

Pilotage des contrôles de saisie via JavaScript

1. Contrôle de saisie sur un champ texte

L'objectif ici va être simple, contrôler que l'utilisateur a bien effectué une saisie clavier dans un champ de type texte inclus dans un formulaire HTML et ceci via une fonction JavaScript. Dans ce premier script, il n'est pas prévu de vérification du type de données (numérique, alphanumérique, longueur...). Ces aspects seront vus dans les sections suivantes.

Présentation du script HTML/JavaScript

Pour ce script, le code source complet est proposé ci-après :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<!--
NOM DU SCRIPT : FORM_01.htm
REALISATION INFORMATIQUE : Christian VIGOUROUX
DATE DE CREATION : 01/01/2014
DATE DE DERNIERE MODIFICATION : 01/01/2014
OBJET : Gestion de formulaire (contrôle de saisie)
-->

<!-- Début script HTML -->
<html>

    <!-- Début en-tête script HTML -->
    <head>

        <!-- Balise meta -->
        <meta HTTP-equiv="Content-Type"
        content="text/html; charset=utf-8" />
        <!-- Titre du script HTML -->
        <title>FORM_01</title>

        <!-- Début script JavaScript -->
        <script type="text/JavaScript">

            /* Fonction de test de saisie */
            function nonVide(champ, messageAlerte)
            {
                if (champ.value.length == 0)
                {
                    /* Affichage d'un message d'alerte */
                    alert(messageAlerte);
                    /* Focus sur le champ en erreur */
                    champ.focus();
                    /* Valeur de retour */
                    return false;
                }
            }
            /* Valeur de retour */
```

```

        alert("Votre nom est "
        + document.getElementById('nom').value);
        return true;
    }

</script>

</head>

<!-- Début section body du script HTML -->
<body>

    <!-- Titre du traitement -->
    <h1>Editions ENI - JavaScript - FORM_01</h1>

    <!-- Début script JavaScript -->
    <script type="text/JavaScript">

        /* Affichage du nom du script */
        alert("FORM_01");

    </script>

    <!-- Formulaire de saisie HTML -->
    <form>
        Votre nom (champ obligatoire) :
        <input
            type='text'
            id='nom'
        />
        <input
            type='button'
            onclick="nonVide(document.getElementById('nom'),
            'Saisie obligatoire')"
            value='Validation saisie'
        />
    </form>

    <!-- Affichage du code source -->
    <br /></br> <br />
    <center>
    <a href="JavaScript:window.location='view-source:
'+window.location">
        Code source
    </a>
    </center>

</body>

</html>

```

Commentaires du script du formulaire HTML

Intéressons-nous dans un premier temps au formulaire HTML lui-même :

```
<!-- Formulaire de saisie HTML -->
<form>
  Votre nom (champ obligatoire) :
  <input
    type='text'
    id='nom'
  />
  <input
    type='button'
    onclick="nonVide(document.getElementById('nom'),
      'Saisie obligatoire')"
    value='Validation saisie'
  />
</form>
```

En HTML un formulaire est délimité par un jeu de balises `<form> ... </form>`. À l'intérieur de ces balises viendront se placer des balises spécifiques correspondant aux contrôles de saisie (widget) que l'on veut faire paraître dans le formulaire :

- Champ de saisie simple
- Champ de saisie multiligne
- Champ caché (saisie de mot de passe)
- Bouton radio
- Bouton checkbox (case à cocher)
- Listes déroulantes
- Sélecteur de fichier
- Boutons de fonction (soumission, annulation...)

Il n'est pas prévu dans le cadre de ce livre portant essentiellement sur le JavaScript de revenir en détail sur les balises de gestion des différents dispositifs de saisie. Vous trouverez une documentation abondante sur Internet et dans de nombreux livres dédiés à HTML.

Par contre, dans les exemples qui vont suivre, la balise HTML `<input>` mise en œuvre sera commentée a minima.

Dans le script précédent, une première balise `<input>` de type `text` affiche dans le formulaire un emplacement rectangulaire dans lequel l'opérateur pourra effectuer sa saisie :

```
<input
  type='text'
  id='nom'
/>
```

La valeur `text` de l'attribut `type` sert à préciser qu'il s'agit bien d'une saisie de texte qui sera envisagée. Dans ce champ de saisie, vous pourrez indifféremment saisir des données textuelles, numériques ou encore alphanumériques. Le contrôle de la nature de la saisie est justement du ressort d'une séquence de code JavaScript.

L'attribut `id` est une information qui permettra d'identifier ce champ de saisie dans le code JavaScript. Le nom est choisi librement par le programmeur mais doit être dans la mesure du possible significatif. La définition de la zone de texte se termine par un `</>` qui est un abrégé de `</input>`. Rappelons à l'occasion que les balises HTML vont très souvent par paires, la balise ouvrante étant orthographiée génériquement `<balise>` et la balise fermante `</balise>`.

En ce qui concerne les conventions d'écriture au niveau du HTML, sachez qu'il n'y a pas de sensibilité à la casse (vous pouvez orthographier vos balises en MAJUSCULES, en minuscules). Dans ce livre, le choix arbitraire a été aussi de présenter des codes HTML en minuscules.

Le second contrôle de saisie est un bouton dont le rôle va justement être de déclencher la vérification de ce qui a été saisi dans la zone de texte identifiée par l'id `nom` :

```
<input
  type='button'
  onclick="nonVide(document.getElementById('nom'),
    'Saisie obligatoire')"
  value='Validation saisie'
/>
```

Le type du contrôle est `button`.

L'attribut `onclick` est un déclencheur qui provoquera l'exécution du code intégré dans les quotes qui suivent (`"nonVide(document.getElementById('nom'), 'Saisie obligatoire')"`).

L'attribut `onclick` précise que la fonction JavaScript `nonVide` (que nous étudierons plus loin) sera déclenchée lors d'un clic sur le bouton. Il est à noter que dans notre exemple le bouton n'a pas été nommé (il aurait pu l'être via l'attribut `name`) car dans notre cas ce bouton est le seul présent dans notre formulaire.

La fonction `nonVide` appelée par le clic sur le bouton comporte deux paramètres :

- `document.getElementById('nom')` qui signale le champ qui va être testé par la fonction,
- un message (`'Saisie obligatoire'`) qui serait affiché par la fonction dans le cas où il n'y a pas eu saisie dans la zone de texte identifiée par l'id `nom`.



Vous avez aussi remarqué les contraintes d'utilisation des guillemets et des quotes dans la formulation de l'opération déclenchée dans le `onclick`.

Commentaires du script de la fonction JavaScript `nonVide`

Passons maintenant au décryptage de la fonction JavaScript `nonVide`. Le choix a été fait de la positionner dans le script HTML/JavaScript dans la section `<head> ... </head>`. Elle aurait pu aussi être placée dans la section `<body> ... </body>`, ce qui a été souvent le cas dans les exemples du début de ce livre. Enfin les fonctions JavaScript peuvent être placées dans un fichier séparé (appelé) depuis le script en cours.

La première ligne de la fonction mentionne le nom de la fonction (`nonVide`) ainsi que le nom des deux paramètres formels alimentés par `document.getElementById('nom')` et par `'Saisie obligatoire'` :

```
function nonVide(champ, messageAlerte)
```

Ensuite la longueur du texte saisi dans la zone de texte nom est évaluée par la fonction `length` :

```
if (champ.value.length == 0)
```

Le paramètre formel `champ` correspond bien évidemment à la zone de texte et `value` est la propriété représentant le contenu de cette zone.

Si la longueur est nulle, une alerte est affichée ('Saisie obligatoire') et le focus est placé sur le champ de saisie pour que la saisie puisse être effectuée (méthode `focus()` appliquée à `champ`). Dans le cas contraire un rappel du nom est proposé ('Votre nom est ...').

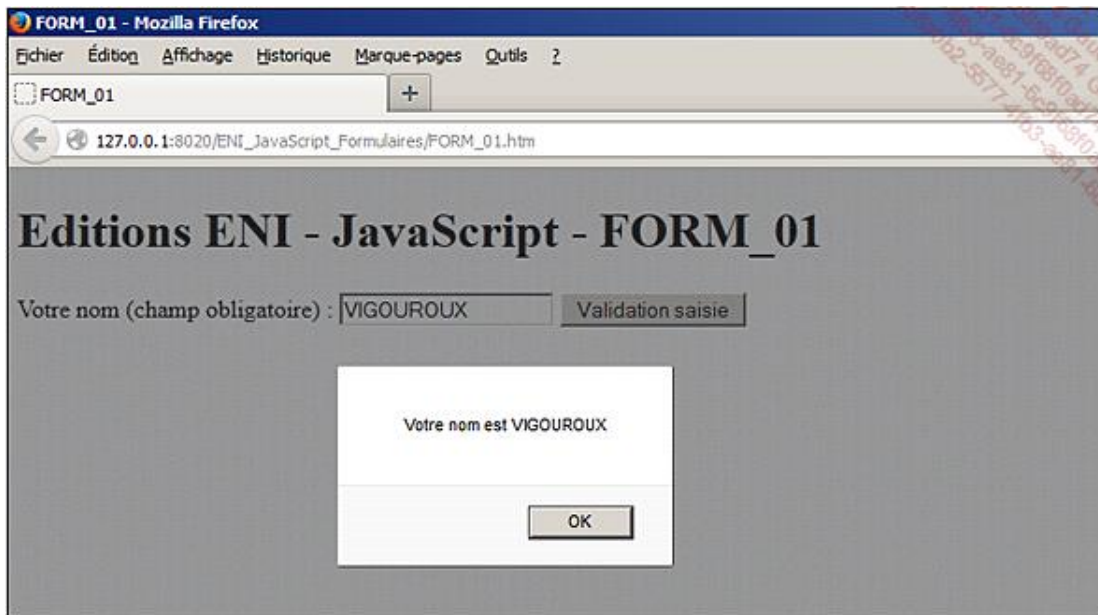
Pour que l'exécution de la fonction soit interrompue, il faut impérativement prévoir un `return` dans chacun des cas. Peu importe dans notre script la valeur rendue (`true`, `false`...), nous n'avons pas prévu de tester la valeur retournée ultérieurement.

Compte rendu d'exécution

L'exécution de ce script donne ceci dans le cas d'une validation sans saisie dans le champ texte nom :



et ceci dans le cas inverse (message de confirmation de ce qui a été saisi) :



2. Contrôle de numéricité d'une saisie dans un champ texte

L'objectif est un peu différent de celui recherché dans l'exemple précédent. Ici un contrôle de numéricité sera appliqué sur la saisie effectuée dans une zone de texte.

Dans la mesure où le traitement présente de grandes similitudes avec le script précédent, seule une partie du script sera présentée et commentée.

Présentation du script du formulaire HTML

Le code du formulaire est le suivant :

```
<!-- Formulaire de saisie HTML -->
<form>
  Montant (saisie d'un nombre entier SVP) :
  <input
    type='text'
    id='nombre'
  />
  <input
    type='button'
    onclick="estNumerique(document.getElementById('nombre'),
      'Saisie nombre entier SVP')"
    value='Validation saisie'
  />
</form>
```

Le champ de saisie de type `text` n'est guère différent du précédent, sa dénomination est `nombre`.

Pour le bouton de déclenchement du contrôle de numéricité, une fonction `estNumerique` est prévue avec deux paramètres :

- `document.getElementById('nombre')` qui signale le champ qui va être testé par la fonction,
- un message ('Saisie d'un nombre entier SVP') qui serait affiché par la fonction dans le cas où la saisie ne serait pas celle d'un nombre entier.

Présentation (partielle) du script de la fonction JavaScript estNumerique

Dans la fonction `estNumerique`, positionnée dans la section `<head> ... </head>`, un test est effectué sur la saisie avec la séquence de code suivante :

```
/* Définition des valeurs acceptées */
var valeursAcceptees = '/^[0-9]+$/' ;
/* Test de numéricité */
if (champ.value.match(valeursAcceptees))
{
    /* Valeur de retour */
    alert("Vous avez saisi " + document.getElementById('nombre').value);
    return true;
}
else
{
    /* Affichage d'un message d'alerte */
    alert(messageAlerte);
    /* Focus sur le champ en erreur */
    champ.focus();
    /* Valeur de retour */
    return false;
}
```

Une variable `valeursAcceptees` est déclarée et initialisée avec une expression régulière contenant les caractères acceptés dans le cadre d'une saisie d'un nombre entier.

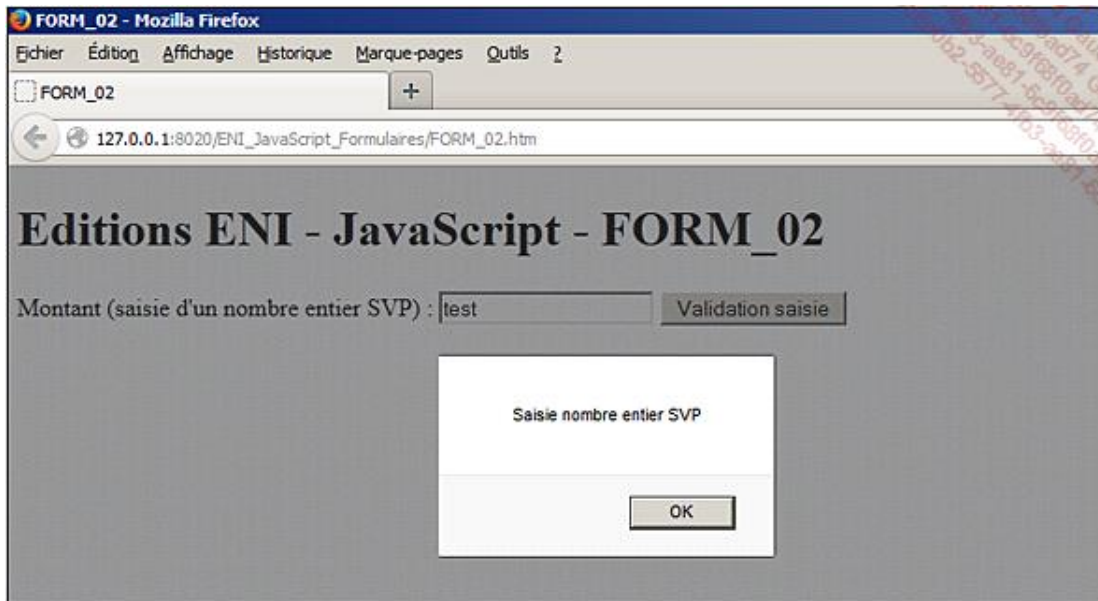
Il reste ensuite à tester le contenu (propriété `value`) du champ par rapport à cette liste de signes acceptés (chiffres de 0 à 9) pour annoncer une erreur de saisie ou non (affichage d'une confirmation de la valeur saisie dans ce cas).

Dans le cadre de ce livre, il n'est pas prévu de vous présenter de manière exhaustive les nombreuses possibilités offertes par les expressions régulières. Vous pourrez consulter de nombreuses ressources sur Internet qui concernent ce sujet, notamment :

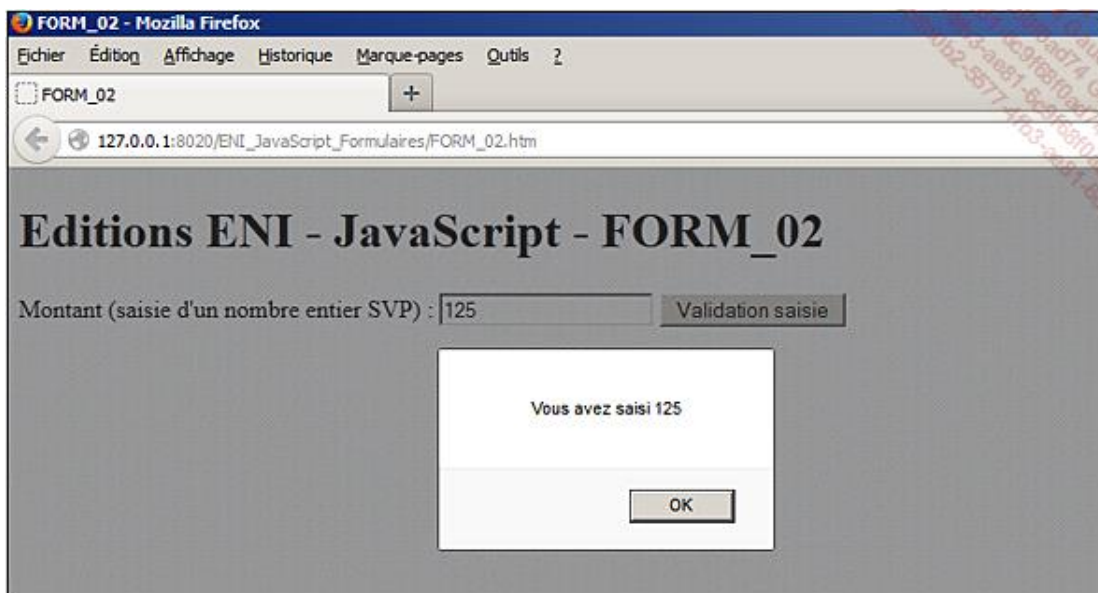
- <http://www.commentcamarche.net/contents/585-JavaScript-l-objet-regexp>
- <http://jm.davalan.org/lang/jsc/js08.html>
- <http://www.siteduzero.com/informatique/tutoriels/dynamisez-vos-sites-web-avec-javascript/les-expressions-regulieres-1-2>
- <http://magali.contensin.online.fr/html/JAVASCRIPT/cours/PREDEF/ExpressionsReg.html>
- http://www.w3schools.com/jsref/jsref_obj_regexp.asp

Compte rendu d'exécution

Si une valeur non entière est saisie, le message suivant s'affiche :



sinon en cas de saisie correcte, nous obtenons :



3. Contrôle de caractères alphabétiques d'une saisie dans un champ texte

Vous l'aurez compris, la problématique est proche de celle vue dans le précédent exemple.

La différence va se jouer au niveau de l'expression régulière (la variable `valeursAcceptees`) :

```
/* Définition des valeurs acceptées */  
var valeursAcceptees = /^[a-zA-Z]+$/;
```

Les valeurs acceptées seront uniquement les lettres de l'alphabet indifféremment en majuscules ou en minuscules.

4. Contrôle de caractères alphabétiques et numériques d'une saisie dans un champ texte

Ici pour contrôler que l'on puisse indifféremment saisir des chiffres entiers ou des chaînes alphabétiques (minuscules ou majuscules), l'expression régulière sera modifiée comme suit :

```
/* Définition des valeurs acceptées */  
var valeursAcceptees = /^[0-9a-zA-Z]+$;/
```

Bien évidemment les autres signes comme les caractères accentués par exemple seront rejetés par la fonction de test.

5. Contrôle de longueur d'une saisie dans un champ texte

Cette problématique est fréquente, notamment quand il s'agit de forcer une saisie d'un mot de passe d'une taille minimale et/ou d'une taille maximale.

Le code du formulaire HTML se présente comme suit :

```
<!-- Formulaire de saisie HTML -->  
<form>  
  Champ (entre 5 et 10 caractères impérativement) :  
  <input  
    type='text'  
    id='saisie'  
  />  
  <input  
    type='button'  
    onclick="longueurSaisie(document.getElementById('saisie'),  
    5, 10, 'Saisie entre 5 et 10 caractères SVP')"  
    value='Validation saisie'  
  />  
</form>
```

La fonction `longueurSaisie` comporte quatre paramètres :

- `document.getElementById('saisie')` où `saisie` est le nom du champ texte,
- 5 la valeur minimale de la longueur acceptée pour la saisie,
- 10 la valeur maximale de la longueur acceptée pour la saisie,
- `'Saisie entre 5 et 10 caractères SVP'`, le message d'erreur à afficher le cas échéant.

Au niveau de la fonction JavaScript `longueurSaisie`, le code est :

```
/* Test de la longueur de la saisie */  
if (champ.value.length >= longueurMinimale && champ.value.length <=  
longueurMaximale)
```

6. Contrôle de saisie sur une adresse e-mail

Une fois de plus, il s'agit de la mise en œuvre d'un contrôle basé sur une expression régulière pour vérifier la structure de l'adresse proposée par l'opérateur. Bien évidemment il n'y aura pas de vérification d'une adresse e-mail réellement associée à un utilisateur.

Le test est fait par rapport à l'expression régulière suivante :

```
/* Définition des valeurs acceptées */
var valeursAcceptees = /^[w\-\.\.]+\@[a-zA-Z0-9\.\-]+\.[a-zA-Z0-9]{2,4}$/;
```

Le contrôle lui-même s'effectue comme suit avec l'utilisation de la méthode `match` :

```
/* Test de la structure de l'adresse e-mail */
if (email.value.match(valeursAcceptees))
{
    /* Valeur de retour */
    alert("Structure d'adresse e-mail correcte");
    return true;
}
else
{
    /* Affichage d'un message d'alerte */
    alert(messageAlerte);
    /* Focus sur le champ en erreur */
    email.focus();
    /* Valeur de retour */
    return false;
}
```

7. Contrôle d'un choix dans une liste déroulante (version simplifiée)

Abandonnons maintenant les champs de type `text`, pour lesquels les contrôles via des expressions régulières peuvent être multiples, pour aborder les listes déroulantes (ou listes de sélection).

Présentation du script du formulaire HTML

Le code du formulaire HTML se présente comme suit :

```
<!-- Formulaire de saisie HTML -->
<form>
    Sport pratiqué :
    <select id='liste'>
        <option>Votre sport favori</option>
        <option>Golf</option>
        <option>Football</option>
```

```

        <option>Natation</option>
        <option>Tennis</option>
    </select>
    <input
        type='button'
        onclick="controlerChoixListe(document.getElementById('liste'),
        'Merci de choisir un sport dans la liste')"
        value='Validation saisie'
    />
</form>

```

Un jeu de balises `<select> ... </select>` entoure les options proposées en choix à l'utilisateur. La liste déroulante, identifiée par l'id `liste`, est passée en paramètre à la fonction `controlerChoixListe` sur un clic sur le bouton habituel de soumission.

Présentation du script JavaScript de la fonction `controlerChoixListe`

Le code de la fonction `controlerChoixListe` est répercuté cette fois totalement :

```

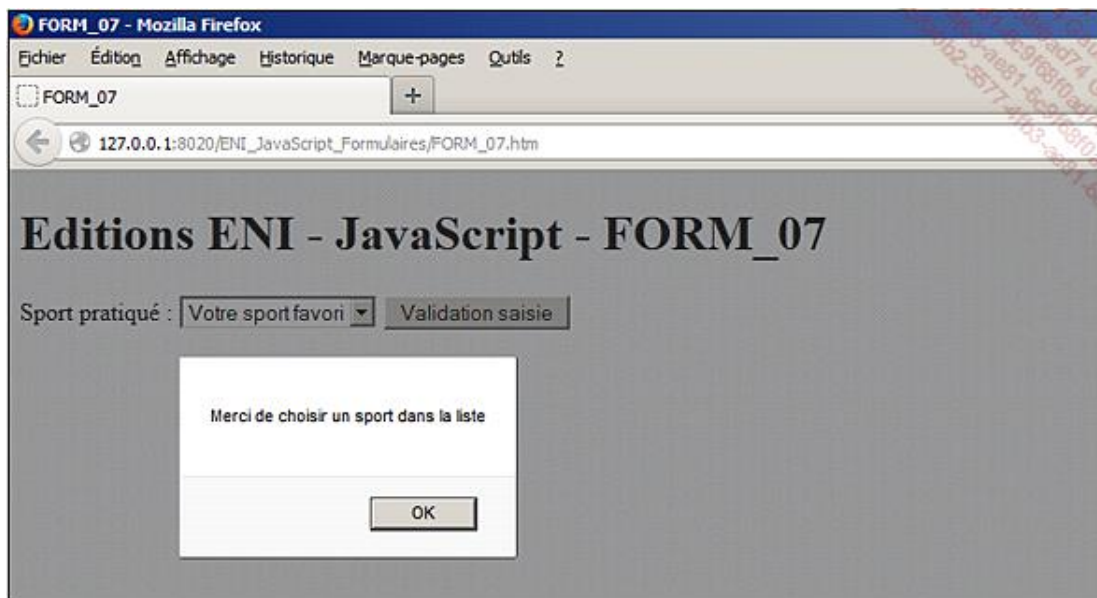
/* Fonction testant si un choix a été fait dans une liste déroulante */
function controlerChoixListe(liste, messageAlerte)
{
    if (liste.value == "Votre sport favori")
    {
        /* Affichage d'un message d'alerte */
        alert(messageAlerte);
        /* Focus sur le champ en erreur */
        liste.focus();
        /* Valeur de retour */
        return false;
    }
    else
    {
        /* Valeur de retour */
        alert("Votre sport préféré : " + liste.value);
        return true;
    }
}

```

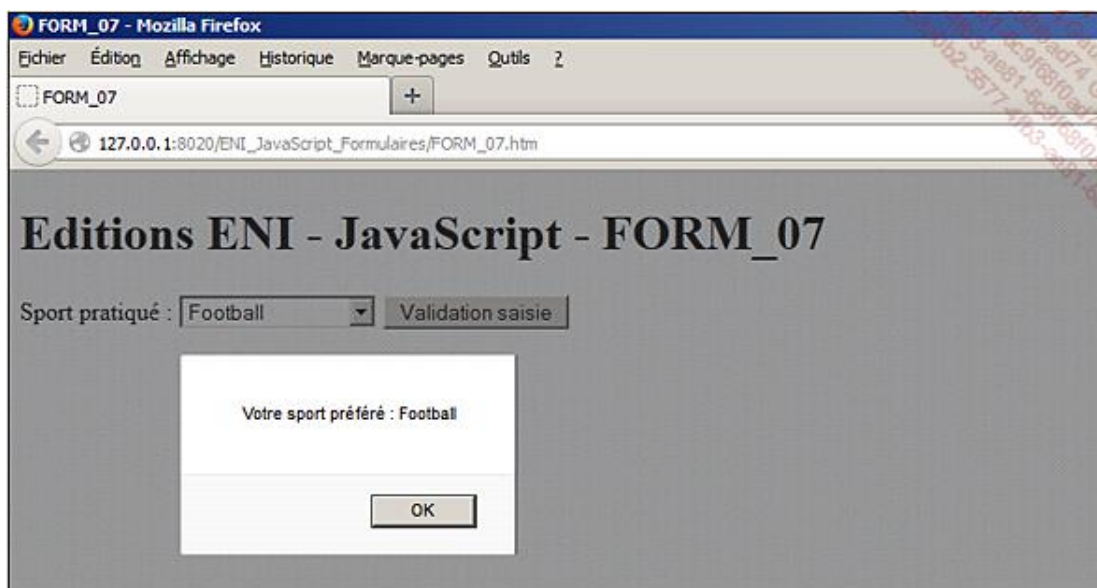
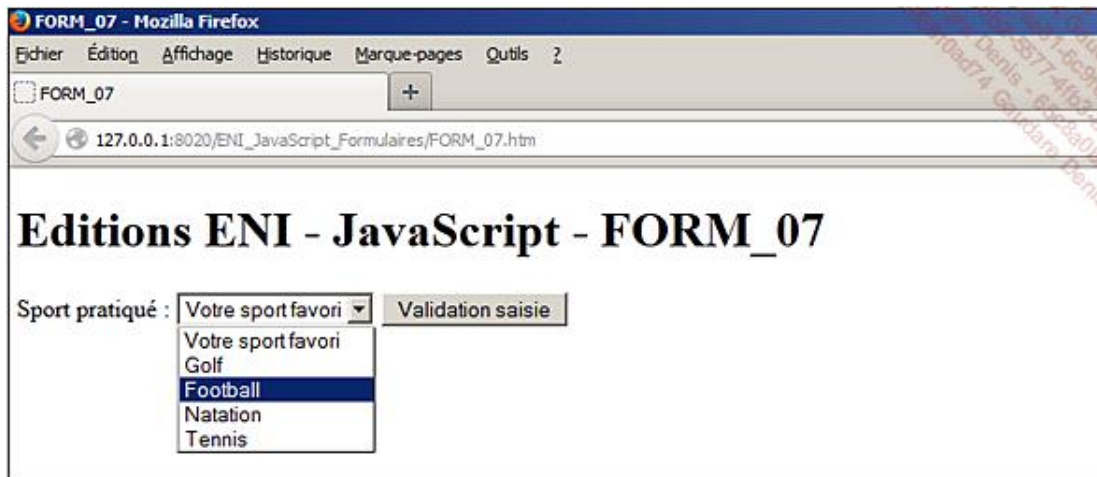
Le code de cette fonction ne présente pas de difficulté particulière. La valeur associée au choix effectué est testée en début de traitement. Si le choix est toujours la valeur initiale (`Votre sport favori`) le message d'alerte habituel est affiché. Dans le cas contraire, le libellé retenu est présenté.

Compte rendu d'exécution

Le résultat obtenu à l'exécution est le suivant si l'on clique sur le bouton **Validation saisie** sans avoir fait un choix dans la liste déroulante :



Si un choix est effectué, l'affichage se présente comme suit :



8. Contrôle d'un choix dans une liste déroulante (version étendue)

Dans l'exemple précédent, pour la liste (`<select> ... </select>`) seul l'attribut de base `id` a été inclus. D'autres attributs sont disponibles :

- `<option selected>`option par défaut`</option>` pour préciser qu'une option particulière sera sélectionnée par défaut.
- `<select name="nomListe" size="5">` pour préciser que cinq options seront visibles dans la liste, les autres nécessitant le déroulement de l'ascenseur.
- `<select name="nomListe" multiple>` pour spécifier que plusieurs options peuvent être sélectionnées simultanément dans la liste.

La balise `<optgroup> ... </optgroup>` permet aussi de subdiviser une liste déroulante en niveaux hiérarchiques.

Présentation du script du formulaire HTML

```
<!-- Formulaire de saisie HTML -->
<form name='formulaire'>
  Sport pratiqué :
  <select id='liste' size='7' multiple>
    <optgroup label="Sports collectifs">
      <option value="football" selected>Football</option>
      <option value="volley">Volley Ball</option>
    </optgroup>
    <optgroup label="Sports individuels">
      <option value="golf">Golf</option>
      <option value="natation" selected>Natation</option>
      <option value="tennis">Tennis</option>
    </optgroup>
  </select>
  <input
    type='button'
    onclick="controlerChoixListe(document.getElementById('liste'))"
    value='Validation saisie'
  />
</form>
```

La liste déroulante est subdivisée en deux sous-groupes ayant pour labels 'Sports collectifs' et 'Sports individuels'. La liste sera totalement déroulée dans la mesure où il est prévu sept lignes d'affichage (deux lignes d'intitulés de sous-groupes et cinq sports) par l'intégration de l'attribut `size=7`. Dans chacun des sous-groupes, un sport est présélectionné (attribut `selected`) et la saisie de plusieurs sports simultanément est possible (attribut `multiple`).

La fonction appelée sur un clic sur le bouton de soumission se nomme `controlerChoixListe` et possède un seul paramètre, l'identifiant de la liste.

Présentation du script JavaScript de la fonction controlerChoixListe

Le code source de la fonction `controlerChoixListe` est présenté intégralement ci-après :

```

/*
Fonction affichant les choix (multiples) effectués
dans une liste déroulante
*/
function controlerChoixListe(liste)
{
    for (i=0; i<document.forms.formulaire.liste.options.length; i++)
    {
        if (document.forms.formulaire.liste.options[i].selected)
        {
            alert("Sport sélectionné : "
                + document.forms.formulaire.liste.options[i].text)
        }
    }
    // Valeur de retour
    return true
}

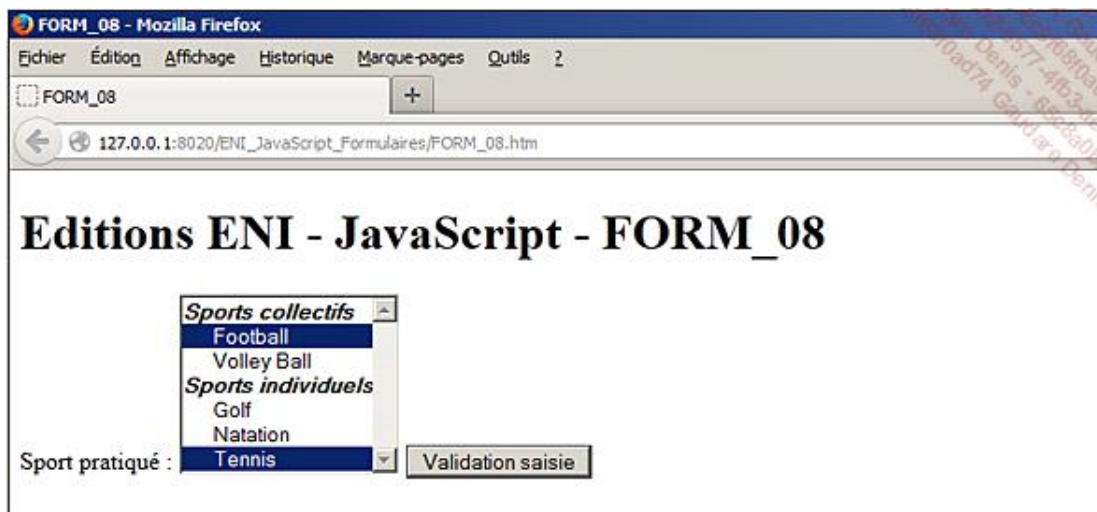
```

Une boucle for balaie les cinq sports proposés dans la liste déroulante. La séquence `document.forms.formulaire.liste.options.length` détermine justement le nombre d'options présentes dans la liste déroulante de nom `liste` du formulaire de nom `formulaire` (cf. `<form name='formulaire'>` dans la description du formulaire en HTML).

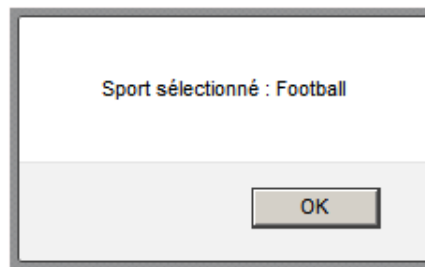
Ensuite `options[i].selected` permet de vérifier que l'option a été retenue ou pas par l'opérateur et si c'est le cas d'afficher le libellé du sport concerné (attribut `text`).

Compte rendu d'exécution

Le résultat obtenu à l'exécution est le suivant :

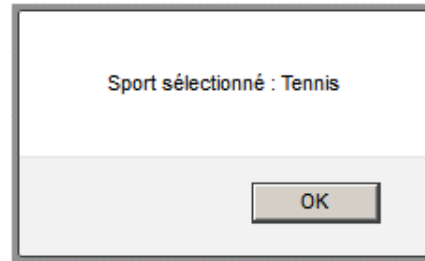


Les deux confirmations suivantes apparaissent une fois la validation faite par le bouton **Validation saisie** :



Sport sélectionné : Football

OK



Sport sélectionné : Tennis

OK

9. Contrôle d'un choix par bouton radio

Proposer un choix entre plusieurs options au travers d'un jeu de boutons radio est une solution intéressante. Cela constitue une alternative à la sélection simple dans une liste déroulante dans la mesure où la sélection dans un jeu de boutons radio ne permet pas la multisélection.

Présentation du script du formulaire HTML

```
<!-- Formulaire de saisie HTML -->
<form name="liste_sports">
  Sport préféré :<br />
  <input
    type="radio"
    name="sports"
    value="golf">
    Golf
  </input><br />
  <input
    type="radio"
    name="sports"
    value="football">
    Football
  </input><br />
  <input
    type="radio"
    name="sports"
    value="natation">
    Natation
  </input><br />
  <input
    type="radio"
    name="sports"
    value="tennis">
    Tennis
```

```

    </input><br />
    <input
      type='button'
      onclick="selectionnerSport
        ('Veuillez sélectionner votre sport préféré')"
      value='Validation saisie'
    />
  </form>

```

Une fois de plus notre formulaire doit être nommé au niveau de la balise `form` :

```
<form name="liste_sports">
```

Après, un intitulé commente le jeu de boutons :

```
Sport préféré :<br />
```

et la liste des boutons radio est proposée :

```

<input
  type="radio"
  name="sports"
  value="golf">
  Golf
</input><br />

```

Le type pour les balises `input` est `radio` et un nom commun (attribut `name`) est précisé (`sports` dans notre cas). L'attribut `value` est renseigné par une valeur différente pour chaque bouton. Enfin, un commentaire (Golf dans notre exemple) termine la syntaxe.

Pour la fonction `selectionnerSport` associée à l'événement `onclick` sur le bouton de soumission, seul un paramètre est prévu, un message d'alerte qui sera affiché en cas de non-sélection du moindre bouton radio :

```

<input
  type='button'
  onclick="selectionnerSport
    ('Veuillez sélectionner votre sport préféré')"
  value='Validation saisie'
/>

```

Présentation du script JavaScript de la fonction `selectionnerSport`

Le script JavaScript, relativement long, de la fonction `selectionnerSport` est présenté dans son intégralité ci-après :

```
/*
```



```

Fonction de test de la valeur sélectionnée
dans un jeu de boutons radio
*/
function selectionnerSport(messageAlerte)
{
    /*
    Récupération des sports proposés
    dans le formulaire dans un tableau
    */
    var tabSports = document.forms.liste_sports.sports;
    /* Détermination du nombre de sports proposés */
    var nombreSports = tabSports.length;
    // alert("Nombre de sports : " + nombreSports);
    if (nombreSports>0)
    {
        /* Parcours du tableau des sports proposés */
        var i=0;
        var sportSelection = false;
        while (i < nombreSports)
        {
            /* Test si le sport est celui qui a été sélectionné */
            if (tabSports[i].checked)
            {
                /* Affichage de contrôle */
                alert("Votre sport favori est " + tabSports[i].value);
                /* Modification du booléen sportSelection à true */
                sportSelection = true;
            }
            i = i +1;
        }
    }
    /* Message d'alerte si aucun sport n'a été sélectionné */
    if (!sportSelection)
    {
        /* Message d'alerte */
        alert(messageAlerte);
        /* Valeur de retour */
        return false;
    }
    else
    {
        /* Valeur de retour */
        return true;
    }
}

```

Le traitement commence par la récupération dans un tableau de la liste des boutons radio intégrés dans le formulaire HTML :

```

/* Récupération des sports proposés dans le formulaire dans un tableau */
var tabSports = document.forms.liste_sports.sports;

```

La syntaxe `liste_sports.sports` précise que l'on fait référence à l'ensemble des boutons radio nommés `sports` du formulaire de nom `liste_sports`.

Le nombre de sports proposés dans le formulaire HTML est déterminé ensuite :

```
/* Détermination du nombre de sports proposés */  
var nombreSports = tabSports.length;
```

Enfin une boucle `while` balaie les différents boutons radio afin de déterminer s'ils ont été sélectionnés ou non (un seul l'a normalement été). Vous noterez que le cas d'une non-sélection a aussi été prévu avec un contrôle par une structure conditionnelle.

Si le bouton radio a été sélectionné :

```
if (tabSports[i].checked)
```

alors son intitulé (attribut `value`) est affiché :

```
alert("Votre sport favori est " + tabSports[i].value);
```

Un dernier mécanisme basé sur une variable booléenne `sportSelection` permet aussi d'afficher un message d'alerte (la variable `messageAlerte` passée en paramètre à la fonction) si aucun bouton radio n'a été sélectionné.

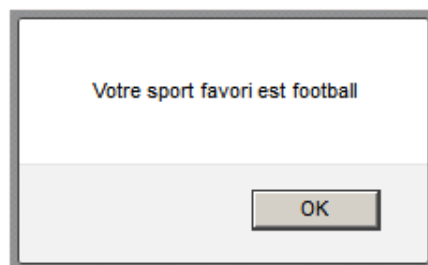
```
/* Message d'alerte si aucun sport n'a été sélectionné */  
if (!sportSelection)  
{  
    // ...  
}  
else  
{  
    // ...  
}
```

Compte rendu d'exécution

Le résultat obtenu à l'exécution est le suivant :



La confirmation suivante apparaît une fois la validation faite par le bouton **Validation saisie** :



10. Contrôle d'un choix par case à cocher

La sélection de choix dans un jeu de cases à cocher présente des similitudes avec la sélection par boutons radio. La principale différence est que la sélection multiple va être possible.

Présentation du script du formulaire HTML

```
<form name="liste_sports">
  Sports préférés :<br />
  <input
    type="checkbox" name="sports[]" value="golf">Golf
  </input><br>
  <input
    type="checkbox" name="sports[]" value="football">Football
  </input><br>
  <input
    type="checkbox" name="sports[]" value="natation">Natation
  </input><br>
  <input
    type="checkbox" name="sports[]" value="tennis">Tennis
  </input><br>
  <input
    type='button'
    onclick="selectionSports
      ('Veuillez sélectionner vos sports préférés')"
```

```
        value='Validation saisie'
    />
</form>
```

Le formulaire, comme pour les boutons radio, est nommé (l'attribut name est fixé à liste_sports). L'attribut type est checkbox (au lieu de radio). L'attribut name est un tableau (sports[]) et chaque input a sa propre valeur de value et a un intitulé différent.

La programmation du bouton de soumission est quasiment identique à celle du paragraphe précédent, une fonction selectionnerSports est appelée avec toujours un message d'alerte en paramètre.

Présentation du script JavaScript de la fonction selectionnerSports

Le code de la fonction est reproduit ici intégralement bien qu'il soit très proche de celui vu dans le cadre du paragraphe précédent.

```
/*
Fonction de test des valeurs sélectionnées
dans un jeu de cases à cocher
*/
function selectionnerSports(messageAlerte)
{
    /*
    Récupération des sports proposés
    dans le formulaire dans un tableau
    */
    var tabSports = document.forms.liste_sports.elements["sports[]"];
    /* Détermination du nombre de sports proposés */
    var nombreSports = tabSports.length;
    // alert("Nombre de sports : " + nombreSports);
    /* Parcours du tableau des sports proposés */
    var i = 0;
    var nombreSportsSelectionnes = 0;
    while (i < nombreSports)
    {
        /* Test si le sport a été coché */
        if (tabSports[i].checked)
        {
            /* Incrémentation du nombre de sports sélectionnés */
            nombreSportsSelectionnes = nombreSportsSelectionnes + 1;
            /* Affichage de de contrôle */
            alert("Votre sport favori est " + tabSports[i].value);
        }
        i = i + 1;
    }
    /* Message d'alerte si aucun sport n'a été sélectionné */
    if (nombreSportsSelectionnes == 0)
    {
        /* Message d'alerte */
        alert(messageAlerte);
        /* Valeur de retour */
        return false;
    }
}
```

```

    }
    else
    {
        /* Valeur de retour */
        return true;
    }
}

```

La seule spécificité est sans doute la récupération des sports affichés dans le formulaire HTML :

```

/* Récupération des sports proposés dans le formulaire dans un tableau */
var tabSports = document.forms.liste_sports.elements["sports[]"];

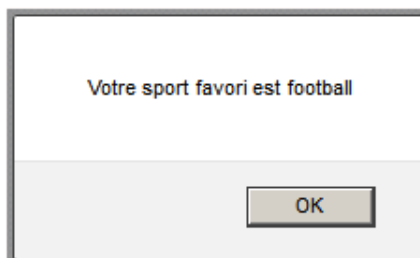
```

Compte rendu d'exécution

Le résultat obtenu à l'exécution est le suivant :



Les confirmations suivantes apparaissent une fois la validation faite par le bouton **Validation saisie** :



et

