But de la séance : Durant ce TP on va construire une page, comprenant un formulaire. La mise en page utilisera CSS. Les données seront contrôlées au moyen d'expressions régulières, afin d'éviter la validation du formulaire alors que la saisie n'est pas correcte. Un peu de javascript viendra compléter l'ensemble pour améliorer l'UX (*User eXperience*, soit l'interaction entre la page et l'utilisateur).

Pré-requis :

- Navigateur, éditeur de texte

Références :

- Celene : https://celene.univ-tours.fr/course/view.php?id=9279
-

Formulaire de paiement par carte bancaire

Un site marchand, qui permet de payer en ligne, propose un formulaire de saisie des informations bancaires. Ces informations sont ensuite transmises à la banque pour vérification, qui autorise ou non le paiement. Cela permet au site de s'assurer que la vente a bien eu lieu.

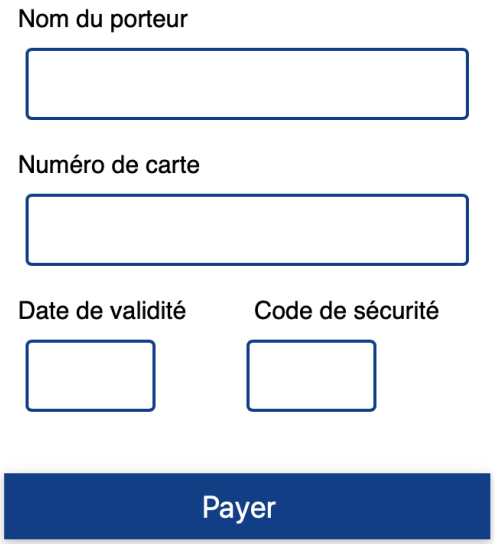
Le formulaire permet de demander : le nom (lastname) et le prénom (firstname) de l'utilisateur, tels que portés sur la carte bancaire, le numéro de la carte (16 chiffres), la date d'expiration (2 chiffres pour le mois puis 2 chiffres pour l'année, par ex. 0419), et le code de sécurité (3 chiffres). Toutes ces informations figurent sur le devant de la carte, à l'exception de la dernière qui est portée au dos. Un exemple de carte bancaire est représenté ci-dessous :

1 Créer un formulaire, en HTML, permettant de saisir ces informations, sans contrôle de validité. Vous indiquerez cependant dans les champs, un exemple de valeur correcte. Aucune présentation

particulière n'est demandée.

Il s'agit maintenant d'améliorer cette base de façon à la rendre : 1) visuellement plus attractive, et 2) plus fonctionnelle, afin d'éviter les erreurs de saisie.

2► Au moyen d'un fichier CSS séparé, modifiez la mise en forme du formulaire de façon à aboutir à un résultat proche de celui ci-contre.
Quelques indications de style :
Les polices utilisées sont de type sans-serif. La couleur retenue a pour code 0E3C87. Les champs de saisie ont les coins arrondis (4px), et présentent un espace entre le cadre et le contenu de 10px, sauf à gauche (20px). Le contour de ces champs fait 2px. La taille des caractères saisis sera un peu plus grande que la normale (1.25em).



La saisie pose problèmes. On peut mettre le contenu qu'on souhaite dans les champs, et envoyer à la banque des demandes de vérification qui ne sont pas correctes. Une première approche consiste à utiliser la capacité des champs du formulaire à vérifier leur contenu. Ceci est réalisé au moyen de l'attribut *pattern* du champ de saisie. Les règles de validation sont données page précédente, dans les noms et prénoms on n'autorise que les lettres majuscules.

3► Modifiez le paramétrage des champs du formulaire de façon à en contrôler la saisie. Vous vérifierez notamment que les saisies sont obligatoires, que chaque champ respecte la syntaxe imposée. Certains cas ne peuvent pas être décelés par cette technique : syntaxe incorrecte du numéro de carte (celui-ci n'est pas déterminé au hasard), inversion de valeurs entre le nom et le prénom, contrôle de validité sur le mois et l'année, car ils sont situés dans le même champ. Vous prendrez soin également de paramétrer les champs afin que ceux-ci renseignent l'utilisateur sur ce qui est attendu.

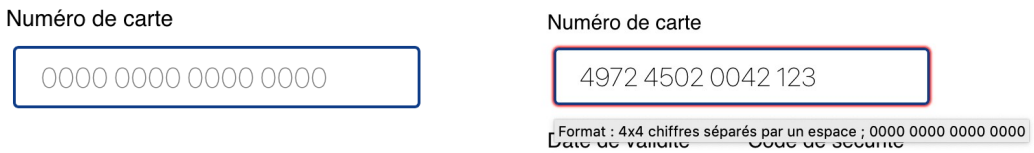
Cette version est celle minimum qu'on puisse attendre avec les technologies HTML/CSS. Voyons maintenant si on peut allez un peu plus loin.

Le numéro de carte est formé de 4 blocs de 4 chiffres, ce qui facilite sa lecture. Or quand on le saisit, les 16 chiffres se suivent. Plusieurs possibilités s'offrent à nous.

- Proposer quatre champs de saisie, un pour chaque bloc ;
- Conserver un seul champ, en autorisant la saisie d'un espace entre chaque bloc de 4 chiffres

La première solution demande un peu de javascript de façon à recomposer le numéro complet avant envoi vers le site. Elle demande aussi à modifier le HTML et le CSS. La seconde repose uniquement sur la modification du pattern de contrôle du champ. On va donc s'orienter vers celle-ci.

4 ▶ Définissez une nouvelle expression régulière de façon à autoriser une séquence formée de 4 blocs de 4 chiffres, séparés les uns des autres par un seul espace : XXXX XXXX XXXX XXXX. Toute autre syntaxe ne sera pas acceptée (en particulier celle qui comporte un espace à la fin). Modifiez alors le HTML en conséquences et testez. Voici par exemple le résultat sous Firefox (à gauche, avant la saisie, à droite, après une saisie incorrecte :



Le mois et l'année sont pour le moment fusionnés dans le même champ. Ceci interdit de façon simple un contrôle des valeurs saisies. Pour cela on peut comme précédemment :

- proposer deux champs distincts, et faire un contrôle séparé sur chaque champ ;
- laisser un seul champ, et trouver une astuce … en javascript !

Javascript va permettre d'obtenir de la machine qu'à chaque modification de la date, on considère de façon séparée le mois (les deux premier chiffres) et l'année (les deux derniers). Il sera alors possible d'interdire certaines saisies (mois en dehors de 01...12 et année en dehors de 19...24 (pour 2019-2024) par exemple. La démarche suivante est simplement à reproduire, en prenant soin de lire les explications auparavant.

On commence par nommer les éléments utiles. Un algorithme traite des entrées pour produire des sorties. Afin de limiter l'impact des changements dans le périmètre de l'algorithme, celui-ci suit toujours la même trame : on rassemble les données, on calcule le résultat, on place ces derniers dans l'environnement qui a fait appel à l'algorithme. Ici, on a besoin de lire la valeur du champ qui contient la date. Pour cela, le plus simple est de donner, en HTML, un identifiant à ce champ, de la même manière que si on voulait le faire pour faire le lien avec du CSS. Par exemple :

```
<input id="expiration" type="text">
```

Le script doit s'exécuter sur des valeur réelles, ce qui signifie qu'il ne peut s'exécuter qu'après la création de ce champ du formulaire. En HTML on peut définir des scripts où on le souhaite. On peut par exemple le placer juste avant la balise de fermeture du *body* :

```
<script >
   var cardExpiration = document.getElementById('expiration');
</script >
```

Il reste à trouver comment interagir avec les actions de l'utilisateur. Dans notre cas il faut pouvoir répéter le contrôle de la date à chaque modification de celle-ci. On peut par exemple utiliser le *callback* (une fonction appelée lorsqu'il se passe un évènement particulier) associé à la modification du champ. Il s'agit d'une variable, couplée à un évènement. On affecte cette variable avec une fonction, ou avec le nom d'une fonction. Et lorsque l'évènement se produit, la fonction affectée à la variable est appelée. Dans notre cas on utilisera *onkeyup*, qui permet l'exécution d'une fonction lorsqu'une touche est relâchée (voir en ligne https://www.w3schools.com/jsref/event_onkeyup.asp) :

```
cardExpiration.onkeyup = function(){
```

```
    // lecture entrées
    // traitement
    // écriture sorties
}
```

Pour les entrées, on lit le champ qui contient la date, et on le découpe en blocs de deux caractères au moyen de la fonction slice (https://www.w3schools.com/jsref/jsref_slice_string.asp):

```
    // lecture entrées
    var expirationMonth = (cardExpiration.value).slice(0,2);
    var expirationYear = (cardExpiration.value).slice(2,4);
```

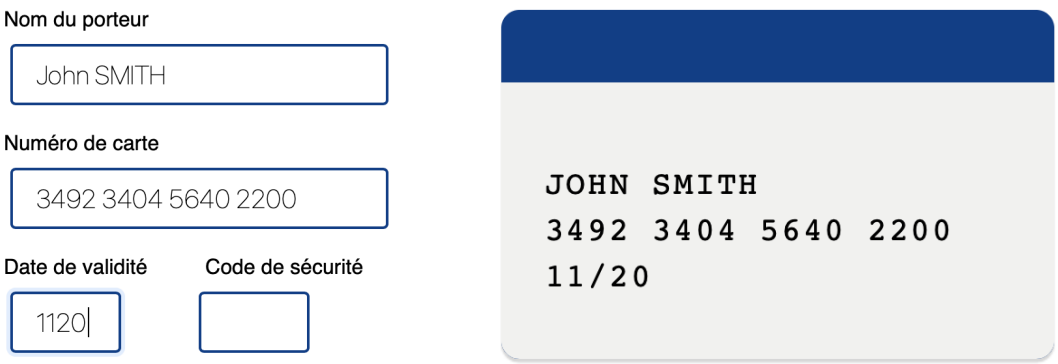
Coté traitement, il faut que le mois soit compris entre 1 et 12, et l'année entre 19 et 24. Attention, le test n'a de sens que sur un bloc complet, sinon on va perturber la saisie (au risque que la machine ne nous autorise jamais à saisir une valeur correcte). Imaginons qu'on applique systématiquement ce contrôle, pour saisir 22 il faut saisir 2, puis saisir 2. La fonction s'exécutant après chaque saisie on a alors : saisie du premier 2, contrôle du champ, rejet de la valeur car non comprise dans la bonne plage… Tout dépendra de ce qu'on fera dans le cas « rejet ». On peut essayer par exemple de remplacer le champ défaillant par des étoiles :

```
    if (expirationMonth.length==2) {
       var month = expirationMonth.valueOf();
       if (month <1 || month>12)
          expirationMonth = "**" ;
    }

    if (expirationYear.length==2) {
       var year = expirationYear.valueOf();
       if (year <19 || year>24)
          expirationYear = "**";
    }

    cardExpiration.value = expirationMonth + expirationYear;
```

De cette façon, une saisie incorrecte du mois ou de l'année remplace la partie concernée, au fur et à mesure de la saisie, pas des étoiles.

5► Modifiez le ficher HTML, insérez le script à la fin, puis testez. Faites bien attention à la syntaxe : chaque signe a son importance, y compris la différence majuscule-minuscule.

Une autre manière d'améliorer l'interface, est d'aider l'utilisateur à visualiser sa saisie. On pourrait ici représenter un carte bancaire vierge, et les mentions saisies par l'utilisateur apparaîtraient dessus. De cette façon, l'utilisateur pourrait mieux faire l'association entre l'objet réel qu'il a en sa possession, et les réponses qu'il donne car celles-ci apparaissent sur la carte affichée à l'écran. Un exemple, simplifié, est donné ci-après :



La carte est constituée d'un *div*, contenant 3 champs (un bloc *span*, identifié). La couleur de fond a pour coordonnées (241,241,239) dans l'espace RGB. Les coins sont arrondis (10px). La bande supérieure est obtenue simplement avec un contour, de couleur #0E3C87 et d'une épaisseur « suffisante ». Un petit ombrage en bas vient donner du relief (https://www.w3schools.com/CSSref/css3_pr_box-shadow.asp). L'écriture est en noir, police de la famille « *courier* », et en majuscule.

6► Codez, en HTML/CSS, la carte. Pour vous aider, vous pouvez mettre des valeurs arbitraires en HTML afin de visualiser le résultat.

Il reste à synchroniser la vue avec la saisie. Là aussi du javascript est nécessaire. Lors de la modification d'un champ (nom, prénom, numéro de carte, date), il faut recopier la valeur saisie, après contrôle, dans la zone HTML *span* associée. Dans le cas du numéro de carte par exemple, on aurait :

```
<div class="credit-card">
  <span id="card-name-display"></span>
  <span id="card-number-display"></span>
  <span id="expiration-display"></span>
</div>
```

et

```
<script >

   var cardNumber     = document.getElementById('card-number');
   var cardNumberDisp = document.getElementById('card-number-display');

   cardNumber.onkeyup = function(){
       cardNumberDisp.innerHTML = cardNumber.value;
   }

</script >
```

Le script ci-dessus nomme le champ de saisie du formulaire, et le champ d'affichage. Puis, il installe une fonction callback sur le champ de saisie, sur le retour d'appui d'une touche, de façon à recopier dans le champ d'affichage la valeur saisie dans le champ de saisie.

7► Modifiez le HTML de façon à ce que la vue de la carte soit synchronisée avec la saisie.

◊ ◊ ◊ Fin du sujet ◊ ◊ ◊