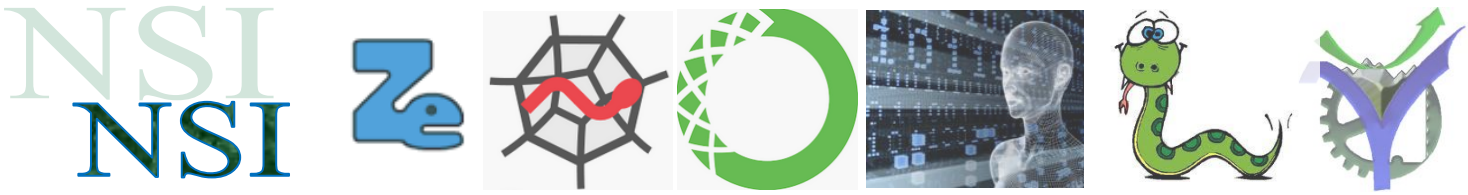


TP3-Interaction homme machine sur le web coté client

Sommaire :

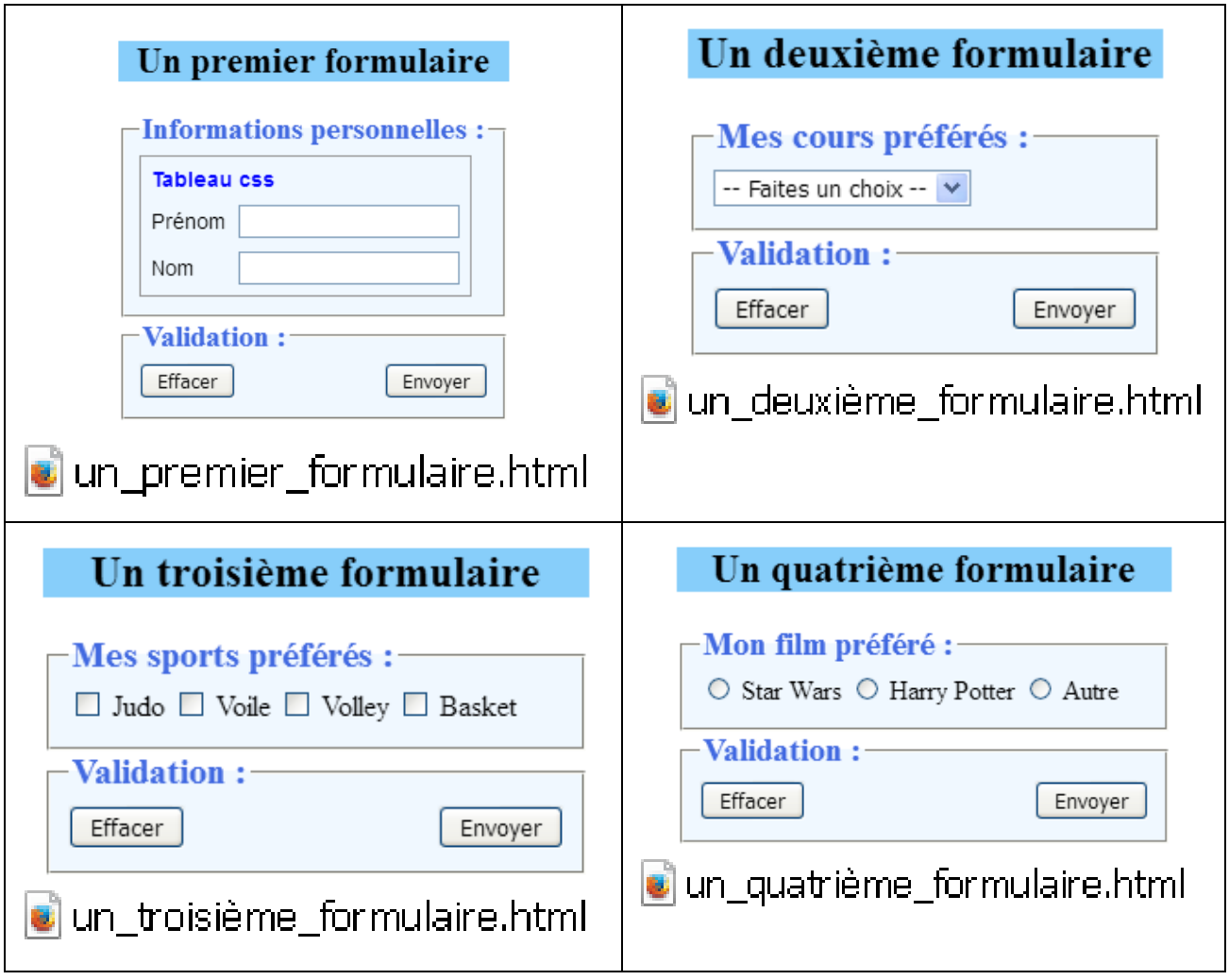
1 Un formulaire HTML

La première interactivité client est réalisée avec des formulaires html comme il a été vu dans le document de cours. Nous listons ci-dessous quelques éléments de formulaire qui vont être utiles pour le premier travail de mise en œuvre.

1.1 Quelques formulaires html :

a) Présentation de quatre cas typiques

Voilà quelques formulaires avec l'indication du fichier html correspondant. Après étude de ce paragraphe vous compléterez la feuille réponse.



Nous trouvons quatre types d'éléments de formulaire de saisie :

[1] saisie texte ; [2] saisie d'un élément dans une liste ; [3] cases à cocher multiples

[4] boutons radio choix unique.

b) Quelques éléments techniques sur la form

Un point particulier : pour l'analyse de ces formulaires nous avons configuré le formulaire de la manière suivante :

```
<form method="get" action="">
```

La méthode GET permet de tester visuellement sans code supplémentaire la réponse du formulaire au bouton submit.

Par ailleurs aucune action n'est paramétrée, ce qui est considéré comme une faute dans les outils de vérification de code. Cela n'est pas gênant ici car nous souhaitons uniquement tester notre saisie html sans analyse postérieure.

Les ressources principales disponibles sont ici à consulter si besoin :

https://www.pierre-giraud.com/html-css-apprendre-coder-cours/form-input-label-textarea-select

et https://developer.mozilla.org/fr/

c) Un petit tableau en css

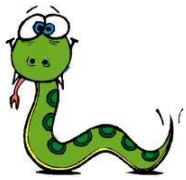
Pour faciliter la mise en page nous avons utilisé un petit tableau en css.

```
<table class="signup" border="0" cellpadding="2" cellspacing="5" bgcolor="aliceblue">
  <th colspan="2"><span id="titre">Tableau css</th>
    <tr><td><label for="prenom" >Prénom</label></td>
      <td><input type="text" id="prenom" maxlength="32" name="prenom"></td></tr>
    <tr><td><label for="nom" >Nom</label></td>
      <td><input type="text" id="nom" maxlength="32" name="nom"></td></tr>
</table>
```

Avec les ressources citées vous pourrez répondre aux questions posées dans le TP.

https://developer.mozilla.org/fr/docs/Apprendre/CSS/Building_blocks/Styling_tables




1.2 Réalisez votre formulaire personnalisé

Exercice Interface1.

Vous êtes responsable des services techniques d'un Hôtel et pour faciliter l'organisation du service des petits déjeuners. Vous proposez à vos clients la saisie d'un formulaire personnalisé leur permettant de choisir ce qu'ils désirent comme plateau de petit déjeuner le lendemain. Une seule contrainte utiliser au moins trois sur quatre des options de saisies expliquées ci-dessus.

2 L'interactivité avec le JavaScript

2.1 Introduction au langage

a) Introduction

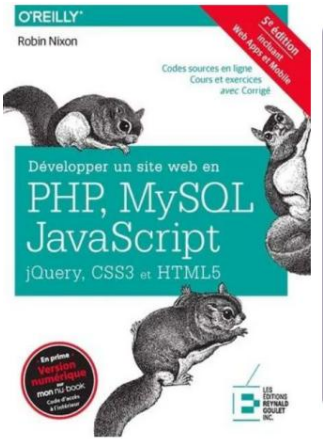
Le JavaScript est tout d'abord un langage de programmation complet et donc il contient comme tout langage de haut niveau **(résumé)** :

- Des variables, des constantes et des types
- Des structures de contrôles
- Des opérateurs
- Des fonctions
- Une approche orientée objet
- Des tableaux

Il est doté en plus de possibilités associées aux traitements, sur et dans, les pages HTML à savoir **(résumé)** :

- Manipulation du DOM
- Utilisation des expressions régulières
- Gestion asynchrone

Deux excellentes ressources pour étudier plus en détail le JavaScript en fonction des besoins :

En librairie :	Sur la toile :
 Développer un site web en PHP, MySQL JavaScript. Robin Nixon, Éditions REYNALD GOULET.	 PIERRE GIRAUD https://www.pierre-giraud.com/javascript-apprendre-coder-cours/

b) Quelques éléments du langage pour nos réalisations

Un bon résumé ici : la_syntaxe_javascript.pdf

c) Comment tester nos exemples JavaScript sans se prendre la tête

Pour tester nos exemples il suffit de les incorporer à un fichier html ne contenant que la balise <script> et de l'ouvrir dans un navigateur :

Exemple a) exemple_fonction_1.html

```
1  <script>
2    // Script js contenu dans exemple13-4.html
3    function produit(a, b)
4    {
5      return a*b
6    }
7    valeur1 = 4; valeur2 = 3;
8    document.write("JavaScript essai d'une fonction" +'<br>')
9    document.write(valeur1 + '*' + valeur2 + '=' + produit(valeur1,valeur2))
10 </script>
```

Saut de ligne

Et voilà le résultat :

JavaScript essai d'une fonction
4*3=12

Exemple b) exemple_tableau_1.html

```
<script>
  monTableau = Array(
    Array(-2,4),
    Array(1,-3))

  for (j = 0 ; j < 2 ; ++j)
  {
    for (k = 0 ; k < 2 ; ++k)
      document.write(monTableau[j][k] + " ")

    document.write("<br>")
  }

</script>
```

Un exemple de calcul sur un tableau de valeur.

Affiché à l'écran avec deux boucles imbriquées.

-2 4
1 -3

Exemple c) exemple_tableau_2.html

Ici dans cet exemple on met un en-tête un peu plus complet en html

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <title>Interface client</title>
        <meta charset= "utf-8">
    </head>
    <script>
      function determinant(T)
      {
        return T[0][0]*T[1][1] - T[0][1]*T[1][0]
      }

      monTableau = Array(
        Array(2,4),
        Array(1,-3))

      document.write("Matrice"+"<br>")
      document.write("<pre>")

      for (j = 0 ; j < 2 ; ++j)
      {
        for (k = 0 ; k < 2 ; ++k)
          document.write(monTableau[j][k] + " ")

        document.write("<br>")
      }

      document.write("</pre>")

      document.write("Déterminant = " +determinant(monTableau))
    </script>
</html>
```

Matrice

2 4
1 -3

Déterminant = -10

2.2 A vous de jouer, vos premiers scripts en JavaScript

a) Solutions de l'équation $y = a \cdot x^2 + b \cdot x + c$

Exercice Interface2.

En vous inspirant de l'exemple exemple_fonction_1.html ci-dessus, réalisez un script qui calcul les racines d'un polynôme du second degré.

b) Calcul de déterminant de matrice 3x3

Exercice Interface3.

Les matrices sont des objets mathématiques organisés en tableau de nombres. Vous les rencontrerez beaucoup dans vos études supérieures. Elles sont pour nous uniquement l'occasion de calculer des scripts avec des données de type tableaux.

A partir des éléments du site ci-dessous proposer un script qui calcul le déterminant dune matrice [3,3].

Vous vérifierez vos résultats avec la calculette du site.

https://www.dcode.fr/determinant-matrice

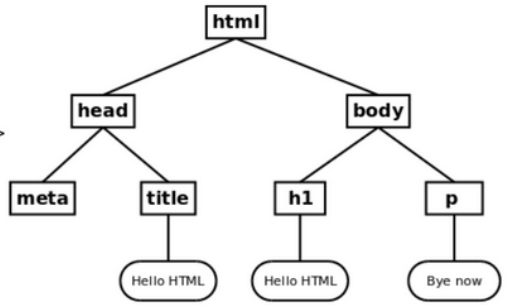
Pour les matrices de taille supérieure comme 3x3, on effectue le calcul :

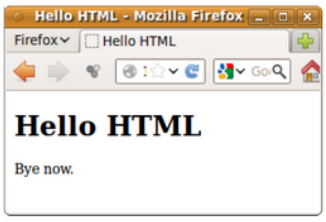
$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$
$$= aei - afh + bfg - bdi + cdh - ceg$$

2.3 JavaScript et le DOM

Le JavaScript accède au DOM et peut en modifier les caractéristiques[1].

```
<!doctype html>
<html lang="fr">
<head>
  <meta charset="utf-8">
  <title>Hello HTML</title>
</head>
<body>
  <h1>Hello HTML</h1>
  <p>Bye now.</p>
</body>
</html>
```



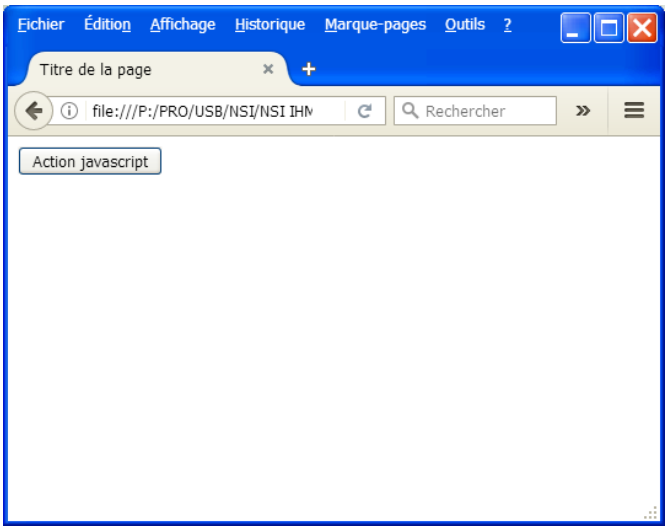


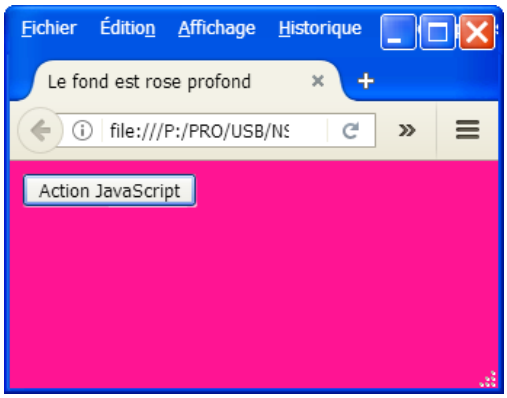
Passage du HTML au DOM puis à la représentation graphique

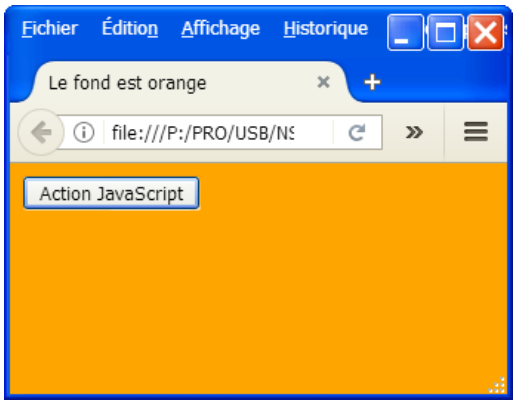
[1] https://ensweb.users.info.unicaen.fr/pres/dom-manipulation/

Pour le vérifier vous pouvez tester l'exemple suivant :

Interactivite_client_javascript_1.html








```
<!doctype html>
<html lang="fr">
    <head>
        <title>
            Action sur un bouton
        </title>
        <meta charset="utf-8"/>
    </head>
    <body>
            <button onClick="changeCouleur()">Action JavaScript</button>
            <script>
                function changeCouleur(){
                    const coul = document.body.style.backgroundColor;
                    if (coul == 'deeppink' )
                        {
                            document.body.style.backgroundColor = 'orange';
                            document.title = 'Le fond est orange'
                            }
                    else
                        {
                            document.body.style.backgroundColor = 'deeppink';
                            document.title = 'Le fond est rose profond'
                            }
                    }
            </script>
    </body>
</html>
```

a) Modification de texte dans la page

Expérimentez l'exemple : javascript_modification_texte.html

Vous avez du constater quelques petits problèmes entre ce qui est affiché et ce qui est réalisé,……

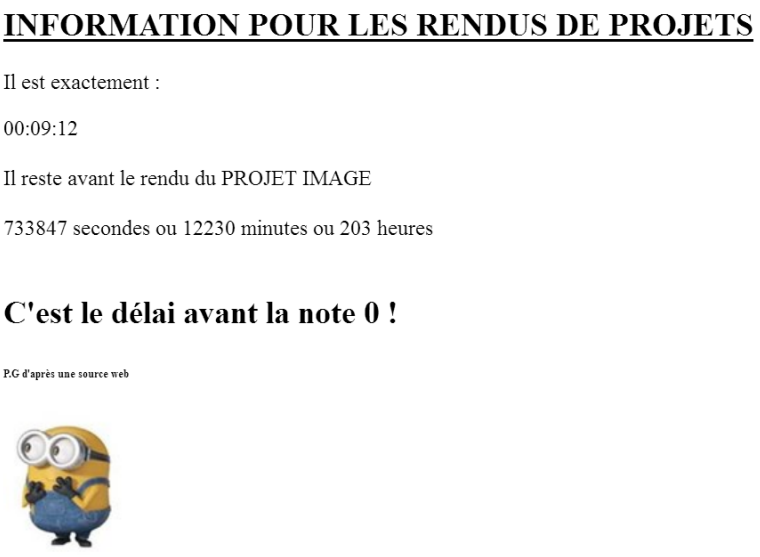
Exercice Interface4.

A vous de corriger le code. Vous pouvez ajouter deux couleurs supplémentaires.

b) Insertions de texte dans la page dans des champs identifiés par des id.

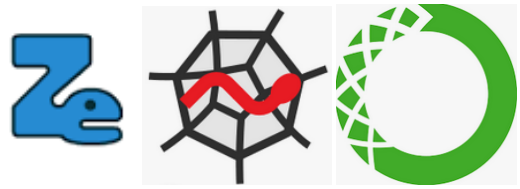
Ce script se passe de commentaires !!

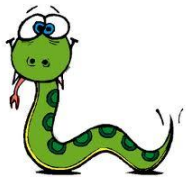
javascript_affiche_heure_rendu_DS.html



Exercice Interface5.

Après avoir testé le script identifiez puis corrigez les erreurs d'orthographe.






Le code de la page :

```html
<!DOCTYPE html>
<html lang="fr">
    <head>
        <title>Cours JavaScript</title>
        <meta charset="utf-8">
    </head>

    <body>
        <h2><u>INFORMATION POUR LES RENDUS DE PROJETS</u></h2>
        <p>Il est exactement :
        <div id="heure_exacte"></div><br>
        Il reste avant le rendu du PROJET IMAGE<br><br>
        <div id="délai_restant"></div><br>
        <h2>C'est le délai avant la note 0 !</h2>
        <h2 id="message"></h2>
        <div id="texte" style="font-weight:bold; font-size:8px">P.G d'après une source web</div>
        </p>
        <script>
            function affichZero(nombre) {
                // cette fonction prend en paramètre un nombre
                // si ce nombre est inférieur à 10, on affiche 0 + ce nombre
                // Ex: il est 07h00
                if ( nombre < 10 ) {
                    nombre = '0' + nombre;
                    }
                return nombre;
            }
```

```
            function dateEtHeure() {
                // Cette fonction est appelée toutes les secondes

                // Création de la variable date_courante qui contient la date et l'heure courante
                // const permet de créer une constante nommée qui ne pourra pas être redéfinie
                // mais elle peut avoir sa valeur modifiée.
                const date_courante = new Date();

                // Création de la variable qui contient la date du DS
                const date_rendu = new Date(2020,00,22,12,00,00)

                // Affichage de l'heure courante dans l'ID 'heure_exacte' du DOM de la page web
                document.getElementById('heure_exacte').innerHTML = ' ' +
                    affichZero(date_courante.getHours()) + ':' +
                    affichZero(date_courante.getMinutes())  + ':' +
                    affichZero(date_courante.getSeconds());

                // Message d'alerte personnalisé dès qu'il ne reste plus de temps
                // Calculé en fonction de la différence de temps restante
                message_alerte = '';
                difference = date_rendu - date_courante;
                if (difference < 0) {
                    difference = 0;
                    message_alerte = "C'est trop tard !!!"
                    }

                // Affichage du temps restant dans l'ID 'délai_restant' du DOM de la page web
                delai_restant_secondes =   Math.trunc(difference / 1000);
                delai_restant_minutes  =   Math.trunc(difference / 60000);
                delai_restant_heures   =   Math.trunc(difference / 3600000);

                document.getElementById('délai_restant').innerHTML = delai_restant_secondes +
                    ' secondes  ou ' + delai_restant_minutes +  ' minutes   ou '+
                    delai_restant_heures + ' heures';
```

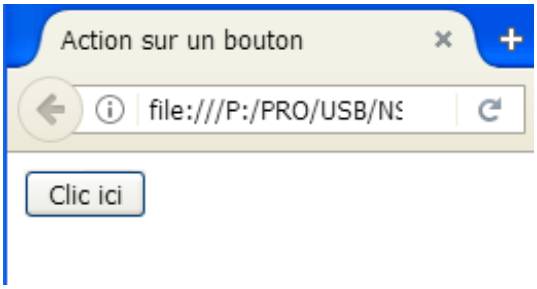
```
64                  // Modification des paramètres de l'élément ID 'message' du DOM de la page web
65                  document.getElementById('message').style.color="#f00";
66                  document.getElementById('message').innerHTML = message_alerte;
67                  document.getElementById('message').style.fontSize=32+'px';
68              }
69
70              // Pour actualiser l'heure chaque minutes, on rappelle la fonction dateEtHeure()
71              // toutes les 100 millisecondes, donc toutes les secondes
72              window.onload = function() {
73                  setInterval("dateEtHeure()", 100);
74              };
75          </script>
76          <!-- On revient dans l'html -->
77          <img src="minion.jpg"></img>
78      </body>
79  </html>
```

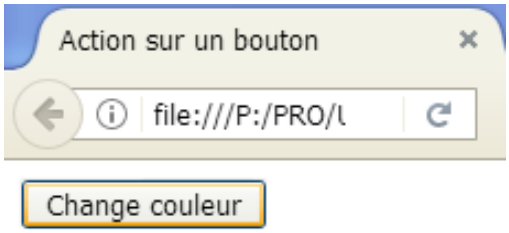
2.4 Interactivité avec les formulaires quelques éléments de fonctionnement

a) Accès à l'élément via les paramètres de l'appel par (this)

Nous pouvons insérer des interactions JavaScript un peu partout dans nos pages web. Essayons l'exemple ci-dessous :

Interactivite_client_javascript_2.html





Nous avons déjà observé l'accès à la couleur du fond du body via le DOM. Maintenant nous agissons également sur le bouton en modifiant son intitulé avec le survol de la souris.

```
<button onClick = "changeCouleur()"
        onmouseover = "messageBouton1(this)"
        onmouseleave = "messageBouton2(this)"
        >Clic ici</button>
```

Quand on clic sur le bouton la fonction changeCouleur() est appelé sans paramètre particulier dans l'appel, c'est indiqué avec les deux parenthèses 'vide' ().

Quand la souris survole le bouton la fonction messageBouton1 est appelée avec comme paramètre les éléments du bouton lui-même avec (this).

Processus identique quand la souris quitte le survol du bouton avec la fonction messageBouton2(this).

https://www.w3schools.com/jsref/event_onmouseleave.asp

https://developer.mozilla.org/fr/docs/Web/HTML/Element/Input/button

https://developer.mozilla.org/en-US/docs/Web/API/Element/mouseleave_event

Voilà le code des deux fonctions :

⇦

```
function messageBouton1(bouton){
    bouton.innerHTML = "Change couleur";
    }
function messageBouton2(bouton){
    bouton.innerHTML = "Clic ici";
    }
```

Le paramètre dans l'appel intitulé this s'intitule bouton dans la description de la fonction. La propriété **.innerHTML** permet d'accéder et / ou de modifier le contenu d'un élément de la page. Ici le bouton.

b) Accès à l'élément via un nom d'identifiant 'id'

N'importe quelle partie de la page web peut-être identifiée avec un id. Il est ensuite possible d'agir dessus comme l'exemple ci-dessous :

w3schools.com

https://www.w3schools.com/jsref/prop_html_innerhtml.asp

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

<p id="demo" onclick="myFunction()">Click me to change my HTML content (innerHTML).</p>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed!";
}
</script>

</body>
</html>
```

On peut également enregistrer l'élément du DOM dans une variable pour agir ensuite dessus. Cela permet d'effectuer plusieurs traitements sans relire le DOM pour chacun d'entre eux. Démonstration ci-dessous sur l'évènement on blur (blur = perte de focus) :

https://www.w3schools.com/jsref/event_onblur.asp

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

Enter your name: <input type="text" id="fname" onblur="myFunction()">

<p>When you leave the input field, a function is triggered which transforms the input text to upper
case.</p>

<script>
function myFunction() {
  var x = document.getElementById("fname");
  x.value = x.value.toUpperCase();
}
</script>

</body>
</html>
```

c) A vous de jouer

Exercice Interface6.

Essayez d'interagir avec au moins deux évènements au choix dans une de vos pages web : https://www.w3schools.com/jsref/obj_mouseevent.asp

Event Types

These event types belongs to the MouseEvent Object:

Event	Description
onclick	The event occurs when the user clicks on an element
oncontextmenu	The event occurs when the user right-clicks on an element to open a context menu
ondblclick	The event occurs when the user double-clicks on an element
onmousedown	The event occurs when the user presses a mouse button over an element
onmouseenter	The event occurs when the pointer is moved onto an element
onmouseleave	The event occurs when the pointer is moved out of an element
onmousemove	The event occurs when the pointer is moving while it is over an element
onmouseout	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children
onmouseover	The event occurs when the pointer is moved onto an element, or onto one of its children
onmouseup	The event occurs when a user releases a mouse button over an element

Attribut	Description
onblur	Le script démarre lorsque l'élément, le champ du formulaire, perd le focus
onchange	Script à exécuter lorsque un élément est modifié
oncontextmenu New	Script à exécuter quand un menu contextuel est déclenché
onfocus	Script à exécuter quand un élément gagne le focus
onformchange New	Script à exécuter lorsqu'un formulaire est modifié
onforminput New	Script à exécuter lors de la saisie d'un formulaire
oninput New	Script à exécuter lors de la saisie d'un élément
oninvalid New	Script à exécuter quand un élément n'est pas valide
onselect	Script à exécuter quand un élément est sélectionné
onsubmit	Script à exécuter lors de la soumission d'un formulaire

2.5 Interactivité avec les formulaires étude détaillée d'un exemple

a) Le formulaire de saisie

Nous reprenons l'exemple présenté dans le cours.

code_css_eleve_formulaire_1.css
code_html_eleve_formulaire_1_javascript.html
code_javascript_eleve_formulaire_1.js

Vous avez à ce stade beaucoup d'éléments pour comprendre le fonctionnement de ce code. Examinons quelques subtilités :

- La définition raccourcie de la valeur de la couleur #fba s'interprète comme #ffbbaa

```
champ.style.backgroundColor = "#fba";
```

- Faire un test en 'ou' entre plusieurs conditions

```
if(champ.value.length < 2 || champ.value.length > 21)
```

```
if(isNaN(age) || age < 5 || age > 111)
```

- Déclencher une analyse de conformité à une expression régulière `regex` décrivant comment une séquence de caractères doit s'écrire :

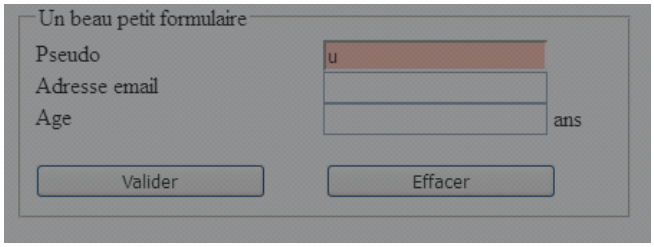
```
var regex = /^[a-zA-Z0-9._-]+@[a-z0-9._-]{2,}\.[a-z]{2,4}$/;
if(!regex.test(champ.value))
```

La méthode test() vérifie s'il y a une correspondance entre un texte et une expression rationnelle. Elle retourne true en cas de succès et false dans le cas contraire.
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/RegExp/test

- Attention à **l'évaluation paresseuse** des conditions multiples, pour tester les trois champs de saisie si nous écrivons le test de cette manière :

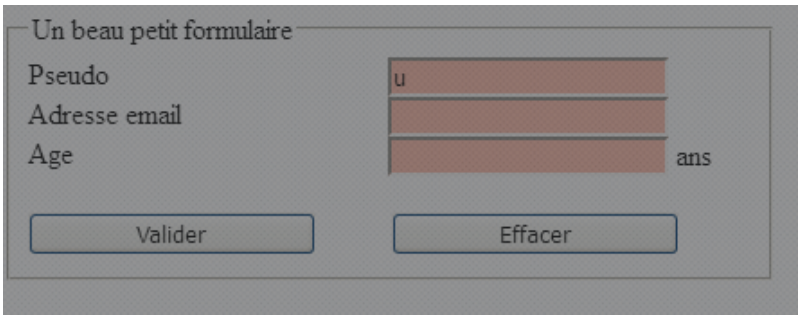
```
if(verifPseudo(f.pseudo) && verifMail(f.email) && verifAge(f.age))
   return true;
```

Comme la première valeur 0 rencontrée impose un résultat nul pour l'ensemble du test il n'est pas nécessaire d'effectuer tous les tests. Cela gagne du temps, c'est l'évaluation paresseuse. Dans notre cas il faut faire attention car si le premier test sur le champ pseudo est nul par exemple, alors les appels verifMail et verifAge ne sont pas effectués, et les champs correspondants ne sont pas testés, donc ne sont pas coloriés si faux. Le problème est identique pour un test de plusieurs conditions logiques 'ou' dès que l'un des tests renvoie une valeur vraie ou 1.



La bonne solution est d'effectuer chaque test isolément puis de regrouper les résultats le fonctionnement est correct tous les champs sont testés :

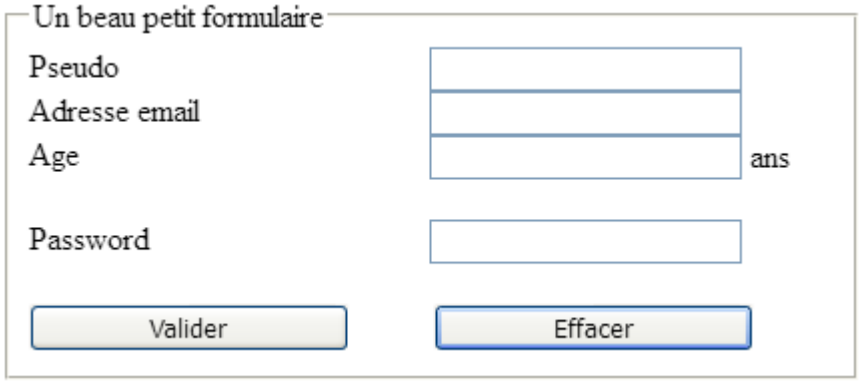
```
var pseudoOk = verifPseudo(f.pseudo);
var mailOk = verifMail(f.email);
var ageOk = verifAge(f.age);

if(pseudoOk && mailOk && ageOk)
    return true;
```



b) Ajout d'un champ de saisie de mot de passe

Exercice Interface7.

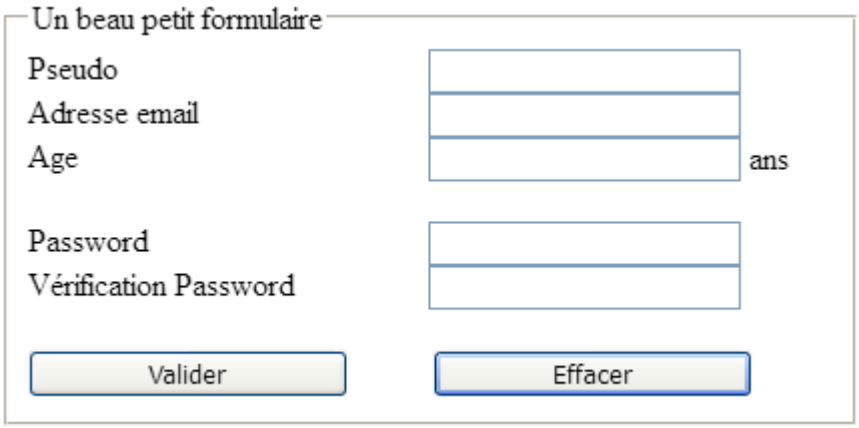
Ajoutez un champ de saisie de mot de passe au formulaire. Et vérifiez que le mot de passe fait au moins 8 caractères.



c) Ajout d'un champ double champ de saisie du mot de passe

Exercice Interface8.

Ajoutez un deuxième champ de saisie de mot de passe au formulaire. Et vérifiez que le mot de passe fait au moins 8 caractères. Validez le formulaire si les deux mots de passes sont identiques.



d) Un complément 'de propreté'

Pour complément ajout d'une fonction qui efface le coloriage des champs erronés en cas d'appui sur le bouton de réinitialisation du formulaire.

```
<form method="GET" action="" onsubmit="return verifForm(this)"  onreset="effaceForm(this)">
```

```
function effaceForm(f)
{
    surligne(f.pseudo, false);
    surligne(f.email, false);
    surligne(f.age, false);
    surligne(f.mdp, false);
    surligne(f.vmdp, false);
}
```

effaceForm(this) est appelé dans la balise form donc **this = form**

Les différents champs sont ensuite sélectionnés par leurs attributs 'id'.