CHAPITRE 6 : Interactions sur le web

CHAPITRE 6 : Interactions sur le web

1 Modalités de l'interaction Homme-Machine - Evènements

1.1 Histoire

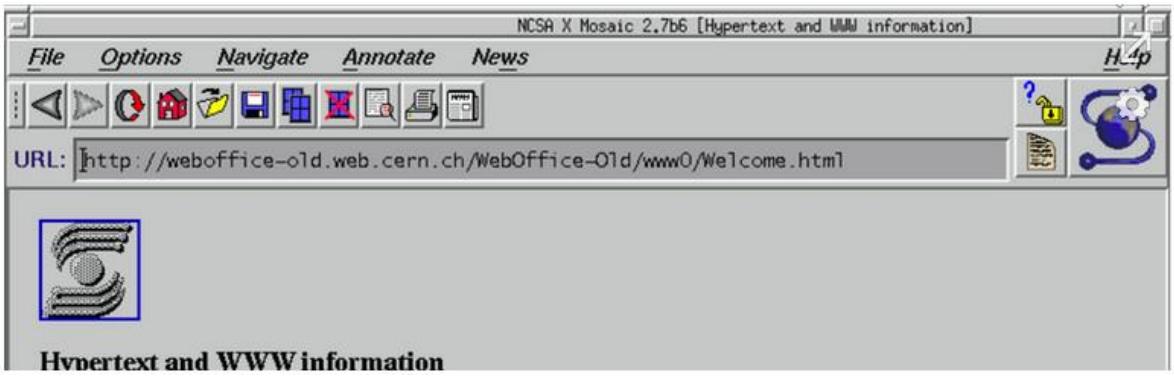
1989

Depuis 1983 les protocoles TCP et IP permettent à des ordinateurs en réseau de communiquer. Mais la recherche d'une information donnée sur le réseau est difficile. En 1989 deux chercheurs du CERN à Genève (Tim Berners-Lee et Robert Cailliau) mettent au point le protocole ***HTTP*** (HyperText Transfer Protocol) qui permet à un navigateur situé sur la machine cliente (celle de l'utilisateur) de demander à une machine hôte (sur un site) d'envoyer une page écrite en langage ***HTML*** (Hyper Text Markup Language). Enfin, pour retenir facilement l'adresse d'une ressource (par exemple le fichier de la page d'accueil d'un site) sur une machine distante, les adresses IP sont traduites en adresses avec des mots : les ***URL*** (Uniform Resource Locator).

- Ce **système hypertexte** public fondé sur http, HTML et URL s'appellera le ***www*** (World Wide Web) c'est à dire toile (d'araignée) mondiale.

1993

- Le web compte 623 sites. Lancement du premier navigateur web doté d'une interface graphique : **Mosaïc**. C'est ce navigateur qui rend le web populaire.



1994

Lancement du navigateur web **Netscape Navigator**. Il gère les **cookies**, le protocole sécurisé **SSL** et le langage **JavaScript** qui, lorsqu'il est présent dans les pages HTML envoyées par le serveur aux clients, permet à l'utilisateur d'interagir avec son écran (en plus des liens hypertextes). Grace au JavaScript, l'utilisateur peut par exemple remplir un formulaire ou cliquer sur un bouton pour envoyer sa réponse au serveur.

1997

Apparition de l'ensemble des protocoles **wifi** qui permettent de se connecter à un réseau sans câble.
C'est aussi le **début du commerce en ligne**.

1.2 Composants graphiques d'une page web.

- Afin de permettre à l'utilisateur d'interagir avec la page web affichée, des composants graphiques (champs de formulaires) peuvent être mis en place sur une page web. On trouve une grande variété de champs de formulaires. Par exemple :
    - Champ input de type button
    - Champ input de type text
    - Champ input de type radio
    - Champ input de type password
    - Champ input de type checkbox
    - Champ select
    - Champ input de type number

Composants graphiques	Aspect	Composant graphique HTML
• Le bouton poussoir		`<input type="button" …`
• La zone de texte		`<input type="text" …`
• Le bouton radio		`<input type="radio" …`
• La zone mot de passe		`<input type="password" …`
• La case à cocher		`<input type="checkbox" …`

• Le menu déroulant		`<select name="menu">`
		`<option value="volaille">Poulet fr...`
		`<option value="veggie">Riz et ...`
		`<option value="poisson">Anchois...`
		`</select>`

• La zone de nombres		`<input type="number" min="2" max="36" value="2"…`

A l'intérieur de chaque balise <input>, <select> etc on trouve des "attributs" comme name="...", value="...", min="..." etc. qui permettent de préciser le comportement des composants graphiques.

- Afin de pouvoir désigner chaque composant graphique, on peut lui attribuer **un identifiant *unique*** à l'aide de **l'attribut id**, par exemple `<input type="text" id="texte_0"`...

1.3 Lier des balises HTML à des fonctions JavaScript

- Afin de créer une interaction entre la page et l'utilisateur, on prévoit que lorsqu'un "**évènement**" survient (par exemple clic sur un bouton, survol de la souris au-dessus etc.) un bloc de code (une **fonction JavaScript**) sera exécuté.

Logo de JavaScript

Pour réaliser cela, on associe une fonction à une balise (<body>, <img>, <p>, <input>, etc.) grâce à un attribut appelé **gestionnaire d'évènement**. En voici quelques-uns :

Attribut **gestionnaire d'évènement**	Effet	Exemple
onchange	Déclenche l'exécution de la fonction lorsque le composant a été changé.	`<input type="number" onchange="convert('gbp')" …` Si l'utilisateur change le nombre, alors la fonction `convert('gbp')` est exécutée.
onclick	Déclenche l'exécution de la fonction lorsque l'utilisateur clique dessus.	`<input type="button" onclick="Calcul()" …` Si l'utilisateur clique sur le bouton, alors la fonction `Calcul()` est exécutée.
onkeypress	Déclenche l'exécution lorsque l'utilisateur appuie sur une touche du clavier.	`<input type="text" id="entier" onkeypress="maFonction()"` Si l'utilisateur clique dans la zone de texte et qu'il appuie sur une touche du clavier, alors `maFonction()` est exécutée.
onload	Déclenche l'exécution dès que la page a été chargée par le navigateur.	`<body onload="init()">` A l'ouverture de la page la fonction `init()` est exécutée.
onmouseover	Déclenche l'exécution de la fonction lorsque la souris survole un élément.	`<img src="smiley.jpg" onmouseover="bigImg()" …` Si la souris passe au-dessus de l'image alors la fonction `bigIm()` est exécutée.
onmousedown	Déclenche l'exécution lorsqu'un bouton de la souris est appuyée **et** que la souris survole l'élément.	`<p id="monParagraphe" onmousedown="maFonction()" …` Si la souris est au-dessus du paragraphe **et** que l'utilisateur appuie sur un bouton de la souris alors la fonction `maFonction()` est exécutée.

2 Interactions avec l'utilisateur dans une page web

2.1 Principe du fichier JavaScript séparé

Les fonctions Javascript référencées dans la page reçue par le navigateur vont être exécutées **sur le poste client**. Elles peuvent faire des calculs et agir sur les composants de la page web elle-même. Il est préférable[1] d'enregistrer les fonctions JavaScript dans un fichier séparé de la page HTML. Ce fichier aura pour extension **.js**

[1] Il est cependant possible d'inclure du script JavaScript à la fin du corps de la page HTML avant la balise </body>. On encadre le script entre les balises. <script> et </script>. Des exemples sur ostralo.net/javascript

Si dans un dossier, on a déjà enregistré **convertisseur.html** et **style.css** alors on ajoute **script.js.** A la fin du corps du fichier HTML, **juste avant </body>** on écrit la balise **script** qui indique le fichier des fonctions JavaScript.

convertisseur.html
script.js
style.css

Exemple

Les styles sont dans ce fichier

Les fonctions JavaScript sont dans ce fichier

***Fichier* style.css**

```css
body {
  font-family: Georgia, "Times New Roman",
       Times, serif;
  color: purple;
  background-color: #38daad }
h3 {
  font-family: Helvetica, Geneva, Arial,
       SunSans-Regular, sans-serif }
```

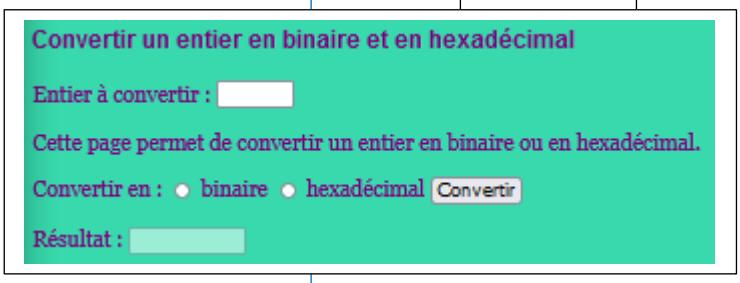
***Fichier* script.js**

```js
function init() {
  document.getElementById("date").innerHTML = Date();
}

function Calcul() {
  entier = Number(document.getElementById("entier").value);

  if (document.getElementById("binaire").checked) {
    let nbBinaire = entier.toString(2);
    document.getElementById("resultat").value = nbBinaire;
  }

}
```

***Fichier* convertisseur.html**

```html
<!doctype html>
<html>

<head>
  <meta charset="UTF-8">
  <title>Convertisseur</title>
  <link href="style.css" type="text/css" rel="stylesheet">
</head>



<body onload="init()">

  <h3>Convertir un entier en binaire et en hexadécimal</h3>

  <p>Entier à convertir : <input type="text" size="5" id="entier" maxlength="5"></p>
  <p>Cette page permet de convertir un entier en binaire ou en hexadécimal.</p>
  <p>Convertir en :
    <input type="radio" name="choix" id="binaire"> binaire
    <input type="radio" name="choix" id="hexa"> hexadécimal
    <input type="button" onclick="Calcul()" value="Convertir">
  </p>
  <p>Résultat : <input type="text" size="10" id="resultat" disabled="disabled"></p>
  <script src="script.js"></script>
</body>

</html>
```

Affichage :



2.2 Comment écrire une fonction en JavaScript ?

Faisons le parallèle entre une fonction en Python et une fonction en JavaScript :

`def calcul(a):`
`  b = a*a`
`  return b`
	`function calcul() {`
	`  let a = document.getElementById("texte1").value;`
	`  let b = a * a;`
	`  document.getElementById("resultat").value = b;`
	`}`

- JavaScript est un langage de programmation basé sur le navigateur web.
- Le corps des fonctions est entre une accolade **ouvrante au début** et une accolade **fermante à la fin**.
- On retrouve dans le ***if*** le ***for*** et le ***while*** l'accolade ouvrante et l'accolade fermante.
- A la fin d'une ligne on termine par un ***point virgule***.
- Si un nom de variable est formé de plusieurs mots, alors ils sont attachés et commencent par une majuscule, sauf le premier mot. Exemple : `nombreDeLivres`
- Si une variable est déclarée sans mot clé, à l'intérieur ou non d'une fonction, sa portée est globale.
- Avec le mot clé ***var*** si la variable est déclarée dans une fonction alors elle est locale à cette fonction.
- ***let*** donne à la variable une portée locale à un bloc d'instructions limité par { ... } .

`document.getElementById("texte1")` désigne l'élément ayant pour identifiant unique `"texte1"`

`document.getElementById("texte1").value` est la valeur de l'élément (ex. le texte de la zone).

Exemple 1

***Fichier* `style.css`**

```
body {
  font-family: Georgia;
}
```

***Fichier* `script.js`**

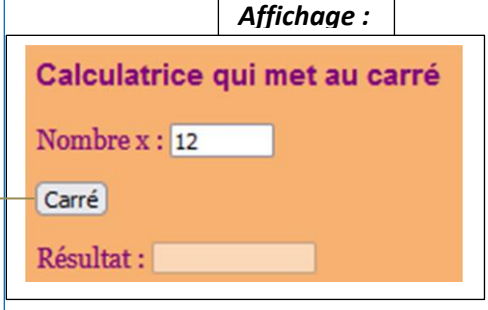
```
function calcul() {
  let a = document.getElementById("texte1").value;
  let b = a * a;
  document.getElementById("resultat").value = b;
}
```

***Fichier* `calculatrice.html`**

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Convertisseur</title>
  <link href="style.css" type="text/css" rel="stylesheet">
</head>
<body>
  <h3>Calculatrice qui met au carré</h3>
  <p>Nombre x : <input type="text" size="5" id="texte1" maxlength="5"></p>
  <p><input type="button" onclick="calcul()" value="Carré"> </p>
  <p>Résultat : <input type="text" size="10" id="resultat" disabled="disabled"></p>
<script src="script.js"></script>
</body>
</html>
```

L'évènement `onclick` exécute `"calcul()"`

Affichage :



Utilisation de la console dans l'interface " Programmation Web " sur le site cahier-nsi.fr

Pour déboguer un programme JavaScript en cours d'écriture, on peut afficher une valeur de variable dans la console. On insère par exemple dans la fonction `calcul()` la ligne **`console.log("b vaut : ", b)`**

Exemple 2

- Le contenu d'une balise dans le document HTML peut être modifié à l'aide de la propriété **innerHTML** de cette balise. Attention : le nouveau contenu remplace le précédent.

***Fichier* `style.css`**

```
body {
  font-family: Georgia;
}
```

***Fichier* `script.js`**

```
function affiche() {
document.getElementById("texte").innerHTML = "Hello !";
}
```

***Fichier* `innerhtml.html`**

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>innerHTML</title>
  <link href="style.css" type="text/css" rel="stylesheet">
</head>
<body>
  <h5>Essai de document.getElementById("texte").innerHTML</h5>
  <p id="texte" onclick="affiche()">
  Cliquez moi ! <br>
  pour changer le texte d'une balise (innerHTML).
  </p>
  <script src="script.js"></script>
</body>
</html>
```

Affichage :

Essai de document.getElementById("texte").innerHTML

Cliquez moi !
pour changer le texte d'une balise (innerHTML).

Lorsque l'utilisateur clique sur le paragraphe "Cliquez moi ! pour ..." le texte du paragraphe est remplacé par Hello !

`document.getElementById("texte1")` désigne l'élément ayant pour identifiant unique `"texte1"`

`document.getElementById("texte1").innerHTML` est le contenu HTML de l'élément

Exemple 3

On peut accéder au style d'un élément, par exemple body (toute la page).

`document.body` désigne l'élément body.

`document.body.style.backgroundColor="rgb(255,0,0)"` affecte la couleur rouge au fond de page.

Dans cet exemple, la couleur rouge est affectée au fond de page lorsque l'utilisateur clique sur le bouton. On remarque qu'il n'y a pas de feuille de style associée.

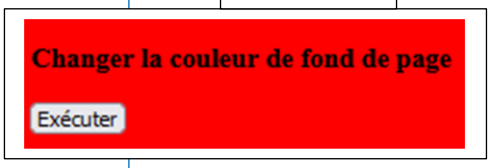
Fichier **script.js**

```
function couleur() {
    document.body.style.backgroundColor = "rgb(255,0,0)";
}
```

Fichier **couleur_dynamique.html**

```
<!doctype html>
<html>

<head>
  <meta charset="UTF-8">
  <title>Couleur</title>
</head>

<body>
  <h3>Changer la couleur de fond de page</h3>
  <p><input type="button" onclick="couleur()" value="Exécuter"></p>
  <script src=script.js></script>
</body>

</html>
```

Affichage :



2.3 Encore quelques fonctionnalités du JavaScript

- **Arrondir à 2 ou 3 décimales**

On utilise la fonction JavaScript intégrée Math.round(). Mais il y a une difficulté : cette fonction ne fait qu'arrondir à l'entier le plus proche.

Cependant, si nous voulons arrondir à deux décimales, il faut contourner cette difficulté ainsi :

```
let nombre = 5.56845;
arrondi = nombre * 100;  // 556.845
arrondi = Math.round(arrondi);  // 557
arrondi = arrondi / 100; //5.57
```

ou bien en une seule ligne :

```
arrondi = Math.round(nombre * 100) / 100;
```

Si nous voulons arrondir à trois décimales, selon le même principe, on doit écrire :

```
arrondi = Math.round(nombre * 1000) / 1000;
```

- **Structure Si … alors …sinon si …**

Il y a plusieurs possibilités selon qu'il n'y a qu'une instruction ou un bloc de plusieurs :

```
if ( condition ) une_seule_instruction ;
else une_seule_instruction ;
```

Remarque : Une condition est *toujours entre parenthèses*.

```
if ( condition ) {
    instruction_1 ;
    instruction_2 ;

} else {
    instruction_3 ;
    instruction_4 ;
}
```

```
if ( condition_1 ) {
    instruction_1 ;
    instruction_2 ;

} else if ( condition_2 ) {
    instruction_3 ;
    instruction_4 ;

} else if ( condition_3 ) {
    instruction_5 ;
    instruction_6 ;

} else {
    instruction_7 ;
    instruction_8 ;
}
```

- **Boucle pour …**

```
for ( initialisation_du_compteur ; condition_pour_maximum ; incrémentation )
    une seule instruction;
```

```
for ( initialisation_du_compteur ; condition_pour_maximum ; incrémentation )
{
    un_bloc_instructions;
}
```

2.4 Le Document Object Model (DOM)

Comme on l'a vu, la page web se présente sous forme d'objets.

Le **document** lui-même est un objet.

Ensuite on trouve **l'objet html**

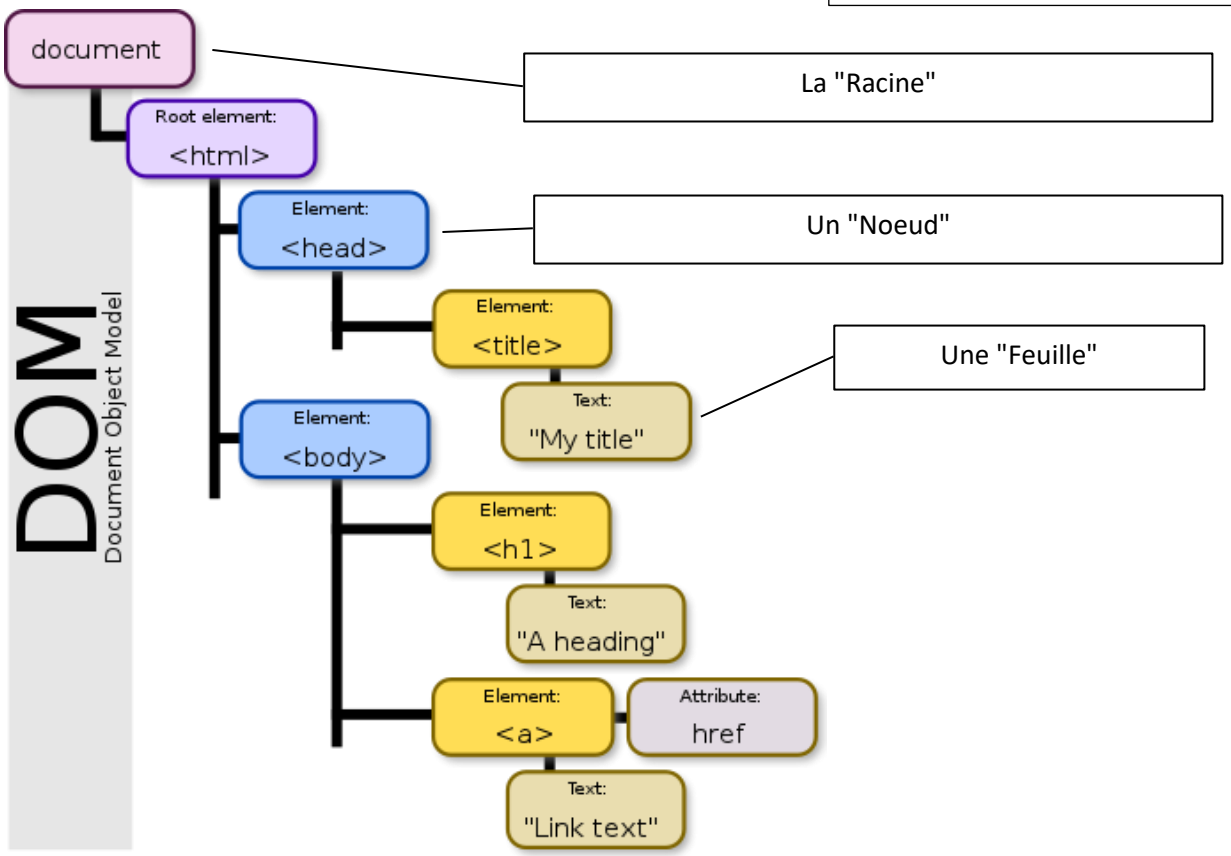
Puis **head et body**

Dans **head** on trouve **title** etc.

Dans **body** on trouve **h1, h2, h3, ..., p** etc.

```
<!doctype html>
<html>

<head>
  <meta charset="UTF-8">
  <title>Convertisseur</title>
</head>

<body>
  <p id="date"></p>
  <h3>Convertir un entier en base
  <p>Entier à convertir : <input
  <p>Cette page permet de convert
  <p>Saisir la base n : <input ty
    <p><input type="button" oncli
  <p>Résultat : <input type="text
</body>

</html>
```

- On peut se représenter ces objets comme des nœuds d'un arbre. Ou des feuilles quand ils sont au bout. C'est le Document Object Model (DOM) HTML.



La méthode **`getElementById(id)`** est la plus utile du DOM HTML car elle permet d'accéder aux objets. Grace à un script on peut lire ou modifier un attribut d'un objet. ***Exemple :***

```
document.getElementById("boutonD").innerHTML = valeurD;
```

affecte au *texte HTML* du bouton `boutonD` la variable `valeurD`.

2.5 JavaScript : appel d'une fonction en employant le DOM

2.5.1 Méthode précédente avec le gestionnaire d'évènements

Dans le § 2.2 on appelle la fonction `calcul()` en employant *le gestionnaire d'évènements* HTML.

```
<form id=monFormu>
  <h3>Appel de fonction par le gestionnaire d'évènements HTML</h3>
  <p>Nombre x : <input type="text" size="5" id="texte1" maxlength="5"></p>
  <p><input type="button" onclick="calcul()" value="Carré">  </p>
  <p>Résultat : <input type="text" size="10" id="resultat" disabled="disabled"></p>
</form>

<script type="text/javascript">
  document.getElementById("monFormu").reset(); // Pour réinitialiser le formulaire.
  function calcul(){
    let a = document.getElementById("texte1").value;
    let b = a*a;
    document.getElementById("resultat").value = b;
  }
</script>
```

Dans le code HTML on affecte à l'attribut d'évènement `onclick` la fonction `calcul()`. Un évènement est la réaction du navigateur à certaines actions de l'utilisateur ou à l'interaction interne des scripts. Les attributs d'évènements du HTML commencent toujours par le préfixe "on".

Dans le code JavaScript on a écrit le code de la fonction `calcul()`.

2.5.2 Méthode en employant le DOM dans le code JavaScript

Avec cette méthode, l'appel d'une fonction se fait directement dans le JavaScript. Dans le code HTML, il n'y a plus de référence au nom de la fonction JavaScript.

Exemple

```
<form id=monFormu>
  <h3>Appel de fonction par le DOM dans le JavaScript</h3>
  <p>Nombre x : <input type="text" size="5" id="texte1" maxlength="5"></p>
  <p> <input type="button"  id="monBouton" value="Carré">  </p>
  <p>Résultat : <input type="text" size="10" id="resultat" disabled="disabled"></p>
</form>

<script type="text/javascript">
  document.getElementById("monFormu").reset(); // Pour réinitialiser le formulaire.
  document.getElementById("monBouton").onclick = function calcul(){
    let a = document.getElementById("texte1").value;
    let b = a*a;
    document.getElementById("resultat").value = b;
  }
</script>
```

On a sélectionné un nœud du DOM (ici l'input dont l'identifiant est monBouton) en utilisant la méthode de sélection `document.getElementById("monBouton").`

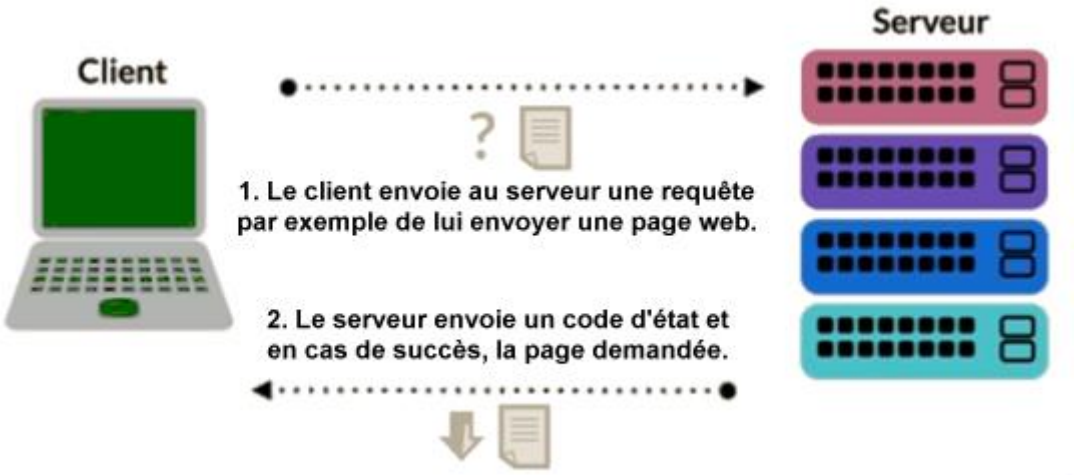
3 Interactions client-serveur – Requêtes HTTP et réponses du serveur

3.1 Modèle client-serveur

C'est une architecture de communication entre deux machines.

Le **client** est la machine qui demande l'exécution d'un service, comme l'envoi d'une page web. Cette demande s'appelle une **requête**.

Le **serveur** est une machine robuste qui fournit au client la réponse composée d'un **code d'état** (1xx, 2xx, 3xx, 4xx ou 5xx où xx sont des chiffres). Le code d'état renseigne par exemple sur "succès" ou "redirection" ou "échec" de la demande. En cas de succès, le contenu demandé dans la requête est envoyé au client.



3.2 Scripts côté serveur et côté client

On a vu qu'une page web peut contenir un script JavaScript s'exécutant directement côté navigateur. Cela permet essentiellement de :

- Agir sur la fenêtre de navigateur actuelle via le DOM (Document Object Model).
- Ouvrir de nouvelles fenêtres de navigateur et de dialogue.
- Animer, afficher et masquer ou modifier la conception d'éléments de la page.
- Valider des valeurs saisies.
- Charger Ajax[2] .
- Transmettre à d'autres sites Internet des informations sur le type de navigateur, les habitudes de lecture et les activités de navigation de l'utilisateur.

Bien que JavaScript ne puisse qu'obtenir uniquement l'accès aux cookies de l'internaute et à d'autres mémoires de données réservées aux sites Internet, l'internaute peut le désactiver sur son navigateur.

[2] **AJAX :** Technologie permettant de transférer des données en arrière-plan entre le client et le serveur, sans avoir à recharger la page web. Exemple : les suggestions de Google lorsqu'on écrit dans la zone de recherche.

En dehors des scripts s'exécutant côté client, les scripts côté serveur permettent par exemple de collecter des données qu'un client envoie par un formulaire et de les stocker dans une base de données. Le langage de script côté serveur le plus couramment utilisé est le langage de programmation **PHP** (**P**HP **H**ypertext **P**reprocessor). Il a permis de créer des sites webs tels que Facebook ou Wikipedia.

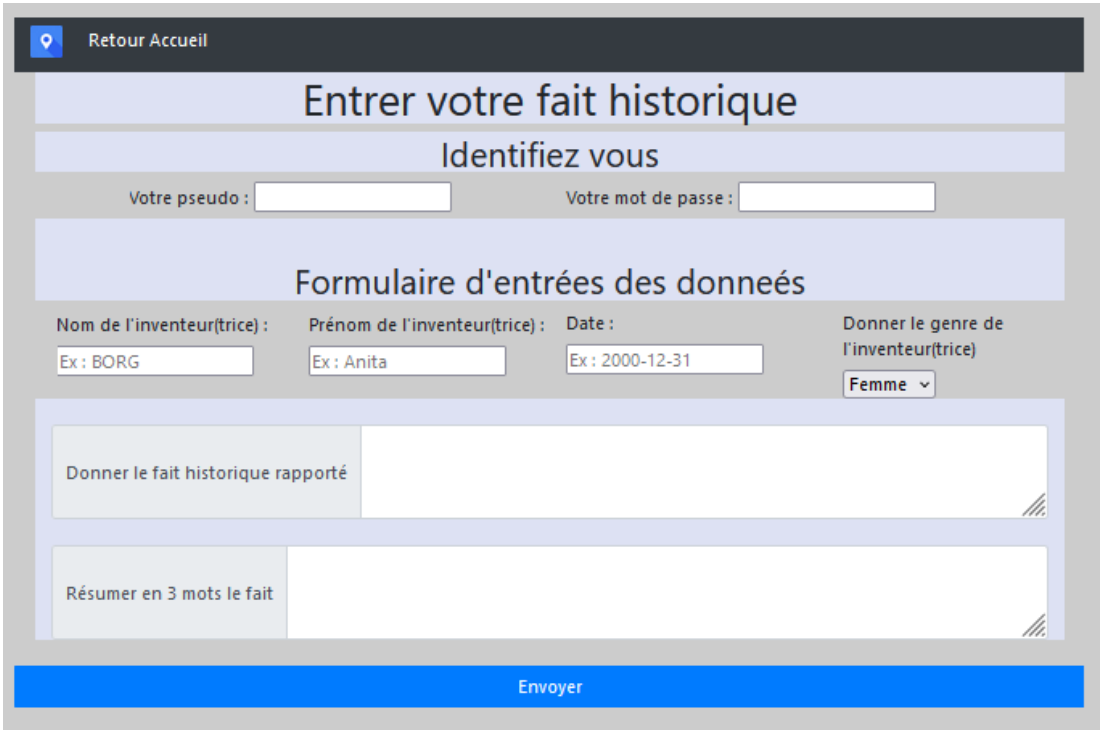
Logo de PHP

Le code source de la page PHP n'est pas visible par le client. Le client voit seulement le code HTML généré par le serveur.

- JavaScript **peut aussi être utilisé côté serveur** avec nodeJS.
- L'affirmation "*JavaSript est un langage client*" est **fausse**.

4 Formulaire d'une page web

4.1 Exemple de formulaire



L'analyse et le traitement du formulaire devant être effectués côté serveur, la page du formulaire sera écrite en langage PHP.

Un formulaire simple est introduit dans un page web par la balise <form> :

```
<form method="post" action="cible.php">
  ...
</form>
```

En cliquant sur le bouton "Envoyer", les contenus des zones de texte du formulaire sont transmis à une page `cible.php` sous forme de paramètres directement intégrés dans la requête HTTP envoyée par le client au serveur.
Il existe au moins deux types de requêtes : les requêtes de type GET et les requêtes de type POST.

⚠ Aucun des deux types de requêtes (GET et POST) n'est sécurisé lorsque la communication n'est pas chiffrée. Cela signifie qu'en l'absence de protocole sécurisé (protocole http utilisé seul) la communication peut être "lue" par un troisième acteur situé entre le client et le serveur.

4.2 Requête de type GET

Supposons que la page web `https://www.le_site_marchand.fr/formulaire.html` contienne un formulaire. Dans le formulaire HTML suivant, la méthode get permet de préparer une requête HTML de type GET. On écrit les attributs `method` et `action` :

```
<form method="get" action="page_cible.php">
….Balises du formulaire, par exemple des <input>
</form>
```

Lorsque l'internaute clique sur le bouton de soumission situé dans le formulaire (balise `<input type="submit">`), les données du formulaire sont envoyées au serveur sous la forme d'une URL visible dans la barre d'adresses. L'URL est composée :

- Du nom de la **page cible** vers laquelle seront envoyées les informations du formulaire. Ce nom est la valeur de l'attribut `action`. C'est un fichier avec du code php **qui s'exécutera sur le serveur**.
- Le symbole **?**
- **Les informations du formulaire** séparées entre elles par le symbole **&**

Ainsi si on envoie à `page_cible.php` les valeurs "bleu" pour la variable `'couleur'` et "rectangle" pour la variable `'forme'`, l'URL construite par le navigateur sera :

```
https://www.le_site_marchand.fr/formulaire/page_cible.php?couleur=bleu&forme=rectangle
```

Dans la requête de type GET, les paramètres sont donc *visibles directement dans l'URL*. Par ailleurs la taille de l'URL est limitée.

Cette requête est utile par exemple pour enregistrer une requête sur un site marchand sous forme de marque-page dans le navigateur.

4.3 Requête de type POST

Dans la requête de type POST, les paramètres ne sont *pas visibles directement dans l'URL*. Par ailleurs la requête est de taille illimitée. Cependant, les paramètres sont *accessibles à l'aide de l'inspecteur*[3] *de propriétés* du navigateur web.

Cette requête est utile pour transmettre tout type de donnée concernant l'utilisateur, comme des textes courts, des photos ou des vidéos sous forme de caractères ASCII ou en binaire. Les données transmises ne peuvent pas être enregistrées sous forme de marque-page.

Dans le formulaire HTML suivant, la méthode post permet de préparer une requête HTML de type POST. On écrit les attributs `method` et `action` :

```
<form method="post" action="page_cible.php">
….Balises du formulaire, par exemple des <input>
</form>
```

---

[3] **Inspecteur de propriétés :** lorsque la page web est affichée dans le navigateur, cet outil permet de voir le code HTML de la page. Appuyer sur la **touche F12** pour ouvrir l'inspecteur de propriétés.

4.4 Protocole SSL et sécurité

Le protocole SSL (Secure Sockets Layer) devenu en 2001 le protocole TLS (Transport Layer Security) fait partie de la *couche application* **du modèle** des protocoles de communication **OSI**.

Il fonctionne également sur le modèle client-serveur. Il assure :

- L'intégrité des données échangées entre le client et le serveur. C'est la garantie que les données reçues en provenance du site internet visité **n'ont pas été modifiées par une tierce personne** durant la communication navigateur/serveur.
- L'authentification du serveur. Il vérifie **qu'une autorité de certification** dont le certificat est lié au service SSL **a signé le certificat de serveur et vérifie la validité du certificat** de serveur.
- La confidentialité des données échangées grâce à une transmission chiffrée (utilisée par transmettre un numéro de carte bancaire, un mot de passe etc.).

La présence de la transmission chiffrée se reconnait par la présence d'une *icône "cadenas"* dans la barre d'adresse du navigateur et par les lettres *https* au leu de http au début de l'URL.

Cependant, bien qu'une transmission soit sécurisée, l'internaute doit s'assurer que l'adresse présente dans la barre d'adresses de son navigateur correspond à la bonne adresse du site.

Exemple : Saisir par erreur https://www.lemone.fr/ redirige vers un site frauduleux au lieu de celui du journal le Monde https://www.lemonde.fr/ ; La présence de https n'est pas une garantie de sécurité.