TP 3

JavaScript : évènements et DOM

Ce TP est dédié à JavaScript. Sauf mention contraire, aucune modification des fichiers HTML et CSS fournis n'est autorisée.

EXERCICE

Exercice 3.1 (Clic et CSS)**.** Télécharger les fichiers `clic.html` et `clic.css`. Créer un fichier JavaScript `clic.js` devant gérer les actions suivantes de l'internaute :

Question 1 : Quand l'internaute clique sur le titre « Description », les bords de l'image deviennent arrondis (classe `arrondi`) ; il faudra utiliser `image.classList`, où `image` est l'élément du DOM correspondant à l'image. Après un second clic sur le titre « Description », les classes de style susmentionnées redeviennent celles présentes à l'ouverture de la page Web.

Il conviendra d'associer un auditeur à l'élément d'identifiant `description` avec un évènement `click`.

(Penser à utiliser `toggle` !)

Rappel : l'association d'auditeurs à des éléments du DOM doit être impérativement être réalisée dans la partie « principale » du fichier JavaScript, de la forme :

```
window.addEventListener("load", () => {
  // ici le programme principal
}
```

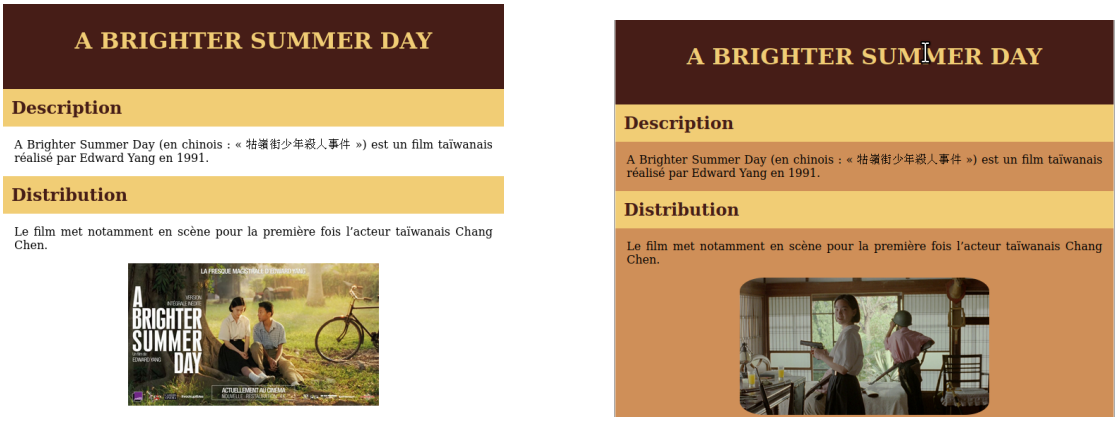
Question (optionnelle) 2 : Quand l'internaute clique sur le titre « Description », le style associé aux articles prend la classe `fondencouleur`. Après un second clic sur le titre « Description », les classes de style susmentionnées redeviennent celles présentes à l'ouverture de la page Web.

Solution : créer une fonction (auditeur) effectuant le changement de classe pour tous les articles ; associer cet auditeur à l'élément d'identifiant `description` avec un évènement `click`. Squelette possible pour cet auditeur :

```
function auditeur_clic (e) {
  // On permute la classe des éléments
  articles.forEach( (p) => {
    // TODO
  }
}
```

Et, dans la fonction principale :

```
// Les éléments du DOM (variables globales)
let articles;
let description;
```



(a) Normal

(b) Après clic + mouseover

FIGURE 3.1 – Rendu exo clic

```
4
5  window.addEventListener("load", () => {
6    // Récupération des nœuds concernés
7    articles    = document.querySelectorAll("TODO");
8    description = document.getElementById("TODO");
9
10   // Ajout d'un auditeur sur description
11   description.addEventListener("click", auditeur_clic);
12 }
```

Question 3 : Quand l'on survole avec le curseur le titre général de la page, l'image (qui a pour source `images/brighter1.png`) doit changer pour `images/brighter2.png`. Quand le curseur sort du titre général, l'image de départ (`images/brighter1.png`) revient. Utiliser *deux* auditeurs, l'un avec l'évènement `mouseover`, l'autre avec `mouseout`. Il faudra modifier (avec JavaScript) l'attribut `src` de l'image.

La figure 3.1a montre le rendu par défaut, et la figure 3.1b montre le rendu lorsque l'on a cliqué une fois sur « Description » *et* que l'on passe le curseur sur le titre de la page.

Solution partielle pour l'évènement `mouseover` :

```
1  // Les éléments du DOM (variables globales)
2  let image;
3
4  window.addEventListener("load", () => {
5    // ...
6    image       = document.querySelector("TODO");
7    // ...
8
9    titre.addEventListener("mouseover", (e) => {
10     // TODO: changer la source de l'image
11   });
12 }
```

■

EXERCICE

Exercice 3.2 (Diaporama)**.** Télécharger le fichier `diaporama.html`.

Question 1 : Créer un fichier `diaporama.js` et le compléter de façon à ce que chacune des 10 photos puisse être visualisée de façon circulaire : une nouvelle photo s'affiche après un clic de l'internaute dans l'image et, quand la dernière photo a été visualisée, la première est de nouveau affichée.

Indication : créer un tableau d'images :

```
const sources = new Array('diapo1.jpg', 'diapo2.jpg', 'diapo3.jpg',
    'diapo4.jpg', 'diapo5.jpg', 'diapo6.jpg', 'diapo7.jpg', 'diapo8.jpg',
    'diapo9.jpg', 'diapo10.jpg');
```

puis remplacer l'image courante par l'élément suivant du tableau, avec une variable globale encodant l'indice courant du tableau.

Question (optionnelle) 2 : Ajouter des instructions au fichier JavaScript de façon à ce que la photo change automatiquement toutes les 2 secondes. Pour cela, consulter la documentation de la méthode `window.setInterval`.

Question (optionnelle) 3 : Faire en sorte que le changement de photo automatique ne se fasse *que sans action de l'internaute au bout de 2 secondes.*

■

EXERCICE

Exercice 3.3 (Puzzle)**.** Télécharger les fichiers `puzzle.html` et `puzzle.css` ; créer un fichier `puzzle.js`. Au chargement de la page Web, le rendu est celui de la figure 3.2a.

Question 1 : Remplir le fichier `puzzle.js` de façon à ce que l'on puisse reconstituer le puzzle (voir figure 3.2b) en cliquant sur une première image puis sur une deuxième : celles-ci sont alors échangées.

Indications : récupérer les images dans un tableau avec un `querySelectorAll` approprié (attention : il ne doit pas être possible de permuter l'une des douze images avec l'image de la solution en bas de page !). Chaque case du tableau contient donc une image du puzzle. En cliquant consécutivement sur deux images, on les permute.

Il conviendra d'implémenter une fonction `auditeur_clic(evenement)`, et de l'associer comme auditeur aux nœuds correspondant aux douze images (avec un `forEach`). Cette fonction devra bien sûr différencier le premier clic du deuxième clic (utiliser une variable booléenne globale) :

- si premier clic : sauvegarder l'image courante (`evenement.target`) dans une variable globale ;
- si deuxième clic : intervertir la source (`src`) de l'image courante avec la source de l'image sauvegardée lors du premier clic.

Question (optionnelle) 2 : Changer l'opacité de la première image sur laquelle on clique, puis rétablir l'opacité à 1 après l'échange. Ajouter une animation de l'opacité (c'est-à-dire que son opacité va graduellement changer).

Question (optionnelle) 3 : Ajouter une fonction permettant de mélanger de façon aléatoire les images à chaque chargement de la page.

Question (optionnelle) 4 : Ajouter un compteur qui actualise (en temps réel) le nombre de coups joués jusqu'à présent.

(a) Au chargment

(b) Objectif

FIGURE 3.2 – Rendu exo puzzle

Question (optionnelle) 5 : Lorsque l'image est parfaitement reconstituée, indiquer un message de victoire.

■

EXERCICE

Exercice 3.4 (Modification du DOM)**.** Télécharger le fichier `boutonlog.html`. Créer un fichier JavaScript `boutonlog.js`.

Question 1 : Ajouter un auditeur `changeCouleur(e)` sur le bouton « Cliquer ici » pour que, lorsque que l'on clique dessus, celui-ci change de style et prenne une couleur de fond aléatoire.

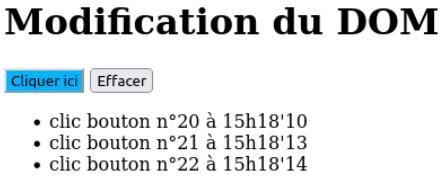
Question 2 : Ajouter un auditeur `ajouteElementListe(e)` sur le bouton « Cliquer ici » pour que, lorsque que l'on clique dessus, un item `<li>` soit ajouté à la liste (identifiant `liste`) avec pour contenu une chaîne de la forme :

```
clic bouton n°1 à 15h18'10
```

où « 1 » représente le numéro du clic, et « 15h18'10 » est l'heure courante.

Question 3 : Ajouter un auditeur `videListe(e)` sur le bouton « Effacer » de façon à ce que, lorsque l'on clique sur ce bouton, la liste des clics se vide complètement.

Un exemple de rendu où l'on a cliqué 19 fois sur le bouton puis cliqué sur « Effacer », puis re-cliqué de nouveau 3 fois est donné ci-après :



Question (optionnelle) 4 : Ajouter un auditeur qui fait que, à chaque clic sur le bouton « Cliquer ici », un message est affiché dans la console de la forme :

```
Clic sur le bouton aux coordonnées (60,96)
```

où (60,96) représente les coordonnées du clic de la souris (forcément au sein du bouton).

Question (optionnelle) 5 : Modifier le script de façon à ce que les éléments de la liste s'affichent chacun dans la couleur qui était celle du bouton au moment du clic concerné.

■

Version du TP : 29 avril 2024