JAVASCRIPT : TP1

Durée prévue : 3h.
Problématique : réalisation de pages web

Plan de l'étude	Remarque
Introduction	
I. Découverte des structures, variables et de quelques fonctions	
II. Des exemples d'utilisation dans une page web	
III. D'autres fonctions à travers des exemples	

Introduction

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives. C'est un langage de type « orienté objet ».

Et quand on dit que le JavaScript est un langage de scripts, cela signifie qu'il s'agit d'un langage interprété (et non compilé)! Il est donc nécessaire de posséder un interpréteur pour faire fonctionner du code JavaScript. Cet interpréteur est inclus dans votre navigateur Web.

Le JavaScript est un langage dit client-side (« côté client » en français), c'est-à-dire que les scripts sont exécutés par le navigateur chez l'internaute (le client). Cela diffère des langages de scripts dits server-side (« côté serveur ») qui sont exécutés par le serveur Web. C'est le cas de langages comme le PHP.

Dans ce TP, nous n'allons pas réellement apprendre à programmer en Javascript mais apprendre à comprendre la structure de petits programmes afin de pouvoir mettre en œuvre des programmes ou fonctions javascript créés par d'autres.

Pour toute recherche d'infos sur javascript (règles, fonctions, syntaxe, …) :
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference

I. Découverte des structures , variables et de quelques fonctions

1. La balise « script »

Pour dire au code HTML5 qu'il va y avoir du javascript on utilise la balise <script>

Voici un premier code qui va utiliser la fonction **alert()**:

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>premier programme!</title>
</head>

<body>

   <script>
      // Ceci est ma première instruction
      alert('utilisation d une première fonction javascript!');
   </script>

</body>
</html>
```

a) Essayez ce premier code (copiez-le dans Notepad++, enregistrez-le sous le nom 'essai_js_1.html' et ouvrez-le avec le navigateur Mozilla Firefox). Expliquez ce que fait ce code.

b) Surlignez sur le code précédent, de couleurs différentes, les parties en HTML et les parties en javascript

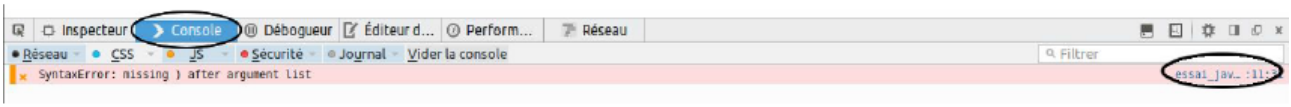
c) Par quel élément de ponctuation la fonction Javascript du code est-elle terminée ?
d) Comment écrit-on les commentaires dans notre script ?
e) Essayez Maintenant avec : alert('utilisation d'une première fonction javascript!'). Que se passe-t-il ? Où est l'erreur ?

Comment déboguer Javascript ?

Tous les navigateurs internet possèdent un outil permettant de chercher les erreurs (bugs) des codes javascript.

Les bugs syntaxiques sont les plus simples à résoudre, car l'interpréteur JavaScript vous signalera généralement l'endroit où l'erreur est apparue mais la description de l'erreur n'est pas toujours juste !

Dans Mozilla Firefox : allez dans le menu « outils » puis « développement web » puis « outils de développement ». Une fenêtre s'ouvre en bas de votre navigateur (et si vous relancez la page un code erreur doit apparaître dans l'onglet « console ») :



Si vous cliquez à droite, une fenêtre s'ouvre avec la ligne où se situe l'erreur de syntaxe.

2. Les variables

Déclarer une variable

Déclarer une variable consiste à lui réserver un espace de stockage en mémoire. Une fois la variable déclarée, vous pouvez commencer à y stocker des données.

Pour déclarer une variable, il vous faut d'abord lui trouver un nom. Il est important de préciser que le nom d'une variable ne peut contenir que des caractères alphanumériques, autrement dit les lettres de A à Z et les chiffres de 0 à 9 ; l'underscore (_) et le dollar ($) sont aussi acceptés. Le nom de la variable ne peut pas commencer par un chiffre et ne peut pas être constitué uniquement de mots-clés utilisés par le JavaScript. Par exemple, vous ne pouvez pas créer une variable nommée var car vous allez constater que ce mot-clé est déjà utilisé, en revanche vous pouvez créer une variable nommée var_.

Pour déclarer une variable, il vous suffit d'écrire la ligne suivante :

```
var ma_variable;
```

Le JavaScript étant un langage sensible à la casse, faites bien attention à ne pas vous tromper sur les majuscules et minuscules utilisées (ma_variable ≠ ma_Variable) .

- Pour affecter une valeur à la variable (par exemple le chiffre 2) c'est simple :

```
ma  variable=2 ;
```

- On peur aussi le simplifier en une seule ligne de code (déclaration et affectation) :
  var ma_variable=2 ;

- De même, vous pouvez déclarer et assigner plusieurs variables sur une seule et même ligne :
  var ma_variable1, ma_variable2 = 4, toto;

Vous remarquerez qu'il n'y a qu'un seul « var » et que les variables sont alors séparées par des virgules

Le JavaScript est un langage typé dynamiquement. Cela veut dire, généralement, que toute déclaration de variable se fait avec le mot-clé « var » sans distinction du contenu (entier, texte, ...). Ainsi on peut y mettre du texte en premier lieu puis l'effacer et y mettre un nombre quel qu'il soit, et ce, sans contraintes.

Quelques exemples de déclaration de variables sous forme de chaînes de caractères :
var text1 = "Mon premier texte"; // Avec des guillemets
var text2 = 'Mon deuxième texte'; // Avec des apostrophes

Un premier exemple de programme :

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>premier programme!</title>
</head>

<body>

   <script>
      var age=18, age2=21, age_total;
         age_total = age + age2;
         var message = 'l\'age total est: ' + age_total;
         alert(message);
   </script>

</body>
</html>
```

a) Essayez ce code. Expliquez, en détail, ce qu'il fait et comment il le fait.

Un deuxième exemple de programme :

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>premier programme!</title>
</head>

<body>
```

```
    <script>
        var age='18', age2='21', age_total;
           age_total = age + age2;
           var message = 'l\'age total est: ' + age_total;
        alert(message);
    </script>

</body>
</html>
```

b) Essayez ce nouveau code. Expliquez, en détail, ce qu'il fait et comment il le fait (différence avec l précédent) Quel est le problème de ce deuxième code ?

c) Essayez de résoudre le problème (on ne change pas le type de variables) en utilisant la fonctio « parseInt() » (les infos pour utiliser la fonction "parseInt()" sont ici https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/)

3. Les structures de tests

✗ les structures conditionnelles : la structure **if**

Soit le code suivant :

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>premier programme!</title>
</head>

<body>

   <script>
        if (2 < 8 && 8 >= 4) {
          alert('La condition est bien vérifiée.');
        }

        if (2 > 8 || 8 <= 4) {
           alert("La condition n'est pas vérifiée mais vous ne le saurez pas vu que ce code ne s'exécute pas.");
        }
   </script>

</body>
</html>
```

Complément : les opérateurs logiques sont: ET(&&) , OU (||) et PAS(!)

a) Essayez ce code. Expliquez, en détail, ce qu'il fait.

✗ les structures conditionnelles : la structure **if … else**

Dans l'exemple de code suivant, nous allons utiliser la fonction « confirm » qui demande de répondre à une question dont le résultat est oui ou non (en « booléen javascript » cela signifie que la fonction renvoie un entier signifiant « true » ou « false »).

Pour bien comprendre « true » et « false », essayez le programme suivant :

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>premier programme!</title>
</head>

<body>

   <script>
            if (true) {
                  alert('condition vérifiée');
            }

            if (false) {
                  alert('condition non vérifiée');
            }

   </script>

</body>
</html>
```

b) Essayez ce code. Expliquez, en détail, ce qu'il fait.

remplacez maintenant dans le code précédent « if (false) » par « if (!false) »

c) Essayez et expliquez ce qui se passe maintenant.

Soit le code suivant :

```
<!DOCTYPE html>
<html>
```

```
<head>
   <meta charset="utf-8" />
   <title>premier programme!</title>
</head>

<body>

   <script>
            if (confirm('Pour accéder à ce site vous devez avoir 18 ans ou plus, cliquez sur "OK" si c\'est le
cas.')) {
                  alert('Vous allez être redirigé vers le site.');
            }

            else {
                  alert("Désolé, vous n'avez pas accès à ce site.");
            }
   </script>

</body>
</html>
```

d) Essayez ce code. Expliquez, en détail, ce qu'il fait.

✗ les structures conditionnelles : la structure **if** … **else if** pour dire « sinon si »

Complément : la fonction prompt() pose une question et renvoie la réponse.

Soit le code suivant :

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>premier programme!</title>
</head>

<body>

   <script>
         var etage = parseInt(prompt("Entrez l'étage où l'ascenseur doit se rendre (de -2 à 30) :"));

         if (etage == 0) {

               alert('Vous vous trouvez déjà au rez-de-chaussée.');

         } else if (-2 <= etage && etage <= 30) {

               alert("Direction l'étage n°" + etage + ' !');
```

```
            } else {
                    alert("L'étage spécifié n'existe pas.");
                    }

    </script>

</body>
</html>
```

e) Essayez ce code. Expliquez, en détail, ce qu'il fait.
f) Essayez de rentrer une lettre au lieu d'un chiffre. Le programme plante-t-il ?

Remarque : il existe aussi la structure Switch (comme en PHP)

4. Les structures de boucles

✗ La boucle While (répéter Tant que)

Soit le code suivant :

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>premier programme!</title>
</head>

<body>

   <script>
        var number = 1;

        while (number < 10) {
           number++;
        }

        alert(number);
   </script>

</body>
</html>
```

a) Essayez ce code. Expliquez, en détail, ce qu'il fait.

Soit le code suivant :

```
<!DOCTYPE html>
</head>

<body>

   <script>
        for (var iter = 0; iter < 5; iter++) {
          alert('Itération n°' + iter);
        }
   </script>

</body>
</html>
```

c) Essayez ce code. Expliquez, en détail, ce qu'il fait.

Remarque : il existe aussi la boucle DoWhile

5. Les fonctions

Dans les chapitres précédents vous avez découvert quelques fonctions de javascript: alert(), prompt(), confirm() et parseInt(). En les utilisant, vous avez pu constater que chacune de ces fonctions avait pour but de mener à bien une action précise, reconnaissable par un nom explicite.
Mais on peut aussi créer nos propres fonctions :

Structure :

```
function nom_de_la_fonction(arguments) {
    // Le code que la fonction va devoir exécuter
}
```

Soit le code suivant :

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>premier programme!</title>
</head>

<body>

   <script>
        var res;
```

```
        var nb = parseInt (prompt('Entrez un nombre:'));

        function div_par_2(valeur) {
                var resultat;
                resultat=valeur/2;
                return resultat;
        }


        if (nb){
                res = div_par_2(nb);
                alert('le résultat est: ' + res);
        }
        else {
                alert('il faut rentrer un nombre!');
        }

    </script>

</body>
</html>
```

a) Essayez ce code. Expliquez, en détail, ce qu'il fait.
b) Quelle est la fonction créée dans ce code ?
c) Quels arguments reçoit-elle ?
d) Que retourne-t-elle ?

Variable locales/ variables globales :

Une variable déclarée dans une fonction est une variable locale (elle n'existe que dans la fonction). A l'inverse, une variable déclarée hors fonction est une variable globale. Il faut toujours bien faire attention à l'emplacement de la déclaration des variables (on les crée là où on en a besoin).

e) Dans le code précédent, quels sont les variables globales et les variables locales ?

6. Le javascript externe

Il est possible, et même conseillé, d'écrire le code JavaScript dans un fichier externe, portant l'extension ".js". Ce fichier est ensuite appelé depuis la page Web au moyen de l'élément <script> et de son attribut src qui contient l'URL du fichier .js.

Le code précédent devient :

le fichier « essai_js_1.html » :

```
<head>
   <meta charset="utf-8" />
   <title>premier programme!</title>
</head>

<body>

   <script src="mon_java.js">
  </script>

</body>
</html>
```

et le fichier « mon_java.js » : (dans le même répertoire ici)

```
var res;
var nb = parseInt (prompt('Entrez un nombre:'));

function div_par_2(valeur) {
        var resultat;
        resultat=valeur/2;
        return resultat;
}


if (nb){
        res = div_par_2(nb);
        alert('le résultat est: ' + res);
}
else {
        alert('il faut rentrer un nombre!');
}
```

Remarque : Positionner l'élément <script> :
La plupart des cours de JavaScript, et des exemples donnés un peu partout, montrent qu'il faut placer l'élément <script> au sein de l'élément <head> quand on l'utilise pour charger un fichier JavaScript. C'est correct, oui, mais il y a mieux !

Une page Web est lue par le navigateur de façon linéaire, c'est-à-dire qu'il lit d'abord le <head>, puis les éléments de <body> les uns à la suite des autres. Si vous appelez un fichier JavaScript dès le début du chargement de la page, le navigateur va donc charger ce fichier, et si ce dernier est volumineux, le chargement de la page s'en trouvera ralenti. C'est normal puisque le navigateur va charger le fichier avant de commencer à afficher le contenu de la page.

Pour pallier ce problème, il est conseillé de placer les éléments <script> juste avant la fermeture de l'élément <body>, comme ceci :

```
<!DOCTYPE html>
```

```
<html>
<head>
   <title>mon site!</title>
</head>

<body>
   <p>
   <!--
      Contenu de la page Web
   -->
   </p>

   <script>
      // Un peu de code JavaScript si besoin ...
   </script>

   <script src="hello.js"></script>

</body>
</html>
```

Il est à noter que certains navigateurs modernes chargent automatiquement les fichiers JavaScript en dernier, mais ce n'est pas toujours le cas. C'est pour cela qu'il vaut mieux s'en tenir à cette méthode.

II. Des exemples d'utilisation dans une page web

On va maintenant rentrer dans le vif du sujet, notamment la partie "objet" de javascript. Ce cours n'a pas la vocation de vous apprendre à programmer en javascript mais plutôt à intégrer des éléments javascript (en général créés par d'autre). Pour le faire il va quand même falloir comprendre comment ça fonctionne …. et ce n'est pas forcément facile !

1. Structure d'une page web du côté javascript

L'objet window

Avant de véritablement parler du document représentant notre page web, nous allons parler de l'objet window. L'objet window est ce qu'on appelle un objet global qui représente la fenêtre du navigateur. C'est à partir de cet objet que le JavaScript est exécuté.

Contrairement à ce qui a été dit jusqu'à maintenant, alert() n'est pas vraiment une fonction. Il s'agit en réalité d'une méthode (c'est comme ça que ça s'appelle dans un langage de programmation orienté "objet") appartenant à l'objet window. Mais l'objet window est dit implicite, c'est-à-dire qu'il n'y a généralement pas besoin de le spécifier. Ainsi, ces deux instructions produisent le même effet, à savoir ouvrir une boîte de

dialogue :

```
alert('Hello world!');   // on a utilisé cette forme
window.alert('Hello world!');  // écriture complète
```

La structure DOM (Document Object Model)

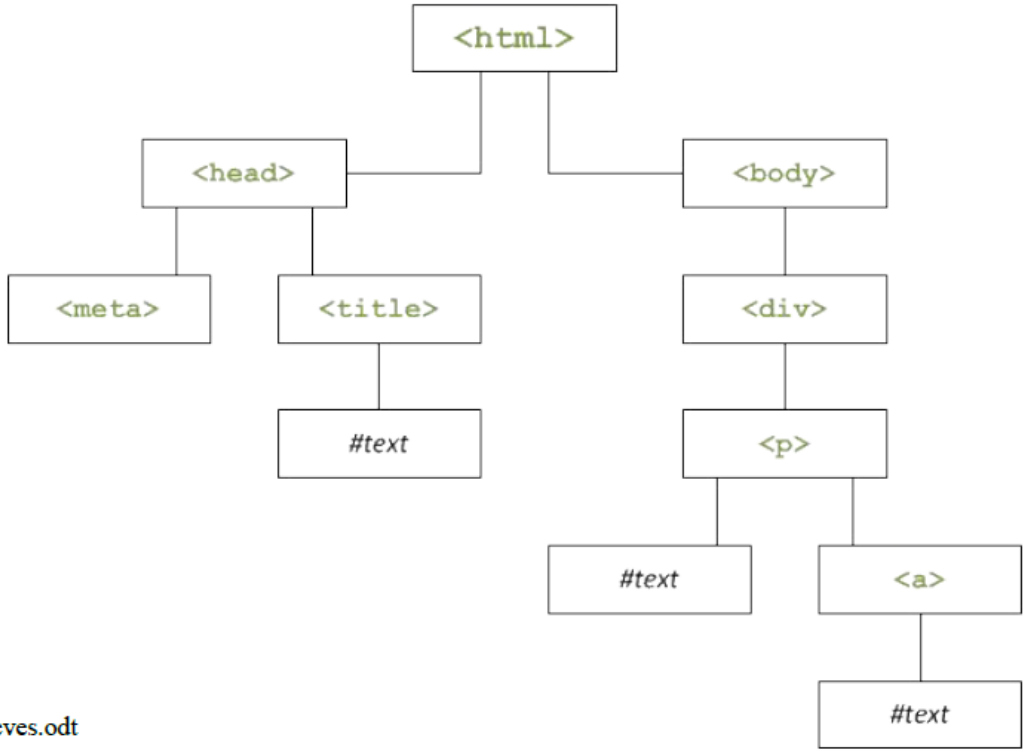
L'objet document est un sous-objet de window, l'un des plus utilisés. Et pour cause, il représente la page Web et plus précisément la balise <html>. C'est grâce à cet élément-là que nous allons pouvoir accéder aux éléments HTML et les modifier. Voyons donc, dans la sous-partie suivante, comment naviguer dans le document.Comme il a été dit précédemment, le DOM pose comme concept que la page Web est vue comme un arbre, comme une hiérarchie d'éléments.

Soit la page web suivante :

```
<!doctype html>
<html>
   <head>
      <meta charset="utf-8" />
      <title>Le titre de la page</title>
   </head>

   <body>
      <div id= "monDiv1">
         <p>Un peu de texte <a>et un lien</a></p>
      </div>
   </body>
</html>
```

On peut donc schématiser cette page Web simple comme ceci :



Remarque : on va utiliser des balises pour les différents éléments composant le corps (body) de la page. Les éléments vont avoir un nom donné par la structure suivante : . Ainsi, avec javascript, on va pouvoir modifier ces éléments grâce à leur nom.

Accéder aux éléments :

L'accès aux éléments HTML via le DOM est assez simple. L'objet document possède trois méthodes principales : getElementById(), getElementsByTagName() et getElementsByName().

getElementById()

Cette méthode (cela s'appelle comme ça) permet d'accéder à un élément en connaissant son "Id" qui est simplement l'attribut id de l'élément. Cela fonctionne de cette manière :

```
<div id="monDiv1">
   <p>Un peu de texte <a>et un lien</a></p>
</div>

<script>
   var objet_div = document.getElementById('monDiv1');
   alert(div);
</script>
```

Ainsi la variable objet_div contient l'objet qui s'appelle "monDiv1"

Accéder aux éléments grâce aux technologies récentes

Ces dernières années, le JavaScript a beaucoup évolué pour faciliter le développement Web. Les deux méthodes que nous allons étudier sont récentes et ne sont pas supportées par les très vieilles versions des navigateurs, leur support commence à partir de la version 8 d'Internet Explorer, pour les autres navigateurs vous n'avez normalement pas de soucis à vous faire.

Ces deux méthodes sont querySelector() et querySelectorAll() et ont pour particularité de grandement simplifier la sélection d'éléments dans l'arbre DOM grâce à leur mode de fonctionnement. Ces deux méthodes prennent pour paramètre un seul argument : une chaîne de caractères !

Soit la page web suivante :

```
<div id="menu">

   <div class="article">
      <span>Élément 1</span>
      <span>Élément 2 : suite</span>
   </div>

   <div class="publicite">
```

```
      <span>Élément 3</span>
      <span>Élément 4 : dernier élément</span>
   </div>

</div>

<div id="contenu">
   <span>Introduction au contenu de la page...</span>
</div>
```

la partie javascript :

```
<script>
   var recherche1 = document.querySelector('#menu .article span'),
   var recherche_tous = document.querySelectorAll('#menu .article span');
   alert(recherche1.innerHTML); // Affiche : "Élément 1"
   alert(recherche_tous.length); // Affiche : "2"
   alert(recherche_tous[0].innerHTML + ' - ' + recherche_tous[1].innerHTML); // Affiche : "Élément 1 -
Élément 2 : suite"
</script>
```

que se passe-t-il ?

La variable " recherche1" contient l'objet Élément 1 c'est à dire " Élément 1" (c'est le premier élément correspondant à la recherche rencontré dans la page

La variable "recherche_tous" contient tous les éléments correspondant à "article". Pour connaître le nombre d'éléments on utilise "recherche_tous.length" (ici 2) et pour les récupérer : recherche_tous[i].innerHTML avec i = 0 ou 1 (0= 1er élément, 1= 2ème élément etc …. s'il y a plus de 2

2. Des exemples de manipulation d'une page web avec javascript

Exemple 1 : modification dynamique d'un lien (méthode : getAttribut et setAttribute)

soit le code suivant :

```
<!DOCTYPE html>
<html>
<head>    <meta charset="utf-8" />
<title>Essais javascript</title>
</head>
<body>
<br/><br/>
<a id="monlien" href="http://lycee-jeanrostand.fr/">Le lien sur lequel cliquer</a>
```

```
<script>
if (confirm('si êtes vous êtes élèves en BTSSN1 ou BTSSN2, cliquez sur "OK" '))
{ alert('Vous allez pourvoir cliquer sur le lien pour aller vers le site qui vous correspond.');
var le_lien = document.getElementById('monlien');
var href = le_lien.getAttribute('href'); // On récupère l'attribut « href »
le_lien.setAttribute('href', 'https://openclassrooms.com/courses/apprenez-a-coder-avec-javascript'); // On édite l'attribut « href »
}
else
{
alert("Vous allez pourvoir cliquer sur le lien pour aller vers le site du lycée");
}
</script>
</body>
</html>
```

a) ***Essayez ce code (en répondant différemment à la question et en rechargeant la page) et expliquez son rôle.***

b) ***Expliquez, de manière détaillée, comment le code fonctionne.***

Avertissement sur la limite du "CTRL + U" : on a pris l'habitude, notamment quand on travaille en PHP, de récupérer le code de la page avec "CTRL + U". Cela permettait notamment de voir les erreurs. Avec Javascript cela ne fonctionne pas car la page récupérée est celle chargée par le navigateur et les modifications n'apparaissent passe-t-il

Vérification :

c) ***Toujours sur la page précédente, pour chaque réponse possible, faites "CTRL + U" et comparer les résultats obtenus. Cela correspond-il à l'avertissement précédent ?***

Exemple 2 : modification d'un attribut CSS (méthode : class.name)

Remarque : au lieu de créer un fichier CSS on va utiliser la balise <style> qui permet de gérer les attribut CSS dans le code lui-même. Pour notre exemple c'est plus pratique et plus "parlant". Par contre ce n'est pas recommandé (il doit toujours y avoir un fichier CSS séparé).

soit le code suivant :

```
<!doctype html>
<html>
  <head>
```

```
    <meta charset="utf-8" />
    <title>Le titre de la page</title>
    <style>
      .fond_bleu {
        background: blue;
        color: white;
      }
      .fond_rouge {
        background: red;
        color: white;
      }
    </style>
  </head>

  <body>
    <div id="mon_texte">
      <p>Un peu de texte <a>et un lien</a></p>
    </div>

    <script>
        var texte = document.getElementById('mon_texte');
      if (confirm('voulez-vous un  changement de couleur?')) {
            texte.className = 'fond_bleu';
        }
        else {
            texte.className = 'fond_rouge';
        }
    </script>
  </body>
</html>
```

a) Essayez ce code (en répondant différemment à la question et en rechargeant la page) et expliquez son rôle.

b) Expliquez, de manière détaillée, comment le code fonctionne.

Exemple 3 : rajout dans un élément HTML (méthode : innerHTML)

soit le code suivant :

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Le titre de la page</title>
  </head>

  <body>
    <div id="mon_texte">
```

```
        <a>Un peu de texte </a>
    </div>

    <script>
        var texte = document.getElementById('mon_texte');
        texte.innerHTML += ' <strong>et une portion en plus</strong>.';
    </script>

  </body>
</html>
```

a) S'il n'y avait pas la partie en Javascript, qu'afficherai notre page ?
b) Essayez maintenant ce code et décrivez ce qu'il fait.
c) Expliquez, de manière détaillée, comment le code précédent fonctionne.

Soit le code modifié :

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Le titre de la page</title>
  </head>

  <body>
    <div id="mon_texte">
      <a>Un peu de texte </a>
    </div>

    <script>
        document.getElementById('mon_texte').innerHTML += ' <strong>et une portion en plus</strong>.';
    </script>

  </body>
</html>
```

d) Essayez maintenant ce code et expliquez la différence avec le précédent

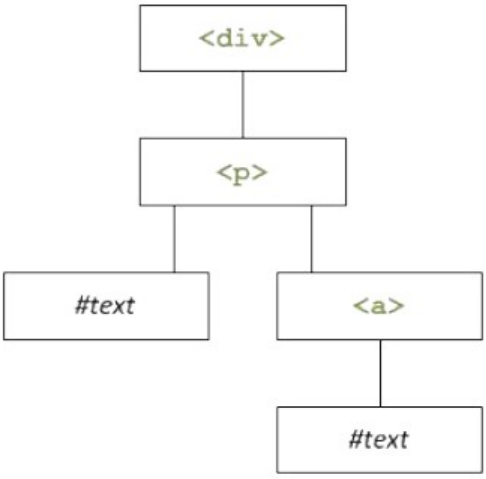
Exemple 4 : rajout d'éléments HTML

Là ça va être un peu plus compliqué

on a le code suivant : (partie "body") :

```
<body>
<div>
   <p id="mon_texte1">Un peu de texte <a>et un lien </a></p>
</div>
</body>
```

La structure DOM de notre code est la suivante :

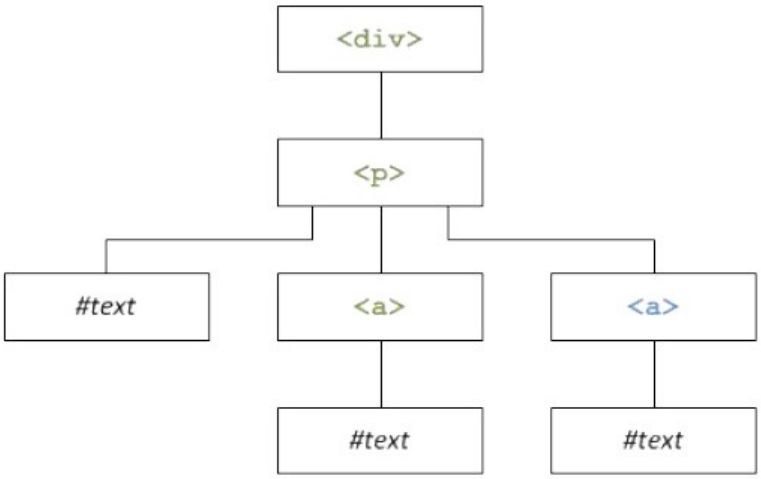


On voudrait avoir le code suivant :

```
<body>
<div>
   <p id="mon_texte1">Un peu de texte <a>et un lien </a>
      <a id="sti_lien" href="http://sti2d.ecolelamache.org/">Le Site de la section STI</a></p>

</div>
</body>
```

La structure DOM devient :



Si l'on veut faire ça en Javascript, il va falloir créer la balise <a>, lui associer le lien (avec href) et un texte et placer tout ça dans la branche voulue .

Voici le code proposé ;

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>Le titre de la page</title>
</head>
   <body>
         <div>
         <p id="mon_texte1">Un peu de texte <a>et un lien </a></p>    </div>
   <script>
                  var mon_ajout = document.createElement('a');
      mon_ajout.id    = 'Open_lien';
                  mon_ajout.href  = 'https://openclassrooms.com/courses/apprenez-a-coder-avec-
javascript';
      document.getElementById('mon_texte1').appendChild(mon_ajout);
                  var newLinkText = document.createTextNode("Le Site OpenClassRoom");
                  mon_ajout.appendChild(newLinkText);
         </script>
                  </body>
                  </html>
```

a) Essayez ce code. Fait-il exactement ce que l'on voulait ?.
b) Expliquez, de manière détaillée, comment le code précédent fonctionne.

III. D'autres fonctions à travers des exemples

Les événements Javascript (clic de la souris, déplacement de la souris, frappe au clavier, ….)

Exemple 1 : methode addEventListener

Soit le code suivant :

```
<!doctype html>
<html>
   <head>
      <meta charset="utf-8" />
      <title>Le titre de la page</title>
```

```
</head>

<body>
<div id="position"></div>

    <script>
        var position = document.getElementById('position');

        document.addEventListener('mousemove', function(e) {
            position.innerHTML = 'Position X : ' + e.clientX + 'px<br />Position Y : ' + e.clientY + 'px';
        }, false);
    </script>
</body>
</html>
```

Essayez ce code et décrivez ce qu'il fait.

Exemple 2 : JavaScript Events

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>essai onclick</title>

</head>
<body>

<p>Click sur le bouton</p>

<button onclick="displayDate()">On est le ?</button>

<script>
function displayDate() {
    document.getElementById("demo").innerHTML = Date();
}
</script>

<p id="demo"></p>

</body>
</html>
```

Essayez ce code et décrivez ce qu'il fait.

Exemple 3 :

Soit le code suivant :

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>des carrés à déplacer</title>
 </head>

 <body>

  <style type="text/css">
     .draggableBox {
        position: absolute;
        width: 80px; height: 60px;
        padding-top: 10px;
        text-align: center;
        font-size: 40px;
        background-color: #222;
        color: #CCC;
        cursor: move;
     }
  </style>

  <div class="draggableBox">1</div>
  <div class="draggableBox">2</div>
  <div class="draggableBox">3</div>

  <script>
     (function() { // On utilise une IIFE pour ne pas polluer l'espace global
        var storage = {}; // Contient l'objet de la div en cours de déplacement

        function init() { // La fonction d'initialisation
           var elements = document.getElementsByTagName('div'),
              elementsLength = elements.length;

           for (var i = 0 ; i < elementsLength ; i++) {
              if (elements[i].className === 'draggableBox') {

                 elements[i].addEventListener('mousedown', function(e) { // Initialise le drag & drop
                    var s = storage;
                    s.target = e.target;
                    s.offsetX = e.clientX - s.target.offsetLeft;
                    s.offsetY = e.clientY - s.target.offsetTop;
                 }, false);

                 elements[i].addEventListener('mouseup', function() { // Termine le drag & drop
```

```
                storage = {};
            }, false);
        }
    }

    document.addEventListener('mousemove', function(e) { // Permet le suivi du drag & drop
        var target = storage.target;

        if (target) {
            target.style.top = e.clientY - storage.offsetY + 'px';
            target.style.left = e.clientX - storage.offsetX + 'px';
        }
    }, false);
}

init(); // On initialise le code avec notre fonction toute prête.
})();
</script>

</body>
</html>
```

Essayez ce code et décrivez ce qu'il fait.

Exemple 4 :un jeu en javascript et HTML

récupérez l'archive "serpent html.zip" et essayez le jeu réalisé en Javascript

IV. Pour aller plus loin

Bien évidemment nous n'avons fait que survoler Javascript.

Pour compléter ce TP :

- sur le site "openclass Room" il y a une formation complète sur javascript (https://openclassrooms.com/courses/dynamisez-vos-sites-web-avec-javascript)