

Interaction web



Pour vérifier les réponses aux exercices, effectuer des recherches (par exemple sur Internet).

Exercice 1 (Vrai/Faux)

Pour chacune des affirmations suivantes, dire si elle est vraie ou fausse.

1. Les documents hypertextes ne se trouvent que dans des pages Web.
2. Si du code JavaScript est écrit dans un fichier séparé, il peut être utilisé dans plusieurs pages Web distinctes.
3. Le code JavaScript d'une page Web est exécuté sur la machine du client.
4. Le code JavaScript d'une page Web est toujours exécuté après le chargement de la page.
5. Le code JavaScript dépendant d'un événement Mouseover n'est pas exécuté s'il n'y a pas de souris.

Exercice 2 (QCM)

Pour chaque question, une seule réponse parmi celles proposées est exacte.

1. Durant quelle décennie le World Wide Web s'ouvre-t-il au public ?
(a) 1970 (b) 1980 (c) 1990 (d) 2000
2. Parmi les termes suivants, lequel ne désigne pas un protocole qui est utilisé dans la communication sur un réseau ?
(a) IP (b) HTTP (c) PHP (d) TCP
3. Tim Berners Lee, principal inventeur du World Wide Web (www) a travaillé dans de nombreux domaines. Dans lequel n'a-t-il pas participé ?
(a) PHP (b) HTML (c) HTTP (d) URL
4. Il existe de nombreux événements qui peuvent être gérés par JavaScript. Parmi les termes suivants, lequel ne désigne pas un événement ?
(a) Dblclick (b) Mouseout (c) Keypush (d) Submit
5. Pour gérer un événement Click dans un paragraphe limité par les balises `<p>` et `</p>`, on utilise l'attribut `OnClick`. À quel endroit écrit-on cet attribut ?
(a) avant `<p>` (b) dans `<p>` (c) dans `</p>` (d) entre `<p>` et `</p>`
6. Quelle proposition est vraie ?
(a) Un site dynamique est un site qui change souvent d'adresse.
(b) Un site dynamique est un site qui s'adapte au visiteur.
(c) Un site statique est un site qui ne contient pas de lien vers d'autres sites.
(d) Un site statique est un site qui ne contient pas d'animation.

Pour les exercices suivants, on récupérera les fichiers de départ sur Moodle.

Exercice 3 (Pas à Pas)

La page HTML `pasapas.html` contient une liste d'étapes dont la première est mise en évidence grâce à la classe CSS `courant`.

Les étapes sont numérotées grâce à leur attribut `id`.

Le code JavaScript contient deux fonctions qui permettent de mettre en évidence (avec la fonction `en_evidence(etape)`) ou non (avec `normal(etape)`) un élément de la liste à partir de son numéro `etape`.

Ces deux fonctions utilisent la fonction `pas(etape)` qui retourne l'élément de numéro `etape`.

Une variable globale `i`, qui vaut 1 au départ, désigne l'étape courante et devra être gérée par la suite.

1. Modifier le contenu de la page en ajoutant deux boutons '<<' et '>>' qui permettent de mettre en évidence l'étape précédente ou l'étape suivante de la liste.
Attention : l'étape `i` doit toujours rester entre 1 et 5; il n'y a en particulier pas d'étape précédente quand on est à la première étape.
2. Ajouter deux boutons '|<' et '>|' qui permettent d'aller directement respectivement à la première et à la dernière étape.
3. Ajouter une règle CSS pour que le texte des éléments de la liste soit blanc.

Exercice 4 (Convertisseur)

La page HTML `conversion.html` affiche un élément `<input>` de type `range`, c'est-à-dire une tirette permettant de saisir une valeur numérique.

1. Étudier le code de cette page. En particulier :
 - (a) Comprendre l'utilité de l'élément `<span>`;
 - (b) Trouver quel est l'événement capté sur l'élément `<input>`;
 - (c) Comprendre ce que fait la fonction de rappel de cet événement.
2. Définir dans le fichier `script.js` une fonction `Celcius2Fahrenheit(c)` qui retourne la valeur en degrés Fahrenheit d'une température donnée en degrés Celsius `c`.
On donne la formule suivante de conversion : $F = C \times \frac{9}{5} + 32$.
3. Modifier les fichiers `conversion.html` et `script.js` pour que s'affiche dans la page HTML le résultat de la conversion du nombre spécifié par la tirette de degrés centigrades (C) en degrés Fahrenheit (F).

Le texte devra être de la forme (cas où la température en Celsius vaut 100) :

Température en degrés Fahrenheit : 212

La conversion doit être faite dès le chargement de la page.

4. Ajouter un bouton 'F<->C' permettant d'inverser de sens de conversion.
Pour cela, il sera nécessaire de définir une fonction `Fahrenheit2Celcius(f)`.
Penser à échanger les mots Fahrenheit et Celcius dans la page. Pour cela, on pourra penser à utiliser des `<span>` comme vu dans le code initial.
De plus, on pourra définir une variable globale `convertisseur`, égale au départ à la fonction `Celcius2Fahrenheit`, qui est modifiée par la fonction appelée lors du clic sur le bouton.
5. (facultatif) On pourra faire en sorte que la valeur convertie soit arrondie à 2 chiffres après la virgule.

L'ensemble des exercices qui suivent est à faire en groupe.

Le travail sera à rendre et noté. Il nécessite d'utiliser un peu de HTML et de CSS.

Il est possible de se partager les exercices dans le groupe, sachant qu'ils sont *a priori* triés par ordre croissant de difficulté technique.

Il sera tenu compte du respect des instructions.

Tout élément d'amélioration technique supplémentaire apporté sera valorisé pour la note finale.

Il s'agit d'un travail sur JavaScript, donc un gros travail sur les fichiers de style CSS ne sera pas forcément pris en compte.

Exercice 5 (Conversion de monnaie)

Écrire un programme qui demande successivement de saisir un montant en euro puis une monnaie comme 'dollar', 'livre' ou 'yen'. Le montant est alors converti dans la monnaie choisie puis le résultat est affiché.

Exercice 6 (Plus ou moins)

Programmer un jeu qui consiste pour le joueur à trouver un nombre choisi au hasard par la machine entre 1 et 100. À chaque proposition du joueur, le programme répond par « plus grand » ou « plus petit » ou « gagné » selon sa réponse. On pourra ajouter une limite au nombre de tentatives du joueur (quelle serait une limite raisonnable?), et même permettre éventuellement de rejouer. Pour le choix du nombre au hasard, utiliser `Math.random()` et `Math.round()` ou `Math.floor()`.

Exercice 7 (Calcullette)

Pour cet exercice, récupérer le fichier compressé contenant les fichiers de base sur Moodle.

Le but ici est de créer une calculatrice basique.

1. Compléter le fichier `calcullette.html` pour qu'elle représente une calculatrice avec les boutons comme ci-dessous :

1.6666666666666667			
7	8	9	/
4	5	6	*
1	2	3	+
C	0	=	-

2. Ajouter une fonction de rappel pour l'événement `click` à chaque bouton, qui appelle l'une des fonctions suivantes du fichier `script.js` : `chiffre(n)` pour les chiffres, `operation(op)` pour les quatre opérations, `egale()` pour le bouton '=' et `effacer()` pour le bouton 'C'.
 3. Compléter le code JavaScript pour que la calcullette soit fonctionnelle.
- Lorsque l'on divise par 0, l'écran d'affichage doit contenir le message 'Erreur'.



l'opération « + » appliquée à des chaînes de caractères produit la concaténation. Il conviendra donc, avant d'effectuer les opérations, de transformer les valeurs lues avec la fonction `obtenirValeur()`, qui retourne une chaîne de caractères, en nombres avec la fonction `Number`.

Remarquer que la calculatrice ne permet d'entrer que des nombres entiers naturels.

Les détails de fonctionnement de la calculatrice ne sont pas donnés et il vous faudra y réfléchir et sans doute faire des choix.

Des propositions de variables globales sont données dans le fichier `script.js` mais, bien que leur utilisation est conseillée, elle n'est pas obligatoire.

En particulier, la variable `resultatEnCours` a pour but d'indiquer si la valeur affichée est un résultat donné par la calculatrice ou non, ce qui permet de savoir si, lorsque l'utilisateur clique sur un chiffre, il s'agit de commencer un nouveau nombre ou de continuer à en écrire un.

Les variables `resultatPrecedent` et `operationAEffectuer` ont pour but de se rappeler la valeur précédemment entrée (ou calculée) et la prochaine opération à effectuer (en attente de la deuxième valeur pour le calcul).

Il est tout à fait possible de définir d'autres variables, mais aussi d'autres fonctions que celles données au départ.

On pourra considérer ajouter des difficultés supplémentaires :

- Faire en sorte que la touche '**C**' ne supprime pas l'affichage mais efface seulement le dernier chiffre tapé.

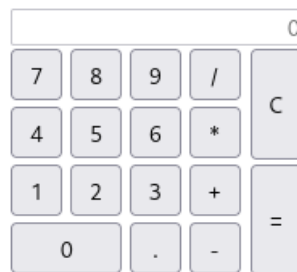
Quand l'utilisateur a déjà effacé tout le nombre (remplacé alors par le chiffre '**0**'), la touche '**C**' permet d'oublier le résultat précédent ;

- Permettre l'utilisation de la calculatrice en tapant au clavier et pas seulement en cliquant sur les boutons ;

Pour cela, il s'agit d'utiliser la méthode `addEventListener` permettant d'associer des événements à des fonctions.

Il sera nécessaire de faire des recherches pour savoir comment réaliser cela.

- Permettre de taper des nombres à virgule, avec une interface ressemblant par exemple à celle-ci :



Pour la taille des boutons, il s'agit entre autres par exemple d'ajouter une classe pour certains boutons et de compléter le fichier de style CSS pour modifier les longueurs ou largeur des boutons d'une classe donnée. Pour les boutons en hauteur, il pourra être nécessaire en plus de modifier l'attribut `position`.

- Toute autre amélioration est possible.

Si l'on veut admettre par exemple le signe '**-**', il sera conseillé d'ajouter un bouton distinct du bouton d'opération (comme sur les calculatrices TI).

Ne pas pousser trop loin cependant (par exemple avec des parenthèses ou des opérations trop compliquées), puisqu'il s'agit d'un travail à réaliser en peu de temps.