

Exercices JavaScript

Auteur : E.Thirion - 07/12/2015

Ce document est extrait du site <http://cours.thirion.free.fr/Cours/Javascript>

Les exercices suivants sont des fichiers à compléter.

Les corrigés peuvent être exécutés à distance à partir du site contenant ce [cours](#) (cliquez sur **Test** dans la partie **Exercices** du menu).

Par contre, pour faire les exercices, il vous faudra nécessairement télécharger un certain nombre de dossiers (en particulier le site web contenant ce cours et les fichiers à compléter) sur votre machine. Pour savoir comment procéder cliquez [ici](#). Lorsque vous aurez recopié le cours sur votre machine, vous pourrez tester vos solutions via l'entrée **Solution des exercices / Testez vos solutions** du menu.

Exercice I

Fichiers à compléter

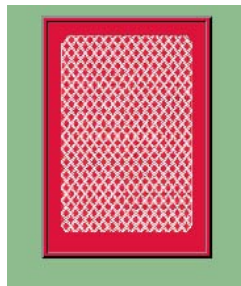
- Etudiant/RetournerUneCarte.html
- Etudiant/RetournerUneCarte.js

Connaissances nécessaires du langage Javascript

Principes fondamentaux + Base de la prog: Type des variables, conditionnelle.

I-1) Objectif

La page contient à son premier chargement, l'image d'une carte à l'envers contenu dans un tableau à une cellule:



Lorsque l'on clique dans la cellule du tableau, la carte se retourne. Deux cas possibles:

- si elle était à l'envers, on voit apparaître la dame pique:



- si elle était à l'endroit, on voit apparaître l'arrière de la carte.

I-2) Votre travail

- Ecrire la balise script permettant d'inclure le code javascript dans la page.
- Complétez la balise **body** et en lui associant la fonction **CarteALEnvers** pour l'évènement **onLoad** (chargement de la page) .
- Complétez la balise **td** du tableau en lui donnant un identifiant et en lui associant la fonction **RetournerLaCarte** pour l'évènement **onClick**.
- Complétez la fonction **CarteALEnvers** du fichier javascript. Cette fonction doit insérer l'image de l'arrière d'une carte (fichier **Images/Cartes/Arriere1.gif**) dans la page.
- Complétez la fonction **RetournerLaCarte**. Si la carte apparait actuellement à l'envers, cette fonction remplace l'image **Arriere1.gif** par l'image de la dame de pique (**PiqueDame.gif**). Sinon, elle la remet à l'envers.

Indication: utilisez un booléen pour mémoriser l'état (envers/endroit) de la carte.

Exercice II

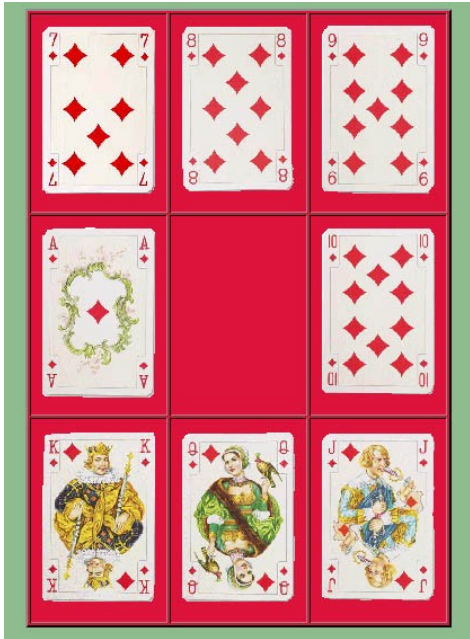

Fichiers à compléter: Etudiant/LesHuitsCartes .js

Connaissances nécessaires du langage Javascript

Principes fondamentaux + Bases de la prog : Type des variables, conditionnelle, boucles, tableau à une dimension.

II-1) Objectif

La page contient à son premier chargement, les images des huit cartes de carreau réparties dans un tableau à trois lignes et trois colonnes:

	
Page de départ	Après un clic sur la cellule centrale

Lorsque l'on clique dans la cellule centrale du tableau, les valeurs des cartes sont augmentées de un, sauf l'as qui est transformé en un sept.

Lorsqu'on clique sur une cellule contenant une carte, sa famille (carreau, coeur, pique, trèfle) change, mais

pas sa valeur:

- Une carte de carreau est transformée en coeur.
- Une carte de coeur est transformée en pique.
- Une carte de pique est transformée en trèfle.
- Une carte de trèfle est transformée en carreau.

II-2) Contenu de la page HTML

La page HTML de cette exercice est contenue dans le fichier **LesHuitsCartes.html**. Vous n'aurez pas à modifier ce fichier, mais ouvrez le pour analyser son contenu.

Vous constaterez qu'il contient un tableau HTML de trois lignes et trois colonnes. Toutes les cellules sont vides. La cellule centrale ne possède pas d'identifiant et les huit autres possèdent les identifiants suivants:

C0	C1	C2
C7		C3
C6	C5	C4

Les appels de fonctions provoquées par un clic sur une cellule (attribut **onClick**) ont été définis de la manière suivante:

FamilleSuivante (0)	FamilleSuivante (1)	FamilleSuivante (2)
FamilleSuivante (7)	AugmenterLesValeurs ()	FamilleSuivante (3)
FamilleSuivante (6)	FamilleSuivante (5)	FamilleSuivante (4)

Un clic sur la cellule centrale provoquera donc l'appel de la fonction **AugmenterLesValeurs** et un clic sur une des huit autres cellules provoquera l'appel de la fonction **FamilleSuivante** avec le numéro de la cellule comme valeur de paramètre.

II-3) Codage des valeurs et des familles

Pour faciliter la programmation, les valeurs et les familles de cartes seront codées par des nombres entiers de la manière suivante:

Valeur	Sept	Huit	Neuf	10	Valet	Dame	Roi	As
Code	1	2	3	4	5	6	7	8


Famille	Carreau	Coeur	Pique	Trèfle
Code	0	1	2	3

II-4) Représentation des données

L'état de la page est défini par les cartes contenues dans les huit cellules du tableau HTML. On utilisera deux tableaux pour mémoriser leur contenu:

- **Valeur** : tableau de 8 éléments. **Valeur [i]** contient le code de la valeur de la carte contenue dans la cellule **Ci**.
- **Famille** : tableau de 8 éléments. **Famille [i]** contient le code de la famille de la carte contenue dans la cellule **Ci**.

Par exemple, la page suivante:

	<table><tr><td>C0</td><td>C1</td><td>C2</td></tr><tr><td>C7</td><td></td><td>C3</td></tr><tr><td>C6</td><td>C5</td><td>C4</td></tr></table>	C0	C1	C2	C7		C3	C6	C5	C4
C0	C1	C2								
C7		C3								
C6	C5	C4								
Page affichée par le navigateur	Identifiants des cellules									

Sera codée de la manière suivante:

i	0	1	2	3	4	5	6	7
Contenu de la cellule Ci	10 de Coeur	Valet de Carreau	Dame de Carreau	Roi de Pique	As de Trèfle	Sept de Coeur	Huit de Pique	Neuf de Trèfle
Valeur	4	5	6	7	8	1	2	3
Famille	1	0	0	2	3	1	2	3

II-5) Conventions pour nommer les images de cartes

Pour cet exercice, les images de cartes à utiliser se trouvent dans le dossier **Images/CartesCode**.

Ces fichiers image ont un nom de la forme **XY.gif** où :

- **X** est le code de la valeur de la carte.
- **Y** est le code de la famille de la carte.

II- 6) Votre travail

II-6-1 Page de départ

Il s'agit de faire fonctionner la fonction **PageDeDepart** appelée au chargement de la page. Voici le code de cette fonction:

```
function PageDeDepart ( )
{
  Initialiser ( ) ;
  for (i=0; i<=7; i++) CodeHTMLDeLaCellule(i) ;
}
```

Votre travail consistera à compléter les deux fonctions **Initialiser** et **CodeHTMLDeLaCellule** afin que la fonction **PageDeDepart** fonctionne correctement.

Pour cela, la fonction **Initialiser** doit créer et initialiser les deux tableaux **Valeur** et **Famille** afin qu'ils représentent l'état initial de la page (voir paragraphe 1) avec les conventions de représentation qui ont été définies (paragraphe 3, 4 et 5).

La fonction **CodeHTMLDeLaCellule** place l'image de la ième carte dans la cellule **Ci** du tableau HTML. On utilisera une largeur d'image de 100 pixels.

Par exemple, supposons que le contenu des tableaux **Valeur** et **Famille** soit le suivant:

i	0	1	2	3	4	5	6	7
Contenu de la cellule Ci	10 de Coeur	Valet de Carreau	Dame de Carreau	Roi de Pique	As de Trèfle	Sept de Coeur	Huit de Pique	Neuf de Trèfle
Valeur	4	5	6	7	8	1	2	3
Famille	1	0	0	2	3	1	2	3

alors l'appel de fonction **CodeHTMLDeLaCellule (2)** placera l'image de la dame de carreau dans la cellule **C2** du tableau. Plus précisément, il faudra donc y insérer la balise HTML suivante:

```

```

II-6-2 Clic sur une cellule

Un clic sur la cellule centrale du tableau HTML provoque l'appel de fonction **AugmenterLesValeurs()**. Un clic sur une des huit autres cellule provoque l'appel de fonction **FamilleSuivante (i)** où **i** est le numéro de cette cellule. Complétez le code de ces deux fonctions de manière à obtenir le comportement décrit au paragraphe 1.

Exercice III

Fichiers à compléter: Etudiant/ExoForm.js


Connaissances nécessaires du langage Javascript

Principes fondamentaux + Formulaires + Expressions régulières + Bases de la prog : Type des variables, conditionnelles.

III-1) Objectif

Il s'agit de contrôler les données d'un formulaire donnant différentes informations sur une personne. Les données sont contrôlées à l'aide d'expressions régulières. Le formulaire n'est validée que si toutes les données sont correctes.

III-2) Le formulaire

Le fichier html contenant le formulaires est ExoForm.html .	De haut en bas et de gauche à droite: type des composants et identifiants associés (en gras)
	<p>Zone de texte - nom</p> <p>Zone de texte - prenom</p> <p>Boutons radio - H - F</p> <p>Case à cocher - SansEmploi</p> <p>Zone de textes - jour - mois - annee</p> <p>Liste de sélection - Lieu</p> <p>Bouton de type "button"</p> <p>Zone de texte multi-ligne - DonneesSaisies</p> <p>Bouton de type "submit"</p>

Voici le code HTML de la liste de sélection:

```
<select id="Lieu">
  <option value="COLMAR">Colmar</option>
  <option value="PARIS">Paris</option>
  <option value="STRASBOURG">Strasbourg</option>
</select>
```

III-3) Le bouton Phrase

Le but de cette question est de travailler l'accès aux composants d'un formulaire. Lorsque l'utilisateur clique sur le bouton **Phrase**, une phrase résumant les données saisies est affichée dans la zone de texte multi-lignes. La phrase est différente si la personne est sans emploi:

The image shows two side-by-side screenshots of a web form. The form has a light green background and contains the following fields:

- Nom:** A text input field containing "thirion".
- Prenom:** A text input field containing "eric".
- Sexe:** Two radio buttons labeled "H" (selected) and "F".
- Sans emploi:** A checkbox. In the left screenshot, it is unchecked; in the right screenshot, it is checked.
- Date de naissance:** Three text input fields containing "6", "6", and "61".
- Lieu de naissance:** A dropdown menu showing "Colmar".
- Phrase:** A button with a dotted border.
- Summary text:** A multi-line text area showing the generated phrase.
 - Left screenshot: "Monsieur thirion eric, né le 6/6/61 à COLMAR"
 - Right screenshot: "Monsieur thirion eric, né le 6/6/61 à COLMAR, actuellement sans emploi"
- Valider:** A button at the bottom of each form.

Le gestionnaire de ce bouton pour l'évènement **onClick** est la fonction **FormerPhrase** que l'on vous demande de compléter.

III-4) Validation du formulaire

Voici l'entête de la balise **form** du formulaire:

```
<form action="ExoFormOut.html" onsubmit="return Verifier()">
```

Un clic sur le bouton **Valider** provoque donc l'appel de la fonction **Verifier**. Si elle retourne la valeur **true**, la page **ExoFormOut.html** est appelée, sinon il ne se passe rien.

Cette page affiche le message suivant:

Les données du formulaire ont été envoyées au serveur !

Voici code de la fonction **Verifier** :

```
function Verifier ()
{
    if (!NomCorrect()) {alert("Nom incorrect!"); return false; }
    if (!PrenomCorrect()) {alert("Prenom incorrect!"); return false;}
    if (!JourCorrect()) {alert("Jour incorrect!"); return false;}
    if (!MoisCorrect()) {alert("Mois incorrect!"); return false;}
    if (!AnneeCorrecte()) {alert("Année incorrecte!"); return false;}

    return true;
}
```

Elle ne retournera donc la valeur **true** que si les cinq fonctions **NomCorrect**, **PrenomCorrect**, **JourCorrect**, **MoisCorrect** et **AnneeCorrecte** retournent toutes la valeur **true**.

Votre travail consistera à écrire le code de ces cinq fonctions en utilisant des expressions régulières.

III-4-a) Vérification du nom et du prénom

Complétez les fonctions **NomCorrect** et **PrenomCorrect** de manière à ce qu'elles ne retournent la valeur **true** que si le nom et le prénom sont des suites non vide de lettres de l'alphabet en minuscules ou majuscules.

III-4-b) Vérification du jour de naissance

Complétez la fonction **JourCorrect** de manière à ce qu'elle ne retourne la valeur **true** que si:

1. Le jour de naissance est composé d'un ou deux chiffres, le premier chiffre pouvant être un 0. On fera cette vérification à l'aide d'une expression régulière.
2. Le nombre défini par cette chaîne de caractères est compris entre 1 et 31. On utilisera pour cela la fonction **eval**.

III-4-c) Vérification du mois de naissance

Complétez la fonction **MoisCorrect** de manière à ce qu'elle ne retourne la valeur **true** que si:

1. Le mois de naissance est composé d'un ou deux chiffres, le premier chiffre pouvant être un 0.
2. Le nombre défini par cette chaîne de caractères est compris entre 1 et 12.

III-4-d) Vérification de l'année de naissance

Complétez la fonction **AnneeCorrect** de manière à ce qu'elle ne retourne la valeur **true** que si l'année de naissance est composée de 4 chiffres exactement.

III-4-e) Amélioration de la vérification du nom et du prénom

Vous allez constater ici que plus on est tolérant et plus les expressions régulières deviennent complexes !

Modifiez l'expression régulière des fonctions **NomCorrect** et **PrenomCorrect** afin qu'un nom ou un prénom puisse contenir des tirets (attention, pas deux tirets de suite !) et des blancs (comme par exemple "jean-paul", "van Gogh ") avec en prime, des blancs optionnels avant ou après.