WEB STORAGE

Projeto I - TSIW - 17/18

Ricardo Queirós

ESCOLA
SUPERIOR
DE MEDIA
ARTES
E DESIGN

WEB STORAGE

INDEX

WEB STORAGE

1. INTRODUCTION

- **Web Storage** is related with software methods for storing data in a web browser
- Usually associated to **local storage** as an improvement on HTTP cookies
- **Why?**
  - Quicker access
  - Less network traffic
  - Less strain on your server
  - Better browsing experience with fast start-up
  - Better offline support, with no server required

- **Solutions:**
  1. Cookies
  2. Web Storage API
  3. IndexedDB API
  4. Libraries (Lockr, localForage, Dexie and many others)

WEB STORAGE

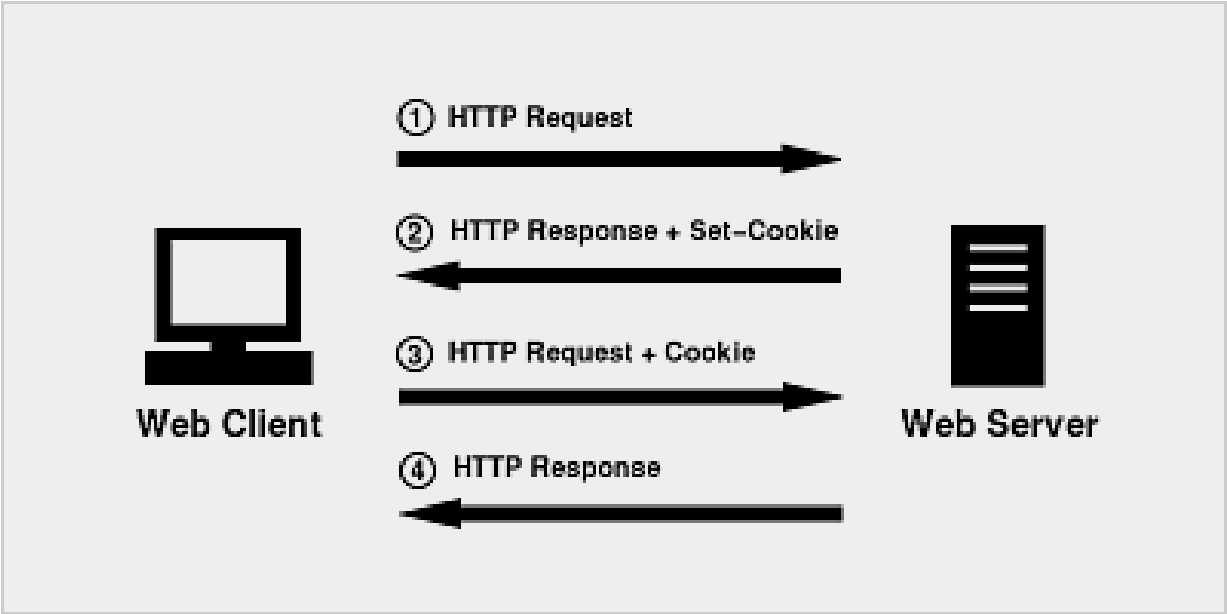
2. COOKIES

- **Small piece of data** typically used to know if two requests came from the same browser allowing, for instance, to keep a user logged-in
- It remembers stateful information for the **stateless HTTP protocol**
- Mainly used for three purposes:
  - Session management (user logins, shopping carts)
  - Personalization (user preferences)
  - Tracking (analyzing user behavior)

WEB STORAGE

2. COOKIES

- **Cookies architecture**



WEB STORAGE

2. COOKIES

- **Cookies example**

1. A client request is made to the server
2. The server send a **Set-Cookie header** with the response

3. The cookie is stored in the browser and, afterwards, the cookie value is sent along with every request made to the same server as the content of a **Cookie HTTP header**

WEB STORAGE

2. COOKIES

- **Cookies example (with headers view)**

1. A client request is made to the server
2. The server send a **Set-Cookie header** with the response

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: session_id=12345

[page content]
```

3. The cookie is stored in the browser and, afterwards, the cookie value is sent along with every request made to the same server as the content of a **Cookie HTTP header**

```
GET /sample_page.html HTTP/1.1
Host: www.example.org
Cookie: session_id=12345
```

WEB STORAGE

2. COOKIES

- Cookies have also been used for general client-side storage
- **Main disadvantages:**
  - Additional performance burden (especially for mobile web)
  - Data-capacity limitations (4kb per cookie/20 cookies per domain)
  - Most of the browsers store cookies in text files in clear text
  - User has the option of disabling cookies
- New APIs to consider for local storage are:
  - Web storage API (localStorage and sessionStorage)
  - IndexedDB API

WEB STORAGE

3. WEB STORAGE API

- **Web Storage**
  - API for persistent data storage of **key-value pair data** in Web clients
  - W3C recommendation (19 April 2016)
- Unlike cookies:
  - **Storage size**: the storage limit is far larger (at least 5MB)
  - **Client side interface**: information is never transferred to the server
  - **Local and Session storage**: 2 different storage areas: local and session storage, which differ in scope and lifetime
  - **Interface data model:** better programmatic interface (associative array data model with keys/values both strings)

WEB STORAGE

3. WEB STORAGE API

- **Local storage**
  - Is per origin (the combination of protocol, hostname, and port number)
  - All pages, from one origin, can store and access the same data
  - The data persists after the browser is closed with no expiration date
  - Object: **window.localStorage**

- **Session storage**
  - Is per-origin-per-window-or-tab
  - Is limited to the lifetime of the window
  - Data is lost when the browser tab is closed)
  - Object: **window.sessionStorage**

WEB STORAGE

3. WEB STORAGE API

- Check browser support: https://caniuse.com

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			63						
			64		10.3				
	16	58	65	11	11.2				4
11	17	59	66	11.1	11.3	all	66	11.8	6.2
	18	60	67	TP					
		61	68						
			69						

WEB STORAGE

3. WEB STORAGE API

- Check support programmatically:

```
if (typeof(Storage) !== "undefined") {
   // Code for localStorage/sessionStorage
} else {
   // Sorry! No Web Storage support...
}
```

WEB STORAGE

3. WEB STORAGE API

- Store/retrieve values

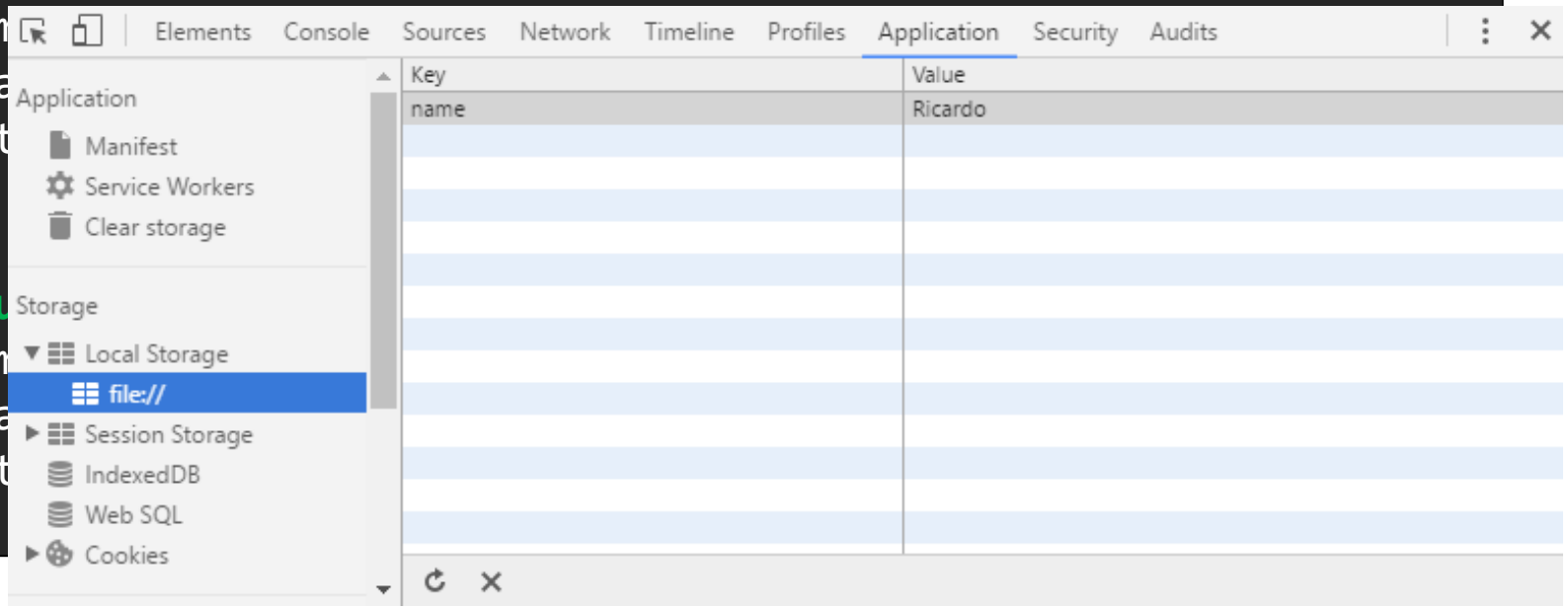
```
// Store value on the browser beyond the duration of the session
localStorage.name = “Ricardo”
localStorage[“name”] = “Ricardo”
localStorage.setItem(“name”, “Ricardo”)


// Retrieve value (persists even after closing and re-opening the browser)
localStorage.name
localStorage[“name”]
localStorage.getItem(“name”)
```

WEB STORAGE

3. WEB STORAGE API

- Store/retrieve values

```
// Store value on the browser beyond the duration of the session
localStorage.nam
localStorage["na
localStorage.set


// Retrieve valu
localStorage.nam
localStorage["na
localStorage.get
```



WEB STORAGE

3. WEB STORAGE API

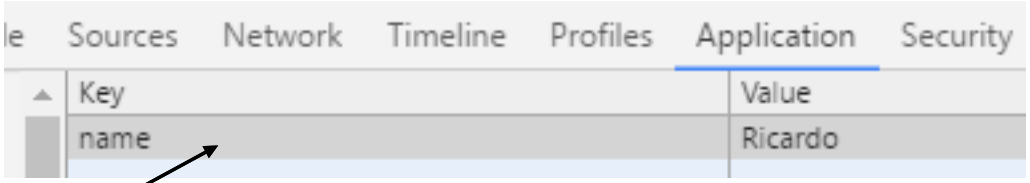
- Iterate over the Local Storage

```
for (let i = 0; i < localStorage.length; i++) {
 ...
}
```

WEB STORAGE

3. WEB STORAGE API

- Iterate over the Local Storage




```
for (let i = 0; i < localStorage.length; i++) {
 console.log(localStorage.key(i))
 console.log(localStorage.getItem(localStorage.key(i))
}
-------------------------------------
name
Ricardo
```

WEB STORAGE

3. WEB STORAGE API

- Clean up actions

```
// Removes item with key ‘name’
localStorage.removeItem(“name”)

// Empties the entire storage object
localStorage.clear()
```

WEB STORAGE

3. WEB STORAGE API

- Only **strings can be stored** via the Storage API
- Attempting to store a different data type will result in an automatic conversion into a string!

```
// Store an integer instead of a string
localStorage.myKey = 1
console.log(typeof localStorage.myKey) // string
// Store an object instead of a string
localStorage.myKey = {name: “Ricardo”}
console.log(typeof localStorage.myKey) // string
```

WEB STORAGE

3. WEB STORAGE API

- Conversion into **JSON** (JavaScript Object Notation)
- Methods:
  - Stingify - Convert a JavaScript object into a string with **JSON.stringify()**
  - Parse - Parse the data with **JSON.parse(),** and the data becomes a JavaScript object

WEB STORAGE

3. WEB STORAGE API

- Conversion into **JSON** (JavaScript Object Notation)
- Methods: stringify and parse

```
const myObj = { name: ‘Skip’, age: 2, favoriteFood: ‘Steak’ }

const myObjStr = JSON.stringify(myObj)

console.log(myObjStr)
// "{"name":"Skip","age":2,"favoriteFood":"Steak"}"

console.log(JSON.parse(myObjStr))
// Object {name:"Skip",age:2,favoriteFood:"Steak"}“
```

WEB STORAGE

3. WEB STORAGE API

- Conversion into **JSON** (JavaScript Object Notation)
- Useful for effective storage of **JavaScript objects**
- Methods: stringify and parse

```
// Store an object using JSON
localStorage.myKey = JSON.stringify({name: “Ricardo”})
console.log(typeof localStorage.myKey) // string
console.log(typeof JSON.parse(localStorage.myKey)) // object
myObj = JSON.parse(localStorage.myKey)
console.log(myObj.name) // Ricardo
```

WEB STORAGE

3. WEB STORAGE API

- **Challenges**
  1. Store the name of a selected school when the user press a button
  2. Store the background color of a page selected by user
  3. Set a database movie for users storing their names, favorites titles and scores

WEB STORAGE

3. WEB STORAGE API

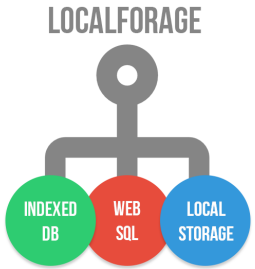
- **Advantages**
  - Supported in almost all the browsers, including iOS and Android. The best part is IE8 onward supports it
  - Very simple, easy to use!

- **Drawbacks**
  - Synchronous
  - Supports only string format
  - Serialization-deserialization is a costly process. Makes things slower
  - Searching is never optimum; may have a visible performance drop in case of large data

WEB STORAGE

4. LIBRARIES

Libraries



WEB STORAGE

4. LIBRARIES

- Foster the use of the API implementations
- Source of the best JavaScript libraries, frameworks, and plugins

https://www.javascripting.com/

- Storage libraries
  - Usually wrappers for the previous studied APIs
  - More popular: Lockr, Local Forage and Dexie

WEB STORAGE

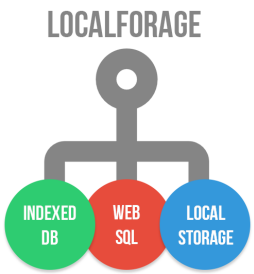
4. LIBRARIES

- **Lockr**
  - A wrapper for **LocalStorage**
  - Redis-like API
  - 2.5 kb
  - Free, Open Source
  - https://github.com/tsironis/lockr

```
<script src="/path/to/lockr.js" type="text/javascript"></script>
...

// Set data
Lockr.set('users', [{name: 'John Doe', age: 18}, {name: 'Jane Doe', age: 19}])

// Get data
Lockr.get('users')

// Return all saved values & objects in a Array
Lockr.getAll()

// Adds a unique value to a particular set under a hash key
Lockr.sadd("wat", 1)
Lockr.sadd("wat", 2)
Lockr.sadd("wat", 1)
Lockr.smembers("wat") // [1, 2]
```

WEB STORAGE

4. LIBRARIES



- **LocalForage**
  - A wrapper for client side storage
  - Uses **IDB, WebSQL** and **LocalStorage**
  - Async/Promise based API
  - Free, Open Source
  - https://mozilla.github.io/localForage

```
<script src="localforage.js"></script>

localforage.iterate(function(value, key, iterationNumber) {
 console.log([key, value])
}).then(function() {
   console.log('Iteration has completed')
}).catch(function(err) {
   // This code runs if there were any errors
   console.log(err)
});
```

WEB STORAGE

4. LIBRARIES

- **Dexie**
  - A wrapper for **IDB**
  - Much simpler API
  - Only ~16k minified and gzipped
  - Promise compatible
  - Free, Open Source
  - http://www.dexie.org

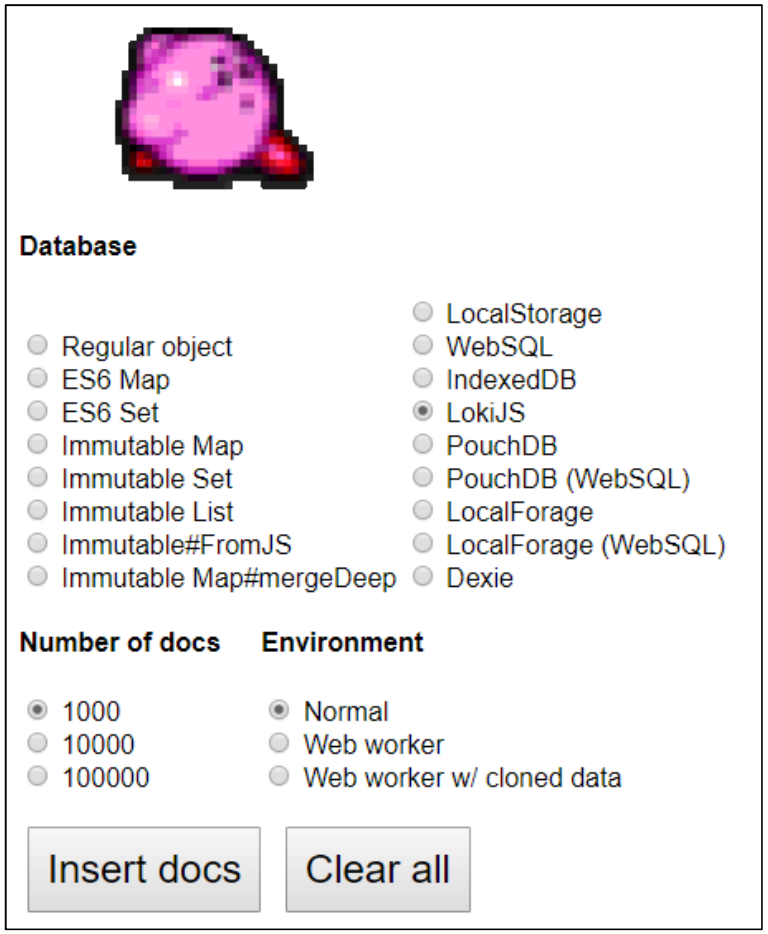
```
// Make a database connection
var db = new Dexie('MyDatabase')

// Define a schema
db.version(1).stores({friends: 'name, age' })

// Open the database
db.open().catch(function(error) {
 alert(error)
})
```

```
// Run some queries
db.friends
 .where('age')
 .above(75)
 .each (function (friend) {
   console.log (friend.name)
})

// or add new friends
db.friends.add({
 name: 'Camilla',
 age: 25
})
```

WEB STORAGE

4. LIBRARIES

- **Browser database comparison**
  - http://nolanlawson.github.io/database-comparison/



WEB STORAGE

REFERENCES

References

- W3C: https://www.w3.org/TR/webstorage/
- Specification: https://html.spec.whatwg.org/multipage/webstorage.html
- MDN - Web Storage: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API
- W3Schools: http://www.w3schools.com/

WEB STORAGE

QUESTIONS?

ricardoqueiros@esmad.ipp.pt