

Travaux Pratiques d'introduction aux systèmes et réseaux

Michel Meynard

22 octobre 2008

1 Introduction

En cas de problème, veuillez lire les lignes suivantes avant de demander de l'aide. Ce TP est destiné à être exécuté dans un environnement comportant :

- un navigateur (client HTTP) ; utilisez plutôt Firefox que Konqueror !
- un éditeur de texte ; utilisez Emacs de préférence !

2 Utilisation du navigateur et de Emacs

Exercice 1 (Firefox) Ouvrez un navigateur Firefox :

1. Comment ouvrir plusieurs onglets dans la même fenêtre ?
2. Atteindre le site de la faculté des sciences à l'adresse `http://www.ufr.univ-montp2.fr/`. Faites rechercher par le navigateur, le mot "département" sur cette page et rendez-vous sur la page des départements ;
3. revenez sur la page d'accueil et examiner le source de la page d'accueil.
4. Faites rechercher par le navigateur par "/", le mot "département" sur ce source : que se passe-t-il ?
5. Enregistrez la page d'accueil dans l'un de vos répertoires personnels puis visualisez le fichier html dans votre navigateur ; existe-t-il des différences ? Combien de fichiers ont-été copiés ? Pourquoi ?
6. Ouvrez la console d'erreurs Javascript.

Solution 1 1. menu Fichier/nouvel onglet ou CTL t ;

2. menu Edition/rechercher ou mieux par / pour la recherche rapide suivis de F3 pour les suivants ;
3. menu Affichage/code source ou Ctrl U ;
4. pas de département mais departement sans accent dans un lien et `départements` dans le texte ;
5. menu Fichier/enregistrer sous ; il existe des différences minimes sur certaines images non copiées mais de nombreux fichiers gif css ou js ont été copiés.
6. menu Outils/Ouvrir la console d'erreurs ;

Exercice 2 (Emacs) Ouvrez un éditeur Emacs puis ouvrez un fichier toto.html ;

1. Comment insérer un nouvel élément tel que html ?
2. Insérer un lien vers le fichier toto.html : remarquez que les attributs possibles sont accessibles en utilisant l'auto-complétion (touche tab).
3. Comment visualiser le fichier HTML sur le navigateur ?
4. A quoi sert la bascule 'toggle auto viewing' ?

Remarquons que le mode HTML d'emacs ne fournit pas un document XHTML valide !

Solution 2 1. C-c C-t ou SGML insert tag puis html

2. C-c C-t a puis h Tab ...
3. C-c C-v ou menu Html/view buffer contents
4. visualisation automatique à chaque sauvegarde

3 Formulaires HTML

Exercice 3 (Inscription) On souhaite écrire un formulaire d'inscription à un site où sont demandés :

- le nom ;
- le prénom ;
- l'adresse email (2 fois)

1. Rédigez un document XHTML conforme inscription.html sans vous préoccuper de la présentation ; la méthode de soumission sera post et l'action sera le formulaire lui-même ;
2. Testez votre formulaire en le soumettant avec la méthode post puis avec la méthode get : quelle différence ?
3. Validez le document en le chargeant (upload) sur le site du W3C ; une fois toutes les erreurs corrigées, ajoutez le logo du W3C à la fin de votre fichier ;

- En examinant le source de la page du validateur du W3C, quelle est la méthode utilisée par le formulaire et quel est le type d'encodage ? Que cela signifie-t-il ?

Solution 3 1. inscription.html

```

<!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" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />

<title>Inscription</title>
</head>
<body>
<h1>Inscription</h1>

<form method="get" action="inscription.html">
Nom :<input type="text" name="nom" size="20" /><br clear="right" />
Prénom :<input type="text" name="prenom" size="20" /><br clear="left" />
Email :<input type="text" name="email1" size="30" /><br clear="left" />
Email :<input type="text" name="email2" size="30" /><br clear="left" />

<input type="submit" value="OK" />
</form>

<p>
<a href="http://validator.w3.org/check?uri=referer">

</a>
</p>

</body>
</html>

```

- avec get, les noms et valeurs des champs de formulaires sont passés visiblement dans la barre d'adrs au format : `inscription.html?nom=toto&prenom=&...`
- se rendre à l'url `http://validator.w3.org/check` et télécharger le fichier ...
- la méthode est post et le type est : `enctype="multipart/form-data"`. Cela indique le postage du fichier indiqué dans le champ de type `input type="file"`

4 Javascript

Exercice 4 (Factorielle) On s'intéresse à la fonction factorielle définie par $0! = 1$ et $(n + 1)! = (n + 1) * n!$.

- Ecrivez la fonction factorielle en JavaScript dans un fichier `fact1.js` ; testez cette fonction en écrivant les valeurs $5!$ et de $15!$ dans un fichier `fact1.html` ;
- écrivez un fichier XHTML `fact2.html` permettant de saisir un nombre entier puis d'afficher sa factorielle sans utiliser de programmation côté serveur ;
- que se passe-t-il exactement si le paramètre passé est incorrect ?
- Comment y remédier ? Ecrivez un programme qui vérifie la chaîne saisie par l'utilisateur tant que celle-ci ne contient pas un nombre puis qui affiche sa factorielle.
- Réécrivez la fonction factorielle afin que celle-ci lève une exception dans le cas où son paramètre n'est pas un entier naturel.
- Réécrivez le fichier XHTML afin qu'il gère l'exception éventuelle !

Solution 4 1. fact1.js

```

function fact(n){
  if (n==0)
    return 1;
  else
    return n*fact(n-1);
}
fact1.html

```

```

<!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" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
<script language='javascript' src='fact1.js'>
</script>

<title>Factorielle</title>
</head>
<body>
<h1>Factorielle</h1>
<script language='javascript'>
document.write("5! = "+fact(5)+'<br />');
document.write("15! = "+fact(15));
</script>
</body>
</html>

```

2. fact2.html

```

<!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" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
<script language='javascript' src='fact1.js'>
</script>

<title>Factorielle</title>
</head>
<body>
<h1>Factorielle</h1>
<script language='javascript'>
var n=prompt('Veuillez saisir un nombre entier positif S.V.P. !');
document.write(n+"! = "+fact(n)+'<br />');
</script>
</body>
</html>

```

3. En saisissant une chaîne quelconque comme 'ab', la fonction boucle à l'infini car 'ab'-1 vaut NaN (Not a Number) et NaN-1 vaut également NaN!

4. En utilisant parseInt(n) qui transforme une chaîne en entier ou en Number.NaN; fact3.html

```

<!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" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
<script language='javascript' src='fact1.js'>
</script>

<title>Factorielle</title>
</head>
<body>
<h1>Factorielle</h1>
<script language='javascript'>
var n;
do{
  n=parseInt(prompt('Veuillez saisir un nombre entier positif S.V.P. !'));
} while (isNaN(n) || n<0)
document.write(n+"! = "+fact(n)+'<br />');
</script>

```

```
</body>
</html>
```

5. fact2.js

```
function fact(n){
  if (isNaN(n) || n.toFixed(0)!=n || n<0)
    throw "fact requiert un entier naturel !";
  if (n==0)
    return 1;
  else
    return n*fact(n-1);
}
```

6. fact4.html

```
<!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" xml:lang="fr" lang="fr">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
<script language='javascript' src='fact2.js'>
</script>

<title>Factorielle</title>
</head>
<body>
<h1>Factorielle</h1>
<script language='javascript'>
var n;
var fini;
do{
  n=parseFloat(prompt('Veuillez saisir un nombre entier positif S.V.P. !'));
  try{
    document.write(n+"! = "+fact(n)+'<br />');
    fini=true;
  } catch (e) {
    fini=false;
    document.write("EXCEPTION !!!"+'<br />');
  }
} while (!fini)
</script>
</body>
</html>
```

Exercice 5 (Bataille navale) On souhaite construire une application de bataille navale simplifiée en JavaScript. L'application contient du code XHTML affichant un tableau de 5 lignes (1 à 5) par 5 colonnes (A à E). Dans chaque cellule un bouton contenant un point d'interrogation est affiché comme dans la figure 1.

Sur l'appui d'un bouton, le texte affiché sur le bouton change en :

- "BOUM" si la cellule contenait un des 4 vaisseaux (de taille 1x1),
- ~~~ si la cellule ne contenait rien.

De plus, on ouvrira une fenêtre pour indiquer les clics sur des positions déjà jouées, ou le gain du jeu. La figure 2 illustre une possible fin de jeu.

1. Indiquer la ou les structure de données JavaScript représentant le tableau de jeu, les bateaux, les coups joués, ...;
2. écrire le code statique html du tableau de jeu ; on identifiera les boutons avec 2 chiffres décimaux correspondant au numéro de ligne et de colonne : 00, 01, ..., 04, 10, ..., 44.
3. écrire la fonction `initJeu()` qui initialise les positions aléatoires des 4 bateaux ;
4. écrire la fonction `joue(id)` qui est appelée à chaque clic sur un bouton de case ;

Solution 5 1. le code javascript mémorisera :

- le nombre de bateaux : `var nbBateaux=4;`
- un tableau contenant les 4 indices compris entre 0 et 24 des bateaux : `var bateaux;`
- le nombre de bateaux coulés `var nbBoum=0;` pour afficher la fin du jeu ;