Desarrollo de aplicaciones multiplataforma

Elaborado por: José Antonio Sánchez

Módulo 3

Programación HTML5

Innova4B

SESIÓN 6

Innova4B

11.3. Web SQL

Web SQL **no** forma parte de la especificación de HTML5, pero es ampliamente utilizado para el desarrollo de aplicaciones web

http://www.w3.org/TR/webdatabase/

Se trata de una base de datos basada en SQL, más concretamente en SQLite

La utilización del API se resume en tres simples métodos:

- openDatabase: abrir (o crear y abrir) una base de datos en el navegador del cliente
- transaction: iniciar una transacción
- executeSql: ejecutar una sentencia SQL

Innova4B

11.3. Web SQL

El API de Web SQL es asíncrono, por lo que es necesario tener cuidado con el orden en el que se ejecutan las sentencias SQL

Sin embargo, las sentencias SQL se encolan y son ejecutadas en orden, por lo que podemos estar tranquilos en ese sentido: podemos crear tablas y tener la seguridad que van a ser creadas antes de acceder a los datos

11.3. Web SQL

openDatabase

Al abrir la base de datos por primera vez, ésta es creada automáticamente

```
var db = openDatabase('mydb', '1.0', 'My first database', 2 * 1024 * 1024);
```

Es necesario especificar el **número de versión** de base de datos con el que se desea trabajar, y si no especificamos correctamente éste número de versión, es posible obtener error del tipo INVALID_STATE_ERROR

El valor de retorno de esta función es un objeto que dispone de un método transaction, a través del cual vamos a ejecutar las sentencias SQL.

11.3. Web SQL

openDatabase

```
var db = openDatabase('mydb', '1.0', 'My first database', 2 * 1024 * 1024);
```

Los valores que pasamos a esta función son los siguientes:

- El nombre de la base de datos
- El número de versión de base de datos con el que deseamos trabajar
- Un texto descriptivo de la base de datos
- Tamaño estimado de la base de datos, en bytes

La última versión de la especificación incluye un quinto argumento en la función, pero no es soportado por muchos navegadores

11.3. Web SQL

transaction

La idea de utilizar transacciones, en lugar de ejecutar las sentencias directamente, es por la posibilidad de realizar rollback, es decir, volver al estado inicial

```
db.transaction(function (tx) {
    tx.executeSql('DROP TABLE foo');
    tx.executeSql('INSERT INTO foo (id, text) VALUES (1, "foobar")');
}, function (err) {
    alert(err.message);
});
```

En ocasiones se fuerza un rollback indicando una consulta errónea (dirty trick), ya que no existe un método “abort”

Innova4B

11.3. Web SQL

executeSql

El método executeSql es utilizado tanto para sentencias de escritura como de lectura

Incluye protección contra ataques de inyección SQL y proporciona llamadas a métodos (callback) para procesar los resultados devueltos por una consulta SQL

```
db.transaction(function (tx) {
    tx.executeSql(sqlStatement, arguments, callback, errorCallback);
});
```

Innova4B

11.3. Web SQL

executeSql

```
tx.executeSql(sqlStatement, arguments, callback, errorCallback);
```

- **sqlStatement**: indica la sentencia SQL a ejecutar: creación de tabla, insertar un registro, realizar una consulta, etc
- **arguments**: corresponde con un array de argumentos que pasamos a la sentencia SQL. El propio método se ocupa de prevenir inyecciones SQL
- **callback**: función a ejecutar cuando la transacción se ha realizado de manera correcta. Toma como parámetros la propia transacción y el resultado de la transacción
- **errorCallback**: función a ejecutar cuando la transacción se ha producido un error en la sentencia SQL. Toma como parámetros la propia transacción y el error producido

11.3. Web SQL

executeSql

```
db.transaction(function (tx) {
    tx.executeSql('SELECT * FROM foo WHERE id = ?',
        [5],
        function callback(tx, results) {
            var len = results.rows.length, i;
            for (i = 0; i < len; i++) {
            alert(results.rows.item(i).text);
            }
        },
        function errorCallback(tx, error) {
            alert(error.message);
        }
    );
});
```

Innova4B

11.3. Web SQL

executeSql

La función de callback recibe como argumentos la transacción (de nuevo) y un objeto que contiene los resultados

Este objeto contiene una propiedad rows, donde **rows.item(i)** contiene la representación de la fila concreta

Si nuestra tabla contiene un campo que se llama nombre, podemos acceder a dicho campo de la siguiente manera:

```
results.rows.item(i).nombre
```

11.3. Web SQL

Crear tablas

La primera tarea a realizar cuando trabajamos con una base de datos es crear las tablas necesarias para almacenar los datos

Este proceso se realiza a través de una sentencia SQL, dentro de una transacción:

```
db = openDatabase('tweetdb', '1.0', 'All my tweets', 2 * 1024 * 1024);
db.transaction(function (tx) {
    tx.executeSql('CREATE TABLE IF NOT EXISTS tweets(id, user, date, text)', [], getTweets);
});
```

Innova4B

11.3. Web SQL

Insertar datos

El siguiente paso es insertar los datos correspondientes en las tablas. En este ejemplo la función getTweets realizaría una llamada AJAX para obtener ciertos datos

```
function getTweets() {
        var tweets = $.ajax({...});
        $.each(tweets, function(tweet) {
                db.transaction(function (tx) { //CADA TWEET EN UNA TRANSACCION → Procesa todas
                        var time = (new Date(Date.parse(tweet.created_at))).getTime();
                        tx.executeSql('INSERT INTO tweets (id, user, date, text) VALUES (?, ?, ?, ?)',
                                [tweet.id, tweet.from_user, time / 1000, tweet.text]);
                });
        });
}
```

Innova4B

11.3. Web SQL

Obtener datos

Una vez que los datos se encuentran almacenados en la base de datos, sólo nos queda consultarlos y proporcionar la función que los procese

```
db.transaction(function (tx) {
    tx.executeSql('SELECT * FROM tweets WHERE date > ?', [time],
        function(tx, results) {
            var html = [], len = results.rows.length;
            for (var i = 0; i < len; i++) {
                html.push('<li>' + results.rows.item(i).text + '</li>');
            }
            tweetEl.innerHTML = html.join('');
        });
    });
}
```

Innova4B

11.4. Ejercicio práctico

Crear un objeto que encapsule una base de datos WebSQL, que nos permitar acceder a una base de datos para añadir, modificar, eliminar y obtener registros. Dicha base de datos va a almacenar tweets procedentes de Twitter, que tienen asociado el hashtag #html5. Los requisitos son los siguientes:

- Disponer de una tabla para almacenar los tweets. Los campos mínimos son: identificador del tweet, texto, usuario, y fecha de publicación
- Disponer de una tabla para almacenar los usuarios que publican los tweets. Esta tabla debe estar relacionada con la anterior. Los campos mínimos son: identificador del usuario, nombre e imagen
- Crear un método addTweet que dado un objeto que corresponde con un tweet, lo almacene en la base de datos. Almacenar el usuario en caso de que no exista, o relacionarlo con el tweet si existe

Innova4B

11.4. Ejercicio práctico

- Crear un método removeTweet que dado un identificador de tweet, lo elimine de la base de datos. Éste método debe devolver el tweet eliminado
- Crear un método updateTweet que dado un objeto que corresponde con un tweet, actualice los datos correspondientes al tweet en la base de datos
- Crear un método getTweets que dado un parámetro de fecha, me devuelva todos los tweets posteriores a esa fecha. Cada tweet debe incluir sus datos completos y el usuario que lo creó

Crear un archivo JSON con tweets de prueba y simular una consulta AJAX que realice la petición de los datos

Innova4B

11.5. IndexedDB

IndexedDB **no** es una base de datos relacional, sino que se podría llamar un almacén de objetos

En la base de datos existirán almacenes y en su interior añadimos objetos

```
{
    id:21992,
    nombre: "Memoria RAM"
}
```

Innova4B

11.5. IndexedDB

Los pasos que seguiremos en IndexedDB serán:

- Abrir una base de datos indicando su nombre y la versión concreta
- Crear los almacenes de objetos, que es muy parecido a un archivador con índices, que nos permite encontrar de una manera muy rápida el objeto que buscamos
- Almacenar cualquier tipo de objeto con el índice que definamos
- No importa el tipo de objeto que almacenemos, ni tienen que tener las mismas propiedades

11.5. IndexedDB

Abrir la base de datos

Al abrir una base de datos por primera vez se creará, después podremos trabajar directamente con el objeto que nos devuelve

Al igual que en Web SQL, las peticiones a la base de datos se realizan de manera asíncrona

Previamente, necesitaremos hacer comprobaciones sobre el navegador

11.5. IndexedDB

Abrir la base de datos

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

window.IDBTransaction = window.IDBTransaction ||
        window.webkitIDBTransaction || window.msIDBTransaction;

window.IDBKeyRange = window.IDBKeyRange ||
        window.webkitIDBKeyRange || window.msIDBKeyRange;
```

Innova4B

11.5. IndexedDB

Abrir la base de datos

Para abrir la base de datos únicamente necesitamos un nombre y la versión

```
var request = indexedDB.open('videos', 1);
request.onerror = function () {
        console.log('failed to open indexedDB');
};
request.onsuccess = function (event) {
        // handle version control first
        // then create a new object store
};
```

Innova4B

11.5. IndexedDB

Abrir la base de datos

Ahora que la base de datos esta abierta (y asumiendo que no hay errores), se ejecutará el evento onsuccess

Antes de poder crear almacenes de objetos, tenemos que tener en cuenta los siguiente:

- Necesitamos un manejador para poder realizar transacciones de inserción y obtención de datos
- Hay que especificar la versión de la base de datos. Si no existe una versión definida, significa que la base de datos no está aún creada

11.5. IndexedDB

Abrir la base de datos

El método onsuccess recibe como parámetro un evento, dentro de este objeto, encontramos una propiedad llamada target, y dentro de ésta otra propiedad llamada result, que contiene el resultado de la operación

```
var db = null;
var request = window.indexedDB.open('videos');
request.onsuccess = function (event) {
        db = event.target.result;
        // Create a new object store
};
```

Innova4B

11.5. IndexedDB

Abrir la base de datos

```
request.onerror = function (event) {
        alert('Something failed: ' + event.target.message);
};
```

En IndexedDB, los errores que se producen escalan hasta el objeto request

Si un error ocurre en cualquier petición (por ejemplo una consulta de datos), se ejecutaría “request.onerror”

Innova4B

11.5. IndexedDB

Control de versiones

Como identificador de la versión podemos utilizar cualquier cadena, aunque lo lógico es seguir un patrón típico de software, como '0.1', '0.2'...

Al abrir la base de datos, debemos comprobar si la versión actual de la base de datos coincide con la última versión de la aplicación

En “setVersion” es el único lugar que podemos modificar la estructura de la base de datos: crear y eliminar almacenes de objetos e índices

setVersion se ha marcado como **deprecated** así que será necesario utilizar alguna otra forma de comprobar la versión

11.5. IndexedDB

Control de versiones

```
var db = null, version = '0.1';
request.onsuccess = function (event) {
        db = event.target.result; // cache a copy of the database handle for the future
        if (version != db.version) { // handle version control
                var verRequest = db.setVersion(version); // ERROR on modern browsers
                verRequest.onsuccess = function (event) {
                        // now we're ready to create the object store!
                };
                verRequest.onerror = function () {
                        alert('unable to set the version :' + version);
                };
        }
};
```

Innova4B

11.5. IndexedDB

Control de versiones

```
request.onupgradeneeded = function(event) {
        var db = event.target.result;
        // Create object store
}
```

Si se necesita actualizar, crear o borrar la base de datos, debe utilizarse el manejador para “onupgradeneeded”

Este evento se llama cuando se modifica la versión de la base de datos, o cuando se crea, por tanto es equivalente al caso anterior

Es en el **único** lugar que deberíamos poner las consultas que modifiquen la estructura de la base de datos

Innova4B

11.5. IndexedDB

Crear almacenes de objetos

Tanto la primera vez que creamos nuestra base de datos, como a la hora de actualizarla, debemos crear los almacenes de objetos correspondientes

```
request.onupgradeneeded = function(event) {
        db = event.target.result;
        if(db.objectStoreNames.contains('blockbusters')) db.deleteObjectStore('blockbusters');

        var store = db.createObjectStore('blockbusters', {
                keyPath: 'title',
                autoIncrement: false
        });
        // Object store is ready!!
};
```

Innova4B

11.5. IndexedDB

Crear almacenes de objetos

Lo habitual es disponer de varios almacenes que puedan relacionarse entre ellos, así el método createObjectStore admite dos parámetros:

- name: nombre del almacén de objetos
- optionalParameters: define el índice de los objetos, a través del cual se van a realizar las búsquedas (**único**). Si no deseamos que este índice se incremente automáticamente al añadir nuevos objetos, también lo podemos indicar aquí (autoincrement: false)

Al añadir un objeto al almacén, es importante que disponga de la propiedad que se ha definido como índice (“title”), y que su valor sea único

11.5. IndexedDB

Crear almacenes de objetos

Es posible que deseemos añadir nuevos índices a nuestros objetos, con el fin de poder realizar búsquedas posteriormente

```
store.createIndex('director', 'director', { unique: false });
```

Hemos añadido un nuevo índice al almacén (“director”) y hemos indicado que el nombre de la propiedad del objeto es director (segundo argumento), a través del cual vamos a realizar las búsquedas

Como, varias películas pueden tener el mismo director, por lo que este valor no puede ser único

Innova4B

11.5. IndexedDB

Crear almacenes de objetos

De esta manera, podemos almacenar objetos y realizar búsquedas tanto por “title” (único) como por “director”

```
{
    title: "Belly Dance Bruce - Final Strike",
    date: (new Date).getTime(),
    director: "Bruce Awesome",
    length: 169, // in minutes
    rating: 10,
    cover: "/images/wobble.jpg"
}
```

Innova4B

11.5. IndexedDB

Añadir objetos al almacén

- add: añade un nuevo objeto al almacén. Es obligatorio que los nuevos datos no existan en el almacén, si no provocaría un ConstraintError
- put: actualiza el valor del objeto si existe, o lo añade si no existe en el almacén.

```
var video = {
        title: "Belly Dance Bruce - Final Strike",
        date: (new Date).getTime(),
        director: "Bruce Awesome",
        length: 169, // in minutes
        rating: 10,
        cover: "/images/wobble.jpg"
}
```

Innova4B

11.5. IndexedDB

Añadir objetos al almacén

```
var transaction = db.transaction(['blockbusters'], 'readwrite');
var store = transaction.objectStore('blockbusters');
// var request = store.add(video);
var request = store.put(video);
```

Crea una nueva transacción de lectura/escritura, sobre los almacenes indicados (en este caso sólo 'blockbusters')

La constante IDBTransaction.READ_WRITE se ha marcado como “deprecated”

```
var myIDBTransaction = window.IDBTransaction  || window.webkitIDBTransaction
   || { READ_WRITE: "readwrite" };
var transaction = db.transaction(['blockbusters'], myIDBTransaction.READ_WRITE);
```

Innova4B

11.5. IndexedDB

Añadir objetos al almacén

```
var transaction = db.transaction(['blockbusters'], 'readwrite');
var store = transaction.objectStore('blockbusters');
// var request = store.add(video);
var request = store.put(video);
```

Obtiene el almacén de objetos sobre el que queremos realizar las operaciones, que debe ser uno de los indicados en la transacción

Con la referencia a este objeto, podemos ejecutar las operaciones de add, put, get, delete, etc

Innova4B

11.5. IndexedDB

Añadir objetos al almacén

```
var transaction = db.transaction(['blockbusters'], 'readwrite');
var store = transaction.objectStore('blockbusters');
// var request = store.add(video);
var request = store.put(video);
```

Inserta el objeto en el almacén

Si la transacción se ha realizado correctamente, se llamará al evento onsuccess, si no se llamará a onerror

Innova4B

11.5. IndexedDB

Obtener objetos del almacén

Se necesita una transacción, pero en este caso de sólo lectura

```
var key = "Belly Dance Bruce - Final Strike";
//var transaction = db.transaction(['blockbusters']);
var transaction = db.transaction(['blockbusters'], 'read');
var store = transaction.objectStore('blockbusters');
var request = store.get(key);
```

La variable “key” del método get, buscará el valor que contiene en la propiedad que definimos como “keyPath” al crear el almacén

Innova4B

11.5. IndexedDB

Obtener objetos del almacén

El método “get” produce el mismo resultado tanto si el objeto existe en el almacén como si no (devuelve undefined)

Es recomendable utilizar el método “openCursor()” para que si el objeto no existe, el valor del resultado sea *null*

Este método “openCursor” también permitirá obtener todos los objetos de un almacén, en lugar de un único objeto, pasando como parámetro un objeto de tipo IDBKeyRange

11.5. IndexedDB

Obtener objetos del almacén

https://developer.mozilla.org/en-US/docs/Web/API/IDBKeyRange

- IDBKeyRange.upperBound(x) – Claves ≤ x
- IDBKeyRange.upperBound(x, true) – Claves < x
- IDBKeyRange.lowerBound(y) – Claves ≥ y
- IDBKeyRange.lowerBound(y, true) – Claves > y
- IDBKeyRange.bound(x, y) – Claves ≥ x && ≤ y
- IDBKeyRange.bound(x, y, true, true) – Claves > x && < y
- IDBKeyRange.bound(x, y, true, false) – Claves > x && ≤ y
- IDBKeyRange.bound(x, y, false, true) – Claves ≥ x && < y
- IDBKeyRange.only(z) – Clave = z

Innova4B

11.5. IndexedDB

Obtener objetos del almacén

```
var transaction = db.transaction(['blockbusters']);
var store = transaction.objectStore('blockbusters');
var data = [];
var request = store.openCursor();
// var singleKeyRange = IDBKeyRange.only("Belly Dance Bruce - Final Strike");
// var request = store.openCursor(singleKeyRange);
request.onsuccess = function (event) {
        var cursor = event.target.result;
        if (cursor) {
                data.push(cursor.value); // value is the stored object
                cursor.continue(); // get the next object
        } else {
                //Objects are in data[]
        }
};
```

Innova4B

11.5. IndexedDB

Eliminar objetos del almacén

```
var transaction = db.transaction(['blockbusters'], 'readwrite');
var store = transaction.objectStore('blockbusters');
var request = store.delete(key);
```

Para eliminar todos los objetos de un almacén, podemos utilizar el método clear()

```
var request = store.clear();
```

Innova4B

11.5. IndexedDB

Uso de un índice

Si existe otro índice por el que queremos buscar

```
store.createIndex('director', 'director', { unique: false });
```

Se tiene que indicar el índice sobre el almacén de objetos y realizar la búsqueda con “get()”

```
var index = objectStore.index('director');
index.get("Bruce Awesome").onsuccess = function(event) {
        alert("Bruce Awesome film: " + event.target.result.title);
};
```

Innova4B

11.5. IndexedDB

Uso de un índice

Igualmente, podremos realizar la búsqueda con un cursor y especificar límites con IDBKeyRange

```
var index = objectStore.index('director');
var boundKeyRange = IDBKeyRange.bound("Bill", "Donna", false, true);

index.openCursor(boundKeyRange).onsuccess = function(event) {
        var cursor = event.target.result;
        if (cursor) {
                // Do something with the matches.
                cursor.continue();
        }
};
```

Innova4B

11.6. Ejercicio práctico

Crear un objeto que encapsule una base de datos IndexedDB, que nos permita acceder a una base de datos para añadir, modificar, eliminar y obtener registros. Dicha base de datos va a almacenar una sencilla lista de tareas pendientes. Los requisitos son:

- Disponer de un almacén de tareas pendientes. Sus propiedades son: un identificador único que actúa como índice, el texto descriptivo, una propiedad que nos indique si la tarea está completada o no y la fecha/hora de creación
- Crear un método addTask que dado un objeto que corresponde con una tarea, lo almacene en la base de datos
- Crear un método removeTask que dado un identificador de una tarea, lo elimine de la base de datos. Éste método debe devolver la eliminada.
- Crear un método updateTask que dado un identificador de una tarea, actualice los datos correspondientes a la tarea en la base de datos
- Crear un método getTasks que dado un parámetro booleano completado, nos devuelva las tareas que se encuentran completadas o no