HTML5

Jean-Yves Didier

UFR S&T

A propos du module

Détails pratiques

Intervenant : Jean-Yves Didier (didier@iup.univ-evry.fr)

Volume horaire : 16h (6h CM + 8h TP + 2h partiel)

Prérequis : connaissance de HTML4, XML, CSS

Avertissement

HTML5 est encore en cours d'élaboration !

Certains éléments de ce cours pourront être obsolètes d'ici quelques temps lorsque la spécification définitive de HTML5 sera au point.

HTML

HTML

HTML : HyperText Markup Language.
Langage à balises utilisé pour décrire le contenu des pages web.

Pourquoi HTML5 ?

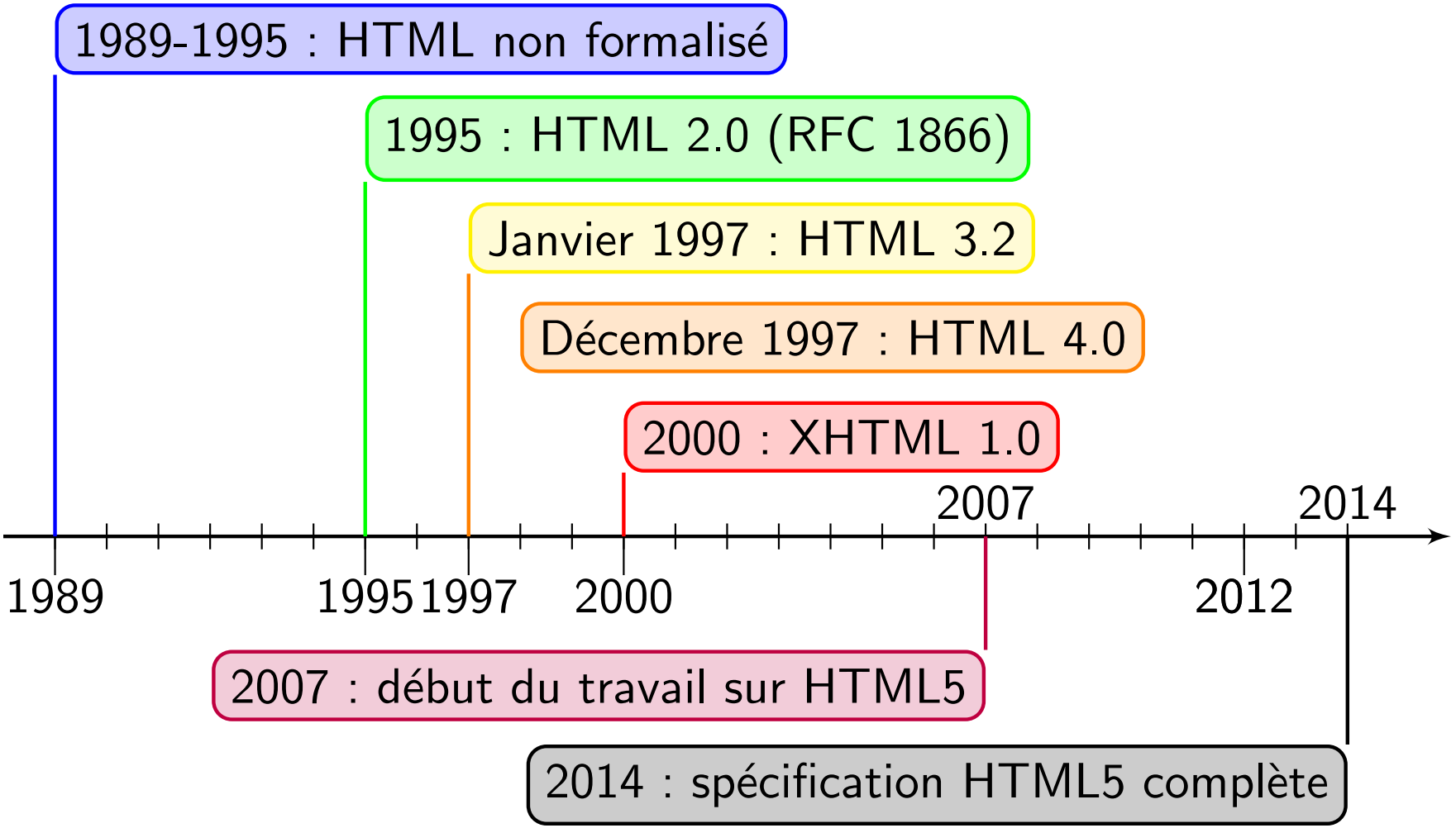
Un nouveau standard

- Car :
  - Standard HTML vieillissant (v4 : 1997) ;
- Pour :
  - Un support multi-média amélioré ;
  - Faciliter l'écriture d'applications web complexes ;
  - Se rapprocher des pratiques de mise en page adoptées.

Les ambitions affichées

- Faire du navigateur une plate-forme de développement ;
- Faciliter la portabilité d'applications ;
- Remplacer les plugins qui faisaient ce que HTML ne gérait pas (flash, etc ...).

Historique HTML



Qui supporte HTML5 ?

- Des organismes de normalisation :
  - W3C : World Wide Web Consortium ;
  - WHATWG : Web Hypertext Application Technology Working Group ;
- Des acteurs connus du monde informatique :
  - Google, Mozilla, Palm, Apple ;
- Des navigateurs internet qui suivent l'évolution d'HTML5 :
  - Google Chrome ;
  - Mozilla Firefox ;
  - Opera ;
  - Apple Safari ;
  - Microsoft Internet Explorer.

Où en est HTML 5 ?

Support par les navigateurs

Navigateur	Score (/500) [a]
Chrome 25	463
Opera 12.10	419
Firefox 19	393
Safari 6.0	378
Internet Explorer 10	320

[a] Source (15/3/2013) : `http://html5test.com`

Définition du standard

17/12/2012 : HTML5 Candidate Recommendation (W3C)

Les différences avec HTML4

- Une classification différente des balises ;
- Un nouveau jeu de balises ;
- Quelques balises supprimées ;
- De nouvelles API Javascript ;

Ce qu'apportent les APIs d'HTML5

- Gestion multi-média (vidéo et son) plus poussée ;
- Introduction de la 3D dans le navigateur ;
- Exploitation des périphériques (*webcam*, *gamepad*) ;
- Gestion des fichiers ;
- Gestion du stockage de données ;
- Applications hors-ligne ;
- Multi-threading et communication.

Structure générale d'un fichier HTML5

La façon HTML5 (transitionnel)

```
<!DOCTYPE html>
<html>
  <head>
    <title>Un document</title>
  </head>
  <body>
    <p>Un paragraphe</p>
  </body>
</html>
```

La façon XML pour HTML5

```
<?xml version="1.0" encoding="
    UTF-8"?>
<html xmlns="http://www.w3.org
    /1999/xhtml">
  <head>
    <title>Un document</title>
  </head>
  <body>
    <p>Un paragraphe</p>
  </body>
</html>
```

Classification des balises (1/6)

HTML4

Types de balises mutuellement exclusifs :

inline pour le contenu en ligne ;

block pour les contenu en bloc.

Classification des balises (2/6)

HTML5

Une balise peut appartenir à plusieurs catégories :

metadata ;

flow ;

sectioning ;

heading ;

phrasing ;

embedded ;

interactive .

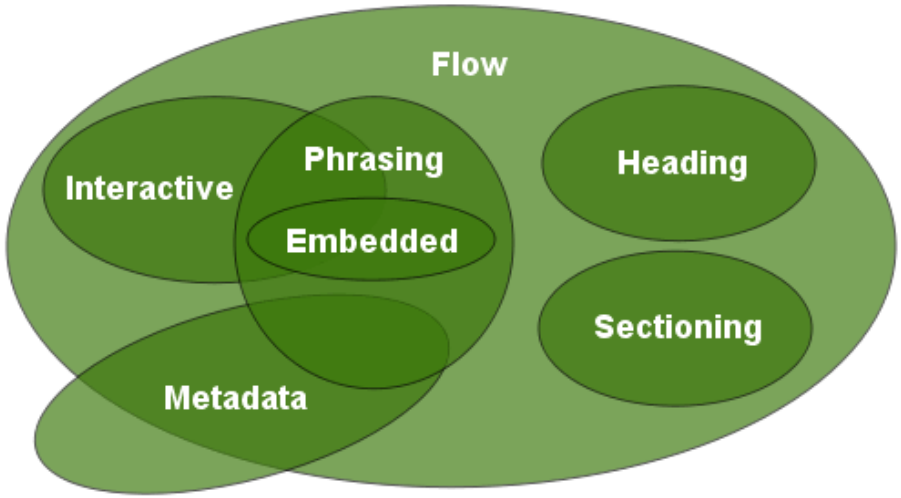


FIGURE: Les intersections entre catégories de balises HTML5

Classification des balises (3/6)

metadata

Pour les informations relatives au document.

Balises : `base command link meta noscript script style title`

sectioning

Définit la portée des en-têtes et bas de page.

Balises : `article aside nav section`

heading

Définit l'en-tête d'une section.

Balises : `h1 h2 h3 h4 h5 h6 hgroup`

Classification des balises (4/6)

flow

Caractérise la plupart des éléments du corps du document.

Balises : `a abbr address area article aside audio b bdi bdo blockquote br button canvas cite code command data datalist del details dfn div dl em embed fieldset figure footer form h1 h2 h3 h4 h5 h6 header hgroup hr i iframe img input ins kbd keygen label link map mark math menu meta meter nav noscript object ol output p pre progress q ruby s samp script section select small span strong style sub sup svg table textarea time u ul var video wbr` et tout texte.

Classification des balises (5/6)

phrasing

Pour les éléments textuels et les liens. Balises : `a abbr area audio b bdi bdo br button canvas cite code command data datalist del dfn em embed i iframe img input ins kbd keygen label link map mark math meta meter noscript object output progress q ruby s samp script select small span strong sub sup svg textarea time u var video wbr` et tout texte.

Classification des balises (6/6)

embedded

Pour les ressources importées dans le document.

Balises : `audio canvas embed iframe img math object svg video`

interactive

Pour le contenu destiné à être modifié par l'utilisateur.

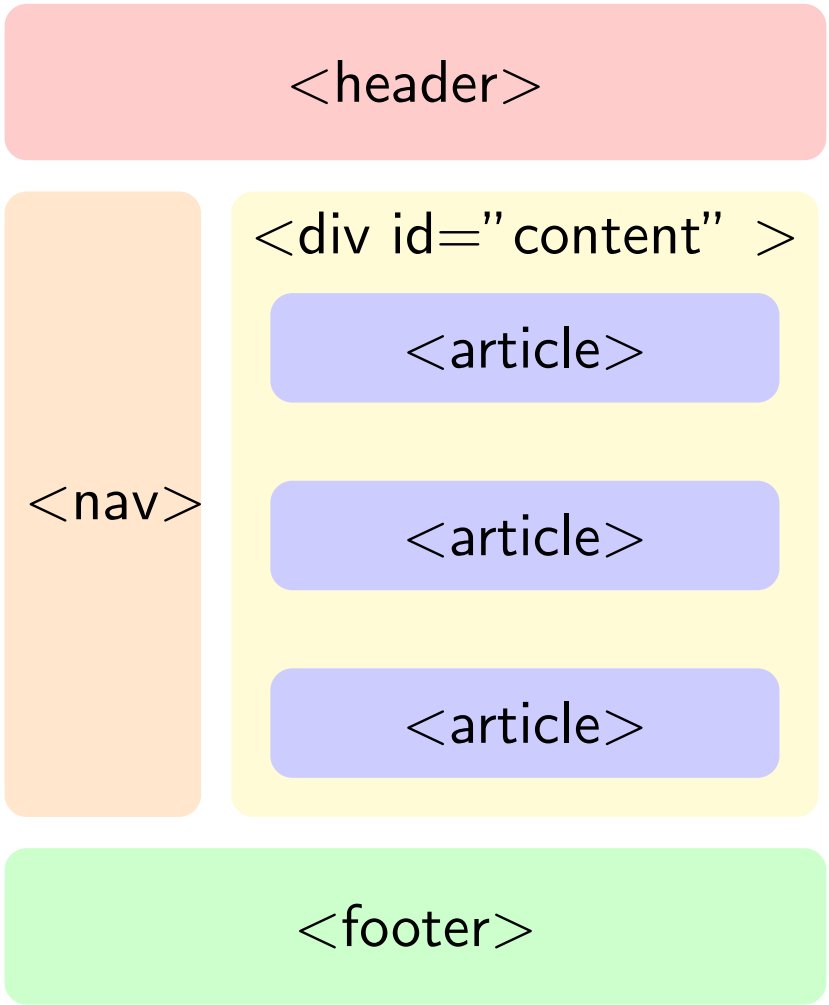
Balises : `a audio button details embed iframe img input keygen label menu object select textarea video`

Les nouvelles balises de mise en page

La mise en page standard

Nouveau jeu de balises inspirées de pratiques courantes :

- `header` ;
- `footer` ;
- `nav` ;
- `section` ;
- `hgroup` ;
- `article` ;
- `aside`.



Les balises de mise en page générale

Les balises de mise en page générale

header en-tête de page : logo, bannière, aides de navigation. Contient souvent un élément d'en-tête de section ;

footer bas de page : appendice, index, notice de *copyright*, informations sur l'auteur ;

article texte indépendant ou une portion de contenu réutilisable : article de blog, journal ;

nav partie du document destinée à la navigation ;

section portion générique de document. Une section est un regroupement thématique de contenu ;

hgroup représente l'en-tête d'une section ;

aside désigne des informations tangentes au contenu : publicité, barre de navigation.

Les balises incorporant des objets

Balises incoporant des objets

figure	partie de document auto-contenue référencée dans le document principal ;
figcaption	associe une légende à une figure ;
math	formule mathématique au format MathML ;
svg	graphique au format SVG (*Scalable Vector Graphics*) ;
video	pour l'inclusion de vidéos ;
audio	pour l'inclusion de fichiers audios ;
source	pour spécifier la source d'un fichier multimedia ;
track	pour une piste de sous-titres ;
embed	pour les plugins (ex : flash) ;
canvas	pour le dessin interactif.

Exemples d'utilisation (1/5)

De l'utilisation de **figure** et **figcaption**

```
<figure>
 <video src="example.webm" controls></video>
 <figcaption>Example</figcaption>
</figure>
```

De l'utilisation de **svg**

```
<svg width="10cm" height="10cm" viewBox="-100 -100 500 500">
  <line x1="0" y1="0" x2="400" y2="0" stroke="blue" />
  <line x1="0" y1="0" x2="0" y2="300" stroke="blue" />
  <line x1="0" y1="300" x2="400" y2="0" stroke="blue" />
  <text x="200" y="-20">a</text>
  <text x="-40" y="150">b</text>
  <text x="200" y="200">c</text>
</svg>
```

Exemples d'utilisation (2/5)

De l'utilisation de **math**

```
<math>
  <mrow>
    <msup><mi>c</mi><mn>2</mn></msup>
    <mo>=</mo>
    <msup><mi>b</mi><mn>2</mn></msup>
    <mo>+</mo>
    <msup><mi>a</mi><mn>2</mn></msup>
  </mrow>
</math>
```

Exemples d'utilisation (3/5)

De l'utilisation de **track**

```
<video src="brave.webm">
 <track kind=subtitles src=brave.en.vtt srclang=en label="
     English">
 <track kind=captions src=brave.en.vtt srclang=en label="
     English for the Hard of Hearing">
 <track kind=subtitles src=brave.fr.vtt srclang=fr label="
     Français">
 <track kind=subtitles src=brave.de.vtt srclang=de label="
     Deutsch">
</video>
```

Exemples d'utilisation (4/5)

De l'utilisation de **video** et **source**

```
<video controls autoplay>
 <source src='video.mp4' type='video/mp4; codecs="avc1.42
     E01E, mp4a.40.2" '>
 <source src='video.ogv' type='video/ogg; codecs="theora, 
     vorbis" '>
 ...
</video>
```

De l'utilisation de **embed** ou de son équivalent **object**

Avec la balise `embed` :

```
<embed src="catgame.swf" quality="high">
```

Avec la balise `object` :

```
<object data="catgame.swf">
 <param name="quality" value="high">
</object>
```

Exemples d'utilisation (5/5)

De l'utilisation de **canvas**

```
<!DOCTYPE html >
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <style type="text/css" media="screen" />
    <script type="text/javascript">
      function init() {
        var canvas = document.getElementById("monCanvas");
        var ctx = canvas.getContext("2d");
        ctx.fillStyle = "rgb(0,0,255)";
        ctx.fillRect(10, 10, 100, 100);
      }
      </script>
  </head>
  <body onload="init();">
    <canvas id="monCanvas" width="100px" height="100px" />
  </body>
</html>
```

Les nouveaux champs de formulaires (1/3)

De nouveaux types d'entrées

- *datalist* utilisé conjointement avec les listes dans les formulaires ;
- *search* pour effectuer des recherches dans des listes ;
- *tel*, *url*, *email* pour spécifier des adresses et des coordonnées ;
- *number*, *range* pour spécifier l'entrée d'un nombre et vérifier l'intervalle de débattement ;
- *datetime*, *date*, *month*,*week*, *time* pour situer un instant dans le temps (pour fournir un calendrier pour choisir la date par exemple) ;
- *color* pour choisir une couleur.

Les nouveaux champs de formulaires (2/3)

De l'utilisation de **datalist**

```
<input list="browsers">
<datalist id="browsers">
 <option value="Safari">
 <option value="Internet Explorer">
 <option value="Opera">
 <option value="Firefox">
</datalist>
```

Les nouveaux champs de formulaires (3/3)

De l'utilisation de **range**

```
<input type="range" min="-100" max="100" value="0" step="10"
      name="power" list="powers">
<datalist id="powers">
 <option value="0">
 <option value="-30">
 <option value="30">
 <option value="+50">
</datalist>
```

Avec un certain style :

```
input { height: 75px; width: 49px; background: #D5CCBB;
    color: black; }
```

Les balises supprimées

Balises supprimées dans HTML5

- Les balises à effets mieux gérées par CSS :
  - `basefont big center font strike tt`
- Les balises dommageables pour la navigation :
  - `frame frameset noframes`
- Les balises qui sont confuses ou gérées également par d'autres :
  - `acronym` car remplacée par `abbr` ;
  - `applet` car remplacée par `object` ;
  - `dir` car remplacée par `ul`
  - ...

Les nouvelles APIs

- API multimedia pour jouer videos et fichiers sons avec les nouvelles balises *audio* et *video* ;
- API pour les applications Web hors-ligne ;
- API qui permet aux applications Web application de s'associer à des protocoles ou des types de media ;
- API d'édition en combinaison avec l'attribut *contenteditable* ;
- API Drag & drop en combinaison avec l'attibut *draggable* ;
- API qui gère l'historique et permet aux pages de contrôler le bouton *"Précédent"*.

L'API classList (1/2)

L'API classList

- Permet de gérer les classes de styles associées aux éléments du document ;
- Un élément peut être associé à plusieurs classes ;
- Un objet particulier est associé à chaque élément : l'objet classList.

L'API classList (2/2)

Contenu de l'objet `classList`

- length :
  - Donne le nombre de classes associées à l'élément ;
- item(i) :
  - Donne le nom de la $i^{ème}$ classe associée à l'élément ;
- contains(className) :
  - Booléen vrai si className est bien dans la liste de classes ;
- add(className) :
  - Ajoute className à la liste de classes ;
- remove(className) :
  - Enlève className à la liste de classes ;
- toggle(className) :
  - Ajoute ou enlève className à la liste.

Example d'utilisation de classList

A mettre dans le corps de body :

```
<style>
  #blink  { text-align: center; }
  .bold { font-weight: bold }
</style>
<script>
  function toggle() {
    var p = document.getElementById("blink");
    p.classList.toggle("bold");
  }
  setInterval(toggle, 500); //timer activé toutes les 1/2 s.
</script>
<p id="blink">Ceci est un texte clignotant !</p>
```

L'API de géolocalisation (1/2)

L'API de géolocalisation

- Situe l'utilisateur du navigateur à l'échelle mondiale ;
- API associée à HTML5 mais qui n'en fait pas partie ;
- Introduit de nouveaux objets Javascript.

L'API de géolocalisation (2/2)

Les objets associés à la géolocalisation

- Geolocation :
  - objet principal pour manipuler l'interface de géolocalisation ;
- PositionOptions :
  - options à passer à l'objet de géolocalisation ;
- Position :
  - positions retournées lors d'un requête de géolocalisation ;
- Coordinates :
  - coordonnées géographiques associées à la position ;
- PositionError :
  - nature de l'erreur si cette dernière est levée.

L'objet `Geolocation` (1/3)

L'objet `Geolocation`

Si supporté par le navigateur, alors existe sous la forme d'une instance nommée `navigator.geolocation`.

Méthodes de l'objet `Geolocation`

- getCurrentPosition(successCallback,[errorCallback],[options])
  - Donne la position courante ;
- watchPosition(successCallback,[errorCallback],[options])
  - Attache une fonction d'observation sur la position courante ;
  - Retourne un identifiant d'observation (watchId) ;
- clearWatch(watchId)
  - Détache une fonction d'observation.

L'objet `Geolocation` (2/3)

Paramètres des méthodes

successCallback fonction de rappel utilisée en cas de succès ;

errorCallback fonction de rappel utilisée en cas d'erreur ;

options options de types PositionOptions ;

watchId identifiant retourné par watchPosition().

Fonctions de rappel en cas de succès

```
function maFonction(position /* type: Position */ ) { }
```

Fonctions de rappel en cas d'erreur

```
function maFonction(error /* type: PositionError */) { }
```

L'objet Geolocation (3/3)

Squelettes d'utilisation

```
function displayPosition(position) {
    ...
}

function displayPositionError(positionError) {
    ...
}

if (navigator.geolocation) {
    // Test si la geolocalisation est disponible
    navigator.geolocation.getCurrentPosition(displayPosition
        ,displayPositionError);
    // Mise en place des appels aux fonctions de rappel.
}
```

L'objet `PositionOptions` (1/2)

L'objet `PositionOptions`

Permet de spécifier les options de localisation.

Attributs de l'objet `PositionOptions`

- enableHighAccuracy : booléen qui active la demande du meilleur résultat possible de localisation ;
- timeout : temps maximum en millisecondes pendant lequel on attend un résultat ;
- maximumAge : temps maximum pendant lequel une position en cache est acceptée.

L'objet PositionOptions (2/2)

Example d'utilisation

```
navigator.geolocation.getCurrentPosition(displayPosition,
    displayPositionError,{enableHighAccuracy:true, timeout
    :1000});
// active la haute précision de localisation et attend au
    maximum 1 seconde.
```

L'objet `Position`

L'objet `Position`

Contient les informations de position.

Les attributs de l'objet `Position`

- coords : coordonnées de type Coordinates ;
- timestamp : timestamp donné en millisecondes depuis l'Epoch Unix (1/1/1970).

L'objet Coordinates (1/3)

L'objet Coordinates

Position en coordonnées géodésique WGS84 (celle du GPS).

L'objet `Coordinates` (2/3)

Les attributs de l'objet `Coordinates`

- latitude ;
- longitude ;
- altitude : optionnel [a] ;
- accuracy : précision ;
- alitudeAccuracy : optionnel ;
- heading : cap en degrés par rapport au nord géographique dans le sens anti-trigonométrique (optionnel) ;
- speed : vitesse (optionnel).

[a] Les objets optionnels valent `null` si non présents.

L'objet Coordinates (3/3)

Utilisation des positions et des coordonnées

```
function displayPosition(position) {
    var p = document.getElementById("infos");
    if (positions.coords.latitude == null) {
        p.innerHTML = "Position non définie" ;
    } else {
        p.innerHTML = "Latitude " + position.coords.latitude
            + ", longitude " + position.coords.longitude ;
    }
}
```

L'objet PositionError (1/2)

L'objet PositionError

Permet de comprendre pourquoi la géolocalisation n'a pas fonctionné.

Attributs de l'objet PositionError

- code : code de l'erreur. Peut prendre trois valeurs :
  - PERMISSION_DENIED : la géolocalisation n'est pas autorisée ;
  - POSITION_UNAVAILABLE : la position n'est pas disponible ;
  - TIMEOUT : la géolocalisation ne s'est pas effectuée dans le temps imparti.
- message : message précisant la nature du problème.

L'objet PositionError (2/2)

Utilisation de PositionError

```
function displayPositionError(positionError) {
    var p = document.getElementById("infos");
    switch(positionError.code) {
        case error.TIMEOUT:
            p.innerHTML = "Position non acquise dans le
                temps imparti";
            break ;
        case error.PERMISSION_DENIED:
            p.innerHTML = "Vous devez autoriser l'
                utilisation de la fonction de
                géolocalisation de votre navigateur." ;
    break ;
    };
}
```

Les données personnalisées (1/2)

Les données personnalisées

- Permettent d'associer des données particulières à n'importe quel élément ;
- Se matérialisent par les balises data-* ;
- Peuvent être modifiées :
  - En manipulant l'attribut directement ;
  - En manipulant l'objet dataset associé à l'élément.

Associer des données à un élément

```
<div id="ship1" class="spaceship" data-ship-id="92432"
     data-weapons="laser 2" data-shields="50%"
     data-x="30" data-y="10" data-z="90">
... </div>
```

Les données personnalisées (2/2)

Manipuler les données

```
var ship = document.getElementById("ship1");
var x = ship.dataset.x ; // via l'objet dataset
var x = ship.getAttribute("data-x"); // via getAttribute
```

Spécifier le style par les données associées

```
div[data-x="30"] {
  ...
}
```

L'API de Drag'n drop

L'API de Drag'n drop

- Permet de gérer le glisser/déplacer entre :
  - Des éléments du document ;
  - Des fichiers et le navigateur ;
- Introduit un certain nombre d'éléments :
  - Des nouveaux attributs de balises ;
  - Des nouveaux évènements JavaScript ;
  - Des nouveaux objets JavaScript.

Les attributs de balise pour le Drag'n Drop (1/2)

L'attribut `draggable`

- Possédé par tous les éléments HTML ;
- Permet de spécifier ce qui peut être déplacé dans une page ;
- Prend trois valeurs : `true`, `false` ou `auto` :
  - `true` : l'élément peut-être glissé ;
  - `false` : l'élément ne peut pas être glissé ;
  - `auto` : en fonction des capacités du navigateur.

Les attributs de balise pour le Drag'n Drop (2/2)

L'attribut `dropzone`

- Possédé par tous les éléments HTML ;
- Spécifie les zones dans lesquels on peut glisser du contenu ;
- Spécifie le contenu accepté par un assemblage de mots-clés :
  - copy : déposer un élément va le copier ;
  - move : déposer un élément va le déplacer ;
  - link : déposer un élément crééera un lien vers ce dernier ;
- Suivi d'un ou plusieurs types de données préfixés par :
  - string : pour les éléments textuel ;
  - file : pour les fichiers ;
- ... et suivies d'un type MIME.

Les évènements Drag'n Drop

Liste des évènements

- dragStart() : déclenché à l'initialisation du D'nD ;
- drag() : déclenché pendant la poursuite du D'nD ;
- dragEnter() : déclenché lors d'un glissement sur une cible potentielle ;
- dragLeave() : déclenché lors d'un glissement hors d'une cible potentielle ;
- dragOver() : identique par bien des aspects à dragEnter() ;
- dragEnd() : déclenché lorsque le glissement est interrompu ;
- drop() : déclenché lors d'un dépôt sur une cible potentielle.

Les objets introduits par l'API de Drag'n Drop (1/3)

L'objet `DragEvent`

- Vient en paramètre des nouvelles fonctions d'évènements ;
- Contient les attributs classique de MouseEvent ;
- Contient un attribut supplémentaire : `dataTransfer` de type DataTransfer.

Les objets introduits par l'API de Drag'n Drop (2/3)

Les attributs de l'objet `DataTransfer`

- dropEffect : type d'opération en cours parmi `none`, `copy`, `link`, `move` ;
- effectAllowed : type d'opérations autorisées parmi `none`, `copy`, `copyLink`, `copyMove`, `link`, `linkMove`, `move`, `all`, `uninitialized` ;
- items : pour retourner plusieurs valeurs transférées ;
- types : formats de données véhiculés ;
- files : liste de fichiers glissés, si présents ;

Les objets introduits par l'API de Drag'n Drop (3/3)

Les méthodes de l'objet `DataTransfer`

- setDragImage(element,x, y) utilise `element` pour mettre à jour l'opération de glissement ;
- addElement(element) ajoute l'élément donné à la liste des éléments utilisés pour matérialiser l'opération de D'nD ;
- setData(format,data) ajoute les données spécifiées pour le transfert ;
- clearData([format]) retire les données au format spécifié ou tout.

Mettre en place un Drag'n Drop ... (en théorie) (1/4)

Préparer les éléments qui seront déplaçables

```
<p>What fruits do you like?</p>
<ol ondragstart="dragStartHandler(event)">
 <li draggable="true" data-value="fruit-apple">Apples</li>
 <li draggable="true" data-value="fruit-orange">Oranges</li>
 <li draggable="true" data-value="fruit-pear">Pears</li>
</ol>
<script>
   // préparation des données à transférer en cas de D'nD
</script>
```

Mettre en place un Drag'n Drop ... (en théorie) (2/4)

Préparer les données à transférer

```
var internalDNDType = 'text/x-example'; // pseudo type MIME
function dragStartHandler(event) {
  if (event.target instanceof HTMLLIElement) {
    // utilisation de dataset pour récupérer les données à
        déplacer
    event.dataTransfer.setData(internalDNDType, event.target
        .dataset.value);
    event.dataTransfer.effectAllowed = 'move'; // autoriser
        seulement les déplacements.
  } else {
    event.preventDefault(); // sinon ne pas autoriser le
        déplacement.
  }
}
```

Mettre en place un Drag'n Drop ... (en théorie) (3/4)

Préparer la zone d'acceuil

```
<p>Drop your favorite fruits below:</p>
<ol dropzone="move string:text/x-example" ondrop="
    dropHandler(event)"> </ol>
```

Mettre en place un Drag'n Drop ... (en théorie) (4/4)

Accepter les données transférées

```
var internalDNDType = 'text/x-example'; // set this to
    something specific to your site
function dropHandler(event) {
  var li = document.createElement('li');
  var data = event.dataTransfer.getData(internalDNDType);
  if (data == 'fruit-apple') {
    li.textContent = 'Apples';
  } else if (data == 'fruit-orange') {
    li.textContent = 'Oranges';
  } else if (data == 'fruit-pear') {
    li.textContent = 'Pears';
  } else {
    li.textContent = 'Unknown Fruit';
  }
  event.target.appendChild(li);
}
```

Mettre en place un Drag'n Drop ... (en pratique) (1/5)

A l'heure actuelle

- Attribut dropzone non supporté par touts les navigateurs (+Chrome,Safari –Firefox,IE) ;
- Type MIME non supporté par tous les navigateurs ;
  - Utiliser Text à la place ;
- 3 évènements à implémenter quoiqu'il arrive :
  - source : ondragstart ;
  - cible : ondragover, drop.

Mettre en place un Drag'n Drop ... (en pratique) (2/5)

Préparer les éléments qui seront déplacés

```
<p>What fruits do you like?</p>
  <ol ondragstart="return dragStartHandler(event)">
    <li draggable="true" data-value="Apples" >Apples</li>
    <li draggable="true" data-value="Oranges" >Oranges</li>
    <li draggable="true" data-value="Pears" >Pears</li>
  </ol>
```

Mettre en place un Drag'n Drop ... (en pratique) (3/5)

Préparer les données à transférer

```
function dragStartHandler(event) {
  if (event.target instanceof HTMLLIElement) {
    // utilisation de "Text" en tant que type MIME
    event.dataTransfer.setData("Text", event.target.dataset.
        value);
    event.dataTransfer.effectAllowed = 'move'; // only allow
         moves
    } else {
      event.preventDefault();
    }
    return true;
}
```

Mettre en place un Drag'n Drop ... (en pratique) (4/5)

Préparer la zone d'acceuil

```
<p>Drop your favorite fruits below:</p>
  <ol id="dropzone"  ondrop="return dropHandler(event)"
      ondragover="return dragOverHandler(event)" >
  </ol>
```

Mettre en place un Drag'n Drop ... (en pratique) (5/5)

Implémenter les évènements

```
function dropHandler(event) {
  var li = document.createElement('li');
  var data = event.dataTransfer.getData("Text");
  li.textContent=data;
  event.target.appendChild(li);
  return false;
}

function dragOverHandler(event) {
  return false;
}
```

Les applications hors-ligne

Les applications hors-ligne

- Permet aux utilisateurs d'interagir avec les applications Web y compris hors-ligne ;
- Est mis en place par l'adjonction d'un manifeste donnant la liste des fichiers nécessaires pour travailler hors-ligne ;
- Fonctionne avec un cache d'application.

Le manifeste (1/3)

Le manifeste

- Donne la liste des fichiers nécessaires pour travailler hors-ligne ;
- Suit une syntaxe précise :
  - Commence par la ligne `CACHE MANIFEST` ;
  - Comprend trois sections ;
  - La section `CACHE:` indique les fichiers à mettre en cache. Les premiers fichiers listés n'appartenant pas à une section sont considérés comme étant dans `CACHE:`
  - La section `NETWORK:` indique les fichiers qui sont à mettre à jour impérativement par le réseau ;
  - La section `FALLBACK:` indique les substitutions à opérer.

Le manifeste (2/3)

Exemple de manifeste

```
CACHE MANIFEST
NETWORK:
comm.cgi
CACHE:
style/default.css
images/sound-icon.png
images/background.png
FALLBACK:
/ /offline.html
```

Le manifeste (3/3)

Associer un manifeste à un fichier html

Au moyen de l'attribut manifest de la balise html.

Exemple :

```
<!DOCTYPE html>
<html manifest="monapp.manifest">
...
</html>
```

Manipuler le cache d'applications (1/2)

L'objet `ApplicationCache`

- Instance récupérable sous la forme window.applicationCache ;
- Contient un attribut : status qui peut valoir :
  - UNCACHED : l'application n'est pas en cache ;
  - IDLE : l'application est en cache et à jour ;
  - CHECKING : le cache est en cours de vérification ;
  - DOWNLOADING : le cache est en cours de mise à jour ;
  - UPDATEREADY : le cache n'est pas à jour ;
  - OBSOLETE : le cache n'est plus valide.
- Contient trois méthodes :
  - update() : met à jour le cache ;
  - abort() : met fin au processus de mise à jour ;
  - swapCache() : bascule vers une version plus récente du cache.

Manipuler le cache d'applications (2/2)

Les évènements associés au cache

- Correspondent à chaque statuts du cache ;
- checking, error, noupdate, downloading, progress, updateready, cached et obsolete ;
- Les fonctions de rappel prennent toutes en paramètre un objet de type Event à l'exception de progress qui attend un objet de type progressEvent.

L'objet `ProgressEvent`

- Dérive de Event et rajoute 3 attributs :
  - lengthComputable : à vrai si la longueur totale est connue ;
  - loaded : taille chargée ;
  - total : taille totale à charger.

Exemple de manipulation du cache (1/3)

Page HTML principale

```
<!DOCTYPE html>
<html manifest="cacheTest.manifest">
  <head><title>Test de cache</title></head>
  <body>
    <style>
      body {
        background-image: url('wallpaper.jpg');
      }
    </style>
    <script type="text/javascript"> ... </script>
    <h1>Test du cache</h1>
    <p>Statut du cache : <span id="infos">...</span></p>
    <p><input type="button" id="update" value="Update" /></p>
  </body>
</html>
```

Exemple de manipulation du cache (2/3)

Le fichier manifeste associé

```
CACHE MANIFEST
#Version 2.0

CACHE:
cache.html
wallpaper.jpg
```

Exemple de manipulation du cache (3/3)

Script associé

```
var statusList = [ "UNCACHED", "IDLE", ... ];
var events = ["checking", "error", ... ];
var cache = window.applicationCache ;

function updateStatus(event) {
  document.getElementById('infos').textContent = "ev : " +
      event.type + ", st : " + statusList[cache.status] ;
  if (event.type == 'updateready')
    cache.swapCache();
}
var i = events.length ;
while (i--) {
  cache.addEventListener(events[i], updateStatus,false );
}
document.getElementById('update').addEventListener('click',
    function(event) { cache.update(); }, false);
```

Les problèmes de mise en oeuvre du cache

Cache et priorité

- Les fichiers en cache sont prioritaires !
- La mise à jour se réalise :
  - En vidant manuellement le cache ;
  - En modifiant le fichier manifeste ;
  - En mettant à jour le cache avec JavaScript .

Mise en ligne de l'application

- Le cache n'est testable que par la mise en place d'un serveur ;
- Ne pas oublier d'ajouter le type MIME suivant :
  - **text/manifest manifest**

La détection de la mise hors ligne du navigateur (1/2)

La navigation hors-ligne

- Associée à l'attribut booléen window.navigator.onLine ;
- Utilise les évènements online et offline.

La détection de la mise hors ligne du navigateur (2/2)

Détection du statut de la navigation

```
<!DOCTYPE HTML>
<html>
 <head>
  <title>Online status</title>
  <script>
   function updateIndicator() {
     document.getElementById('indicator').textContent =
         navigator.onLine ? 'online' : 'offline';
   }
  </script>
 </head>
 <body onload="updateIndicator()" ononline="updateIndicator
     ()" onoffline="updateIndicator()">
  <p>The network is: <span id="indicator">(state unknown)</
      span>
 </body>
</html>
```

L'API webstorage

Du stockage de données côté client

- Mécanisme similaire aux cookies ;
- Les transactions multiples en plus !
- L'optimisation de bande-passante en plus !
- Une classe d'objets de type Storage ;
- Deux webstorages : sessionStorage et localStorage ;
- Des évènements pour actualiser les informations (storage).

La classe Storage

La classe Storage

- Permet de stocker des paires de type clé/valeur ;
- Un attribut :
  - length : le nombre de paires stockées.
- 5 méthodes :
  - key(i) : retourne la $i^{\text{ème}}$ clé ;
  - getItem(key) : retourne la valeur associée à la clé `key` ;
  - setItem(key,value) : stocke une paire clé/valeur ;
  - removeItem(key) : retire la paire associée à la clé `key` ;
  - clear() : vide l'objet de toutes ses paires clé/valeur.

Les objets `sessionStorage` et `localStorage`

`sessionStorage`

- Représente le plus haut niveau de stockage de contenu Web ;
- Réinitialisé à chaque nouvelle création du document ;
- Accessible pour chaque page du site ouverte dans la même fenêtre.

`localStorage`

- Stockage persistent lors de la destruction du document ;
- Fournit un contexte de stockage par site ;
- Accessible pour chaque page du même site.

L'évènement `storage`

L'évènement `storage`

- Fait partie de la classe Window ;
- Est traité par une fonction de rappel prenant en paramètre un objet de class StorageEvent.

La classe `StorageEvent`

- Dérive de Event ;
- Comprend 5 attributs :
  - key : la clé sur laquelle survient l'évènement ;
  - oldValue : l'ancienne valeur associée à la clé ;
  - newValue : la nouvelle valeur associée à la clé ;
  - url : l'adresse du document dont la clé a été modifiée ;
  - storageArea : référence à la zone de stockage utilisée.

Exemple d'utilisation de l'API webstorage (1/2)

Exemple d'utilisation

```
<p>
  You have viewed this page
  <span id="count">an untold number of</span>
  time(s).
</p>
<script>
  if (!localStorage.pageLoadCount)
    localStorage.pageLoadCount = 0;
  localStorage.pageLoadCount = parseInt(localStorage.
      pageLoadCount) + 1;
  document.getElementById('count').textContent =
      localStorage.pageLoadCount;
</script>
```

Exemple d'utilisation de l'API webstorage (2/2)

Mise en place des évènements

```
function handleStorage(event) {
  if (event.key == 'pageLoadCount') {
    document.getElementById('count').textContent = event.
        newValue;
  }
}
window.addEventListener("storage", handleStorage, false);
```

La communication

La communication

- L'idée générale est de permettre aux documents de communiquer entre eux ;
- Partagent un type d'objet : MessageEvent :
- 4 situations différentes l'exploitent :
  - Les évènements serveurs : pour faire du « *push* » ;
  - *Les websockets pour une communication bilatérale ;*
  - *Les messages inter-documents ;*
  - *Les canaux de communication.*

Communication et sécurité

Les nouvelles possibilités de communication de HTML5 ouvrent aussi des potentiels problèmes de sécurité !

L'objet MessageEvent

L'objet MessageEvent

- Dérive de Event ;
- Possède 5 attributs :
  - data : les données du message ;
  - origin : l'origine du message ;
  - lastEventId : réservé au push
  - source : la fenêtre à la source du message ;
  - ports : les ports exprimés sous la forme d'un tableau d'objets de type MessagePort.

La communication inter-documents (1/2)

Poster un message

- Se fait entre deux documents dont l'un est chargé dans une balise iframe ;
- Utilisation de la méthode window.postMessage(data,origine) ;
- data représente les données à envoyer ;
- origine est une chaîne de caractère permettant d'identifier la provenance du message ;

Recevoir un message

- Se fait par l'intermédiaire de l'évènement message ...
- Et une fonction de rappel prenant en paramètre un objet de type MessageEvent

La communication inter-documents (2/2)

Côté document qui poste le message

```
var o = document.getElementsByTagName('iframe')[0];
o.contentWindow.postMessage('Hello world', 'http://b.example
    .org/');
```

Côté réception

```
window.addEventListener('message', receiver, false);
function receiver(e) {
  if (e.origin == 'http://example.com') {
    if (e.data == 'Hello world') {
      e.source.postMessage('Hello', e.origin);
    } else {
      alert(e.data); } }
}
```

Les canaux de communication (1/2)

Contexte d'utilisation

Le multi-tâche !

L'objet `Message Channel`

- Possède deux attributs :
  - port1 : de type MessagePort ;
  - port2 : de type MessagePort ;
- Les données envoyées par un port sont reçues par l'autre.

Les canaux de communication (2/2)

L'objet `MessagePort`

- Réagit à un évènement de type message ;
- Possède trois méthodes :
  - postMessage(data) : pour les données à envoyer ;
  - start() : pour indiquer que le port est actif et peut distribuer les messages ;
  - close() : pour fermer le port.

Les websockets (1/5)

L'objet `WebSocket`

- Dérive de EventTarget
- Possède 5 attributs en lecture seule :
  - url : l'url à laquelle s'est connectée la websocket ;
  - readyState : renvoie le statut de la websocket :
    - 4 valeurs : CONNECTING, OPEN, CLOSING, CLOSED ;
  - bufferedAmount : donne le nombre d'octets présents dans le tampon et qui n'ont pas encore été transférés au travers du réseau
  - protocol : retourne le protocole utilisé lors de la connection (chaîne vide sinon) ;

Les websockets (2/5)

Les méthodes de `websocket`

- new WebSocket(url,protocoles) : création de la websocket :
  - url : url de la websocket distante ;
  - protocoles : liste de protocoles alternatifs pouvant être utilisés ;
- close() : fermeture de la websocket ;
- send(data) : envoi de données
  - data : données à envoyer (n'importe quel objet JavaScript) ;

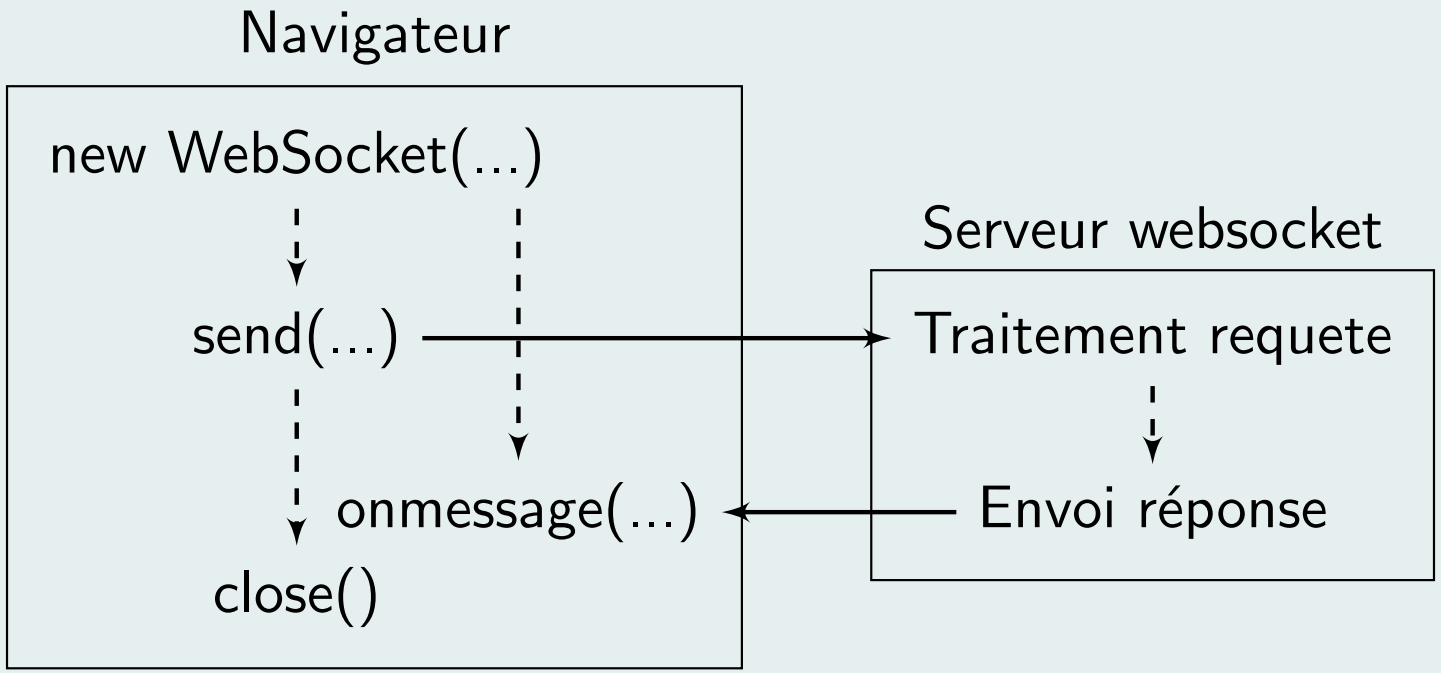
Les websockets (3/5)

Les évènements de `websocket`

- onopen : déclenché à l'ouverture de la websocket ;
- onerror : déclenché lorsqu'une erreur survient sur la websocket ;
- onclose : déclenché lors de la fermeture de la websocket ;
- onmessage : déclenché lorsque le serveur a répondu et envoyé des données à la websocket.

Les websockets (4/5)

Schéma d'exécution



Les websockets (5/5)

Exemple d'utilisation de websocket

```
var socket = new WebSocket('ws://game.example.com:12010/
    updates');
socket.onopen = function() {
  setInterval(function() {
    if (socket.bufferedAmount == 0)
      socket.send(getUpdateData());
    }, 50);
};
```

WebSockets et sécurité

- Un protocole en cours de fiabilisation ;
- Un comportement délicat à ajuster lorsqu'il y a des proxys ;
- Eviter de mixer protocole `https` et websockets non sécurisés.

Bases de données et HTML5

Types de bases de données

WebSQL : abandonné car implémenté seulement avec Sqlite :

- Repose sur l'utilisation de SQL comme les bases de données classiques.

IndexedDB : en cours, spécification terminée en mai 2012 :

- Utilise des objets au format JSON ;
- Ne nécessite pas le langage SQL.

IndexedDB

Deux APIs différentes

- Une API synchrone (à utiliser avec les WebWorkers – les processus côté navigateur et hors de ce cours) ;
- Une API asynchrone basé sur le système des évènements.

Politiques de sécurité

L'utilisation en local de la base de donnée n'est pas soumise à la même politique de sécurité suivant le navigateur.

Objets concernés

Objets en lien avec l'API IndexedDB

IDBFactory : objet de base permettant d'obtenir la base de données ;

IDBDatabase : base de données ;

IDBObjectStore : équivalent d'une table ;

IDBTransaction : transaction ;

IDBRequest : requête ;

IDBCursor : curseur pour se déplacer dans une liste d'objets renvoyés par une requête.

L'objet `IDBFactory` (1/2)

Localisation

- Déjà présent dans le navigateur ;
- Parfois préfixé ;
- Existe donc sous différents noms :
  - `window.indexedDB` ;
  - `window.mozIndexedDB` ;
  - `window.webkitIndexedDB` ;
  - `window.msIndexedDB`.

L'objet IDBFactory (2/2)

Méthodes associées

Retours de type IBDRequest :

- open : ouverture de la base en donnant un nom et un numéro de version (optionnel) ;
- deleteDatabase : destruction de la base en donnant son nom.

L'objet IBDRequest (1/1)

Membres associés

- result : Résultat de la requête ;
- transaction : transaction liée à la requête ;

Evènements associés

- onsuccess : requête effectuée avec succès ;
- onerror : requête terminée mais infrutueuse.
- onblocked : lorsque l'ouverture est empêchée par l'utilisateur ;
- onupgradeneeded : lorsque le numéro de version demandé ne concorde pas avec celui de la base à l'ouverture.

Exemple d'ouverture de base de données

Ouverture de base de données

```
var db ;
db.indexedDB = window.indexedDB || window.webkitIndexedDB ||
  window.mozIndexedDB || window.msIndexedDB ;
if (db.indexedDB == null) {
  alert("IndexedDB not found.");
  return ;
}
var request = db.indexedDB.open("pointage",2);
```

Création de la base de données

Procédure non encore totalement standardisée

Deux manières de faire :

- Utilisation de l'évènement onupgradeneeded ;
- Par comparaison de version.

Création de la base de données (1/1)

Mise en place des fonctions de rappel

```
request.onsuccess = function(e) {
    var version = 2;
    var database = e.target.result;
    // façon 1
    if (version != database.version) {
        var setVersion = database.setVersion(version);
        setVersion.onsuccess = function (e) {
          createDatabase(database);
        };
    }
};
// façon 2
request.onupgradeneeded = function (e) {
    createDatabase(e.target.result);
};
```

L'objet IDBDatabase

Lien avec IBDRequest

Champ target.result des évènements envoyés par les IBDRequest.

Attributs

name nom de la base de données ;

version version associée (entier) ;

objectStoreNames tableau des noms de tables ;

Méthodes

- createObjectStore(nom,[params]) : création d'une table ;
- deleteObjectStore(nom) : destruction d'une table ;
- transaction(nim, mode) : initialisation d'une transaction ;
- close() : fermeture de la base de données.

L'objet IBDObjectStore

Attributs

- name : nom de la table ;
- keyPath : nom de la clé ;
- transaction : retourne la transaction liée à la table ;
- autoIncrement : indique si la clé doit s'autoincrémenter.

Méthodes

- put(valeur,[cle]) : stocke un objet dans la table ;
- add(valeur,[cle]) : idem ;
- delete(cle) : effacer un objet par sa clé ;
- get(cle) : récupérer un objet par sa clé ;
- openCursor(...) : parcours la table ;
- clear() : vider la table de ses objets.

Initialisation d'une table

Création d'une table

```
const students = [
  { id: "jydidier", nom: "Didier", prenom: "Jean-Yves",
      niveau: "Enseignant", filiere: "MCF" },
  { id: "20021091", nom: "Clinton" , prenom: "Bill", niveau:
      "President", filiere : "USA" },
  { id: "20020007", nom: "Bond", prenom: "James", niveau: "
     M1", filiere: "GEII"}
];

function createDatabase(database) {
  database.createObjectStore("evenements",{keyPath:"
      timeStamp"});
  var store = database.createObjectStore("etudiants", {
      keyPath:"id"});
  for (var i in students) {
    store.add(students[i]);
  };
};
```

L'objet `IBDCursor`

Permet de parcourir une table.

Attributs

- source : objet sur lequel opère le curseur ;
- key : clé courante sur laquelle est arrivé le curseur.

Opérations

- continue() : fait avancer le curseur d'un cran si possible.

L'objet `IBDTransaction` (1/2)

Permet d'accéder à une base de données de manière asynchrone.

Attributs

- mode : mode d'ouverture de la base (`readonly`, `readwrite`) ;
- db : base de données associée ;
- error : message d'erreur (si cela se produit).

Opérations

- abort : annulation de la transaction (annulation de tous les changements de la transaction en cours ;
- objectStore(nom) : retourne la table au nom approprié.

L'objet `IBDTransaction` (2/2)

Evènements

- onabort : déclenché en cas d'annulation ;
- oncomplete : déclenché lorsque la transaction est achevée ;
- onerror : déclenché en cas d'erreur.

Exemple de manipulation d'une transaction

Mise en place d'une transaction

```
var transaction = database.transaction(["etudiants"], "
    readonly");
var store = transaction.objectStore("etudiants");
var cursorRequest = store.openCursor();

cursorRequest.onsuccess = function(e) {
  var result = e.target.result ;
  if (!! result == false)
    return ;
  // effectuer traitement sur result.value
  // objet formatté comme à l'initialisation
  result.continue();
}

cursorRequest.onerror = function(e) {
  // traitement en cas d'erreur
}
```