WEB DESIGNER UI/UX

Approche développement frontend

5XFRO-1

Source d'inspiration du cours : Thibault Six, https://cours.tsix.be/cours/projet-web-multimedia/javascript/

Pour atteindre le seuil de réussite, l'étudiant sera capable :

L'étudiant sera capable,

face à une structure informatique opérationnelle connectée à Internet, disposant des logiciels appropriés et de la documentation nécessaire, en utilisant le vocabulaire technique et l'orthographe adéquate, et en respectant les normes et standards en vigueur,

à l'aide d'un outil approprié :

- d'identifier, dans une application web, les éléments impliquant l'usage d'un script client ;
- d'analyser un script client en termes de:
  - définition des variables,
  - structures conditionnelles et itératives,
  - fonctions,
  - structures interactives (gestion des évènements,...),
  - …
- d'exploiter un script client dans une application web ;
- de modifier ou de créer un script et de l'intégrer dans une application web ;
- de décrire et de caractériser objets, propriétés et méthodes ;
- d'identifier les évènements et les objets impliqués dans une séquence interactive ;
- de mettre en œuvre une séquence interactive en implémentant un gestionnaire d'évènement ;
- d'utiliser, dans le langage choisi, les variables, les structures conditionnelles, les structures itératives, les tableaux, l'affichage dans une page web,... ;
- d'exploiter le côté orienté objet du langage choisi (les classes prédéfinies et leurs composants) ;
- d'exploiter une source de données externes dans le cadre d'une interaction avec un service tiers (API, …) ;
- de choisir et d'exploiter une bibliothèque tierce ou un framework et éventuellement des plugins,... en vue du développement de scripts spécifiques pour RIA (interfaces riches, transmissions asynchrones,...) ;
- d'identifier des erreurs de programmation au moyen d'outils ou de techniques de débogage et d'y apporter une solution pertinente ;
- d'optimiser les ressources en vue de leur mise en production ;
- de recourir à bon escient à la documentation disponible.

Table des matières

1.Introduction

1.1 Historique[1]

1. **Création par Brendan Eich (1995) :** JavaScript a été créé par Brendan Eich en 1995 alors qu'il travaillait chez Netscape Communications Corporation. Il a été conçu pour être un langage de script léger et dynamique pour les pages web.
2. **Première Apparition dans Netscape Navigator (1996) :** La première version publique de JavaScript a été introduite avec Netscape Navigator 2.0 en 1996.
3. **Standardisation avec ECMAScript (1997) :** En 1997, JavaScript a été soumis à l'ECMA International pour standardisation afin d'éviter les problèmes liés à son contrôle par une seule entreprise. Cela a conduit à la création du standard ECMAScript, qui est la spécification de base du langage.
4. **Croissance de la Popularité (2000s) :** JavaScript a gagné en popularité avec l'avènement du développement web dynamique et des applications côté client. L'utilisation intensive d'AJAX (Asynchronous JavaScript and XML) a également contribué à l'essor du langage.
5. **Node.js (2009) :** L'introduction de Node.js par Ryan Dahl en 2009 a permis l'exécution de JavaScript côté serveur. Cela a ouvert de nouvelles possibilités pour JavaScript en permettant son utilisation côté serveur.
6. **L'ère des Frameworks (années 2010) :** L'émergence de nombreux frameworks JavaScript tels que Angular, React, et Vue.js a facilité le développement d'applications web complexes et interactives.
7. **ES6/ECMAScript 2015 (2015) :** L'édition 2015 d'ECMAScript, également connue sous le nom d'ES6, a introduit de nouvelles fonctionnalités et améliorations significatives pour le langage, telles que les classes, les modules, les fonctions fléchées, et bien d'autres.
8. **Évolutions continues (depuis 2015) :** Depuis ES6, de nouvelles versions d'ECMAScript sont publiées chaque année, apportant des fonctionnalités et des améliorations continues au langage.

Aujourd'hui, JavaScript est l'un des langages de programmation les plus utilisés au monde, jouant un rôle central dans le développement web, tant côté client que côté serveur.

[1] OpenAI, 2024

2. La balises script

La balise script permet d'inclure du code JavaScript dans une page HTML. Elle peut être utilisée de différentes manières :

2.1 Inline

- L'inclusion inline consiste à incorporer directement du code JavaScript dans le corps du document HTML en utilisant des attributs tels que onclick, onload, ou onsubmit.
- Cependant, cela peut rendre la maintenance difficile et entraîner une duplication de code.

```
<button onclick="alert('Hello, world!')">Cliquez ici</button>
```

2.2 Balise script

- La méthode la plus courante utilise la balise script, qui permet d'inclure du code JavaScript directement dans le document HTML.

```
<script>
  // Code JavaScript ici
</script>
```

- On peut placer le code directement entre les balises ou faire référence à un fichier externe en utilisant l'attribut src.

```
<script src="chemin/vers/fichier.js"></script>
```

- La balise script peut être positionnée dans l'en-tête, avant la fermeture du corps, ou à l'intérieur d'un élément spécifique.

2.3 Attributs de la balise script

- src : spécifie l'URL du fichier JavaScript externe.
- type : indique le type de contenu du script*("text/javascript".)*
- async : spécifie une exécution asynchrone *(ce qui signifie qu'il n'attend pas que le script soit chargé avant de continuer à traiter la page.)*
- defer : indique une exécution différée après le chargement de la page.

- crossorigin : gère les politiques de sécurité pour les ressources externes *(Cela peut être utile lorsque vous chargez un fichier JavaScript depuis un autre domaine.)*

2.4 Fichier à part

- Une méthode courante est d'utiliser un fichier JavaScript externe avec l'attribut src de la balise script.
- Cela améliore la séparation entre le code JavaScript et HTML, facilitant la maintenance et évitant la duplication de code.

2.5 Bundler

- Une approche avancée est l'utilisation d'un bundler (comme Webpack, Parcel, Vite) pour regrouper et optimiser le code JavaScript.
- Les bundlers offrent des avantages tels que la gestion des dépendances, l'optimisation du code, et la réduction de la taille finale du fichier.

2.6 Conclusion

- Chaque méthode a ses avantages et inconvénients, adaptés aux besoins du projet.
- L'inclusion inline est utile pour de petits morceaux de code, la balise script ou un fichier externe améliorent la maintenabilité, et l'utilisation d'un bundler est bénéfique pour des projets de grande envergure.

3. Variable

3.1 Les Types de Variables en JavaScript

String : représente des données textuelles.

Number : représente un nombre entier ou à virgule flottante.

Boolean : représente une valeur logique, soit true (vrai) soit false (faux).

Undefined : représente une variable qui n'a pas été initialisée et dont la valeur est indéfinie.

Null : représente une valeur nulle ou vide.

BigInt : représente un entier avec une précision arbitraire.

Symbol : représente une valeur unique et immuable.

Object : représente une collection de données sous la forme de paires clé-valeur.

Array : représente une collection ordonnée d'éléments.

3.2 Déclaration et Initialisation des Variables

En JavaScript, nous utilisons les mots-clés **var**, **let** et **const** pour déclarer et initialiser des variables.

- **var** : déclarer une variable dont la portée est fonctionnelle. Cela signifie que la variable est accessible à l'intérieur de la fonction dans laquelle elle est déclarée, quelle que soit la portée des blocs de code {}. Les variables déclarées avec var peuvent être réaffectées et leur valeur peut être modifiée.

```
for (var i=1; i<=10; i++) {
        var j = i // 1, 2, 3 … 10
}
j // 10
i // 11
```

- **let** : déclarer une variable dont la portée est limitée au bloc de code {} dans lequel elle est déclarée. Les variables déclarées avec let ne peuvent pas être réaffectées dans la même portée, mais leur valeur peut être modifiée.

```
for (let i=1; i<=10; i++) {
        let j = i // 1, 2, 3 … 10
}
j // undefined
i // undefined
```

- **const** : déclarer une variable constante dont la valeur ne peut pas être modifiée une fois qu'elle a été assignée. Les variables déclarées avec const doivent être initialisées avec une valeur au moment de leur déclaration et ne peuvent pas être réaffectées.

Il est recommandé d'utiliser const par défaut pour déclarer des variables, sauf si vous savez que vous allez réaffecter ou modifier la valeur de la variable ultérieurement. Cela permet de rendre le code plus lisible et de prévenir les erreurs de réaffectation accidentelle.

https://www.w3schools.com/js/js_variables.asp

3.3 Types de données

Pour identifier les différents types de données, nous aurons besoin de l'opérateur typeof. Il est possible de lui transmettre une variable, une constante ou une valeur :

```
// Utilisation de l'opérateur "typeof".
const quisuisje = 18;
console.log(typeof quisuisje); // Affiche: "number" (nombre)
console.log(typeof 'du texte'); // Affiche: "string" (chaîne de caractère)
```

4. Commentaires

Il est possible d'intégrer des commentaires dans du code JavaScript, de la même manière que dans les CSS :

```
/* Tout ce qui est écrit ici est entre commentaires. */
```

Si votre commentaire tient sur une ligne, vous pouvez utiliser deux barres obliques pour indiquer un commentaire :

```
// Voici un commentaire
```

5. Opérateurs

5.1 Opérateurs mathématiques

+ : addition

- : soustraction

* : multiplication

/ : division

% : modulo (reste de la division)

** : exponentiation (élévation à la puissance)

```
cost x = 5;
const y = 3;
const addition = x + y; // 8
const multiplication = x * y; // 15
```

```
const division = x / y; // 1.6666666666666667
```

5.1 Opérateurs de chaînes de caractères

+ : concaténation

```
const message1 = "Bonjour";

const message2 = "le monde";

const messageConcatene = message1 + " " + message2; // "Bonjour le monde"
```

5.2 Opérateurs booléens

== : égal à

!= : différent de

=== : strictement égal à

!== : strictement différent de

> : supérieur à

< : inférieur à

>= : supérieur ou égal à

<= : inférieur ou égal à

! : négation

&& : ET logique

|| : OU logique

```
const x = 5;

const y = 3;

const estEgal = x == y; // false

const estDifferent = x != y; // true

const estSuperieur = x > y; // true

const estInferieurOuEgal = x <= y; // false
```

6. Chaine de caractères

6.1 Différentes façons d'écrire des chaines de caractères

Utilisation de guillemets simples '

```
const chaine1 = 'Ceci est une chaine de caractères';
```

Utilisation de guillemets doubles "

```
const chaine2 = "Ceci est une autre chaine de caractères";
```

Utilisation du back quote `

```
const nom = "John";

const age = 25;

const chaine3 = `Je m'appelle ${nom} et j'ai ${age} ans`;

//chaine3 contient « je m'appelle John et j'ai 25 ans »
```

6.2 Méthodes sur les chaines de caractères

- `length()` : Retourne la longueur d'une chaine de caractères.

```
const chaine = "Hello";

console.log(chaine.length); // Affiche 5
```

- `toUpperCase()` : Convertit une chaine de caractères en majuscules.

```
const chaine = "hello";

console.log(chaine.toUpperCase()); // Affiche "HELLO"
```

- `toLowerCase()` : Convertit une chaine de caractères en minuscules.

```
const chaine = "WORLD";

console.log(chaine.toLowerCase()); // Affiche "world"
```

- `charAt(index)` : Retourne le caractère à l'index spécifié.

```
const chaine = "Hello";

console.log(chaine.charAt(1)); // Affiche "e"
```

- `substring(start, end)` : Extrait une partie de la chaine de caractères.

```
const chaine = "Hello world";

console.log(chaine.substring(0, 5)); // Affiche "Hello"
```

- `split(separator)` : Divise la chaine de caractères en un tableau de sous-chaines.

```
const chaine = "Hello,world";

console.log(chaine.split(",")); // Affiche ["Hello", "world"]
```

replace(oldValue, newValue) :

Remplace une sous-chaine par une autre.

```
const chaine = "Hello world";

console.log(chaine.replace("world", "JavaScript")); // Affiche "Hello JavaScript"
```

Ces méthodes permettent d'effectuer diverses manipulations sur les chaines de caractères selon les besoins du programme.

https://www.w3schools.com/jsref/jsref_obj_string.asp

7. Tableaux

7.1 Déclaration et initialisation des tableaux

Un tableau vide

```
const tableau = [];
```

Un tableau avec des valeurs

```
const tableau = [1, 2, 3, 4, 5];
```

Un tableau multidimensionnel est un tableau qui contient d'autres tableaux à l'intérieur.

```
const tableauMultidimensionnel = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];
```

7.2 Accès aux éléments d'un tableau

```
const tableau = [1, 2, 3, 4, 5];

console.log(tableau[0]); // Affiche 1

console.log(tableau[2]); // Affiche 3
```

7.3 Méthodes sur les tableaux

7.3.1Ajout/Suppression d'éléments à un tableau

La méthode push() permet d'ajouter un ou plusieurs éléments à la fin d'un tableau. La méthode pop() permet de supprimer le dernier élément d'un tableau et de le renvoyer.

Méthode push().

```
const tableau = [1, 2, 3];

tableau.push(4);

console.log(tableau); // Affiche [1, 2, 3, 4]
```

Méthode pop()

```
const tableau = [1, 2, 3, 4];

tableau.pop();

console.log(tableau); // Affiche [1, 2, 3]
```

7.3.2 shift() et unshift()

La méthode shift() permet de supprimer le premier élément d'un tableau et de le renvoyer. La méthode unshift() permet d'ajouter un ou plusieurs éléments au début d'un tableau.

```
const tableau = [1, 2, 3];

tableau.shift();

console.log(tableau); // Affiche [2, 3]


tableau.unshift(0);

console.log(tableau); // Affiche [0, 2, 3]
```

7.3.3 concat()

La méthode concat() permet de fusionner deux tableaux (ou plus) en créant un nouveau tableau.

```
const tableau1 = [1, 2];

const tableau2 = [3, 4];

const nouveauTableau = tableau1.concat(tableau2);
```

```
console.log(nouveauTableau); // Affiche [1, 2, 3, 4]
```

7.3.4 slice()

La méthode slice() permet de créer une copie superficielle d'une partie d'un tableau dans un nouveau tableau.

```
const tableau = [1, 2, 3, 4, 5];

const sousTableau = tableau.slice(1, 3);

console.log(sousTableau); // Affiche [2, 3]
```

7.3.5 indexOf() et lastIndexOf()

La méthode indexOf() permet de rechercher la première occurrence d'un élément dans un tableau et de renvoyer son index. La méthode lastIndexOf() permet de rechercher la dernière occurrence d'un élément dans un tableau et de renvoyer son index.

```
const tableau = [1, 2, 3, 2, 1];

console.log(tableau.indexOf(2)); // Affiche 1

console.log(tableau.lastIndexOf(2)); // Affiche 3
```

7.3.6 sort()

La méthode sort() permet de trier les éléments d'un tableau dans l'ordre lexicographique (par défaut) ou en utilisant une fonction de comparaison personnalisée.

```
const tableau = [3, 1, 2];

tableau.sort();

console.log(tableau); // Affiche [1, 2, 3]
```

7.3.7 reverse()

La méthode reverse() permet d'inverser l'ordre des éléments d'un tableau.

```
const tableau = [1, 2, 3];

tableau.reverse();

console.log(tableau); // Affiche [3, 2, 1]
```

Exercices utilisation tableau

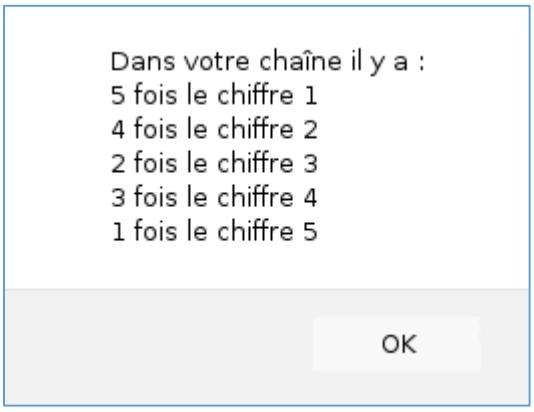
1. **tableau1.html** - Calcul de la somme des éléments d'un tableau

Par exemple, le tableau

```
const tableau = [1 5,8,3,9,10,25,4];
```

Affichera 65

2.tableau2.html - Ecrivez un code qui demande à l'utilisateur de saisir une chaîne de caractères ne contenant **que** des chiffres et qui affiche dans une fenêtre le nombre d'occurences de chaque chiffre apparaissant au moins une fois dans la chaîne. Par exemple, si l'utilisateur saisit la chaîne `"112522312411443"`, le programme affiche la fenêtre suivante :



On peut supposer que la chaîne de caractères saisie par l'utilisateur est valide (c'est à dire ne contient que des chiffres).

3. tableau3.html

Ecrivez un programme qui demande à l'utilisateur de saisir une chaîne de caractères contenant des mots. Celui-ci affichera la liste des mots triée alphabétiquement.

8. Conditions

Les structures de contrôle conditionnelles permettent d'exécuter des blocs de code en fonction du résultat d'une condition.

8.1 if then else

```
if (condition) {
   // Code à exécuter si la condition est vraie
}
if (condition) {
   // Code à exécuter si la condition est vraie
```

```
} else {
   // Code à exécuter si la condition est fausse
}
```

8.2 switch

```
switch (expression) {
   case valeur1:
      // Code à exécuter si l'expression est égale à valeur1
      break;
   case valeur2:
      // Code à exécuter si l'expression est égale à valeur2
      break;
   default:
      // Code à exécuter si aucune des valeurs ne correspond à l'expression
      break;
}
```

https://www.w3schools.com/js/js_if_else.asp

Exercices conditions

1. Condition1.html A l'aide de la fonction prompt(), demander un prix HT puis un taux de la TVA à l'utilisateur puis insérer le prix TTC dans la page HTML à l'aide de document.body.innerHTML. Si le prix TTC est strictement supérieur à 100 on affichera le prix en rouge sinon en vert.

2. condition2.html - Ecrivez un programme qui demande à l'utilisateur sa moyenne et affiche le grade obtenu de l'étudiant.

Exemple :

0≤moyenne <10 : refusé

10≤moyenne <14 : satisfaction

14≤moyenne <16 : distinction

16≤moyenne <18 : grande distinction

18≤moyenne <20 : la plus grande distinction

9. Boucles

Une boucle est une structure de contrôle qui permet de répéter un bloc de code tant qu'une condition est vraie. Elle permet d'exécuter plusieurs fois les mêmes instructions sans avoir à les répéter manuellement.

9.1 Les boucles for

```
for (initialisation; condition; incrémentation) {
   // bloc de code à exécuter
}
```

Exemple :

```
const fruits = ["pomme", "banane", "orange", "kiwi"];
for (let i = 0; i < fruits.length; i++) {
   console.log(fruits[i]);
}
```

9.2 Les boucles while

```
while (condition) {
   // bloc de code à exécuter
}
```

Exemple :

```
let i = 1;
while (i <= 5) {
   console.log(i);
   i++;
```

9.3 Les boucles do-while

La boucle do-while est utilisée lorsque vous souhaitez exécuter une action au moins une fois, puis continuer à répéter cette action tant qu'une condition spécifiée est vraie

```
do {
   // bloc de code à exécuter
} while (condition);
```

Exemple, demande à l'utilisateur de saisir un nombre positif *:*

```
let number;
do {
   number = parseInt(prompt("Entrez un nombre positif :"));
} while (number <= 0);

console.log("Le nombre saisi est :", number);
```

Exercices boucles

1. boucle1.html - Calcul de la somme des nombres de 1 à 10

2. boucle2.html - Grâce à la boucle de votre choix, afficher en console tous les multiples de 10 jusqu'à 1000.

3. boucle3.html - Génération d'une séquence de nombres en doublant la valeur précédente

4. boucle4.html - Affichage des nombres pairs de 1 à 10

5. boucle5.html - Affichage de la table de multiplication d'un nombre **:** Demandez à l'utilisateur de saisir un nombre, puis afficher la table de multiplication de ce nombre jusqu'à 10.

6. boucle6.html - Affichage d'une séquence de Fibonacci : Écrivez un programme qui affiche les n premiers termes de la séquence de Fibonacci, où n est saisi par l'utilisateur.

7. boucle7.html – L'objectif de l'exercice suivant est de réaliser un petit programme avec la boucle while qui vous demandera de trouver un nombre mystère :

1. Déclarez et initialisez la constante **nombreMystere** avec la méthode **Math.random()** pour qu'elle renvoie un nombre entier de façon pseudo-aléatoire compris entre 1 et 20.

```
const nombreMystere = Math.floor(Math.random() * 20) + 1;
```

```
/* génère un nombre entier aléatoire compris entre 1 (inclus) et 20 (inclus) */
```

2. Déclarez et initialisez la variable **compteur** qui sera utiliser pour stocker le nombre de tentatives.
3. Utilisez la fonction **prompt()** pour que le programme vous demande de proposer un nombre jusqu'au moment votre nombre sera égal à celui du nombre mystère.
4. Si le nombre que vous proposez n'est pas le bon, le programme vous affichera ceci dans la console avant de vous demander une nouvelle proposition :
   - Si le nombre mystère est plus petit que le nombre proposé : "Le nombre mystère est plus petit!"
   - Si le nombre mystère est plus grand que le nombre proposé : "Le nombre mystère est plus grand!"
5. Finalement, lorsque le nombre mystère est trouvé, le programme affichera ceci : `Félicitations, vous avez découvert le nombre mystère : **${nombreMystere}** en **${compteur}** tentatives.`

Pour aller plus loin :

Refaites les exercices 4 du cours d'Approche developpement Backend en javascript.

10. Fonctions et méthodes

10.1 Objets

Les objets sont un élément fondamental de JavaScript. Ils vous permettent de **stocker et d'organiser des données de manière structurée** en utilisant des paires clé-valeur.

```
const personne = {
  nom: 'Claudy',
  age: 52,
  profession: 'Photographe'
};
```

Dans cet exemple, personne est un objet qui a trois propriétés : nom, age, et profession. Chacune de ces propriétés a une valeur associée.

10.1.1 Accès aux propriétés d'un objet

Pour accéder aux propriétés d'un objet, vous pouvez utiliser la notation point objet.nomDeLaPropriété ou la notation crochets objet['nomDeLaPropriété'] :

```
const personne = {
   nom: 'Claudy',
   age: 52,
   profession: 'Photographe'
};
console.log(personne.nom); // Affiche : 'Claudy'
console.log(personne['age']); // Affiche : 52.
```

10.1.2 Modification d'un objet

Les objets sont mutables, ce qui signifie que vous pouvez modifier leurs propriétés après leur création :

```
const personne = {
   nom: 'Claudy',
   age: 52,
   profession: 'Photographe'
};
// Modification de la propriété 'age'.
personne.age = 53;
personne['profession'] = 'contremaître';
console.log(personne);
/* Affiche :
   Object { nom: "Claudy", age: 53, profession: "contremaître" }
*/
```

10.1.3 Ajout de nouvelles propriétés

Vous pouvez également ajouter de nouvelles propriétés à un objet existant à tout moment :

```
const personne = {
   nom: 'Claudy',
   age: 52,
   profession: 'Photographe'
};
// Ajout d'une nouvelle propriété 'ville'.
personne.ville = 'Bruxelles';
personne['codePostal'] = 1000;
console.log(personne);
/* Affiche :
   Object { nom: "Claudy", age: 52, profession: "Photographe", ville: "Bruxelles", codePostal: 1000 }
*/
```

10.1.4 Suppression de propriétés

Pour supprimer une propriété d'un objet, vous pouvez utiliser l'opérateur **delete :**

```
const personne = {
   nom: 'Claudy',
   age: 52,
   profession: 'Photographe'
};
// Suppression de la propriété 'profession'.
delete personne.profession;
console.log(personne.profession); // Affiche : undefined
```

Exercice objets

1. Manipulationobjet.html - Manipulez un objet contenant les informations du film « Les survivants » :

1. Créez un objet avec la structure suivantes :
   - nom du film : Les survivants
   - réalisateur : Guillaume Renusson
2. Affichez le nom du réalisateur dans la console.
3. Remplacez le nom du réalisateur par : Frank Marshall.
4. Affichez le nom du réalisateur dans la console.
5. Ajoutez l'année de sortie du film dans ses propriétés : 1993.
6. Affichez toutes les propriétés de l'objet

10.2 Différence entre les fonctions et les méthodes

En JavaScript, une fonction est un bloc de code réutilisable qui peut être appelé à partir d'autres parties du programme. Les fonctions peuvent prendre des paramètres en entrée et renvoyer une valeur en sortie.

D'autre part, une méthode est une fonction qui est associée à un objet spécifique. Elle est appelée sur cet objet à l'aide de la syntaxe de point (objet.méthode()). Les méthodes sont souvent utilisées pour effectuer des opérations spécifiques sur un objet donné.

Exemples de fonctions :

```
console.log()
```

Afficher des messages dans la console du navigateur

```
Math.random()
```

Renvoie un nombre aléatoire compris entre 0 et 1.

```
parseInt() :
```

Convertir une chaîne de caractères en nombre entier.

```
parseFloat()
```

Convertir une chaîne de caractères en un nombre à virgule flottante (nombre décimal).

```
isNaN()
```

Vérifie si une valeur est de type NaN (Not a Number).

```
Date.now()
```

Renvoie le nombre de millisecondes écoulées depuis le 1er janvier 1970

Exemples de méthodes :

```
string.length
```

Obtenir la longueur d'une chaîne de caractères.

```
array.push()
```

Utilisée pour ajouter un élément à la fin d'un tableau.

```
const fruits = ["apple", "banana", "orange"];

fruits.push("grape");

console.log(fruits);
```

```
string.toUpperCase()
```

Renvoie une nouvelle chaîne de caractères avec tous les caractères en majuscules.

```
string.toLowerCase()
```

Renvoie une nouvelle chaîne de caractères avec tous les caractères en minuscules.

11. Sélecteurs en JavaScript

Un sélecteur en JavaScript est une méthode qui permet de cibler et de sélectionner des éléments spécifiques dans une page HTML. Ces éléments peuvent être des balises, des classes, des IDs, des attributs ou d'autres critères.

Les sélecteurs en JavaScript sont utilisés pour interagir avec les éléments sélectionnés, tels que la modification de leur contenu, l'ajout ou la suppression de classes CSS, la modification des attributs, etc.

11.1 Méthodes

- **querySelector() :** Cette méthode permet de sélectionner le premier élément correspondant à un sélecteur CSS spécifié.
  Sélectionner l'élément avec l'ID "myElement" :

```
document.querySelector("#myElement")
```

- **querySelectorAll()** : Cette méthode permet de sélectionner tous les éléments correspondant à un sélecteur CSS spécifié et de les stocker dans une liste.
  Sélectionner tous les éléments ayant la classe "myClass".

```
document.querySelectorAll(".myClass")
```

```
const allElements = document.querySelectorAll('*');
```

>> la variable `allElements` contiendra une liste de tous les éléments de la page.

- getElementById() : sélectionner des éléments en fonction de leur ID
- getElementsByClassName() sélectionner des éléments en fonction de leur classe

```
const elements = document.getElementsByClassName('myClass');
```

Dans cet exemple, la variable elements contiendra une liste de tous les éléments ayant la classe "myClass".

- getElementsByTagName(),sélectionner des éléments en fonction de leur balise HTML spécifique

```
const paragraphs = document.getElementsByTagName('p');
```

>> `paragraphs` contiendra une liste de tous les éléments `<p>` présents dans la page.

Sélectionner tous les éléments `<h1>` :

```
const headings = document.getElementsByTagName('h1');
```

Sélectionner tous les éléments `<a>` :

```
const links = document.getElementsByTagName('a');
```

Sélectionner tous les éléments `<li>` :

```
const listItems = document.getElementsByTagName('li');
```

Combinaison de classes et d'IDs pour des sélections plus précises :

```
const element = document.querySelector('.myClass#myId');
```

La variable `element` contiendra l'élément ayant à la fois la classe "myClass" et l'ID "myId"

11.2 Manipulation des éléments sélectionnés

11.2.1 Modifier le contenu des éléments sélectionnés

```
const element = document.querySelector('.myElement');

element.textContent = 'Nouveau texte';
```

Modifie le texte à l'intérieur d'un élément en utilisant la propriété `textContent`.

Modifier le style d'un élément :

```
const element = document.querySelector('.myElement');

element.style.color = 'red';

element.style.fontSize = '20px';
```

11.2.2 Modification du contenu HTML

Si vous souhaitez modifier le contenu HTML d'un élément, vous pouvez utiliser la propriété `innerHTML`. Cette propriété permet de récupérer ou de définir le contenu HTML à l'intérieur d'un élément.

```
// Récupérer le contenu HTML d'un élément

const html = element.innerHTML;

// Définir le contenu HTML d'un élément

element.innerHTML = "<p>Nouveau contenu</p>";
```

11.3 Gestion des événements

La gestion des événements dans la DOM[2] nous permet de détecter et de réagir aux événements (tels que des clics de souris, des pressions de touches ou des mouvements de la souris…) en exécutant du code JavaScript spécifique.

11.3.1 Ajout de gestionnaires d'événements

addEventListener

```
element.addEventListener('click', function() {

  // Code à exécuter lorsque l'événement click se produit
```

[2] Le DOM, ou Document Object Model, est une représentation hiérarchique et structurée des éléments d'une page web

```
});
```

Ajout d'un gestionnaire d'événements pour l'événement "click" à l'élément spécifié. Lorsque l'utilisateur clique sur cet élément, la fonction sera exécutée

Exemples complets :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Test</title>
<body>
   <button id="monBouton">Mon bouton</button>
  <script>
    const bouton = document.getElementById('monBouton');
     function afficherAlerte() {
       alert("Bouton cliqué !");
     }
     bouton.addEventListener('click', afficherAlerte);
</script>
</head>
 </body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Test</title>
<body>
  <button id="monBouton">Mon bouton</button>
  <script>
    const bouton = document.getElementById('monBouton');
    function changerCouleur() {
      bouton.style.backgroundColor = "blue";
    }
    bouton.addEventListener('click', changerCouleur);
  </script>
</head>
</body>
</html>
```

11.3.2 Écouter les événements souris

`click`, `mouseover`, `mouseout`, `mousedown`, `mouseup`, etc.

11.3.3 Écouter les événements de clavier

`keydown`, `keyup` et `keypress`

11.3.4 Écouter les événements de formulaire

`submit`, `input`, `change`, etc.

Exemple :

« event.preventDefault() » empêche la soumission par défaut du formulaire, ce qui nous permet de gérer la soumission du formulaire de manière personnalisée.

```
// Sélectionner le formulaire
const formulaire = document.querySelector('#monFormulaire');
// Ajouter un gestionnaire d'événements pour l'événement 'submit'
formulaire.addEventListener('submit', function(event) {
```

```
  event.preventDefault(); // Empêcher la soumission par défaut du formulaire

  // Code à exécuter lorsque le formulaire est soumis

});
```

Exercices Sélecteur / Evènements

1. formulaireEmail.html - Valider un formulaire en temps réel : afficher un message d'erreur si l'adresse e-mail n'est pas valide ou si le mot de passe est trop court

2. diaporamaiamge.html – créer, en javascript, un diaporama d'image avec deux boutons (previous, next)

3. ensavoirplus.html - Créez un programme qui permet de masquer ou de révéler le contenu d'un conteneur en le pliant ou en le dépliant lorsque le bouton associé est activé.

4. coordonneesouris.html – Créez un programme qui affiche les coordonnées de la souris en temps réel.

5. Suivresouris.html - Créez un programme qui permet de faire suivre le curseur de la souris avec une image.

6. fleches.html – à l'aide de l'événement « event.key » affiche à l'écran les flèches du clavier :

←→↑←→↓←→↓↑←→↓←→↑↓↓↓↓↓↓↑←→↓←→↑←

7. Listevilles.html – Créer une liste déroulante des villes qui se met à jour en fonction d'une liste de pays

14. Document Object Model (DOM)

- **Structure hiérarchique :** Le DOM organise les éléments de la page web sous forme d'une structure arborescente, où chaque élément est un nœud dans l'arbre.
- **Accès et manipulation :** Le DOM fournit une interface de programmation qui permet aux développeurs d'accéder aux éléments de la page et de les manipuler dynamiquement à l'aide de langages de script comme JavaScript.
- **Interopérabilité :** Le DOM est une spécification standardisée par le W3C (World Wide Web Consortium), ce qui signifie qu'il est largement pris en charge par les navigateurs web modernes. Cela garantit une certaine cohérence dans la manière dont les développeurs peuvent interagir avec les éléments des pages web.
- **Modèle objet :** Chaque élément de la page web est représenté par un objet dans le DOM. Ces objets peuvent être manipulés en ajoutant, supprimant ou modifiant des éléments de la page, ainsi que leurs attributs et leur contenu.
- **Dynamicité :** Le DOM permet de créer, modifier et supprimer des éléments de la page web de manière dynamique en réponse à des événements utilisateur ou à des actions de script.

14.1 Création d'éléments avec `createElement()`

`createElement()` est une méthode JavaScript qui crée un nouvel élément HTML avec le nom spécifié.

```
document.createElement('p').
```

14.2 Modification des éléments créés

Après avoir créé un élément avec `createElement()`, vous pouvez définir ses propriétés (comme `innerHTML`, `textContent`, `className`, etc.) ou ses attributs (comme `id`, `class`, etc.) en utilisant les propriétés JavaScript.

```
// Créer un lien
var link = document.createElement('a');
// Définir le texte du lien
link.textContent = "Cliquez ici";
// Définir l'URL du lien
link.href = "https://www.example.com";
```

```
// Définir la classe du lien

link.className = "exemple";
```

14.3 Ajout d'éléments dans le DOM avec `appendChild()`

Une fois que vous avez créé un élément et configuré ses propriétés, vous devez l'ajouter à votre page web.

Vous pouvez le faire en utilisant `appendChild()`. Cette méthode attache un nœud (élément) à la fin d'une liste d'enfants d'un nœud parent spécifié.

Par exemple, si vous avez créé un paragraphe (`<p>`) avec `createElement('p')` et que vous souhaitez l'ajouter à un élément div existant avec l'ID "container", vous pouvez le faire comme ceci :

```
var div = document.getElementById('container');

var paragraph = document.createElement('p');

paragraph.textContent = 'Contenu du paragraphe';

div.appendChild(paragraph);
```

14.4 Autres méthodes pour ajouter des éléments

En plus de `appendChild()`, vous pouvez également utiliser d'autres méthodes pour ajouter des éléments, comme `insertBefore()`, `replaceChild()`, etc., selon vos besoins spécifiques.

Exemple complet d'ajout dynamique de paragraphe :

```
<!DOCTYPE html>

<html lang="fr">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Création dynamique d'éléments avec le DOM</title>

</head>

<body>

  <h1>Création dynamique d'éléments avec le DOM</h1>
```

```
  <!-- Bouton pour ajouter un paragraphe -->
  <button id="ajouterParagraphe">Ajouter un paragraphe</button>
  <!-- Conteneur où les paragraphes seront ajoutés -->
  <div id="paragraphes"></div>
</body>


<script>
    const ajouterParagraphe=document.getElementById('ajouterParagraphe');
    ajouterParagraphe.addEventListener('click', function() {
        // Création d'un paragraphe
        var nouveauParagraphe = document.createElement('p');
        nouveauParagraphe.textContent = "Ceci est un nouveau paragraphe ajouté
dynamiquement.";
        // Ajout du paragraphe au conteneur
        document.getElementById('paragraphes').appendChild(nouveauParagraphe);
    });
  </script>
</html>
```

Exercices - DOM

tache.html – Réaliser un code javascript qui affiche des taches à l'écran dès qu'on clique avec la souris

Ajoutparagraphe.html - Modifier le code précédant pour ajouter un paragraphe contenant du texte que l'utilisateur tape au clavier.

15. Exercices récapitulatifs

pierrepapierciseaux.html

Réaliser le jeu pierre/papier/ciseaux en javascript

pendu.html

Réaliser le jeu du pendu en javascript

nbaleatoire.html

L'ordinateur génère un nombre aléatoire entre 1 et 100, et le joueur doit deviner ce nombre (l'ordinateur lui précise, plus grand ou plus petit et compte le nombre d'essais)

question.html

Affichez une question avec plusieurs options de réponse. Le joueur doit choisir la bonne réponse parmi les options proposées.

dactylo.html

Le joueur doit appuyer sur une touche spécifique du clavier dès qu'elle est affichée à l'écran.

Couleur.html - Changement de couleur du fond

Dans cet exercice, vous devez créer un bouton qui permet de changer la couleur de fond d'une page HTML. Lorsqu'on clique sur le bouton, la couleur du fond doit passer du bleu au rouge. Si on reclique sur le bouton, la couleur du fond doit revenir au rouge.

Like.html - Bouton "Like/Dislike"

Dans cet exercice, vous allez créer un bouton "Like/Dislike" en utilisant HTML, CSS et JavaScript. Ce bouton permettra aux utilisateurs d'indiquer s'ils aiment ou n'aiment pas un contenu spécifique.

Compteur.html - Compteur de clics

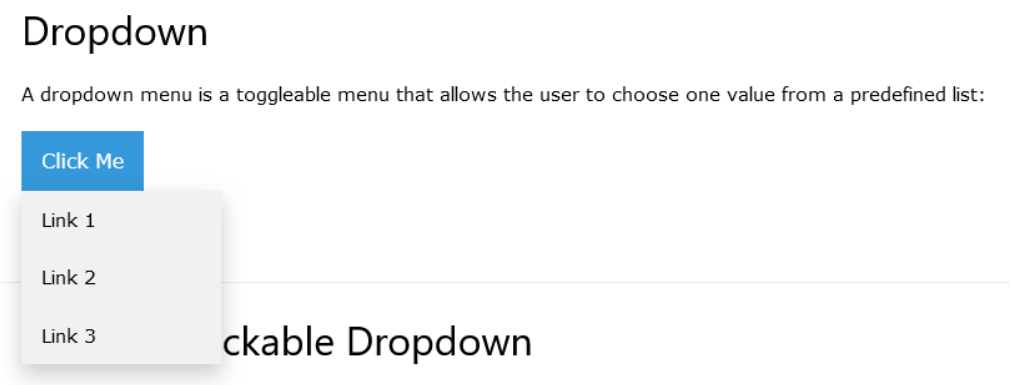
Dans cet exercice, vous allez créer un compteur de clics en utilisant HTML, CSS et JavaScript. Ce compteur permettra de suivre le nombre de fois où l'utilisateur clique sur un bouton spécifique.

Color.html - Changement de couleur de fond avec un sélecteur de couleur

Dans cet exercice, vous allez créer un formulaire avec un sélecteur de couleur et un bouton. L'objectif est de permettre à l'utilisateur de choisir une couleur à l'aide du sélecteur de couleur, puis de changer la couleur de fond de la page en cliquant sur le bouton

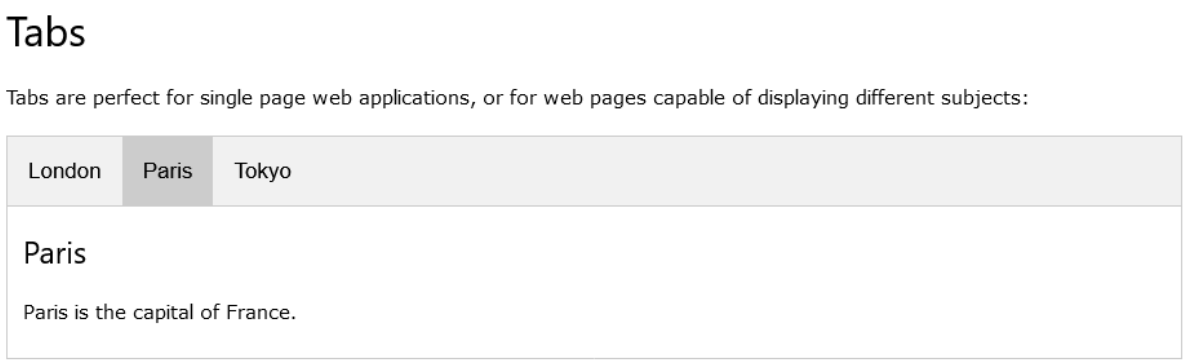
Menudropdown.html - Menu dropdown

Dans cet exercice, vous allez créer un menu dropdown en utilisant HTML, CSS et JavaScript. Un menu dropdown est un élément de navigation qui affiche une liste d'options lorsque l'utilisateur clique dessus.



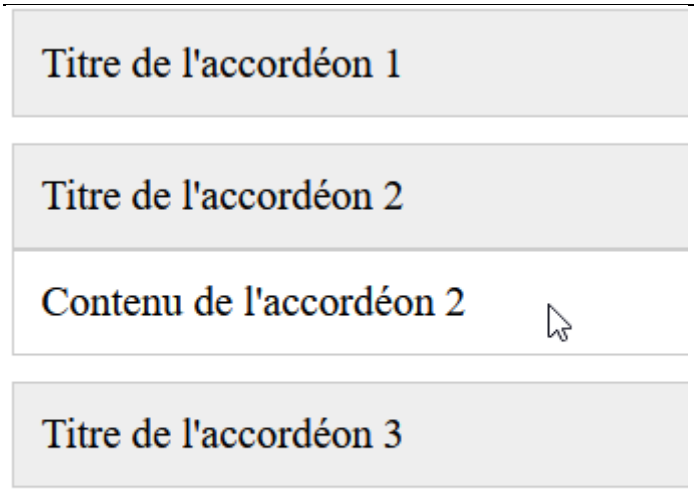
Tabs.html - Un élément avec plusieurs "tabs"

Dans cet exercice, vous allez créer un élément avec plusieurs "tabs" ou onglets en utilisant HTML, CSS et JavaScript. Chaque onglet affichera un contenu différent lorsque l'utilisateur clique dessus.



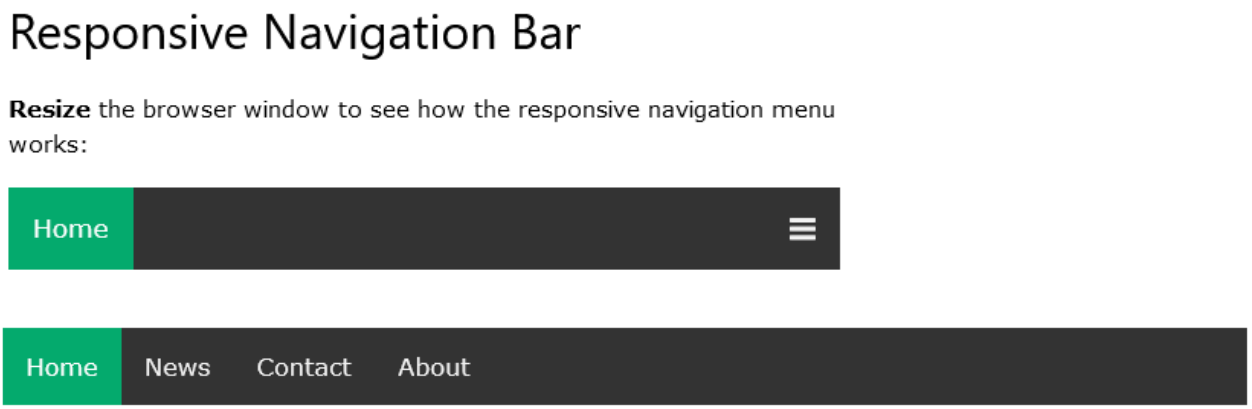
Accordeon.html - Accordéon

Dans cet exercice, vous allez créer un accordéon interactif en utilisant HTML et JavaScript. Un accordéon est un élément qui comporte un titre et une description, et lorsque l'utilisateur clique sur le titre, la description s'affiche ou se cache.



Navigation.html - Navigation responsive avec menu hamburger

Dans cet exercice, vous allez créer une navigation responsive avec un menu hamburger en utilisant HTML, CSS et JavaScript. Cette navigation s'adaptera à différents appareils et affichera un menu hamburger lorsqu'elle est consultée sur des écrans plus petits.



Sources :

https://www.w3.org/WAI/ARIA/apg/patterns/

Face à une structure informatique opérationnelle connectée à Internet, disposant des logiciels appropriés et de la documentation nécessaire, en utilisant le vocabulaire technique et l'orthographe adéquate, en respectant les normes et standards en vigueur,

et au départ d'un cahier des charges contenant un projet d'application web interactives :

- de développer une solution répondant aux consignes/objectifs du cahier des charges en développant et exploitant une bibliothèque tierce ou framework.

Pour la détermination du degré de maîtrise, il sera tenu compte des critères suivants :

- le niveau de pertinence technique du code en relation avec les consignes du cahier des charges : l'optimalisation du code et la pertinence des commentaires;
- le niveau de pertinence des choix techniques des outils d'aide à l'optimalisation du code (bundler, …)
- le niveau de lisibilité du code, (facilité de partage, indentation, conventions et respect des bonnes pratiques d'écriture, …)
- le degré d'autonomie atteint dans l'exploitation et l'utilisation de la documentation référencée.