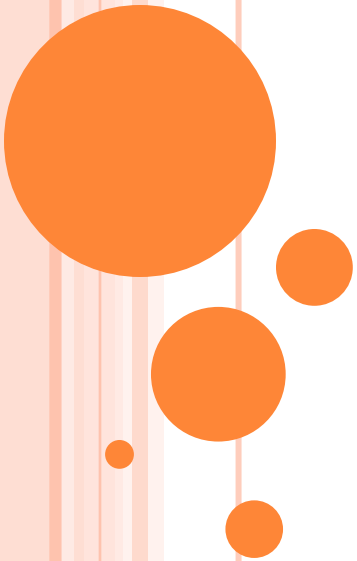

IndexedDB

Acceso a datos

Modelos de bases de datos

- Modelo local: localStorage, permite almacenar atributos y valores.
- WebSQL: base de datos SQLite integrada en el navegador. Lo implementan la mayoría de los navegadores. Ha sido abandonado por W3C.
- IndexedDB: almacén de objetos.

IndexedDB

- IndexedDB es una API del lado del cliente, para el almacenamiento de grandes cantidades de datos estructurados y para búsquedas de alto rendimiento en esos datos, usando índices.

- IndexedDB almacena pares llave-valor. Los valores pueden ser objetos con estructuras complejas, y las llaves pueden ser propiedades de esos objetos.

- IndexedDB está hecho sobre un modelo de base de datos transaccional.

Características

- La API de IndexedDB es asincrona. La API no devuelve datos, es necesario pasarle una función *callback*.
- Para acceder a IndexedDB se utilizan peticiones.
- Las respuestas a las peticiones se reciben en forma de eventos DOM: onsuccess, onerror.
- Es orientado a objetos.

Limitaciones

- En principio no hay ningún límite de tamaño.
- No está soportado el ordenamiento internacional de los datos.
- No está implementada la sincronización contra un servidor.
- El usuario puede borrar los datos del navegador.
- El usuario puede desinstalar el navegador.
- No utiliza SQL.

Creación de una base de datos

```
var pet = indexedDB.open('nombrebasedatos', version);
pet.onsuccess = function (evt) {
    db = this.result;
};
pet.onerror = function (evt) {
};
pet.onupgradeneeded = function (evt) {
    var tabla =
        evt.currentTarget.result.createObjectStore(
        'nombrecoleccion',
        { keyPath: 'idcoleccion', autoIncrement: true });
    tabla.createIndex('nombre', 'nombre', { unique: true });
    tabla.createIndex('desc', 'desc', { unique: false });
};
```

Insertar un registro

```
var obj = {nombre: v1, desc: v2};
var trans = db.transaction('nombrecoleccion', 'readwrite');
var tabla = trans.objectStore('nombrecoleccion');
var pet;
try {
    pet = tabla.add(obj);
} catch (e) {
    throw e;
}
pet.onsuccess = function (evt) {
};
pet.onerror = function() {
};
```

Número de elementos

```
var trans = db.transaction('nombrecoleccion',
    'readonly');
var tabla = trans.objectStore('nombrecoleccion');
var pet = tabla.count();
pet.onsuccess = function(evt) {
    var r = evt.target.result;
};
pet.onerror = function(evt) {
};
```

Listado de elementos

```
var trans = db.transaction('nombrecoleccion', 'readonly');
var tabla = trans.objectStore('nombrecoleccion');
var pet = tabla.openCursor(), i = 0;pet.onsuccess = function(evt) {
    var cursor = evt.target.result;
    if (cursor) {
        pet = tabla.get(cursor.key);
        pet.onsuccess = function (evt) {
            var value = evt.target.result;
            var r1 = cursor.key;
            var r2 = value.nombre;
            var r3 = value.desc;
        };
        cursor.continue();
        i++;
    }
};
pet.onerror = function(evt) {
}
```

Eliminar todos los registros de una tabla

```
var trans = db.transaction('nombrecoleccion',
    'readwrite');
var tabla = trans.objectStore('nombrecoleccion');
var pet = tabla.clear();
pet.onsuccess = function(evt) {
};
pet.onerror = function (evt) {
};
```

Buscar un registro usando la clave

```
var trans = db.transaction('nombrecoleccion', 'readonly');
var tabla = trans.objectStore('nombrecoleccion');
var clave = Number(v); //!!
var pet = tabla.get(clave);
pet.onsuccess = function(evt) {
    var registro = evt.target.result;
    if (typeof registro == 'undefined') {
        return; //no encontrado
    }
    r = registro.idcoleccion+" "+registro.nombre+"
   "+registro.desc;
};
pet.onerror = function (evt) {
};
```

Buscar un registro sin usar la clave

```
var trans = db.transaction('nombrecoleccion', 'readonly');
var tabla = trans.objectStore('nombrecoleccion');
var pet = tabla.index("nombre");
pet.get(v).onsuccess = function(evt) {
    if (typeof evt.target.result == 'undefined') {
        return; //no encontrado
    }
    var clave = evt.target.result.idcoleccion;
    var nombre = evt.target.result.nombre;
    var desc = evt.target.result.desc;
};
pet.onerror = function (evt) {
};
```

Eliminar un registro usando la clave

```
var trans = db.transaction('nombrecoleccion', 'readwrite');
var tabla = trans.objectStore('nombrecoleccion');
var clave = Number(v); //!!
var pet = tabla.get(clave);
pet.onsuccess = function(evt) {
    var registro = evt.target.result;
    if (typeof registro == 'undefined') {
        return; //no encontrado
    }
    pet = tabla.delete(clave);
    pet.onsuccess = function(evt) {
    };
    pet.onerror = function (evt) {
    };
};
pet.onerror = function (evt) {
};
```

Eliminar un registro sin usar la clave

```
var trans = db.transaction('nombrecoleccion', 'readwrite');
var tabla = trans.objectStore('nombrecoleccion');
var pet = tabla.index("nombre");
pet.get(v).onsuccess = function(evt) {
    if (typeof evt.target.result == 'undefined') {
        return; //no encontrado
    }
    var clave = evt.target.result.idcoleccion;
    pet = tabla.get(clave);
    pet.onsuccess = function(evt) {
        var registro = evt.target.result;
        if (typeof registro == 'undefined') {
            return;
        }
        pet = tabla.delete(clave);
        pet.onsuccess = function(evt) {
        };
        pet.onerror = function (evt) {
        };
    };
    pet.onerror = function (evt) {
    };
};
```

Editar un registro

```
var trans = db.transaction('nombrecoleccion', 'readwrite');
var tabla = trans.objectStore('nombrecoleccion');
var clave = Number(v1);
var pet = tabla.get(clave);
pet.onerror = function(event) {
};
pet.onsuccess = function(event) {
    var registro = pet.result; //evt.target.result
    registro.desc = v2;
    pet = tabla.put(registro);
    pet.onsuccess = function(event) {
    };
    pet.onerror = function(event) {
    };
};
```

Listado selectivo

- var rango = IDBKeyRange.only(valor);
- var rango = IDBKeyRange.lowerBound(valor);
- var rango = IDBKeyRange.upperBound(valor);
- var rango = IDBKeyRange.bound(valor1, valor2, true, true);

Ejemplo

```
var trans = db.transaction('nombrecoleccion',
   'readonly');
var tabla = trans.objectStore('nombrecoleccion');
var pet = tabla.index("nombre");
var rango = IDBKeyRange.bound("a", "f", true, false);
pet.openCursor(rango).onsuccess = function(evt) {
    var cursor = evt.target.result;
    if (cursor) {
        var jsonStr = JSON.stringify(cursor.value);
        cursor.continue();
    }
};
```

Transacciones

- Las transacciones se finalizan implícitamente.
- El método transaccion.abort(); permite cancelar una transacción antes de que haya finalizado.
- Definición de función callback para las transacciones:

```
transaccion.oncomplete = function(evt){…}

transaccion.onerror = function(evt){}
```

- https://developer.mozilla.org/en-US/docs/IndexedDB/Using_IndexedDB
- http://www.ibm.com/developerworks/library/wa-indexeddb/