Explication des scripts Javascript

Ce rapport a pour but d'expliquer en détail l'ensemble des scripts *Javascript* utilisés sur ma page personnelle.

I. Script pour changer de feuille de style : styleswitcher.js

Ce script permet de changer facilement le design de son site en sélectionnant une autre feuille de style. De plus, il repose sur la création d'un cookie nommé « blogstyle » qui permet de garder en mémoire le choix du style lorsqu'on navigue sur le site et qu'on change de page.

1) La fonction *setActiveStyleSheet* permet de changer la feuille de style active par celle dont le nom est donné en argument : *title*.

```
function setActiveStyleSheet(title) {
  var i, a, main;
  for(i=0; (a = document.getElementsByTagName("link")[i]); i++) {
    if(a.getAttribute("rel").indexOf("style") != -1 &&
a.getAttribute("title")) {
      a.disabled = true;
      if(a.getAttribute("title") == title) a.disabled = false;
    }
  }
}
```

La fonction fait une boucle sur les balises *link* présentes dans la page : si les attributs nécessaires sont présents dans la balise *link* alors on désactive les feuilles de style dont le nom est différent de celui donné en argument de la fonction, et active la feuille de style dont le nom correspond à celui donné en argument.

2) La fonction *getActiveStyleSheet* permet d'obtenir le nom de la feuille de style active.

```
function getActiveStyleSheet() {
  var i, a;
  for(i=0; (a = document.getElementsByTagName("link")[i]); i++) {
    if(a.getAttribute("rel").indexOf("style") != -1 &&
a.getAttribute("title") && !a.disabled) return
a.getAttribute("title");
  }
  return null;
}
```

La fonction fait une boucle sur les balises *link* présentes dans la page : si les attributs nécessaires sont présents dans la balise *link* alors on retourne le nom de la feuille de style qui n'est pas désactivée.

3) La fonction *getPreferredStyleSheet* permet de récupérer le nom de la feuille de style « préférée », c'est-à-dire celle dont l'attribut *rel* vaut « stylesheet » et non « alternate stylesheet ».

```
function getPreferredStyleSheet() {
  var i, a;
  for(i=0; (a = document.getElementsByTagName("link")[i]); i++) {
    if(a.getAttribute("rel").indexOf("style") != -1
       && a.getAttribute("rel").indexOf("alt") == -1
       && a.getAttribute("title")
       ) return a.getAttribute("title");
  }
  return null;
}
```

La fonction fait une boucle sur les balises *link* présentes dans la page : si les attributs nécessaires sont présents dans la balise *link* alors on retourne le nom de la feuille de style dont l'attribut *rel* vaut uniquement « stylesheet ».

4) La fonction *createCookie* permet de créer un cookie.

```
function createCookie(name,value,days) {
  if (days) {
    var date = new Date();
    date.setTime(date.getTime()+(days*24*60*60*1000));
    var expires = "; expires="+date.toGMTString();
  }
  else expires = "";
  document.cookie = name+"="+value+expires+"; path=/";
}
```

Si le nombre de jours de validité du cookie est spécifié à l'appel de la fonction, alors on calcule la durée de validité de ce cookie. Ensuite, on crée le cookie avec le nom spécifié, la valeur donnée et la durée de validité calculée.

5) La fonction *readCookie* permet de lire la valeur d'un cookie.

```
function readCookie(name) {
  var nameEQ = name + "=";
  var ca = document.cookie.split(';');
  for(var i=0;i < ca.length;i++) {
    var c = ca[i];
    while (c.charAt(0)==' ') c = c.substring(1,c.length);
    if (c.indexOf(nameEQ) == 0) return
c.substring(nameEQ.length,c.length);
  }
  return null;
}
```

Cette fonction sépare les différents éléments de la chaîne de caractère du cookie récupéré afin de pouvoir retourner seulement la valeur du cookie.

6) La suite du script permet, au chargement de la page, de lire le style sauvegardé dans le cookie « blogstyle » et de l'appliquer.

```
window.onload = function(e) {
  var cookie = readCookie("blogstyle");
  var title = cookie ? cookie : getPreferredStyleSheet();
  setActiveStyleSheet(title);
}
```

Ce script lit le nom de la feuille de style du style courant, sauvegardé dans le cookie « blogstyle ». Si aucun cookie n'est encore créé, alors on récupère le nom de la feuille de style « par défaut ». Pour finir, on active le style dont le nom est sauvegardé dans le cookie, si le cookie existe sinon on active le style par défaut.

7) La suite du script permet, lors d'un changement de page, de sauvegarder le style courant dans le cookie « blogstyle ».

```
window.onunload = function(e) {
  var title = getActiveStyleSheet();
  createCookie("blogstyle", title, 365);
}
```

On récupère le nom de la feuille de style active et on l'enregistre dans le cookie nommé « blogstyle ».

8) La fin du script permet d'appliquer dynamiquement le changement de style dès lors que le visiteur le demande.

```
var cookie = readCookie("blogstyle");
var title = cookie ? cookie : getPreferredStyleSheet();
setActiveStyleSheet(title);
```

Ce script lit le nom de la feuille de style du style courant, sauvegardé dans le cookie « blogstyle ». Si aucun cookie n'est encore créé, alors on récupère le nom de la feuille de style « par défaut ». Pour finir, on active le style dont le nom est sauvegardé dans le cookie, si le cookie existe sinon on active le style par défaut.

II. Script pour vérifier le formulaire de contact : verif_formulaire.js

Ce script permet la vérification de la cohérence des valeurs rentrées par le visiteur lorsqu'il remplit le formulaire de la page *contact.html*. Cette page est composée des champs : *nom*, *prénom*, *date de naissance*, *email* et *commentaire*.

1) La fonction *verif_nom_prenom* permet de verifier que la valeur donnée en argument correspond à un nom ou à un prénom.

```
function verif_nom_prenom (nom_prenom) {
    /* On crée l'expression régulière correspondante au motif d'un
nom ou d'un prénom :
     * au moins deux caractères alphabétiques insensibles à la case
avec ou sans espace et tiret */
    var regexp = new RegExp("^[a-z \-]{2,}$", "i");

    if (regexp.test (nom_prenom))
        return true;

    return false;
}
```

On crée l'expression régulière qui stipule que le nom et le prénom doivent chacun comportés au moins deux caractères et peuvent contenir des espaces ou des tirets (prise en charge des deux noms de famille et des prénoms composés). On vérifie ensuite que la chaîne de caractères donnée en argument corresponde au motif.

2) La fonction *verif_date_naissance* permet de vérifier que la valeur donnée en argument correspond à une date de naissance de la forme JJ/MM/AAAA.

```
function verif_date_naissance (date_naissance) {
    /* On crée l'expression régulière correspondante au motif d'une
date de naissance */
    var regexp = new RegExp("^(([0-9]{2})/([0-9]{2})/([0-9]{4}))$",
"");

    if (regexp.test (date_naissance)) {
        var nb_jours_mois = [31, 28, 31, 30, 31, 30, 31, 31, 30,
31, 30, 31];
        var decoupage = date_naissance.split (regexp);
        var jours   = parseInt (decoupage[2].charAt(0)) * 10
                    + parseInt (decoupage[2].charAt(1));
        var mois    = parseInt (decoupage[3].charAt(0)) * 10
                    + parseInt (decoupage[3].charAt(1));
        var annees  = parseInt (decoupage[4]);

        /* Si l'année est bissextile */
        if ((annees % 4 == 0 && annees % 100 != 0) || annees % 400
== 0)
            nb_jours_mois[1] = 29;

        /* Si la date de naissance est cohérente */
        if  (mois <= 12 && jours <= nb_jours_mois[mois-1] && annees
>= 1900) {
            var now = new Date ();

            /* Si la date de naissance entrée est ultérieure à la
date actuelle */
            if  (annees < now.getFullYear()
            || (annees == now.getFullYear() && mois <
now.getMonth()+1)
            || (annees == now.getFullYear() && mois ==
now.getMonth()+1
                && jours <= now.getDate()))
                return true;
        }
    }

    return false;
}
```

On crée d'abord l'expression régulière qui permet de vérifier que la chaîne de caractères entrée en argument est bien de la forme JJ/MM/AAAA (deux chiffres, slash, deux chiffres, slash, quatre chiffres). Si la chaîne de caractères est de cette forme, alors on découpe la chaîne de caractères afin de récupérer séparément la valeur du jour, du mois et de l'année de naissance. On convertit à l'aide la fonction *parseInt* les caractères en chiffres afin de pouvoir tester leur cohérence.

On vérifie alors que la valeur du mois n'excède pas 12 (12 mois dans une année), la valeur du jour ne dépasse pas le nombre de jour du mois considéré (avec prise en charge des années bissextiles) et que l'année est supérieure à 1900 (on considère qu'aucun être humain âgé de plus de 112 ans viendra poster un commentaire).

La dernière condition vérifie que la date de naissance donnée est ultérieure à la date actuelle.

3) La fonction *verif_email* permet de vérifier que la chaîne de caractères donnée en paramètre a bien un format d'adresse email.

```
function verif_email (email) {
    /* On crée l'expression régulière correspondante au motif d'un
l'email */
    var regexp = new RegExp("^[a-zA-Z0-9._-]+@[a-z0-9._-]{2,}\.[a-
z]{2,4}$", "");

    if (regexp.test (email))
        return true;

    return false;
}
```

On crée une expression régulière qui vérifie qu'au moins un caractère alphanumérique (en incluant le point, le tiret et l'underscore) est entré avant le @ puis qu'au moins deux caractères alphanumériques (en incluant le point, le tiret et l'underscore) sont entrés, ensuite qu'un point est présent suivi d'entre deux et quatre caractères alphabétiques minuscules.

4) Finalement, c'est la fonction *verif_formulaire* qui appelle successivement toutes les fonctions citées précédemment afin de vérifier la validité de tous les champs et d'envoyer le cas échant les messages d'erreur correspondants.

```
function verif_formulaire () {
    var msg_erreur;
    var nom = document.getElementById ('nom');
    var prenom = document.getElementById ('prenom');
    var date_naissance = document.getElementById
('date_naissance');
    var email = document.getElementById ('email');
    var commentaire = document.getElementById ('commentaire');

    /* Si les champs existent tous */
    if (nom == null || prenom == null || date_naissance == null ||
email == null || commentaire == null)
        return false;

    nom = nom.value;
    prenom = prenom.value;
    date_naissance = date_naissance.value;
    email = email.value;
    commentaire = commentaire.value;

    /* Si tous les champs sont remplis */
    if (nom.length > 0 && prenom.length > 0 &&
date_naissance.length > 0 && email.length > 0 && commentaire.length
> 0) {
        /* Si le champs nom est correct */
        if (verif_nom_prenom (nom)) {
            /* Si le champs prenom est correct */
            if (verif_nom_prenom (prenom)) {
                /* Si le champs date_de_naissance est correct */
                if (verif_date_naissance (date_naissance)) {
                    /* Si le champs email est correct */
                    if (verif_email (email)) {
                        alert ('Message correctement envoyé');
```

```
                            return true;
                        }
                        else msg_erreur = "Email invalide";
                    }
                    else msg_erreur = "Date de naissance invalide";
                }
                else msg_erreur = "Prénom invalide";
            }
            else msg_erreur = "Nom invalide";
        }
        else msg_erreur = "Veuillez remplir tous les champs";

        alert (msg_erreur);

        return false;
    }
```

On récupère tous les champs par le biais de la fonction *getElementById*, puis on récupère leur valeur en prenant d'abord le soin de vérifier que ces id ont correctement été définis dans la page HTML qui appelle cette fonction. On vérifie que les champs sont bien tous remplis puis on vérifie successivement le nom, le prénom, la date de naissance puis l'email. Si une erreur est détectée, une fenêtre d'alerte est alors transmise au visiteur pour lui spécifier le champ invalide, sinon on lui affiche que le message a correctement été envoyé.