TRAITEMENT COTÉ CLIENT : INITIATION JAVASCRIPT

1. Présentation

Avec HTML (contenu) et CSS (forme), langage JavaScript (comportement de la page web) est le 3ème langage essentiel du développement web. Il permet de réagir aux interactions de l'utilisateur (modification du contenu de la page web, de sa forme, animations ….) Son traitement s'effectue coté client donc directement dans le navigateur.

Javascript a été créé en 1995 par **Brendan Eich** (en seulement 10 jours !). Le langage a depuis été standardisé sous le standard **ECMAScript** (abrégé en ES) qui évolue chaque année. La dernière principale version (et compatible avec tous les navigateurs) est **ES6**.
Aujourd'hui, il est même possible d'utiliser JavaScript pour faire de la programmation coté serveur (cf. Nodejs).

Remarque : En dépit de son nom, Javascript est très différent d'un autre langage très populaire : Java.

Programme officiel NSI : Normalement, les capacités attendues au programme de NSI consistent juste à repérer puis modifier du code JavaScript afin de programmer une interaction simple (ex : clic sur un bouton -> modification d'un élément de page). Vous ne devriez pas avoir à créer un programme JavaScript de zéro.

2. Où écrire du JavaScript

De manière un peu similaire à CSS, JavaScript est lié à HTML et peut s'écrire soit :

- Directement dans le code html grâce à la balise <script>
- Dans un **fichier séparé** (méthode recommandée essentiellement pour des raisons de maintenance de code). On fait le lien vers le fichier .js (JavaScript) grâce à l'instruction : `<script src="fichierjs.js"></script>`

```
<!DOCTYPE html>
<html>
    <head>
        <title>Hello World!</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <p>Les instrcutions Jaavscript peuvent se placer
        directement dans le code HTML</p>
        <script>console.log("Bonjour (dans la console)");
        </script>
        <p> Ou dans un fichier javascript séparé,
        indiquer par l'instruction suivante :</p>
        <script src="fichierjs.js"></script>
    </body>
</html>
```

Il est conseillé de placer le lien vers le fichier script juste **avant la fermeture de la balise fermante </body>** de manière à la page HTML soit d'abord chargée avant d'exécuter le code JavaScript.(ou rajouter l'attribut **defer** dans la balise script)

3. Quelques éléments de syntaxe

On va ici se contenter de fournir les **règles de base** du langage JavaScript :

- Chaque instruction doit **terminer par un point-virgule**
- Les variables sont déclarées par **let (**avec une portée de bloc) ou **var (**portée de fonction plus d'info ici).

(Javascript est un langage à typage dynamique, il reconnait automatiquement le type de variable)

- Chaque bloc d'instruction est délimité par des **accolades** :
- La boucle for a la même syntaxe qu'en PHP
- La concaténation de caractères se fait par le signe +

```
function carré(n)
{
    return n*n;
}

let mavariable="du texte";
mavariable="maintenant un autre";
mavariable=5;
for (let i = 0; i <= mavariable; i++)
{
    console.log("le carré de "+i+" est "+carré(i));
}
```

☞ Ouvrir : ***Exemple_1.html***

La création de tableau se fait comme dans python : **let monTableau=[1,2,3,4];**
L'accès à un élément également : console.log(**monTableau[0]**); affichera 1 dans la console.

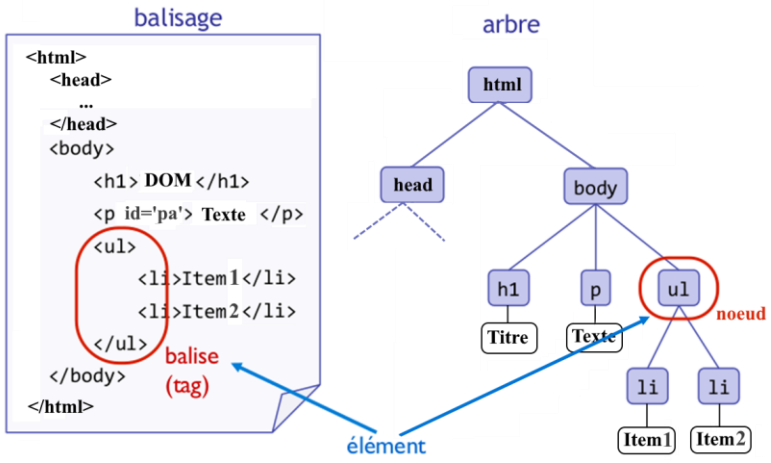
Les mots clés pour les instructions conditionnelles sont : **if, else if, else**.
Les opérateurs logiques de bases : et ↔ **&&** , ou ↔ **||** , différent ↔ **!=** , egalité ↔ **==**

4. Modifications des éléments d'une page HTML

a. Accès aux éléments de la page

Les éléments (balises) contenus dans votre page web sont "stockés/représentés" dans le document object model (DOM). En fait, DOM est une interface de programmation qui permet à des scripts d'examiner et de modifier le contenu du navigateur web. On le représente sous forme d'un arbre, où chaque nœud est un objet (balise) du document (page html).

Exemple partiel de DOM:



Pour accéder aux objets de la page, donc du DOM (afin de le modifier, de lire son contenu, son style…), plusieurs méthodes existent :

- Accès par l'identifiant : **document.getElementById('identifiant')**

Avec l'instruction **let para= document.getElementById('pa');** la variable para est l'objet "correspondant" la balise p repérée par l'identifiant '**pa**'.
On peut alors par exemple modifier son contenu avec l'instruction : **para.innerHTML='un autre texte';**

- Accès par type de balise : **document.getElementsByTagName ('typebalise')**

Renvoie un tableau contenant **les** objets ayant la classe demandée :

document.getElementsByTagName('li')[0].innerHTML='Ligne1'; changera le texte de la première balise li. (Item1)

- Accès par classe : **document.getElementsByClassName ('nomclasse')**

- Accès par sélecteur : **document.querySelector('sélecteur') ou querySelectorAll('sélecteur')**

querySelector() : retourne le premier élément trouvé satisfaisant au sélecteur ou null si rien n'est trouvé.
querySelectorAll() : retourne un tableau contenant tous les éléments satisfaisant au sélecteur, dans l'ordre dans lequel ils apparaissent dans l'arbre du document.

Ex : **document.querySelector('#pa')**.innerHTML='un autre texte'; équivalent à **document.getElementById('pa')**

b. Lecture/ modification des contenus et propriétés des éléments

Maintenant que vous savez comment accéder aux éléments, il est très simple les lire ou les modifier

- Pour accéder et donc modifier ou lire le contenu d'un élément, on utilise l'attribut innerHTML : ex : element**.innerHTML+**="du texte en plus"; (ajoute "du texte en plus" au texte déjà présent dans l'element

- Pour accéder et donc modifier ou lire le style d'un élément, on utilise l'attribut style suivi de la propriété : ex : element.**style.color='red' (colorie en rouge le texte de l'element)**

Exemple : pour colorier en jaune la couleur d'arrière-plan d'une balise div ayant l'identifiant **ecran :**

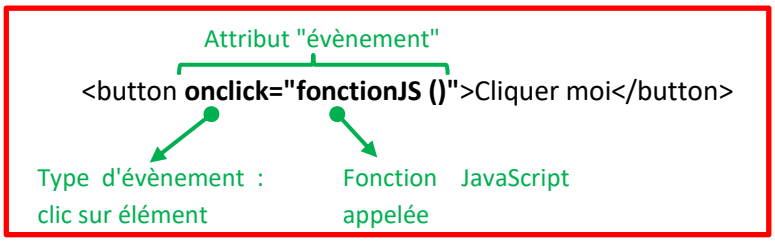
document.querySelector('#ecran').style.background='yellow';

Rq : Les propriétés n'ont pas forcement la même syntaxe qu'en css : ici, en CSS background-color, mais background en JavaScript (à cause du tiret,qu'il faut oter)

5. Evènements

Grâce à JavaScript, on peut réagir à des évènements provoqués par l'utilisateur en ajoutant un attribut "évènement" dans la **balise HTML** de l'élément "cible", qui pointera sur une fonction JavaScript avec la syntaxe : **évènement="fonction_à_appeler".**

Exemple de code HTML :



Voici les principaux évènements gérés :

Attribut évènement (Eventhandler)	Description
onclick	Clic souris sur l'élément
ondblclick	Double-clic de la souris sur l'élément
onkeypress	Appui sur une touche caractère du clavier sur l'élément
onkeydown	Appui sur n'importe quelle touche de clavier sur l'élément
onmouseover / onmouseout	Survol de la souris sur l'élément / quand la souris quitte l'élément
onfocus / onblur	Au ciblage (≈sélection) de l'élément / Quand on quitte l'élément (perte du focus)

Vous trouverez plus d'infos et d'évènements notamment sur Openclassroom.

☞ **Exemples :** code à tester sur ***Exemple_2.html***

- **Instructions directement écrite en valeur de l'attribut** :

```
<button  type="button"   onclick="document.getElementById('demo').innerHTML='date
:'+Date()" > Donner la date </button>
<span id="demo">Date : </span>
```

Rq : Dans ce cas veuillez à bien utiliser les guillemets double et simples (ex : ici ne pas utiliser les guillemets double demo et la chaîne de caractère 'date :' pour que la valeur soit un seul bloc entre " ")

- **Utilisation avec appel d'une fonction** (définie dans des balises ou un fichier JavaScript)

```
<button id='bouton1' type="button"   onclick="changeCouleur1()" >
Changez ma couleur</button>
<script type="text/javascript">
      function changeCouleur1()
      {
            document.getElementById('bouton1').style.background='yellow';
      }
</script>
```

☞ ***Petit exercice d'application :***

Modifier le code JSFiddle pour que le changement de couleur (code ci-dessus) se fasse par une instruction javascript directement mise en valeur d'attribut HTML, et au contraire pour l'écriture de la date (exemple précédent), que cela soit fait par ue fonction JS. (Enregistrer une copie avant de modifier le code !)

- **Récupération de la valeur d'un élément de formulaire :**

```
<input type="range" name="curseur" min="0" max="100"
onchange="document.getElementById('valeurCurseur').innerHTML=this.value" >
<p>valeur du curseur : <span id='valeurCurseur'> </span></p>
```

☞ ***Petit exercice d'application :***

Ajouter sur Jsfiddle un élément de formulaire input de type number, et rajouter l'instruction javascript permettant d'afficher le nombre rentré par une commande alert (cf https://wprock.fr/blog/javascript-boites-dialogue-alert-confirm-prompt/) dès que le nombre rentré change.

- **Utilisation du mot clé this (**)**

L'opérateur this (utilisation complexe) dans l'exemple ci-dessous, représente l'objet dans lequel il est appelé par une fonction (en fait méthode car POO), donc respectivement le bouton puis l'élément input. Cela permet d'utiliser une fonction javascript qui changera l'élément qu'elle recevra en paramètre.

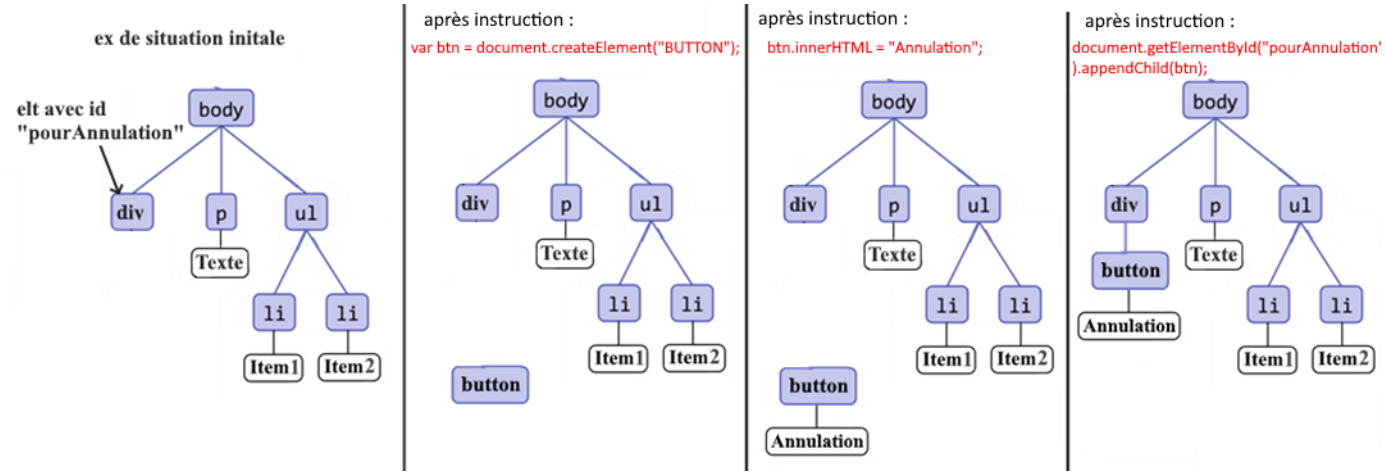
```
<button type="button"   onclick="changeCouleur2(this)" >Changez ma couleur</button>

<input type="text" name="entree" onclick="changeCouleur2(this)" >

<script type="text/javascript">
      function changeCouleur2(element)
      {
            element.style.background='blue';
            element.style.color='white';
      }
</script>
```

- **Parcours dans le DOM (***)**

Il est possible de repérer/sélectionner les éléments de la page HTML en parcourant le DOM en utilisant les notions de parent d'un élément (balise incluant l'élément) ou d'enfant (balises contenues dans l'élément).

```
var btn = document.createElement("BUTTON");
btn.innerHTML = "Annulation";
document.getElementById("pourAnnulation").appendChild(btn);
```

Ces instructions créent un élément noté btn, de type button, dans lequel on place le texte Annulation, puis on place ce bouton dans la balise ayant pour identifiant pourBouton. (appendChild rajoute un élément enfant dans le DOM)
On peut représenter ce qu'il se passe dans le DOM à chaque instruction par le schéma ci-dessous (sur un DOM fictif où l'élément ayant pour identifiant "pourAnnulation" est une balise div :



Autre méthode de gestion des évènements :

Pour réagir aux événements sur la page web, il est aussi possible d'utiliser la méthode **addEventListener()** (cette méthode offre même plus de possibilité que nous n'étudierons pas ici) , avec la syntaxe suivante :

elementASurveille.addEventListener('evenement', fonction)

Exemple : 2 instructions réalisant ma même chose :

- document.getElementById('bouton1').addEventListener('click', changeCouleur1) (dans le code JS)
- `<button id='bouton1' type="button" onclick="changeCouleur1()" >` (dans le code HTML)

Remarque importante :

Avec addEventListener, le nom des événements n'est plus exactement le même, il faut enlever le "on" au début du mot.