TP Javascript

Sommaire

Les objectifs de ce tp sont d'être capable d'intégrer des scripts clients dans un site ou une page en respectant les bonnes pratiques.

Avec *Mozilla Firefox*, vous pouvez utiliser la **console d'erreurs** ou le *plugin* **Développeur Web** pour vous aider à déboguer vos scripts.

Séquence n°1 : les bases

Insertion de code javascript

Il est possible d'insérer du code *javascript* de plusieurs manières dans une page HTML :

- interne au document en utilisant l'élément `SCRIPT`
- externe au document : le code *javascript* se trouve dans un fichier séparé portant le plus souvent l'extension `.js`. On utilise l'attribut `SRC` de l'élément `SCRIPT` pour préciser le chemin du fichier du script.
- directement dans des éléments HTML en précisant le mot clé `javascript` :

```
<A HREF="javascript:message()">une fonction js</A>
```

Exercice n°1.1 : code javascript interne au document

Question 1. Créer le document à partir du code source fourni ci-dessous. Quelle est l'extension à donner à ce document ?

```
<HTML>
   <HEAD>
      <TITLE>Exercice 1 JavaScript</TITLE>
      <SCRIPT TYPE="text/javascript">
         document.write("<P>Du texte écrit en javascript.</P>");
         alert("Hello world ! en javascript");
      </SCRIPT>
   </HEAD>
   <BODY>
      <P>Du texte écrit en HTML.</P>
   </BODY>
</HTML>
```

Exercice 1

Question 2. Tester dans un navigateur. Pourquoi le texte écrit en HTML ne s'affiche-t-il pas tout de suite ?

Question 3. Comment appelle-t-on le type de boîte de dialogue utilisé dans cet exemple ?

Question 4. Que se passe-t-il si on place le code *javascript* après le texte écrit en HTML ?

Exercice n°1.2 : code javascript externe au document

Question 5. Reprendre l'exemple précédent mais en plaçant le script dans un fichier externe `script1.js`. Tester dans un navigateur.

Exercice n°1.3 : code javascript dans des balises HTML

Question 6. Afficher une boîte de dialogue modale lorsqu'on clique sur un lien. Le texte à afficher sera : « Vous venez de cliquer sur un lien ! ».

Attention, les URLs utilisent un codage : par exemple l'espace sera codé %20 (voir aussi les fonctions `escape()` et `unescape()`).

Exercice n°1.4 : gestionnaire d'évènement en javascript

Question 7. Créer le document à partir du code source fourni ci-dessous.

```
<HTML>
   <HEAD>
      <TITLE>Exercice 4 JavaScript</TITLE>
   </HEAD>
   <BODY onLoad=alert("Bienvenue sur ma page !")>
      <P>Je suis un document HTML.</P>
   </BODY>
</HTML>
```

Exercice 4

Question 8. Quel est le nom de l'évènement géré ici par le code *javascript* ? Tester dans un navigateur. Que se passe-t-il lors du chargement de la page dans un navigateur ?

Question 9. Modifier l'exemple précédent pour qu'une boîte de dialogue modale affiche « À bientôt ! » lorsqu'on quitte la page. Quel est alors le nom de l'évènement à gérer ? Tester dans un navigateur le changement produit.

Exercice n°1.5 : les fonctions

On vous demande de créer un document qui affiche une boîte de dialogue de saisie avec le message suivant : " Entrez une année : ".

Remarque : La méthode `window.prompt(texte, texte_par_défaut)` ouvre une boîte de dialogue dotée d'une zone de saisie et retourne les données saisies par l'utilisateur.

Le script testera ensuite si l'année `yyyy` saisie par l'utilisateur est une année bissextile. Pour cela, vous devez écrire une fonction `isAnneeBissextile()` qui :
– reçoit en argument l'année à tester et
– retourne vrai (`true`) dans le cas où l'année est bissextile sinon faux (`false`).

Cela se termine par l'affichage dans la page HTML du résultat du test : "`yyyy` est une année bissextile" ou "`yyyy` n'est pas une année bissextile".

Question 10. Éditer le document HTML et le script dans un fichier externe `script2.js`. Tester dans un navigateur.

La règle des années bissextiles :
Le pape Grégoire XIII a mis au point en 1582 un calendrier, appelé encore aujourd'hui la calendrier grégorien. Il introduit les règles de calcul des années bissextiles : tous les 4 ans donc chaque année divisible par 4 (comme 1992 ou 1996). Mais ces règles ne suffisent pas. En effet, lorsqu'il s'agit de la première année d'un nouveau siècle (par exemple 2000, 1900 ou 2100), cette année doit être divisible non pas par 4 mais par 400. Ainsi 1700, 1800 et 1900 ne sont pas des années bissextiles.

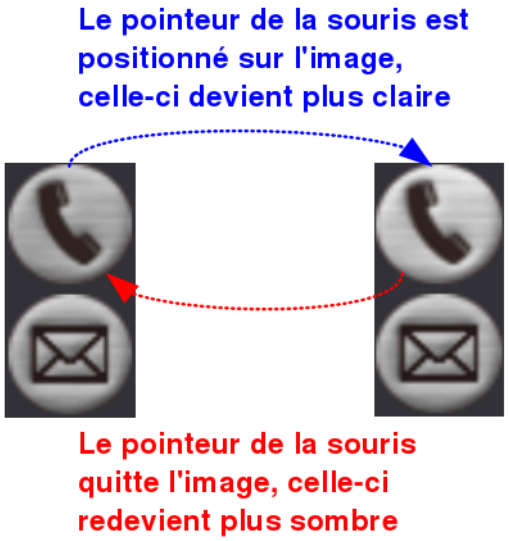
Conclusion

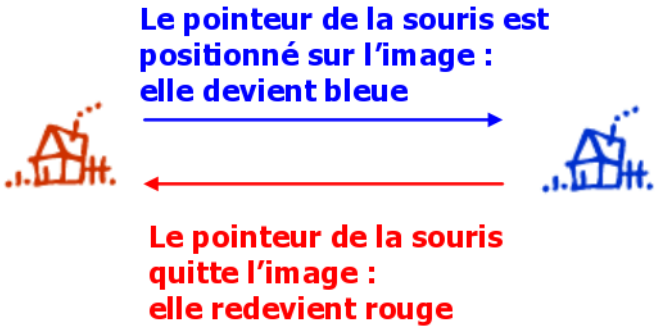
Le code *javascript* s'insère dans un document HTML et s'exécute côté client.

Séquence n°2 : interactivité côté client

Exercice n°2.1 : changement d'image

Placer une image dans un document HTML. Vous pouvez utiliser votre page **contact.html** ou créer un nouveau document pour faire cet exercice.





Il vous faut installer un gestionnaire d'événement (`mouseover`) qui permet de changer l'image lorsque le pointeur de la souris se trouve sur celle-ci. On revient sur l'image d'origine lorsque le pointeur de la souris se déplace hors de l'image (`mouseout`).

Pour rendre cette fonctionnalité ré-utilisable, le gestionnaire d'événement sera réalisé par une seule et unique fonction `changerImage()` qui recevra en arguments l'`id` de l'élément et l'image à afficher.

On recommande d'utiliser la méthode `getElementById()` pour accéder à un élément HTML par son identifiant (attribut `ID` de l'élément HTML en question).

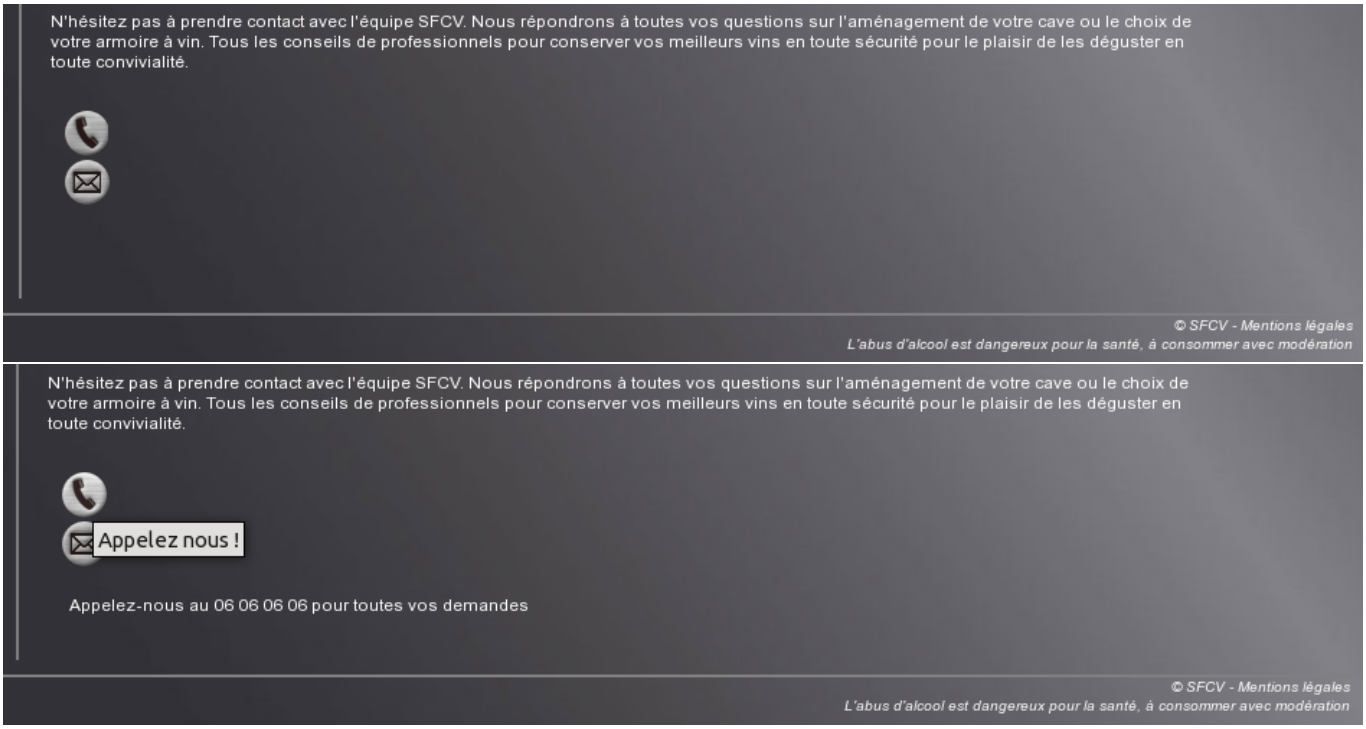
Question 11. Écrire le document HTML et le code *javascript* permettant de réaliser le changement d'image interactif.

Exercice n°2.2 : modification du document

Reprendre l'exercice précédent et permettre l'affichage dans le document HTML des informations associées aux deux images lorsque l'on déplace le pointeur de la souris sur celles-ci :

– sur l'image du téléphone : on affichera dans un paragraphe (`<P>`), situé dessous, le message suivant « Appelez-nous au 06 06 06 06 pour toutes vos demandes. »
– sur l'image du courrier : on affichera dans un paragraphe (`<P>`), situé dessous, le message suivant « Envoyez-nous un courriel à contact[at]sfcv.com, nous vous répondrons immédiatement. »

Cela donnera par exemple pour l'image du téléphone :



Pour insérer dynamiquement le paragraphe dans la document `contact.html`, on prévoira un élément `DIV` en n'oubliant pas de préciser son attribut `id` :

```
<div id="renseignement" style="height:40px;"><p> </p></div>
```

Le gestionnaire d'événements (`mouseover` et `mouseout`) de l'image concernée sera réalisé par une seule et unique fonction `afficherInformation()` qui recevra en arguments l'`id` de l'élément `DIV` et le message à insérer.

En *javascript*, on manipule la propriété `innerHTML` d'un élément pour insérer du code HTML dans celui-ci.

Question 12. Écrire le document HTML et le code *javascript* permettant de réaliser l'affichage dynamique de texte.

Exercice n°2.3 : fenêtre popup

Question 13. Écrire un script `exo-2-3.js` qui permet de créer, au chargement de la page `exo-2-3.html`, une nouvelle fenêtre (`window.open()`) qui se fermera automatiquement (`window.close()`) au bout de 3 secondes.

L'objet `window` possède une fonction `setTimeout()`. Exemple : `setTimeout("toto()", 100)` exécutera la fonction `toto()` à l'expiration du délai (*timeout*) de 100 ms.

Conclusion

Le code *javascript* permet d'améliorer l'interactivité côté client.

Séquence n°3 : vérification de formulaire

L'objectif principal de cette étape est d'assurer l'intégrité des données envoyées par les formulaires vers le serveur. En effet, il serait inutile de surcharger le serveur avec l'envoi de données manquantes, incomplètes ou incorrectes. Cette vérification peut donc se faire côté client par le langage *Javascript*.

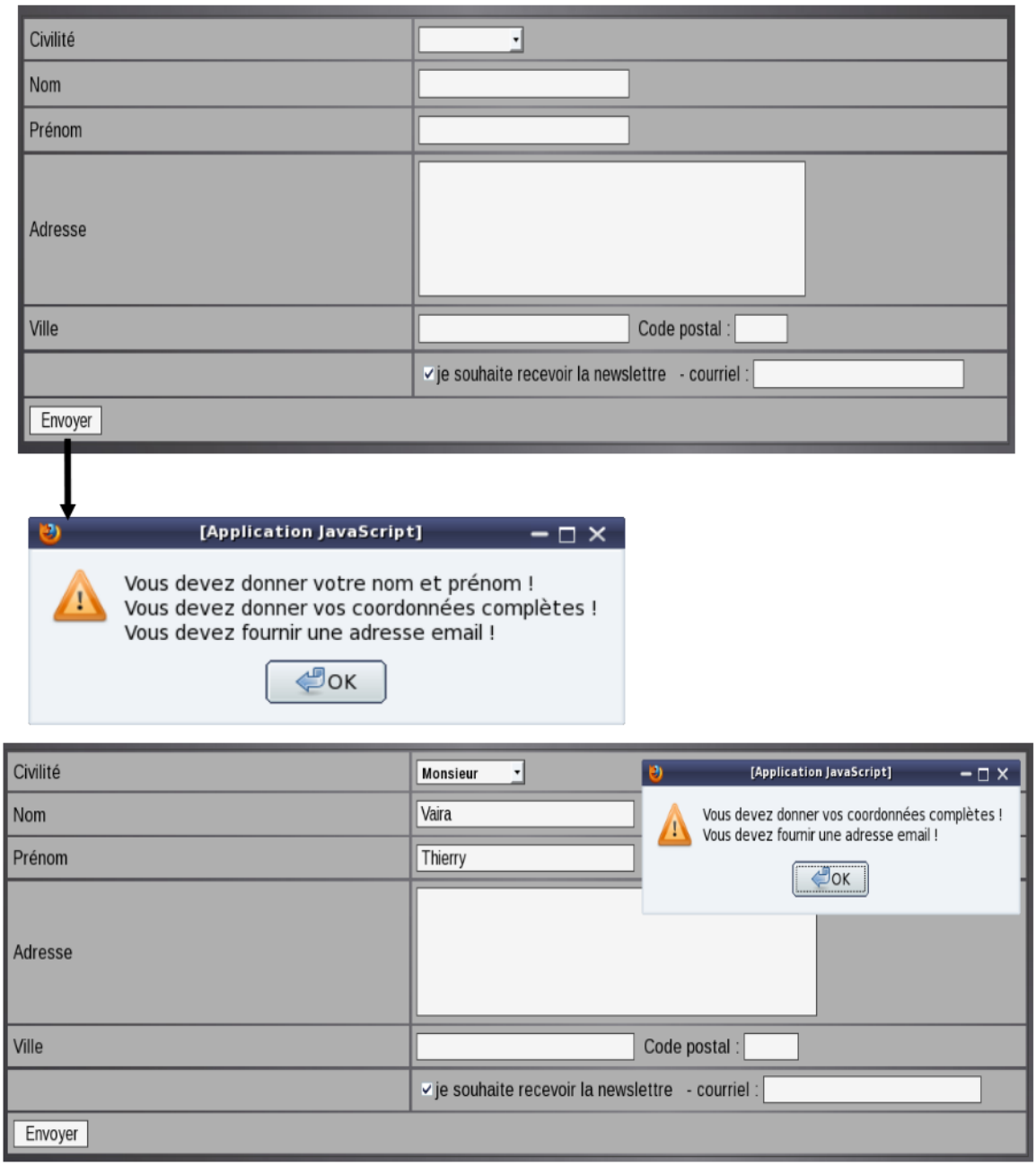
Une fois les données du formulaire contrôlées (toutes les informations sont présentes et cohérentes), réaliser son envoi par *mail* sous forme d'un texte simple :

Modifier le bouton "Envoyer" dans la page `contact.html` de telle manière que :
– le formulaire ne soit plus envoyé automatiquement (il faut donc changer le `type="submit"` en `type="button"`)
– lorsqu'on clique sur le bouton, la fonction verifierFormulaire() soit appelée :

```
function verifierFormulaire()
{
  var formulaire = document.getElementById('demande') ;
  var valid = 1;
  // vérifications ...
  // Le formulaire est-il validé ?
  if(valid == 1)
  {
        // Alors envoyer le formulaire (appel de la méthode submit)
        // et retourner true
  }
  else // Sinon retourner false
}
```

Squelette de la fonction verifierFormulaire()

Question 14. Reprendre le formulaire réalisé dans le TP HTML (`contact.html`) et assurer son contrôle en *javascript* : on n'acceptera pas les champs non remplis ou les choix ou options non renseignés (cf. exemples).



Question 15. Bonus : Vous pouvez ajouter les contrôles ci-dessous pour améliorer l'interactivité avec le client.

- Si aucune civilité n'a été choisie, fixer "Monsieur" par défaut
- Si la case à cocher "je souhaite recevoir la newslettre" est activée alors donner le focus à la saisie de l'adresse email
- Si il y a saisie d'adresse email alors activer automatiquement la case à cocher "je souhaite recevoir la newslettre"

Conclusion

Le code *javascript* permet de réaliser des contrôles côté client.

