Master ICONE 1ère année

Base de données avancées - *IndexedDB*

Réaliser par :
Mohammed BENAOU

Liste des exercices

1 Exercice 1

la création du base de données MyBase en version 1 :

```
MyProject.openDatabase = function () {
    var request = indexedDB.open("MyBase", 1);
    request.onerror = function (event) {
        console.log("Error");
    }
    request.onsuccess = function (event) {
        //console.log(event);
```

```
8           MyProject.db=event.target.result;
9           console.log("La base de données est ouvete ");
10      }
```

2 Exercice 2

la suppression de la base de données :

```
1  MyProject.deleteDatabase = function () {
2      var request = indexedDB.deleteDatabase("MyBase");
3      request.onerror = function (event) {
4          console.log("Erreur lors de la suppression de la base");
5      }
6      request.onsuccess = function (event) {
7          MyProject.db=event.target.result;
8           console.log("Suppression de la base réussie");
9      }
10 };
```

3 Exercice 3

```
1  request.onupgradeneeded = function (event) {
2                  var db = event.target.result;
3                  //l'option autoIncrement définie à true.
4                  var objectStore = db.createObjectStore("Person", {
       keyPath: "id",autoIncrement:true});
5                if (event.oldVersion < 3 ) {
6                  // on peut avoir des doublons  alors on n'utilise
       pas d'index unique.
7                  this.transaction.objectStore("Person").createIndex(
       "name", "name",{unique: false});}
8                if (event.oldVersion < 4 ) {
9                  // assurer que deux personnes n'auront pas la même
       adresse mail, donc nous utilisons un index unique.
10                 this.transaction.objectStore("Person").createIndex(
       "address", "email",{unique: true});}
```

4 Exercice 4

l'ajout de personne :

```
1  MyProject.insertPerson = function(person){
2  var request = indexedDB.open("MyBase");
3  var transaction = MyProject.db.transaction(["Person"], "readwrite")
       ;
4  var objectStore = transaction.objectStore("Person");
5  for(var i in person){
6  var request = objectStore.add(person[i]);
7  }
```

```
  request.onerror = function(event){
    console.log("Erreur d'ajout");
  }
  request.onsuccess = function(event){
    console.log("Personne(s) est ajouté à la base de données");
  }
}
```

5 Exercice 5

Recherche de personne

```
// la recherche par l'identifiant passé en paramètre
MyProject.searchPersonById = function(id){
  var T = MyProject.db.transaction(["Person"], "readonly");
  var objectStore = T.objectStore("Person");
  var request = objectStore.get(id);

  request.onerror = function(event){
    console.log("Erreur de recherche");
  }

  request.onsuccess = function(event){
    // si le résultat est inconnu donc la personne n'existe pas
    if (typeof this.result == "undefined") {
      console.log("Cet identifiant n'existe pas dans la base de
    données");
    } else {
      console.log("La personne trouvée => Name: " + request.result.
    name + ", Email: " + request.result.email);
    }
  }
}
```

```
> MyProject.openDatabase()
< undefined
  La base de donnÃ©es est ouvete
> var person = [{ name: "Gopal", email: "contact@gmail.com" },{ name: "Prasad", email: "prasad@tutorialspoint.com" }];
< undefined
> MyProject.insertPerson(person)
< undefined
  Personne(s) est ajoutÃ© Ã  la base de donnÃ©es
>
```

6 Exercice 6

Rechercher et d'afficher une personne à partir son id :

```
// la recherche par l'identifiant passé en paramètre
MyProject.searchPersonById = function(id){
  var T = MyProject.db.transaction(["Person"], "readonly");
  var objectStore = T.objectStore("Person");
  var request = objectStore.get(id);

```

```
7     request.onerror = function(event){
8       console.log("Erreur de recherche");
9     }
10    request.onsuccess = function(event){
11      // si le résultat est inconnu donc la personne n'existe pas
12      if (typeof this.result == "undefined") {
13        console.log("Cet identifiant n'existe pas dans la base de
      données");
14      } else {
15        console.log("La personne trouvée => Name: " + request.result.
      name + ", Email: " + request.result.email);
16      }
17    }
18  }
```

7 Exercice 7

Rechercher et d'afficher une personne à partir son nom :

```
1   MyProject.searchPersonByName = function (name){
2     var T = MyProject.db.transaction(["Person"], "readonly");
3     var objectStore = T.objectStore("Person");
4     var request = objectStore.index("name").get(name);
5     request.onerror = function(event){
6       console.log("erreur  de recherche ");
7     }
8     request.onsuccess = function(event){
9       if (typeof this.result == "undefined") {
10        console.log("Cet identifiant n'existe pas dans la base de
      données");
11      } else {
12              console.log("La personne trouvée => Name: " + request.
      result.name + ", Email: " + request.result.email);
13      }
14    }
15  }
```

```
> MyProject.searchPersonById(1)
< undefined
  La personne trouvÃ©e => Name: Gopal, Email: contact@gmail.com
> MyProject.searchPersonByEmail("prasad@tutorialspoint.com")
< undefined
  La personne trouvÃ©e => Name: Prasad, Email: prasad@tutorialspoint.com
> MyProject.searchPersonByName("Gopal")
< undefined
  La personne trouvÃ©e => Name: Gopal, Email: contact@gmail.com
>
```

8 Exercice 8

Création du schéma des livres :

```
var objectStore = db.createObjectStore("Book", {keyPath: "id",
    autoIncrement: true});
                this.transaction.objectStore("Book").createIndex("
    bookName", "bookName",{unique: false});}
```

Insérer les livres dans la table book :

```
MyProject.insertBook = function(Book){
  //const name = [ { bookName: "le rouge et le noir"},{ bookName: "
    candide"} ];
  var request = indexedDB.open("MyBase");
  var P = MyProject.db.transaction(["Book"], "readwrite");
  var objectStore = P.objectStore("Book");
  for(var i in Book){
  var request = objectStore.add(Book[i]);
 }
  request.onerror = function(event){
    console.log("Erreur d'ajout");
  }
  request.onsuccess = function(event){
    console.log("Livre(s) est ajouté à la base de données");
  }
};
```

rechercher et afficher les noms du livres à partir leur id :

```
MyProject.searchBookById = function(id){
  var T = MyProject.db.transaction(["Book"], "readonly");
  var objectStore = T.objectStore("Book");
  var request = objectStore.get(id);

  request.onerror = function(event){
    console.log("Erreur de recherche");
  }

  request.onsuccess = function(event){
    // si le résultat est inconnu donc la personne n'existe pas
    if (typeof this.result == "undefined") {
      console.log("Cet identifiant n'existe pas dans la base de
    données");
    } else {
      console.log("Le nom du livre est : " + request.result.
    bookName);
    }
  }
}
```

```
> var book = [ { bookName: "le rouge et le noir"},{ bookName: "candide"},{ bookName: "l'ile au trésor"} ];
< undefined
> MyProject.insertBook(book)
< undefined
Livre(s) est ajoutÃ© Ã  la base de donnÃ©es
> MyProject.searchBookById(2)
< undefined
Le nom du livre est : candide
>
```

9 Exercice 9

Création de la table Emprunte :

```
var objectStore = db.createObjectStore("Emprunt", {keyPath: "id",
    autoIncrement: true});
                this.transaction.objectStore("Emprunt").createIndex
    ("idperson", "idperson");
                this.transaction.objectStore("Emprunt").createIndex
    ("idbook", "idbook");}
```

Ajouter un nouvel emprunte dans la table :

```
MyProject.insertBorrow = function (idperson,idbook){
  var request = indexedDB.open("MyBase");
  var P = MyProject.db.transaction(["Emprunt"], "readwrite");
  var objectStore = P.objectStore("Emprunt");
  var borow ={idperson:idperson,idbook:idbook}
    var request = objectStore.add(borow);

    request.onerror = function(event){
    console.log("Erreur d'ajout");
    }

    request.onsuccess = function(event){
    console.log("un nouvel emprunt est ajouté à la base");
    }
}
```

Afficher la personne qui a emprunté un livre en utilisant le id de ce dernier :

```
MyProject.searchBorrowByBookId = function (idbook){
  //var db = event.target.result;
  var T = MyProject.db.transaction(["Emprunt"], "readonly");
  var objectStore = T.objectStore("Emprunt");
  var request = objectStore.index("idbook").get(idbook);
  request.onerror = function(event){
    console.log("Erreur de recherche");
  }
  request.onsuccess = function(event){
    if (typeof this.result == "undefined") {
      console.log("Cet identifiant n'existe pas dans la base de
    données");
    } else {
          console.log("L'emprunteur : ");
           MyProject.searchPersonById(request.result.idperson);
    }
  }
}
```

afficher les livres empruntés par une personne en utilisant son identifiant :

```
MyProject.searchBorrowByPersonId = function(idperson){
  var T = MyProject.db.transaction(["Emprunt"], "readonly");
  var objectStore = T.objectStore("Emprunt");
  var request = objectStore.index("idperson").get(idperson);
```

```
5     request.onerror = function(event){
6       console.log("Erreur de recherche");
7     }
8     request.onsuccess = function(event){
9       if (typeof this.result == "undefined") {
10        console.log("Cet identifiant n'existe pas dans la base de
        données");
11      } else {
12          //console.log("La personne a emprunté  : "+request.result.
        idbook);
13          for (var i in request.result.idbook){
14            console.log("La personne a emprunté  : ");
15             MyProject.searchBookById(request.result.idbook[i]); }
16      }
17    }
18  }
```

afficher les livres empruntés par une personne en utilisant son email :

```
1   MyProject.searchBorrowByPersonEmail = function (email){
2     var T = MyProject.db.transaction(["Person"], "readonly");
3     var objectStore = T.objectStore("Person");
4     var request = objectStore.index("address").get(email);
5     request.onerror = function(event){
6       console.log("Erreur de recherche");
7     }
8     request.onsuccess = function(event){
9       if (typeof this.result == "undefined") {
10        console.log("Cet identifiant n'existe pas dans la base de
        données");
11      } else {
12          MyProject.searchBorrowByPersonId(request.result.id);
13      }
14    }
15  }
```

```
> MyProject.insertBorrow(1,[2,3])
undefined
un nouvel emprunt est ajoutÃ© Ã  la base
> MyProject.searchBorrowByPersonId(1)
undefined
2 La personne a empruntÃ©  :
Le nom du livre est : candide
Le nom du livre est : l'ile au trésor
> MyProject.searchBorrowByPersonEmail("prasad@tutorialspoint.com")
undefined
Cet identifiant n'existe pas dans la base de donnÃ©es
> MyProject.searchBorrowByPersonEmail("contact@gmail.com")
undefined
2 La personne a empruntÃ©  :
Le nom du livre est : candide
Le nom du livre est : l'ile au trésor
```

Vous trouvez ci-joint le fichier bd.js qui contient l'ensemble des travaux demandés