

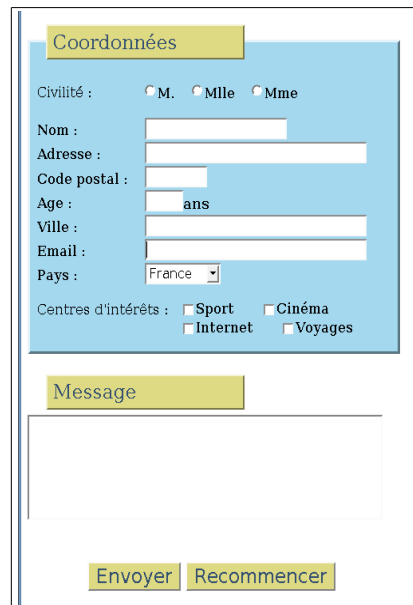
**TP N°1 Technologies XML**  
**- HTML/JavaScript -**  
(Initiation et/ou révision)

**Remarque : Temps approximatif nécessaire à chaque exercice indiqué entre parenthèses.**

**Exercice 1 : (Entrées/sorties + tests) (40 min)**

Réaliser un formulaire permettant de saisir des informations vous concernant et de les envoyer par email (à vous-même!). Le formulaire sera réalisé en 2 temps : d'abord vous vous focaliserez sur sa construction. Puis, si le temps vous le permet, vous construirez une feuille de style que vous associerez au formulaire.

Voici un aperçu du formulaire à obtenir :



Les chiffres saisis de votre âge devront être codés pour que personne ne puisse le voir.

L'action à exécuter par le bouton "Envoyer" doit être définie par l'attribut "action" du formulaire :

action="<mailto:eleve@ecole.ensicaen.fr?subject=TP1 de HTML>"

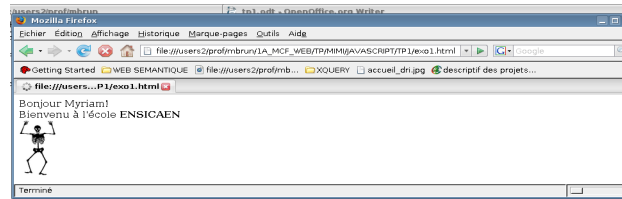
Le résultat obtenu par courrier prend la forme suivante :

- en-tête du mail :  
*De : eleve@ecole.ensicaen.fr*  
*Sujet : TP 1 de HTML*  
*Pour : eleve@ecole.ensicaen.fr*
- contenu du mail :  
*civilite=Mlle&nom=Brun&adresse=Boulevard+Maréchal+Juin&codepostal=14000&age=20&ville=Caen...*

NB : Les données inscrites dans votre formulaire peuvent également être envoyées au serveur pour être traitées par un script PHP (abordé plus tard). De même, les données devraient être vérifiées avant d'être envoyées ; vous ferez cette vérification à l'aide du langage JavaScript (abordé plus tard dans l'Exercice 10).

## Exercice 2 : (Entrées/sorties + tests) (40 min)

Réaliser une page web permettant à l'utilisateur de saisir son prénom puis le nom de son école. En réponse, l'utilisateur verra affiché (s'il s'appelle Myriam et est élève de l'ENSICAEN) :

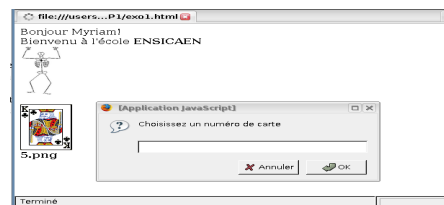


Si le prénom ou le nom de l'école n'a pas été saisi, alerter l'utilisateur par "Saisie incomplète !". L'image squelette est le fichier "Artie.gif" du répertoire "/home/public/2A\_INFO/TECHNO\_XML/TP1/EX2/".

## Exercice 3 : (Entrées/sorties + tests + boucle) (40 min)

Compléter la page précédente pour permettre à l'utilisateur de choisir un numéro de carte à jouer. Si le n° de carte n'est pas compris entre 1 et 52, alerter l'utilisateur par "Désolé ! Numéro incorrect !". Sinon, afficher l'image de la carte correspondante. Les fichiers images de ces cartes ont tous pour nom "num.png" où *num* est le numéro saisi, et se trouvent sous le répertoire "/home/public/2A\_INFO/TECHNO\_XML/TP1/EX3/". Afficher sous la carte le nom du fichier.

Enfin, permettre à l'utilisateur de saisir un nouveau n° de carte tant qu'il réponds "OK" au message de confirmation "Voulez-vous choisir une autre carte?"



## Exercice 4 : (boucles + opérateurs) (40 min)

Réaliser une page web affichant la table des carrés et la table de multiplication selon l'exemple de la Figure ci-contre.

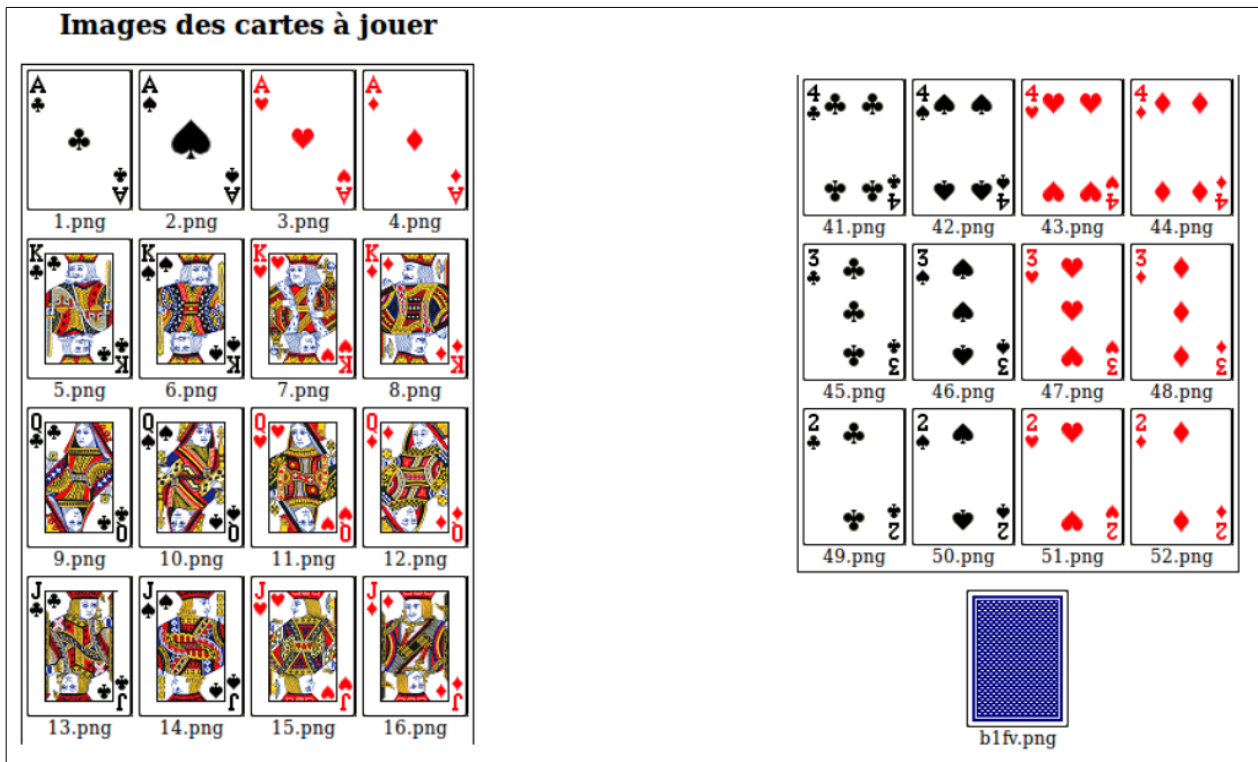
Pour cela, utiliser un tableau HTML centré avec bordure pour chaque table. Pour appliquer une couleur de fond particulière à la 1ère colonne et la 1ère ligne, utilisez la balise "<th>" et appliquer votre couleur à ces éléments à l'aide d'une feuille de style attachée ou intégrée. Donnez un titre "Tables des carrés et de multiplication" à votre page web.

Tableaux générés par JavaScript										
Table des carrés										
0	0									
1	1									
2	4									
3	9									
4	16									
5	25									
6	36									
7	49									
8	64									
9	81									
10	100									
Table de multiplication										
0	1	2	3	4	5	6	7	8	9	
0	0	0	0	0	0	0	0	0	0	
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

## Exercice 5 : (boucles + opérateurs) (30 min)

Réaliser une page web affichant l'ensemble des 52 cartes du jeu sur 4 colonnes. Pour cela, utiliser un tableau HTML centré avec bordure. Le nom du fichier de chacune des cartes sera affiché en dessous de l'image carte. En dernière ligne, afficher la carte retournée (fichier *blfv.png*). En tête de page, afficher "Images des cartes à jouer pour le TP de JavaScript en 2A INFO". Donnez un titre "Cartes de jeux" à votre page web. Enfin, sous les cartes, afficher "Source et information sur le jeu de 52 cartes : [https://fr.wikipedia.org/wiki/Jeu\\_de\\_cartes\\_français](https://fr.wikipedia.org/wiki/Jeu_de_cartes_français)

NB : les cartes se trouvent sous 2A\_INFO/TECHNO\_XML/TP1/EX4/



### Exercice 6 : (boucles + array) (30 min)

Réaliser une page web affichant pour chaque année  $x$  le message " $x$  est une année bissextile" si  $x$  est bissextile, ou affichant " $x$  n'est pas une année bissextile" si  $x$  ne l'est pas.  $x$  sera affichée en gras dans le 1er cas et en italique dans le second. Les années choisies seront initialement réunies dans un tableau. Une année est bissextile si elle divisible par 4 mais pas par 100, ou si elle est divisible par 400.



### Exercice 7 : (fonctions + array) (60 min)

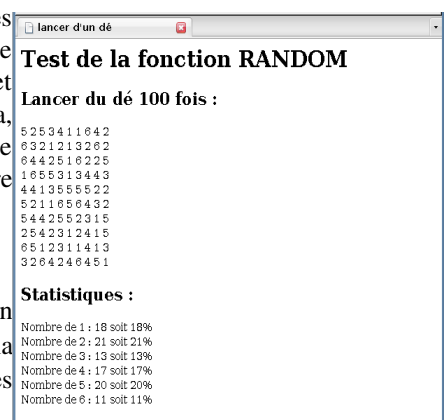
Réaliser une page web affichant les résultats de lancés d'un dé ainsi que ses statistiques. Le nombre de lancés est saisi par l'utilisateur à travers une boîte de dialogue. Pour chaque chiffre du dé, l'on calculera le nombre d'apparitions et son pourcentage (voir un exemple de page web à obtenir ci-dessous). Pour cela, un tableau (*resultat*) de taille 6 sera utilisé pour mémoriser les scores. Le programme JavaScript sera décomposé en fonctions avec un paramètre éventuel, décrites comme suit :

**init()** : initialise à 0 le tableau *resultat* (variable globale)

**de()** : retourne un nombre compris entre 1 et 6. Utiliser la fonction **Math.random()** qui donne un nombre au hasard dans l'intervalle  $[0, 1[$ , et la fonction **Math.floor(x)** qui fournit l'arrondi à l'entier le plus proche (définies avec l'objet Math).

**lancer(nbfois)** : lance *nbfois* le dé. Affiche les numéros obtenus. Mémorise dans le tableau *resultat* le score pour chacun des numéros.

**afficher(nbfois)** : afficher les statistiques (nombre d'occurrences de chaque chiffre de 1 à 6 et pourcentages).



## Exercice 8 (synthèse) : (E/S, tests, boucles, array, fonctions) (120 min + 120 min )

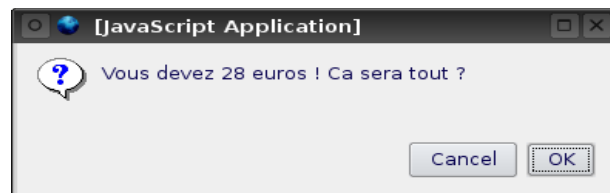
### Le marché, version 1



On veut écrire une page à partir de laquelle on peut faire son marché. Pour remplir son panier, il suffit de cliquer sur le produit désiré. On peut cliquer plusieurs fois sur les mêmes produits. Quand on a terminé, on peut payer, ce qui a pour effet de vider le panier. On peut alors recommencer un nouveau marché. Si on clique sur "payer" alors que le panier est vide, il faut afficher un message du type :



Si on clique sur "payer" alors que le panier n'est pas vide, il faut afficher et confirmer le montant :



Vous pouvez récupérer les images contenues dans le fichier 2A\_INFO/TECHNO\_XML/TP1/EX8/marche.zip.

#### 1. Réalisation de la partie HTML

- Faire un tableau HTML avec :

1ère ligne : les 4 images de fruits et légumes. Chaque image sera associée à un lien représentant un appel de la fonction JavaScript nommée "choisir()". Exemple :

```
<td> <a href= "JavaScript:choisir('asperges')">  </a></td>
```

2ème ligne : les 4 prix en euros (&euros; )

- Ecrire le texte en bas "Quand vous avez terminé .....vous pouvez **payer** ....." en mettant un lien hypertexte devant le mot "payer" représentant un appel de la fonction JavaScript nommée "payer()".

#### 2. Réalisation de la partie JavaScript

- Déclarer une variable globale "total" qui contiendra le total à payer.
- Ecrire les 3 fonctions suivantes (celles-ci sont simples dans cette version, mais elles seront enrichies dans les versions 2 et 3) :

**choisir(produit)** : ajoute à "total" le prix du produit passé en paramètre. Appelle la fonction "prix()".

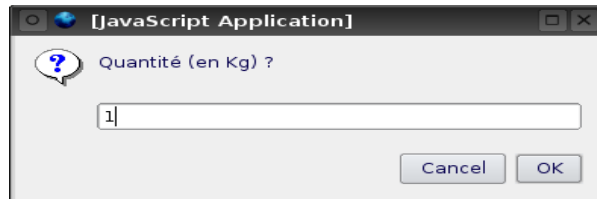
**prix(produit)** : retourne le prix du produit passé en paramètre.

**payer()** : si le total est 0, afficher une boîte alerte "votre panier est vide!", sinon afficher dans une boîte de confirmation "Vous devez .... euros ! Ce sera tout ?). Dans l'affirmatif, remettre "total" à "0".

## Le marché, version 2



Ecrivez maintenant une page HTML pour pouvoir faire son marché de façon plus élaborée. Dans cette nouvelle version, lorsqu'on clique sur un produit, il faut ensuite préciser la quantité désirée :



Si on clique sur "panier" on doit voir s'afficher le contenu et le prix du panier courant :



### 1. Réalisation de la partie HTML

- idem solution 1, mais modifier le texte "Pour faire votre marché, ....." en mettant un lien hypertexte devant le mot "panier" représentant un appel de la fonction JavaScript nommée "voir\_panier()".

### 2. Réalisation de la partie JavaScript

- Déclarer en plus une variable globale "panier" qui représentera le contenu du panier (c-à-d le message à affiché ci-dessus sans le Total).

**choisir(produit) :** demander aussi la quantité. Sauvegarder dans panier la quantité et le nom du produit.

**prix(produit) :** idem solution 1.

**payer() :** si le total est 0, afficher une boîte alerte "votre panier est vide!", sinon afficher dans une boîte de confirmation "Vous devez .... euros ! Ce sera tout ?). Dans l'affirmatif, remettre "total" à 0 et afficher le panier en appelant "voir\_panier()".

**voir\_panier() :** dans une boîte alert, afficher la variable "panier" et le "total".

## Le marché, version 3

Dans cette nouvelle version, le panier doit rester **trié** tout le long du marché : si le client achète 1 Kg de carottes, 0.5 Kg d'asperges, puis de nouveau 1 Kg de carottes, et demande ensuite de voir l'état de son panier courant, il doit voir *2 Kg de carottes, 0.5 Kg d'asperges*.

## Exercice 9 : (Objets) (120 min)

Le but est d'écrire une page présentant des livres sur JavaScript en utilisant une base de données représentée par un tableau d'objets. Chaque livre sera caractérisé par une image, un titre, un auteur, un prix et un éditeur. La liste des livres est contenue dans le fichier *livres.js* et les images sont dans le fichier *livres.zip*. Voici un aperçu de votre futur site : Vous pouvez récupérer les images contenues dans le fichier 2A\_INFO/TECHNO\_XML/TP1/EX8/livres.zip.

Ecrire un fichier *livre.html* qui affichera les données contenues dans *livre.js*. L'affichage du tableau se fera en JavaScript.

### 1. Entête du fichier HTML

Celle-ci contiendra les déclarations JavaScript des objets *Livre*, de la méthode *afficherLivre*, du tableau *ListeLivre* et de la fonction *ajouter*.

#### ○ Objet *Livre*

Ecrire une fonction *Livre(titre, auteur, editeur, prix, image)* permettant de construire un objet de *Livre* de titre 'titre', dont l'auteur est 'auteur', l'éditeur 'editeur', le prix 'prix' et l'image qui l'illustre 'image'. De plus, l'objet *Livre* possédera une méthode *afficher* qui sera initialisée par la méthode *afficherLivre*.

#### ○ Méthode *afficherLivre*

Ecrire une méthode *afficherLivre* qui permet d'afficher du code HTML correspondant à une ligne du tableau HTML final (pour le format de la ligne, voir l'aperçu de la page web ci-dessous).

#### ○ Tableau *ListeLivre*

Déclarer un tableau nommé *ListeLivre* qui contiendra l'ensemble des livres.

#### ○ Fonction *ajouter*

Ecrire une fonction *ajouter* permettant d'ajouter facilement un nouvel enregistrement (*Livre*) à ce tableau.

### 2. Corps du fichier HTML

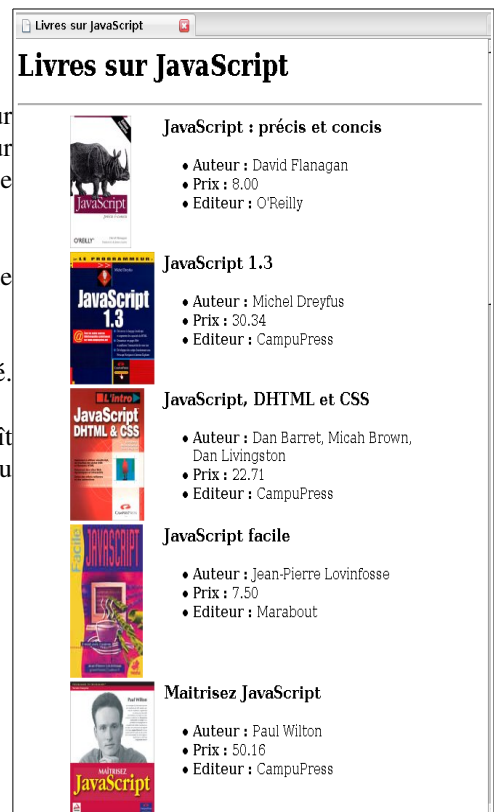
Pour l'ensemble des ajouts, vous utiliserez le fichier *livre.js*.

Celui-ci contient plusieurs appels de la fonction *ajouter* pour ajouter des livres. C'est ce seul fichier qui devra être modifié pour ajouter de nouveaux enregistrements. L'affichage des données se fera de façon indépendante.

Utiliser l'attribut *src* dans la balise `<script> ...</script>` pour le chargement de *livre.js*.

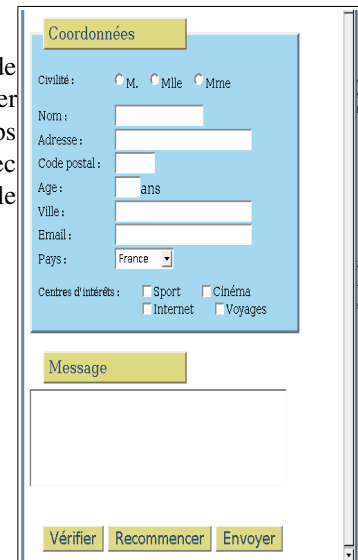
Pour afficher les données, vous utiliserez un tableau HTML centré.

Les lignes du tableau seront écrites en JavaScript (on ne connaît pas leur nombre et leur contenu se trouve dans les éléments du tableau *ListeLivres*).



## Exercice 10 : (Objet Formulaire) (120 min + 120 min)

Dans l'Exercice 0 (HTML), vous avez construit un formulaire permettant la saisie de données. Les données inscrites dans le formulaire ont été envoyées par courrier électronique. Ici, vous allez vérifier avec du JavaScript que les valeurs des champs saisies ou sélectionnées sont valides. Si ce test est positif, l'envoi pourra se faire avec "Envoyer", sinon une boîte de dialogue demandera à l'utilisateur de saisir une nouvelle fois les champs erronés.



### 1. Reprise du formulaire

Editez le fichier html correspondant au formulaire de l'Exercice 0.

### 2. Traitement JavaScript

Ce traitement sera contenu dans une fonction nommée *formtest()*. On associera cette fonction à la propriété *onClick* du bouton *submit* et du bouton *button*, en complétant la ligne HTML lui faisant référence :

```
<input type="submit" value="Envoyer" onClick="return  
formtest(document.questionnaire) && securite('expédier maintenant') ">
```

```
<input type="button" value="Vérifier" onClick="formtest(document.questionnaire) " />  
où questionnaire est le nom du formulaire.
```

De même, on associera la fonction *securite()* à la propriété *onClick* du bouton *reset*, en complétant la ligne HTML lui faisant référence : `<input type="reset" value="Recommencer" onClick="return securite('vider') " />`

Description des fonctions à réaliser :

1. La fonction *securite(action)*  
affiche une boîte de confirmation "Etes-vous sûr de vouloir *action* le document ?" avec pour *action* la valeur '*vider*' ou '*expédier maintenant*' selon l'appel de cette fonction. Retourne aussi la réponse de l'utilisateur.
2. La fonction *formtest(doc)*  
vérifie que les zones du formulaire *doc* sont bien saisies et valides. Les vérifications à faire sont :
  - Vérifier que la civilité a été choisie
  - Vérifier que les zones de texte sont saisies
  - Vérifier que le code postal est bien une suite de 5 chiffres
  - Vérifier que l'âge est bien un nombre et est compris entre 6 et 77 ans
  - Vérifier que le email comporte un @ et un point, et que la longueur est  $\geq 7$
  - Vérifier que le pays a été sélectionné
  - Vérifier qu'au moins un centre d'intérêts a été coché

Lors de chacune des vérifications, si le test n'est pas bon :

- afficher une boîte *alert* avec un message adéquat. Exemple : `alert("Ce n'est pas une adresse E-mail valide !");`
- Re-Positionner le curseur sur la zone concernée
- retourner *false*.

Au fur et à mesure des vérifications, on construira une chaîne *resultat* permettant de mémoriser les valeurs choisies ou saisies par l'utilisateur.

La dernière instruction de la fonction devra afficher dans une boîte *alert* le récapitulatif des réponses saisies, en utilisant la variable *resultat* :

Récapitulatif :

\*\*\*\*\*

Nom : Brun

Prenom : Myriam

Adresse : .....

...

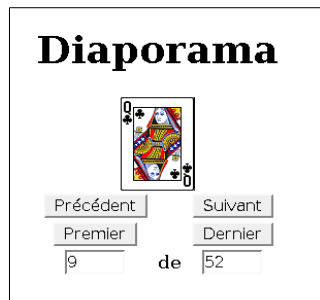
Centre d'intérêts : Cinéma Voyages

### Indications :

- Vérifier que la civilite a été choisie :  
utiliser les propriétés :  
    length (pour le nombre de cases a cocher), value, checked
- Vérifier que les zones de texte sont saisies :  
utiliser les propriétés :  
    length (pour le nombre de zones), elements, type, value, name  
utiliser les méthodes :  
    focus()
- Vérifier que le code postal est bien une suite de 5 chiffres :  
utiliser les propriétés :  
    focus()  
faire :  
    var format = /^[0-9]{5}\$/;      précise le format d'un code postal  
    if (format.test(vcodepostal)) ...pour tester si le code postal vérifie le format
- Vérifier que l'âge est bien un nombre et est compris entre 6 et 77 ans :  
utiliser les méthodes :  
    focus()  
    utiliser la fonction : isNaN()
- Vérifier que le email comporte un @ et un point et que la longueur est >=7 :  
utiliser les propriétés :  
    length (pour la longueur du email)  
utiliser les méthodes :  
    charAt(i), focus()
- Vérifier que le pays a été sélectionné :  
utiliser les propriétés :  
    selectedIndex, text  
utiliser les méthodes :  
    focus()
- Vérifier qu'au moins un centre d'intérêts a été coché :  
utiliser les propriétés :  
    length (nombre de cases à cocher), checked, value



## Exercice 11 : (Objet Formulaire, Document, propriétés, événements) (120 min)



On désire réaliser un diaporama permettant d'afficher une à une les cartes d'un jeu de 52 cartes. Ainsi, l'utilisateur pourra visualiser la première carte, la dernière carte, la précédente, la suivante. Le n° de carte (ici 9) en cours de visualisation sera affiché, ainsi que le nombre total de cartes.

### 1. Réalisation de la partie HTML

La page web commencera par afficher le titre *Diaporama* puis l'image de la première carte (1.png). Puis vous afficherez un formulaire possédant des zones *input* (rem : pour avoir des boutons bien alignés, utilisez un tableau) :

- 4 zones de type *button* pour *Précédent*, *Suivant*, *Premier*, *Dernier*. Pour chacune de ces zones, on associera une fonction à la propriété *onClick*. Exemple pour le bouton *Précédent* :  
`<input type="button" name="previous" value="Précédent" onClick="precedent()" />`
- 2 zones de type *text* pour afficher le n° de carte courant et le nombre total de cartes

### 2. Réalisation de la partie JavaScript

Les fonctions JavaScript à définir :

- *etat(n)*  
Affiche ou cache les boutons selon la valeur de *n* (utiliser la propriété *disabled*). Si *n=1*, cache le bouton *Précédent* (*disabled=true*), sinon affiche le bouton *Précédent* (*disabled=false*) ; si *n=52*, cache le bouton *Suivant* ; sinon affiche le bouton *Suivant*. De même, affiche le n° de carte courant dans la zone de texte correspondante du formulaire.
- *init()*  
Appelée sur chargement du document (*onload=init()*). Initialise la variable globale *nombre* (de cartes) à 52 et remplit la zone de texte correspondante dans le formulaire. Appelle la fonction *etat* avec 1.
- *premier()*  
Appelée sur click du bouton *Premier*. Affiche la première carte. Appelle la fonction *etat* avec 1.
- *dernier()*  
Appelée sur click du bouton *Dernier*. Affiche la dernière carte. Appelle la fonction *etat* avec *nombre*.
- *suivant()*  
Appelée sur click du bouton *Suivant*. Affiche la carte suivante (c-à-d dont le n° est égal au n° de carte courant +1), si elle existe. Appelle la fonction *etat* avec ce nouveau n°.
- *precedent()*  
Appelée sur click du bouton *Précédent*. Affiche la carte précédente (c-à-d dont le n° est égal au n° de carte courant -1), si elle existe. Appelle la fonction *etat* avec ce nouveau n°.