

# *HTML5 en pratique*

## *Gestion des données*



- Introduction
- Les balises
- CSS 3
- Javascript, le langage du web
- Vers des application plus interactives
- Gestion des données
- Multimédia
- Conclusion

- Session Store et Local Store
- Bases de données web
- Applications déconnectées avec Application Cache

- HTML 5 offre deux façons de stocker du contenu côté client
  - Le "*session store*", temporaire, qui est propre à chaque session de navigation (fenêtre ou onglet du navigateur)

```
var storage = window.sessionStorage;
```

- Le "*local store*", permanent, qui est propre à un domaine

```
var storage = window.localStorage;
```



- Les deux Stores proposent la même API
  - `setItem(key, value)`
  - `getItem(key)`
  - `removeItem(key)`
  - `clear()`
  - `key(index)` // Récupère la clé n°index
  - `length`
- Les Stores étant des tableaux, on peut également utiliser la notation avec des crochets

```
window.sessionStorage[<key>] = <value>;
```

- Compatible Chrome 4+, Safari 4+, Opera 10.5+, Firefox 3.5+, IE 8+

# Local store et Session store

## Exemple

```
<h2>My web library</h2>
<form onsubmit="return saveBook();">
  <label for="bookName">Book :</label>
  <input type="text" id="bookName"/>
  <input type="submit" value="Add"/>
</form>

<ul id="books"></ul>
```

### My web library

Book :

- Dune
- Fondation
- Chroniques de Durdane

# Local store et Session store

## Exemple

```
var storage = window.localStorage; // ou window.sessionStorage

window.onload = function() {
    for(var i=0; i<storage.length; i++) {
        displayBook(storage.getItem(i));
    }
}

function displayBook(book){
    var contenu = document.createTextNode(book);
    var puce = document.createElement("li");
    puce.appendChild(contenu);
    document.getElementById("books").appendChild(puce);
}

function saveBook() {
    var book = document.getElementById("bookName").value;
    storage.setItem(storage.length, book);
    displayBook(book);
}
```

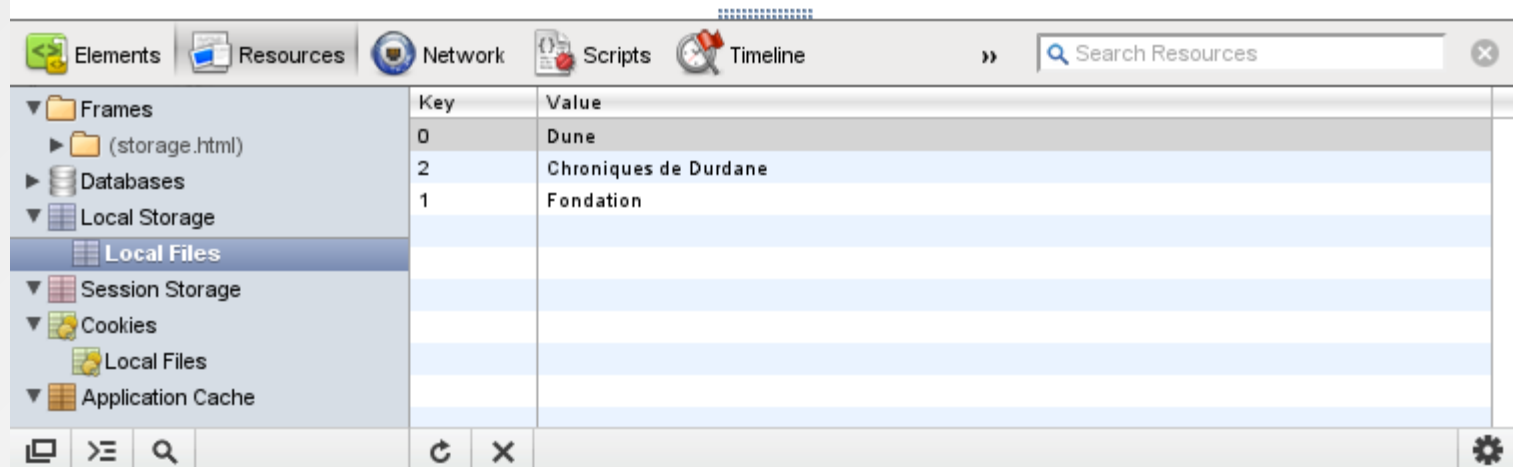
# Local store et Session store

## Exemple

### My web library

Book :

- Dune
- Fondation
- Chroniques de Durdane



Key	Value
0	Dune
2	Chroniques de Durdane
1	Fondation



- Véritable base de données embarquée
  - Gestion des transactions
  - Gestion des mises à jour de schémas
- Pour créer ou ouvrir une base de données, il faut préciser ses
  - ID
  - Version
  - Nom
  - Taille estimée (en octets)

```
var db = openDatabase('mydb', '1.0', 'Books', 1024 * 1024);  
if (db) {...}
```

- Gestion des transactions
  - La fonction *callback* est exécutée dans la transaction

```
db.transaction(function (tx) {  
    ...  
});
```

- Le paramètre *tx* fournit la méthode `executeSql`, qui permet d'effectuer les opérations SQL de lecture et écriture

```
tx.executeSql( query, [params], [callback] );
```

# Web SQL databases

## Exemple

```
var db = openDatabase('mydb', '1.0', 'Books', 1024 * 1024);

if (db) {


    db.transaction(function (tx) {


        // Ecriture
        tx.executeSql("CREATE TABLE BOOK
                        (ID unique, NAME text");
        tx.executeSql("INSERT INTO BOOK (ID,NAME)
                        VALUES (1,'Dune')");

        // Requêtage
        tx.executeSql("SELECT * FROM BOOK", [],
                        function (tx, books) {
                            window.alert(books);
                        });
    });


}
```









# Web SQL databases Exemple


 Elements


 Resources

»



▶  Frames	ID	NAME
▼  Databases	1	Dune
▼  mydb		
 <b>BOOK</b>		
▼  Local Storage		
▼  Session Storage		
▶  Cookies		
▼  Application Cache		





- La spécification n'est plus maintenue par le W3C
  - Pas assez d'implémentations différentes
  - Toutes basées sur SQLite
- Partiellement implémenté et/ou buggué
- Ne pas utiliser !



- Spécification en remplacement de Web SQL database, dépréciée en novembre 2010 par le W3C
- Base de données de type « object store »
  - Pas relationnelle, pas de schéma
  - Permet de stocker des objets javascript (json) identifiables
  - Création d'index pour récupérer les objets suivant des propriétés
  - APIs asynchrone et synchrone
    - *L'API synchrone est cependant peu implémentée et sujette à abandon par le W3C*
- Support : chrome 11+, Firefox 4+, IE 10

```
var indexedDB = window.indexedDB || window.webkitIndexedDB ||  
window.mozIndexedDB || window.msIndexedDB;
```

- Création/ouverture de la base

```
var request = indexedDB.open('AddressBook' , 1); //version 1
request.onsuccess = function(evt) {
    db = request.result;
};
request.onupgradeneeded = function(evt){
    //Alter table
    var reqVersion = db.setVersion('1');
    reqVersion.onsuccess = function (evt) { //operations...}
}
request.onerror = function(evt) {
    console.log("Database error code: " + evt.target.errorCode);
};
```

- Création d'un objet

```
var contact = db.createObjectStore(
    'contact',
    {keyPath:'id', autoIncrement:true}
);
```

# Web databases

## IndexedDB – transaction et opérations

- Création d'index

```
contact.createIndex("name", "name", { unique: false });  
contact.createIndex("email", "email", { unique: true });
```

- Transactions : opérations

- Ajout de données

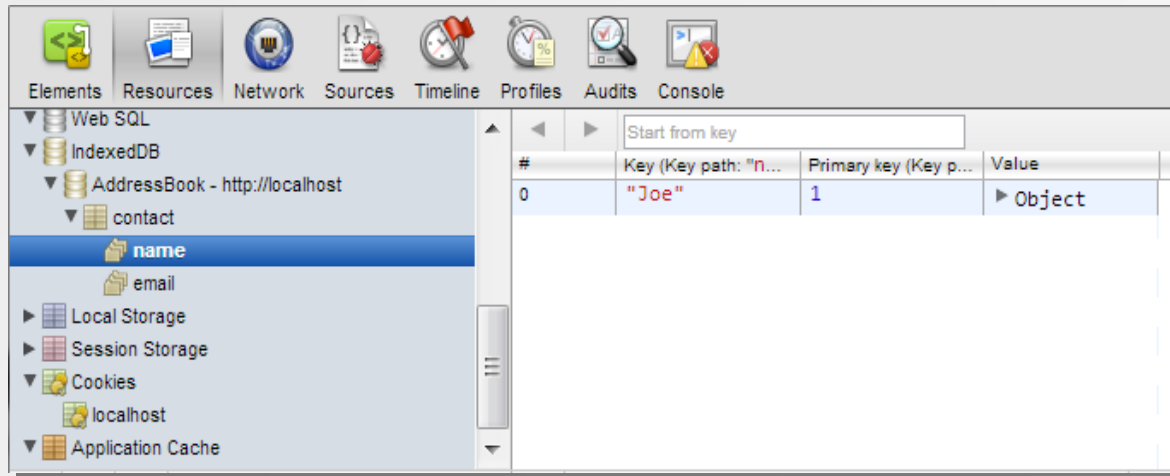
```
var tx = db.transaction(["contact", "readwrite"]);  
var store = tx.objectStore('contact');  
var contact1 = store.add({name: 'Joe', email: 'joe@mail.com'});
```

- Récupération d'objet avec une clé

```
var object = store.get(1);  
object.onsuccess = function(evt) {  
    var object_name = request.result.name;  
};
```



- Résultat : outils de développement chrome



- Conclusion
  - Nouvelle spécification
    - *Peu implémentée encore*
    - *En cours de rédaction → documentation à suivre*

- HTML 5 propose un mode "hors ligne"
  - Autorise une utilisation nomade et/ou déconnectée
  - Stocke les ressources en cache (HTML, scripts, images...)
- Détection de l'état de la connexion à Internet

```
if (navigator.onLine) { ... }  
  
window.addEventListener('online', function(e) {  
    // Re-sync data with server.  
}, false);  
  
window.addEventListener('offline', function(e) {  
    // Queue up events for server.  
}, false);
```



- Compatibilité : Chrome 19, Safari 5.1, Firefox 3.6, Opera 12

- Un Manifeste liste les ressources à mettre en cache
  - Fichier texte brut
  - Extension .appcache
  - Déclaré au niveau de la balise <html>

```
<html manifest="/offline.appcache">
```

- Toutes les pages de l'application devant fonctionner *offline* référencent le même manifeste

# Application déconnectées

## Le Manifeste - structure

- En-tête **CACHE MANIFEST**
- Section **CACHE** obligatoire  
*Liste des ressources à mettre en cache*
- Section **FALLBACK** optionnelle  
*Liste des ressources alternatives, à utiliser si les ressources réseau ne sont pas disponibles*
- Section **NETWORK** optionnelle  
*Liste des ressources devant être systématiquement accédées en ligne*

# Application déconnectées

## Le Manifeste - exemple

### CACHE MANIFEST

#### CACHE:

/css/screen.css  
/img/logo.png  
/js/app.js

#### FALLBACK:

/img/status-online.png /img/status-offline.png  
/js/app.js /js/offline-app.js

#### NETWORK:

/js/websockets.js

- Chaque page HTML référençant un manifeste est automatiquement mise en cache
  - Pas besoin de lister toutes les pages HTML !
- Pour empêcher l'utilisateur d'utiliser l'application, ou une partie de l'application, hors-ligne, il suffit de rediriger toutes ses pages vers une même ressource alternative

**FALLBACK :**  
`/online/ /offline-warning.html`

# Application déconnectées

## Le Manifeste – type MIME

- Configuration du type MIME

- Apache httpd

```
AddType text/cache-manifest .appcache
```

- Application web Java (web.xml)

```
<mime-mapping>  
  <extension>appcache</extension>  
  <mime-type>text/cache-manifest</mime-type>  
</mime-mapping>
```

# Application déconnectées

## Le Manifeste – mise en cache

- Attention à ne pas mettre le manifeste lui-même en cache !
  - Ajouter un commentaire qui change à chaque version

```
CACHE MANIFEST
# 2012-08-01 14:00

CACHE
...
```

- Apache httpd

```
<IfModule mod_expires.c>
  ExpiresActive On
  ExpiresByType text/cache-manifest "access plus 0 seconds"
</IfModule>
```





