|  |
| --- |
| Capacités attendues |
| - Interaction avec l’utilisateur dans une page Web. Analyser et modifier les méthodes exécutées lors d’un clic sur un bouton d’une page Web.  - Interaction client/serveur. Requêtes HTTP, réponses du serveur. Distinguer ce qui est exécuté sur le client ou sur le serveur et dans quel ordre. Distinguer ce qui est mémorisé dans le client et retransmis au serveur. Reconnaître quand et pourquoi la transmission est chiffrée. Il s’agit de faire le lien avec ce qui a été vu en classe de seconde et d’expliquer comment on peut passer des paramètres à un site grâce au protocole HTTP.  - Formulaire d’une page Web. Analyser le fonctionnement d’un formulaire simple. Distinguer les transmissions de paramètres par les requêtes POST ou GET. Discuter les deux types de requêtes selon le type des valeurs à transmettre et/ou leur confidentialité. |

# Interaction avec l’utilisateur dans une page web : utilisation du JavaScript

**a. Introduction au JavaScript**

Le JavaScript est un langage de programmation interprété par le navigateur web. De ce fait il est très dépendant du type de navigateur utilisé et de la puissance de la machine sur laquelle il fonctionne, même si ce dernier point n’est plus vraiment un problème depuis les gains de puissance des processeurs (y compris mobiles) de ces dernières années.

Même s’il y a le mot « java » dans le nom de ce langage, il n’a pas grand-chose à voir avec le langage Java. Sa syntaxe est relativement simple et beaucoup d’éléments sont assez proches du Python. Il y a toutefois une différence majeure de syntaxe à laquelle vous devrez être attentif : **chaque instruction se termine par un point-virgule « ; »** !!

Le JavaScript pour quoi faire ?

Le JavaScript a pour but principal de modifier des éléments de la page HTML et ainsi d’interagir avec la page sur le navigateur du client. Pour cela il s’appuie sur le Document Object Model (DOM) de HTML : chaque partie du code HTML est considérée comme un « objet » (au sens de programmation) qui possède des attributs modifiables.

Pour accéder à un élément HTML, le plus simple est de rajouter une propriété « id » à une balise. Il sera alors possible d’accéder à cet élément pour le modifier.

*Par exemple*: pour modifier le contenu d’un paragraphe codé de la façon suivante en HTML : <p id=“paragraphe1“>texte</p>, le code JavaScript va chercher le paragraphe avec la méthode « getElementById » et en modifier le contenu de la façon suivante :

document.getElementById(“paragraphe1“).innerHTML=“Bonjour“

Comme il est possible de modifier du code HTML, on pourra changer le texte, l’apparence du texte (couleur, taille, emplacement à l’écran), remplacer des images, montrer ou cacher des blocs de textes… Tout cela ouvrant de nombreuses possibilités d’interactions sur une page web.

Où placer le code JavaScript ?

Le code JavaScript peut être placé à plusieurs endroits selon vos besoins et la quantité de code que l’on veut utiliser :

* Entre les balises <script>code</script>. Il est recommandé de placer les scripts de la page dans la partie « head » (en-tête) du document HTML.  
  Si vous utilisez des scripts très longs à exécuter et que votre priorité est que l’utilisateur voit la page avant tout, il faudra alors placer vos scripts dans un élément « <footer></footer> », en fin de page web, pour ne pas ralentir le chargement de la page (voir exemple plus haut).
* Dans un fichier externe qui sera appelé par votre document HTML à l’aide de l’instruction <script src=“monscript.js“ defer></script>. « monscript.js » sera un fichier texte contenant votre code et l’attribut « defer » indique au navigateur (récent) de ne pas mettre en pause l’affichage du document en attendant de charger et d’analyser le code JavaScript.  
  **Cette méthode est à privilégier si vous souhaitez facilement réutiliser votre code sur plusieurs pages web**.
* Directement dans un attribut évènement d’une balise HTML (voir plus bas). Cette méthode étant à réserver à des codes très courts que vous n’allez pas réutiliser par la suite. Vous pouvez la combiner avec la balise « script » en utilisant des fonctions (il n’y a pas de distinction entre fonction et procédures en JavaScript).
* Quelques éléments de syntaxe JavaScript :
  + Les **commentaires** en JavaScript commencent toujours par « // » en début de ligne.
  + La **déclaration de variable** ou de constantes en JavaScript se fait avec la commande « var » et les types seront déterminés par l’interpréteur en fonction du contenu (comme en Python par défaut).

Exemple : var a=5 ;

* + Il est possible de créer des **objets** très simplement avec la commande « var » en indiquant les valeurs des propriétés et les méthodes (fonctions) dans des accolades de la façon suivante :

Exemple :

var voiture = {type :“compacte“, nbroues :4, demarrer : function(){return “vroom vroom“}, annee :2020}

Nous avons ici déclaré l’objet voiture qui possède les propriétés type, nbroues et annee et une méthode, demarrer, qui contient une fonction renvoyant le texte « vroom vroom ». Notez les virgules qui séparent les paramètres de l’objet. Cette fonction pourra être appelée de la façon suivante pour changer un paragraphe dont l’ « id » est « demo » : document.getElementById(“demo“).innerHTML=voiture.demarrer();

* + Les **tableaux de valeurs** (arrays) seront déclarés indiquant les contenus des cases du tableau entre crochets de la façon suivante : var voitures = [“renault“, “citroen“, “peugeot“] ;

On pourra alors accéder aux éléments du tableau par leur indice en partant de 0 pour le premier élément. Ici voiture[0] a pour valeur « renault ».

* + Les **opérateurs arithmétiques** sont les mêmes qu’en python, y compris avec le symbole modulo « % ». Et vous pourrez effectuer des opérations raccourcies comme en python également : « x +=1 » identique à « x = x+1 ».
  + Les **fonctions** seront sans doute plus utilisées qu’en python, car elles simplifient l’affectation d’une action à un objet HTML : il suffit d’exécuter une fonction pour remplacer une suite d’instructions. La déclaration d’une fonction se présente de la façon suivante :

function maFonction(paramètre1,paramètre2) {

// code à exécuter

return réponse;// éventuellement retour d’une valeur

}

On peut donner autant de paramètres que l’on souhaite, voir aucun (mais il faudra tout de même mettre les parenthèses après le nom de fonction). Comme dans un programme en Python, ces paramètres sont des variables que l’on transmet à la fonction pour qu’elle les utilise dans ses opérations. **On appelle ensuite la fonction avec son nom suivi des parenthèses**, même si elles ne contiennent aucun paramètre !.

* + Les **conditions** pour les tests ressemblent, elles aussi, au python avec la syntaxe :

if (condition1) {

// code à exécuter

}

else if (condition2) {

// code à exécuter

}

else {

// code à exécuter

}

« else if » et « else » sont optionnels et les opérateurs de comparaison identiques au python : égalité (==), différence ( !=)…

* + Les **boucles bornées**sont en revanche assez différentes puisqu’on va déclarer la valeur de départ du compter, la condition d’arrêt et l’incrémentation du compteur au début de la boucle, de la façon suivante :

for (i=0 ;i<10 ;i++){ // code}

* + Les **boucles non bornées** ont une syntaxe plus proche de python :

while (condition) { // code}

Références du langage

L’objectif du cours de NSI n’étant pas de vous former à l’ensemble des possibilités du JavaScript, vous pourrez vous rendre sur le site (anglais) très complet et clair : <https://www.w3schools.com/js/default.asp> pour des explications exhaustives avec de nombreux exercices sur le JavaScript.

Notez au passage que le JavaScript est devenu tellement populaire qu’il en existe une version qui peut être exécutée sur le serveur web pour remplacer PHP (que nous verrons plus tard), nommé « node.js ».

Un mot sur jQuery

Le code JavaScript étant souvent assez long (et lourd) à écrire, une version simplifiée, nommée « jQuery » rencontre un succès croissant (y compris chez Google, Microsoft ou Netflix). C’est une bibliothèque JavaScript qu’il faudra copier auprès de votre page web (à télécharger ici : <https://jquery.com/> ), puis charger en premier avant vos autres scripts avec l’instruction de chargement de script vue plus haut (par exemple : <script src=”jquery-3.5.1.min.js”></script>)

Le code jQuery que vous allez utiliser sera ensuite précédé du signe dollar « $ » pour le distinguer du code JavaScript ‘normal’.

JQuery simplifie ensuite la syntaxe de la façon suivant : par exemple pour modifier le contenu HTML d’un bouton dont l’ « id » est définie à « demo », on utilisera le code :

$("button.demo").html("Nouveau texte")

Au lieu du code JavaScript traditionnel :

onclick='document.getElementById("demo").innerText="Nouveau texte"'

Si cela vous intéresse, vous trouverez une documentation plus complète sur le site x3schools : <https://www.w3schools.com/jquery/default.asp>, mais gardez à l’esprit qu’il vaut mieux maîtriser les bases de JavaScript pour tirer pleinement profit de jQuery.

**b. Exécuter un script sur un bouton**

Voici un exemple du code le plus simple que l’on puisse faire en JavaScript : une page HTML contenant un bouton dont on va changer le texte. Le choix a été fait de mettre le code directement dans l’évènement « onclic » du bouton :

Page HTML avec code JavaScript simple :

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

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

<title>test</title>

</head>

<body>

<p>Ceci est un test de fonction sur un bouton</p>

<button type="button" id="bouton1"

onclick='document.getElementById("bouton1").innerText="bien joué !"'>Cliquez ici</button>

</body>

</html>

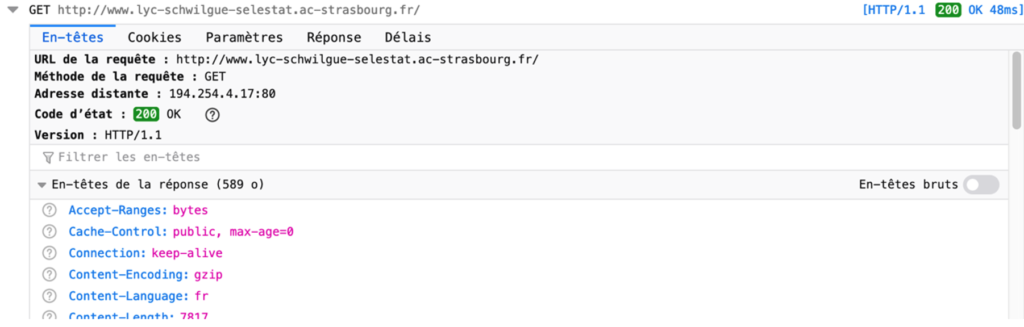
La feuille de style css n’est pas jointe ici, mais elle contient simplement des styles pour améliorer l’apparence du texte et du bouton.

En cliquant sur le bouton, le texte de celui-ci va passer de « Cliquez ici » à « bien joué ! ». Notez l’utilisation du code d’identification **unique** du bouton « id » (bouton1) qui aurait pu nous permettre de modifier n’importe quel autre élément de la page possédant cet « id » en cliquant sur ce bouton.

# Interaction client-serveur

**a. Protocole HTTP**

Lorsqu’une adresse URL est validée par un navigateur Internet, celui va effectuer une requête HTTP (HyperText Transfer Protocol) pour demander l’information au serveur. Le HTTP est donc un protocole client-serveur géré par un logiciel qui se trouve sur le serveur et que l’on nomme serveur HTTP (le plus célèbre étant « Apache »). Il prend appui sur le protocole TCP (vu en classe de SNT en seconde et utilise le port 80 en version standard et le port 443 en version sécurisée (HTTPS). Voici un exemple de requête vers le site d’un lycée :



Les navigateurs modernes permettent de visualiser le contenu de cette demande afin de lire ce qui a été demandé. Il existe deux principales méthodes de requêtes : « GET » pour demander une ressource au serveur (page, image, feuille de style…) et « POST » pour transmettre des informations qui devront être traitées par le serveur (un formulaire de commande par exemple). Cette information apparaît en premier dans la requête.

La méthode de communication est suivie de l’URL. On voit dans l’exemple ci-dessus que cette adresse a été traduite en IPv4 par un serveur DNS.

L’en-tête indique également la version du protocole HTTP, 1.1 ici, ce qui représente la dernière version (en 2019), suivi d’un certain nombre de paramètres :



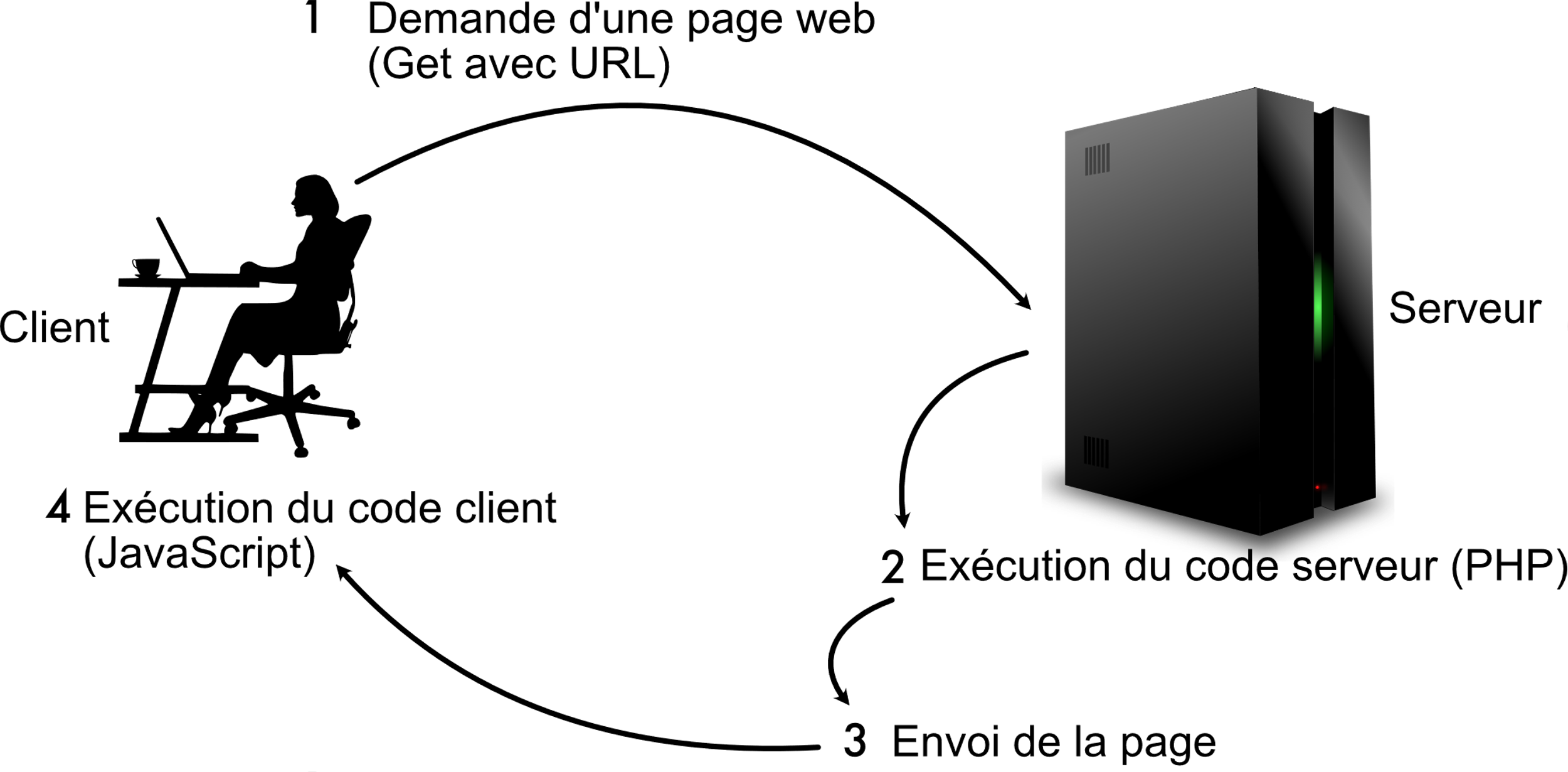
Ces en-têtes indiquent plus précisément quel type de document est demandé (ici on s’attend à du texte ou du HTML et ses variantes), si ces informations peuvent être envoyées sous forme compressée (ici en gzip) ou de quelle page vient l’utilisateur (ici on vient du moteur de recherche Qwant).

Afin d’éviter de faire transiter trop d’informations redondantes sur le réseau, un site peut demander au navigateur de conserver en local des informations données par l’utilisateur, par exemple son identification (pour indiquer qu’il a bien validé sa connexion avec un login et un mot de passe) ou ses préférences de style (si vous offrez le choix du style de votre page avec plusieurs feuilles de style) … Celles-ci sont écrites dans un fichier texte nommé « Cookie » qui restera stocké en local jusqu’à ce qu’il soit effacé, soit automatiquement, soit sur demande de l’utilisateur.

S’il permet de faciliter les échanges, ce cookie peut aussi poser un souci de sécurité quand il conserve des données privées d’un utilisateur, surtout si celles-ci sont lues par un autre serveur à qui ces données n’étaient pas destinées !

**b. Exécution de code sur le serveur**

Pour le moment nous avons vu les moyens de rendre une page web interactive avec du code JavaScript. Cette interactivité a toutefois ses limites, car elle ne permet de travailler qu’avec des informations contenues dans la page chargée par le navigateur du client. Si l’on souhaite avoir des fonctionnalités avancées, comme la personnalisation d’une page en fonction du client pour de la vente en ligne ou l’accès à des données en fonction d’un contexte, il est possible de mettre en place du code qui sera exécuté « avant » d’envoyer la page vers le client.

Le code sur le serveur fonctionne alors de la façon suivante :

Dans l’illustration ci-dessus, on voit que le code côté serveur sera toujours exécuté avant que la page ne soit envoyée au client. Celui-ci ne verra jamais ce code, contrairement au code client JavaScript qui se trouve dans la page web et qui est lisible. Cette confidentialité peut être intéressante pour certains types de traitements. Le code serveur peut aussi accéder à l’ensemble des ressources qui se trouvent sur le serveur, en particulier des bases de données.

Le langage PHP

L’un des langages les plus utilisés pour le code côté serveur est le PHP pour PHP Hypertext Preprocessor (c’est un acronyme récursif). C’est un langage Open Source qui existe depuis 1994 et dont la syntaxe est proche du C. Le code se trouve dans le code source de la page web, comme le JavaScript, mais ne sera visible que si on peut accéder aux fichiers sur le serveur (par exemple avec le protocole FTP si l’on doit gérer un serveur web distant).

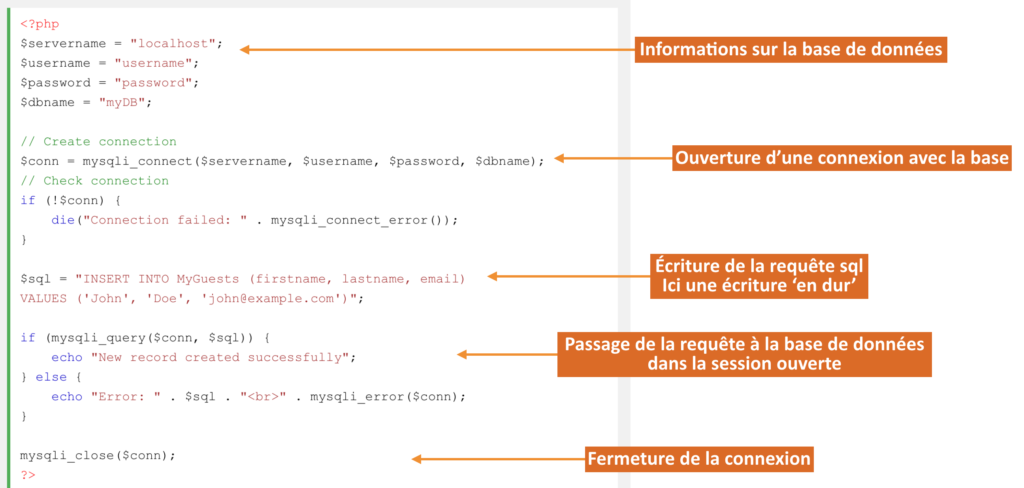
Beaucoup de choses sont possibles en PHP sans base de données :

* La gestion de sessions (pour sécuriser des parties d’un site par exemple)
* La gestion des cookies
* L’assemblage d’une page à partir de ses sous-éléments HTML : en-têtes, menus, pied de page, contenu… sans réécrire ce code pour chaque page…

Mais l’interfaçage avec une base de données permet encore beaucoup plus : créer une seule « page » pour le site pour un contenu qui va se trouver dans une base de données et qui sera appelée selon la demande de l’utilisateur : commentaires, achat en ligne, listes d’élèves…

Mais pour accéder aux bases de données il faudra intégrer des requêtes SQL dans le code PHP. Le SQL (Structured Query Language) est un langage qui permet d’interagir avec une base de données pour y stocker ou en extraire des informations. Le SQL sera vu plus en détail en classe de terminale NSI.

Voici un exemple d’utilisation de PHP avec du SQL :



On voit que le code en PHP se trouve entre les balises « < ?php » et « ?> ». Vous pouvez reconnaître la structure de comparaison « if…else » qui est également semblable à ce qu’on peut trouver dans le JavaScript, ainsi que les points virgules en fin de ligne de code.

Un élément distinctif du PHP est la présence du signe dollar « $ » au début des variables, ainsi que la présence de mots clefs spécifiques pour accéder à certaines fonctions du serveur.

L’exécution d’un script en PHP nécessite d’installer un serveur web en local pour le développement ! Le serveur le plus simple, et Open Source est Apache. On l’associe au plug-in PHP pour exécuter des scripts côté serveur et à MySQL pour gérer des bases de données sur le serveur.

Pour développer sous Windows on peut installer WAMP (<http://www.wampserver.com/> ) ou EasyPHP (<https://www.easyphp.org/>)

**c. Gestion des cookies**

Comme nous l’avons vu plus haut, les cookies sont des fichiers textes qui sont stockés sur les ordinateurs des clients d’un site web. On peut y enregistrer la dernière page vue par un utilisateur pour y retourner à l’ouverture du site, des identifiants pour faciliter la connexion, le contenu d’un panier d’achat…

Comme les cookies sont de simples fichiers textes non cryptés et que l’on peut y accéder dès lors que l’on connaît le nom de ce fichier texte, il faut éviter d’y stocker des données sensibles (mots de passe, informations de payement…), car un autre site web pourrait y accéder (il vaut mieux limiter la lecture des cookies au site, voir plus loin les paramètres « path » et « domain »).

Il est facile de créer un cookie en PHP avec la fonction « setcookie(name, value, expire, path, domain, secure, httponly) » :

* name : nom du cookie.
* value : information que l’on veut stocker dans le cookie.
* expire : date d’expiration d’un cookie au format timestamp UNIX qui indique le nombre de secondes écoulées depuis le 1er janvier 1970. Sans cette information, le cookie expire lorsque l’utilisateur ferme son navigateur (fin de session).
* path : chemin serveur où le cookie sera disponible. Par défaut sa valeur est « / » et le cookie est accessible depuis l’ensemble du serveur web.
* domain : domaine pour lequel le cookie est accessible
* secure : si la valeur est « true », le cookie n’est accessible qu’en https (voir plus loin).
* httponly : si la valeur est « true », le cookie n’est accessible qu’avec le protocole HTTP (et donc pas depuis le JavaScript par exemple).

L’écriture ou la lecture d’un cookie se fait avant le code HTML. Il faut donc mettre le code « setcookie() » du PHP avant le code HTML.

Exemple :

**<?php**

$visit = false;

$idrand = random\_bytes(12);

if (!isset($\_COOKIE['id\_utilisateur'])){

$visit=setcookie('id\_utilisateur',$idrand,time()+3600);

}

**?>**

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

**<?php**

if ($visit) {

echo "Bienvenu à notre nouveau visiteur";

}

else {

echo "Nous sommes heureux de vous revoir";

}

**?>**

</body>

</html>

Dans l’exemple ci-dessus, nous commençons par mettre à « false » la valeur d’une variable « $visit » et par créer un code d’identification aléatoire avec la fonction « random\_bytes » d’une longueur de 12 bit. Nous testons ensuite la présence d’un cookie « id\_utilisateur ». Si celui-ci n’existe pas, nous le créons, ce qui mettra la variable « $visit » à false. Si le cookie avait déjà existé, la création aurait échoué et la variable « $visit » aurait été affectée de la valeur « False ».

Plus loin dans le code il sera possible de savoir si l’utilisateur est nouveau sur la page ($visit vaut alors « True ») ou s’il est déjà venu. Le texte affiché sera modifié en conséquence.

Il est également possible de lire la valeur d’un cookie avec la commande PHP d’accès à la variable « $\_COOKIE » (en majuscule) suivie de son nom. Par exemple on pourrait lire la valeur de l’identification utilisateur précédente avec le code PHP :

echo “votre code utilisateur est “ .$\_COOKIE[‘id\_utilisateur’] ;

Si l’on souhaite modifier la valeur d’un cookie, il faudra à nouveau appeler la fonction « setcookie() ». Pour effacer le cookie il faudra utiliser la même fonction avec une valeur vide : setcookie(‘id\_utilisateur’,’’) ;

**d. Chiffrement HTTPS**

Le protocole HTTP fait transiter toutes les informations entre le serveur et l’utilisateur sous forme de textes lisibles par quiconque peut intercepter les paquets d’informations qui circulent sur le réseau. Cela peut être très dangereux si l’on se connecte avec un mot de passe ou si l’on partage des informations sur une carte de crédit.

Il est donc conseillé d’utiliser plutôt le protocole HTTPS : HyperText Transport Protocole Secure, qui rajoute au protocole un chiffrement des paquets. Le site web sur le serveur doit alors posséder un certificat d’authentification validé par une autorité de certification.

Au moment du dialogue entre un client et un serveur (lorsqu’on cherche à accéder à une page web sécurisée dont l’adresse URL commence par « https:// »), le client demande d’abord une connexion sécurisée et indique au serveur quelles méthodes de chiffrement il peut utiliser. Le serveur choisit alors l’une de ces méthodes, présente son certificat qui permet de savoir qu’il est bien le serveur choisi.

Le client va alors vérifier la validité de ce certificat auprès de l’autorité de certification et si celui-ci est valide, récupère une clé publique (code chiffré) qui va lui permettre de crypter toutes les informations qu’il va transmettre et que seul le serveur en question sera capable de décrypter avec sa clef privée. Dans le même temps, le navigateur du client va générer une clé permettant de l’authentifier et au serveur de coder les informations pour son usage unique.

De cette façon, même si quelqu’un est capable d’intercepter les paquets entre le client et le serveur, ils sont incompréhensibles sans les clés privées du serveur ou du client pour les décrypter.

# Formulaire d’une page web

**a. Éléments de formulaire web**

Dans la partie consacrée au HTML, nous avons vu que nous pouvions introduire de l’interactivité dans une page web en insérant des balises « input » permettant à l’utilisateur de saisir des informations, de faire des choix de dates, de transmettre des fichiers…

Ces éléments ont vocation à être utilisés dans des formulaires qui seront traités par le serveur au moyen de code PHP. Pour cela il faudra utiliser la balise « form » qui possède deux attributs qu’il faudra définir :

* method : qui indique le type de requête que l’on va utiliser (POST ou GET) (voir plus loin)
* action : qui indique où se trouve le script PHP qui va traiter la requête et renvoyer une réponse à l’utilisateur.

Exemple :

<form action="traitement.php" method="post">…</form>

A l’intérieur de cette balise on pourra placer des éléments de saisie, principalement avec les balises « input » ainsi que du texte de mise en forme. Il ne faudra pas oublier d’inclure un élément de type « submit » pour pouvoir soumettre (et transmettre) le formulaire.

Il est évidemment possible d’aller plus loin en ajoutant du code JavaScript pour, par exemple, contrôler que l’utilisateur n’utilise pas certaines adresses e-mail ou pour faire une pré analyse de ses réponses.

N’oubliez toutefois pas que certaines balises ont des contrôles intégrés qui peuvent vous faire gagner du temps : il existe, par exemple, un « input » de type « email » et c’est alors le navigateur qui vérifiera que l’adresse entrée est bien une adresse mail valide. Il existe également des types pour les mots de passe, où les caractères seront remplacés par des étoiles, et même un type « hidden » qui permet de rajouter des valeurs à un formulaire sans que l’utilisateur ne puisse les voir (comme un code d’identification de l’utilisateur ou du formulaire par exemple).

**b. Différence entre requêtes POST et GET**

Nous venons de voir que deux méthodes de transfert du contenu d’un formulaire sont possibles : GET et POST. La façon dont les informations sont transmises et le type d’information à transmettre sont différents dans les deux cas.

GET

La méthode GET devrait être réservée à **l’extraction d’information depuis un serveur**. C’est par exemple la méthode utilisée lorsque vous faites une **recherche** sur un moteur de recherche. Par exemple sur Google, on pourra voir que la méthode utilisée est GET car le contenu de la requête va être visible dans l’URL : <https://www.google.com/search?q=get>

Comme le contenu de la requête est visible dans l’URL, il faut absolument éviter d’utiliser ce type de requête pour transmettre une donnée sensible ! La taille des informations est limitée à la longueur totale d’une adresse (il faut absolument éviter de dépasser 2000 caractères), et c’est bien normal car quelques mots clefs devraient suffire à retrouver une information sur un serveur bien indexé.

POST

La méthode POST est à privilégier lorsque l’on souhaite **enregistrer des informations sur un serveur**, par exemple pour enrichir une base de données. Par défaut, la taille maximale des informations que l’on peut transmettre par POST est de 8 Mo, mais cette taille peut être augmentée dans la configuration du serveur. De plus, dans ce cas les informations ne sont pas transmises dans l’URL et peuvent facilement être cryptées si l’on utilise le protocole HTTPS.

**c. Exemples d’utilisation de formulaires**

Écriture du formulaire HTML

Voici un exemple d’utilisation des formulaires avec une page web HTML qui contient le formulaire nommé « form.html » et la page qui va traiter le formulaire, nommé « traitement.php ». La feuille de style pour ces deux pages n’est pas donnée ici, mais elle n’a aucune importance sur le fonctionnement du formulaire.

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="UTF-8">

<title>Formulaire</title>

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

</head>

<body>

<h3>Merci de remplir le formulaire ci-dessous</h3>

<form action="traitement.php" method="post">

<p><label for="nom">Nom :</label><input type="text" id="nom" name="nom"></p>

<p><label for="prenom">Prénom :</label><input type="text" id="prenom" name="prenom"></p>

<p><label for="mail">E-Mail :</label><input type="email" id="mail" name="mail" required="TRUE"></p>

<p><label for="raison">Raison de votre demande :</label>

<select id="raison">

<option value="signaler">Signaler un problème sur le site</option>

<option value="contacter">Contacter l'auteur du site</option>

<option value="abonnement">M'abonner à la lettre hebdomadaire</option>

<option value="don">Faire un don à l'auteur du site</option>

</select></p>

<p><label for="contenu">Précisez votre demande : </label>

<textarea type="text" maxlength="500" id="contenu" name="contenu"></textarea></p>

<p><input type="submit" value="envoyer"></p>

</form>

</body>

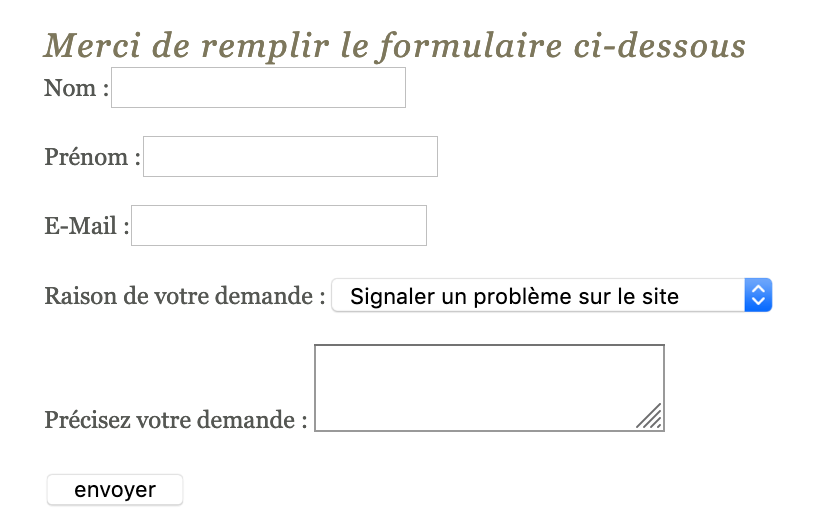
</html>

Dans ce formulaire on utilise les input de type « texte » en leur attribuant une propriété « name » qui pourra être réutilisée lors du traitement par le serveur. La balise « select » permet de faire une liste déroulante avec plusieurs éléments dont un seul pourra être sélectionné. Cet élément est identifié par sa propriété « value ».

Pour créer une zone plus grande de saisie de texte, pour un mail par exemple, on utilise ici la balise « textarea ».

Enfin il faut penser à ajouter un « input » de type « submit » (il est aussi possible de faire cela avec un élément HTML bouton qui sera plus facile à personnaliser avec du CSS : <button class=“submitbtn“ type=“subit“>texte</button>) qui permettra de valider le formulaire pour en envoyer le contenu au serveur.

Voici l’aspect de ce formulaire :



Traitement par le serveur

Une fois que l’utilisateur aura cliqué sur le bouton « envoyer », le serveur va se voir demander la page indiquée dans le paramètre « action » (ici « traitement.php ») et les données envoyées par le formulaire seront stockées dans une variable « $\_GET » ou « $\_POST » selon le paramètre « method » choisi dans le formulaire (ici nous avions choisi « post »).

Dans le script PHP qui se trouve dans la page demandée, « traitement.php », nous pouvons effectuer un traitement sur les données transmises. En pratique, ce traitement sera le plus souvent un stockage dans une base de données (que nous verrons en terminale), mais ici on peut se contenter d’afficher les données reçues par le formulaire avec le code suivant :

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="UTF-8">

<title>Réponse au formulaire</title>

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

</head>

<body>

<strong>Votre demande a bien été transmise</strong>

<p>Vous nous avez envoyé les informations suivantes :</p>

**<?php**

echo "<p>Nom : ".$\_POST["nom"]."</p>";

echo "<p>Prénom : ".$\_POST["prenom"]."</p>";

echo "<p>E-Mail : ".$\_POST["mail"]."</p>";

echo "<p>Raison de votre demande : ".$\_POST["raison"]."</p>";

echo "<p>Votre texte : ".$\_POST["contenu"]."</p>";

**?>**

</body>

</html>

Notez les crochets après « $\_POST[“champ“] » car cette variable est une matrice (array) qui indexe les informations passées par le formulaire. Il est donc possible d’accéder à chaque valeur par son nom et de s’en servir dans le code PHP, par exemple pour l’afficher comme ici.