Concepteur Développeur en Informatique

Module 2 Développer pages web en lien avec base de données

Programmation client side	Séance	S02	Activité	A-001

Au cours de cette activité, vous allez découvrir la manipulation des éléments HTML à partir de l'API du DOM et les bases du langage javascript.

Sommaire de l'activité proposée :

1. Fonctions

Contrairement à C#, en javascript une fonction et ses surcharges sont définies en une seule version.

Une fonction javascript dispose d'une variable intrinsèque toujours définie **arguments** qui s'apparente à un tableau et contient les valeurs passées en argument à la fonction.

Regardez l'exemple du paragraphe 4.4 de votre support d'apprentissage A-découverte..Javascript.

Vous pouvez la programmer de plus belle manière en utilisant **arguments** (voir référence MDN arguments) et un groupe de choix multiples **switch** (voir mdn)

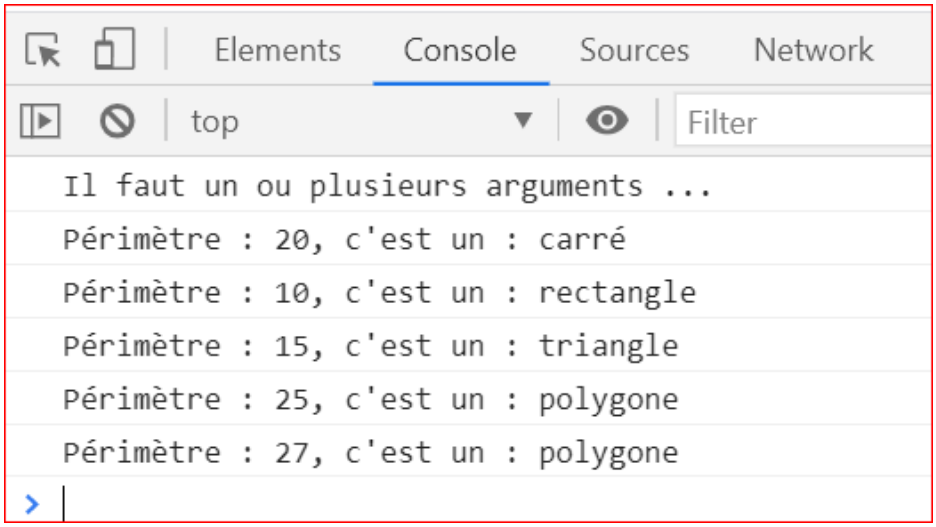
Modifiez la fonction du squelette proposé dans la page A001-1.html

Utilisez les outils du développeur et sa console javascript.

Testez les points d'arrêt et l'exploration des variables. Testez l'interaction avec la console.

```
// script principal
console.log(perimetre());
console.log(perimetre(5));
console.log(perimetre(3,2));
console.log(perimetre(5,5,5));
console.log(perimetre(5,5,5,10));
console.log(perimetre(10,5,4,3,5));
```

Vous devez obtenir le résultat suivant :



Plus :
Vous pouvez ajouter un mécanisme qui permet de s'assurer que toute valeur passée en argument est numérique pour éviter les scratchs.

Pour le test des scripts voir le chapitre 3 du support d'apprentissage sur JavaScript

Article présentant les outils de debug :
https://blog.arcoptimizer.com/10-fonctionnalites-de-loutil-de-developpement-chrome-que-vous-avez-peut-etre-manquees

2. Sélectionner les éléments

Voici la capture de la page affichée dans ce premier atelier.

Travail à réaliser à partir de la page html A001-2.html.

Vous devez respecter, dans tous les exercices, la séparation du code HTML des Scripts et des feuilles de styles.
Je vous communique les éléments de syntaxe. Utiliser le support de cours et/ou mdn (plus à jour..) pour vos aider.

Associer un gestionnaire à l'événement click du bouton :

- AddEvenListener

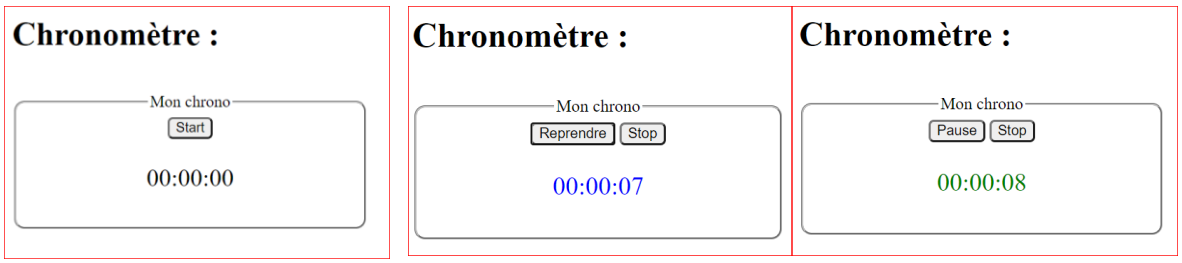
Déterminer quel est le bouton radio sélectionné.

- Acquérir une référence à l'objet groupe de boutons radio **getElementByName**
- Parcourir le tableau des éléments avec une boucle **for** classique
- **Ou** Vous pouvez aussi parcourir le tableau et utiliser un cycle **for** … **of**
- **Ou** Vous pouvez utiliser aussi un cycle **forEach** (approche fonctionnelle) plus difficile à mettre en place
- Afficher la valeur du bouton radio sélectionné **checked** et assigner celle-ci à la propriété **value** de la boite de texte dont vous aurez récupérée la référence avec **getElementById**

3. Evénements et modification dynamique de styles

Réalisez un petit chrono qui affiche un champ texte () et 3 boutons Start, Pause et Stop.

- L'appui sur le bouton Start démarre le chrono.
- L'appui sur le bouton Pause arrête puis redémarre le chrono.
- L'appui sur le bouton Stop arrête et remet à zéro le chrono.



Vous avez à votre disposition dans le dossier A000-3:

- la page HTML
- la feuille de style
- le script vierge

Le code JavaScript sera écrit de manière non intrusive.

Il s'agit de développer l'application pas-à-pas en testant à chaque étape.

Variables globales nécessaires :

- Références des objets JavaScript correspondant aux 3 boutons et au *span*
- 1 variable pour référencer un timer JavaScript
- 1 variable pour compter le nombre de secondes écoulées (initialisée à 0)

Une première **fonction auto-exécutable** permet d'ajouter une propriété '`paramTps`' aux objets boutons *Start* et *Pause*, et de capturer l'événement *clic* sur le bouton *Start* ; la fonction associée ('`startChrono`') à cet événement réalise les traitements :

- Désactiver la capture de l'événement *clic* sur ce bouton *Start*
- Masquer ce bouton *Start* et afficher les 2 autres boutons (par modification des classes de styles CSS associées aux objets *button*)
- Activer un *timer* JavaScript associé à une fonction anonyme qui calcule le temps écoulé et le convertit en nombre d'heures, minutes et secondes (voir code fourni)

NB : pour ajouter une propriété à un objet JavaScript, il suffit de l'initialiser :

```
// ajoute une propriété à l'objet button
btnStart.paramTps = tpsEcoule;
```

Je vous donne la fonction de calcul du temps écoulé :

Vous remarquerez :
- l'accès à l'événement et à sa propriété target.
- la fonction setInterval programmée avec une fonction de rappel anonyme est exécuté chaque seconde (1000 millisecondes)
- Que la fonction ajouteUnZeo n'est pas programmée…

```
// calcul de nombre heures, minutes et secondes écoulées
var startTime = new Date();
decompte = setInterval(function() {

      // 1- Convertir en secondes :
      var seconds = Math.round(
            (new Date().getTime() - startTime.getTime()) / 1000
            + e.target.paramTps); // e représente l'event déclencheur
            // e.target représente l'objet déclencheur
            // ici : bouton start ou bouton pause
            // (cette propriété a été ajoutée aux boutons)

      // 2- Extraire les heures :
      var hours = parseInt( seconds / 3600 );
      seconds = seconds % 3600; // secondes restantes
      // 3- Extraire les minutes:
      var minutes = parseInt( seconds / 60 );
      seconds = seconds % 60; // secondes restantes

      // 4- afficher dans le span
      chronoP.innerHTML = ajouteUnZero(hours)
            +":"+ajouteUnZero(minutes)
            +":"+ajouteUnZero(seconds);
      // 5- incrémenter le nombre de secondes
      tpsEcoule += 1;

}, 1000); // fin de fonction anonyme dans setInterval()
```

La petite fonction `ajouteUnZero(temps)` reste à écrire ; elle permet simplement d'afficher systématiquement chaque nombre sur 2 chiffres ('`04`' et non '`4`') par simple concaténation du « `0` » manquant.

Testez à l'aide du débogueur et mettez au point jusqu'à ce que cette première étape se passe correctement (les boutons *Pause* et *Stop* restent inactifs).

Activez la capture des événements *clic* sur chacun des boutons *Pause* et *Stop* ; associez-les aux fonctions '`pauseChrono`' et '`stopChrono`' et écrivez déjà les structures de ces fonctions.

Ecrivez le contenu de la fonction '`pauseChrono`' de manière à :

- Désactiver le *timer*
- Modifier la fonction associée à l'événement *clic* du bouton *Pause* (suppression puis ajout) de manière à ce qu'il puisse relancer le décompte à l'aide de la fonction '`startChrono`' quand l'utilisateur cliquera à nouveau sur ce bouton
- Mémoriser le temps écoulé dans la propriété `paramTps` du bouton *Pause*

Testez la mise en pause et la reprise.

Ecrivez le contenu de la fonction '`stopChrono`' de manière à :

- Désactiver le *timer*
- Désactiver la capture des événements *clic* des boutons *Pause* et *Stop*
- Réactiver la capture de l'événement *clic* du bouton *Start*
- Réinitialiser le temps écoulé et l'affichage en *span*
- Masquer les boutons inutiles

Testez l'application complète.