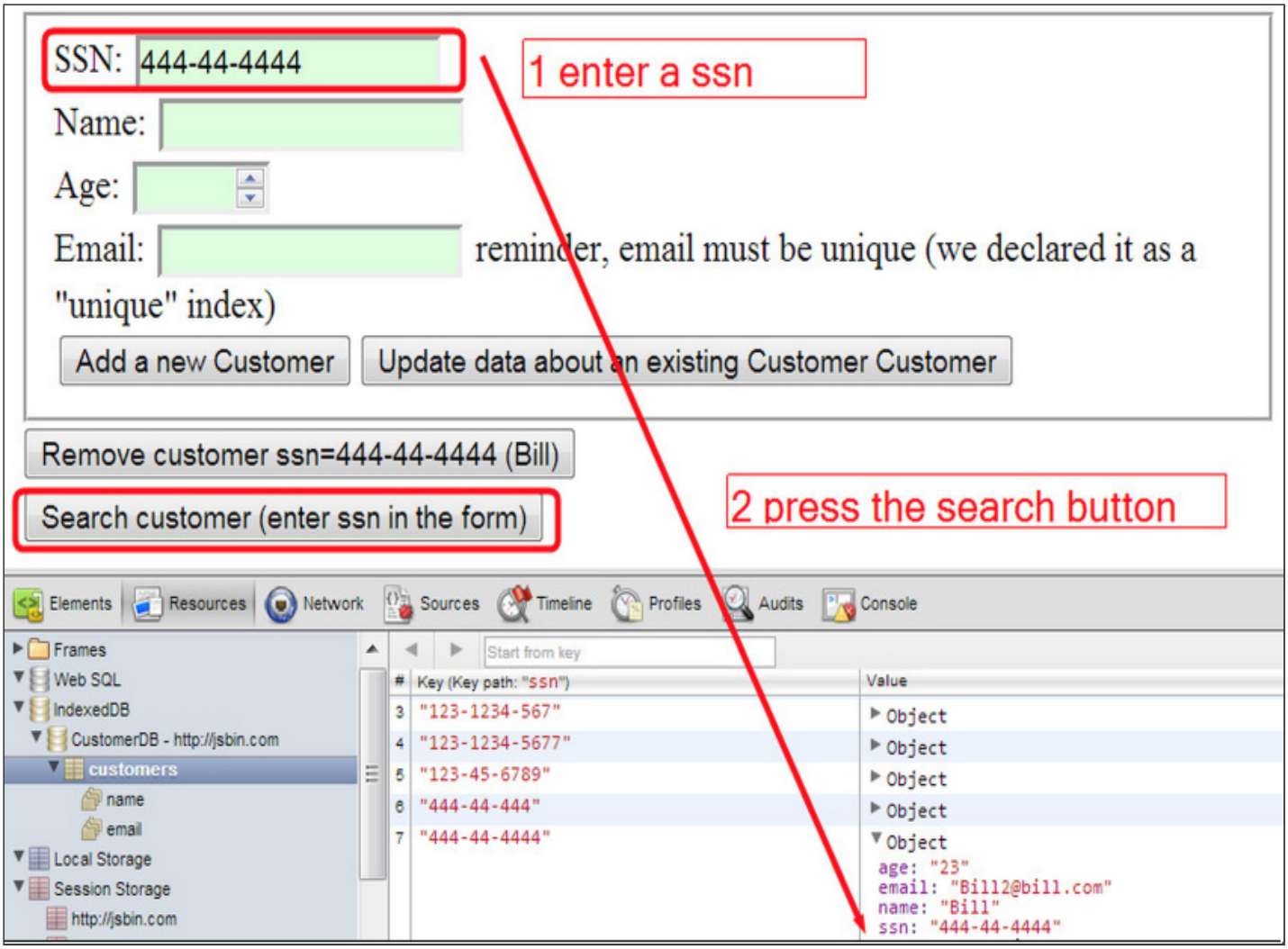
Getting data from a data store

There are several ways to retrieve data from a data store.

FIRST METHOD: GETTING DATA WHEN WE KNOW ITS KEY

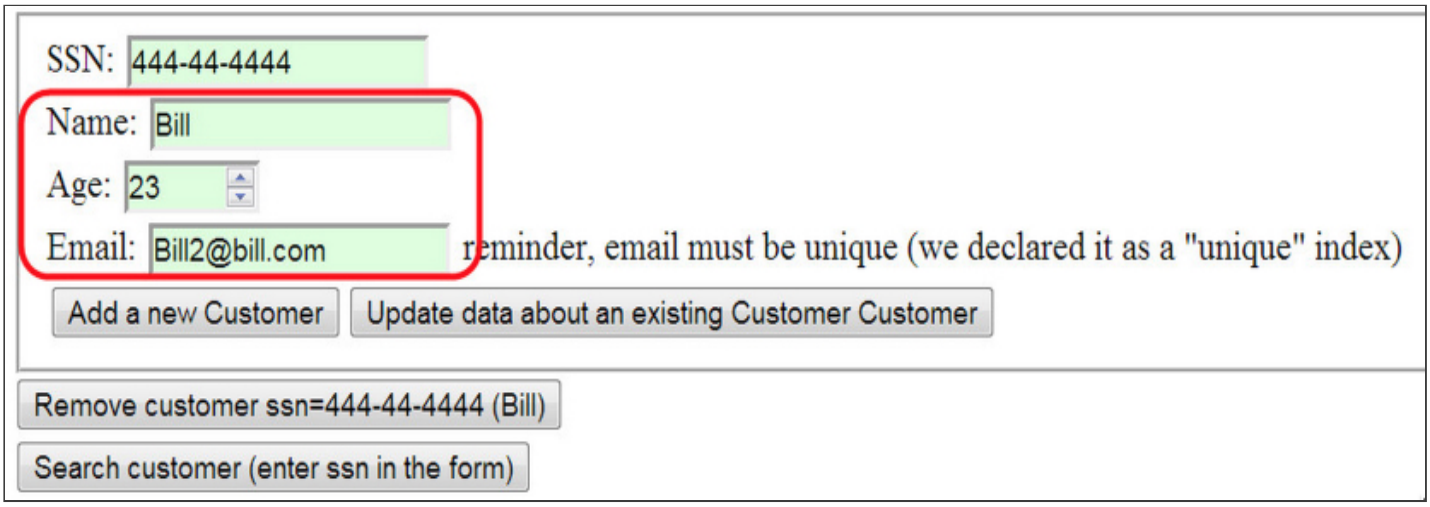
The simplest function from the API is the `request.get(key)` function. It retrieves an object when we know its key/keypath.

Online example at JSBin:



If the `ssn` exists in the object store, then the results are displayed in the form itself (the code that gets the results and that updates the form is in

the `request.onsuccess` callback).



Here is the code added to that example:

```
    function searchACustomer() {
       if(db === null) {
         alert('Database must be opened first, please click the
    Create
                CustomerDB Database first');
         return;
       }

     var transaction =db.transaction(["customers"], "readwrite");
       // Do something when all the data is added to the
    database.
11.    transaction.oncomplete = function(event) {
         console.log("All done!");
       };
       transaction.onerror = function(event) {

     console.log("transaction.onerror" +event.target.errorCode);
       };
       var objectStore =transaction.objectStore("customers");
21.    // Init a customer object with just the ssn property
    initialized
22.    // from the form
       var customerToSearch={};
```

```
 customerToSearch.ssn =document.querySelector("#ssn").value;
   alert('Looking for customer ssn=' +customerToSearch.ssn);
   // Look for the customer corresponding to the ssn in the
object
   // store
   var request =objectStore.get(customerToSearch.ssn);
   request.onsuccess = function(event) {
     console.log("Customer found" +event.target.result.name);

 document.querySelector("#name").value=event.target.result.name;

 document.querySelector("#age").value =event.target.result.age;
     document.querySelector("#email").value

=event.target.result.email;
   };

   request.onerror = function(event) {
     alert("request.onerror, could not find customer, errcode
= " +              event.target.errorCode + ".
            The ssn is not in the Database");
  };
}
```

The search is done in *line 30*, and the callback in the case of success is `request.onsuccess`, *lines 32-38*. `event.target.result` is the resulting object (*lines 33 to 36*).

Well, this is a lot of code isn't it? We can do a much shorter version of this function (though, admittedly it we won't take care of all possible errors). Here is the shortened version:

```
function searchACustomerShort() {
  db.transaction("customers").objectStore("customers")
 .get(document.querySelector("#ssn").value).onsuccess =
   function(event) {
```

```
      document.querySelector("#name").value =

event.target.result.name;
      document.querySelector("#age").value =

event.target.result.age;
      document.querySelector("#email").value=

event.target.result.email;
  }; // end of onsuccess callback
}
```

You can try on JSBin this version of the online example that uses this shortened version (the function is at the end of the JavaScript code):

```
function searchACustomerShort() {
   if(db === null) {
      alert('Database must be opened first, please click the
Create
            CustomerDB Database first');
      return;
   }
   db.transaction("customers").objectStore("customers")
     .get(document.querySelector("#ssn").value)
     .onsuccess =
       function(event) {
          document.querySelector("#name").value =

 event.target.result.name;
          document.querySelector("#age").value =
                                      event.target.result.age;
          document.querySelector("#email").value =

 event.target.result.email;
       };
}
```

Explanations:

- Since there's only one object store, you can avoid passing a list of object stores that you need in your transaction and just pass the name as a string (*line 8*),
- We are only reading from the database, so we don't need a "readwrite" transaction. Calling transaction() with no mode specified gives a "`readonly`" transaction (*line 8*),
- We don't actually save the request object to a variable. Since the DOM event has the request as its target we can use the event to get to the result property (*line 9*).

SECOND METHOD: GETTING MORE THAN ONE PIECE OF DATA

Getting all data in the datastore: using a cursor

Using `get()` requires that you know which key you want to retrieve. If you want to step through all the values in your object store, or just between a certain range, then you can use *a cursor*.

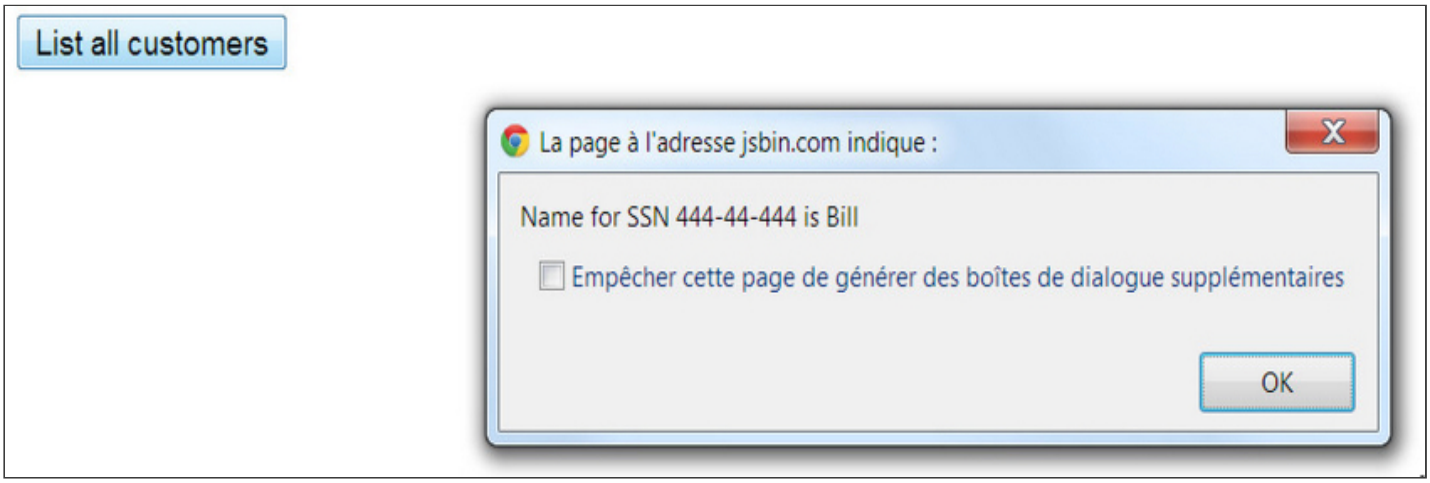
Here's what it looks like:

```
function listAllCustomers() {
   var objectStore =
     db.transaction("customers").objectStore("customers");
   objectStore.openCursor().onsuccess =function(event) {
     // we enter this callback for each object in the store
     // The result is the cursor itself
     var cursor = event.target.result;
11.  if (cursor) {
       alert("Name for SSN " + cursor.key + " is " +
             cursor.value.name);
       // Calling continue on the cursor will result in this
callback
       // being called again if there are other objects in
the store
       cursor.continue();
     } else {
       alert("No more entries!");
     }
```

```
  }; // end of onsuccess...
} // end of listAllCustomers()
```

You can try this example on JSBin.

It adds a button to our application. Clicking on it will display a set of alerts, each showing details of an object in the object store:



The `openCursor()` function takes several arguments.

- First, you can limit the range of items that are retrieved by using a key range object that we'll get to in a minute.
- Second, you can specify the direction that you want to iterate.

In the above example, we're iterating over all objects in ascending order.
The `onsuccess` callback for cursors is a little special. **The cursor object itself is the `result` property of the request** (above we're using the shorthand, so it's `event.target.result`). Then **the actual key and value can be found on the key and value properties of the cursor object**. If you want to keep going, then you have to call `cursor.continue()` on the cursor.

When you've reached the end of the data (or if there were no entries that matched your `openCursor()` request) you still get a `success`callback, but the `result` property is undefined.

One common pattern with cursors is to retrieve all objects in an object store and add them to an array, like this:

```
function listAllCustomersArray() {
  var objectStore =
      db.transaction("customers").objectStore("customers");
  var customers = []; // the array of customers that will
hold
                      // results
  objectStore.openCursor().onsuccess =function(event) {
    var cursor = event.target.result;
    if (cursor) {
      customers.push(cursor.value); // add a customer in the
                                    // array
      cursor.continue();
    } else {
      alert("Got all customers: " +customers);
    }
 }; // end of onsucess
} // end of listAllCustomersArray()
```

You can try this version on JSBin.

Getting data using an index

Storing customer data using the `ssn` as a key is logical since the `ssn`uniquely identifies an individual. If you need to look up a customer by `name`, however, you'll need to iterate over every `ssn` in the database until you find the right one.

Searching in this fashion would be very slow, so instead you can use*an index*.

Remember that we added two indexes in our data store:

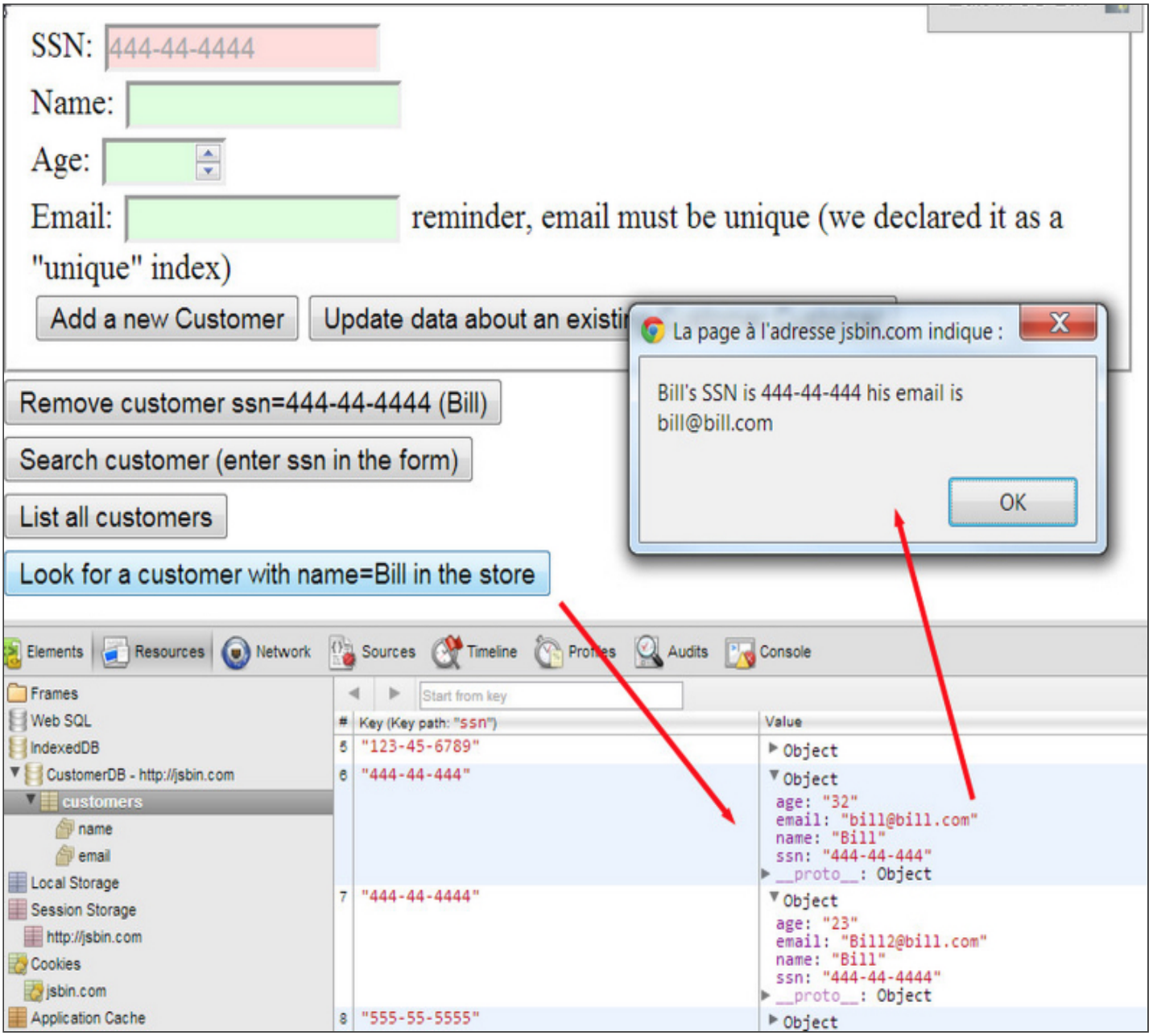
1. one on the `name` (non unique) and
2. one on the `email` properties (unique).

Here is a function that lists by `name` the objects in the object store and returns the first one it finds with a name equal to "Bill":

```
function getCustomerByName() {
   if(db === null) {
     alert('Database must be opened first, please click the
Create
            CustomerDB Database first');
     return;
   }
   var objectStore =
      db.transaction("customers").objectStore("customers");
   var index = objectStore.index("name");
12.
   index.get("Bill").onsuccess =function(event) {
      alert("Bill's SSN is " +event.target.result.ssn +
            " his email is " +event.target.result.email);
   };
}
```

The search by index occurs at lines 11 and 13: line 11 gets back an "index" object that corresponds to the index on the "`name`" property. Line 13 calls the `get()` method on this object to retrieve all objects that have a name equal to "Bill" in the dataStore.

Online example you can try at JsBin



The above example retrieves only the first object that has a name/index with the value="Bill". Notice that there are two "Bill"s in the object store.

Getting more than one result using an index

In order to get all the "Bills", we again have to use *a cursor*.

When we work with indexes, we can open two different types of cursors on indexes:

1. **A normal cursor** that maps the index property to the object in the object store, or,

2. **A key cursor** that maps the index property to the key used to store the object in the object store.

The differences are illustrated below.

Normal cursor:

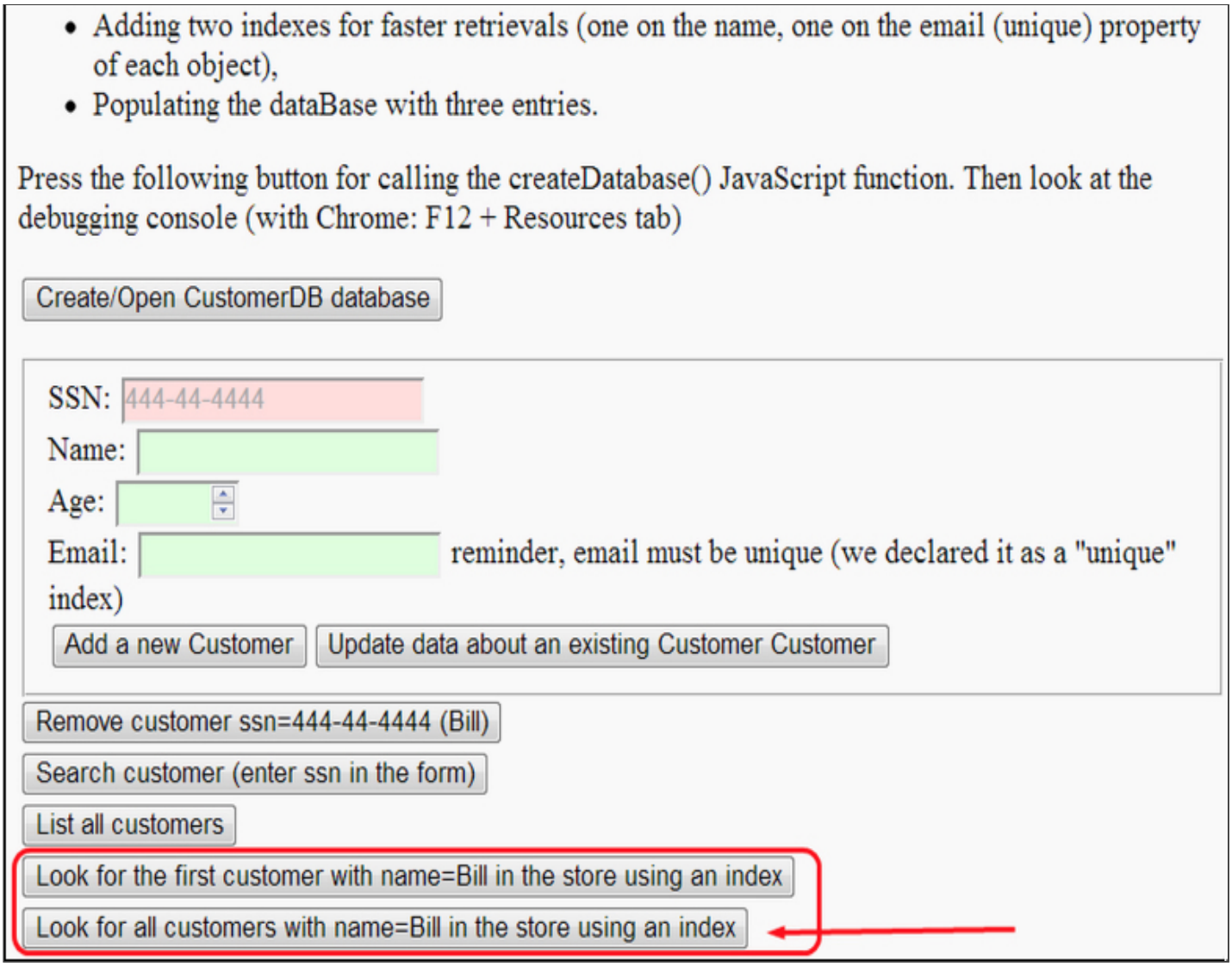
```
index.openCursor().onsuccess =function(event) {
  var cursor = event.target.result;
  if (cursor) {
    // cursor.key is a name, like "Bill", andcursor.value is
the
    // whole object.
    alert("Name: " + cursor.key + ", SSN:
" +cursor.value.ssn + ",
            email: " + cursor.value.email);
    cursor.continue();
  }
};
```

Key cursor:

```
index.openKeyCursor().onsuccess =function(event) {
   var cursor = event.target.result;
   if (cursor) {
     // cursor.key is a name, like "Bill",and cursor.value is
the
     // SSN (the key).
     // No way to directly get the rest of the stored object.
     alert("Name: " + cursor.key + ", "SSN: "
+ cursor.value);
     cursor.continue();
   }
};
```

Can you see the difference?

You can try an online example at JSBin that uses the above methods:



How to try this example:

1. Press the create/Open CustomerDB database,
2. then you may add some customers,
3. then press the last button "look for all customers with name=Bill ...". This will iterate all the customers whose name is equal to "Bill" in the object store. There should be two "Bills", if this is not the case, add two customers with a name equal to "Bill", then press the last button again.

Source code extract from this example:

```
function getAllCustomersByName() {
  if(db === null) {
    alert('Database must be opened first, please click the
Create
           CustomerDB Database first');
    return;
  }
  var objectStore =
     db.transaction("customers").objectStore("customers");
  var index = objectStore.index("name");

  // Only match "Bill"
  var singleKeyRange =IDBKeyRange.only("Bill");
  index.openCursor(singleKeyRange).onsuccess= function(event) {
    var cursor = event.target.result;
    if (cursor) {
      // cursor.key is a name, like "Bill", and cursor.value
is the
      // whole object.
      alert("Name: " + cursor.key + ", SSN:
"+ cursor.value.ssn ",
            + email: " +cursor.value.email);
      cursor.continue();
    }
  };
}
```