Table des matières

Mise en situation

Une société de développement désire mettre en place un site pour le suivi des incidents sur ses projets logiciels. On va donc réaliser étape par étape ce site Web.

La planification est la suivante :

- **étape n°1 : réalisation des pages HTML**

  Cette étape a pour but de découvrir et de prendre en main le langage ***HTML*** qui est interprété côté client par le navigateur. L'objectif principal de cette étape est la réalisation des pages **statiques** du site : menu, affichage des résultats, formulaires pour rapporter une anomalie, effectuer une recherche, etc ...

- **étape n°2 : mise en place de l'interactivité côté client**

  Cette étape a pour but de découvrir et de prendre en main le langage ***JavaScript*** qui s'exécute côté client (donc par le navigateur). L'objectif principal de cette étape est d'assurer l'intégrité des données envoyées vers le serveur par les formulaires. En effet, il serait inutile de surcharger le serveur avec l'envoi de données manquantes, incomplètes ou incorrectes. Cette vérification peut donc se faire côté client par le langage ***Javascript***.

- **étape n°3 : réalisation des pages dynamiques**

  Cette étape a pour but de découvrir et de prendre en main le langage ***PHP*** qui s'exécute côté serveur. Les objectifs principaux de cette étape sont :
  - de traiter les données reçues par les différents formulaires et de les insérer dans la base de données
  - d'interroger la base de données et de fournir un résultat adapté aux requêtes clientes (rapporter une anomalie, lister toutes les rapports d'anomalies, effectuer une recherche, etc ...).

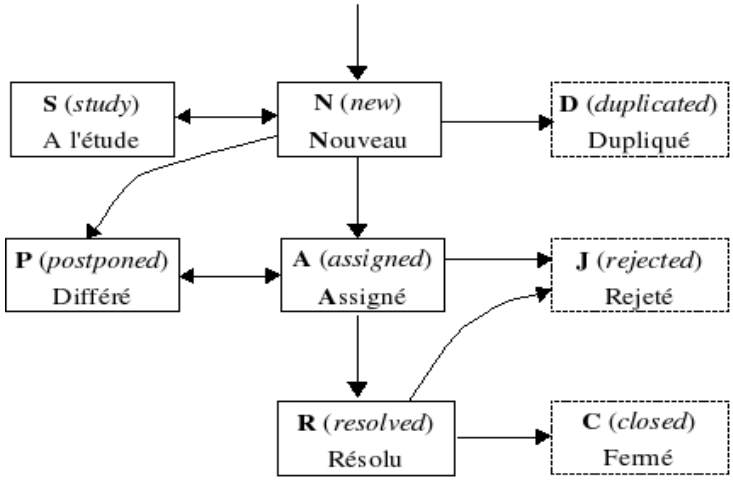
Gestion des incidents

Présentation

Tous les incidents (*bugs*) doivent être référencés de façon unique dans une base de données. Le rapport d'incident est un enregistrement contenant différents champs: un identifiant unique (numéro), un titre, une description, un état, une sévérité, la reproductibilité, une catégorie, un niveau de priorité, etc ...

Etat d'un incident

Un rapport d'incident a différents états en fonction de l'avancement de son traitement; ceux-ci doivent suivre le cycle de vie du rapport défini sous forme de diagramme :



Remarques

Gestion

A l'état N est associé le nom (login) du créateur (le rapporteur) ;
Aux états S et A est associé le nom d'un développeur ou d'un correcteur ;
Une commission de traitement des rapports d'incident s'occupe des incidents dans les états N, D et P, et effectue l'assignation, l'étude ou le rejet ;
Une autre commission dite d'intégration s'occupe des incidents dans les états R et J et décide la clôture ;

Priorité

Lors de l'assignation d'un incident, il faudra fixer une priorité (de basse à urgente par exemple).

Sévérité - Reproductibilité

Chaque rapport d'incident doit également se voir affecter une sévérité, de mineure à bloquante par exemple. Le rapporteur se doit de préciser la reproductibilité de l'incident.

Catégorie

Il est intéressant d'ajouter un champ pour classer l'incident dans une catégorie selon sa nature.
Enfin, une recommandation élémentaire, mais souvent oubliée, est qu'un rapport d'incident ne doit traiter que d'un seul problème. Il faut prendre la peine d'écrire deux rapports distincts pour deux problèmes connexes mais différents; l'un pourra être résolu et vérifié indépendamment de l'autre.

Recherche

Enfin, il apparaît fondamental de pouvoir accéder aux bases de tous les composants logiciels utilisés (génériques ou spécifiques) et pouvoir y faire des recherches par mots-clés.

Le service Web

Serveur Web (RFC HTTP 1945)

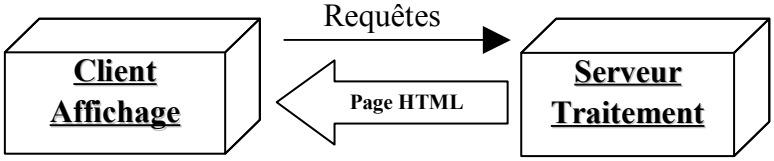
Le serveur Web est un programme applicatif acceptant des connexions dans le but de traiter des requêtes HTTP en délivrant une réponse HTTP. Il peut être considéré comme serveur de fichiers ou de documents.
Les deux principaux serveurs Web du marché sont : *Microsoft IIS* et ***Apache*** (serveur libre multi-plateforme représentant à lui seul environ 60% du marché)

Client Web (RFC HTTP 1945)

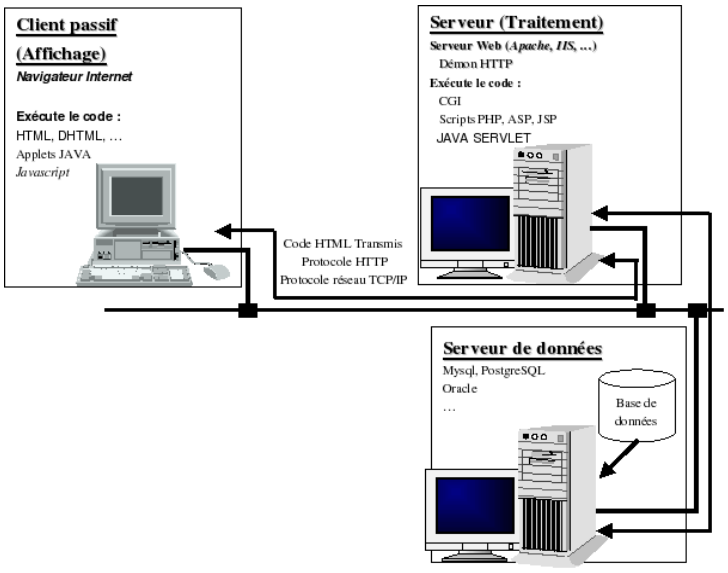
Un client est un programme applicatif qui permet d'émettre des requêtes HTTP et interpréter les réponses HTTP afin d'extraire les données reçues (HTML, Javascript, etc …). Le plus souvent, l'applicatif client est un **navigateur** (en anglais *browser*, en québécois *fureteur*).
Les plus utilisés sont : Internet Explorer, Firefox, Safari, Chrome, Opéra, Lynx, Konqueror, …

Architecture Client/Serveur

C'est la topologie la plus courante actuellement. Le serveur est en attente des demandes en provenance du client : c'est le fournisseur du service.



L'architecture client/serveur Web plus détaillée peut être schématisée de la manière suivante :



Remarques:

- Le serveur de données peut, éventuellement, être installé dans la même machine que le serveur Web, si ces performances le permettent.

- Les documents à base de langages s'exécutant côté client ne nécessitent pas forcément la présence d'un serveur Web. De manière générale, le serveur permet l'accès distant et la centralisation des documents web à partager.

Séquence 1 : page d'accueil

Objectifs

Cette première séquence a pour objectif de réaliser la première page HTML du site. Elle met en oeuvre :

- la structure des pages en utilisant quelques balises de base
- les liens vers les autres pages
- l'utilisation des listes
- l'insertion d'image pour le logo de l'entreprise

Moyens disponibles

Un éditeur de texte (***vim, kwrite, kate, notepad++, ...***) pour la réalisation de la page HTML
Un navigateur (***Firefox, IE, Opera, ...***) pour visualiser la page HTML réalisée

Travail préliminaire

Créer un répertoire **incidents** pour le stockage des fichiers du site. Dans le répetoire **incidents**, créer un sous-répertoire **images** dans lequel on stockera toutes les images nécessaires pour le site.
L'ensemble des pages HTML porteront l'extension **.html**.

Contraintes

A . Pour la réalisation des pages HTML de ce site, on utilisera la **DTD HTML 4.01 Strict** en spécifiant la déclaration de type de document suivante :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
```

B . Pour la réalisation des pages HTML de ce site, on utilisera l'encodage de caractères **ISO-8859-1** (ou ISO-8859-15 ou éventuellement UTF-8) :

```
<meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
```

Remarques : Il est important de comprendre qu'il faut obtenir une cohérence entre l'encodage utilisé pour créer le document (l'éditeur de texte) et celui pour l'interpréter (le navigateur) :

- Les outils d'édition (par exemple, un éditeur de texte) peuvent coder des documents HTML avec l'encodage de caractères de leur choix. Ces outils peuvent utiliser tout encodage commode lequel couvre la plupart des caractères contenus dans le document, pourvu que l'encodage soit correctement étiqueté. Les caractères occasionnels, qui ne sont pas contenus dans cet encodage, peuvent tout de même être représentés par des références de caractères (numérique ou entité).
- Le paramètre « charset » identifie un encodage de caractères, qui représente une méthode pour convertir une séquence d'octets en une séquence de caractères. Cette conversion s'intègre naturellement au système de l'activité du Web : les serveurs envoient des documents HTML aux agents utilisateurs sous la forme d'un flux d'octets, les agents utilisateurs les interprètent comme séquence de caractères.

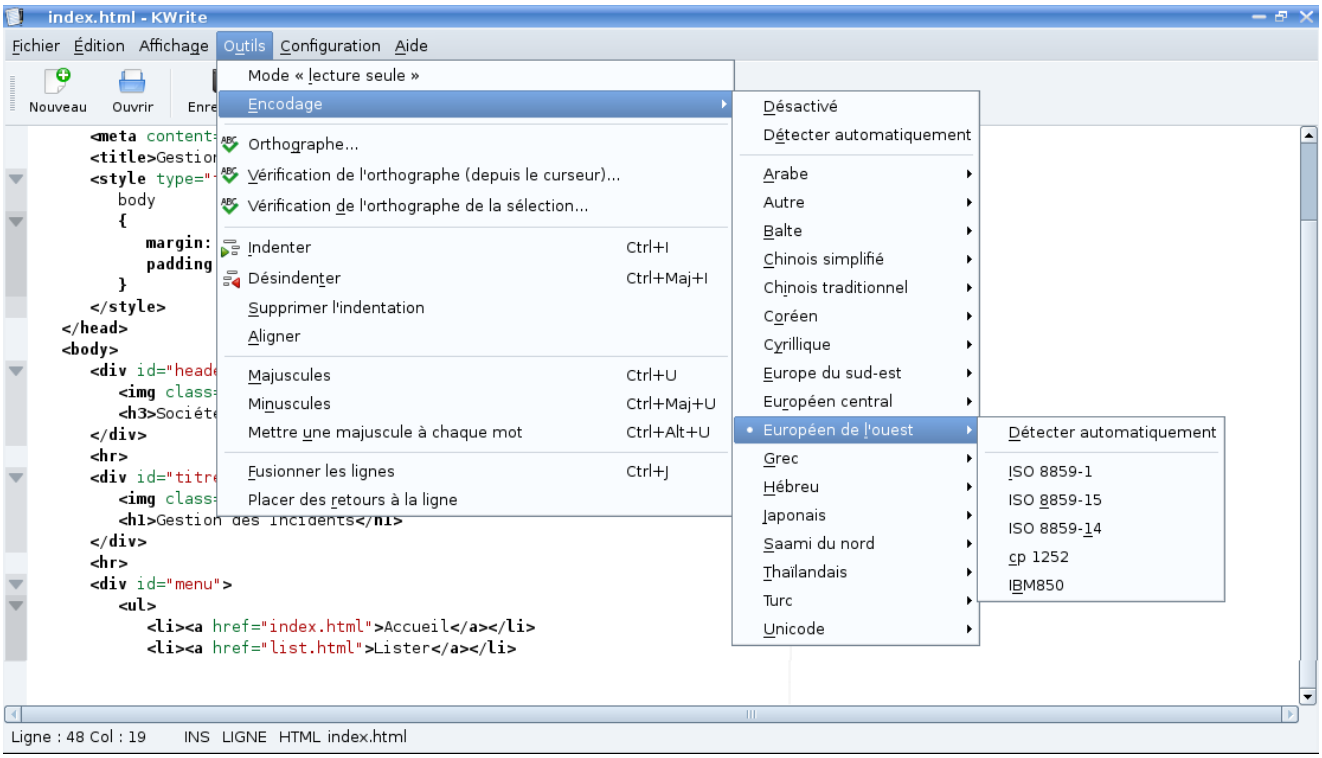
Exemples de références de caractères :

- < → < (permet d'éviter la confusion avec la balise ouvrante)
- > → > (permet d'éviter la confusion avec la balise fermante)
- © → ©

Choix de l'encodage dans l'éditeur de texte :

- vim (latin1 correspond à ISO-8859-1) :
  : set encoding=latin1(variable d'environnement)
  : set fileencoding=latin1 (l'encodage choisi pour l'enregistrement du fichier)
  : set fileencodings=latin1,utf-8 (liste des encodages)

- kwrite :

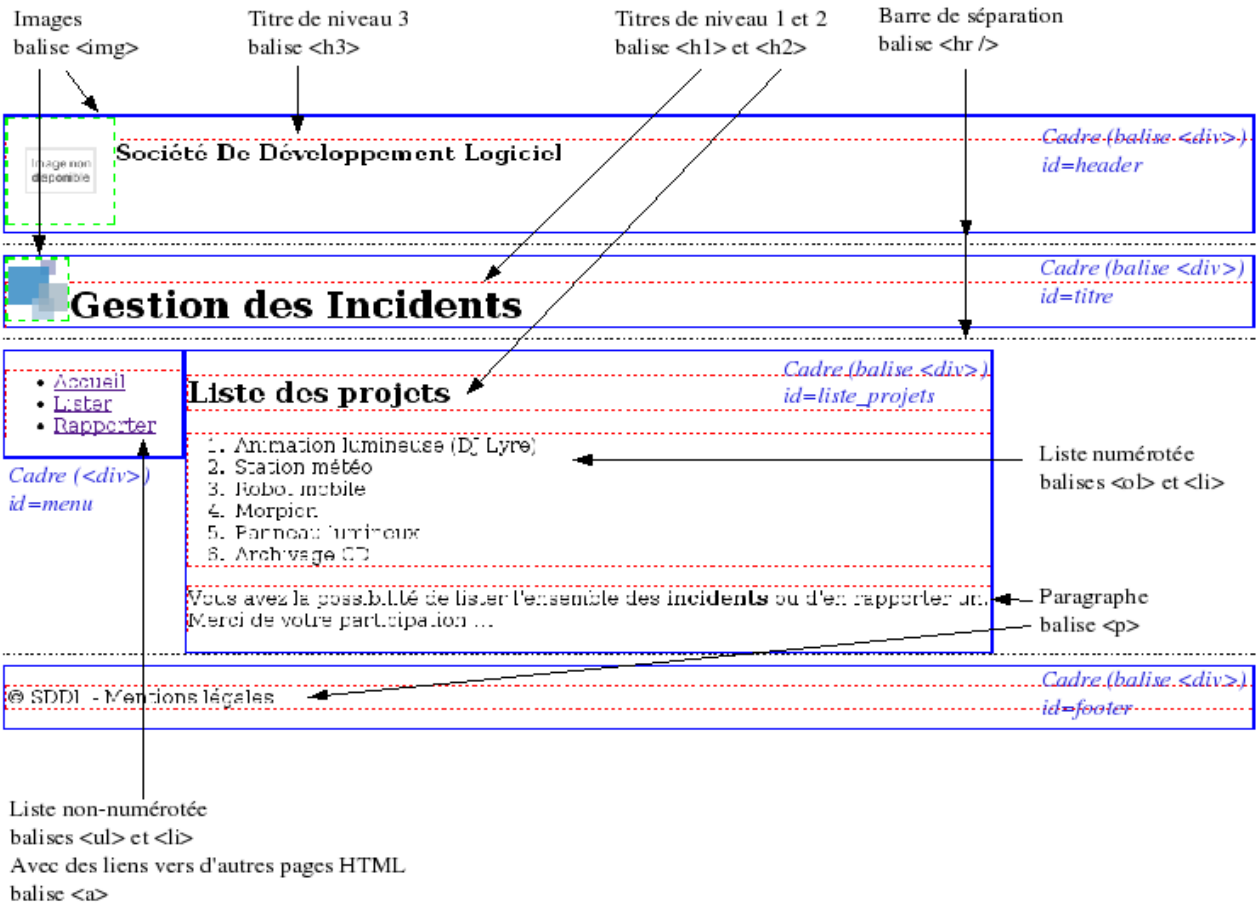


C . Respecter les noms de fichier indiqués dans le TP. Dans tous les cas, ne pas utiliser d'espace, de majuscules et de caractères spéciaux (pas d'accents notamment).

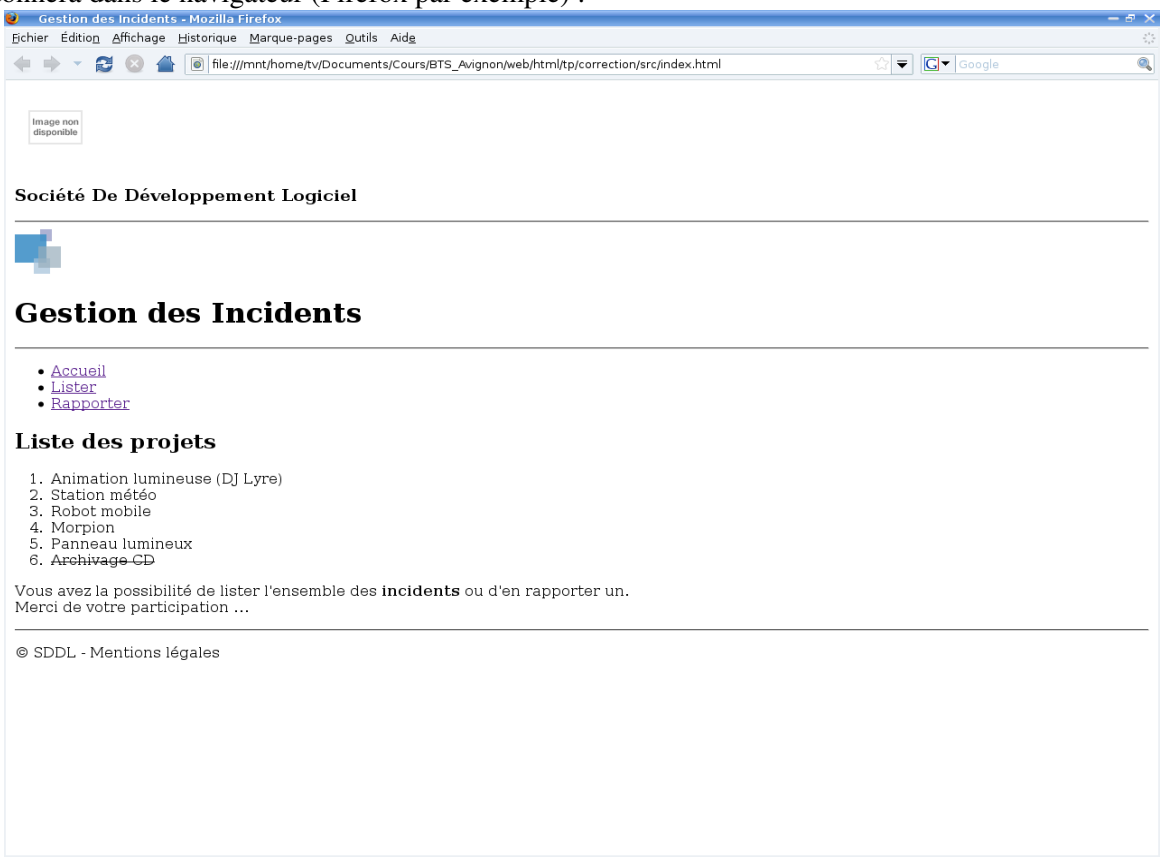
Screenshot

Le premier travail est de définir une structure à son document (le fond) et de ne pas se focaliser sur son apparence dans le navigateur (la forme). La mise en forme du document est une tâche qui sera réalisée plus tard (en séquence 4) notamment grâce à l'utilisation des feuilles de style et en respectant une charte graphique.

Dans cette première séquence, ce travail est important car les choix faits seront réutilisés dans les autres pages. En quelque sorte, cette première page va servir de *squelette* pour les autres.



Ce qui donnera dans le navigateur (Firefox par exemple) :



Travail demandé

1 . Réaliser la page d'accueil **index.html** demandée.
2 . Valider la page **index.html** en utilisant le service en ligne du W3C : http://validator.w3.org/check.
3 . Pourquoi ne devrait-on pas utiliser les balises suivantes : <b> et <strike> ?

Séquence 2 : les tableaux

Objectifs

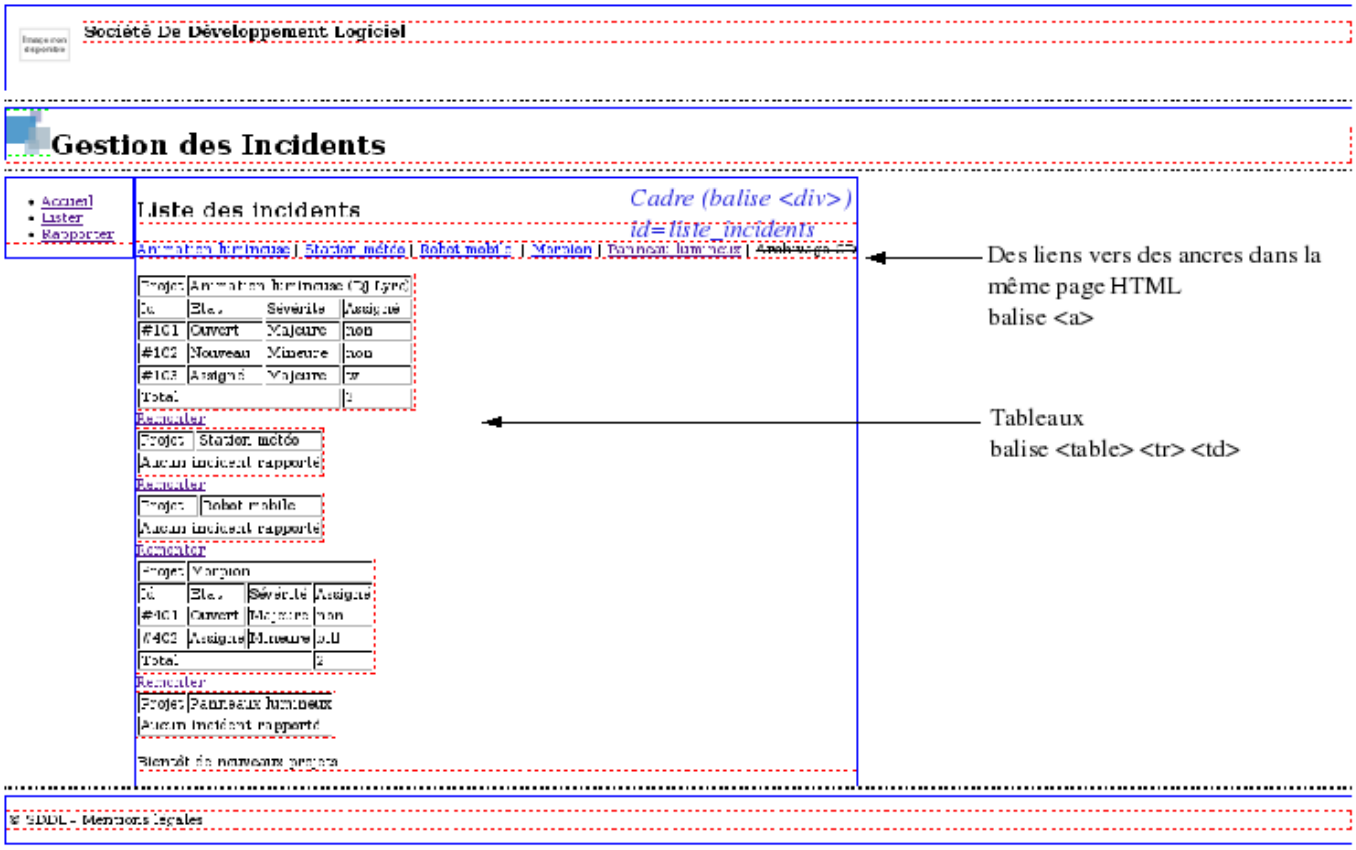
Cette séquence a pour objectif de réaliser la page HTML qui permet de lister l'ensemble des incidents. Elle met en oeuvre l'utilisation des tableaux. Pour naviguer plus facilement dans le codument, on utilisera des liens internes (avec des ancres).

Moyens disponibles

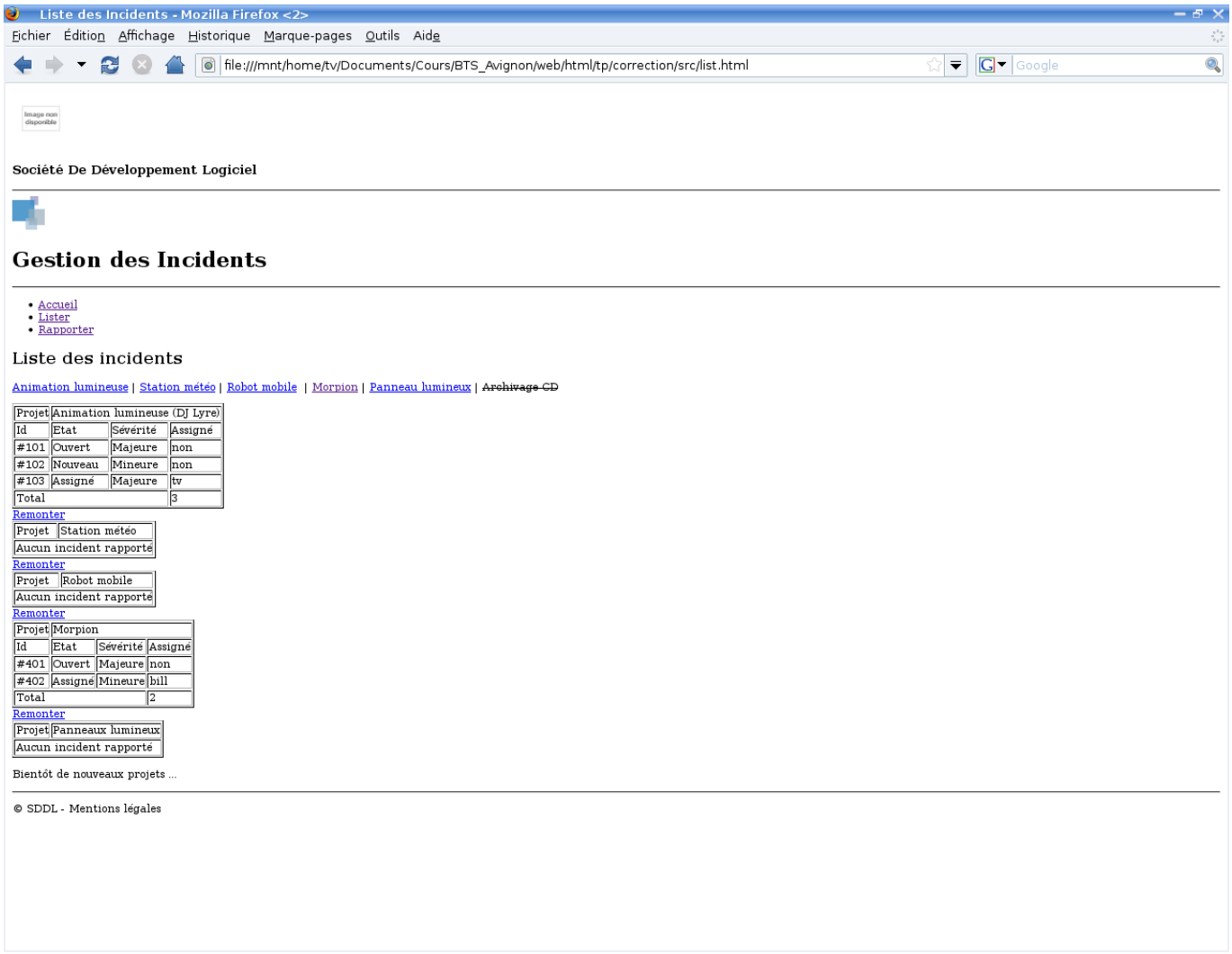
Au choix : éditeurs de texte ou logiciels ***Qanta+, Bluefish, Screem ou Komposer*** pour la réalisation de la page
Un navigateur (***Firefox, IE, Opera, ...***) pour visualiser la page HTML réalisée

Screenshot

Evidemment, on retrouve la même structure, seul le bloc central change de contenu :



Ce qui donnera dans le navigateur (Firefox par exemple) :



Travail demandé

1 . Réaliser la page **list.html** demandée.
2 . Valider la page **list.html** en utilisant le service en ligne du W3C : http://validator.w3.org/check.
3 . Tester les liens internes.

Séquence 3 : formulaire

Objectifs

Cette séquence a pour objectif de réaliser la page HTML qui permet de rapporter un incident. Elle met en oeuvre l'utilisation des formulaires.

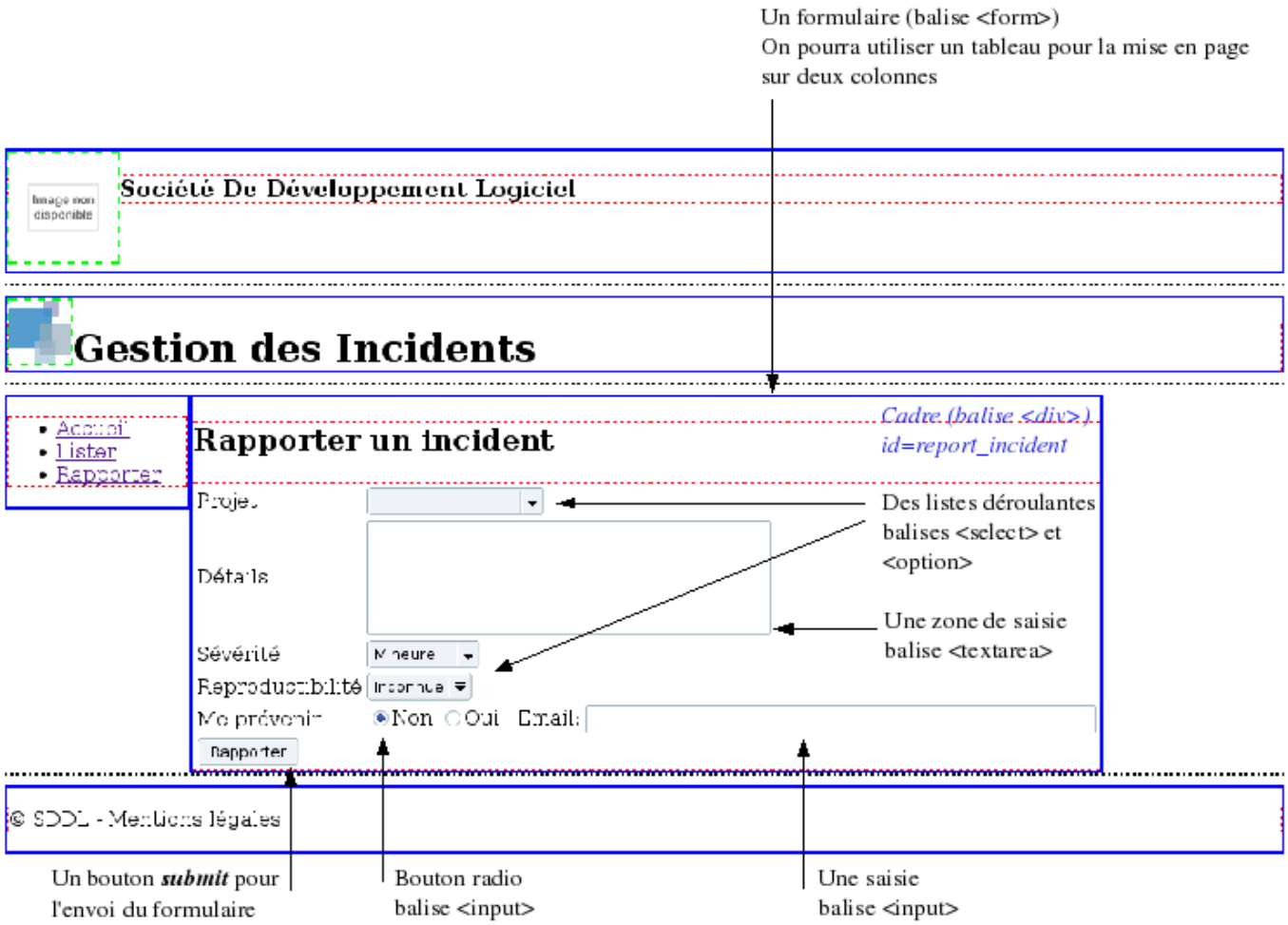
Moyens disponibles

Au choix: les logiciels ***Qanta+, Bluefish, Screem ou Mozilla Composer*** pour la réalisation de la page HTML
Un navigateur (***Konqueror***, ***Mozilla***, ***Firefox*** ou ***Opéra***) pour visualiser la pages HTML réalisées

Travail préliminaire

Créer une page **test.html** vierge.

Screenshot



Contraintes

A . La présentation du formulaire :
On utilisera un tableau à 2 colonnes pour la mise en page du formulaire.

On considère que l'utilisateur qui décide de rapporter un incident se doit de fournir les informations suivantes :

- d'indiquer le projet (par défaut, on ne pré-sélectionnera aucun projet, sinon la liste des projets ouverts)
- de décrire en détails l'incident (l'incident ne doit traiter de qu'un seul problème)
- de préciser la sévérité (*au choix:* Mineure, Majeure, Critique ou Bloquante)
- d'indiquer la reproductibilité (*au choix:* Inconnue, Parfois, Toujours ou Aléatoire)
- de donner son adresse email s'il veut être prévenu sur la résolution de l'incident)

B . Les données du formulaire :

Les données du formulaire seront envoyées à un destinataire précisé par l'attribut ***action*** de la balise **<form>**. On peut indiquer, dans l'attribut ***action***, soit une adresse email (action="mailto:@email") soit l'url d'une page (le plus souvent vers un CGI ou un script PHP).

Toutes les données des éléments du formulaire doivent être nommées (attribut ***name*** **et** ***id***) afin d'être identifiables par le programme de traitement destinataire (notion de variables) ou même par le destinataire du mail. D'autre part, le concepteur du formulaire doit souvent fixer les valeurs associées aux éléments (attribut ***value***) :

- le formulaire : name="report"
- le projet : name="projet" – value="1" pour le projet n°1, etc ...
- les détails : name="details" – value="" (la saisie de l'utilisateur)
- la sévérité : name="level" – value="1" pour Mineure, "2" pour Majeure, etc ...
- la reproductibilité : name="reproduct" – value="1" pour Inconnue, "2" pour Parfois, etc ...
- choix d'être prévenu : name="inform" – value="oui" pour Oui et "non" pour Non
- son adresse email : name="email" – value="" (la saisie de l'utilisateur)
- l'envoi du formulaire : name="action" – value="Rapporter"

C . L'envoi du formulaire :

On distingue deux méthodes d'envoi des données du formulaire en utilisant l'attribut ***method*** : **GET** ou **POST**. Avec POST, les données envoyées au serveur se trouvent dans le **corps de la requête HTTP**. Avec **GET**, les données envoyées au serveur se trouvent dans **l'URL** précisée dans l'attribut ***action*** *(*cela donnera par exemple ***url?month=fevrier&day=24)***.

Il faut aussi indiquer la manière avec laquelle les données du formulaire seront encodées en renseignant l'attribut ***enctype*** (*text/plain*, *multipart/form-data*, ...).

Travail demandé

1 . Réaliser la page **report.html** demandée en respectant les contraintes énoncées ci-dessus.

2 . Valider la page **list.html** en utilisant le service en ligne du W3C : http://validator.w3.org/check.

3 . Test unitaire d'envoi du formulaire avec les balises **<form>** suivantes :

a . <form action="mailto:tvtsii@free.fr" name="report" method="POST" enctype="text/plain">

b . <form action="test.html" name="report" method="GET" enctype="text/plain">

c . <form action="test.html" name="report" method="POST" enctype="text/plain">

d . <form action="http://192.168.52.83/~ tv/report.php" name="report" method="POST">

Indiquer, pour ces quatre tests, les résultats obtenus.

Réponses :

Séquence 4 : feuille de style CSS

Objectifs

Cette séquence a pour objectif d'appliquer une charte graphique à l'ensemble des pages HTML réalisées. Elle met en oeuvre l'utilisation des feuilles de style CSS (*Cascading Style Sheets*). Le principe est simple : il s'agit de redéfinir le rôle d'une balise html en lui imposant de nouvelles propriétés.

Contraintes

A . Fichier externe unique

Les **feuilles de styles CSS** améliorent la qualité des mises en page sur le web (positionnement des éléments au pixel près), diminuent le temps d'affichage et factorisent le code, le rendant ainsi plus lisible et plus facile à maintenir. Le simple changement d'une feuille de styles permet de modifier la présentation du document.

Dans le fichier CSS, il faut aussi faire la séparation entre structure et décoration :

- la **structure** contiendra le positionnement des éléments dans la (ou les) page(s) **ET**
- la **décoration** s'occupera notamment de la couleur et du graphisme

Remarque : en pratique cela pourrait donner deux fichiers CSS : un pour la structure, et un autre pour la présentation. Dans notre cas, on utilisera qu'un SEUL fichier dans lequel on placera de grandes barres de commentaires pour séparer les différentes parties (/* … */). Dans tous les cas, il faut faire la séparation.

On utilisera donc un fichier externe unique **incidents.css** pour redéfinir le style des balises utilisées dans les pages HTML du site. Ce fichier CSS ne doit pas contenir de code HTML. Chaque page du site devra donc indiquer qu'elle utilise ce fichier CSS de la manière suivante :

```
<head>
  <title>Le titre de la page</title>
  <link rel="stylesheet" href="incidents.css" type="text/css">
</head>
```

B . La charte graphique

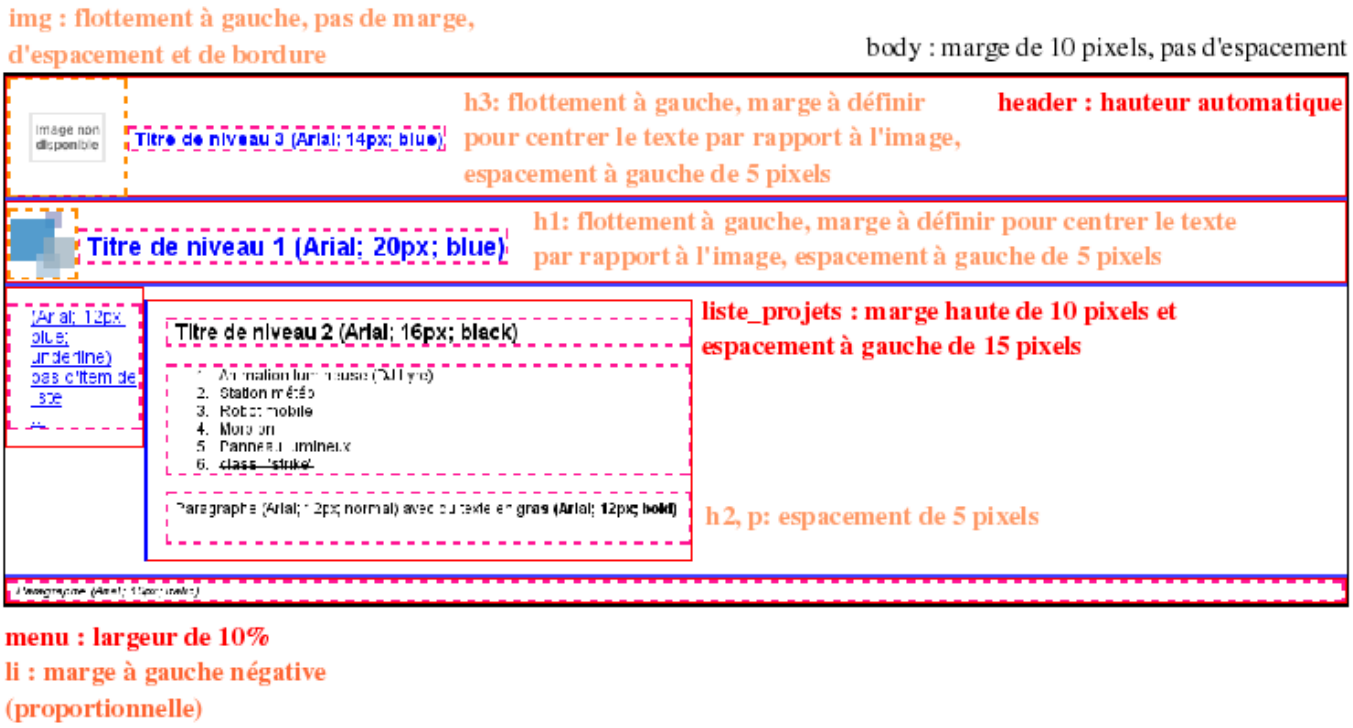
Les séquences précédentes nous ont amené à définir une structure aux documents créés. Il reste à définir, grâce aux feuilles de style, l'interprétation visuel faite par le navigateur.

On découpera ce travail en deux parties :

- la structure visuel des éléments : dimensions, positionnement (flottant, absolu) et leurs relations
- la décoration visuelle des éléments : police, couleur, ...

L'objectif est d'obtenir une cohérence (graphique) pour l'ensemble des pages et de faciliter les tâches de maintenance (modifications).

Quelques élements de structure à respecter :



Screenshot

La page d'accueil **index.html** :

La page **list.html** :

Société De Développement Logiciel

Gestion des Incidents

Accueil
Lister
Rapporter

Liste des incidents

Animation lumineuse | Station météo | Robot mobile | Morpion | Panneau lumineux | ~~Archivage CD~~

Projet	Animation lumineuse (DJ Lyre)		
Id	Etat	Sévérité	Assigné
#101	Ouvert	Majeure	non
#102	Nouveau	Mineure	non
#103	Assigné	Majeure	tv
Total			3

Remonter

Projet	Station météo
Aucun incident rapporté	

Remonter

Projet	Robot mobile
Aucun incident rapporté	

Remonter

Projet	Morpion		
Id	Etat	Sévérité	Assigné
#401	Ouvert	Majeure	non

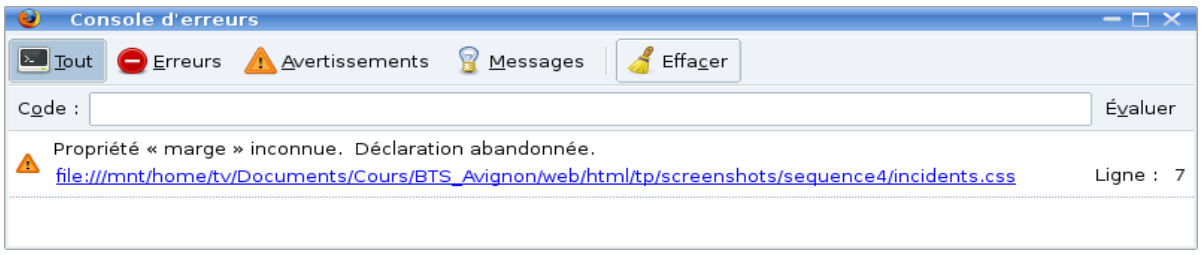
Remarque : on utilisera une ***class*** pour spécifier l'affichage des liens « *Remonter* » (Arial; 10px; italic)

La page **report.html** :

Travail demandé

1 . Réaliser le fichier **incidents.css** demandé en respectant les contraintes énoncées ci-dessus.

2 . Mettre au point sa feuille de style en utilisant la console d'erreurs du navigateur Firefox (Outils → Console d'erreurs)



3 . Valider sa feuille de style en utilisant l'outil de validation du W3C : http://jigsaw.w3.org/css-validator/

4 . Intégrer, dans les pages du site, la feuille de style CSS créée.

Séquence 5 : Bilan

Objectifs

Cette séquence a pour objectif de faire un bilan sur l'étape n°1 de la réalisation du site.

Travail demandé

1 . Détailler le travail du développeur du site lorsqu'un nouvel incident est rapporté ? Lorsqu'un nouveau projet est créé ?
Réponses :

2 . A partir des réponses précédentes, comment qualifie-t-on le contenu des pages HTML dans un développement web ?
Réponse :

3 . La langage HTML permet-il de vérifier les données saisies par un utilisateur dans un formulaire ?
Réponse :

4 . Quel est l'interêt des feuilles de style CSS ?
Réponse :

5 . Proposer une grille d'évaluation des éditeurs utilisés dans ce TP. Vous ferez apparaître au moins 5 critères pertinents de comparaison. A présenter sous forme de tableau (exemple fourni).
Réponses :

Logiciel évalué	Coloration syntaxique	WYSIWYG (*What you see is what you get*)	Complétion	Assistant (*Wizard*)	Support CSS

Annexe : Mise en page

Tout d'abord, il faut savoir que le langage HTML d'origine ne prévoyait pas véritablement d'éléments pour assurer sa mise en page. Donc différentes techniques utilisées par les *webmasters* se sont succédées :

- **les cadres (*frames*)**
- **les tableaux**
- et maintenant les **feuilles de styles CSS** (et la balise conteneur **)**

Les cadres (frames)

Les cadres découpent la fenêtre principale en autant de petits cadres, chacun d'eux jouant le même rôle qu'une fenêtre : un document HTML propre avec barres de défilement indépendantes...

Ces cadres peuvent être chargés de manière totalement indépendante : un cadre sera modifié et l'autre maintenu. Ils peuvent interargir entre eux : un choix dans un premier cadre pourra provoquer le chargement d'une nouvelle page dans un deuxième.

Les cadres permettent donc de créer rapidement une mise en page simple pour un site. L'utilisation classique consiste ainsi à présenter sur un cadre à gauche une liste de liens vers les différentes parties d'un site, qui s'affichent dans la fenêtre principale au centre. Cela facilite de plus la maintenance des différentes pages.

Cependant, les moteurs de recherche indexent les pages à l'intérieur des cadres autant que les autres. Il y a donc un risque pour qu'une page indexée de la sorte soit "orpheline", et ne donne pas accès au reste du site. Plus généralement, une telle page ne se suffit souvent pas à elle-même (il y manque parfois les outils de navigation à l'intérieur du site). Enfin, la présence de cadres s'accompagne souvent de celles de barres de défilement horizontales.

Il faut savoir qu'au début des *frames* sur le Web, leur utilisation posait toutes sortes de problèmes. Par exemple, il n'existait aucun moyen de masquer les bordures séparant les différentes zones dans la fenêtre du navigateur, ce qui fait que l'écran était rempli de traits gris et de barres de défilement. Dans la mesure où la place est limitée sur une page Web, on comprend que l'utilisateur ne se réjouisse pas de voir la fenêtre du navigateur ainsi remplie. C'est l'une des raisons pour lesquelles les *frames* ont (eu) mauvaise réputation. Aujourd'hui pourtant, il est possible de rendre invisibles les bordures entre *frames*.

Remarque: les responsables de sites ont constaté que la plupart des internautes choisissent la version sans *frame* lorsqu'elle est proposée.

Avantages

Pour le *webmaster*: gestion et maintenance des pages plus simple et plus rapide

Pour l'internaute: sommaire toujours présent à l'écran, permettant de naviguer plus rapidement d'une page à l'autre sans revenir en arrière.

Inconvénients

Pour le *webmaster*:

- Pages orphelines (chaque page HTML composant un *frameset* peut être appelée directement par son adresse URL, indépendamment de son environnement *frameset*).
- Se retrouver avec des *frames* dans les *frames* (un site à base de *frames* peut appeler un autre site à base de *frames* etc ...)

Pour l'internaute:

- Sauvegarde ou impression d'une page problématique
- Non accessibilité à l'information pour les aveugles et mal-voyants. (http://www.braillenet.jussieu.fr/education/livreblanc/)
- Espace réduit pour le contenu
- Complexité de la navigation: les *frames* peuvent constituer un véritable labyrinthe, notamment quand le *webmaster* a concocté un *frameset* encapsulant des sites externes, ou des sommaires en cascades.

Exemple

```
<HTML>
  <HEAD>
    <TITLE>Un document avec deux cadres verticaux</TITLE>
  </HEAD>
  <FRAMESET cols="25%,75%" frameborder="0">
    <FRAME src="menu.html" name="menu">
    <FRAME src="accueil.html" name="principal">
  </FRAMESET>
</HTML>
```

Les tableaux

Originellement prévus dans le but de mettre en forme des données, les tableaux ont rapidement été détournés par les développeurs web pour combler un vide qui faisait cruellement défaut au HTML à l'époque : la mise en page.

En effet, dans ce qu'il convient d'appeller la préhistoire du web, point de CSS ni de séparation entre le contenu et la présentation. Les développeurs pensaient avoir trouvé la parade avec les tableaux, et ont donc commencé à utiliser massivement ces derniers pour du positionnement et autres mises en page complexes.

Les tableaux sont franchement inadaptés aux exigences d'aujourdhui, à cause de leurs limites notamment en termes d'accessiblité, de facilité d'utilisation et de maintenance.

Remarque : il est important de noter que les tableaux remplissent parfaitement leur rôle quand il s'agit de structurer des données.

Inconvénients

- Lenteur d'affichage : un tableau est un élément complexe à traiter, notamment pour définir les tailles de chaque cellule et ligne, et nécessite d'analyser toutes les cellules ... surtout si les tableaux sont imbriqués

- Page illisible pour les navigateurs non-graphique (ne pas oublier l'existence de navigateurs spécialisés pour les handicapés, avec plage braille ou synthétiseur vocal, qui ne peuvent pas restituer les informations graphiques)

- Impression problématique : changer complètement le *design* d'une page pour permettre une meilleure impression est impossible ce qui amène souvent à faire une page separée pour l'impression, et avec les coûts associés à sa création et à sa maintenance, et cela se transforme en un gâchis d'énergie et de ressources considérables.

- Complexe et coûteux à produire (même avec des éditeurs WYSIWYG)

- Lourdeur des pages : beaucoup de balises pour peu de contenu !

Les feuilles de style CSS (Cascading Style Sheets)

Les feuilles de style CSS permettent la mise en forme du contenu des pages Web.

Avantages

Les **feuilles de styles CSS** améliorent la qualité des mises en page sur le web (positionnement des éléments au pixel près), diminuent le temps d'affichage et factorisent le code, le rendant ainsi plus lisible et plus facile à maintenir.

Le simple changement d'une feuille de styles permet de modifier la présentation du document.

Exemple

```
<html>
<head>
  <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
  <title>Exemple - CSS</title>
  <link rel="stylesheet" href="./style.css" type="text/css">
</head>
<body>
<div id="header">
   <h1>En-t&ecirc;te</h1>
</div>

<div id="left">
   Texte du c&ocirc;t&eacute; gauche...
</div>

<div id="right">
   Texte du c&ocirc;t&eacute; droit...
</div>

<div id="middle">
   Texte du milieu...
</div>

<div id="footer">
   Texte du pied de page...
</div>

</body>
</html>
```

La feuille de style **style.css** :

```
/* Structure */

  body
  {
    margin: 0px;
    padding: 0px;
  }

  div#header
  {
    padding: 1px;
    clear: both;
    height: 50px;
  }

  div#left
  {
    float: left;
    width: 150px;
  }

  div#right
  {
    float: right;
    width: 150px;
  }

  div#middle
  {
    margin: 0px;
    padding: 0px 160px 5px;
  }

  div#footer
  {
    clear: both;
    border: 1px solid black;
  }

/* Décoration */

  div#header
  {
    background-color: rgb(51, 204, 255);
  }

  div#left
  {
    background-color: #FF8822;
  }

  div#right
  {
    background-color: white;
  }

  div#middle
  {
    text-align: center;
    font-family: Verdana, Geneva, Lucida, Arial;
    font-size: 12px;
    color: white;
    background-color: silver;
  }

  div#footer
  {
    background-color: yellow;
  }
```

Liens pour en savoir plus

Spécification DOM: http://xmlfr.org/w3c/TR/REC-DOM-Level-1/
Spécification HTML 4: http://www.la-grange.net/w3c/html4.01/
Spécification XHTML: http://www.la-grange.net/w3c/xhtml1/
Spécification CSS2: http://www.yoyodesign.org/doc/w3c/css2/cover.html

Les standards du Web: http://www.openweb.eu.org/
CSS pratique: http://pompage.net/pompe/csspratique/

