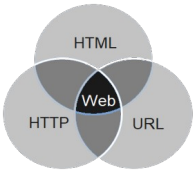
Interactions homme-machine sur le Web

Interaction client-serveur, requête HTTP, méthodes GET et POST

Le Web

Les inventeurs du Web, Tim Berners-Lee et Robert Caillau, ont défini au CERN entre 1989 et 1991 les trois piliers sur lesquels repose le web : HTTP, URL et HTML :



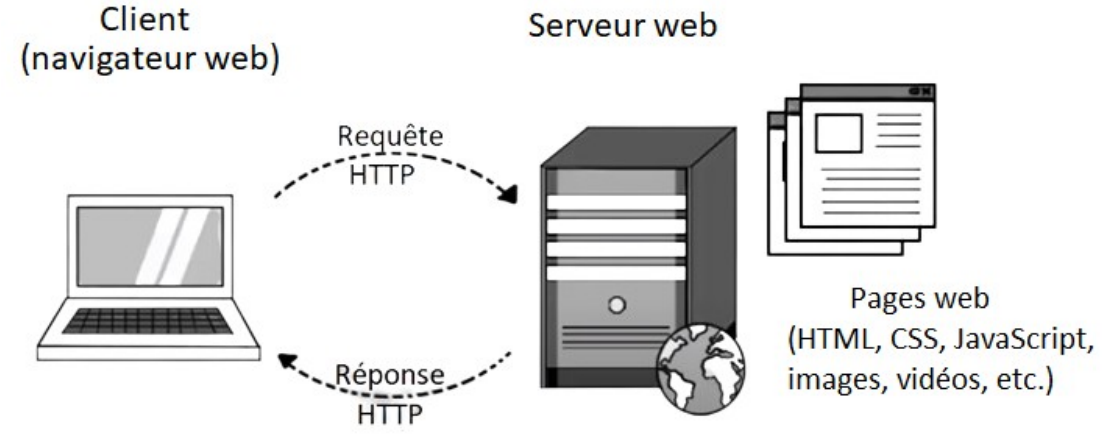
- Le protocole de transfert **HTTP** (*HyperText Transfer Protocol*) ou **HTTPS** en version sécurisée ;
- le langage **HTML** (*HyperText Markup Language*); et
- les **URL** (*Uniform Resource Locator*) couramment appelée adresse web.

On a vu dans une activité précédente comment écrire une page web en utilisant le **langage HTML** (un langage de balisage) accompagné de **code CSS** (mise en forme) et de **programmes Javascript** (programmation de pages dynamiques). Les fichiers permettant à la page de s'afficher étaient stockées localement sur notre ordinateur. Mais sur internet, les fichiers des sites web que l'on consulte ne sont pas à priori sur notre ordinateur, alors comment arrivent-ils jusqu'à nous ?

HTTP et le modèle client-serveur

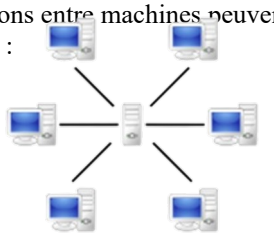
Le Web fonctionne selon le modèle client/serveur[1] : la machine client demande à la machine serveur une ressource identifiée par son adresse URL.

- Les fichiers d'une page web (fichiers HTML et CSS, les images, vidéos et autres ressources) sont stockés sur un ordinateur distant : le **serveur web**.
- L'utilisateur consulte cette page web sur sa machine (ordinateur, tablette, smartphone) via un **navigateur** (Mozilla Firefox, Google Chrome, Safari, etc.) : le **client**.
- Le client fait des **requêtes** au serveur, et le serveur lui **répond** en renvoyant les fichiers de la page web. Ces échanges se font **par internet** en utilisant le **protocole HTTP** (ou **HTTPS** en version sécurisée).

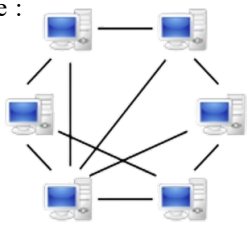


1 Sur internet, les communications entre machines peuvent être organisées selon deux types d'architecture :

Le modèle client-serveur :
Les ordinateurs clients demandent des ressources à un serveur centralisé.



Le pair-à-pair
(*peer-to-peer* ou P2P en anglais)
Les ordinateurs interconnectés partagent les ressources entre eux sans avoir recours à un serveur centralisé.



URL

Une **URL**, pour *Uniform Resource Locator*, couramment appelée **adresse web**, permet **de localiser une ressource** (un document, une image, une vidéo, etc.) **sur le Web avec le protocole utilisé**.

Une URL se compose de différentes parties dont certaines sont obligatoires et d'autres optionnelles :

`https://fr.wikipedia.org/wiki/World_Wide_Web/`

Le **protocole** utilisé, souvent HTTP ou HTTPS sa version sécurisée.

Le **nom de domaine** du serveur web.

Le **chemin** sur le serveur web vers la ressource.

`https://www.google.fr/search?q=nsi&source=hp`

Des **paramètres** supplémentaires sous forme de paires de clé = valeur séparées par « & ».

`https://fr.wikipedia.org/wiki/Informatique#Algorithmique`

Une **ancre** qui désigne un endroit donné de la ressource.

Serveur web

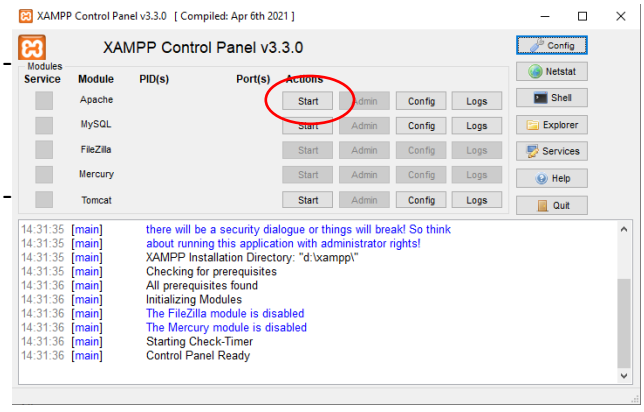
Le rôle du **serveur web** est de répondre à des **requêtes de postes clients** (généralement des navigateurs web tels que Edge, Firefox, Chrome, etc .) à travers un réseau public (Internet) ou privé (intranet), en utilisant le protocole HTTP (la plupart du temps). Le terme serveur web désigne à la fois :

- Un **serveur informatique** mis à disposition par un **hébergeur web**. Les hébergeurs sont des entreprises commerciales (IONOS, Hostinger, LWS, etc.) avec des offres gratuites ou payantes par exemple pour avoir son propre nom de domaine ou plus de fonctionnalités (site WordPress, etc.).
- Un **logiciel** sur le serveur qui prend en charge **les requêtes HTTP** de postes clients et permet d'exécuter des programmes, souvent en **PHP**, pour générer des **pages web dynamiques**. Les plus courants sont **Apache et Nginx**.

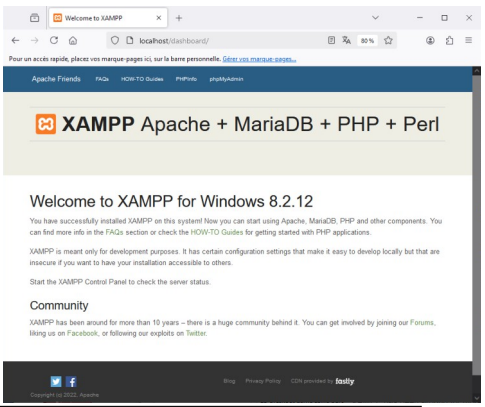
On utilise dans la suite de cette activité notre propre ordinateur à la fois comme logiciel de serveur web et comme client (un navigateur). La partie serveur web local est réalisée avec XAMPP[2] , une suite logiciel comprenant Apache et PHP souvent utilisée pour écrire des applications avec des échanges réseaux, en particulier par les développeurs web.

1 Ouvrir l'application xampp-control puis démarrer le serveur Apache en cliquant sur le bouton Start .

Inutile de démarrer les autres services.



2 Quand Apache apparaît sur un fond vert, ouvrir un navigateur de votre choix et afficher la page : http://localhost/ .



La page de présentation de XAMPP s'affiche.

« *localhost* » (pour hôte local) est associé à l'adresse IPv4 127.0.0.1. Il fait référence à l'ordinateur qu'on utilise quand on envoie une requête HTTP à notre propre ordinateur agissant en tant que serveur.

2 https://www.apachefriends.org/

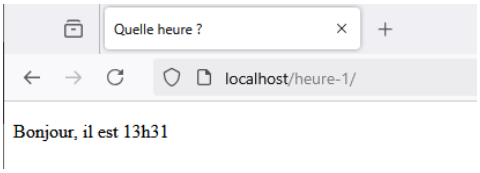
3 Ajouter dans xampp/htdocs un nouveau dossier, appelé « heure-1 », puis créer dans ce dossier un nouveau fichier index.html avec le code HTML suivant :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Quelle heure ?</title>
  </head>
  <body>
    <p>Bonjour, il est <span id="clock"></span></p>
    <script>
      var d = new Date();
      document.getElementById("clock").textContent = d.getHours()+"h"+d.getMinutes();
    </script>
  </body>
</html>
```

Par défaut, le premier fichier d'un projet web s'appelle index.html, c'est le fichier qui se charge en premier quand on saisit l'URL du site.

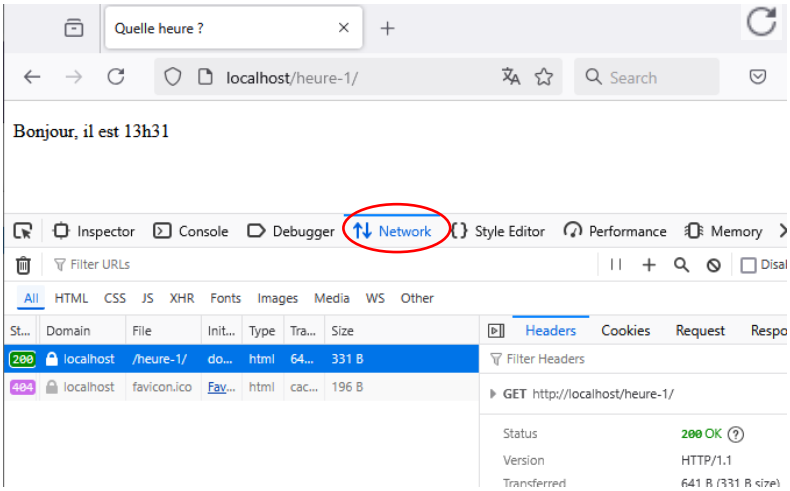
Bien sûr, on peut lire ce fichier HTML dans un navigateur et observer l'affichage obtenu, mais l'idée ici est de l'obtenir par HTTP depuis le serveur web Apache local.

4 Ouvrir dans un navigateur la page depuis le serveur web : http://localhost/heure-1/ (ou http://127.0.0.1/heure-1/) et vérifier qu'on obtient l'affichage attendu.



Requêtes HTTP et réponses du serveur

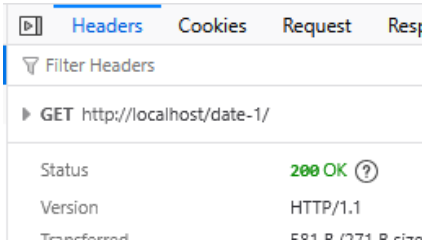
5 Toujours sur cette page, ouvrir les outils de développement avec la touche F12 (ou les touches CTRL + Maj + I simultanément).



Sélectionner l'onglet "Network" ou "Réseau", puis actualiser la page web avec .

La requête HTTP (/heure-1/) du navigateur (le client) au serveur web Apache local pour demander le fichier de la page web s'affiche.

6 Cliquer sur cette requête et observer **l'entête de la requête HTTP** dans l'onglet « Headers » ou « En-têtes » qui contient des informations sur la requête.



7 Quelle méthode est utilisée pour cette requête ?

GET

8 Quel est le code d'état (« Status Code ») de la réponse HTTP ?

200 OK

9 On voit dans l'onglet « Response » ou « Réponse » la réponse du serveur renvoyée au client (passer en mode brut Brut sur Firefox). Qu'a renvoyé le serveur web ?

Le code HTML de la page web.

Le code renvoyé par le serveur au client contient-il le programme JavaScript « `<script> var d…` » ?

Oui il y est

10 Modifier légèrement l'URL dans la barre d'adresse pour demander une ressource qui n'existe pas, par exemple http://localhost/xyz/. Quel est le code d'état (« Status Code ») de la réponse HTTP ?

404

Les réponses à une requête HTTP (ou HTTPS) utilisent un code d'état qui peut prendre par exemple les valeurs : **200** : succès de la requête ; **403** : accès refusé ; **404** : ressource non trouvée .

PHP

Le précédant fichier contient du code HTML avec un programme écrit en JavaScript. Un fichier PHP est aussi capable d'exécuter des programmes, c'est un **fichier HTML « amélioré »**.

11 Ajouter dans xampp/htdocs un nouveau dossier, appelé « heure-2 », puis créer dans ce dossier un nouveau fichier index.php avec le code suivant :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Quelle heure ?</title>
  </head>
  <body>
    <?php
    echo "Bonjour, il est " . date("H:i");
    ?>
  </body>
</html>
```

Les programmes sont écrits à l'intérieur de balises **<?php** et **?>**. L'instruction la plus importante en PHP est **echo** qui permet d'insérer du texte dans la page HTML finale. Il est possible de mélanger texte et valeurs d'expressions en les reliant par des « . ».

12 Ouvrir dans un nouvel onglet du navigateur la page depuis le serveur web : http://localhost/heure-2/. Qu'affiche cette nouvelle page ?

Bonjour, il est 13h35

13 Observer la réponse du serveur dans les outils de développement (touche F12 > onglet "Network" ou "Réseau" > actualiser la page web > cliquer sur la requête HTTP > l'onglet « Response » ou « Réponse ».

Le code renvoyé par le serveur au client contient-il le programme PHP « `<?php echo "Bonjour..` » ?

Non, le programme a été interprété par le serveur. Le serveur n'envoie que du code HTML.

14 Actualiser à nouveau la page web avec ↻ et observer le code envoyé par le serveur. Que constate-t-on ?

Le code HTML a été mis à jour avec la nouvelle heure.

PHP (pour "*PHP: Hypertext Preprocessor*") est un langage de programmation utilisé pour générer des pages web dynamiques. Contrairement à JavaScript, un programme PHP s'**exécute sur le serveur** pour **produire du code HTML** qui est ensuite envoyé au client.

Formulaires dans une page web

On a vu dans une activité précédente l'utilisation de formulaires pour interagir avec l'utilisateur. Les informations saisies étaient traitées par **un programme JavaScript**

Le programme JavaScript s'exécute sur la machine de l'utilisateur, ce qui n'est pas possible dans de nombreuses situations : saisie d'identifiant de connexion, consultation de compte bancaires, achats en ligne, forum, réseaux sociaux, etc. **Les données doivent être traitées sur le serveur : c'est le rôle de PHP**.

15 Ajouter dans le dossier xampp/htdocs un nouveau dossier, appelé « heure-3 », puis créer dans ce dossier un nouveau fichier index.php avec le code suivant :

```
<html>
  <head>
    <title>Quelle heure ?</title>
  </head>
  <body>
    <form action="bonjour.php" method="GET">
       Nom: <input type="text" name="nom">
       Prénom: <input type="text" name="prenom">
       <input type="submit" value ="Envoyer">
    </form>
  </body>
</html>
```

Création d'un formulaire. Les données seront envoyées vers la page définie par l'attribut `action` (« bonjour.php ») en utilisant la méthode définie par l'attribut `method` (GET).

Deux zones de saisie de texte, appelées « `nom` » et « `prenom` ».

Bouton pour envoyer les données du formulaire vers la page définie par l'attribut `action`.[3]

16 Créer un second fichier appelé « bonjour.php » dans le même dossier avec le programme PHP suivant :

```
<?php
$p = $_GET["prenom"];
$n = $_GET["nom"];
echo "Bonjour $p $n, il est " . date("H:i");
?>
```

17 Ouvrir dans un nouvel onglet du navigateur la page http://localhost/heure-3/. Qu'affiche cette page ?

Nom [ ] Prénom [ ] Envoyer

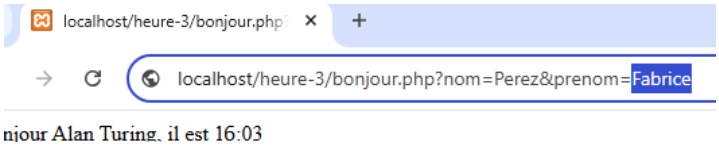
18 Saisir un nom et un prénom dans le formulaire, par exemple Turing et Alan, puis cliquer sur le bouton Envoyer. La page « bonjour.php » s'ouvre. Qu'affiche-t-elle ?

Bonjour Alan Turing, il est 15:12

19 Observer l'URL de la page affichée dans la barre d'adresse. Comment le navigateur a envoyé les valeurs « Alan » et « Turing » à la page « bonjour.php » sur le serveur ?

Dans des paramètres ajoutés à la fin de l'URL :
http://localhost/heure-3/bonjour.php?nom=Turing&prenom=Alan

20 Modifier directement les paramètres de l'URL dans la barre d'adresse en remplaçant Turing et Alan par vos nom et prénom et appuyer sur Entrée. Qu'affiche la page ? À votre avis, que se passe-t-il sur le serveur lorsqu'il reçoit la requête HTTP ?



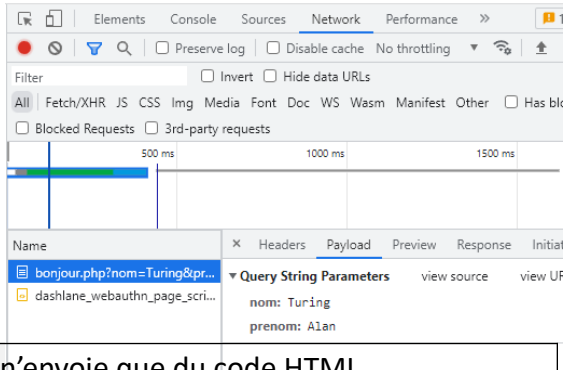
Le programme PHP de « bonjour.php » construit une nouvelle page HTML sur le serveur avec mon nom et prénom et la renvoie au client.

3 On peut aussi utiliser l'élément : `<button type="submit">Envoyer</button>`

21 Toujours sur la page « bonjour.php », ouvrir les outils de développement (F12 ou CTRL + Maj + I) et sélectionner l'onglet "Network" ou "Réseau" et rafraîchir la page.

22 Cliquer sur la requête de la page « bonjour.php » faite au serveur web et observer dans l'onglet « Response » ou « Réponse » la réponse du serveur renvoyée au client.

La réponse du serveur envoyée au client contient-elle le programme PHP « `<?php echo "Bonjour.. » ?` »



Non, le programme a été interprété par le serveur. Le serveur n'envoie que du code HTML.

Lorsqu'un navigateur fait une requête **avec la méthode GET, il peut transmettre au serveur web des paramètres en ajoutant à la fin de l'URL des couples clé = valeur séparés par « & » :**

?param1=val1&param2=val2... .

Ces valeurs peuvent être utilisées sur le serveur dans un programme PHP en écrivant `$_GET['param1']`.

Méthodes GET et POST

On a vu dans l'activité précédente que l'attribut type de l'élément `<input>` permet de personnaliser un formulaire[4], en particulier qu'il permet de saisir un mot de passe de façon discrète en affichant des points à la place des caractères saisis.

23 Modifier le fichier index.php afin que l'utilisateur saisisse aussi un mot de passe (123456) pour avoir l'heure :

```
<html>
  <head>
    <title>Quelle heure ?</title>
  </head>
  <body>
    <form action="bonjour.php" method="GET">
       Nom: <input type="text" name="nom">
       Prénom: <input type="text" name="prenom"><br>
       Mot de passe : <input type="password" name="mdp">
      <button type="submit">Envoyer</button>
    </form>
  </body>
</html>
```

et ajouter le code de vérification du mot de passe saisi dans « bonjour.php » :

```
<?php
$p = $_GET["prenom"];
$n = $_GET["nom"];
$mdp = $_GET["mdp"];
if ($mdp == '123456') {
  echo "Bonjour $p $n, il est " . date("H:i");
} else {
  echo "Mauvais mot de passe !";
}
?>
```

4 Voir la liste complète sur https://developer.mozilla.org/fr/docs/Learn/Forms/Basic_native_form_controls

24 Rouvrir la page http://localhost/heure-3/ dans le navigateur pour vérifier qu'elle s'affiche correctement.

Remplir le formulaire avec un nom, prénom et le mot de passe (123456). Quelle différence voit-on dans le formulaire entre les nom et prénom et le mot de passe ?

Le mot de passe n'apparaît pas en clair dans le formulaire, les caractères sont remplacés par des ronds noirs.

25 Cliquer sur Envoyer et observer les paramètres passés dans l'URL. Que constate-t-on au sujet du mot de passe envoyé au serveur ? Pourquoi la transmission du mot de passe n'est pas satisfaisante ?

Il apparaît en clair dans l'URL. Ce n'est pas satisfaisant car il peut être vu dans l'historique ou en observant l'écran de l'utilisateur.

26 Remplacer dans « index.php » la méthode utilisée pour faire la requête de la page « bonjour.php » avec la méthode POST au lieu GET :

```
<form action="bonjour.php" method="POST">
```

et les variables `$p = $_POST["prenom"]`, `$n = $_GET["nom"]` et `$mdp = $_GET["mdp"]` dans « bonjour.php ».

27 Rouvrir la page http://localhost/heure-3/ dans le navigateur et saisir à nouveau le nom, prénom et mot de passe demandé.

28 La page bonjour.php apparaît avec le nom et le prénom entré (si le mot de passe est correct). Observer l'URL de la page affichée. Est-ce que les paramètres (nom, prenom, mdp) sont toujours visibles dans l'URL ?

Non ils n'apparaissent plus dans l'URL

Le protocole **HTTP** (ou **HTTPS** en version sécurisée) permet d'obtenir un fichier ou une ressource depuis un serveur. Plusieurs **méthodes** peuvent être utilisées, les plus courantes sont :

- **GET** : les paramètres passés au serveur sont **visibles dans l'URL sous la forme cle=valeur** (et donc éventuellement dans l'historique de navigation et le cache). La méthode **GET n'est pas recommandée pour un formulaire contenant un mot de passe**.
- **POST :** les paramètres sont passés **dans le corps de la requête**, ils n'apparaissent pas dans l'URL.

Il existe d'autres différences sur le traitement des paramètres avec les méthodes GET et POST :

	GET	**POST**
Visibilité des paramètres	Visible pour l'utilisateur dans l'URL.	Invisible pour l'utilisateur.
Marque-page et historique de navigation	Les paramètres sont stockés en même temps que l'URL.	L'URL est enregistrée sans paramètres.
Actualisation du navigateur / Bouton « précédent »	Les paramètres de l'URL ne sont pas envoyés à nouveau.	Le navigateur avertit que les données du formulaire doivent être renvoyées.
Type de données	Caractères ASCII uniquement.	Caractères ASCII et binaires (image, son, vidéo, etc.).
Longueur des données	Limitée - longueur maximale de l'URL à 2 048 caractères.	Illimitée.

29 Quelle méthode HTTP sera utilisée pour transférer un fichier image ou vidéo entre un client et un serveur ?

POST car les données sont en binaire et de taille trop grande pour GET.

HTTP et HTTPS

Si l'utilisation de la méthode POST permet de ne pas afficher les paramètres dans la barre de tâche, cela ne suffit pas à sécuriser les données échangées entre le client et le serveur, il faut en plus **chiffrer les échanges** en utilisant l**e protocole HTTPS.**

Quand un navigateur (un client) communique avec un serveur web, ils s'échangent des données sur internet à travers un grand nombre de routeurs (c'est le routage).

- Avec HTTP, les échanges ne sont pas chiffrés, ils peuvent être interceptés et lus par un pirate informatique .
- **Avec HTTPS, les échanges sont chiffrés**, le « S » à la fin signifie qu'ils sont sécurisés grâce au protocole TLS (*Transport Layer Security*), le successeur de SSL (*Secure Sockets Layer*). Un pirate informatique ne peut donc pas les lire et récupérer des informations sensibles.
- Pour utiliser HTTPS, un serveur web doit pouvoir être **authentifié par un certificat SSL obtenu auprès d'une autorité de confiance**, reconnue de tous.

30 Observer dans un navigateur le site de l'école. Par quel autorité de confiance a-t'il été émis ? Jusqu'à quand est-il valide ?

Il est émis par Let's Encrypt. Il est valide jusqu'au xx xxx xxx.

On peut facilement observer quels sites web utilisent quel protocole :

- Le début de l'URL d'un site web indique le protocole utilisé : http://... ou https://...
- Une icône à gauche de l'URL dans un navigateur indique un cadenas ouvert (http) ou fermé (https).
- La plupart des navigateurs bloque l'accès à un site HTTP avec un message d'alerte .

Il faut donc **toujours vérifier qu'un site utilise bien HTTPS** avant d'y entrer un mot de passe ou des données sensibles et se rappeler que **seule la communication est sécurisée** : rien n'assure que les données confidentielles fournies au site sont elles-mêmes protégées sur le serveur et ne seront pas piratées.

Aller plus loin avec PHP

On peut écrire du code PHP à plusieurs endroits dans une même page, à chaque fois à l'intérieur de balises **<?php** et **?>**. L'instruction `echo` permet d'écrire du texte HTML, y-compris les balises :

```
<php? echo "J'<i>aime</i> PHP !" ?>
```

Comme en JavaScript, les instructions PHP sont séparées par des « ; » et les blocs sont délimités par des accolades, mais il y a quelques autres différences, par exemple tous les noms de variable commencent toujours par un « $ » en PHP !

31 Compléter ce formulaire qui demande deux nombres et affiche le plus grand.

index.html

```
<html>
 <head>
  <title>Plus grand</title>
 </head>
 <body>
  <form action="plusgrand.php" method="POST">
     nombre: <input type="text" name="n1">
     nombre: <input type="text" name="n2">
     <button type="submit">Envoyer</button>
  </form>
 </body>
</html>
```

plusgrand.php

```
<?php
 $n1 = $_POST["n1"];
 $n2 = $_POST["n2"];
 if ($n1 > $n2) {
  echo "<p>$n1 est plus grand que $n2</p>";
 } elseif ($n1 < $n2) {
  echo "<p>$n2 est plus grand que $n1</p>";
 } else {
  echo "<p>Les deux nombres sont égaux</p>";
 }
?>
```

32 Écrire un formulaire qui demande deux nombres et une opération (+, -, *, /) puis affiche le résultat.