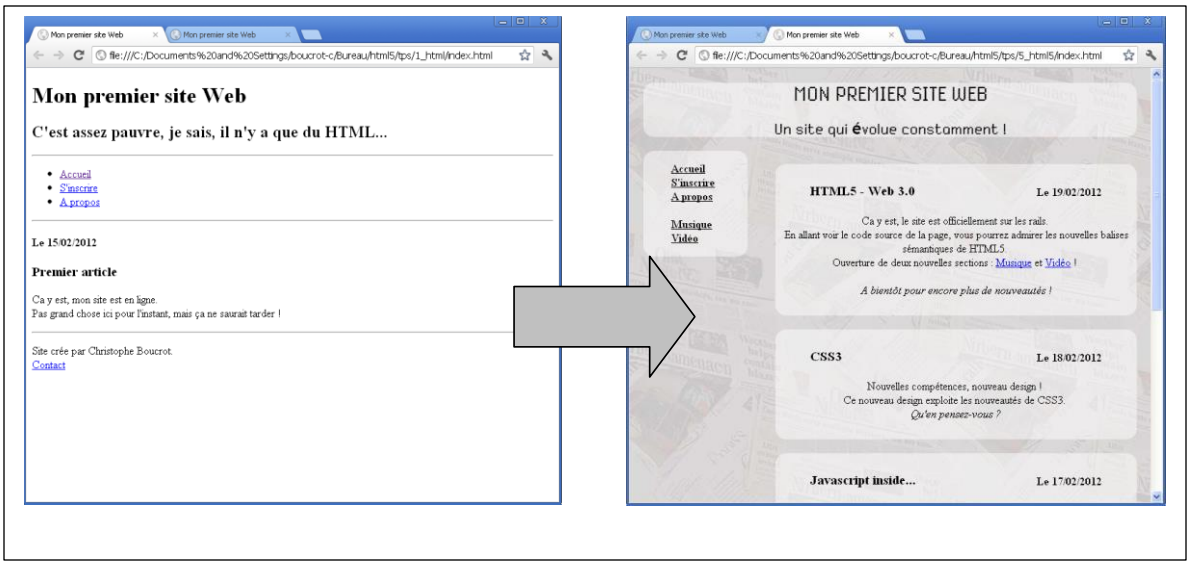
Exercice4 : HTML

L'objectif des TPs va être de créer un site Web simple, et de le faire évoluer au fil des TPs.

La première version sera en HTML pur, et la version finale sera en HTML5/CSS3/JavaScript.



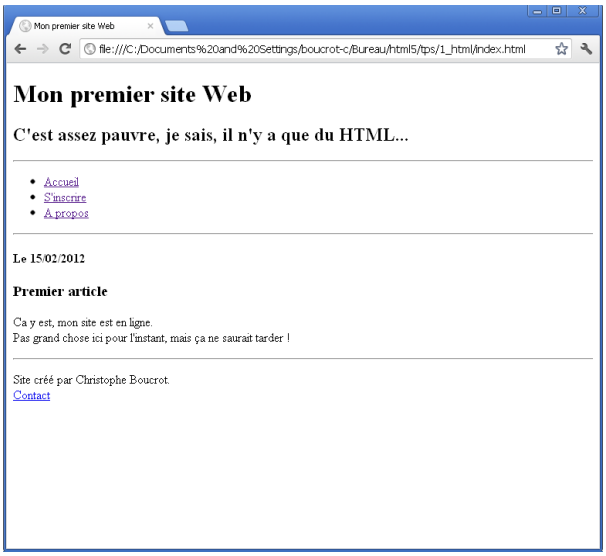
Etape1 : pages HTML

Dans ce TP, nous allons :

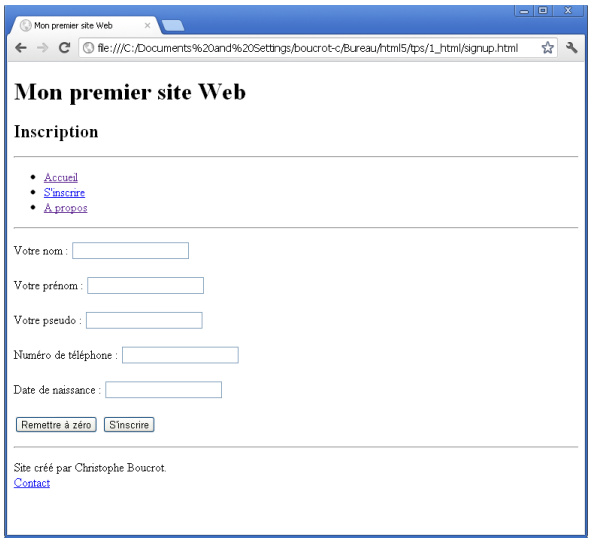
- Créer des fichiers HTML (<html> <head></head> <body></body> </html>)
- Structurer nos pages ( )
- Déclarer des balises :
  - De titres (<h1>, <h2>, <h3>, …)
  - De liste à puces (<ul>, <li>)
  - De lien (<a>)
  - De paragraphe (<p>)
  - De formulaire (<form>, <label>, <input>, ...)

Nous allons ici créer notre site de base. Ce site sera composé de trois pages HTML :

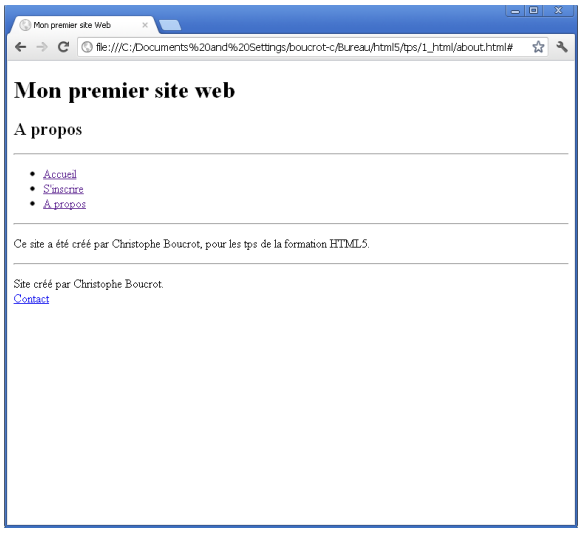
- index.html : C'est la porte d'entrée de notre site Web :



- signup.html : Cette page contient un formulaire qui permettra aux utilisateurs de s'enregistrer :



- about.html : Cette page présente des informations relatives au site Web :



Ces pages sont constituées de 4 principales, séparés par des lignes graphiques (<hr>) :

 : commun aux trois pages

Cette partie contient un titre de niveau 1, et un titre de niveau 2.

 : commun aux trois pages

Cette partie contient une liste à puce de liens.

Ici, nous aurons le contenu de la page.

Pour index.html, ce contenu sera partitionné en sous-éléments .

(Rappel : Un id doit être unique, alors qu'une classe peut être partagée par plusieurs éléments.

Les blocs articles auront le même comportement. Pour les gérer simultanément, on les regroupe en une classe article.)

Pour signup.html, ce contenu sera un élément <form>.

Pour about.html, ce contenu sera un unique sous-élément contenant un bloc <p>.

Exemple d'article de index.html :

```
<div class="article">
<h4>Le 15/02/2012</h4>
	<h3>Premier article</h3>
	<p>
		Ca y est, mon site est en ligne.
		<br />
		Pas grand chose ici pour l'instant, mais ça ne saurait tarder !
	</p>
</div>
```

 : commun aux trois pages

Notre pied de page se composera simplement d'un bloc <p>, contenant un texte et un lien Contact (href="mailto: …").

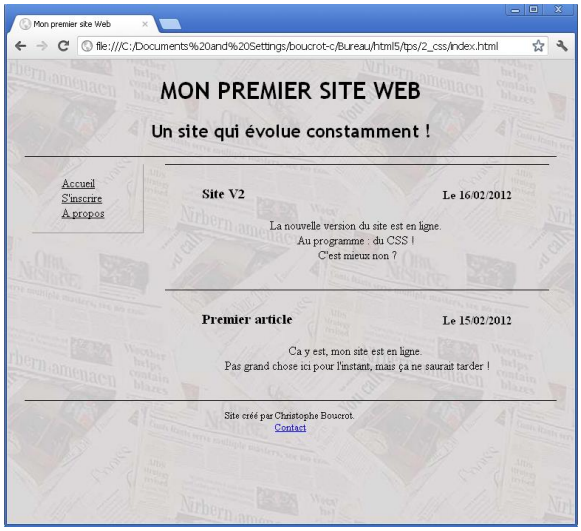
Votre travail consiste donc à créer ces trois fichiers html, puis de les lier dans le menu grâce à des liens relatifs.

Etape2 : pages CSS

Dans ce TP, nous allons :

- Créer une feuille de style commune à nos trois pages HTML
- Utiliser des sélecteurs simples (A ; A B ; A,B ; #monId ; .maClasse)
- Ecrire des règles CSS :
  - text-align
  - width
  - margin
  - background
  - font-family
  - border
  - color, …

Nous allons maintenant créer une feuille de style pour notre site Web. Voici le résultat que nous voulons obtenir :



La première chose à faire est de créer un fichier style.css. Ce fichier sera notre feuille de style externe, qu'il faudra déclarer dans toutes nos pages HTML :

Dans le bloc <head> de chaque page, ajoutez un élément :

<link rel="stylesheet" href="style.css" />.

La feuille de styles est maintenant déclarée dans nos pages HTML. Il faut maintenant écrire les règles de style :

- Pour l'élément <body> :
  - Centrage du texte
  - Largeur : 720 pixels
  - Marges : auto
  - Image de fond : url("img/paper019.jpg")

(Vous trouverez l'image dans le dossier « départ » dans C:\cd-total\... Placez-la dans un répertoire img dans votre répertoire de travail.)

- Pour les éléments <hr> :
  - On ne veut pas que ces éléments apparaissent : display: none;

- Pour l'élément :
  - Polices : on veut que le texte s'affiche en Trebuchet MS si disponible, sinon Arial, et dans le pire cas, sans-serif.
  - Bordure inférieure : 1 pixel, « solid ».
  - Elément <h1> : Affichage en majuscules : text-transform: uppercase;

- Pour l'élément :
  - Flottant à gauche : float: left;
  - Largeur : 150 pixels
  - Marges : 10 pixels
  - Bordures droite et bas : 1 pixel, « outset »
  - Position fixe : position: fixed; (le menu se déplacera avec le scroll)
  - Eléments <ul> :
    - Texte aligné à gauche
    - Pas de puces : list-style-type: none;
  - Eléments <a> :
    - Couleur : noir

Etape3 : pages JS

Dans ce TP, nous allons :

- Créer un fichier js dédié à notre page signup.html
- Ecrire une fonction valider(), appelée lors de la validation du formulaire par l'utilisateur, qui renvoie true si tous les champs sont remplis et valides, false sinon. Cette fonction devra tester le contenu de chaque champ
- Utiliser les expressions régulières

On veut maintenant valider le contenu de notre formulaire avant de l'envoyer au serveur :

- Tous les champs doivent être remplis.
- Le contenu du champ « Numéro de téléphone » doit être de la forme 0XXXXXXXXX.
- Le contenu du champ « Date de naissance » doit être de la forme XX/XX/XXXX.

- Dans le cas ou l'un des champs ne répond pas aux attentes, on désire colorer le label correspondant en rouge.

Nous devons tout d'abord créer un fichier validation.js dans notre répertoire de travail, contenant une fonction valider(). Cette fonction devra renvoyer true si tous les champs sont corrects, false sinon, afin d'empêcher l'envoi de données non valides au serveur.

Ce fichier doit être déclaré dans la page signup.html :

Dans le <head>, ajoutons <script src="validation.js"><script>.

La validation du formulaire par l'utilisateur doit appeler la fonction valider() :

Dans la balise <form>, ajoutons l'attribut onsubmit="valider()".

Les liens sont faits. Il faut maintenant implémenter cette fonction valider() :

- Nous devons récupérer l'ensemble des champs du formulaire :

var elements = document.forms[0].elements;

- Pour renvoyer false si un ou plusieurs champs ne sont pas valides, nous avons besoin d'un booléen local (que l'on appellera boolValid) initialisé à true, que l'on affectera à false quand un champ ne sera pas correct. C'est ce booléen qui sera retourné par la fonction.
- Il faut ensuite tester pour chaque champ que le contenu n'est pas vide :

Pour tester si un champ en particulier est vide (ex : pour le champ « nom ») :

if(elements['nom'] === "") { }

  - Si le champ est vide, on utilise DOM pour récupérer le label correspondant et changer sa couleur à rouge. Il faut aussi affecter false à boolValid.

    Ex. si le champ nom est vide :

    document.getElementById('label_nom').style.color = "red";

  - Sinon, on utilise DOM pour forcer sa couleur à noir (dans le cas ou le champ était vide avant).
- Enfin, on retourne boolValid.

Notre page signup.html est maintenant dynamique. Les champs sont vérifiés, et les labels modifiés, sans échanges client/serveur.

La prochaine étape est de valider le format du numéro de téléphone et de la date de naissance entrés.

Pour se faire, nous allons utiliser les expressions régulières.

(Pour plus d'informations sur les expressions régulières, http://fr.wikipedia.org/wiki/Expression_rationnelle).

En JavaScript, les expressions régulières sont similaires à des chaînes de caractère, mais encadrées par des slashs (/) en lieu et places des guillemets.

- Ici, nous déclarerons deux expressions rationnelles :

```
var regTel = /^0[0-9]{9}$/;

var regNais = /^[0-9]{2}\/[0-9]{2}\/[0-9]{4}$/;
```

- Pour tester une chaîne, on utilisera monExpressionReg.test(maChaine); qui renverra true si la chaine vérifie le motif, false sinon.

Ex : pour le champ numéro de téléphone, le test complet sera :

```
if( (elements['nais'].value === "") || !(regNais.test(elements['nais'].value)) ) {
        //coloration du label en rouge
        //boolValid = false
} else {
                //coloration du label en noir
}
```

A vous de jouer !