Interaction Homme-Machine sur le Web

TP Langages du Web
(JavaScript)

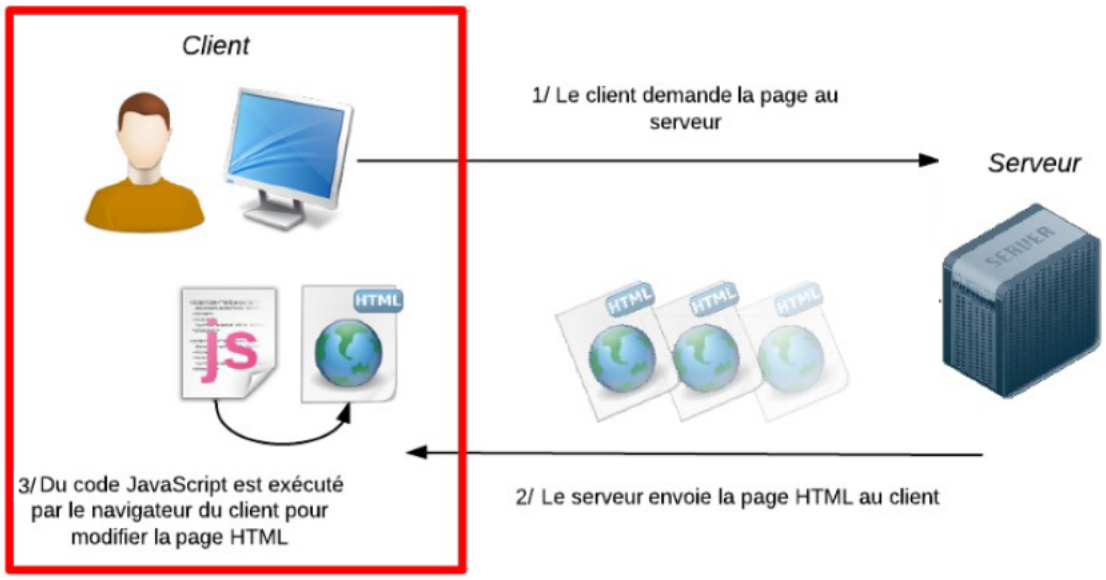
Objectif du TP

Les activités menées dans ce TP visent à répondre à la problématique suivante : *comment rendre une page Web interactive coté client ?*

En effet, après le chargement d'une page dans un navigateur Web, il est courant de pouvoir interagir avec celle-ci afin d'en modifier une partie du contenu. Par exemple :

- lors de la saisie des champs d'un formulaire de calcul ;
- lors du défilement d'un diaporama ;
- lors du filtrage et/ou du tri des résultats d'une recherche ;
- etc.

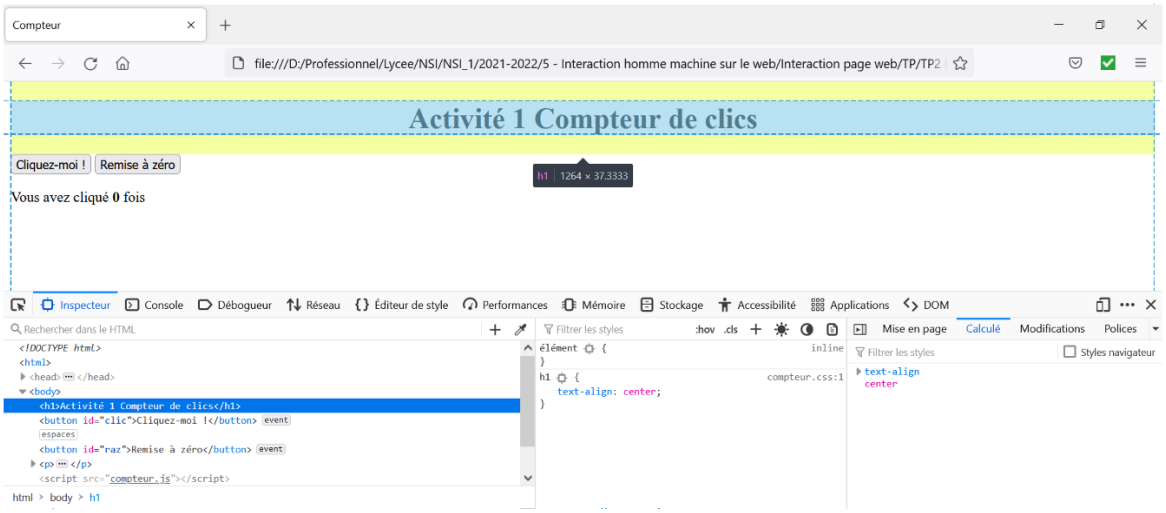
En informatique, le langage *JavaScript* est le troisième langage utilisé, après le HTML et le CSS, pour introduire, au moyen de scripts exécuté coté client (i.e. par le navigateur Web), de l'interactivité dans les pages Web.



Activité 1 : Compteur de clics

Enregistrer sous votre session le dossier `Activité1_Compteur`, puis ouvrir à partir d'un navigateur Web (*Firefox*) le fichier `compteur.html`.

Activer les *outils de développement Web* du navigateur (touche de fonction *F12* du clavier), puis sélectionner l'onglet *Inspecteur* : cet outil permet de faire le lien entre la représentation graphique du contenu d'une page Web et son code source.



Dans cette page Web, les différents éléments qui concourent à assurer l'interaction Homme – Machine sont :

- le bouton *Cliquez moi !* qui permet d'incrémenter (i.e. augmenter de une unité) la valeur du compteur de clics ;
- le bouton *Remise à zéro* qui permet de réinitialiser le compteur ;
- l'élément textuel utilisé pour afficher la valeur du compteur, c'est à dire le nombre de clics réalisés depuis le chargement de la page Web dans le navigateur.

1) Pour chacun des trois éléments précédents, relever le type de balise HTML associé ainsi que la valeur de l'attribut `id` (i.e de son *identifiant*).

La gestion des évènements, comme par exemple le clic sur un bouton, ainsi que les modifications résultantes du contenu de la page Web sont définis (et donc programmés) dans un fichier JavaScript (dans notre cas le fichier `compteur.js`).

Le lien entre les deux fichiers (HTML et JavaScript) est réalisé par l'intermédiaire de la balise `<script>` qui est déclarée dans le code source de la page Web (i.e. dans le fichier `compteur.html`).

Éditer le fichier `compteur.js` à partir d'un éditeur de texte (*NotePad++*).

La méthode `getElementById` renvoie un objet qui représente l'élément du contenu HTML dont l'attribut `id` correspond à la chaîne de caractères spécifiée en argument.

2) A quel élément du contenu HTML (i.e. du fichier `compteur.html`) est associé la constante `btn_clic` ?
Même question pour la constante `val_compteur` déclarée dans le corps de la fonction `comptage`.

La méthode `addEventListener` permet de définir, pour un élément de l'interface Homme-Machine donné, le type d'évènement attendu ainsi que le nom de la fonction qui doit être exécutée lorsque cet évènement se produit.

3) Quel type d'évènement est associé au bouton *Cliquez moi !* ? Quelle est l'action attendue (i.e. le nom de la fonction exécutée) lorsque cet évènement se produit ?

La fonction précédente a pour finalité de modifier le valeur du nombre de clics affiché dans la page Web. Pour ce faire elle doit avoir accès en lecture et en écriture au contenu textuel de l'élément HTML chargé d'afficher cette valeur.

La propriété `textContent` représente le contenu textuel d'un élément HTML. Ce contenu est toujours homogène à une chaîne de caractères. Par conséquent, pour réaliser des opérations arithmétiques sur ce type de données, il faut au préalable les transformer en données numériques au moyen de la fonction `parseInt`.

4) En vous aidant du code existant (i.e. du code source du fichier `compteur.js`), implémenter la fonctionnalité de remise à zéro du compteur de clics.
Sauvegarder le fichier `compteur.js`, puis *actualiser le contenu de la page Web* affiché dans le navigateur (touche de fonction *F5* du clavier). Vérifier que le fonctionnement observé est conforme à celui attendu. Corriger si nécessaire le code source du fichier `compteur.js`.

En résumé (API DOM et JavaScript).

— La méthode `getElementById` renvoie un objet qui représente l'élément du contenu HTML dont l'attribut `id` correspond à la chaîne de caractères spécifiée en argument de la méthode :

```
const element = document.getElementById("identifiant");
```

— La méthode `addEventListener` permet de définir, pour un élément de l'interface Homme-Machine donné, le type d'évènement attendu ainsi que le nom de la fonction qui doit être exécutée lorsque cet évènement se produit :

```
element.addEventListener("type_évènement", nom_fonction);
```

— La propriété `textContent` permet d'accéder en lecture ou en écriture au contenu textuel d'un élément du contenu HTML :

```
let text = element.textContent;
element.textContent = "ceci est un simple exemple de texte";
```

— La méthode `parseInt` permet de convertir une chaine de caractères, fournie en argument, en un nombre entier :

```
let nombre = parseInt(element.textContent);
```

Activité 2 : Changement de couleur

Dans cette activité, il s'agit de modifier la couleur de fond des paragraphes lorsque l'utilisateur appuie sur la touche `R` (rouge), `J` (jaune), `V` (vert) ou `B` (blanc) du clavier. On ne fera pas de distinction entre minuscule et majuscule.

Enregistrer sous votre session le dossier `Activité2_Couleur`.

Le gestionnaire d'évènement chargé de détecter une action sur l'une des touches du clavier est déjà implémenté dans le code source du fichier JavaScript `couleur.js`.

Ouvrir le fichier `couleur.html` à partir d'un navigateur Web. Activer les *outils de développement Web du navigateur*, puis sélectionner l'onglet *Console*.

1) Relever à partir des informations affichées en console les valeurs des codes ASCII associés aux touches `R`, `J`, `V` et `B` du clavier.

Dans la fenêtre des *outils de développement Web du navigateur* sélectionner maintenant l'onglet *Inspecteur*.

2) Relever le type de balise HTML associé aux trois paragraphes dont on souhaite pouvoir modifier la couleur de fond.

Éditer le fichier `couleur.js` à partir d'un éditeur de texte (*NotePad++*).

3) Dans le corps de la fonction `changer_couleur`, définir une variable nommée `couleur`, puis écrire le code JavaScript qui permet de modifier la valeur de cette variable, en fonction du code ASCII de la touche actionnée, puis d'afficher en console cette valeur.

   Le lien suivant rappelle la syntaxe JavaScript d'une structure de test :

   `https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Instructions/if...else`

   Sauvegarder le fichier `couleur.js`, puis actualiser le contenu de la page Web affiché dans le navigateur. Vérifier que le fonctionnement observé est conforme à celui attendu. Corriger si nécessaire le code source du fichier `couleur.js`.

Pour accéder à un élément en particulier du contenu HTML, on peut utiliser la méthode `querySelector` qui renvoie le premier élément du contenu HTML qui correspond au sélecteur CSS spécifié en argument. La syntaxe d'utilisation de cette méthode est rappelée dans le lien suivant :

`https://developer.mozilla.org/fr/docs/Web/API/Document/querySelector`

La propriété `style.backgroundColor` permet d'accéder en lecture et en écriture à la couleur de fond d'un élément. La liste des mots-clés représentant les couleurs est donnée dans le lien suivant :

`https://developer.mozilla.org/fr/docs/Web/CSS/color_value#valeurs`

4) Compléter le code du fichier `couleur.js` afin de modifier la couleur de fond du premier paragraphe du contenu de la page Web en fonction de la touche du clavier (`R`, `V`, `J` ou `B`) qui est actionnée.
Sauvegarder le fichier `couleur.js`, puis actualiser le contenu de la page Web affiché dans le navigateur. Vérifier que le fonctionnement observé est conforme à celui attendu. Corriger si nécessaire le code du fichier `couleur.js`.

Sur le même principe, la méthode `querySelectorAll` renvoie la liste de tous les éléments du contenu HTML qui correspondent au sélecteur CSS spécifié en argument. La syntaxe d'utilisation de cette méthode est rappelée dans le lien suivant :

`https://developer.mozilla.org/fr/docs/Web/API/Document/querySelectorAll`

Pour parcourir tous les éléments d'une liste on peut implémenter une structure de boucle bornée dont la syntaxe JavaScript est rappelée dans le lien suivant :

`https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Statements/for...of`

5) Compléter le code du fichier `couleur.js` afin de modifier la couleur de fond de tous les paragraphes en fonction de la touche du clavier actionnée.
Sauvegarder le fichier `couleur.js`, puis actualiser le contenu de la page Web affiché dans le navigateur. Vérifier que le fonctionnement observé est conforme à celui attendu. Corriger si nécessaire le code du fichier `couleur.js`.

En résumé (API DOM et JavaScript).

— La méthode `querySelector` renvoie le premier élément du contenu HTML qui correspond au sélecteur CSS spécifié en argument sous la forme d'une chaine de caractères :

```
const element = document.querySelector("selecteur_css");
```

— La méthode `querySelectorAll` renvoie la liste de tous les éléments du contenu HTML qui correspondent au sélecteur CSS spécifié en argument sous la forme d'une chaine de caractères :

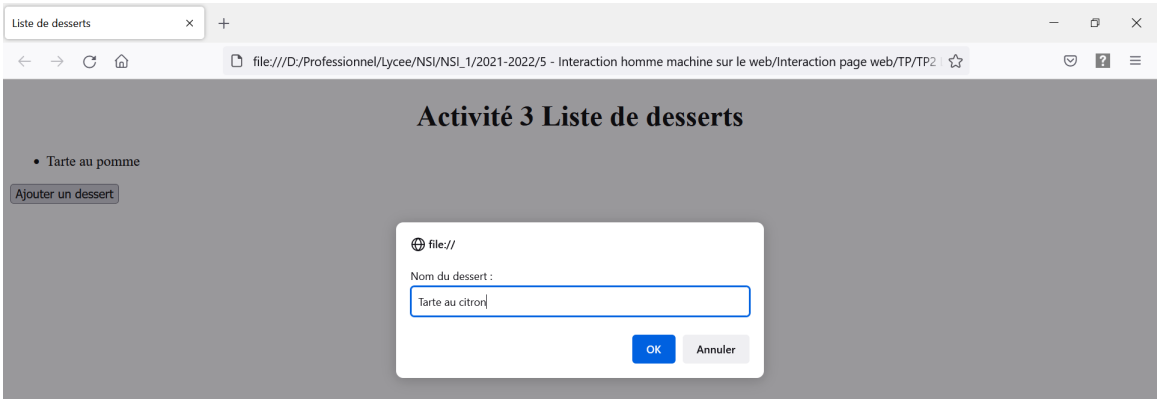
```
const elements = document.querySelectorAll("selecteur_css");
```

— La propriété `style.backgroundColor` permet d'accéder en lecture ou en écriture à la couleur de fond d'un élément :

```
element.style.backgroundColor="red";
```

Activité 3 : Liste des desserts

Dans cette activité, il s'agit de modifier le contenu d'une liste par ajout de nouveaux éléments dont les intitulés sont saisis au moyen d'une fenêtre "pop-up".



Enregistrer sous votre session le dossier `Activité3_Liste`.

Exécuter le fichier `liste.html` dans un navigateur Web, puis activer les *outils de développement Web du navigateur* et sélectionner l'onglet *Inspecteur*.

1) Identifier les différents éléments du contenu de cette page Web qui concourent à assurer l'interaction Homme – Machine ainsi que leur rôle respectif. Pour chaque élément préciser le type de balise HTML associé.

Éditer le fichier `liste.js` à partir d'un éditeur de texte (*NotePad++*).

2) Associer au bouton *Ajouter un dessert* un gestionnaire d'évènement de type `click`. La fonction qui doit être exécutée lors de l'apparition de cet évènement a pour nom `ajouter_dessert`.

   Conseils méthodologiques pour contrôler le bon fonctionnement de votre gestionnaire d'évènement :

   - *dans le corps de la fonction* `ajouter_dessert`*, ajouter l'instruction suivante* `console.log("Test gestionnaire d'évènement")` *puis sauvegarder votre fichier ;*
   - *dans la fenêtre des outils de développement Web, sélectionner l'onglet Console, puis actualiser le contenu de la page Web ;*
   - *réaliser une action sur le bouton "Ajouter dessert", puis vérifier que le message "Test gestionnaire d'évènement" a bien été affiché en console.*

La fonction `prompt` permet d'ouvrir une nouvelle fenêtre avec un champ de saisi. Sa syntaxe d'utilisation est rappelée dans le lien suivant :

`https://developer.mozilla.org/fr/docs/Web/API/Window/prompt`

3) Dans le corps de la fonction `ajouter_dessert`, implémenter la fonctionnalité relative à la saisie d'un nouveau nom de dessert, ce dernier pouvant être affecté à une variable locale à la fonction.

Pour ajouter un nouvel élément HTML au contenu d'une page Web, il faut :

— créer cet élément en spécifiant (au minimum) son type ;

— définir l'ascendant direct (i.e. le père) de cet élément.

La méthode `createElement` permet de créer un nouvel élément HTML dont le type est spécifié en argument de la méthode. La syntaxe d'utilisation de cette méthode est rappelée dans le lien suivant :

`https://developer.mozilla.org/fr/docs/Web/API/Document/createElement`

La méthode `appendChild` permet d'ajouter un nouvel élément HTML à la fin de la liste des enfants de l'élément parent sur lequel elle est appliquée. La syntaxe d'utilisation de cette méthode est rappelée dans le lien suivant :

`https://developer.mozilla.org/fr/docs/Web/API/Node/appendChild`

Enfin, la propriété `textContent` représente le contenu textuel d'un élément HTML. Cette propriété est accessible en lecture et en écriture. La syntaxe d'utilisation de cette propriété est rappelée dans le lien suivant :

`https://developer.mozilla.org/fr/docs/Web/API/Node/textContent`

4) Dans le corps de la fonction `ajouter_dessert`, implémenter à partir des méthodes précédentes les instructions qui permettent de modifier le contenu de la liste des desserts affichée dans la page Web.
Sauvegarder le fichier `liste.js`, puis actualiser le contenu de la page Web affichée dans le navigateur. Vérifier que le fonctionnement observé est conforme à celui attendu. Corriger si nécessaire le code source du fichier `liste.js`.

En résumé (API DOM et JavaScript).

— La méthode `createElement` permet de créer un nouvel élément HTML dont le type (i.e le nom de la balise) est spécifié sous la forme d'une chaine de caractères en argument de la méthode :

```
const nouvel_element = document.createElement("type_element");
```

— La méthode `appendChild` permet d'ajouter un nouvel élément HTML à la fin de la liste des enfants de l'élément parent sur lequel elle est appliquée :

```
element_parent.appendChild(nouvel_element);
```

— La propriété `textContent` permet d'accéder en lecture et en écriture au contenu textuel d'un élément du contenu HTML :

```
let text = element.textContent;
element.textContent = "ceci est un simple exemple de texte";
```

Pour les plus rapides ...

Reprendre l'`Activité1_Compteur`.

D'un point de vue programmation événementielle, les deux boutons *Cliquez moi !* et *Remise à zéro* réalisent en définitive la même modification dynamique du contenu de la page Web à savoir l'actualisation du contenu textuel de l'élément HTML dont l'identifiant est `id = compteurClics` (cf fichier *compteur.html*).
Il en est bien entendu de même des instructions exécutées dans le corps des deux fonctions `comptage` et `remise_a_zero` associées aux deux boutons de l'interface Homme – Machine.

On se propose donc dans la suite de cette activité de modifier le code source du fichier `compteur.js` afin de ne conserver qu'une seule fonction nommée `comptage`.

Éditer le fichier `compteur.js` à partir d'un éditeur de texte (NotePad++).

1) Modifier les instructions de déclaration du gestionnaire d'évènement associé au bouton de remise à zéro du compteur de sorte que la fonction appelée soit la même pour les deux boutons.

Pour pouvoir discriminer les deux types d'évènements liés respectivement aux boutons *Cliquez-moi !* et *Remise à zéro*, on doit ajouter à la définition de la fonction `comptage` un paramètre nommé par exemple `e` (comme *event* ou *évènement*).

2) Afin de prendre en compte cette nouvelle exigence, modifier l'entête de la fonction `comptage`, puis dans le corps de cette dernière ajouter l'instruction :

```
console.log(e.target.id);
```

Sauvegarder le fichier `compteur.js`, puis actualiser le contenu de la page Web. Observer la valeur affichée en console lorsque l'un des deux boutons est actionné.

3) Modifier les instructions du corps de la fonction `comptage` afin d'actualiser le contenu textuel de l'élément d'identifiant `id = compteurClics` en fonction du type d'évènement déclencheur (i.e le bouton *Cliquez-moi !* ou *Remise à zéro*).
Sauvegarder le fichier `compteur.js`, puis actualiser le contenu de la page Web. Vérifier que le fonctionnement observé est conforme à celui attendu. Corriger si nécessaire le code source du fichier `compteur.js`.