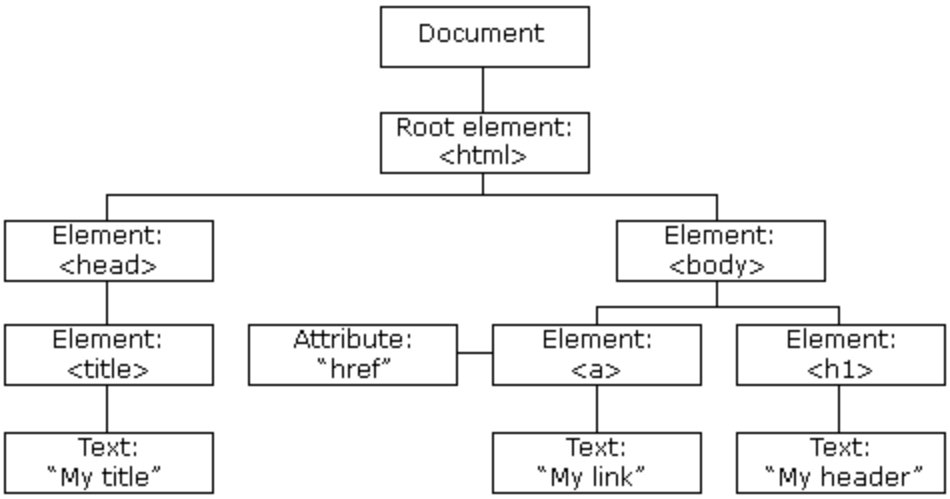
TP1 JavaScript / HTML5 /DOM

JavaScript est un langage de programmation web interprété, sui s'exécute coté client (par le navigateur web). Mais, JavaScript peut être exécuté coté serveur à travers le Framework Node.JS.
Lorsqu'une page est chargée, le navigateur crée une arborescence d'objets (**Document Object Model**, abrégé **DOM**) qui décrit la page web. Le DOM est une interface de programmation (API) pour les documents XML et HTML. Grace à JavaScript, le DOM définit :

- Les éléments HTML comme des objets
- Les propriétés de tous les éléments HTML : Les propriétés HTML DOM sont des valeurs des éléments HTML qu'on peut définir ou modifier
- Les méthodes pour accéder à tous les éléments HTML : Les méthodes HTML DOM sont les actions qu'on peut effectuer sur les éléments HTML (modifier ajouter ou supprimer des éléments)
- Les événements pour tous les éléments HTML

Par exemple, on peut afficher ou masquer un , en ajouter, en déplacer ou même en supprimer.

Le DOM traduit une page Web sous forme d'une arborescence d'objets (voir la figure ci-dessous). Dans la structure de DOM, l'objet *window* représente la fenêtre du navigateur. C'est à partir de cet objet que le code JavaScript est exécuté. Comme racine, on trouve l'objet document ; c'est un sous objet de *window*. Il représente la page Web et plus percement la balise <html>.



Méthodes JavaScript du DOM :

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Method	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element

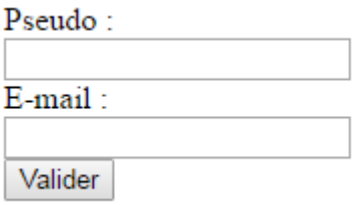
Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*element*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Ce TP s'intéresse aux éléments suivants :

- HTML5
- DOM et JavaScript (Forms API et évènements)
- Fonctions JavaScript (fonctions imbriquées et anonymes, le pattern callback, et la fermeture ou *closure*)
- JavaScript moderne avec les nouveautés ES6 : Fonctions fléchées (*Arrow functions*)

Exercice 1 :

1. Créer le formulaire suivant :



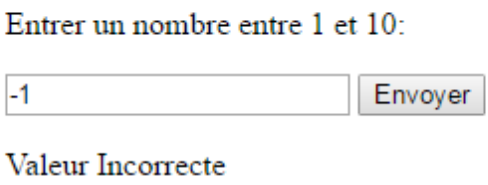
2. Fonctions JavaScript à créer :
   - Une fonction pour vérifier la longueur du pseudo (entre 2 et 30 caractères) ;
   - Une fonction pour vérifier l'adresse e-mail (en utilisant de préférence une **regex**): L'expression régulière que nous allons utiliser est la suivante :
     /^[a-zA-Z0-9._-]+@[a-z0-9._-]{2,}\.[a-z]{2,4}$/
   - Une fonction globale qui vérifie tout (en faisant appel aux fonctions précédentes).

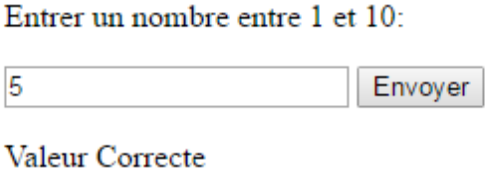
NB. Les deux premières fonctions doivent, en plus de colorer le champ, renvoyer *true* si c'est bon, et false si ça ne l'est pas.

3. Refaire tout le travail demandé avec HTML5.

Exercice2 :

1. Créer le formulaire suivant :

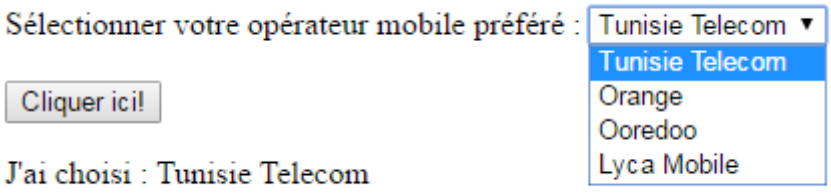




2. Implémenter une Fonction JavaScript permettant de vérifier que le nombre saisi est compris entre 1 et 10.
3. Ré-implémenter le formulaire de la question 1) avec HTML5. Prévoir les éventuelles modifications sur la fonction JavaScript de la question 2).

Exercice 3 :

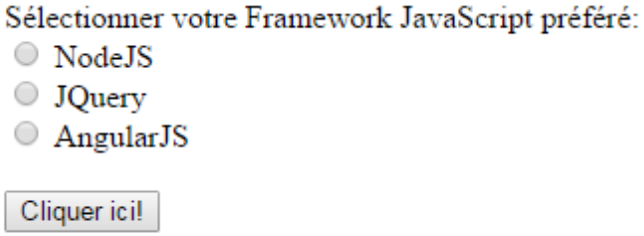
1. Créer le formulaire suivant :



2. Implémenter une Fonction JavaScript permettant d'afficher le choix de l'utilisateur.

Exercice 4:

1. Créer le formulaire suivant :



2. Implémenter une Fonction JavaScript permettant de vérifier le choix de l'utilisateur et de l'afficher à l'aide d'un message d'alerte.

Exercice 5 (Fonctions Imbriquées, Anonymes et Callback et principe de Closure)

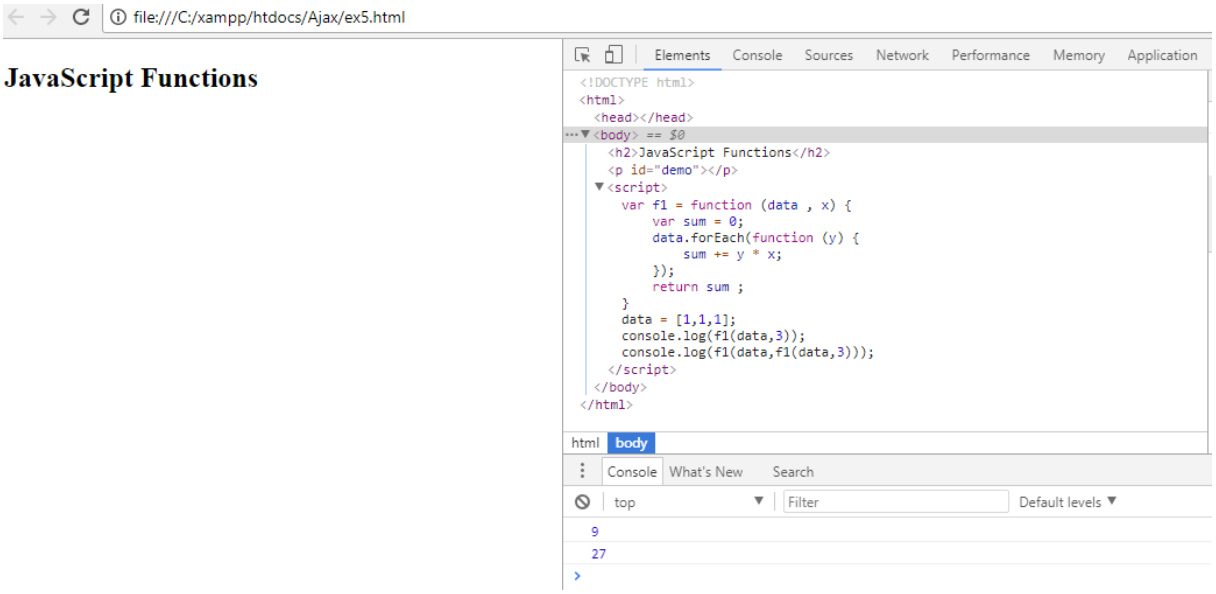
1. Le code JavaScript suivant permet de déclarer une fonction anonyme. La fonction forEach () définit une fonction anonyme comme callback. Taper ce script dans le fichier *ex5.html.*

```
var f1 = function (data , x) {
    var sum = 0;
    data.forEach(function (y) {
        sum += y * x;
    });
    return sum ;
}
data = [1,1,1];
console.log(f1(data,3));
console.log(f1(data,f1(data,3)));
```

Commentaires :

Une fonction de rappel ou bien de retour (callback) est une fonction comme les autres. Sa particularité est qu'elle est appelée par une autre qui l'a reçu en tant que paramètre.
Autrement dit, un callback est une fonction qui est passée à une autre fonction.

Dans le cas des fonctions imbriquées, la *closure* (fermeture) permet de définir des variables et des fonctions au sein de fonctions dont la portée s'applique à tous les enfants.
Le paramètre *x* et la variable *sum* sont visibles dans la fonction définie comme callback de forEach ().



2. Tester le script le fichier *ex5.html*.

Exercice 6 (Fonctions fléchées)

1. Le code JavaScript suivant permet de déclarer une fonction anonyme fléchée. La fonction forEach () définit une fonction fléchée comme callback implicite. Taper ce script dans le fichier *ex6.html*.

```
var f2 = function (data , x) {
    var sum = 0;
    data.forEach((y) => {
        sum += y * x;
    });
    return sum ;
}
data = [1,1,1];
console.log(f2(data,3));
console.log(f2(data,f2(data,3)));
```

Commentaires :

(y) est le seul paramètre de la fonction fléchée.
Dans le cas de la variable *this* une fonction fléchée ne crée pas de nouveau contexte mais elles capturent la valeur *this* de leur contexte (à étudier dans Angular plus tard).

2. Tester le fichier *ex6.html*.