

Manuel de travaux pratiques

Code : HIT

Version : 1.0.1

Développement web HTML5/CSS3

Table of contents

TP 1 : MA PREMIERE PAGE HTML5	3
TP 2 : WEB FORMS 2	6
TP 3 : DU SON ET DES IMAGES	8
TP 4 : COMMUNICATION	10
TP 5 : WEB WORKERS	12
TP 6 : CSS3	13
TP 7 : OFFLINE	15
TP 8 : DRAG AND DROP	17

TP 1 : Ma première page HTML5

1 Objectifs

Créer une première page HTML5 avec les balises sémantiques.
Utiliser quelques sélecteurs CSS.
Vérifier la compatibilité des navigateurs.

2 Configuration

Vous utiliserez un éditeur de texte (comme NotePad) mais vous pourriez tout aussi bien utiliser un éditeur HTML5 de votre choix (type Maquette ou BlueGriffon ou un produit du commerce).
La feuille de style (CSS) utilisée pour cette page vous est fournie mais vous devrez l'utiliser. Elle se trouve dans le répertoire (questions/tp1) ainsi que les images dont vous aurez besoin.
Vous testerez avec un navigateur compatible comme Firefox, Chrome ou IE9 puis vous essaieriez de faire fonctionner l'exemple avec IE8.

3 Etapes

3.1 Création de la page

Notre première page constitue la page d'accueil du nouveau site web de Valtech-training. Les composantes de la page et donc les balises que vous utiliserez seront :

- un header contenant le logo et une ligne de texte puis un ensemble de liens constituant le menu du site inclus dans un tag de navigation. L'ensemble des styles sont fournis dans la feuille de style vt.css. Dans la CSS cela correspond à la partie header et nav. Les conteneurs de base seront des div ou des span classiques.
- un contenu principal composé d'un article (évidemment on pourrait en avoir plusieurs).
- un bas de page (footer) avec les images et une ligne de texte. Là encore il vous faudra étudier la feuille de style pour comprendre la structure exacte de la page.

Le but de tout ça est aussi de vous permettre de replonger dans la syntaxe CSS/HTML avant d'aller plus loin dans les nouveautés HTML5/CSS3.

Voici la structure globale du header qu'il faudra compléter à l'aide de la CSS fournie ainsi que du résultat attendu :

```
<header>
  <div class="inner group">
    <hgroup>
    </hgroup>
    <nav>
      <ul id="menu-main_nav" >
      </ul>
    </nav>
  </div>
</header>
```

La figure ci-dessous vous donne un aperçu du résultat attendu.

Valtech Training

HTML5 and much more ...

[Contact](#)
[Inscription](#)
[FeedBack](#)
[Quizz](#)

Introducing the HTML5/CSS3 training

Aug 1st, 2011 by Xav.

FEATURED

HTML5 is for the time being not fully supported by popular browsers, but most of them are progressively evolving towards this new version of the famous HTML language. Thus, we can yet play with a lot of new features and that is what we are going to do !!!

[Read full post](#)

RECENT COMMENTS


Olivier : [I can't wait](#)

Stéphane : [Works also in IE9 \(?\) ;\)](#)


Denis : [And for ipads ?](#)

Anne-lise : [Women love it too](#)

Jocelyn : [pfff, marketing again ...](#)



This is a training web site in HTML5. Feel free to use and modify it as you wish.



Le corps de la page est vu comme un ensemble d'articles présentant l'actualité de la société. Ca ressemble typiquement à un blog mais nous nous sommes limités à un premier article par souci de simplicité. Plus précisément la structure du div principal (le centre de la page) sera la suivante :

```

<article id="opener" >
  <div class="main">
    <article class="post">
      <header>
      </header>
      <div class="entry">
      </div>
    </article>
  </div>
  <aside class="secondary">
    <section class="mod">
    </section>
  </aside>
</article>

```

3.2 Modification du style

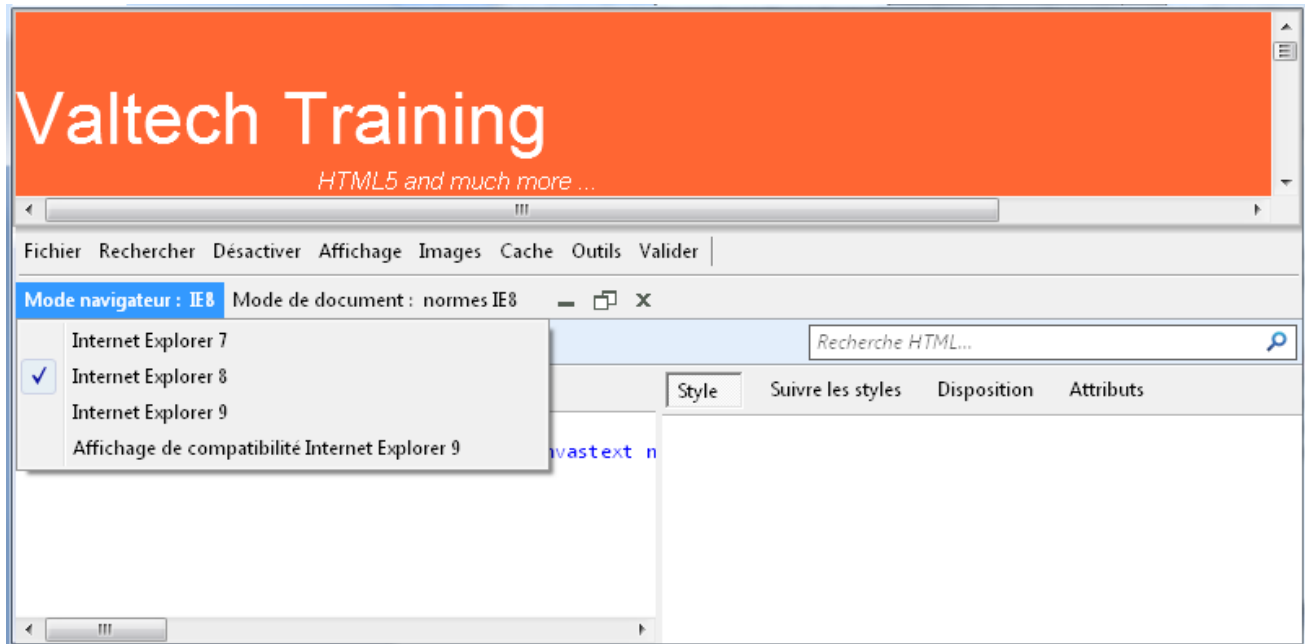
Ajouter un lien en haut et à droite de l'IHM permettant de changer les couleurs de fond du header, de la barre de navigation ainsi que des textes des liens. L'idée est d'ajouter et retirer un style CSS via JavaScript pour jouer entre le bleu de base et l'orange VT.

Etape optionnelle : essayez de rajouter le lien dynamiquement avec du code Javascript. Ce type de manipulation est assez pénible en javascript (sans bibliothèque) et les possibilités HTML5 sont encore peu supportées. Notez qu'IE9 supporte la fonction `insertAdjacentHTML` (qui était pendant longtemps une fonction spécifique d'IE) mais malheureusement IE9 ne permet pas de faire un 'toggle' sur une classe CSS ... Bref pas de solution simple pour l'instant à ce problème !

3.3 Modernizr

Pour commencer vous allez ouvrir votre page avec IE9 si vous n'avez pas encore testé. Ensuite vous regarderez ce que cela donne avec IE8. Pour cela vous ouvrez la barre d'outils de développement d'IE9 (F12 pour y accéder rapidement) et vous sélectionnez le mode navigateur IE8. Vous devriez avoir des modifications importantes de votre page.

Vous intégrerez ensuite le script Modernizr (fourni lui aussi, là où vous avez trouvé la css) puis recommencerez l'opération.



TP 2 : Web Forms 2

1 Objectifs

Créer un formulaire en utilisant les nouveaux champs et les possibilités de validation associées.

2 Configuration

Opéra a une longueur d'avance sur l'implémentation du champ date, il est donc recommandé de l'utiliser mais vous vérifierez le résultat sur d'autres navigateurs.

3 Etapes

3.1 Le formulaire et sa validation

Détails des champs utilisés :

- nom : champ obligatoire qui gagnera le focus au chargement de la page.
- prénom : champ obligatoire
- email : utilisation du type email, champ obligatoire
- téléphone : utilisation d'un placeholder pour créer un masque de saisie correspondant au format du numéro de téléphone, champ obligatoire
- nombre de places : utilisation d'un champ de type number
- code de la formation : utilisation d'un datalist, champ obligatoire
- lieu de la formation : utilisation d'un datalist, champ obligatoire
- date de début de session : utilisation du type date, champ obligatoire
- CB : champ texte, champ obligatoire

Vous ajouterez un bouton de soumission du formulaire.

Vous regarderez le rendu du champ date (et des autres aussi) sur Opéra, Chrome et Firefox.

Nous allons tester plusieurs formes de validation en fonction des types de champ.

- Vérification des champs obligatoires
- Vérification des types de champs spécifiques : date, tel, email, number
- Vérification par expression régulière : le champ carte bleue portera une expression régulière pour vérifier qu'il contient bien 16 chiffres : `[0-9]{16}`.

Vous observerez le comportement du navigateur en cas d'erreur.

3.2 Validation pas à pas

La validation effectuée par le navigateur :

- peut présenter des différences d'un navigateur à l'autre
- n'est effectuée par défaut qu'à la soumission du formulaire

Si cela ne nous convient pas, on peut interagir avec l'API de validation. Nous allons pour cela implémenter une validation pas à pas. Le principe en est simple, dès que l'utilisateur sort d'un champ, nous allons demander au navigateur de valider cette valeur.

En outre nous ajoutons une progress bar en dessous du formulaire. A chaque champ valide nous mettons à jour la barre de progression et affichons le pourcentage de complétion du formulaire.

Infos supplémentaires :

Les valeurs possible de l'objet validity sont

- `valueMissing.`
- `typeMismatch.`
- `patternMismatch.`
- `tooLong.`
- `rangeUnderflow.`
- `rangeOverflow.`
- `stepMismatch.`
- `customError.`
- `valid`

et notez au passage que la syntaxe CSS reconnaît le sélecteur `:invalid` pour changer le style d'éléments invalides (mais c'est souvent fait par défaut par le navigateur).

Etapas supplémentaires :

Rendre le bouton inactif jusqu'à ce que le pourcentage de la progress bar soit à 100%.

Remplacer un des messages d'erreur du navigateur par le vôtre (ou ajouter un message sur un champ spécifique qu'un navigateur ne prend pas en compte).

TP 3 : Du son et des images

1 Objectifs :

Utiliser les possibilités audio et vidéo.
Réaliser un graphique côté client avec Canvas et SVG.

2 Configuration

Les fonctionnalités graphiques ou video font partie des mieux supportées par les navigateurs. Nous pourrons donc utiliser différents navigateurs pour ce TP sachant cependant que ceux-ci ne lisent pas les mêmes codecs vidéo. La vidéo à intégrer se trouve donc dans le répertoire tp3 en quatre versions différentes.

3 Etapes

3.1 Vidéo

Il s'agit simplement d'afficher la vidéo fournie dans une nouvelle page. La vidéo doit s'afficher dès qu'elle est chargée.

Maintenant nous voulons que l'utilisateur puisse contrôler le démarrage et la mise en pause de la vidéo en déplaçant la souris. La vidéo ne se lance donc plus automatiquement mais seulement lorsque le pointeur de la souris est dessus et elle s'arrête si le pointeur n'y est plus.

3.2 Canvas

Il s'agit d'implémenter un petit questionnaire d'évaluation pour la formation. Une fois les valeurs saisies, elles s'afficheront dans un graphique situé au dessous du formulaire.

Nous allons utiliser un champ input de type range, pouvant aller de 0 à 8 par incrément de 2 soit 5 positions correspondant respectivement à Mauvais, Insuffisant, Moyen, Bon et Excellent. Pour aider les stagiaires dans leur saisie, cette valeur textuelle doit apparaître à coté du champ lorsqu'une valeur est sélectionnée (c.f. figure ci-dessous). Les éléments à noter sont : le manuel, le formateur, les travaux pratiques et les locaux.

Saisissez votre évaluation

Nom

E-Mail

Manuel Insuffisant

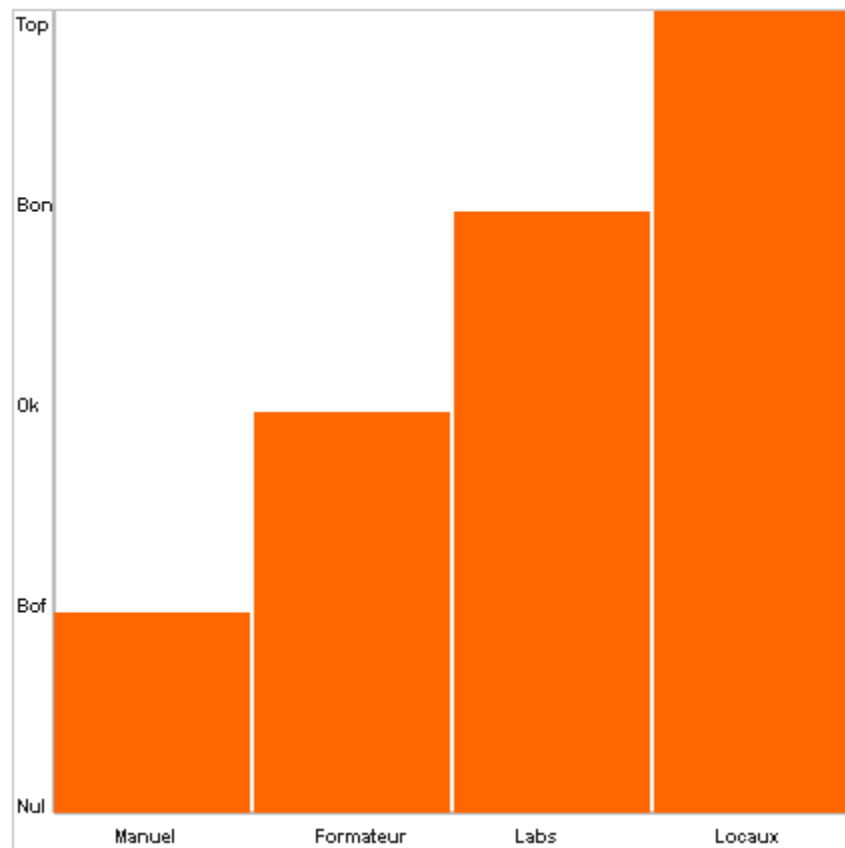
Formateur Moyen

TP Bon

Locaux Excellent

Le bouton afficher permet de visualiser un histogramme de l'évaluation. Il contient aussi deux axes de coordonnées avec des légendes comme dans la figure ci-dessous. Nous irons puiser de l'aide dans l'aide mémoire Canvas (répertoire 'Learning Aids') pour trouver les méthodes nécessaires.

La zone d'affichage fait 420*420 pixels dont 20 sont utilisés pour la légendes des deux axes de coordonnées.



3.3 SVG

Avec HTML5 le format SVG devient lui aussi une possibilité d'affichage standard. Certes on peut regretter d'avoir des formats différents permettant d'aboutir au même résultat. Pourtant il existe bien évidemment des différences entre ces deux façons de faire. Pour cela, vous pouvez lire l'article <http://blogs.msdn.com/b/davrous/archive/2011/05/09/introduction-aux-apis-graphiques-d-html5-svg-amp-canvas-2-4.aspx> mais en attendant pour vous faire une opinion par vous-même, nous allons reprendre l'exemple précédent à iso-fonctionnalité mais en SVG.

Pour vous aider voici les éléments de syntaxe SVG permettant de tracer une ligne, puis un rectangle et enfin d'écrire du texte.

```
<line x1="" y1="" x2="" y2="" style="stroke:rgb(99,99,99);stroke-width:2"/>
```

```
<rect id="" x="" y="" width="" height=""
style="fill:orange;stroke:black;stroke-width:1;opacity:0.8">
```

Vous pouvez y ajouter les attributs (rx="" et ry="") pour arrondir les angles.

```
<text x="" y="" style="font-size:12px;">Top</text>
```

N'oubliez pas que la principale différence avec l'API Canvas réside dans le fait que les éléments SVG font partie du DOM et sont donc accessibles par ce biais.

Un petit plus : animez le graphique pour voir les barres d'histogramme se construire se dessiner.

TP 4 : Communication

1 Objectif

Utiliser les possibilités de communication asynchrone et duplex : AJAX et WebSockets.

2 Configuration

Pour ce TP nous repartons du sujet précédent mais nous avons besoin de pouvoir communiquer avec une partie serveur pour faire la moyenne des évaluations saisies par les stagiaires. Cette partie est fournie sous la forme d'une application web java tournant sur jetty, un des premiers moteurs de servlet à implémenter le protocole des web sockets.

Pour lancer jetty en ligne de commande, il suffit d'exécuter « `java -jar start.jar` » depuis le répertoire d'installation de jetty.

L'application fournie est déployée sous le contexte 'wss' et répond à deux url : '/posteval' et '/evalwss'.

Les fichiers statiques (html, css, js) de cette application doivent être placés à la racine de l'application (soit 'pathjetty/webapps/wss'). Ils sont modifiables même lorsque le serveur tourne et après les avoir modifiés, il suffit de rafraîchir votre navigateur.

3 Etapes

3.1 XMLHttpRequest

Lorsque l'utilisateur a fini de saisir son évaluation, il peut toujours l'afficher mais il dispose d'un deuxième bouton lui permettant de soumettre son évaluation.

Cette transmission se fait avec un appel asynchrone en POST à l'url '/posteval'. Les données soumises doivent être de la forme « `note1=XXX¬e2=...` ».

Attention ce n'est plus le navigateur qui se charge de créer correctement la requête, vous devez donc spécifier le format de paramètre en utilisant le header « Content-Type ». Cela se fait en utilisant la fonction `setRequestHeader` de l'objet XMLHttpRequest avec les paramètres : 'Content-Type', 'application/x-www-form-urlencoded'.

Lorsque le serveur reçoit cette requête il calcule la moyenne des évaluations saisies actuellement et renvoie la réponse sous la forme d'un tableau JSON contenant la note moyenne pour chacune des quatre appréciations (donc [moyenne note1, moyenne note 2 ...]).

Lorsque le navigateur reçoit la réponse du serveur il ne lui reste qu'à afficher le graphique de la moyenne des évaluations.

Ici c'est le client qui est à l'origine de la demande et il ne reçoit que la moyenne des valeurs déjà saisies.

3.2 Web Socket

Nous cherchons maintenant à pouvoir visualiser en temps réel la moyenne des notes saisies par les stagiaires.

Nous ouvrons pour cela une connexion websocket avec l'url 'evalwss' et nous ajouterons une zone dans la page pour afficher l'état de la connexion. Dès que le stagiaire saisit ou modifie une des notes, elle est transmise au serveur (sous la forme 'noteX=XXX'). Le serveur répond, comme dans l'exemple précédent, en renvoyant un tableau JSON contenant les quatre moyennes des notes saisies. Il n'y a plus qu'à les afficher.

Une fois codé, il faut faire tourner cet exemple simultanément dans différentes fenêtres de navigateur pour voir les modifications de l'une se propager automatiquement sur les autres.

Si vous êtes sur un même réseau local, c'est encore mieux en centralisant l'exécution du code serveur pour l'ensemble des stagiaires. Faites attention, plus le nombre de valeurs envoyées au serveur augmente, plus il est dur de faire bouger la moyenne générale et donc de voir l'affichage se modifier. Il faut donc relancer le code serveur de temps en temps.

TP 5 : Web Workers

1 Objectifs

Découper un traitement long en javascript et permettre la réactivité de l'IHM à l'aide des Web Workers. Comparer avec une version n'utilisant pas les web workers.

2 Configuration

Nous repartons une fois de plus de notre formulaire de saisie des évaluations.

3 Etapes

3.1 Sans Web Workers

Pour tester l'utilisation des web workers il faut un traitement un peu consistant et comme le temps manque pour développer un quelconque algorithme en javascript nous allons tout simplement faire une grosse double boucle. La première boucle (boucle externe) va de 1 à 4 (une itération pour chaque note) et la boucle interne va de 1 jusqu'à une valeur maximum que nous déclarerons à part pour pouvoir jouer sur cette valeur plus tard.

A l'intérieur de la boucle interne nous recalculons la hauteur de l'histogramme pour quelle aille lentement de 0 jusqu'à la valeur saisie par l'utilisateur. Plus la valeur max est grande plus on met de temps à arriver à la valeur à afficher. Dès que l'on sort de cette boucle, la boucle principale affiche la barre d'histogramme dont la valeur vient d'être calculée.

Avec une valeur max de 1000000, quel affichage voyons-nous se dessiner ?

3.2 Avec Web Workers

La double boucle est maintenant déléguée à un worker et sera activée par l'envoi d'un message contenant les notes saisies. Nous placerons donc un deuxième bouton dans notre page pour provoquer l'envoi de ce message.

Dès que la boucle interne se termine, le worker envoie un message à la page avec les coordonnées à afficher (x et y).

Le comportement est-il le même ?

Refaites tourner cet exemple en provoquant l'affichage des barres d'histogramme à chaque itération de la boucle interne.

TP 6 : CSS3

1 Objectifs :

Utiliser les sélecteurs CSS3 ainsi que certaines propriétés CSS3.
Réaliser des transitions avec CSS3.

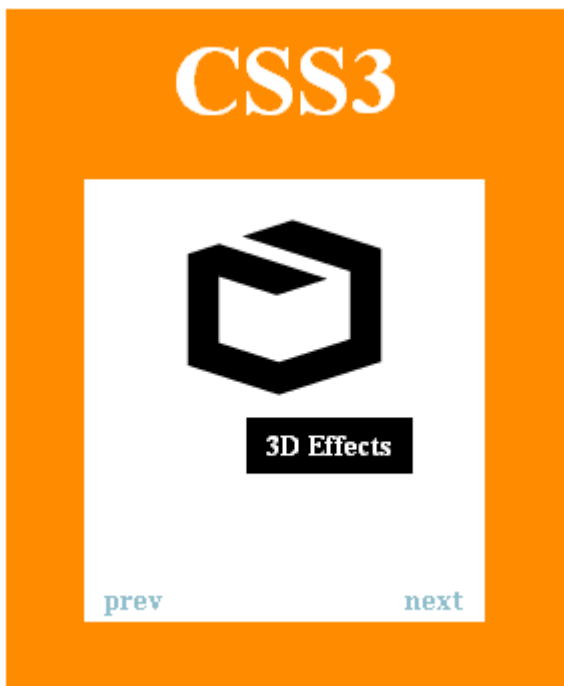
2 Configuration

Le répertoire 'learning-aids' contient un petit guide de référence CSS3 avec différents exemples qui vous permettront de mener à bien ce TP.

3 Etapes :

3.1 Arrondissons les angles

Dans le répertoire du TP6 se trouve une page HTML contenant ses styles CSS. Ci-dessous nous pouvons voir la version initiale et la version à réaliser. Ceci se fait uniquement en jouant sur les propriétés CSS3 : box-shadow, text-shadow, border-radius et l'utilisation des couleurs rgba.



3.2 Soyons sélectifs

Les deux liens 'prev' et 'next' permettent d'afficher l'image précédente et suivante. Toutes les images sont placées au même endroit mais elles ont une opacité de zéro à l'exception de celle qui porte le style 'current' qui fixe son opacité à 1.

Le morceau de code javascript présent dans la page contient une fonction qui reste à implémenter pour rendre la navigation opérationnelle. L'intérêt de cette étape réside essentiellement dans l'écriture du bon sélecteur CSS.

3.3 Passons aux choses sérieuses

Le passage d'une image à l'autre va se faire avec une transition qui durera 1s. Durant cette phase nous appliquerons une transformation 2D à l'image et à son texte. Pour commencer l'image qui disparaît va doubler de taille et celle qui apparaît reprendre sa taille d'origine. La transition étant liée à une transformation, il est possible d'utiliser le mot-clé 'transform' (ou -moz-tranform) mais comme il y a deux éléments à transformer on peut utiliser le mot-clé 'all' (ou rien du tout, bizarrement ça marche aussi).

Nous combinerons ensuite cet effet avec une rotation de 180 degrés.

Enfin nous essayerons la transformation de type matrix qui n'est pas simple à maîtriser mais qui donne des résultats amusants !

TP 7 : Offline

1 Objectifs :

Laisser la possibilité d'utiliser une application lors de la perte de connexion.

2 Configuration :

Nous nous intéressons maintenant à un quizz que le stagiaire peut remplir après la formation pour valider ses acquis. Ce quizz pouvant durer assez longtemps, l'utilisateur doit pouvoir sauvegarder ses réponses au fur et à mesure.

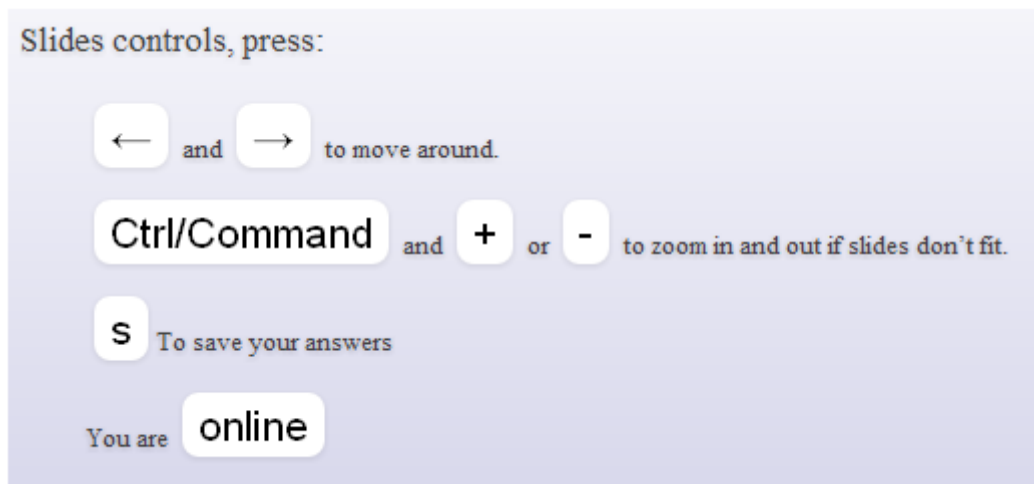
Dans le répertoire consacré à cette étape vous trouverez une page html, une feuille de style css ainsi qu'un fichier javascript. Ce trio permet de visualiser différents écrans (classe Slide dans le code js) sous forme de diaporama (classe Slideshow).

3 Etapes :

3.1 Vérifier la connexion

Pour commencer nous vérifions que nous sommes bien connectés et nous affichons l'état correspondant. Vous devrez pour cela commencer par vous familiariser avec le code fourni.

Nous intercepterons aussi l'évènement de perte de connexion pour en prévenir l'utilisateur. Notre deuxième slide ressemblera donc à la figure ci-dessous.



3.2 Sauvegarde des réponses

Comme nous pouvons le voir sur la figure précédente, lorsque l'utilisateur appuie sur la touche 's' cela provoquera la sauvegarde de la réponse fournie à la question posée par le slide courant. Notez que c'est le keycode 83 qui correspond au s lors de la récupération d'un évènement clavier.

Pour cela nous allons rajouter une question simple, donc nous sauvegarderons le résultat dans le stockage local (localStorage). Si tout se passe bien nous afficherons à l'utilisateur que la sauvegarde a eu lieu comme on le voit sur la figure suivante.

Vous pourrez ensuite rajouter quelques questions supplémentaires et reproduire ce comportement.



Lorsque l'utilisateur retourne sur cette page (en rechargeant l'application par exemple), nous afficherons un message permettant à l'utilisateur de savoir qu'il a déjà répondu à cette question.
 Etape facultative : réafficher la valeur sauvée (donc remettre le formulaire dans son état d'après saisie).

3.3 Cache Applicatif

Chaque serveur doit être configuré pour envoyer le fichier manifeste avec le bon type MIME, en l'occurrence 'text/cache-manifest'. Dans Jetty7 cela se fait dans le fichier webdefault.xml (répertoire etc), en cherchant le tag 'mime-mapping'.

Nous allons définir un cache applicatif contenant les images, css , js et html du quizz. Nous reprenons pour cela la page d'accueil du tout premier tp et nous utilisons son menu pour accéder au quizz (cette page devra donc aussi faire partie du cache applicatif). Nous vérifierons ensuite que l'application est bien accessible même hors connexion.

Nous prévoyons aussi une page qui s'affichera lorsque des pages de la même application ne faisant pas partie du cache sont demandées par l'utilisateur (par exemple la page des évaluations).

Attention : dès que vous avez défini un cache applicatif, les modifications apportées à vos pages ne sont pas chargées automatiquement par le client. Il faut bien penser à sauvegarder le fichier manifest pour que le client pense qu'il a été modifié.

Afin de bien comprendre son fonctionnement nous allons intercepter et afficher les événements liés au cache applicatif.

TP 8 : Drag and drop

1 Objectif

Utiliser le drag and drop natif fourni par HTML5.

2 Configuration

Nous repartons du quizz de l'étape précédente auquel nous allons ajouter des questions.

3 Etapes

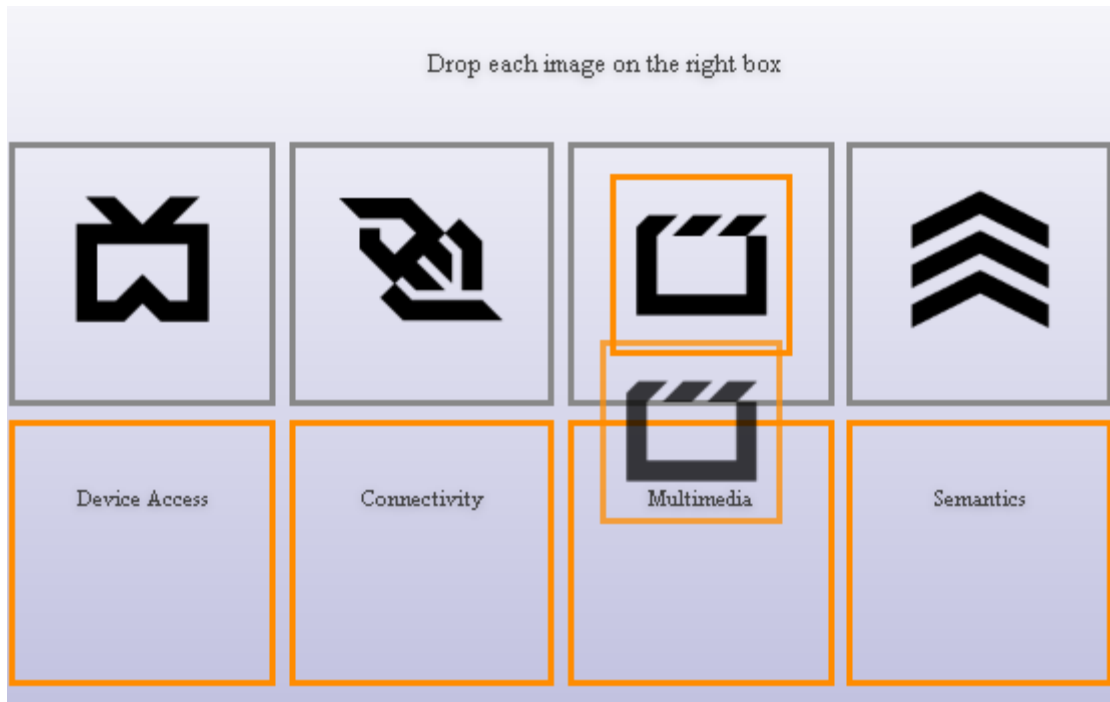
3.1 Drag

Le but de la question est de mettre en correspondance des images et du texte en faisant glisser les images au bon endroit.

Notre point de départ doit donc ressembler à la figure ci-dessous :



Lorsque l'utilisateur commence à bouger un élément nous souhaitons modifier son apparence et montrer à l'utilisateur les emplacements où il peut lâcher l'image. Nous ajouterons donc un style CSS à cet effet comme le montre la figure suivante.



3.2 Drop

La première tâche est d'implémenter dragover pour permettre à l'utilisateur de lâcher l'image sur certaines zones (dropzones)

La deuxième tâche consiste à implémenter le drop. Rappelons-nous que par défaut, le navigateur ne fait rien. L'image lâchée sur une des cases doit maintenant s'y trouver et c'est à vous de vous y coller !