INTRODUCTION

Cet exercice consiste à créer un formulaire pour l'entrée de données personnelles. La particularité de ce formulaire est qu'après chaque entrée de l'utilisateur, un programme vérifie que l'information entrée est plausible. Pour réaliser cet exercice, vous devrez reproduire le plus fidèlement possible le formulaire de la dernière page et créer le code JavaScript nécessaire pour valider les champs de saisie à l'aide d'expressions rationnelles. La présentation du formulaire doit être assurée par une feuille de style CSS.

COMPÉTENCES NÉCESSAIRES

- Connaître les bases du langage JavaScript;
- Connaître les bases des expressions rationnelles;
- Connaître les balises HTML pour la création de formulaires et de contrôles.

COMPÉTENCES VISÉES

- Après cet exercice, vous serez capable de:
    - Utiliser le langage JavaScript pour valider des contrôles à l'aide d'expressions rationnelles;
    - Respecter les normes de codage imposé dans le cours pour le code JavaScript.

SOURCES

- Le fichier compressé « 10.10_Exercice_07_JsValidationForm_RegExp.zip », disponible sur LÉA, comprennent les fichiers suivants :
    - Le présent document;
    - Une capture d'écran du formulaire « 10.11_Exercice_07_CaptureEcran.gif »;
    - L'image d'arrière-plan « 10.12_exercice_07_granit.gif »;
    - Le PowerPoint du cours "10.20_PPT_JavaScript_ExpressionsRatio.pdf".

PRODUCTION

1. Créez un nouveau document nommé « formulaire_valide_expr_ratio.html ».
2. Conseil : Créez un tableau et insérez les éléments du formulaire dans le tableau. Assurez la présentation avant de commencer à créer le code de validation.

3. Note : Le tableau et le formulaire ne sont pas générés avec du code JavaScript, ils sont codés directement dans la partie <body> du document (n'oubliez pas de respecter les normes de codage XHTML du cours).
4. Créer une fonction « Valider » qui recevra en paramètre une référence sur l'objet à valider. Votre code devra à l'aide de ce paramètre déterminer la validation à effectuer. Dans le cas où le contenu saisi par l'utilisateur est incorrect, votre code devra afficher un message à l'utilisateur pour indiquer l'erreur et le type de contenu attendu avec les exemples nécessaires. Lorsque l'utilisateur fermera cette fenêtre d'information, le contenu du champ en faute devra être sélectionné. Le contrôle en faute doit rester sélectionné tant et aussi longtemps que le contenu ne sera pas valide. Vous devez effectuer la validation après la saisie de contenu par l'utilisateur. Si un utilisateur donne l'attention (focus) à un contrôle, mais ne fait aucune modification, votre code de validation ne doit pas être appelé. (Note : commencer par vous assurer de valider correctement les valeurs par l'entremise des expressions rationnelles avant de vous attaquer à la question de sélectionner le contrôle en faute).
5. Créer une fonction nommée « FormulaireComplet » qui vérifiera que tous les contrôles ont une valeur lorsque l'utilisateur cliquera sur le bouton « Soumettre ». Dans le cas contraire, votre code devra indiquer à l'utilisateur que le formulaire est incomplet et lister les champs qui doivent obtenir une valeur. Lorsque l'utilisateur fermera cette fenêtre d'information, le curseur devra se retrouver dans le premier champ en faute. (Note : Encore là, la sélection du contrôle en faute n'est pas la priorité).
6. Enfin, il s'agit d'ajouter un peu de finition à cette interface. Pour ce faire, vous devez lier les étiquettes des contrôles aux contrôles de façon à ce qu'un clic sur l'étiquette provoque l'insertion du curseur dans ce contrôle (utilisez la balise HTML dédiée à ce rôle). De plus, lorsque le pointeur de souris se retrouve au-dessus d'un élément cliquable (bouton et étiquettes liées), le pointeur de la souris doit se transformer en main (utilisez la pseudo classe CSS pour réaliser cette action).

REMISE

- Cet exercice terminé est à présenter à l'enseignant en classe (Date de remise sur LÉA).
- Votre code source doit être parfaitement présenté (règles imposées dans le cours).
- Votre résultat final doit correspondre **exactement** au résultat présenté ci-dessous.

Saisies d'écrans

Tout le texte est de type « Verdana », de taille 14px et de couleur #999999 sauf indication contraire.

Bordure de 4px de large, de couleur #999999, avec un effet en relief.

Couleur d'arrière-plan de la page du navigateur est #AAAAAA

Ce texte a la couleur #DDDDDD

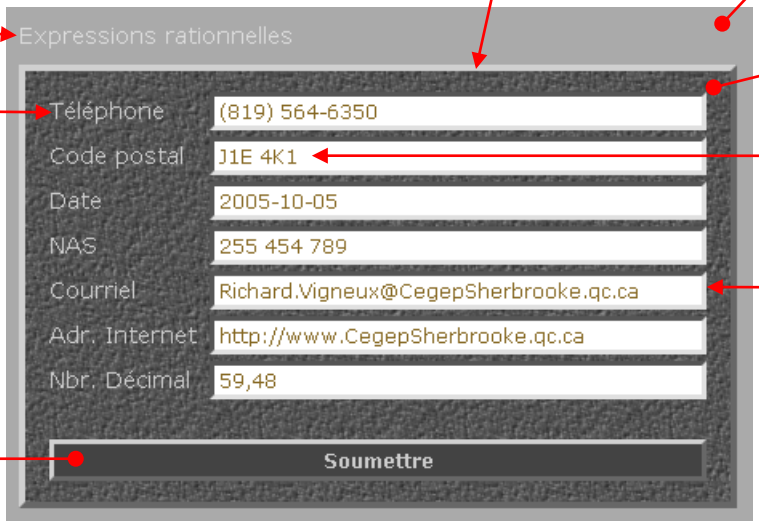
Ce texte a la couleur #CCCCCC

Image de fond granit.gif

Ce texte a la couleur #866822 et une taille de 12 px

Ce bouton affiche le texte à 12px en gras. Il a une bordure de 3px de large, de couleur #999999, avec un effet en relief. Enfin, sa couleur de fond est #434343

Bordure de 3px de large, de couleur #999999, avec un effet incrustation. La largeur de ce contrôle est de 300px.



Formats acceptés

(999) 999-9999 | (999)999-9999 | (999) 999.9999 | (999)999.9999 | 999.999.9999 | 999-999-9999 | 999.9999 | 999-9999

A9A 9A9 | A9A-9A9 | A9A9A9

AAAA-MM-JJ | AAAA/MM/JJ | J MMMM AAAA | JJ MMMM AAAA

999 999 999 | 999-999-999 | 999999999

compte@domaine.ext | compte@[255.255.255.0]

protocole://adresse

Celui-ci est particulier. Vous n'avez pas à valider la valeur. Par contre, lorsqu'un utilisateur entre un nombre avec une virgule, la virgule est échangée par un point. Bien entendu, vous devez effectuer cette détection et l'échange à l'aide des outils et méthodes liés aux expressions rationnelles.