APPROFONDISSEMENT

1. MANIPULATION DES VARIABLES EN JAVASCRIPT (A PARTIR DE 2015 -> ES6)

1.1. Déclarer une constante "data" qui contient le nombre 3

1.2. Ecrire des données correspondant aux types définis ci-dessous :

// String (texte)
let str = ……..

// Number (nombres)
let nb = ……..

// Boolean (booléen)
let bool = ……..

1.3. Déclarer trois variables sans valeur(undefined) sur une seule ligne.

2. OPERATEURS ET FONCTIONS

2.1. Écrire une fonction en javascript qui retourne un nombre multiplié par deux en utilisant un paramètre.

2.2. Déterminer ce qui est vrai ou faux dans les extraits de code javascript suivants :

let solde = 150 000;
solde += 50 000;
// Ce code permet d'assigner la valeur 50 000 à solde ?

let resultat = 10 % 7;
// resultat est égal à 5 ?

let str = `J'ai besoin de $(nbPneus) pour ma voiture.`
// Est-ce la bonne façon pour intégrer une expression dans une chaîne ?

3. TABLEAUX ET OBJETS

3.1. Créer un tableau avec trois valeurs à l'intérieur : 5124, true, et un objet qui contient une propriété.

3.2. Créer un objet avec trois propriétés

// Object (object)
let myObj = ……

4. UN PEU D'ALGORITHME

4.1. Analyser le script suivant permettant de « logger » dans la console le résultat de l'expression d'une opération ternaire :

```
10  let metreCarre=20;
11  console.log(metreCarre > 70 ? "Grand logement" : "Petit ou Moyen logement");
```

4.2. Créer une boucle **for** (structure itérative) qui affiche dans la console une suite de nombres de **100 à 0** en pas de 1.

Exemple : 100,99,98,97,96 ...jusqu'a 0

4.3. Afficher à l'aide d'une boucle "**for…in**" toutes les valeurs des propriétés de l'objet défini ci-dessous dans la console.

```
const capitales = {
  Maurice : "Port Louis",
  Madagascar : "Antananarivo",
  Comores : "Moroni",
  Australie : "Canberra",
  Seychelles : "Victoria",
  Mozambique : "Maputo",
}
```

- Documentation instruction **for …in** : https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Instructions/for...in

4.4. Afficher à l'aide de la console avec une boucle "**for…of**" les éléments du tableau suivant :

const fruits = ['Banane', 'Papaye', 'Mangue', 'longani', 'Goyave', 'Orange', 'Citron', 'Ananas'];

- Documentation instruction **for…of** : https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Instructions/for...of

4.5. Créer une fonction qui sert à repérer si un texte qu'on lui passe en argument contient la lettre "z". Si c'est le cas, la fonction écrit dans la console "Alerte : le texte contient la lettre z".

5. LES EVENEMENTS

On souhaite gérer un évènement en utilisant la méthode **addEventListener()** → Ecoute d'évènements :

- Exemple de script :

```
// Utilisation de la méthode addEvenListener()

const btn = document.querySelector('button');

btn.addEventListener('click', function(){
    console.log("Hello world");
})
```

La **cible** de l'évènement est un **bouton** ici et une fonction dite « **anonyme** » est appelée (fonction qui n'a pas de nom).

Ne pas oublier la balise <button> dans le fichier HTML.

- Script pour supprimer une écoute d'évènement :

```
//Utilisation d'une fonction nommée

btn.addEventListener('click', foo);

function foo(){
    console.log("Hello world");
}

btn.removeEventListener('click', foo)
```

- Script pour utiliser l'évènement une seule fois :

```
//Utilisation d'une fonction nommée

btn.addEventListener('click', foo);

function foo(){
    console.log("Hello world");
    btn.removeEventListener('click', foo);
}
```

- Script pour déclencher un évènement lors de l'appui sur une touche :

```
// Utilisation de la méthode addEvenListener()

const btn = document.querySelector('button');


document.addEventListener('keydown', function(){
    console.log("Hello world");
})
```

- Script pour avoir des informations sur l'évènement avec le paramètre « objet d'évènement **e** » pour la fonction :

```
// Utilisation de la méthode addEvenListener()

const btn = document.querySelector('button');


document.addEventListener('keydown', function(e){
    console.log(e);
})
```

- Résultat obtenu pour l'objet d'évènement dans la console javascript du navigateur :

```
▼KeyboardEvent {isTrusted: true, key: "a", code: "KeyQ", location: 0, ctrlKey: false, …}
   altKey: false
   bubbles: true
   cancelBubble: false
   cancelable: true
   charCode: 0
   code: "KeyQ"
   composed: true
   ctrlKey: false
   currentTarget: null
   defaultPrevented: false
   detail: 0
   eventPhase: 0
   isComposing: false
   isTrusted: true
   key: "a"
   keyCode: 65
   location: 0
```

- Script pour afficher directement dans la console la touche du clavier qui a été appuyée :

```
document.addEventListener('keydown', function(e){
    console.log(e.key);
})
```

- Script pour récupérer des données dans un champ input :

<input type="text"> dans le fichier HTML

```
const inp = document.querySelector('input');

inp.addEventListener('input', function(){
    console.log("Je suis en train d'écrire dans le champ input");
})
```

- Récupération des données saisies dans la console :

```
inp.addEventListener('input', function(e){
    console.log(e);
    //console.log(e.data);

})
```

⇒ Liste des événements : https://developer.mozilla.org/fr/docs/Web/Events

Travail demandé : On met maintenant à disposition les codes HTML et CSS d'une page web pour laquelle il sera demandé de faire des modifications d'aspect avec une gestion d'évènements en javascript.

Code HTML :

index.html > ...

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Activité Evénements - Avancé</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="cercle"></div>
    <div class="carre"></div>
    <script src="script.js"></script>
</body>
</html>
```

Fichier de style CSS

style.css

```css
.cercle {
    width: 250px;
    height: 250px;
    border-radius: 50%;
    background: lightgreen;
    transition: background-color 0.6s ease;
}

.carre {
    margin-top: 10px;
    width: 250px;
    height: 250px;
    background: lightblue;
    transition: background-color 0.6s ease;
}
```

5.1. Prendre connaissance du fichier HTML et du fichier de style. Expliquer en quelques lignes ce qui a été réalisé.

5.2. Faire la saisie du fichier HTML et du fichier de style.

5.3. Mettre en place un évènement dans un fichier de script « javascript » : changer la couleur du cercle en "**crimson**" quand votre **souris le survole**. On demande d'utiliser les méthodes :

- querySelector()
- addEventListener()

⇒ Couleur **crimson** (cramoisi)

Quelques exemples de codes couleurs Web de la même gamme que la **couleur crimson** :

Nom	code Hexa.	code RGB	code HSL
crimson	#DC143C	rgb(220,20,60)	hsl(348,83%,47%)
lightsalmon	#FFA07A	rgb(255,160,122)	hsl(17,100%,74%)
salmon	#FA8072	rgb(250,128,114)	hsl(6,93%,71%)
darksalmon	#E9967A	rgb(233,150,122)	hsl(15,72%,70%)
lightcoral	#F08080	rgb(240,128,128)	hsl(0,79%,72%)
indianred	#CD5C5C	rgb(205,92,92)	hsl(0,53%,58%)
firebrick	#B22222	rgb(178,34,34)	hsl(0,68%,42%)
darkred	#8B0000	rgb(139,0,0)	hsl(0,100%,27%)
red	#FF0000	rgb(255,0,0)	hsl(0,100%,50%)

Source : https://web-color.aliasdmc.fr/couleur-web-crimson-rgb-hsl-hexa.html

5.4. Mettre en place un deuxième évènement : changer la couleur du carré en "**mediumordchid**" quand **vous cliquez dessus**.

Nom	code Hexa.	code RGB	code HSL
lightsalmon	#FFA07A	rgb(255,160,122)	hsl(17,100%,74%)
papayawhip	#FFEFD5	rgb(255,239,213)	hsl(37,100%,92%)
slateblue	#6A5ACD	rgb(106,90,205)	hsl(248,53%,58%)
goldenrod	#DAA520	rgb(218,165,32)	hsl(43,74%,49%)
seashell	#FFF5EE	rgb(255,245,238)	hsl(25,100%,97%)
dimgray dimgrey	#696969	rgb(105,105,105)	hsl(0,0%,41%)
mediumorchid	#BA55D3	rgb(186,85,211)	hsl(288,59%,58%)
tomato	#FF6347	rgb(255,99,71)	hsl(9,100%,64%)
lightpink	#FFB6C1	rgb(255,182,193)	hsl(351,100%,86%)

5.5. Rajouter l'évènement "**click**" au document, puis loggez la position(x,y) des clics que vous effectuez sur le document.

6. JAVASCRIPT ASYNCHRONE

Que signifie le terme « **Asynchrone »** ici ? ⇒ On peut dire que les actions sont différées dans le temps

Nous allons nous intéresser aux méthodes **setTimeout()** et **setInterval()**

La méthode javascript **setTimeout()** va permettre d'exécuter une fonction (« quelque chose ») au bout d'un temps imparti :

Cette méthode prend 2 paramètres : la fonction à exécuter et la durée (en ms) que l'on va attendre avant de l'exécuter.

- Exemple de script :

```
//Méthode setTimeout()

setTimeout(presentation, 2000);

function presentation(){
    console.log("Hello world");
}
```

- Exemple de script pour comprendre la notion d'asynchrone :

```
//Méthode setTimeout()

setTimeout(presentation, 2000);

console.log("Hello 1");

function presentation(){
    console.log("Hello world");
}

console.log("Hello 2");
```

La fonction va s'exécuter de façon différer dans le tempsl

- Exemple de script pour empêcher l'exécution :

```
//Méthode setTimeout()

const timeout = setTimeout(presentation, 2000);

console.log("Hello");

clearTimeout(timeout);

function presentation(){
    console.log("Hello world");
}
```

- Autre façon de procéder :

```
//Méthode setTimeout()
//Faire un clear

console.log("Hello");

clearTimeout(setTimeout(presentation, 2000));

function presentation(){
    console.log("Hello world");
}
```

La méthode **setInterval()** va se lancer toutes les intervalles de temps passés en paramètre. Répétition de l'intervalle de temps passé en paramètre de cette méthode.

- Exemple de script :

```
let compteur = 0;

setInterval(incr,1000);

function incr(){
    compteur++;
    console.log(compteur);
}
```

- On peut arrêter la répétition avec le script suivant :

```
let compteur = 0;

clearInterval(setInterval(incr,1000));

function incr(){
    compteur++;
    console.log(compteur);
}
```

Travail demandé : Réaliser maintenant l'évolution de l'activité 3 du TP1 Javascript. On demande en effet de faire apparaitre un décompte (sous une forme libre) de 5 secondes entre l'appui sur le bouton d'autodestruction et une indication signifiant la destruction.