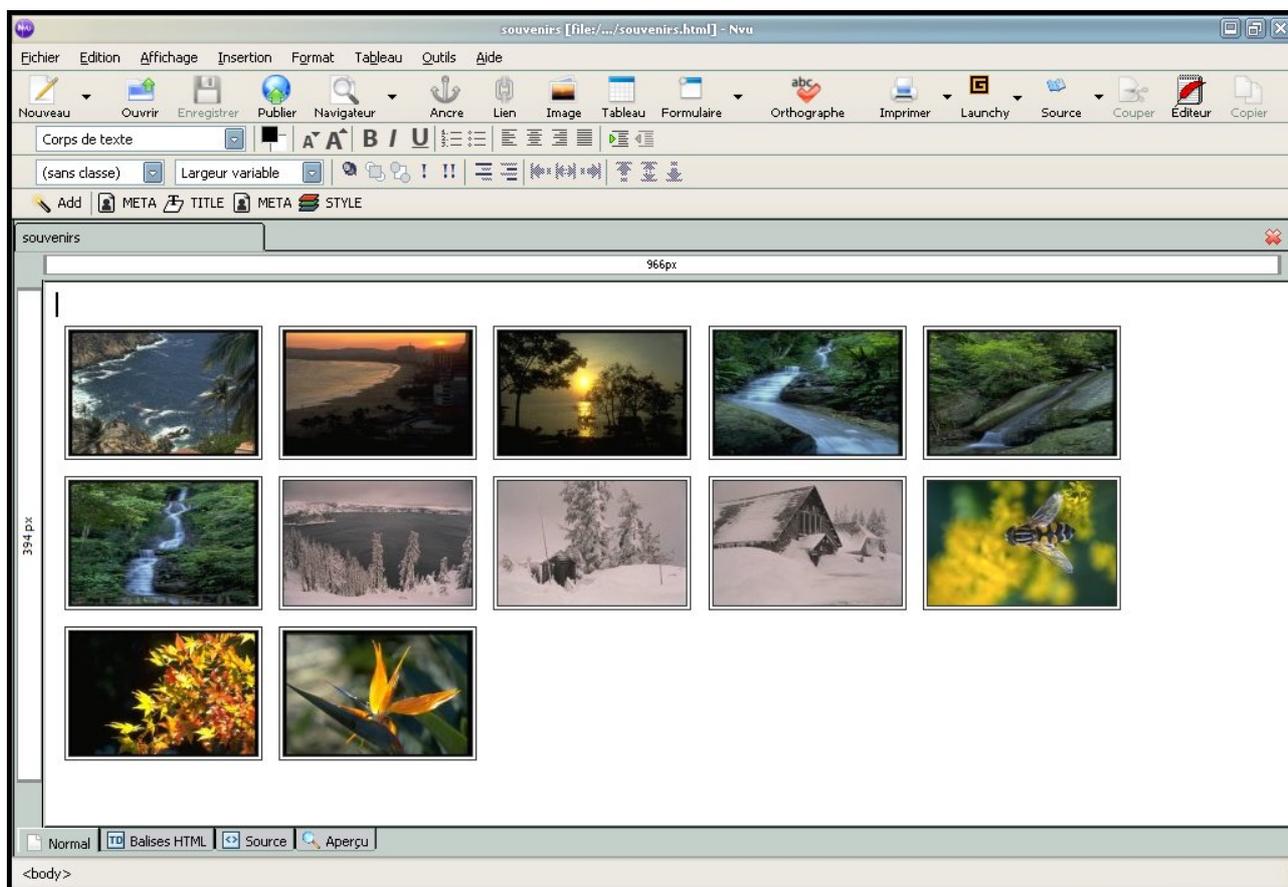


Dans la série  
**Les tutoriels libres**  
présentés par le site FRAMASOFT



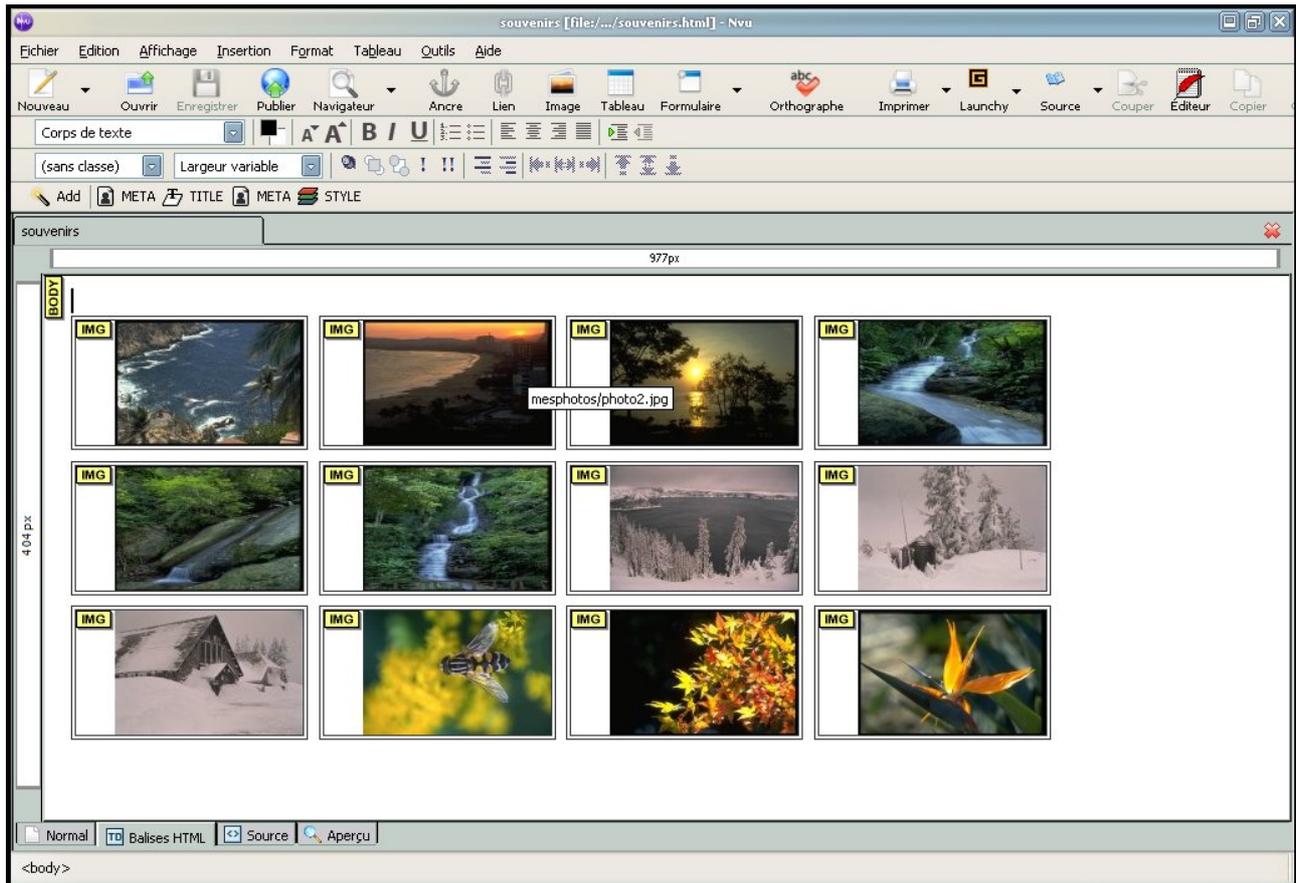
# N|VU Modulaire

*Ze|N tutoriel*

<b>Logiciel:</b>	<b>Nvu</b>
<b>Auteur(s):</b>	<b>Lindows Inc.</b>
<b>Plateforme(s):</b>	<b>Linux, Windows</b>
<b>Version:</b>	<b>1</b>
<b>Licence:</b>	<b>mpl/gpl/lgpl</b>
<b>Site:</b>	<b><a href="http://nvu.com">http://nvu.com</a></b>

## **Framasoft**

« Partir de Windows pour découvrir le libre... »  
<http://www.framasoft.net>



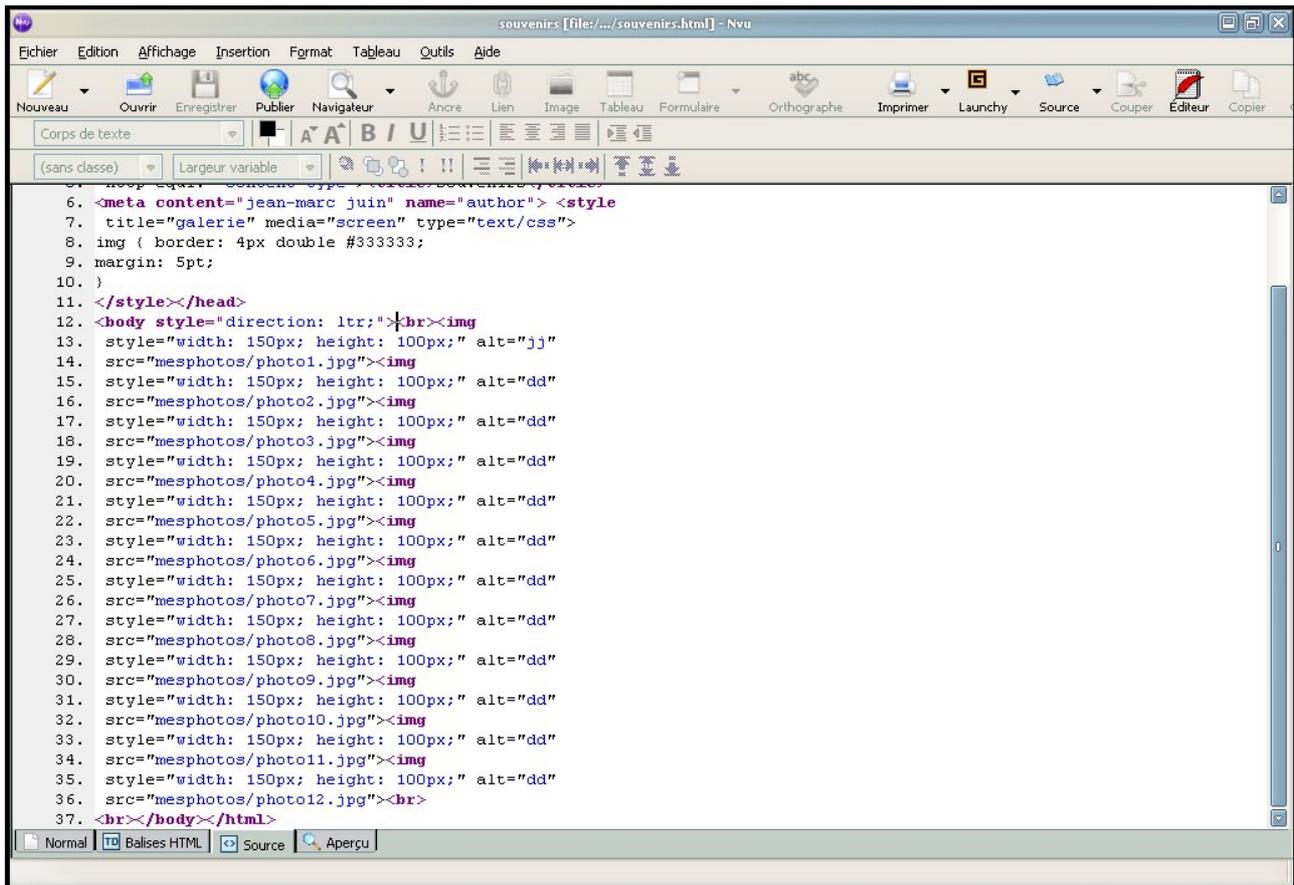
Par

jean-marc juin (fun sun)

janvier 2005 (Nvu-Modulaire-première version)

mai 2005 (deuxième version- plus légère, enfin presque...)

janvier 206 (modification de la licence + une nouvelle extension : hand coder)



```
6. <meta content="jean-marc juin" name="author"> <style
7. title="galerie" media="screen" type="text/css">
8. img { border: 4px double #333333;
9. margin: 5pt;
10. }
11. </style></head>
12. <body style="direction: ltr;"><br><br>
37. <br></body></html>
```

## Correcteurs

Christian (Choul)

Goofy

Basé sur la version francisée 1



Publié sous licence **Creative Commons By-ShareAlike- V2**

<http://creativecommons.org/licenses/by-sa/2.0/fr/>

Ce document sera mieux lu avec la police bitstream, la télécharger à cette adresse :

<http://ftp.gnome.org/pub/GNOME/sources/ttf-bitstream-vera/1.10/>

## Table des matières

<b>A) Les extensions .....</b>	<b>6</b>
1) Présentation .....	6
2) Changements importants.....	6
3) Pour un site Web de qualité.....	6
3.1) Extensions supplémentaires non Nvu.....	7
4) Les différents modes de balisage.....	7
4.1) Les différents accès au balisage : les règles de style.....	8
4.2) Les différents accès au balisage : Javascript et DOM.....	9
5) HTML, CSS et Javascript.....	11
<b>B) L'extension Htmlheader.....</b>	<b>12</b>
1) Petit rappel entête et corps de page en langage HTML.....	12
1.1) Où trouver les extensions pour Nvu ? .....	13
1.2) Installation.....	13
2) Problèmes détectés.....	14
<b>C) Exerçons-nous.....</b>	<b>14</b>
1) Modifications des règles CSS.....	14
2) Ajout de données méta pour faciliter une recherche.....	15
3) Résoudre le problème du positionnement pour les navigateurs basés sur Internet Explorer.....	17
3.1) Reprenons les données du problème .....	17
3.2) La solution au problème tient en une ligne .....	19
3.3) Procédure d'installation.....	20
3.4) Insérez le code avec Htmlheader.....	21
3.5) problème réglé !.....	22
<b>D) Launchy, Viewsourcewith, handcoder.....</b>	<b>23</b>
1) L'extension Launchy.....	23
2) L'extension Viewsourcewith.....	24
3) L'extension HandCoder.....	24
<b>E) Aborder l'insertion de Javascript.....</b>	<b>25</b>
1) Création d'une fenêtre pop up.....	25
1.1) Insertion du code.....	26
1.2) Vérification du code à l'aide de la console Javascript.....	27
2) Créer des Formulaires et vérification avec javascript.....	28
2.1) Ou'est-ce qu'un formulaire ? .....	29
2.2) La méthode choisie.....	29
2.2.1) Méthode suivie.....	30
2.3) Créer un liste de choix pour simplifier la navigation dans un site Internet. ....	30
2.3.1) Définir les mentions générales du formulaire .....	31
2.3.2) Création du bouton de navigation.....	32
2.3.3) Création de la liste de choix .....	34

2.4) Code source complet du formulaire .....	35
3) Insertion du code Javascript.....	37
3.1) Explication du code Javascript.....	38
3.2) L'explication du code.....	38
<b>F) Réaliser un formulaire interactif.....</b>	<b>41</b>
1) Définition générale du formulaire.....	42
1.1) Définition de l'événement à l'aide de JavaScript.....	43
1.2) Définir des champs particuliers.....	44
1.2.1) Les champs texte nom et prénom.....	44
1.2.2) Les boutons envoyer et annuler.....	45
1.2.3) Le champ Zone de texte.....	46
1.3) Définition d'étiquettes relationnelles pour les champs nom et prénom.....	47
1.3.1) Définir les propriétés de champs des formulaires. ....	48
1.3.2) Styliser votre formulaire avec l'éditeur Cascades .....	49
2) Vérifier les saisies du formulaire à l'aide de Javascript.....	49
2.1) Courte présentation du code à insérer.....	50
<b>G) Et si on passait au XHTML Strict ?.....</b>	<b>52</b>
<b>H) Conclusion.....</b>	<b>54</b>
1) Pour finir... et continuer !.....	54

## A) Les extensions

---

### 1) Présentation

Nvu accepte désormais l'intégration de modules externes ou extensions qui étendent les fonctionnalités du programme. C'est l'occasion d'approfondir l'étude du logiciel et de voir jusqu'où il peut nous aider dans la création de pages Web.

Ce tutoriel est donc la continuité du précédent mais supposera que vous possédez et maîtrisez les bases du langage HTML, du logiciel ainsi que les bases de manipulation de fichiers pour l'installation des extensions. Les descriptions des fenêtres seront beaucoup plus succinctes et iront à l'essentiel.

### 2) Changements importants

Tout comme la version standard plusieurs parties de la version modulaire ont subi une réécriture plus ou moins complète bien que les changements soient moins importants.

### 3) Pour un site Web de qualité

Nous allons étudier ou plutôt voir les possibilités d'extensions du programme à travers :

- **htmlheader**, pour manipuler plus facilement l'entête d'une page HTML,
- **Launchy** (Ou NsmContext, extension plus évoluée basée sur Launchy) pour vérifier la construction de vos pages dans plusieurs navigateurs différents et
- **Viewsourcewith** pour manipuler la source HTML avec plus de précision.
- **Handcoder** une autre extension pour manipuler la source HTML avec tidy et la prise en charge des modifications !

Ces extension nous permettent de peaufiner pas mal de choses et donc de rendre nos pages Web plus performantes et de prendre un tour quasi professionnel.

### 3.1) Extensions supplémentaires non Nvu

Si vous possédez un navigateur basé sur Gecko (Mozilla, Firefox) je ne peux que vous conseiller fortement l'installation de l'extension webdevelopper. Voyez la présentation sur le site de Framasoft. ([www.framasoft.net](http://www.framasoft.net))

Vous consulterez, avec profit, le S5, disponible sur Framasoft dédié au développement web avec Firefox, une multitude d'extensions plus qu'utiles !!!

## 4) Les différents modes de balisage

La version 1 de Nvu alloue le choix de plusieurs modes de balisage soit HTML 4.01 de transition ou strict, soit XHTML de transition ou strict. Pour des raisons pratiques nous restons dans ce mode ou vous pouvez passer en mode HTML 4.01 strict.

La différence entre le langage HTML 4.01 et ses balises et le langage XHTML réside dans la conception du balisage. Les balises pour HTML ne sont pas obligées de respecter un ordre logique (sémantique).

```
> <p><strong>Ma voiture</p></strong>
```

Tandis qu'avec XHTML les balises doivent respecter la sémantique (ordre logique) de construction.

```
><p><strong>Ma voiture</strong></p>
```

## 4.1) Les différents accès au balisage : les règles de style

Le langage XHTML distingue très clairement contenu (Les balises structurantes de la page) et le contenant (le style appliqué au document entier et à chacune des ses balises.)

Cette distinction ou, plutôt préférence, donne un accès aux balises HTML beaucoup plus facile. Pour accéder à une balise et définir son style, il suffit de nommer cette dernière dans la partie style. Body pour le corps entier de la page, p pour les paragraphes de la page.

Pourquoi accéder à une balise ? Pour modifier son apparence d'une part avec le langage CSS et d'autre part en influencer le comportement suivant telle ou telle action avec l'utilisateur avec Javascript d'autre part.

```
><html>

<head>

<style>

body {background-color:rgb(242,242,242); color : rgb(51,51,51);
padding: 3pt;}

p {text-indent:1.5em; padding-top:0.5em; padding-bottom:0.5em;}

</style>

</head>

<body>

<p><strong>Ma voiture</strong></p>

</body>

</html>
```

Je viens de définir une page de couleur blanche (*background-color*) mais légèrement neigeuse pour ne pas être ébloui ; une couleur de texte grisonnante (*color*), un écart de 3pt par rapport au bord intérieur de la page (*padding*) ; ce qui évite que le texte ne colle aux bords du bloc *body* si cela avait été par rapport aux bords de la fenêtre du navigateur, j'aurai choisi (*margin*).

Enfin pour chaque paragraphe, j'ai choisi de décaler le début chaque première ligne de 1.5em (*text-indent*) par rapport au reste du paragraphe, puis je m'assure que l'écart haut et bas de chaque paragraphe soit suffisant (*padding-top; padding-bottom*).

Bref quelques règles de style simples et bien définies peuvent rendre au documents Web de grands services quant au confort de lecture.

## 4.2) Les différents accès au balisage : Javascript et DOM

Javascript est un langage qui permet d'influencer le comportement de telle ou telle partie des balises de la page. On notera que les règles CSS permettent aussi, mais dans une moindre mesure, d'influencer le comportement de certaines balises comme la création de boutons hyperliens dynamiques en CSS pur grâce au pseudo élément `:hover` par exemple. Ou aux propriétés d'affichage (`display: block`, `display: none`) lesquelles peuvent faire apparaître ou disparaître un bloc identifié clairement.

Toutefois javascript intervient dans d'autres domaines (création d'une fenêtre d'information ou pop up ; vérification de la validité de formulaires ; création dynamique d'éléments supplémentaires à la page en fonction du type de navigateur, etc).

En Javascript on accède de la même manière à une balise, il suffit de préciser laquelle.

```
➤ Function essai()  
  
  {  
  
    if (Document.getElementsByTagName('p'))  
  
    {  
  
      faire ceci  
  
    }  
  
  }
```

Je pose un test qui vérifie si je peux accéder à la balise 'p' du document. Si dans le document on peut obtenir les éléments par le nom de leur balise qui est 'p' alors il faut faire ceci. Sinon il ne faut rien faire. Le sinon est inclus implicitement

dans le fait que l'accolade de fin de la fonction est juxtaposée à celle de la fin de la condition et qu'il n'y a aucune instruction entre.

DOM est une spécification du w3c, organisme qui propose et pose les règles de définition des langages HTML, XHTML, CSS pour ceux qui nous intéressent. Le DOM est un Javascript spécifique lié aux documents XML dont fait partie le langage XHTML. Il peut créer, générer, enlever dynamiquement des balises suivant un langage particulier (que nous verrons pas ici, il est relativement complexe).

```
{
var x = document.createElement('dt');
x.appendChild(document.createTextNode('Table des matières : '));
var y = x.appendChild(document.createElement('a'));
y.setAttribute('href','javascript:tdmdisp()');
y.className='link';
y.appendChild(document.createTextNode('enlever')); }
```

Voici une version en javascript classique qui produit la même chose :

```
{
var placerdansid = document.getElementById('tdm');
var ajouterceci =("<dt>Table des mati&egrave;res <a
href=\"javascript:tdmdisp();\" class=\"link\">Enlever</a>
</dt>");
placerdansid.innerHTML += ajouterceci;
}
```

Le premier exemple (DOM) considère chaque élément à ajouter ou à enlever comme un noeud du document (Document.createElement('dt');). Il est ajouté dans le document une nouvelle balise 'dt' à laquelle on va accoler un texte :

x.appendChild(document.createTextNode('Table des matières : ')); et ainsi de suite.

Dans le second exemple (Javascript classique) il est ajouté des éléments

HTML à la page sans que ceux-ci n'appartiennent réellement à la structure de la page alors qu'avec DOM ils entrent dans la structure de la page. (Si je ne me trompe pas).

## 5) HTML, CSS et Javascript

Ils forment les outils de base et essentiels pour la réalisation de pages web de qualité côté client, c'est-à-dire les modifications, altérations sont directement exécutées dans le navigateur qui vient rendre visite à votre site. Un bon mélange des trois peut rendre un site beaucoup plus efficient et efficace.

Pour d'autres utilisations plus conséquentes mais côté serveur (généralement votre fournisseur d'accès) alors d'autres langages sont appelés comme le célèbre php.

Notez que les quelques bouts de Javascript vus dans ce tutoriel appartiennent à la deuxième catégorie d'exemple, nous laissons la première à des codeurs plus chevronnés que nous !

## B) L'extension Htmlheader

---

Attention ! Je garde la version 0.01 de htmlheader bien que la version 0.03 soit traduite et francisée. Elle ne m'a pas paru satisfaisante pour parler de cette nouvelle version. Chose rare me direz-vous, une nouvelle version devant être améliorée. Ce n'est pas le cas.

### 1) Petit rappel *entête* et *corps de page* en langage HTML

La structure basique d'une page HTML est composée de deux parties ou éléments essentiels à l'intérieur des balises générales <html></html>. La première partie correspond à l'*entête* ou <head></head> et la seconde au *corps de page* ou <body></body>. Voici le code source d'une page vierge au format xhtml

```
><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="fr">

<head><meta content="text/html; charset=ISO-8859-15" http-
equiv="content-type" />

  <title></title>

<meta content="jean-marc juin" name="author" /></head>

<body style="direction: ltr;"><br />

</body>

</html>
```

C'est sur le *corps de page* où nous intervenons le plus souvent, il contient toutes les informations nécessaires au contenu que nous souhaitons transmettre. Avec un outil en mode *WYSIWYG*, nous aurions tendance à oublier l'autre partie importante d'une page HTML, l'*entête*.

Comme nous l'avons vu dans le précédent tutoriel et nous le verrons dans celui-ci l'*entête* d'une page peut contenir de nombreuses informations complémentaires et souvent très utiles.

Dans sa version standard Nvu ne nous permet pas d'intervenir aisément dans l'entête d'une page sans passer par le code source. Grâce à cette extension, insérer ou modifier des données dans l'*entête* d'une page est facilité.

### 1.1) Où trouver les extensions pour Nvu ?

A cette adresse <http://www.geckozone.org/extensions/>

L'ensemble des extensions sont répertoriées sur ce site, véritable ressource française du monde gecko (Mozilla, firefox, nvu) d'où son nom.

### 1.2) Installation

L'installation est des plus simples. Il suffit d'aller dans le menu Outils/Extensions et cela se fait tout seul. Pensez à redémarrer Nvu pour que l'extension s'affiche correctement.

L'extension *HtmlHeader* se présente sous la forme d'une nouvelle barre d'outils. Suivant le contenu préalable de l'entête de la page courante, un plus ou moins grand nombre d'icônes apparaîtra automatiquement.



Cette barre d'outils se compose de deux parties essentielles.

**En mauve :** Le bouton *Add* (*Ajouter*)

**En jaune :** une série d'icônes qui correspondent aux informations trouvées dans la page active. Ici, nous avons un titre de page (*Title*), deux informations de données Meta (*Meta*), une information sur le style (*Style*), ainsi qu'un script (*Script*).

Il suffira de cliquer soit sur *Add* pour ajouter quelque chose de nouveau soit

sur une des icônes pour modifier le contenu correspondant au nom de l'icône.

## 2) Problèmes détectés

Un des plus gros défauts de l'extension est de supprimer l'ensemble des styles une fois ces derniers modifier avec l'extension. Ne paniquez pas ! Cet effacement est temporaire.

Il suffit d'enregistrer vos modifications, de fermer la page puis de la recharger et le problème est résolu. Un peu ennuyeux tout de même...

## C) Exerçons-nous

---

### 1) Modifications des règles CSS

Si l'éditeur CSS nous aide beaucoup toutes les propriétés CSS ne sont pas intégrées. Prenons deux exemples concrets :

- Le premier consiste à apporter une indentation, ou décalage de la première ligne de paragraphe, de 1,5em par exemple. Les boutons d'indentation (*augmenter/diminuer le retrait*) créent un conteneur. Ce qui ne nous intéresse pas. Ce que nous voulons, c'est attribuer d'une manière générale à chaque paragraphe une indentation. C'est là où l'extension intervient, il me suffit de cliquer sur le bouton style et d'ajouter à p une indentation de 1,5em.

```
➤ P {text-indent:1.5em;}
```

Chaque paragraphe est maintenant stylisé comme je le souhaite.

- Le second revient à attribuer à toutes les listes un même format. Avec le dialogue des *Propriétés de Liste*, le choix de type de liste est plutôt facile cependant ce choix est attribué pour la liste qui m'intéresse ; je souhaite, au contraire, formater toutes mes listes de la même manière et, si possible, avec des items de liste Grecs.

→ Format de liste général : attribution du format alphabétique minuscule

Au moment de créer la liste, je n'interviens que sur le style de liste ; ici numérotation. Ensuite, il suffit de cliquer sur le bouton style de la barre

htmlheader afin de définir toutes mes listes de la même manière.

```
→ ol {list-style-type : lower-alpha;}
```

→ Format de liste général : attribution d'un format alphabétique grec minuscule

Idem. Seule la propriété change

```
→ ol {list-style-type : lower-greek;}
```

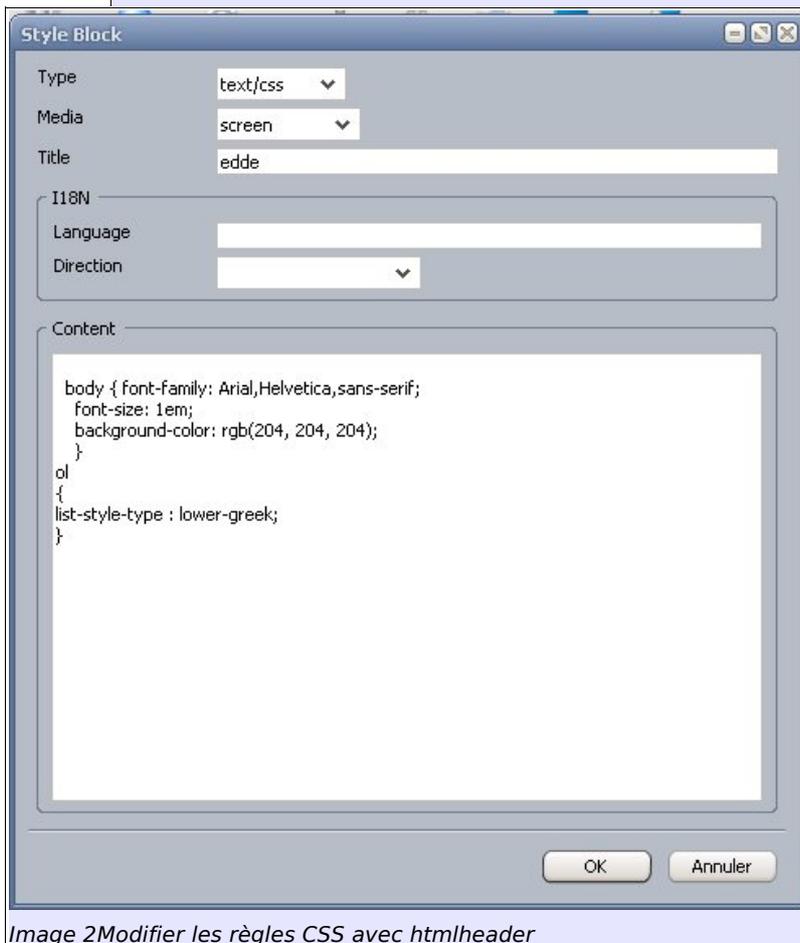


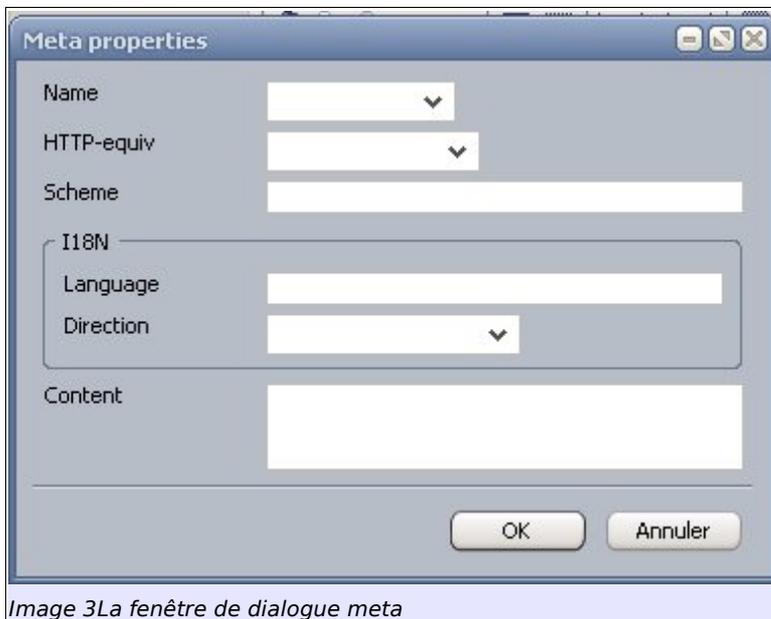
Image 2 Modifier les règles CSS avec htmlheader

## 2) Ajout de données méta pour faciliter une recherche

Je souhaite donner quelques indications aux programmes robots des  *moteurs de recherche*  qui visitent mon site afin de leur permettre de mieux indexer mes pages. Par exemple, je souhaite qu'ils re-visitent la page tous les 10 jours et je

les autorise à suivre l'ensemble des pages de mon site afin qu'elles soient mieux indexées. De plus je vais créer une liste de mots clefs pour faciliter la recherche.

Pour ajouter tout cela, il suffit de cliquer sur le bouton *Add* puis de glisser vers le dialogue *Meta...* que voici :



Reprenons les données :

- Revisiter tous les 10 jours

Il suffit d'inscrire dans le champ *Name* : **revisit-after** et dans le champ *Content* : **10 days**.

- Indexer et suivre l'arborescence de mon site

À nouveau, il faut cliquer sur *add/Meta...* (création d'une nouvelle balise Meta). Puis inscrire dans le champ *Name* : **robots** et dans le champ *Content* : **index, follow**

- Création de mots clefs afin de faciliter la recherche

Encore une fois, il faut ajouter une nouvelle balise Meta, sauf qu'ici dans le champ *Name* le mot **keywords** (pré-sélection) sera choisi et, inscrire dans le champ *Content*, une série de mots clefs séparés par une virgule.

Il y a d'autres possibilités encore. Nous n'en voyons que quelques unes. Je vous renvoie au site [open-web](http://www.open-web.com) qui vous expliquera quelles sont les données meta les plus utiles à renseigner.

### 3) Résoudre le problème du positionnement pour les navigateurs basés sur Internet Explorer

Dans le précédent tutoriel, Nvu-Standard, nous avons vu à la section G, *Étude de cas : création d'un menu de navigation*, comment créer facilement un menu de navigation simple. Cependant nous avons soulevé le problème suivant.

Un menu fixé et positionné à l'aide des feuilles de style est parfaitement supporté par les navigateurs qui respectent les standards tandis que les navigateurs basés sur Internet Explorer, notamment pour les versions 5 et 6, ce support est plutôt mauvais et nous empêche d'employer cette technique pourtant si simple et facile d'utilisation.

#### 3.1) Reprenons les données du problème

A l'aide d'une liste de définition, nous avons créé un menu de navigation. Voir l'image ci-dessous :



Image 4 menu navigation brut

Voici le code HTML :

```
><dl>
<dt>menu</dt>
<dd><a href="#">ma page un</a></dd>
<dd><a href="#">ma page deux</a></dd>
<dd><a href="#">ma page trois</a></dd>
<dd><a href="#">ma page quatre</a></dd>
</dl>
```

Par la suite nous avons stylisé cette liste de définition pour lui donner un aspect plus agréable à l'oeil

```
>dl { border: 4px double rgb(255, 255, 255); /*style général à la
liste de définition*/
padding: 5pt;
background-color: rgb(153, 0, 0);
color: rgb(255, 255, 255);
background-image:
url(file:///C:/tutoriel_Nvu/acrose/img/fleur.png); /*il faudra
transformer l'adresse physique en adresse relative*/
background-position: center;
width: 9em;
}

dt { border: 1px solid rgb(51, 51, 51); /*style pour le titre
menu*/
font-size: 1em;
font-family: Verdana;
text-align: left;
background-image:
url(file:///C:/tutoriel_Nvu/acrose/img/trans.png); /*idem pour
cette image*/
padding-left: 5pt;
color: rgb(255, 204, 51);
font-weight: bold;
}

dd { border: 1px none rgb(51, 51, 51); /*style pour les éléments
définis*/
padding-left: 1pt;
color: rgb(255, 255, 255);
font-weight: bold;
}

a { text-decoration: none; /*style pour les liens*/
color: rgb(255, 255, 255);
}

a:hover {color: rgb(51, 51, 51); /*style quand le lien est activé*/
}
```

Voici une photo du résultat obtenu :



Enfin nous avons ajouté un code CSS afin de déterminer la position de ce menu par rapport au reste de la page et nous avons précisé que ce dernier sera fixe.

```
>dl { border: 4px double rgb(255, 255, 255); /*style général à la
liste de définition*/
padding: 5pt;
background-color: rgb(153, 0, 0);
color: rgb(255, 255, 255);
background-image:
url(file:///C:/tutoriel_Nvu/acrose/img/fleur.png); /*il faudra
transformer l'adresse physique en adresse relative*/
background-position: center;
width: 9em;

position : fixed; top : 5%; left : 1%;
}
```

Nous ajoutons ce code à la balise mère de la liste de définition, la balise <dl>.

### 3.2) La solution au problème tient en une ligne

Vous pouvez vérifier aisément ce problème de lecture avec l'extension-Launchy (voir ci-dessous). Avec Internet explorer version 6, le menu n'est pas fixe et avec Internet Explorer version 5, c'est encore pire.

Que faire ?

La solution nous vient d'un script très astucieux nommé ie7. Il tient en une trentaine de kilos. Vous rendez-vous vous compte ? 30 kilos supplémentaires pour

obtenir un meilleur support des feuilles de style, ce n'est rien comparé aux milliers de lignes pour faire tourner un programme qui se mesurent en mega-octets!!!

Écrit par Dean Edwards (<http://dean.edwards.name/ie7>) ce script nous offre le support CSS qui manque à Internet Explorer version 6 ; il faudra ajouter un second script, moins d'un kilo, pour le support adapté à Internet Explorer version 5.

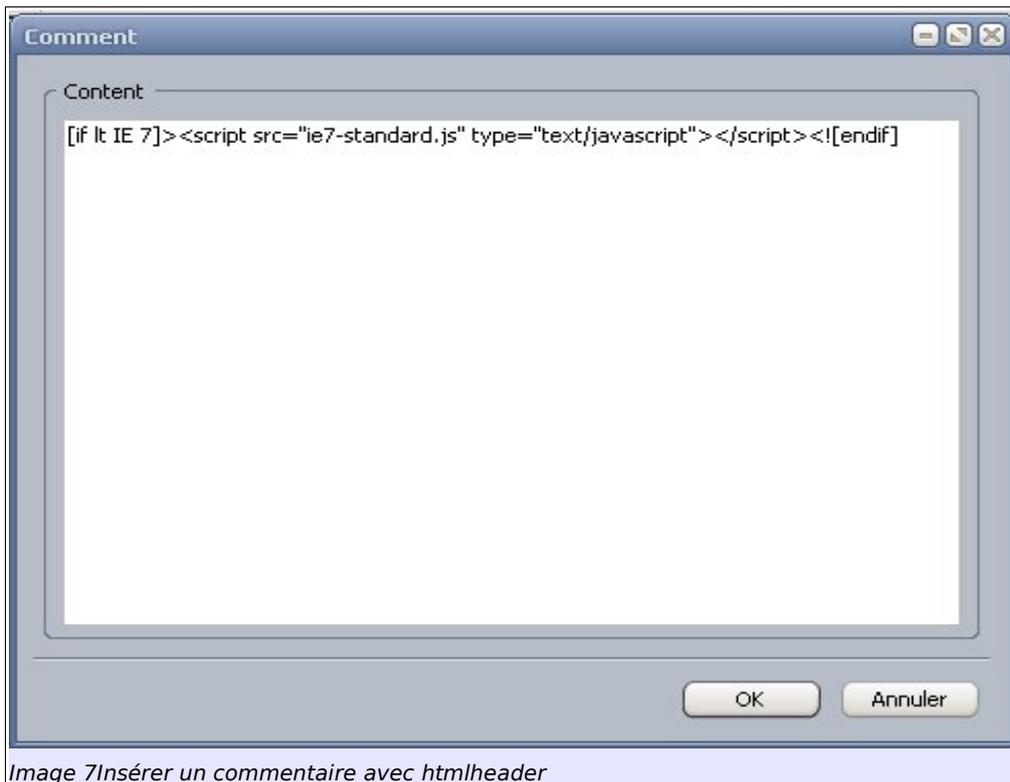
### 3.3) Procédure d'installation

Téléchargez l'archive au format zip et décompressez-la. Tout ce que vous avez à faire est de placer les deux fichiers suivants : ie7-standard.js et ie7-ie5.js dans le fichier de votre choix puis d'insérer le code suivant dans l'entête de la page à l'aide de Htmlheader.

**Attention !**

L'adresse proposée qui correspond à l'emplacement du répertoire scripts est dépendante de l'exercice. Il a été créé un pseudo-site (voir image 4 – ci-dessous) qui nous sert de base référentielle. Vous adapterez en conséquence l'adresse à vos propres besoins.





[if lt IE 7] ? Cette partie est un langage conditionnel propre à Microsoft qui signifie « Si la version est inférieure (lt = lower than) à Internet Explorer 7 » alors lancer le script à l'adresse suivante.

Étant donné que la version 7 d'Internet Explorer n'est pas encore apparue, forcément tous les navigateurs sont inférieurs à celle-ci et, en conséquence activent le script !

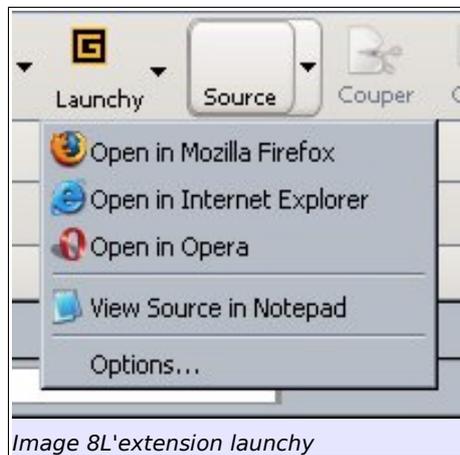
### 3.5) problème réglé !

Voilà un problème épineux réglé. La seule contrainte est l'insertion de ce code pour chacune de vos pages où un tel type de menu apparaîtra. Désormais les utilisateurs d'Internet Explorer 5 et 6 pourront voir la même chose que ceux qui possèdent des navigateurs plus performants sans qu'ils ne s'aperçoivent de quoi que ce soit.

En contrepartie cet effort ne les aidera pas à changer de navigateur puisqu'ils ne verront rien... À vous de choisir en connaissance de cause.

## D) Launchy, Viewsourcewith, handcoder

### 1) L'extension Launchy



Cette dernière vous permet de visualiser vos pages dans plusieurs navigateurs différents. Indispensable si vous souhaitez assurer une mise en page cohérente entre chaque navigateur.

La détection des applications est automatique. Le choix des applications à retenir ou non se trouve dans l'onglet général du menu option. D'autres options mais à réserver aux plus chevronnés !

Comment donc ? Ne m'a-t-on pas dit et rabâché que si j'utilisais une construction basée sur XHTML + CSS, j'étais sûr d'avoir une présentation identique et ce dans n'importe quel navigateur ?

C'est un argument souvent entendu et lu, sachez que pour des conceptions de pages un peu pointu c'est plus ou moins faux à cause du mauvais support CSS d'Internet Explorer d'une part et des différences d'interprétations entre chaque navigateur d'autre part. Si Mozilla et Firefox restent proches, Opéra ne réagit pas toujours de la même manière bien que les différences soient plus subtiles. Donc prenez garde ! Launchy vous aidera à vérifier qu'il n'existe pas de trop grandes différences ou des écarts d'interprétation entre chaque navigateur.

**Note aux utilisateurs de Linux :** L'extension ne fonctionne pas dans sa configuration par défaut, il faut la configurer avec le fichier launchy.xml. Voir le site officiel pour plus d'explications ainsi que le site de l'auteur de l'extension

qui a laissé un petit tuto très clair sur le sujet.

## 2) L'extension Viewsourcewith



Voici une autre extension intéressante mais optionnelle. Elle vous permet d'éditer la page en cours dans un éditeur externe autre que Nvu. Si dans les versions précédentes cela était quasi obligatoire pour rattraper et transformer le code en mode strict par exemple. Aujourd'hui ce n'est plus le cas.

Cependant, je fais partie de ceux qui ont appris à écrire des pages en mode texte. J'ai gardé certaines habitudes et j'aime bien éditer le code dans un éditeur en mode texte.

Cette extension est faite pour cela. Sa configuration est simple, il suffit de lui indiquer le chemin du ou des programmes à utiliser dans l'espace de configuration et le tour est joué.

### **Attention !**

Vous ferez attention que les modifications apportées avec un éditeur externe ne sont pas prises en compte par Nvu. Il faudra donc fermer la page et l'ouvrir à nouveau pour la prise en compte des modifications par Nvu.

## 3) L'extension HandCoder

Cette extension, comme la précédente, permet d'ouvrir et d'éditer votre

page HTML dans un éditeur externe. Quelle est la différence avec la précédente ? Son atout majeur est de prendre en compte les modifications apportées à votre page avec un éditeur externe même si vous ne l'éditez pas avec cette extension ! Sans compter qu'elle intègre aussi le correcteur Tidy ! De plus elle supporte les langages PHP, ASP, JSP. À utiliser sans modérations ! Merci Kaze !

Vous retiendrez que les arguments par défaut pour Tidy ne prennent pas en charge la correction des erreurs mais uniquement l'indentation du texte. Donc pensez à modifier les arguments !

## E) Aborder l'insertion de Javascript

---

### 1) Création d'une fenêtre pop up

Soit vous créez une nouvelle page, soit vous reprenez une ancienne page si vous avez suivi le précédent tutoriel. Vous créez une autre page HTML qui nous servira de fenêtre pop up que vous nommerez information.html par exemple.

Le script en langage Javascript et son code d'appel en HTML :

```
>function fenetrepopup(url)
{
    fenetre=document.open(url, 'infos', 'height=200,width=300,top=100,lef
t=100,toolbar=no,menubar=yes,location=no,resizable=yes,scrollbars=y
es,status=no');
    if (window.focus)
    {fenetre.focus()}
    return false;
}
><a href="information.html"
    onclick="return fenetrepopup('information.html')"
    title="fen&ecirc;tre d'information">Infos ?</a>
```

Attention !

J'ai, volontairement, inséré une erreur dans le script. Cette erreur nous servira ultérieurement lorsque nous vérifierons le code à l'aide de la console Javascript.

Merci de ne pas en tenir compte.

Ce script, très simple, ouvrira la page *information.html* dans une fenêtre dont nous déterminons l'aspect, taille (*width*, *height*), son emplacement par rapport à la fenêtre du navigateur (*top*, *left*), ensuite nous indiquons toute une série d'informations complémentaires :

voulons-nous la barre d'outils (*toolbar*), la barre des menus (*menubar*), la barre d'adresses (*location*), le redimensionnement de la fenêtre (*resizable*), les barres de défilement (*scrollbar*), la barre des tâches (*status*) ?

La seconde partie du script s'intéressera au focus de la page et évitera que celle-ci ne s'affiche comme une page HTML classique.

### 1.1) Insertion du code

Dans la barre htmlheader, cliquez sur *Add* puis sur *Script* insérez le code vu ci-dessus :

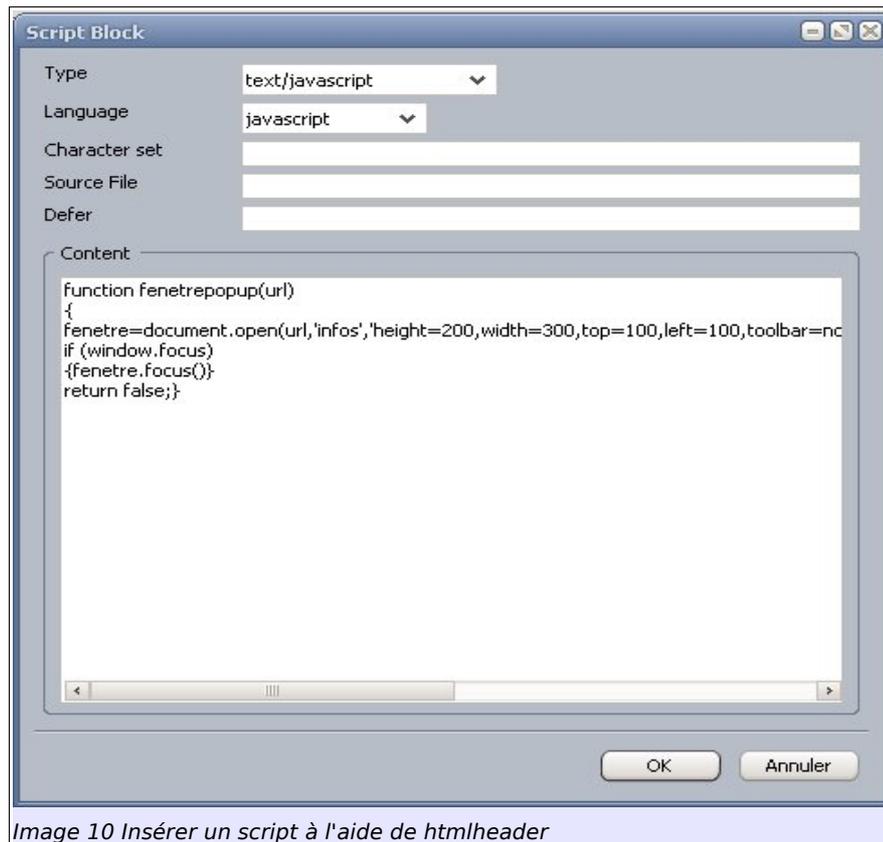


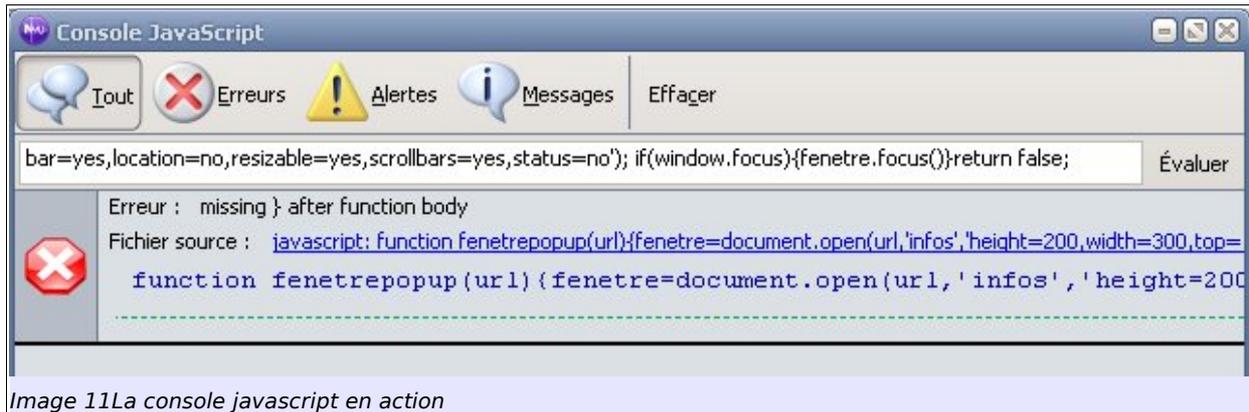
Image 10 Insérer un script à l'aide de htmlheader

## 1.2) Vérification du code à l'aide de la console Javascript

Cette console est le dernier outil qui reste intégré à Nvu du menu *Outils/Développement Web*. Elle est certainement moins pratique à utiliser que le débogueur Venkman mais elle est largement suffisante pour de petits scripts de débutants. L'insertion du code dans la console n'est pas très aisé, faites un copier/coller (utilisez le raccourci clavier ctrl+v) en passant par le code source de la page en cliquant sur l'onglet *<HTML> source*. Pour vous éviter un trop grand travail, placer votre code sur une seule ligne. Ce problème n'existe pas sous linux !

Une fois le code inséré, cliquez sur *Évaluer*, le script sera analysé et la console vous indiquera s'il y a une erreur.

La console nous indique qu'il y a une erreur. Il manque une accolade qui devrait fermer une fonction (*Erreur : missing } after function body*).



Pour notre exemple, c'est l'accolade de fin qui manque. Une fonction est toujours suivie d'une paires d'accolades { }. Dans notre exemple erroné, il manque l'accolade de fermeture pour la fonction.

## 2) Créer des Formulaires et vérification avec javascript

### Introduction

Nous avons vu comment fabriquer avec Nvu un site web classique et plutôt statique mais respectueux des standards du Web. Le fin du fin est de pouvoir ajouter une partie dynamique qui rendra votre site plus attractif. La technique du formulaire est un moyen particulier parmi d'autres. Elle permet de recueillir et de traiter les données fournies par un utilisateur.

Pour cela, il nous faut deux choses principales : un langage de formulation de données (le formulaire) défini par des balises *HTML* et un langage qui traitera les données entrées par l'utilisateur, un langage de *script*, mais pas nécessairement.

Cependant un problème subsiste. Les formulaires sont faits pour échanger des données avec le visiteur de votre site. Pour donner un aperçu de l'utilisation des formulaires, nous allons contourner le problème en créant deux types de formulaires. Le premier aura pour but de simplifier la navigation dans votre site (Une autre possibilité de navigation différente dans son approche du menu de navigation que nous avons vu). Le second vous présentera un cas typique de récupération de données.

Que les choses soient claires : ce n'est pas avec ce manuel que vous ap-

prenez à utiliser *Javascript*. Pour cela, je vous renvoie à l'excellent ouvrage *SELFHTML* qui traite d'une manière quasi exhaustive de « *HTML, JavaScript et des scripts cgi* » pour un traitement complet de vos formulaires. Nous nous attachons, uniquement, à découvrir l'option formulaire de Nvu.

L'idée est de créer des formulaires que *Javascript* traitera et dont il donnera une sortie en fonction de ce que nous lui demanderons.

## 2.1) Qu'est-ce qu'un formulaire ?

Un formulaire est un moyen de récupérer des données entrées par un utilisateur. Ces données peuvent être réduites à trois catégories principales :

- L'obtention d'informations sur l'utilisateur (nom, prénom, adresse électronique et/ou physique) ou son utilisation en fonction de sa navigation.
- Une série de questions optionnelles à cocher par l'utilisateur (que l'on rencontre sous la forme de boutons radio, de cases à cocher et un choix parmi une liste).
- La possibilité pour l'utilisateur de laisser un commentaire.

Pour les besoins de l'étude, nous ne verrons que deux exemples, un simple (une liste de choix qui pourra servir de liste de navigation sur votre site) et un formulaire un peu plus élaboré avec la récupération et l'envoi de données.

## 2.2) La méthode choisie

Maîtrisant mal les langages de script, je suis parti à la recherche de quelque chose qui correspondrait à des critères spécifiques : un exemple simple, fonctionnel de formulaire utilisant, s'il y a lieu, un code tout aussi simple.

De plus, je désirais que ce code fonctionne au moins sur deux grandes catégories de navigateurs : Internet Explorer et ceux de la famille Gecko (Mozilla, Firefox).

J'ai trouvé mon bonheur sur le site [quirksmode](http://quirksmode). L'auteur, Peter-Paul Koch<sup>1</sup>,

---

1) Le site de traduction des articles de fond sur HTML et Css [www.pompage.net](http://www.pompage.net) vient de publier ce mois-ci (août 2004) une traduction d'un des nombreux articles de l'auteur. Cependant, je ne conseille pas la lecture de cette article pour un débutant.

laisse libre l'utilisation des exemples bien qu'ils ne soient sous aucune licence particulière.

### 2.2.1) Méthode suivie

Pour pouvoir réaliser nos formulaires, nous allons utiliser trois langages possédant leur spécialité propre.

- Pour construire le squelette du formulaire, nous utiliserons le langage HTML qui est amplement suffisant.
- Pour construire les tests de vérification des formulaires, nous utiliserons le langage *JavaScript* qui est le plus adapté<sup>2</sup>.
- Enfin pour l'habillage, nous utiliserons le langage des feuilles de styles particulièrement adapté au design Web.

### 2.3) Créer un liste de choix pour simplifier la navigation dans un site Internet.

Si vous avez suivi le tutoriel dans ses grandes lignes, vous devriez disposer de trois pages ayant pour noms respectifs *mapage1.htm*, *mapage2.htm* et *mapage3.htm* (objet du précédent tutoriel consacré à l'interface standard). Je vous invite à en créer une quatrième qui nous servira de page d'accueil pour notre pseudo site que vous nommerez *index.htm*.

Dans cette page, nous allons utiliser les propriétés des formulaires afin de créer une liste de navigation à l'intérieur de notre site. Nous allons donc créer une courte liste qui contiendra deux groupes avec plusieurs sélections possibles, le premier groupe se nommant « navigation » et le second, « liens externes » par exemple, et un bouton de navigation qui, lorsqu'il sera cliqué par l'utilisateur, l'amènera à la page sélectionnée.

---

2) L'ajout de javascript est optionnel pour le déroulement de l'exercice. Il est ici survolé car le couple HTML + CSS n'est pas assez complet pour rendre totalement opérationnel un formulaire.

### 2.3.1)Définir les mentions générales du formulaire

#### Attention !

La définition de formulaires en langage XHTML ne fonctionne pas ! Veuillez utiliser le langage HTML 4.01

Cliquez sur *Formulaire/Définir un formulaire* le dialogue ci-après apparaît :



#### ➤ Paramètres

##### → Nom du formulaire

C'est dans ce champ que vous précisez le nom du formulaire. Pourquoi donner un nom à un formulaire ? Le nom est essentiel, si vous souhaitez par la suite utiliser Javascript par exemple (nous verrons cela plus en détail au moment de l'insertion de Javascript).

◆Pour l'exercice, vous insérez : navigation

##### → URL de l'action

Dans ce champ, vous insérez le type d'action que vous souhaitez avec l'adresse correspondante, afin que votre formulaire soit adressé correctement.

◆Pour notre exemple, nous insérons un nom invalide, ici,

« rapide ». puisque nous ne nous servons pas de du formulaire pour envoyer des données.

Dans la réalité, nous mettrions, une adresse électronique courante du genre : [tito@toti.org](mailto:tito@toti.org) (mais attention aux programmes-robots, collecteurs d'adresses. Ils en sont friands.) Un bon moyen est soit d'encoder votre adresse ; soit d'utiliser un script plus performant comme « cgi » à titre d'exemple.

### → **Méthode**

Il n'est pas utile préciser la méthode.

## 2.3.2)Création du bouton de navigation

Ma logique voudrait commencer d'abord par la liste de choix, puis continuer avec la création du bouton de navigation. Or cette logique ne semble pas convenir à Nvu qui écrase systématiquement la liste de choix une fois le bouton créé. L'inverse n'est pas vrai cependant. Donc, commençons par créer le bouton de navigation.

Dans le menu *Formulaire*, activez le dialogue *Champ de formulaire...*



Image 13 Définir le bouton de navigation

Dans *Type de champ*, sélectionnez *Bouton de validation*. Dans *Paramètres du champ*, inscrivez à *Nom du champ*, *naviguer* et la même chose à *Valeur du champ*.

Ensuite sélectionner la balise <form> et activez le dialogue des *Propriétés avancées*. À l'onglet Javascript, sélectionner **onsubmit** et inscrivez le nom de la fonction : **rapidement(); return false;**

Le terme *onsubmit* correspond à *bouton de validation*

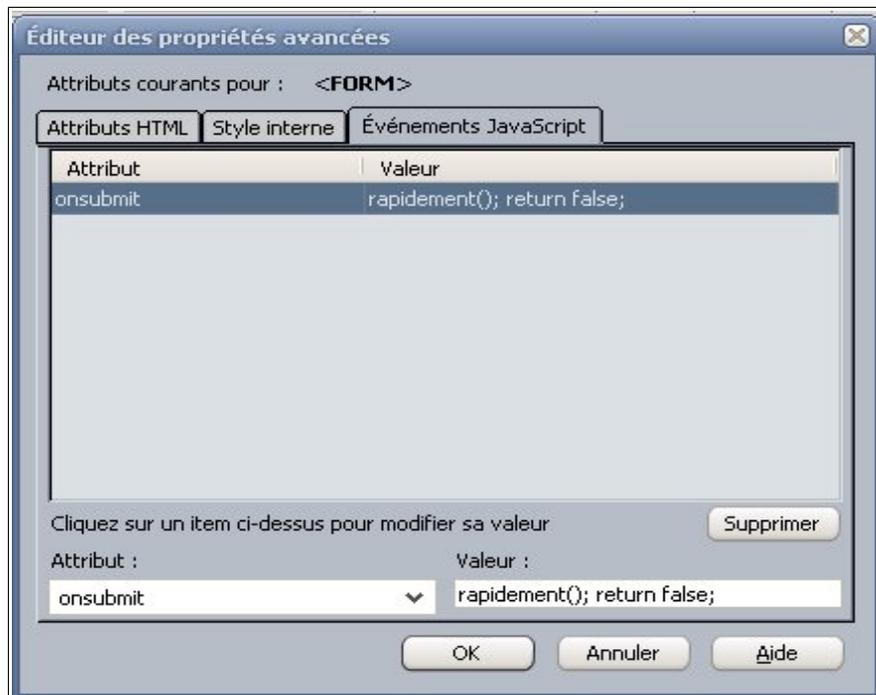


Image 14 Fenêtre propriétés avancées, onglet javascript

### 2.3.3) Création de la liste de choix

Dans le menu *Formulaire* vu plus haut, activez le dialogue *Liste de sélection...* La fenêtre suivante s'ouvre :

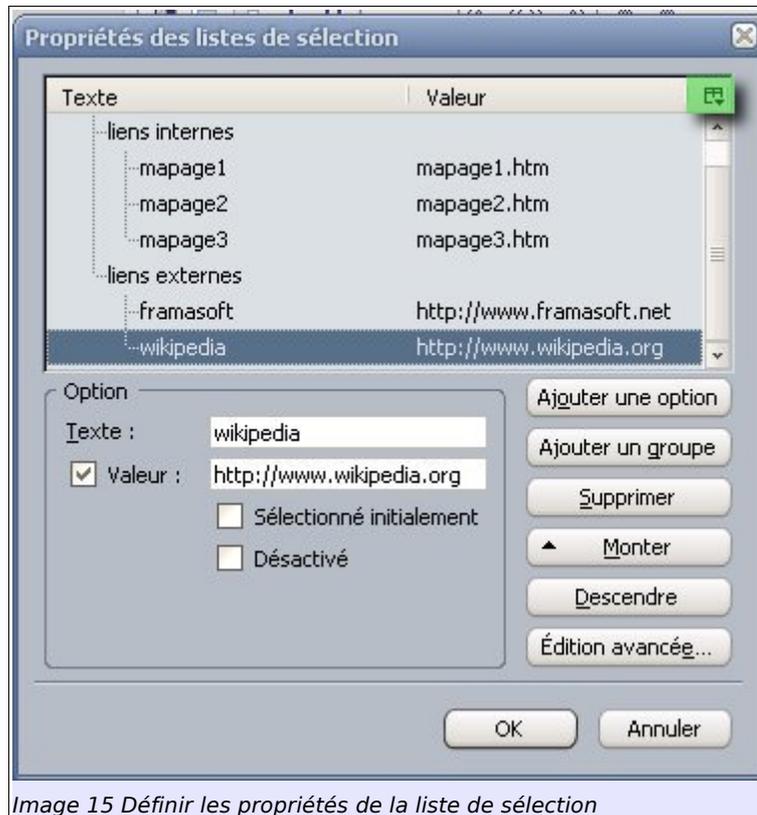


Image 15 Définir les propriétés de la liste de sélection

#### ➤ Fenêtre Texte, Valeur

#### Astuce !

Pour sélectionner les champs qui vous sont utiles (dans l'image ci-dessus, je n'ai gardé que les champs *Texte*, *valeur*) cliquez sur le bouton mis en évidence (vert) et choisissez les champs qui vous intéressent !

Inscription en arbre de la liste de sélection suivant cet ordre :

```
|-----Nom de la liste (s'active à l'ouverture du dialogue)
|-----Nom du groupe 1 (bouton Ajouter un groupe)
|         |-----Nom d'option de la liste (bouton Ajouter une option)
|         |-----Nom d'option de la liste
|         |-----Nom d'option de la liste
|-----Nom du groupe 2
|         |-----Nom d'option de la liste
|         |-----Nom d'option de la liste
|         |-----Nom d'option de la liste
```

Nom de la liste: liens

➤ Nom du groupe 1 : liens internes

Texte : mapage 1 ; valeur : mapage1.htm

Texte : mapage 2 ; valeur : mapage2.htm

Texte : mapage 3 ; valeur : mapage3.htm

➤ Nom du groupe 2 : liens externes

Texte : framasoft ; valeur http://www.framasoft.net

Texte : wikipedia ; valeur http://www.wikipedia.org

## 2.4) Code source complet du formulaire

```
<body>

<form id="activation" onsubmit="rapidement(); return false;"
action="rapide" name="navigation">

<input name="naviguer" value="naviguer" type="submit">

<select name="liens">
<optgroup label="liens interne">
<option value="mapage1.htm">mapage1</option>
<option value="mapage2.htm">mapage2</option>
<option value="mapage3.htm">mapage3</option>
</optgroup>

<optgroup label="liens externe">
<option value="www.framasoft.net">framsoft</option>
<option value="www.wikipedia.org">wikipedia</option>
```

```

</optgroup>

</select>
</form>

```

Nous retrouvons bien tout ce qui a été inséré en mode graphique avec les divers outils de Nvu. D'abord la définition générale du formulaire avec les balises `<form></form>`, puis la définition d'un bouton avec la balise `<input type="button">`, enfin notre liste de sélection `<select></select>`, les entrées de groupe avec `<optgroup></optgroup>` et, en dernier, les balises `<option></option>`. Ce qui donne en mode graphique sur une page HTML (profitons de l'extension Launchy pour avoir plusieurs vues dans différents navigateurs :



Il y a bien un sélecteur de navigation et un bouton nommé *naviguer*. Dans la liste de sélection, les groupes apparaissent en gras.

### Attention !

Pour les utilisateurs d'Internet Explorer 5. Les noms de groupe n'apparaissent pas.

### 3) Insertion du code Javascript

Avec L'extension HTMLheader cela ne doit plus poser de problème, il suffit de cliquer sur *Add/Script*.

```
<script type="text/javascript">
<!--
function rapidement()
{
location =
document.navigation.liens.options[document.navigation.liens.selectedIndex].value;
}
-->
</script>
```

Les parties en vert sont à ajouter à la main dans le champ *Content* du dialogue. Dans le champ *Type*, sélectionnez *Text/Javascript*.

### 3.1) Explication du code Javascript

Maintenant que nous avons terminé les préparatifs, passons à l'explication de ce que nous avons fait.

- Nous avons créé un formulaire avec un bouton de navigation ; il nous assure une navigation au clavier si la personne ne peut pas utiliser la souris par exemple.
- Nous avons créé une liste de sélection. Celle-ci contient plusieurs options de navigation pour l'utilisateur.
- Nous avons ajouter un code pour activer l'ensemble.

### 3.2) L'explication du code

```
➤<script type="text/javascript">

<!--

function rapidement()

{

location =
document.navigation.liens.options[document.navigation.liens.selecte
dIndex].value;

}

-->

</script>
```

Une fonction a été créée qui se nomme rapidement. Celle-ci est appelée dans le formulaire ainsi

```
<form onsubmit="rapidement(); return false;" action="rapide"
name="navigation">
```

Pour comprendre la suite, nous allons partir d'une représentation hiérarchisée en arbre de la page HTML :

```
|<html lang="fr">
+-----+<head>
|
|-----+<script type="text/javascript">
|   |<!--
|   |-----+function rapidement();
|   |   |
|   |   |- {location = document.navigation.liens.options
|   |       [document.navigation.liens.selectedIndex].value;}
|   |-->
|   +</script>
|
+</head>
|
+-----+<body style="direction: ltr;">
|--+<form action="rapide" name="navigation">
|----<input name="naviguer" value="naviguer" type="button">
|----+<select name="liens">
|
|----+<optgroup label="liens interne">
|----<option value="mapage1.htm">mapage1</option>
|----<option value="mapage2.htm">mapage2</option>
|----<option value="mapage3.htm">mapage3</option>
|----</optgroup>
|----+<optgroup label="liens externe">
|----<option value="www.framasoft.net">framsoft</option>
|----<option value="www.wikipedia.org">wikipedia</option>
|----</optgroup>
|----</select>
|--</form>
|</body>
</html>
```

En Javascript une page HTML se nomme *document*. Pour indiquer à Javascript où il doit chercher dans le *document*, nous lui indiquons des noms précis dans la hiérarchie du *document*, le premier niveau qu'il doit rencontrer est le nom qui définit le formulaire d'une manière générale `<form name="navigation">`. Ensuite nous allons lui dire de regarder encore plus bas dans la hiérarchie avec un nouveau nom qu'il doit rencontrer `<select name="liens">`.

Qu'est-ce qu'il cherche ? Cela est indiqué au tout début de la fonction, il

cherche une adresse HTML, *location* =

Donc *location* = *document.navigation.liens*, ce qui correspond bien à

```
<html>                                -->Document
+-----+<head>                        |
+-----+<body>                        |
      |----+<form name="navigation">    -->navigation
          |-----+<select name="liens"> -->liens
```

Une fois arrivé aux liens, il est indiqué que le script doit regarder dans les options, *location* = *document.navigation.liens.options*. Ensuite, il va être dit au script de retenir une option précise, celle qui sera sélectionnée par l'utilisateur, *selectedIndex*. Vous avez certainement remarqué la redondance, la première indiquant où regarder dans le document et la profondeur du niveau de hiérarchie du document, la seconde re-précisant tout cela avec la profondeur choisie : l'index d'option sélectionné.

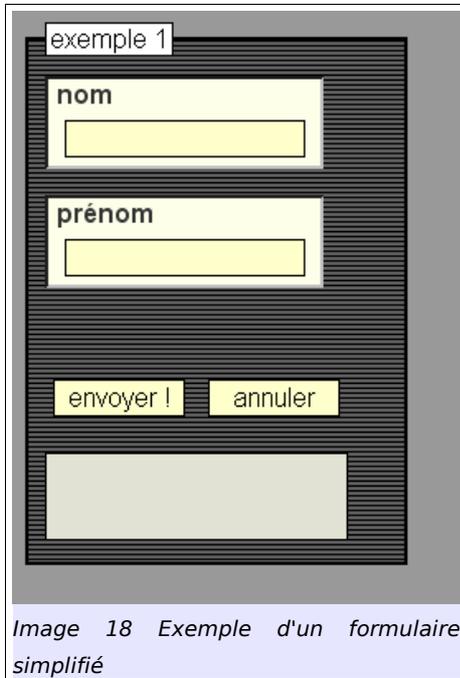
```
location =
document.navigation.liens.options                                             .value
                                     [document.navigation.liens.selectedIndex]
```

Enfin, une fois que le script a repéré quelle option a été sélectionnée par l'utilisateur, il doit transmettre la valeur (*value*) associée à la sélection pour pouvoir se terminer. La fin du script est signalée par un point-virgule.

Ce qui correspond bien à

```
<html>
+-----+<head>
+-----+<body>                                document.navigation.liens.options
      |----+<form name="navigation">
          |-----+<select name="liens">
              |-----+-----+-----+-----+-----+-----+-----+-----+
              |               [document.navigation.liens.selectedIndex].value
              |
              |-----+<option value="mapage1.htm">mapage1</option>      value
              |-----+<option value="mapage2.htm">mapage2</option>      value
              |-----+-----+-----+-----+-----+-----+-----+-----+
              |
```

## F) Réaliser un formulaire interactif.



tre un formulaire simple dans lequel nous avons deux (nom et prénom).

rez sur le bouton « *envoyer !* », le résultat de la saisie du formulaire apparaît sur l'écran du bas.

Si l'utilisateur ne remplit pas tous les champs, une fenêtre lui demandera de compléter le formulaire. À tout moment, l'utilisateur pourra annuler sa saisie.

Les boutons « *envoyer !* » et « *annuler* », lorsqu'ils seront survolés, auront un effet visuel (un curseur qui se transformera en pointeur (la fameuse main) et une indication graphique supplémentaire et pratique)!

Nous avons besoin des champs suivants :

- Un champ général qui définit le formulaire.
  - Deux champs textes pour recueillir le *nom* et le *prénom* de la personne.
  - Deux boutons, un *envoyer* et un *annuler*. Ils nous serviront respectivement à envoyer le formulaire et à annuler la saisie de l'utilisateur si ce dernier s'est trompé.
  - Une aire de texte, qui, ici, nous servira d'écran de sortie. Ainsi, nous pourrions voir ce qui est envoyé (c'est le détournement que nous utilisons pour utiliser un formulaire hors-ligne).
  - Un code Javascript un peu plus complet qui vérifiera si les champs *nom* et *prénom* ont bien été remplis. S'ils ne le sont pas, alors le curseur est envoyé dans le champ à remplir. S'ils le sont, alors le programme recueille les données inscrites par l'utilisateur et les envoie... vers notre écran de sortie.

## 1) Définition générale du formulaire



Image 19 Définition du second formulaire

➤ Le nom du formulaire est *exemple*, nous créons une pseudo action que nous nommons *essai*.

➤ Nous retenons *text/plain* dans les propriétés. Ceci nous permet de travailler avec du texte.

Regardons la fenêtre d'*Édition avancée* et intéressons-nous à l'onglet *Attributs HTML*

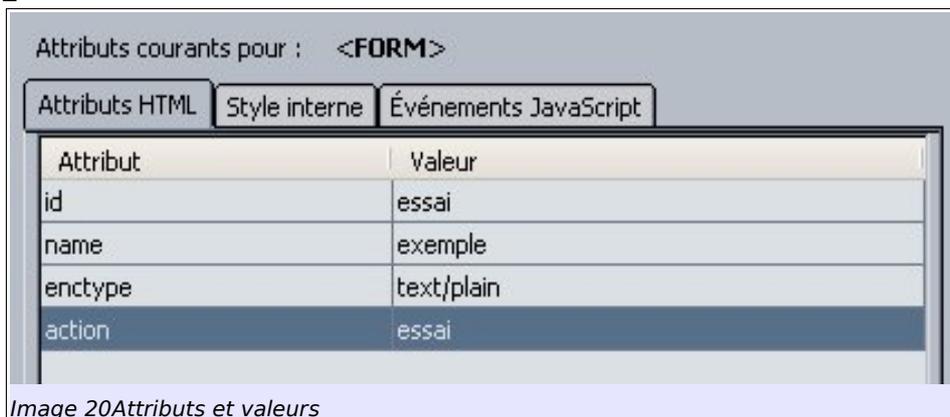


Image 20 Attributs et valeurs

Ce qui donne en langage HTML :

```
<form action="essai" enctype="text/plain" name="exemple"
id="essai">

</form>
```

### 1.1) Définition de l'événement à l'aide de JavaScript

Nous n'avons pas tout à fait terminé la définition générale du formulaire, il nous reste à définir l'événement déclencheur qui activera le code Javascript.

Cliquez sur le bouton *Formulaire* afin de réactiver la fenêtre générale. Maintenant cliquez sur *Édition avancée...*, sélectionnez l'onglet Événements Javascript. Dans le champ *Attribut*, à l'aide du menu déroulant, choisissez **onsubmit** et dans le champ *valeur*, insérez **verifier\_envoi(); return false ;** l'explication va venir... un peu plus tard.

En fait, nous précisons directement dans la définition générale du formulaire quel événement JavaScript doit lui être associé. Ceci pour des raisons pratiques et ergonomiques que je n'expliquerai pas ici<sup>3</sup>.

Nous indiquons ceci : lorsque l'utilisateur appuiera sur le bouton de validation du formulaire (onsubmit) *envoyer*, cet événement déclenchera une fonction Javascript *verifier\_envoi()*;

Voici le code HTML complet de la définition générale de notre formulaire

```
<form action="essai" onsubmit="verifier_envoi(); return
false;" enctype="text/plain" name="exemple" id="essai">

</form>
```

---

3) Consultez avec profit l'article de Laurent Jouanneau *Bien valider ses formulaires avec JavaScript* qui vous explique le pourquoi du comment. Page disponible sur le site [www.openweb.eu.org](http://www.openweb.eu.org)

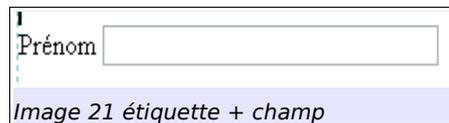
## 1.2) Définir des champs particuliers

### 1.2.1) Les champs texte *nom* et *prénom*

#### Astuce !

Avant de définir un champ, vous inscrirez manuellement le mot, *prénom*, et vous laissez le curseur à côté. Ensuite, vous créez votre *champ de formulaire texte*. Il en va de même pour le champ *nom*.

C'est la solution la plus simple pour créer des étiquettes avec Nvu. Toute opération contraire risque de vous surprendre beaucoup ! Le rôle des étiquettes sera expliqué plus bas.



Ouvrez le dialogue en passant par *Formulaire/Champ de formulaire...* Sélectionnez *Texte* dans *Type de champ* puis dans *Paramètres du champ* inscrivez *nom* ; enfin dans *Taille du champ*, inscrivez *30*. Réitérez la même opération pour créer le champ prénom.

### 1.2.2) Les boutons envoyer et annuler

Toujours le même principe. Menu *Formulaire/Champ de formulaire...* Dans *Type de champ*, choisissez *Bouton de validation*. Puis dans *Nom du champ* et *Valeur du champ* inscrivez : *envoyer !*. Réitérez la même opération sauf pour *Type de champ* vous choisissez *Bouton reset*. Le nom et la valeur : *annuler*.

Nous venons de définir deux champs textes, un nommé *nom* et un autre *prénom*. Nous leur avons donné une taille identique de *30*. Nous avons aussi défini deux boutons, un *bouton de validation* ayant pour *Valeur de champ* *envoyer !* et un *bouton Reset* ayant pour *Valeur de champ* : *annuler*.

Voici le code HTML complet correspondant :

```
<form action="essai" onsubmit="verifier_envoi(); return false"
enctype="text/plain" name="exemple">

nom<input size="30" name="nom">

pr&eacute;nom<input size="30" name="prénom">

<input value="envoyer !" type="submit">

<input value="annuler" type="reset">

</form>
```

Il ne nous reste plus qu'à définir un dernier champ : la zone de texte qui nous servira d'écran de sortie.

### 1.2.3) Le champ Zone de texte

Pour l'activer, sélectionnez dans *Formulaire...*, *Zone de texte...*



Image 23 Définir les propriétés de la zone de texte

Dans Paramètres inscrivez à *Nom du champ : sortie*, à *Lignes : 5* et à *Colonnes : 20*.

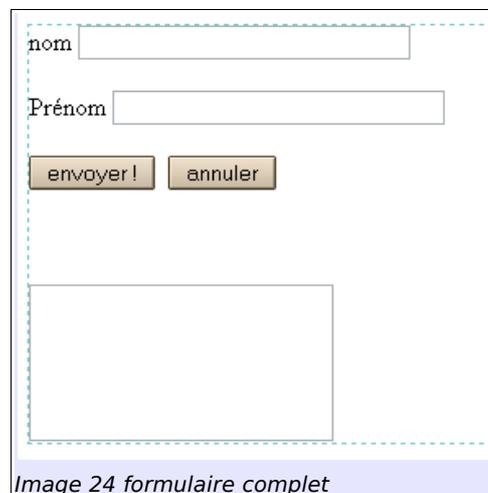


Image 24 formulaire complet

Voici le code HTML complet :

```
<form action="essai" onsubmit="verifier_envoi(); return false"
enctype="text/plain" name="exemple" id="essai">

nom<input size="30" name="nom">

pr&eacute;nom<input size="30" name="prénom">

<input value="envoyer !" type="submit">

<input value="annuler" type="reset">

<textarea cols="20" rows="5" name="sortie"></textarea></form>
```

### 1.3) Définition d'étiquettes relationnelles pour les champs *nom* et *prénom*

Afin de faciliter la logique interne d'un formulaire et donc de ses différentes relations avec ses éléments, il a été défini des balises spécifiques nommées *label* (*Étiquette en français*) qui, accompagnées de l'instruction *for*, désignent la relation exacte d'un terme avec un champ du formulaire.

La mention *for* apparaît dans la version 0.4x de Nvu ; dans les versions précédentes, elle était absente, il faudra donc l'insérer manuellement pour ces versions.

Sélectionnez à l'aide de la souris *nom*. Maintenez la touche shift (représentée par une flèche sur le clavier) et cliquez sur le champ à côté. Ces deux zones devraient être sélectionnées.

Ensuite, il suffit de se reporter au menu *Formulaire...*, de descendre jusqu'à *Créer une Étiquette*, de cliquer dessus ! Dans le dialogue insérer dans le champ *Pour contrôler : nom*. Réitérer la même opération pour *prénom*.

Voici le code HTML complet :

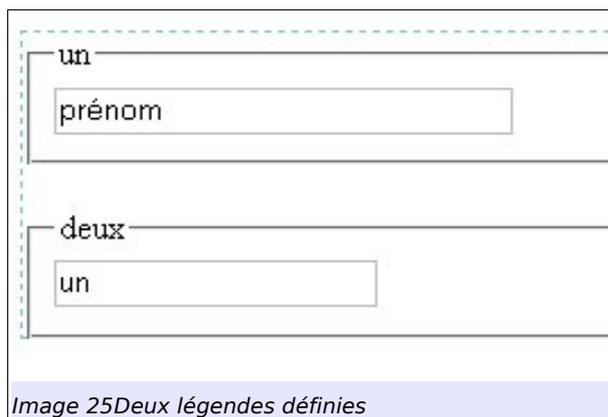
```
<form action="essai" onsubmit="verifier_envoi(); return false"
enctype="text/plain" name="exemple" id="essai">
```

```
<label for="nom"><input size="30" name="nom"></label>
<label for="prénom"><input size="30" name="prénom"></label>
<input value="envoyer !" type="submit">
<input value="annuler" type="reset">
<textarea cols="20" rows="5" name="sortie"></textarea>
</form>
```

### 1.3.1) Définir les propriétés de champs des formulaires.

Les propriétés de champs sont utiles lorsque vous avez plusieurs parties à distinguer dans un même formulaire. Pour éditer rapidement et simplement cette propriété, deux opérations simples : sélectionnez le ou les champs qui vous intéressent. Ensuite, dans le menu *Formulaire...*, descendez jusqu'à *Définir les propriétés du champ...*

Un dialogue apparaît. Inscrivez le nom que vous souhaitez, une liste de sélection vous permettant de choisir l'emplacement de votre légende, *À gauche*, *Au centre*, *À droite* ou laissez l'option *Par défaut*.



### 1.3.2) Styliser votre formulaire avec l'éditeur Cascades

Je ne vous donne pas d'instructions spécifiques. À vous de travailler un peu ! Cela devrait être un jeu d'enfant avec Nvu !

Voici quelques conseils tout de même. Définissez le style de votre formulaire suivant sa sémantique : *form* pour le style général; *label* pour les champs *nom* et *prénom*. Si vous souhaitez distinguer le champ *input* de l'étiquette *label*, alors insérez une règle pour *input*. Faites attention à ceci : votre formulaire est composé de quatre entrées *input*, il va falloir les distinguer par paires. Donc, il faudra créer deux classes particulières. N'oubliez pas d'insérer en conséquence des noms de classe pour chaque champ *input*. Cela n'est pas strictement nécessaire et cela n'empêchera pas la suite de l'exercice.

L'astuce pour obtenir la main à la place du curseur.

Comment transformer le curseur *flèche* en *pointeur main* lorsque l'utilisateur survolera les boutons *envoyer* et *annuler* ?

Voici l'instruction CSS à placer, manuellement, pour vos boutons

*input*

```
{cursor: pointer;}
```

Malheureusement cette règle n'est pas reconnue par Internet Explorer. Cependant, vous connaissez le script ie-7 de Dean Edwards. Il suffira de créer un lien vers ce script et Internet Explorer reconnaîtra cette propriété. Pratique !

## 2) Vérifier les saisies du formulaire à l'aide de Javascript

L'insertion de Javascript en mode graphique ne nous pose plus de problèmes avec htmlheader. Il suffit de cliquer sur le bouton *Add*, puis sur *Script* Une fenêtre apparaît et nous n'avons plus qu'à insérer le code tranquillement sans nous inquiéter de quoi que ce soit. Le module se chargera de placer le code là où il faut dans l'entête.

## 2.1) Courte présentation du code à insérer

```

<!--
function verifier_envoi()
{
  var texte = '';
  for (i=0;i<2;i++)
  {
    var verifier = document.exemple.elements[i];
    if (!verifier.value)
    {
      alert('Vous n\'avez pas rempli le champ ' + verifier.name + ' !');
      verifier.focus()
      return;
    }
    texte += verifier.name + ': ' + verifier.value + '\n';
  }
  document.exemple.sortie.value = texte;
}
-->

```

```

<form action="essai" onsubmit="verifier_envoi(); return false"
enctype="text/plain" name="exemple" id="essai">

<label for="nom"><input size="30" name="nom"></label>

<label for="prénom"><input size="30" name="prénom"></label>

<input value="envoyer !" type="submit">

<input value="annuler" type="reset">

<textarea cols="20" rows="5" name="sortie"></textarea>

</form>

```

Il vérifie si l'utilisateur a inséré quelque chose, soit dans le champ *nom*, soit dans le champ *prénom*. Pour cela, il a été déclaré une variable qui s'appelle *verifier* qui va regarder ce qui se passe dans les *éléments* du *document* qui se réfèrent au nom *exemple*.

Tout ce qui suit se déroulera lorsque l'utilisateur cliquera sur le bouton *envoyer* ; d'où, dans notre travail précédent, l'insertion du code : <form action="es-

```
sai" onsubmit="verifier_envoi(); return false" enctype="text/plain"
name="exemple" id="essai">.
```

Comme il y a plusieurs éléments, ici deux champs *input*, il a été défini une boucle au préalable à l'aide de l'instruction `for` qui vérifiera que les champs sont bien remplis sans aller au delà des deux `i<2`.

Une fois que l'on a indiqué où le programme devait regarder, on lui demande de vérifier s'il y a quelque chose ou non à l'aide d'une instruction conditionnelle. Cette dernière est prise à l'envers, ce qui permet de gagner en lisibilité : s'il n'y a rien (aucune valeur) dans `verifier`, alors il faut alerter l'utilisateur et le lui dire. Le point d'exclamation sert pour dire en gros *différent de*.

Une boîte de dialogue de type `alert` s'ouvre et inscrit la phrase suivante « Vous n'avez pas rempli le champ + `verifier.name` ».

`verifier.focus()` est une instruction supplémentaire qui cale le curseur au bon endroit ; là où le champ n'a pas été rempli.

Une fois les champs bien remplis, il va falloir sortir le résultat pour qu'il puisse être traité. Pour cela, une nouvelle variable a été définie dont le contenu n'a pas été défini explicitement, d'où « `var texte = ''` » ; il le sera par l'utilisateur.

« `texte += verifier.name + ': ' + verifier.value + '\n'` » La variable `texte` doit contenir (`+=`) les valeurs suivantes : d'abord les valeurs de chaque champ *input* (`verifier.name`) auxquelles seront ajoutées les valeurs de l'utilisateur (`verifier.value`). Enfin, pour bien les distinguer, on force un retour à la ligne avec « `\n` »

Une fois que ces différentes valeurs ont été récupérées, le script inscrit ces valeurs dans la zone de texte nommée sortie.

Ce qui correspond bien à : (pour le plaisir:-)

```
+<html>
+-----+<head>
+-----+<body>
  +-----+<fieldset>
    +-----+<legend>      var verifier = document.exemple2.elements[i];
      +-----+<form name="exemple2">
        |
        =====|=====
        =      +-----+<label>      =
        =      |      |      =
        =      |      |      =
        =      |      +-----+<input name="nom">      =
for (i=0;i<2;i++)      |      if (!verifier.value)+
        =      |      =      |
        =      +-----+<label>      =      |
        =      |      |      =      +alert()
        =      |      |      =
        =      |      +-----+<input name="prénom">      =
        =      |      if (!verifier.value)+
        =====|=====      |
[retourner à la boucle tant |      verifier.focus() return;      |
que la condition n'est pas |      |
remplie. Pas de valeur dans +-----+<input type="submit">      |
champs nom et/ou prénom.] |      +alert()
        |
        +-----+<input type="reset">
        |
        |
        +-----+<textarea name="sortie">
          var texte = ' ';
          texte += verifier.name + ': ' + verifier.value + '\n';
          document.exemple2.sortie.value = texte;
```

## G) Et si on passait au XHTML Strict ?

Si vous avez suivi ces deux tutoriels. D'abord bravo ! Ensuite, vous possédez les outils de bases pour passer au XHTML sans trop de problèmes.

La construction d'une page dans ce format suit l'ordre que nous avons découvert tout au long de ces tutoriels.

- Écriture du document au format brut, uniquement avec des balises qui structurent le texte.
- Puis l'ensemble des menus de navigation sont écrits avec des listes soit une liste non ordonnée soit une liste de définition comme nous l'avons vu.
- Pourquoi cela ? Tout simplement nous nous assurons que la page est lisible par n'importe quel type de navigateur qu'il soit graphique ou non.
- Puis ajout des styles. Nous savons maintenant que chaque balise HTML peut recevoir un style, qu'elle est considérée comme un bloc auquel nous pouvons ajouter, insérer différentes propriétés de mise en page.
- Étant donné que chaque balise HTML est entièrement paramétrable, il ne nous est plus utile d'utiliser des frames ou des cadres pour obtenir ce que nous voulons. Il suffit de créer des conteneurs génériques ou div que nous placerons et dimensionnerons comme bon nous semble.
- La feuille de style sera, de préférence externe, ainsi chaque page du site sera attachée à cette dernière et gardera une homogénéité pour l'ensemble du site.
- Nous savons comment pallier certains problèmes de mise en page pour les navigateurs de type Internet Explorer grâce au script ie-7 de Dean Edwards.
- Enfin nous avons survolé la manière dont Javascript peut nous aider à améliorer le confort du visiteur de notre site avec une fenêtre pop up et un script de vérification de formulaire. Javascript vous servira à beaucoup de choses croyez moi !

**Astuce !**

Si vous tentez l'expérience XHTML Strict vous vous apercevrez que certaines parties de l'interface graphique de Nvu sont désactivées. Ce qui est tout à fait normal. Seules sont activées les interfaces qui structurent le texte et la mise en page avec des styles.

Donc ne soyez pas surpris par ce changement de comportement !

**Bienvenue dans le monde de l'édition Web moderne et de qualité**

**DERNIER CONSEIL**

**Prenez votre temps ! Apprenez régulièrement sans vous forcer.**

**Vous ne comprenez pas aujourd'hui ? Attendez demain !**

## H) Conclusion

---

Nvu s'avère être un outil très complet capable de nous surprendre sur de nombreux points. Malgré quelques problèmes (l'utilisation de l'extension viewsourcewith reste utile pour des manipulations fines du document), il apporte une grande fraîcheur dans l'environnement des éditeurs HTML et montre que le meilleur est possible avec un éditeur en mode graphique.

Htmlheader s'avère être un outil très utile et complet que je vous invite à découvrir et à utiliser ; cette extension se marie très bien avec le logiciel et peut nous aider à aller plus vite, par exemple, dans la correction d'une mention de style. Au lieu de passer par l'éditeur CSS assez lourd dans son utilisation, Htmlheader nous donne un accès direct au code.

Au cours de l'exercice suivant, nous avons vu comment réaliser des formulaires. La création de formulaires est relativement simple pour une structure HTML. Ce qui donnera un plein sens à vos formulaires sera sa description stylistique à l'aide d'une feuille de style et le traitement des données à l'aide d'un langage simplifié comme Javascript.

### 1) Pour finir... et continuer !

Vous venez de parcourir la seconde version du tutoriel Nvu-Modulaire. L'idée est de pouvoir approfondir la connaissance du programme tout en essayant de trouver des solutions relativement simples.

La seconde idée directrice, qui permettrait de pousser le concept du tutoriel plus en avant est de créer une collection (dans la mesure du possible cela va de soi) de tutoriels consacrés à telle ou telle extension.

J'espère que cette nouvelle version vous satisfera.