- C'est le code que le développeur écrit en utilisant des langages côté serveur comme **Node.js**, **Python**, **PHP**, ou **Ruby**.
- Elle inclut également des **API** (REST ou GraphQL) pour communiquer avec le front-end ou d'autres systèmes.

Fonction principale :

- Traiter les requêtes des utilisateurs, exécuter la logique nécessaire (ex. : calculer un prix, vérifier une connexion), et renvoyer les réponses appropriées.

3. La Base de Données : est l'endroit où toutes les informations utilisées par l'application sont stockées et récupérées.

- Par exemple : comptes utilisateurs, produits, commandes, etc.
- Types courants : **SQL** (MySQL, PostgreSQL) pour les données relationnelles ou **NoSQL** (MongoDB) pour les données non structurées.

Fonction principale :

- Stocker, organiser et fournir les données nécessaires à l'application en réponse aux requêtes.

WEB BACK-END: SERVEUR

Un **serveur** est un programme ou une machine qui joue le rôle de médiateur entre l'utilisateur (le client) et l'application ou les données stockées en arrière-plan. Il est chargé de gérer les demandes des utilisateurs et de fournir des réponses adaptées.

1. Recevoir les requêtes des clients

1. Quand un utilisateur clique sur un bouton ou visite une page web, une requête est envoyée au serveur.
2. Exemple :
 1. Vous allez sur Amazon et cherchez "chaussures". Votre navigateur envoie une requête au serveur d'Amazon pour afficher une liste de chaussures.

2. Gérer la logique métier

1. La logique métier, ce sont les règles qui définissent comment l'application fonctionne. Le serveur applique ces règles pour décider quoi faire avec la requête.
2. Exemple :
 1. Si vous ajoutez un produit à votre panier, le serveur vérifie :
 1. Si le produit est encore en stock.
 2. Si l'utilisateur est connecté.
 3. Combien d'articles peuvent être ajoutés.
3. Une fois tout vérifié, il met à jour votre panier en conséquence.

3. Prendre des décisions sur les réponses

1. Le serveur décide quoi renvoyer à l'utilisateur en fonction des données disponibles et des règles définies.
2. Exemple :
 1. Si vous essayez de vous connecter avec un mauvais mot de passe, le serveur renverra une réponse disant : "Mot de passe incorrect."
 2. Si votre mot de passe est correct, il vous redirige vers votre espace personnel.

4. Communiquer avec la base de données

1. Souvent, les données nécessaires pour répondre à une requête ne sont pas directement disponibles. Le serveur doit interroger une base de données pour récupérer ou mettre à jour les informations.
2. Exemple :
 1. Vous commandez un produit sur un site de vente.
 1. Le serveur demande à la base de données : "Quel est le stock restant de cet article ?"
 2. La base de données répond : "10 articles disponibles."
 3. Le serveur met à jour le stock pour indiquer qu'il reste 9 articles après votre commande.

WEB BACK-END: APPLICATION

L'**application** est une couche logicielle située entre le serveur et l'interface utilisateur. Elle sert d'intermédiaire pour gérer les données et les interactions, tout en appliquant des règles spécifiques au fonctionnement de l'application.

1. Relier le serveur et l'interface utilisateur

1. L'application prend les données brutes envoyées par le serveur (souvent dans un format comme JSON) et les transforme en informations compréhensibles pour l'interface utilisateur.
2. Exemple :
 1. Le serveur renvoie une liste d'articles sous forme de code. L'application convertit cela en une présentation visuelle (des images, des prix et des descriptions) que vous voyez sur une boutique en ligne.

2. Gérer les interactions utilisateur

1. L'application interprète ce que l'utilisateur fait sur l'interface (clics, saisies de texte) et décide comment répondre.
2. Exemple :
 1. Si vous cliquez sur "Ajouter au panier", l'application met à jour le panier en local (dans le navigateur) avant même de demander au serveur de sauvegarder ce changement.

3. Valider les entrées utilisateur

1. L'application vérifie que les informations saisies par l'utilisateur sont correctes avant de les envoyer au serveur.
2. Exemple :
 1. Si vous entrez une adresse e-mail sans le symbole "@", l'application vous affiche un message d'erreur sans envoyer cette donnée au serveur. Cela évite des requêtes inutiles.

4. Effectuer des opérations sans interagir avec la base de données

1. Certaines actions peuvent être gérées localement par l'application sans consulter le serveur ou la base de données.
2. Exemple :
 1. Quand vous remplissez un formulaire, l'application peut précharger des informations comme votre adresse ou les options de livraison selon vos préférences.

5. Contenir des composants fonctionnels spécifiques

1. L'application inclut souvent des modules spécialisés qui facilitent des tâches complexes.
 1. **Gestionnaires d'authentification** : Vérifient si un utilisateur est connecté avant de lui permettre d'accéder à certaines pages.
 2. **Contrôleurs de flux** : Organisent les étapes d'un processus (comme passer une commande ou compléter un formulaire).

WEB BACK-END: BASE DE DONNÉES

Une **base de données** est un système conçu pour stocker, organiser et gérer les informations nécessaires au bon fonctionnement d'une application ou d'un site web. Elle garantit que ces données sont accessibles, modifiables, et sécurisées.

1. Stocker les données de manière persistante

1. La base de données conserve les informations même lorsque le serveur ou l'application est hors ligne.
2. Exemple :
 1. Les comptes utilisateur, les commandes d'un site e-commerce, ou les messages dans une application de chat sont stockés de façon durable dans la base de données.

2. Organiser les données de manière structurée

1. Les données sont rangées selon un schéma ou une structure spécifique, ce qui facilite leur gestion.
2. Exemple :
 1. Une base de données d'un site de vente peut contenir :
 1. Une table pour les **utilisateurs** (nom, e-mail, mot de passe).
 2. Une table pour les **produits** (nom, prix, quantité en stock).
 3. Une table pour les **commandes** (utilisateur associé, produits commandés, total).

3. Offrir des mécanismes de manipulation des données

1. La base de données permet au serveur de :
 1. **Récupérer** : Chercher des informations spécifiques.
 2. **Mettre à jour** : Modifier des données existantes.
 3. **Insérer** : Ajouter de nouvelles données.
 4. **Supprimer** : Effacer des données qui ne sont plus nécessaires.
2. Exemple :
 1. Quand un utilisateur met à jour son profil, la base de données enregistre ces changements.
 2. Si un produit est acheté, le stock est mis à jour automatiquement.

4. Assurer la cohérence et la centralisation des données

1. La base de données garantit que toutes les parties de l'application utilisent les mêmes informations actualisées.
2. Exemple :
 1. Si un produit est ajouté au panier par un utilisateur, cette information est disponible partout : sur le site web, dans l'application mobile, ou même pour l'équipe de gestion.
 3. Cette centralisation évite les conflits, comme deux utilisateurs achetant simultanément le dernier article en stock.

EXEMPLE

Considérons un exemple simple d'une application web de **gestion de tâches (to-do list)**.

Dans cette application, nous aurons un serveur (PHP), une application (HTML, JAVASCRIPT et AJAX) et une base de données (POSTGRESQL).

Partie Serveur (PHP) :

- Gère les requêtes provenant du client.
- Traite les opérations liées à la base de données, notamment l'ajout de tâches et la récupération de la liste des tâches.
- Établit une connexion à la base de données.

Partie Application (Côté Client - HTML, JavaScript avec AJAX) :

- Affiche l'interface utilisateur permettant à l'utilisateur d'interagir avec l'application.
- Utilise AJAX pour envoyer des requêtes asynchrones au serveur sans recharger la page entière.
- Réagit aux actions de l'utilisateur, notamment l'ajout de nouvelles tâches.
- Contient la structure HTML de base de la page.
- Utilise JavaScript pour définir des fonctions qui interagissent avec le serveur via AJAX.
- Fournit un formulaire pour ajouter de nouvelles tâches.
- Affiche une liste dynamique des tâches récupérées depuis le serveur.

Partie Base de Données (PostgreSQL) :

- Stocke de manière persistante les données relatives aux tâches.
- Fournit une structure de base de données pour la table tasks.
- Permet au serveur d'effectuer des opérations CRUD (Create, Read, Update, Delete) sur les données des tâches.

RÔLE DU BACKEND DANS LE TRAITEMENT DES REQUÊTES

Réception des Requêtes :

- Lorsqu'un utilisateur interagit avec une application web, le frontend génère des requêtes HTTP qui sont envoyées au serveur backend.
- Ces requêtes peuvent inclure des actions telles que l'envoi de formulaires, la demande de données, ou d'autres interactions de l'utilisateur.

Traitement des Requêtes :

- Le backend reçoit ces requêtes et commence le processus de traitement. Il détermine comment répondre en fonction du type de requête et de la logique métier de l'application.
- La logique métier peut inclure la validation des données, l'accès à la base de données, l'exécution d'algorithmes spécifiques, etc.

Envoi des Réponses au Frontend :

- Une fois les réponses générées, le backend les envoie au frontend, qui les affiche ensuite à l'utilisateur.
- Les réponses peuvent également inclure des informations sur le succès ou l'échec de l'opération, des erreurs éventuelles, etc

Sécurité :

- La programmation côté serveur permet de protéger les logiques métiers sensibles et les données, car le code serveur n'est pas accessible ni modifiable par les utilisateurs

Traitement Efficace :

- Les opérations complexes et les calculs intensifs peuvent être effectués côté serveur, libérant ainsi les ressources côté client pour une expérience utilisateur plus fluide

IMPORTANCE DE LA PROGRAMMATION SERVEUR-SIDE :

Gestion des Sessions :

- La programmation côté serveur permet la gestion des sessions, ce qui est crucial pour suivre l'état de l'utilisateur lors de différentes interactions avec l'application

Interaction avec la Base de Données :

- Les langages côté serveur, tels que PHP, facilitent l'interaction avec les bases de données. Cela permet de stocker, récupérer et mettre à jour des données en réponse aux actions de l'utilisateur.

Support de Plusieurs Clients :

- Une application serveur-side peut traiter simultanément les requêtes de plusieurs clients, assurant ainsi une expérience utilisateur cohérente et réactive.

- **PHP** (Hypertext Preprocessor) est un langage de programmation populaire côté serveur. Il est utilisé pour générer dynamiquement des pages web et interagir avec les bases de données, tout en assurant la logique métier et en répondant aux requêtes des utilisateurs.
- PHP est largement utilisé pour développer des sites web et des applications web interactives.

1. Le Serveur avec PHP :

- PHP s'exécute côté serveur, ce qui signifie qu'il est responsable de gérer les requêtes envoyées par le client (navigateur web ou application mobile).
- Quand un utilisateur demande une page web, le serveur reçoit la requête, exécute le code PHP et renvoie une réponse (souvent une page HTML dynamique).

Exemple avec PHP :

• Quand un utilisateur visite une page web de votre site, PHP peut générer dynamiquement du contenu, comme l'affichage des produits disponibles, ou l'affichage d'une page de connexion si l'utilisateur n'est pas connecté.

```
<?php
// Exemple de réponse dynamique avec PHP
echo "Bonjour, utilisateur ! Voici la liste des produits :";
// Code pour récupérer et afficher les produits depuis la base de données...
?>
```

2. L'Application avec PHP :

Dans l'application, PHP gère la logique métier. Il reçoit les données du client, applique des règles ou effectue des opérations (comme valider un formulaire ou calculer un prix) avant de communiquer avec la base de données ou de générer une réponse.

Exemple avec PHP :

- Si un utilisateur remplit un formulaire pour s'inscrire, PHP peut vérifier que les informations sont valides (par exemple, que l'email est dans le bon format) avant de les envoyer à la base de données.

```
<?php
// Exemple de validation avec PHP
if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
    // Si l'email est valide, ajouter l'utilisateur dans la base de données
    // Code pour insérer dans la base de données...
} else {
    echo "Adresse email invalide.";
}
?>
```

PHP : LANGAGE BASE DE DONNÉES

3. La Base de Données avec PHP :

PHP est très souvent utilisé pour interagir avec des bases de données, généralement via des extensions comme **MySQL** ou **PDO (PHP Data Objects)**.

Le langage permet de récupérer, ajouter, modifier ou supprimer des données dans une base de données en fonction des actions des utilisateurs.

Exemple avec PHP et MySQL :

- Lorsque vous inscrivez un nouvel utilisateur, PHP va envoyer une requête SQL pour insérer ces informations dans une base de données.

```
<?php
// Connexion à la base de données
$mysqli = new mysqli "localhost" "utilisateur" "motdepasse" "basededonnees"

// Vérification de la connexion
if $mysqli->connect_error
    die "Connexion échouée : " . $mysqli->connect_error

// Insertion des données de l'utilisateur
$sql = "INSERT INTO utilisateurs (email, motdepasse) VALUES ('$email', '$motdepasse')"
if $mysqli->query $sql === TRUE
    echo "Nouvel utilisateur ajouté avec succès."
else
    echo "Erreur : " . $sql . "<br>" . $mysqli->error

// Fermeture de la connexion
$mysqli->close
?>
```

- *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
 - PostgreSQL
- Gestion d'annuaires
- Manipulations des URL
- Génération et « parsing » de fichiers XML
- Exécution de commandes Shell
- ...

POURQUOI UTILISER PHP ?

- **Facilité d'apprentissage :**

- PHP est relativement facile à apprendre, surtout pour ceux qui ont déjà des connaissances en HTML.

- **Compatibilité :**

- Il fonctionne bien avec différentes bases de données, serveurs web et systèmes d'exploitation.

- **Popularité et longévité :**

- PHP est largement utilisé dans l'industrie du web depuis de nombreuses années, assurant la stabilité et la fiabilité.

- **Support étendu :**

- Des frameworks populaires tels que Laravel et Symfony sont construits sur PHP, offrant des solutions structurées et efficaces pour le développement.

- **Intégration facile avec d'autres technologies web :**

- PHP fonctionne bien avec des technologies telles que HTML, CSS, JavaScript, et peut être utilisé avec des serveurs web tels qu'Apache

- **Gratuit :**

- PHP est open source et gratuit à utiliser, ce qui le rend économique pour les projets.

- 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

PHP

Exemple: Un utilisateur demande la page d'accueil de Facebook.

Voici comment se dérouleraient les étapes que vous avez mentionnées :

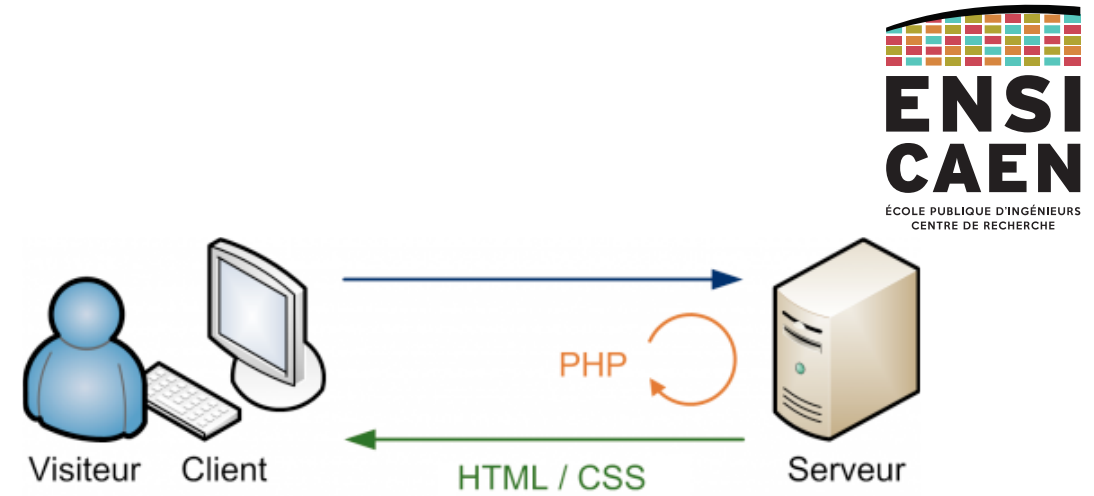
1.L'utilisateur ouvre son navigateur et tape l'URL de Facebook.

2.Le serveur de Facebook reçoit la demande.

- L'interpréteur PHP traite le code.
- Le serveur exécute le code PHP nécessaire.

3.Le serveur renvoie la page générée.

4.Le navigateur affiche la page d'accueil de Facebook.





Mémo PHP

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

```
<HTML>  
<HEAD><TITLE> page statique </TITLE></HEAD>  
<BODY>  
Nous sommes le 10/01/2025  
</BODY>  
</HTML>
```

*.html

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

*.php

- 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 chaînes de caractères (string)**
 - Entre guillemets (interprété) ou apostrophes (non interprété)

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

- **Les nombres entiers (int)**
- **Les nombres décimaux (float)**
- **Les booléens (bool)**

PHP: langage à typage dynamique :

Variables :

- Préfixés par un \$
- Sensible aux majuscules
- **Pas de déclaration, typage implicite**
 - \$var=54 → entier
 - \$var="toto" → chaîne de caractères

```
<?php
$age= 17; // int
$nom="toto"; // string
$poids = 57.3; // float
$var3 = true; // Valeur Booléene
$var4 = Array("a","b","c"); // Tableau
?>
```

- Variables
 - Affichage:

```
<?php
    $age= 17;
    $nom="toto"; // string
    $poids = 57.3; // float
    echo "Le visiteur $nom a";
    echo "$age ans";
    echo " et un poids de
    $poids kg.";
?>
```

!!! ; !!!

!!! Guillemets !!!

Le visiteur toto a 17 ans et un poids de 57.3 kg.

Pour concaténer / fusionner deux variables en PHP nous utiliseront le caractère ' . '

Exemple : <?php

```
$var1 = 1;
```

```
echo "=>" . $var1 . "<="; // =>1<=
```

- 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 une 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++;
}
?>
```

Foreach utilisée pour itérer directement sur des collections/ des ensembles de données comme des tableaux ou des objets.

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

```
<?php
foreach ($collection as $valeur)
    //instruction
}
?>
```

- 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

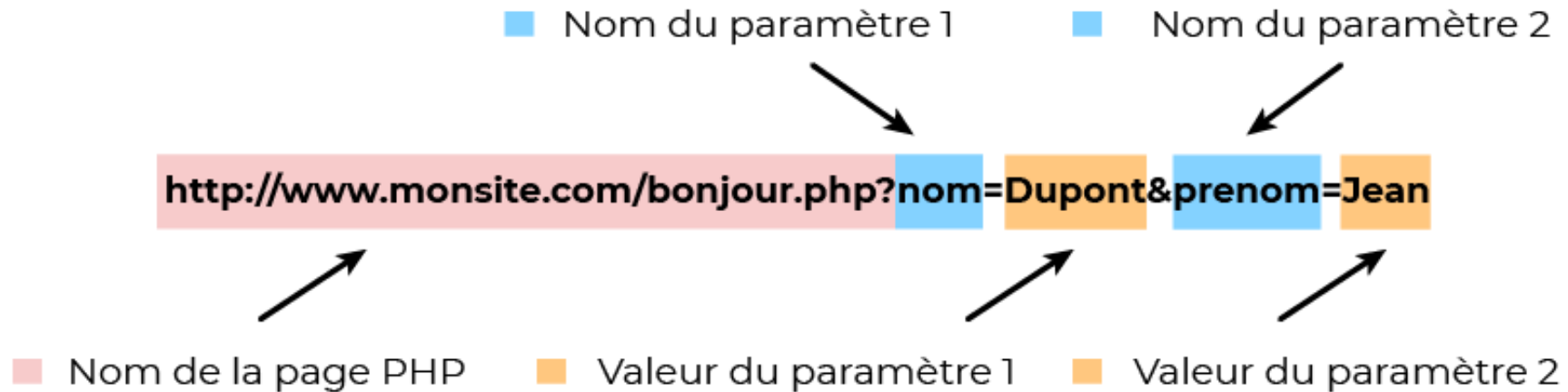
```
$jour = date('d');  
$mois = date('m');  
$annee = date('Y');  
echo $jour;
```

11

?>

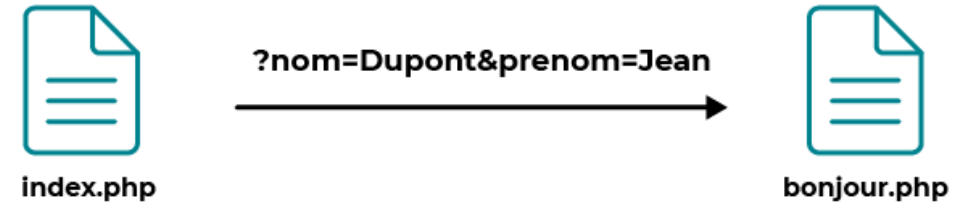
- **Transmission de données**
 - Via l'URL
 - Via un formulaire

- Transmission de données
 - Via l'URL



- Transmission de données
 - Via l'URL

Envoie des données



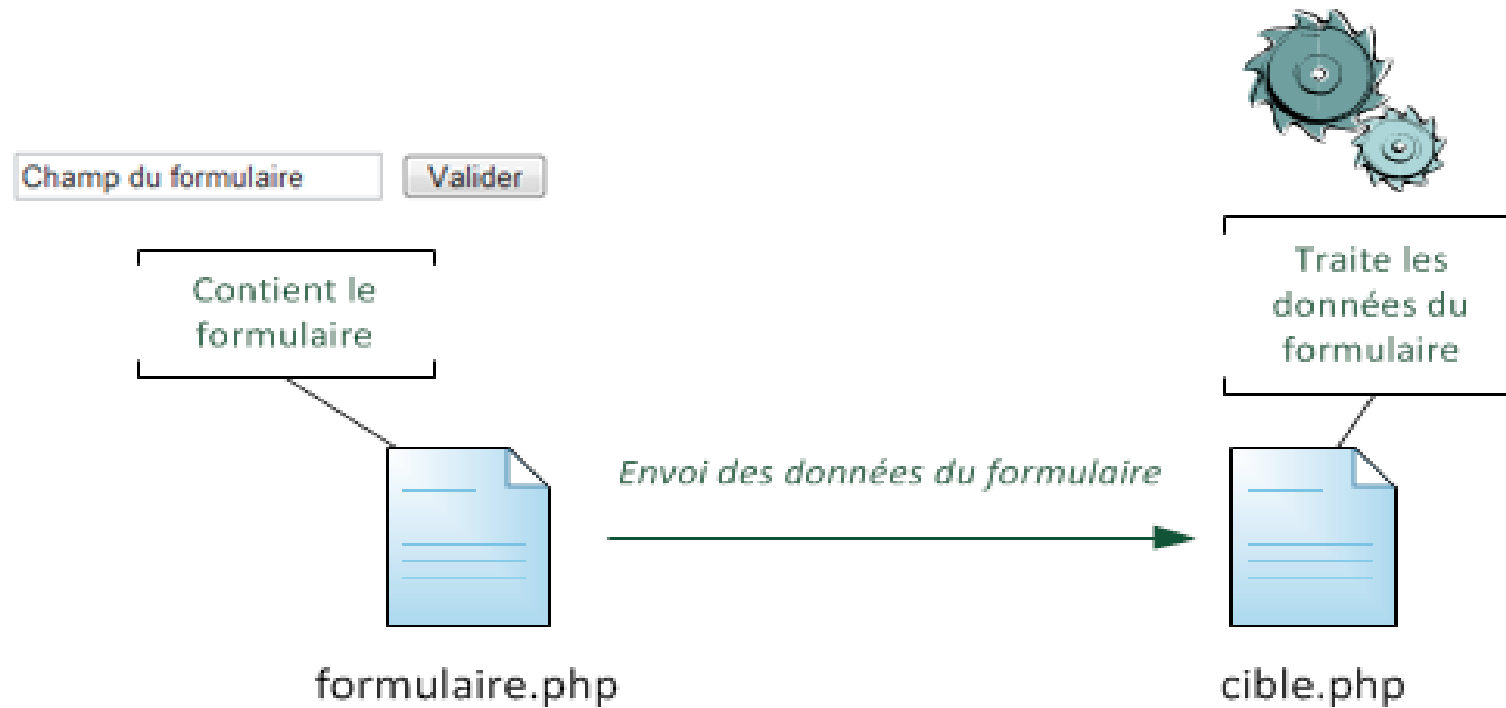
```
<a href="bonjour.php?nom=Dupont&prenom=Jean">Hello</a>
```

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
 - \$_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

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

```
<?php
```

```
$file = fopen("fichier.txt", "w");
```

```
$contenu="Hello";
```

```
fwrite($file,$contenu);
```

```
fclose($file);
```

```
?>
```

```
<?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

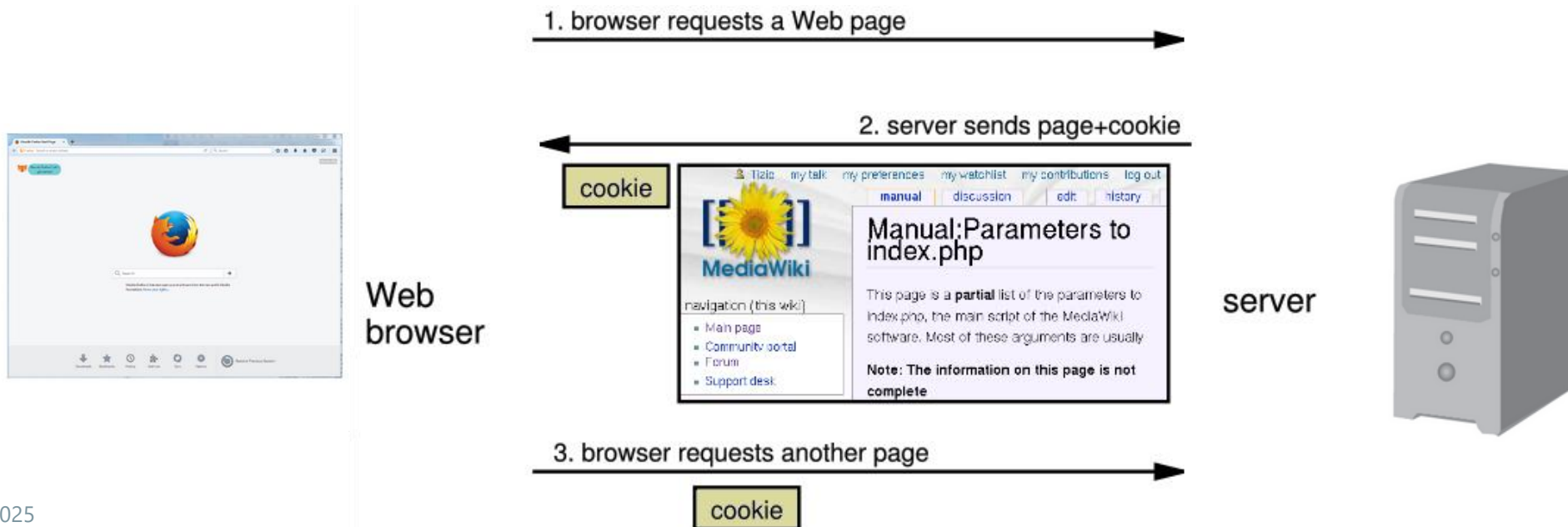
- Ce ne sont pas des virus, juste de petits fichiers texte qui permettent de retenir des informations.
- déposé sur le disque dur de l'internaute par le serveur du site visité ou par un serveur tiers (régie publicitaire, service de web analytique, etc.)

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



COOKIES: EXEMPLES

- **Cookie de session (utilisé par un site de commerce électronique) :**

- Ce cookie stocke un identifiant de session unique pour suivre les actions d'un utilisateur tout au long de sa visite sur le site. Il est essentiel pour maintenir l'état de la session, permettant à l'utilisateur d'ajouter des articles au panier et de passer à la caisse.

- **Cookie persistant (utilisé par un site de blog) :**

- Si un utilisateur personnalise le thème ou la mise en page du blog, ces préférences peuvent être stockées dans ce cookie. Ainsi, lors des visites ultérieures, le site peut charger les préférences personnalisées de l'utilisateur.

- **Cookie de suivi d'analyse (utilisé par un site d'actualités) :**

- Ces cookies sont utilisés par Google Analytics pour collecter des données sur la façon dont les visiteurs interagissent avec le contenu du site. Ils aident les propriétaires du site à comprendre quelles pages sont populaires, combien de temps les utilisateurs passent sur le site, etc.

- ...

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

nom **valeur** **Temps avant expiration**

```
<?php  
setcookie("seen", "yes", time()+3600*24*365);  
?  
<html>  
  <head><title>Hello</title></head>  
  <body><h1>  
    <?php  
    if(isset($_COOKIE["seen"]))  
      echo "Je vous ai déjà vu quelque part";  
    else  
      echo "Hello ";  
    ?>  
  </h1>  
</body>  
</html>
```

Le temps à cet instant (`time()`) + un an (3600 secondes * 24 heures * 365 Jours)

CONNEXION À UNE BASE DE DONNÉES

PDO (PHP Data Objects)

- Une interface pour accéder à une base de données depuis PHP
- Permet de manipuler des 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 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'");
```

```
?>
```

EXEMPLE D'UTILISATION DE LA FONCTION QUERY AVEC LA COMMANDE SELECT 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) //foreach parcourt chaque ligne retournée par la requête SQL
    {

        print_r($row); // Affiche chaque ligne de la table FOO

    }

    $dbh = null;
}

catch (PDOException $e)
{
    print "Erreur !: " . $e->getMessage() . "<br/>";

    die();
}

?>
```

ID	nom	prenom
1	aa	bb
2	cc	dd

Base de données

Résultat de la requête PDO

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)

Un tableau (array) associatif où les clés du tableau correspondent aux noms des colonnes de table. De plus, chaque colonne est également référencée par un indice numérique.

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' ");
```

?>

- 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();
}
```



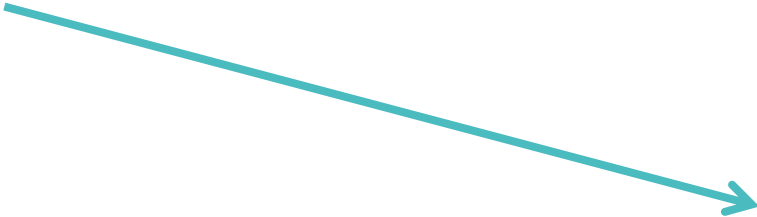
aa bb

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%' ");

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



aa bb

CRÉATION D'UNE 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

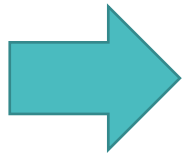
```
<?php
$dbh = new PDO('pgsql:host=localhost;dbname=test', $user, $pass);

// 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); ?>;
```

```
console.log(variableRecuperee); // Affiche "nom" dans la console
```

```
</script>
```

Une fonction qui prend en entrée une structure de données PHP (généralement un tableau associatif ou un objet) et la convertit en une chaîne JSON.

JSON signifie JavaScript Object Notation



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 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 à 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.

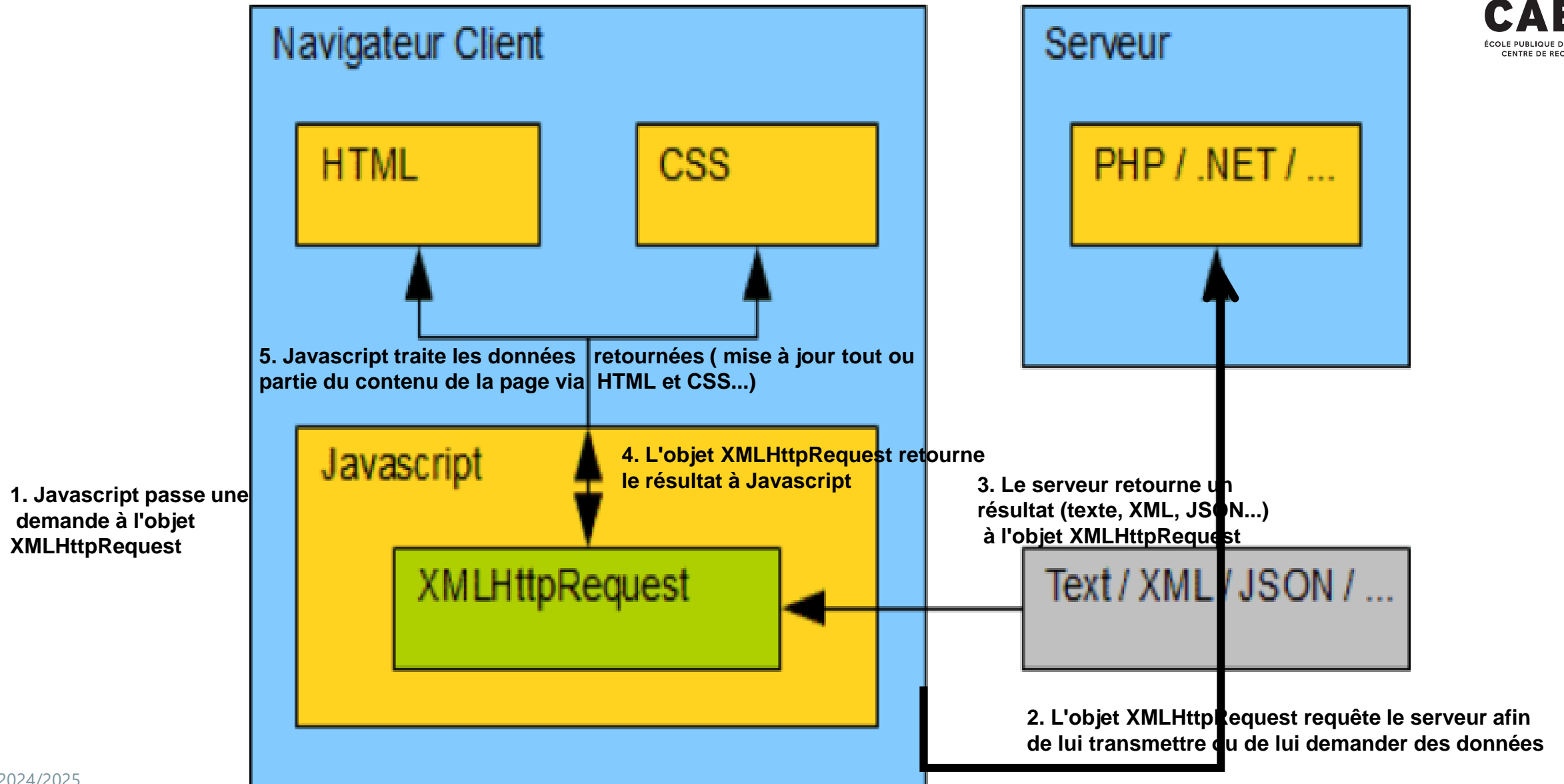
- **Avantage**
 - permettre une plus grande réactivité de l'interface par rapport au web classique

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**,...).



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 champ est à `false`, il indique que l'on est en mode synchrone (le client attend la réponse du serveur avant de continuer l'exécution du code)
- Le mode synchrone (`false`) est non recommandé dans les environnements modernes. Il est recommandé d'utiliser toujours le mode asynchrone (`true`).

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 nom="toto";  
var prenom="tata";  
var requeteHTTP = new XMLHttpRequest() ;  
requeteHTTP.open ( "GET" , "data.php?nom=" + nom +  
"&prenom=" + prenom ,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>
```

- En PHP, les sessions sont un mécanisme qui permet de stocker des données de manière persistante entre différentes requêtes HTTP d'un même utilisateur.
- Les sessions sont souvent utilisées pour maintenir l'état de connexion de l'utilisateur, stocker des préférences utilisateur, ou tout autre type de données que on souhaite conserver tout au long de la navigation d'un utilisateur sur un site web.

- 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**
- Lorsque la session est démarrée avec **session_start()** en PHP, le serveur génère un identifiant de session unique (par exemple, une chaîne aléatoire) et le stocke du côté serveur.
- En même temps, le serveur envoie un cookie au client qui contient cet identifiant de session. Le cookie est généralement nommé PHPSESSID par défaut, mais vous pouvez le personnaliser si nécessaire (**session_name("MON_COOKIE_PERSONNALISE");**);
- 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)

COMMENT LES SESSIONS FONCTIONNENT EN PHP

- Démarrage de la session :

- Avant de pouvoir utiliser des sessions en PHP, vous devez démarrer la session à l'aide de la fonction **session_start()**.
- Vous devriez placer cette fonction au début de chaque script PHP où vous avez l'intention d'utiliser des sessions.
- **session_start** doit être placée avant toute balise HTML, et donc généralement tout en haut de votre page PHP

```
<?php  
    session_start()  
?>
```

DÉMARRAGE D'UNE SESSION 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

CRÉATION OU MODIFICATION DE VARIABLES DE SESSION: \$_SESSION.

- Une fois la session démarrée, vous pouvez créer ou modifier des variables de session en utilisant le tableau associatif \$_SESSION.
- Ces variables seront disponibles pour toute la durée de la session.
- **Accès aux variables de session** : Vous pouvez accéder aux variables de session à tout moment pendant la durée de la session en utilisant le tableau \$_SESSION.
- À la fin du script, le contenu du tableau **\$_SESSION** est sauvegardé sur le serveur

Exemple:

<?php

```
session_start();
```

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

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

FERMETURE DE LA SESSION : LES FONCTIONS ISSET, UNSET

- Vous pouvez supprimer une variable de session en utilisant l'instruction **unset()**
- **isset** et **unset** permettent respectivement de déterminer si une variable est positionnée et de la supprimer.

Exemple:

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

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

`session_destroy()`

- La session se ferme automatiquement lorsque le script PHP se termine. Cependant, vous pouvez explicitement fermer la session en appelant **`session_destroy()`**. Cela détruira toutes les données de session existantes.
- Détruit toutes les variables de la session mais pas la session même qui peut être récupéré par une nouvelle **`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)

```
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;

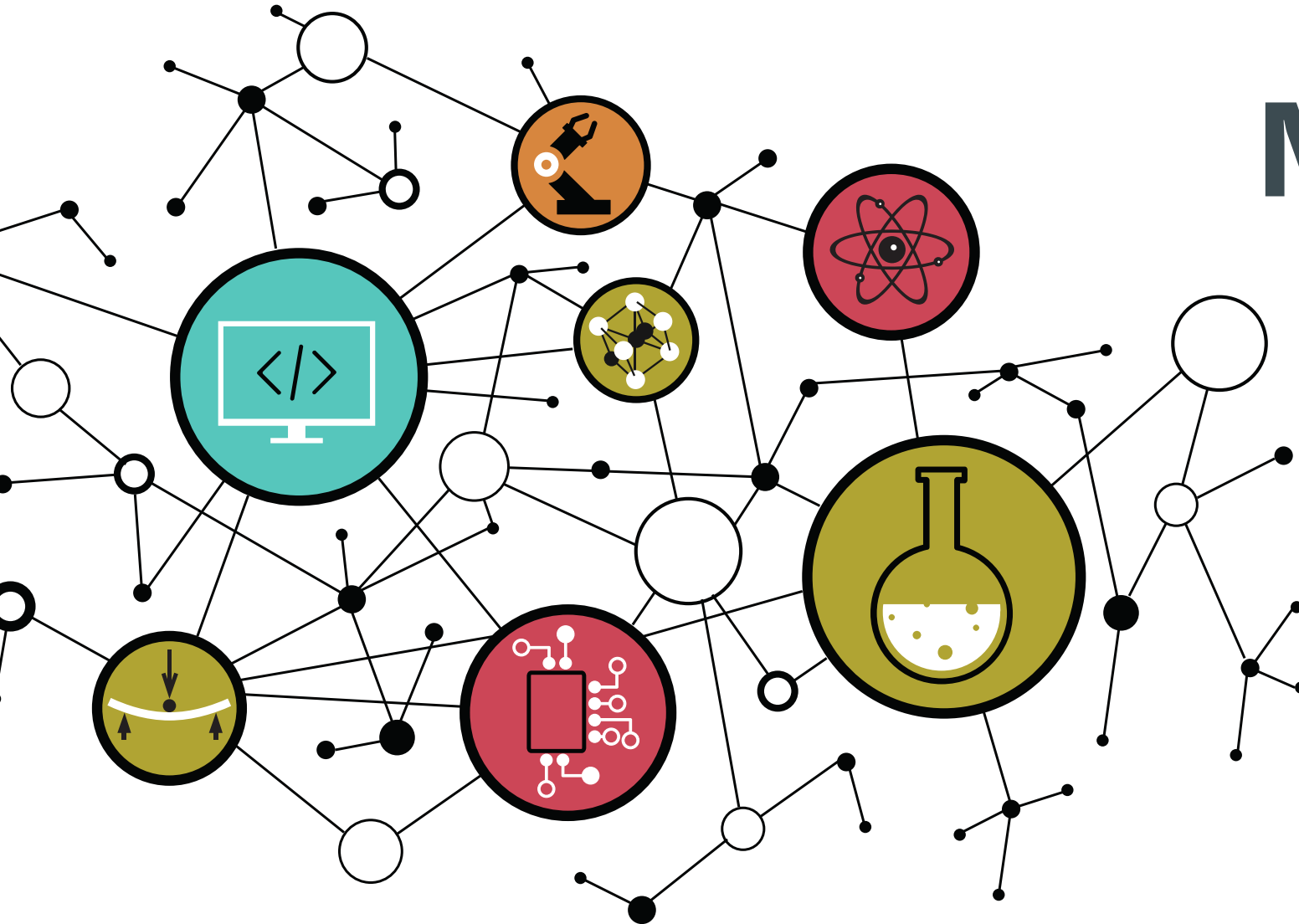
RETURN code;
END;|
$code$ LANGUAGE plpgsql;
```

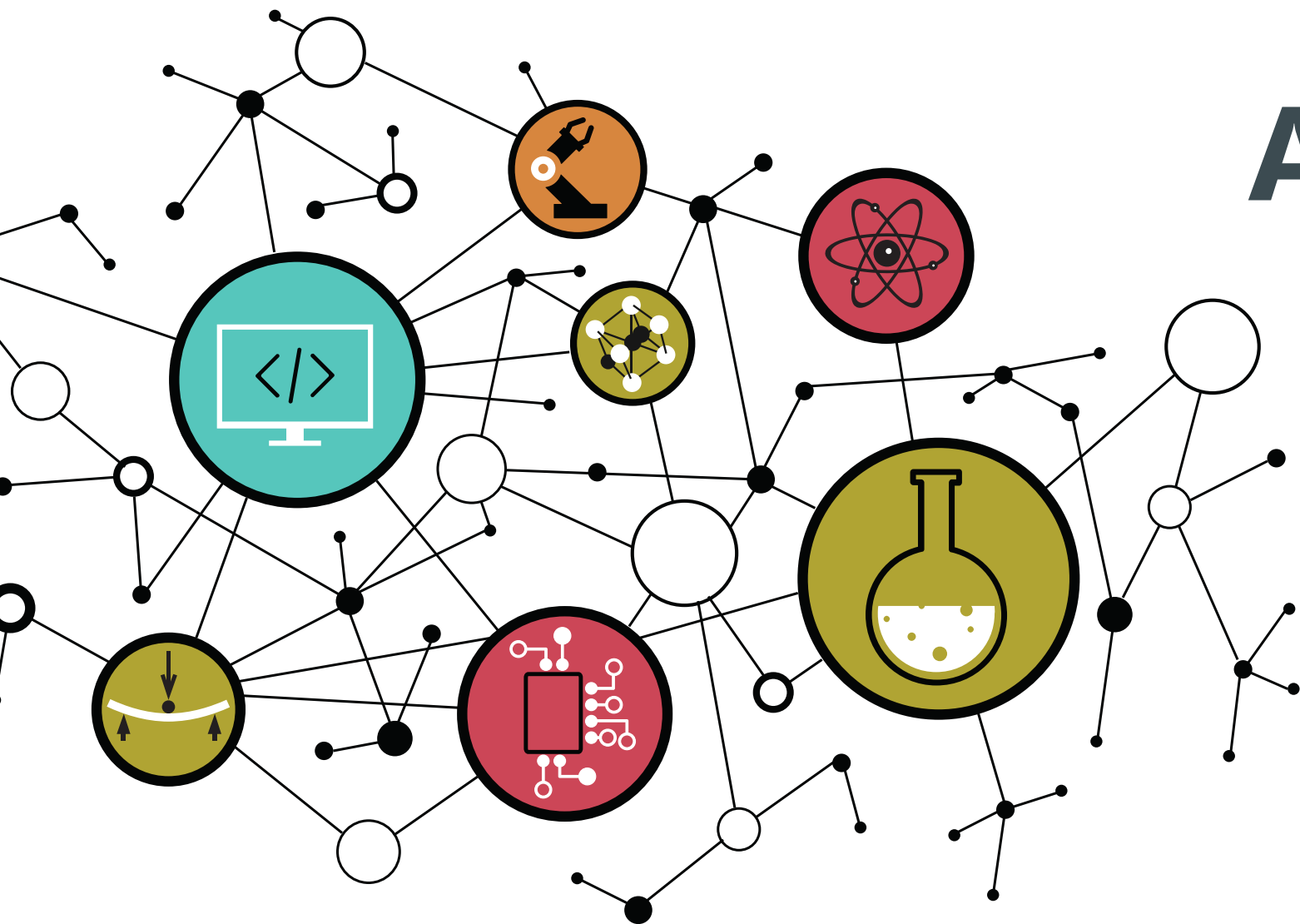
PL/SQL: EXEMPLE FONCTION AVEC PARAMÈTRES

Appel de fonction

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

Merci





Annexe

Comment connecter à la base de données

psql -h postgres -d livres -U nom

\dt pour lister les tableaux de la base de données

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

Comment connecter à la base de données

Si vous rencontrez une erreur de connexion du type "driver non trouvé" :

1. Vérifiez les drivers PDO disponibles :

1. Ajoutez la ligne suivante dans votre code PHP : `<?php phpinfo(); ?>`
2. Exécutez ce code et examinez la section **PDO** du tableau généré pour vérifier les drivers disponibles.
3. Si **pgsql** n'est pas listé, passez à l'étape suivante.

2. Modifiez le fichier php.ini :

1. Recherchez le fichier php.ini.
2. Dans ce fichier, trouvez et décommentez (en supprimant le point-virgule ;) la ligne suivante : `;extension=pdo_pgsql.so`
3. Si cette ligne n'existe pas, ajoutez-la manuellement.
4. Décommentez également toutes les autres lignes contenant **pgsql**.

3. Vérifiez à nouveau les drivers PDO :

1. Rechargez le tableau **phpinfo()** pour vérifier si **pgsql** est maintenant disponible.

4. Installez le driver PDO si nécessaire :

1. Si le driver **pgsql** n'est toujours pas disponible, installez-le à l'aide de la commande suivante : `sudo apt-get install php<versionphp>-pgsql`
2. Remplacez `<versionphp>` par la version de PHP installée sur votre système (par exemple, **php8.1**).

Si vous utilisez un serveur local

Connectez-vous à PostgreSQL via le terminal :

```
sudo -u postgres psql
```

Dans l'interface PostgreSQL, créez une base de données :

```
CREATE DATABASE livres;
```

Quittez PostgreSQL avec :

```
\q
```

Une fois la base de données prête, utilisez la commande suivante pour importer le fichier SQL :

```
psql -U <nom_utilisateur> -d <nom_base> -f /chemin/vers/fichier.sql
```

S'il y a une erreur d'importation, vérifiez le fichier situé dans **/etc/postgresql/<version>/main/pg_hba.conf**. Remplacez toutes les occurrences de **md5** ou **peer** par **trust**.

Rq Version: la version de postgresql

Après l'importation, connectez-vous à la base de données pour vérifier si les données ont été importées correctement

```
psql -U postgres -d livres
```

Dans l'interface psql, vérifiez les tables importées :

```
\dt
```

Si vous voulez inspecter le contenu d'une table, utilisez :

```
SELECT * FROM nom_de_table;
```


Si vous utilisez les machines de l'Ensi

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

web.ecole.ensicaen.fr/~login