Interaction Homme/Machine sur le Web

1) Structure d'un document HTML

Pour d'écrire le contenu d'un document HTML, on le structure avec des **balises** qui vont donner des indications sur le sens du texte. Ces indications permettent :

- au navigateur d'appliquer une mise en forme par défaut, par exemple les titres seront plus gros que le corps du texte

- d'appliquer du style avec le css

- de donner du sens aux éléments du documents de façon à aider les moteurs de recherche ou autre robots à analyser le document.

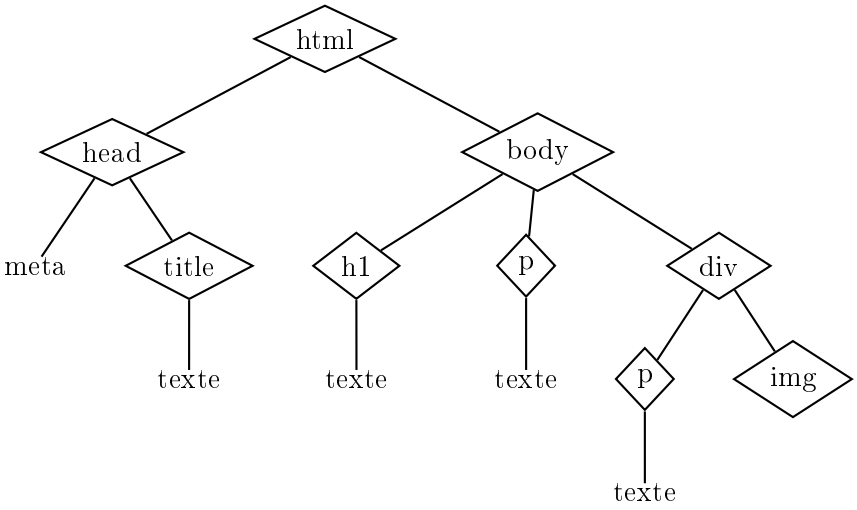
Les balises n'ont pas un rôle de mise en forme mais un rôle sémantique, c'est-à-dire portent toutes un sens.

Le **DOM** (Document Objet Mode) est une interface de programmation qui permet à des scripts d'examiner et de modifier le contenu du navigateur web.

La composition d'un document HTML est représentée sous forme d'objets reliés selon une structure en arbre.

À l'aide du DOM, un script peut modifier le document présent dans le navigateur en ajoutant ou en supprimant des noeuds de l'arbre.

Exemple : Ci-dessous le DOM et le document HTML correspondants.




```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Titre Principal de la Page</title>
  </head>
    <body>
      <h1>Bienvenue dans le DOM !</h1>
      <p>Ici commence l'exploration ...</p>
      <div id="motcle">
        <p>Encore du blabla</p>
        <img src="arbre.jpg" alt="arbre" width="50">
      </div>
    </body>
</html>
```

Activité 1 :

Voici un fichier html :

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Choisir un titre pour l'onglet.</title>
  </head>
    <body>
      <header>
        Placer un grand titre ici.
      </header>
      <main>
        Créer un premier paragraphe d'environ 20 lignes.

        Placer une image.

        Créer un second paragraphe d'environ 20 lignes.
      </main>
      <footer>
        Créer une liste avec quatre éléments.
      </footer>
    </body>
</html>
```

1. Vous devez créer un fichier, nommée **index.html**, qui aura la structure donnée ci-dessus. Vous pourrez choisir le contenu et notamment utiliser le générateur de texte du site : *https ://fr.lipsum.com/*
2. Faites un arbre correspondant au **DOM** de cette page HTML.
3. Vous devez maintenant créer un fichier **style.css**, pour mettre en forme votre page web.
   N'oubliez pas de faire le lien entre ce fichier et le fichier html !
   (a) Commencez par placer une image sur le fond de la page.
   (b) Ensuite, centrez le titre, mettez le à la taille 50 pixels, et utilisez la police de caractère Georgia.
   (c) Affichez le premier paragraphe en vert et le second en bleu.
   (d) Centrez l'image, faites en sorte que sa largeur soit de 500 pixels et encadrez-la avec une bordure noire.
   (e) Utilisez **flexbox** pour placer le premier paragraphe et l'image côte à côte. Pensez à ajuster les tailles des paragraphes et de l'image !

Remarques :

- vous trouverez la documentation html/css sur les sites :
https ://www.w3schools.com/default.asp et **https ://developer.mozilla.org/fr/docs/Web**
- pensez à vérifier l'affichage de votre page à l'aide du navigateur **firefox** ; vous pouvez utiliser les **outils de développement web** qu'il met à votre disposition pour corriger votre code.

⇒ Vous avez crée une **page Web statique**.

2) Interaction avec l'utilisateur

Différents éléments d'une page HTML peuvent permettre une **interaction** avec l'utilisateur et peuvent rendre un **site Web dynamique**.

a) Les liens hypertexte

Un **lien hypertexte** relie des informations situées à différents endroits dans un document ou dans différents documents. Les hyperliens du web sont unidirectionnels, on ne sait pas à priori ce que contient le contenu de la ressource, et si celle-ci a été déplacée ou supprimée, il n'y a plus d'accès possible. Il n'y a pas de sécurité à mettre en place et on ne gère pas les droits d'auteurs sur les ressources.

Ces liens peuvent prendre la forme de textes, d'images...

Avec **l'élément HTML <a>**, on définit la position des hyperliens dans le la page. Ces balises HTML sont aussi appelées anchor (ancre).

Il y a également **l'élément HTML <link>** qui est utilisé exclusivement dans la zone d'en-tête <head> d'un document HTML et permet de lier des documents HTML à d'autres ressources, comme une page de style CSS par exemple.

Activité 2 :

Compléter le fichier index.html :

1. Ajoutez un lien, sous la forme d'une image représentant une flèche vers le haut, en bas de la page , à droite, qui permet de remonter en haut de celle-ci.
2. Ajoutez un second lien, sous forme de texte, en dessous de l'image, qui permet d'accéder à une autre page web en lien avec l'image que vous avez choisie. (Le lien devra s'ouvrir dans un nouvel onglet.)

b) Avec le style CSS

Avec le CSS, on peut réagir au passage de la souris sur des éléments de la page web, les mettre en avant.
Pour cela on utilise le sélecteur **hover** avec la syntaxe :

```
balise-html:hover{
    propriété;
}
```

Activité 3 :

Complétez le fichier style.css :

1. Vous devez faire en sorte que les paragraphes, lorsqu'ils sont survolés par la souris changent de taille et grossissent pour une lecture plus facile.
2. Le lien hypertexte vers le site web extérieur change de couleur et est entouré d'une ombre lorsqu'on le survole.

c) Listes déroulantes

L'élément HTML <**select**> fournit une liste d'options parmi lesquelles l'utilisateur pourra choisir. Par défaut, on ne peut faire qu'un seul choix.

Voici un exemple de la syntaxe HTML :

```
<!-- Titre pour le menu -->
<label for="menu-select">Menu :</label>
<!-- Menu déroulant -->
<select name="Menu" id="menu-select">
  <option value="choix1">--Faites un choix--</option>
  <option value="choix1">Proposition1</option>
  <option value="choix2">Proposition2</option>
  <option value="choix3">Proposition3</option>
</select>
```

Remarque : Pour le moment, le choix de l'utilisateur n'entraîne aucune action ; nous verrons cela au chapitre suivant.

Activité 4 : Ajoutez une liste déroulante à votre code HTML, qui permet de répondre à une question posée. Pensez à modifier le style pour le mettre en valeur.

d) Les formulaires

En HTML, l'élément <**form**> permet l'interaction entre un utilisateur et un site web ou une application. Les formulaires permettent à l'utilisateur d'**envoyer des données au site web**. La plupart du temps, ces données sont **envoyées à des serveurs web** mais **la page peut aussi** les intercepter et **les utiliser elle-même**.

Le formulaire contient des champs de saisie (widgets) appelés <**input**> de différents types : champ texte, mot de passe, choix exclusif (radio), cases à cocher (checkbox), sélection de fichier, bouton de soumission.

Voici les principaux types de champs d'un formulaire :

```
<form>
  <!--Définir un champ de saisie de texte d'une ligne-->
  <p>nom : <input type="text" name="login"></input></p>
  <!--Définir un champ de saisie où le texte tapé n'est pas visible-->
  <p>Mot de passe : <input type="password" name="pass"></input></p>
  <p>Catégorie :
  <!--Définir un bouton radio (pour sélectionner l'un des choix)-->
    <input type="radio" name="cat" id="h" value="hu"></input>
    <label for="h">Humain</label>  <!--cela défini la zone ou l'on peut
   cliquer-->
    <input type="radio" name="cat" id="ET" value="et"></input>
    <label for="ET">Extra-terrestre</label>
    <input type="radio" name="cat" id="a" value="anim"></input>
    <label for="a">Animal</label>
  </p>
</form>
```

```
<form>
  <p>Boissons préférées :
  <!--Définir un bouton checkbox (pour sélectionner plusieurs choix)-->
    <input type="checkbox" name="boisson" id="lait" value="lait"></input
   >
    <label for="lait">lait</label>
    <input type="checkbox" name="boisson" id="coca" value="coca"></input
   >
    <label for="coca">Coca</label>
    <input type="checkbox" name="boisson" id="jus" value="jus"></input>
    <label for="jus">Jus d'orange</label>
  </p>
  <!--Définir un bouton cliquable-->
  <p><input type="button" value="Cliquer !"></p>
  <!--Définir un bouton d'envoi de données (soumettre le formulaire)-->
  <p><input type="submit" value="Envoyer"></p>
</form>
```

Activité 5 : Recopier le formulaire ci-dessus dans le pied de page de votre page HTML, en modifiant le contenu à votre convenance. Pensez au style !

3) Interaction client-serveur

La communication entre applications sur le web se fait suivant un **schéma Client-Serveur** :

- le **navigateur** qualifié de client, envoie des requêtes (par exemple il demande l'accès à une page web)
- l'autre un **ordinateur hébergeant des pages web**, qualifié de serveur, répond à la requête (charge la page demandée dans le navigateur).

Le protocole utilisé lors de cette communication est le **protocole HTTP**.

Les éléments interactifs vu précédemment peuvent être traités du côté client ou du côté serveur.

a) Interaction côté client avec Javascript

Les éléments interactif d'une page web vont permettre sa modification. Le **traitement côté client** permet de ne pas recharger complètement la page, et de n'en modifier qu'une partie, pendant qu'on la consulte.

Pour cela, on utilise principalement le **langage Javascript**.

★ Découverte du langage javascript :

Celui-ci a été crée **en 1995 par Brendan Eich**, un informaticien Américain. Il est rapidement devenu incontournable pour la programmation web côté client.

Le code Javascript peut être écrit dans une balise <**script**>, à la fin du document HTML ou bien dans un **fichier externe**, ayant l'extension **.js**.

Activité 6 : *Les bases du langage Javascript*

1. Créer un fichier html ayant simplement la structure de base.
2. Ajoutez-y le code suivant, puis ouvrez la page html sur votre navigateur.

```
<script>
  alert("Bonjour");
</script>
```

Une boîte de dialogue s'est ouverte sur votre page.

*La méthode **alert()** permet d'afficher un message dans cette boîte.*

3. Les bonnes pratiques actuelles prônent une **séparation stricte du code Javascript, du HTML et du CSS**.

   Supprimez le code précédent de votre page HTML.

   Dans le block <**head**> de votre document HTML, vous allez indiquer où trouver le script Javascript. (L'attribut **defer** indique que ce code ne doit être exécuté qu'une fois le document chargé.)

```
<head>
  <meta charset="utf-8">
  <title>Bases en javascript</title>
  <script type="text/javascript" src="bases.js" defer="defer"></script>
</head>
```

   Puis vous allez créer un fichier **bases.js**, dans lequel vous écrirez le code Javascript suivant :

```
var nom = prompt("Quel est votre nom ?");
alert("Bonjour " + nom);
```

   Actualisez votre page web.

   *La méthode **prompt** permet d'ouvrir un champ de saisie.*

   *On déclare que nom est une variable globale avec **var**. On peut également déclarer une variable locale (par exemple dans une fonction) avec **let**, ou bien une constante avec **const**.*

4. Conditions :

```
if (condition){
  instructions;
}
else{
  instructions;
}
```

   Écrire un script permettant de demander au client si c'est une femme ou un homme, et qui en fonction de la réponse affiche le message "Bonjour Madame !" ou "Bonjour Monsieur !".

   Pour vérifie l'égalité de variables en Javascript on peut utiliser == ou ===.

   La deuxième expression vérifie le type et la valeur des variables ; elle est à privilégier.

5. Boucles :

```
while (condition){
  instructions;
}
/*ou bien*/
for (initialisation;condition;incrémentation){
  instructions;
}
```

   Testez le code ci-dessous :

```
var nombre=parseInt(prompt("Donnez un entier positif : "));
/*parseInt() permet de reconnaitre la réponse comme un nombre*/
var s=0;
var i=1;
while (i<=nombre){
  s+=i;
  i++;      /*i augmente de 1*/
}
alert("La somme des "+nombre+" premiers entiers strictements
   positifs est "+s);

/*on affiche les 10 premiers entiers strictements positifs*/
for (let i=0;i<=10;i++){
  alert(i);
}
```

6. Fonctions :

```
function nomFonction(arguments){
  instructions;
  return resultat;          /*On retourne le résultat*/
}
nomFonction(arguments);     /*Appel de la fonction :*/
```

Créez un script, qui demande à l'utilisateur un nombre et affiche son carré, grâce à une fonction **carre()**. Testez votre code dans le navigateur.

7. La fonction **console.log()** permet d'afficher le résultat d'une fonction, une variable, un texte ; cela permet notamment de tester son code.

On peut visualiser la console grâce aux outils de développement web du navigateur, en sélectionnant l'onglet correspondant.

Créez une fonction **affiche()**, qui lorsqu'on lui donne en paramètre un nom, l'affiche avec un message de bienvenue dans la console.

Testez ce code dans votre navigateur.

★ Gestion des évènements avec javascript :

Lorsque l'utilisateur agit sur certains éléments de la page HTML, il peut déclencher un code javascript. Celui-ci peut **modifier dynamiquement** et **en temps réel** le rendu de la page web.

Les principaux **évènements** qui peuvent déclencher un script sont :

Évènements	Description
onclick	Réponse à un clic
onchange	Changement d'une valeur dans un champ
oninput	Champ d'un formulaire modifié
onkeyup	Touche de clavier relâchée
onkeypress	Touche de clavier pressée
mouseover	Survol d'un élément avec la souris

On peut par exemple changer le rendu graphique d'un élément, changer le texte d'un élément, modifier la valeur d'un champ de saisie, afficher un message...

Le **DOM** va nous donner accès au code HTML via javascript.

L'objet **document** représente la page web ; on peut lui appliquer un certain nombre de méthodes, comme **getElementById()** qui permet de récupérer l'élément HTML à modifier et de le placer dans une variable. Et comme c'est indiqué, on le fait grâce à son identifiant, l'attribut **id** donné dans le code HTML.

On écrit par exemple **document.getElementById("motclé")** ; ensuite on peut agir sur une **propriété de l'élément**.

Exemple : Code HTML

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Titre Principal de la Page</title>
    <script type="text/javascript" src="demo1.js" defer="defer"></script>
  </head>
    <body>
      <h1 id="titre">Bienvenue dans le DOM !</h1>
      <p>Ici commence l'exploration...</p>
      <form>
        <p><input type="button" value="Vers l'infini et au-delà !" id="
   bouton" onclick=grandir()></p>
      </form>
    </body>
</html>
```

Code javascript :

```
function grandir(){
    /*On place l'objet dont l'id est "titre" dans une variable element*/
    var element=document.getElementById("titre");
    element.style.color="red";        /*on change la couleur*/
    element.style.fontSizeAdjust="5";   /*on change la taille*/
}
```

Activité 7 : *Choix dans une liste déroulante*

1. Créez un fichier html comportant un liste permettant de choisir une couleur parmi celles proposées.
2. Créez un fichier javascript comportant une fonction permettant, suivant le choix effectué, de modifier la couleur de fond de la page web.

Aides : Pour cela, il faut rajouter l'attribut **onchange="nomFonction"** dans la balise <**select**>, qui appelle la fonction écrite dans le code javascript.

Dans la fonction, une fois l'élément select récupéré dans une variable "choix", on regarde sa valeur avec **choix.value** ;

Pour modifier la couleur de fond, on utilise l'attribut **style**, avec **document.body.style.backgroundColor**.

Activité 8 : *Formulaires*

1. Créez un fichier html, avec le titre "Bienvenue" et comportant un formulaire, comportant un champ texte où l'on entre son nom et un bouton.

2. Créez un fichier javascript comportant une fonction permettant de modifier le titre de votre page, en y incorporant votre nom, lorsqu'il est renseigné dans le champ texte.

   Puis une seconde fonction modifiant le style du titre.

Aides : Rajouter dans le titre une zone identifiée, où sera inséré le nom avec <**span id="nom"></span**>.
Indiquer dans chaque input **l'évènement** que l'on attend, ainsi que le nom de la fonction javascript à utiliser.
Pour changer le contenu d'un élément de la page on utilise **element.innerHTML**.

Activité 9 : *Boutons radio*

Créez une page web comportant trois icônes ; lorsque l'utilisateur en choisit une, le thème de la page change, selon le choix fait.

Aides : Utiliser un paramètre dans la fonction javascript.

Activité 10 : *Boutons radio*

Créez un fichier html, affichant trois portes sur une page, avec une liste déroulante permettant d'en choisir une.
Ensuite, à l'aide d'un fichier javascript, faites en sorte que, lorsque l'on choisi une porte, les autres disparaissent, et un bouton apparaît. Si on clique sur celui-ci, la porte s'ouvre et nous fait découvrir ce qui est caché derrière.

Aides : On peut utiliser dans le **style**, la propriété **'visibility'** pour cacher ou afficher un élément ;
on peut enlever un élément en javascript avec la méthode **remove()** ; on crée une nouvelle image
avec **newImage=document.createElement('img')** puis on spécifie les attributs de l'objet, comme
par exemple **newImage.src="image.jpg"** ;
enfin on indique où placer l'élément avec **lieu.appendChild(newImage)**.

b) Interaction côté serveur

⋆ Protocole HTTP

On peut également gérer l'interaction d'une page web **côté serveur**, via le **protocole HTTP** (Hypertext Transfer Protocol).

Celui-ci défini les **règles de communication entre les applications du web**.

Une requête est envoyée par le client ; elle passe par le réseau ; le serveur recalcule entièrement la page de destination pour s'adapter aux paramètres envoyés par le client, et lui renvoie.

L'**utilisation de serveurs** permet de centraliser les ressources sur un même lieu, d'en assurer l'accessibilité. On peut gérer la sécurité, les pannes, l'administration, l'évolution.

Par contre, le **coût d'exploitation** est élevé car il faut des ordinateurs puissants, un débit élevé ; il y a également des risques de piratage...

Le protocole HTTP utilise **une adresse URL unique** (Uniform Ressource Locator).

Exemple d'URL : **http ://www.nsi-premiere.fr//index.html**

Les messages envoyés par le client sont appelés des **requêtes**, et ceux envoyés par le serveur sont appelés des **réponses**. La majorité des requêtes faites par les clients sont des demandes de ressources (comme un fichier html).

Exemple : En tapant l'URL **http ://www.nsi-premiere.fr//test.html**, le navigateur envoie la requête :

GET /test.html HTTP/1.1
Host : www.nsi-premiere.fr

Le client annonce ici au serveur www.nsi-premiere.fr qu'il veut communiquer en utilisant la version 1.1 du protocole HTTP et lui demande la ressource /test.html. La réponse du serveur est alors :

HTTP/1.1	200 OK
Serveur :	nginx/1.10.3 (Ubuntu)
Date :	Mon, 20 May 2019 07 :14 :14 GMT
Content-Type :	test/html
Content-Length :	334
Last-Modified :	Sun, 19 May 20019 11 :32 :04 GMT

<!DOCTYPE html>
<html>
    Contenu de la page
</html>

Par cette réponse le serveur informe le client :

• qu'il accepte de communiquer avec la version 1.1 du protocole HTTP ;

• que la ressource demandée est disponible et donc que la requête peut être satisfaite (200 OK étant le code signifiant le succès de la requête) ;

• que le logiciel faisant office de serveur Web est le logiciel nginx, dans sa version 1.10.3, s'exécutant sur un système Ubuntu Linux ;

• que la requête a été reçue le lundi 20 mai 2019 à 7h14 et 14 secondes ;

• que le type du fichier est HTML ;

• que le fichier fait exactement 334 octets ;

• que sa dernière date de modification est le dimanche 19 mai 2019 à 11h32 et 4 secondes.

Ces informations constituent l'**entête** de la réponse HTTP.

Cela permet au navigateur de gérer la réponse : par exemple le type lui permet de savoir si il doit afficher le fichier ou le télécharger... La taille du fichier lui permet de contrôler qu'il a bien reçu l'intégralité du fichier...

Ces entêtes sont suivis d'une ligne vide, puis du **contenu** du fichier.

Á la lecture du code HTML, il peut émettre de nouvelles requêtes si besoin. Comme pour obtenir un fichier CSS par exemple : GET /styles/test.css HTTP /1.1

Remarque : Lorsque le fichier demandé n'a pas été trouvé (il n'existe pas), la réponse du serveur est :
HTTP /1.1 404 Not Found.

Il existe plusieurs code d'erreurs dans le protocole HTTP. En plus de 404, il y a 403 (permission refusée), 500 (le serveur rencontre une erreur interne)...

Le problème avec le protocole HTTP, c'est que les **données échangées ne sont pas protégées** ; n'importe quel ordinateur par lequel transite les données peut les lire. Cela pose problème lors de l'envoie d'un formulaire contenant par exemple un mot de passe...

Pour remédier à ce problème, on a introduit le protocole **HTTPS** (Hypertext Transfer Protocol Secure). Avec celui-ci les messages sont **chiffrés** avant d'être transmis.

⋆ Méthodes de passage des paramètres

Un site peut être **statique**, dans ce cas c'est la **même page** qui est envoyée à tout les clients qui la demandent.

Mais un site peut être **dynamique**. Dans ce cas, une **page différente** peut être conçue pour chaque client.

Le navigateur se connecte au serveur et lui envoie sa requête , en **joignant des informations**.

Il y a **deux méthodes** pour cela : **GET** et **POST**

- avec la méthode GET, les paramètres sont stockés dans **l'URL** ;

Par exemple : **http ://www.nsi-premiere.fr//test.html ?nom=Dupond&prenom=Ana**

Le nom du fichier est suivi d'un point d'interrogation, puis les paramètres sont donnés avec leurs valeurs ; on les sépare avec le symbole &.

Ici, on indique que le paramètre nom prend la valeur Dupond et que le paramètre prenom prend la valeur Ana.

Le serveur recevant la requête va tenir compte des paramètres pour calculer une réponse qui est renvoyée au client.

- avec la méthode POST, les paramètres sont stockés dans **le corps de la requête**.

Par exemple : Ci-dessous les paramètres sont donnés après l'entête.

POST	/test.html HTTP/1.1
Host :	www.nsi-premiere.fr
Content-Type :	test/html
Content-Length :	334

nom=Dupond&prenom=Ana

⋆ Les formulaires

Le mode de passage des paramètres par un formulaire est défini par l'attribut **method** de la balise <**form**>.

Si cet attribut est absent, sa valeur par défaut est **get**. De même que si l'on écrit **method=get** ; sinon en écrit **method=post**.

⋆ Différents usages

L'intérêt de la méthode GET est que toute l'information nécessaire au serveur est contenue dans l'URL. Il est donc possible de mémoriser uniquement cette URL pour s'en servir plus tard.

Un des problèmes de cette méthode est que la taille des URL est limitée, ainsi on ne peut passer des paramètres longs de cette manière. Dans ce cas, le serveur peut renvoyer l'erreur 414 (URL too long).

Enfin, il y a le problème de la confidentialité : une personne à proximité de l'écran pourrait voir quels paramètres sont donnés dans l'URL. Ce qui peut être dangereux, par exemple s'il s'agit d'un mot de passe.

Les caractéristiques de la méthode POST sont exactement inverses. Toutes les informations ne se trouvant pas dans l'URL, cela nécessitera de recharger l'intégralité de la page pour que le serveur tienne compte des paramètres donnés ; cela peut poser problème si la connexion réseau est interrompue à ce moment là...

Si la taille des paramètres est trop importante, il est nécessaire d'utiliser la méthode POST.

De même, pour permettre la confidentialité des paramètres on utilisera cette méthode, à laquelle s'ajoutera le protocole HTTPS.

Par exemple :

Sur un site de réservation de billets de trains, le formulaire permettant de choisir et d'acheter un billet doit utiliser la méthode POST, car la soumission du formulaire provoquera une mise à jour (le billet ne sera pas disponible pour les autres personnes car il aura été vendu).

Mais, si en cliquant sur le bouton de soumission pour finaliser l'achat du billet un problème de connexion réseau survient, la page de confirmation peut ne pas se charger. Le navigateur continue d'attendre le résultat et on pourrait être tenté de recharger la page. Cela impliquerait de renvoyer une nouvelle fois les données du formulaire et d'exécuter une nouvelle fois la mise à jour. Cela pourrait entrainer l'achat d'un second billet !