Web Data Storage

...

Seminário TSIW

Web Data Storage

1. Introduction
2. Storage
    a. Cookies
    b. Web Storage API

Web Data Storage

1. Introduction

Web Data Storage

1. Introduction

- Web Data Storage is related with software methods for storing data in a web browser
- Usually associated to local storage as an improvement on HTTP cookies

Web Data Storage

2. Local Storage

Web Data Storage

2. Local Storage

- Data storage in the Web browser's computer
- 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:
  - Cookies
  - Web Storage API
  - IndexedDB API
  - Libraries (mainly wrappers, such as, Lockr, localForage, Dexie and many others)

Web Data Storage

2. Local Storage

Cookies

Web Data Storage

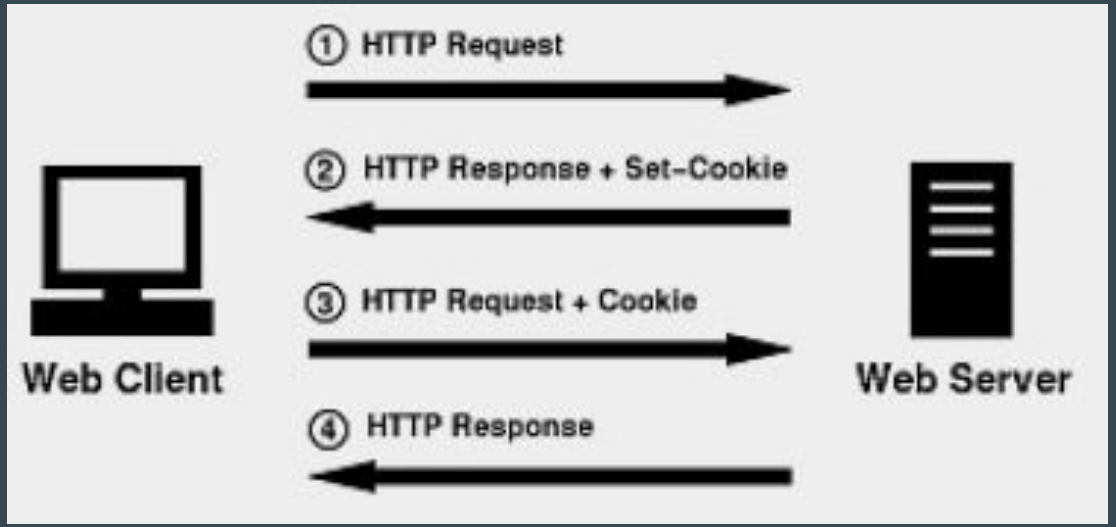
2. Local Storage

- 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 Data Storage

2. Local Storage

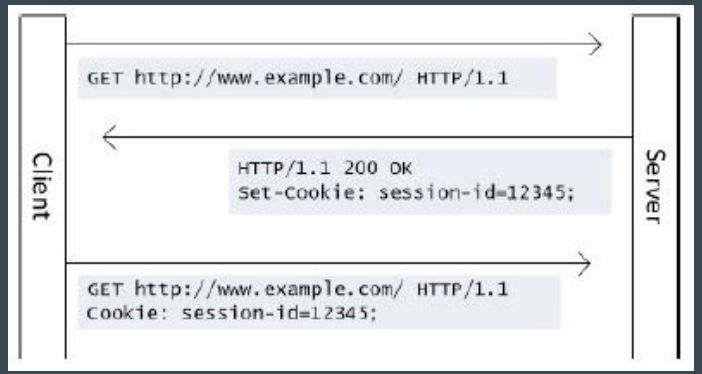
- Cookies
  - Architecture



Web Data Storage

2. Local Storage

- Cookies (example)
  a. A client request is made to the server
  b. The server send a Set-Cookie header with the response
  c. 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 Data Storage

2. Local Storage

- Cookies (example)
  a. A client request is made to the server
  b. 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]
```

  c. 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 Data Storage

2. Local Storage

- 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
    - Search and browsing history can be tracked and privacy is a concern
    - User has the option of disabling cookies
  - New APIs to consider for local storage are:
    - Web storage API (localStorage and sessionStorage)
    - IndexedDB API

Web Data Storage

2. Local Storage

Web Storage API

Web Data Storage

2. Local Storage

- Web Storage API
  - API for persistent data storage of key-value pair data in Web clients
  - W3C recommendation (19 April 2016) - https://www.w3.org/TR/webstorage/
  - Unlike cookies:
    - Storage size: web storage objects are not sent to server with each request. Because of that, we can store much more. 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 Data Storage

2. Local Storage

- Web Storage API
  - The storage is bound to the origin (domain/protocol/port triplet). That is, different protocols or subdomains infer different storage objects, they can’t access data from each other
  - Two different storage objects
    - 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
    - Session storage
      - Is per-origin-per-window-or-tab
      - Is limited to the lifetime of the window

Web Data Storage

2. Local Storage

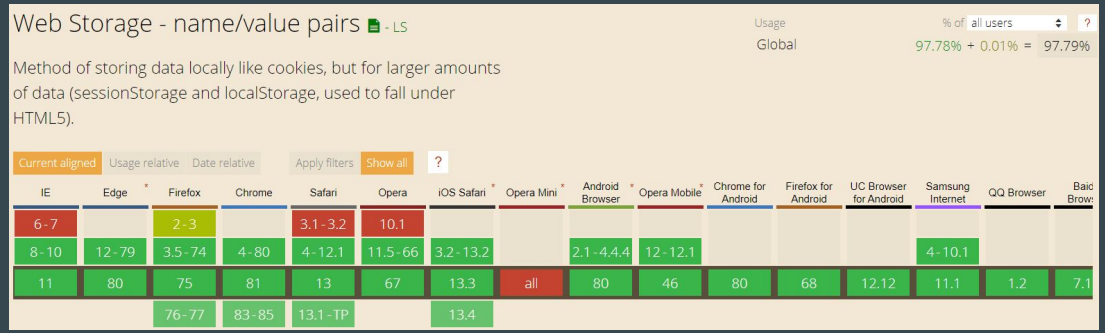
- Web Storage API
  - The storage is bound to the origin (domain/protocol/port triplet). That is, different protocols or subdomains infer different storage objects, they can't access data from each other
  - Two different storage objects
    - Local storage
      - Is per origin (the
      - All pages, from c
      - The data persists
    - Session storage
      - Is per-origin-per-window-or-tab
      - Is limited to the lifetime of the window

`localStorage`	`sessionStorage`
Shared between all tabs and windows with the same origin	Visible within a browser tab, including iframes from the same origin
Survives browser restart	Survives page refresh (but not tab close)

Web Data Storage

2. Local Storage

- Web Storage API
  - The window object has 2 read-only properties to access a Storage object for the Document's origin:
    - window.localStorage - stores data with no expiration date
    - window.sessionStorage - stores data for one session (data is lost when the browser tab is closed)
  - Check support: https://caniuse.com



Web Data Storage

2. Local Storage

- Web Storage API
  - Before using Web Storage API, check browser support for both objects:

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

Web Data Storage

2. Local Storage

- Web Storage API
  - Store data
    - Sets the value of the pair identified by key to value
    - Creates a new key/value pair if none existed for key previously

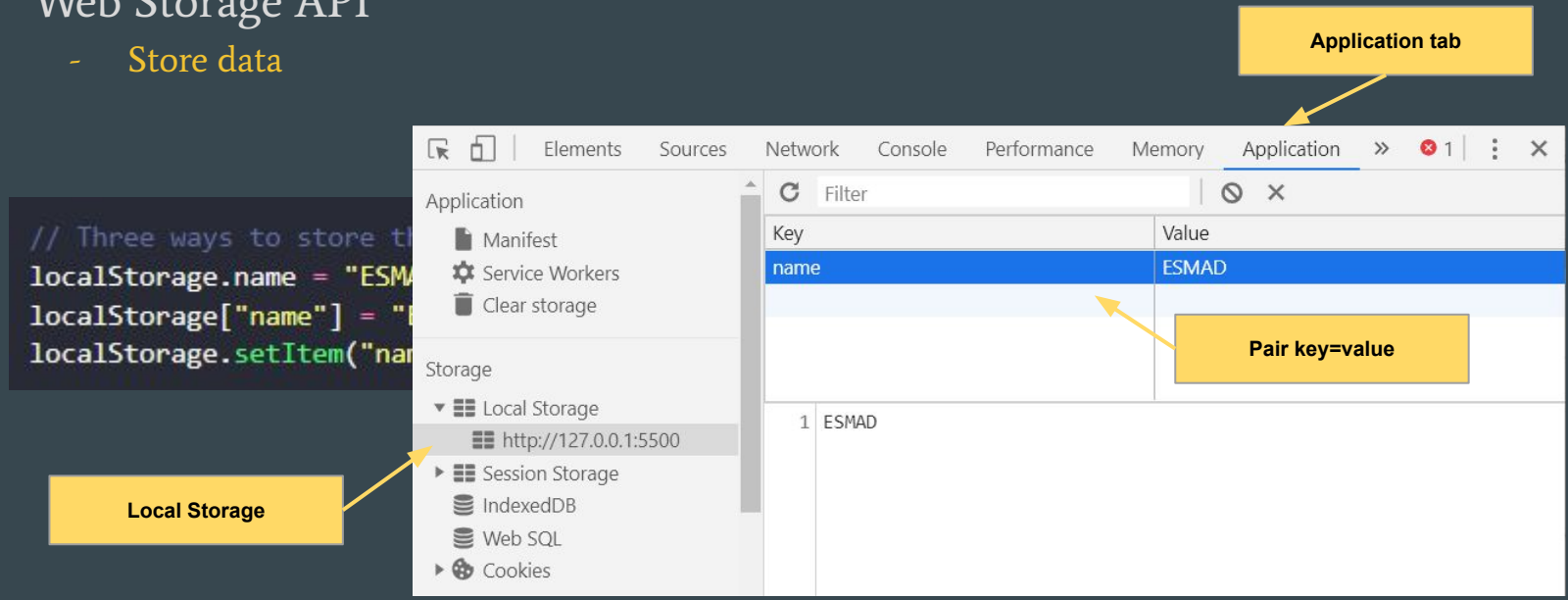
```
// Three ways to store the key "name" with the value "ESMAD"
localStorage.name = "ESMAD"
localStorage["name"] = "ESMAD"
localStorage.setItem("name", "ESMAD")
```

    - The localStorage is shared between all windows with the same origin, so if we set the data in one window, the change becomes visible in another one.
    - Throws a "QuotaExceededError" DOMException exception if the new value couldn't be set. (Setting could fail if, e.g., the user has disabled storage, or if quota has been exceeded.)

Web Data Storage

2. Local Storage

- Web Storage API
  - Store data



Web Data Storage

2. Local Storage

- Web Storage API
  - Get data
    - Returns:
      - the current value (string) associated with the given key
      - null if the given key does not exist in the list associated with the object

```
// Three ways to get the value "ESMAD" using the key "name" from Local Storage
console.log(localStorage.name);
console.log(localStorage["name"]);
console.log(localStorage.getItem("name"));
```

Web Data Storage

2. Local Storage

- Web Storage API
  - Iterate over the Local Storage

  - Property length: to obtain the size of the Local Storage collection
  - Method key(): to return the name of the nth key in the list, or null if n is greater than or equal to the number of key/value pairs in the object

```
// Set values in Local Storage
localStorage.name = "Ricardo"
localStorage.age = "45"
localStorage.country = "Portugal"

// Iterate over the Local Storage collection of key pairs
for (let i = 0; i < localStorage.length; i++) {
    const key =localStorage.key(i)
    console.log(`${key} -> ${localStorage.getItem(key)}`);
}

// CONSOLE
// name -> Ricardo
// age -> 45
// country -> Portugal
```

Local Storage size

Get the key given its position in Local Storage

Web Data Storage

2. Local Storage

- Web Storage API
  - Iterate over the Local Storage

  - Property length: to obtain the size of the Local Storage collection
  - Method key(): to return the name of the nth key in the list, or null if n is greater than or equal to the number of key/value pairs in the objec

```
// Set values in Local Storage
localStorage.name = "Ricardo"
localStorage.age = "45"
localStorage.country = "Portugal"

// Iterate over the Local Storage collection of key pairs
for (let i = 0; i < localStorage.length; i++) {
    const key =localStorage.key(i)
    console.log(`${key} -> ${localStorage.getItem(key)}`);
}
```

Local Storage size

key given its
Local Storage

Key	Value
name	Ricardo
age	45
country	Portugal

Web Data Storage

2. Local Storage

- Web Storage API
  - Iterate over the Local Storage
    - ... or just get the “own” keys with Object.keys and then loop over them if needed:

```
localStorage.name = "Ricardo";
localStorage.age = "45";
localStorage.country = "Portugal";

let keys = Object.keys(localStorage);
for (let key of keys) {
  console.log(`${key} -> ${localStorage.getItem(key)}`);
}
```

Web Data Storage

2. Local Storage

- Web Storage API
  - Clean Up actions

    - Method removeItem(“key”): removes the key/value pair with the given key from the Local Storage, if a key/value pair with the given key exists

    - Method clear(): empties the all key/value pairs from Local Storage, if there are any

```
// Remove item with key "age"
localStorage.removeItem("age")

// Remove all the items from the Local Storage
localStorage.clear()
```

Web Data Storage

2. Local Storage

- Web Storage API
  - Storage event

    When the data gets updated in localStorage or sessionStorage, storage event triggers, with properties:
    - key – the key that was changed (null if .clear() is called).
    - oldValue – the old value (null if the key is newly added).
    - newValue – the new value (null if the key is removed).
    - url – the url of the document where the update happened.
    - storageArea – either localStorage or sessionStorage object where the update happened.
  - The event triggers on all window objects where the storage is accessible, except the one that caused it
  - That allows different windows from the same origin to exchange messages!

Web Data Storage

2. Local Storage

- Web Storage API
  - Only strings can be stored in Local Storage
  - Attempting to store a different data type will result in an automatic conversion to string

```
// Store a number instead of a string
localStorage.setItem("year", 2020)
console.log(typeof localStorage.getItem("year"));   // string
```

Web Data Storage

2. Local Storage

- Web Storage API
  - Only strings can be stored in Local Storage
  - Attempting to store a different data type will result in an automatic conversion to string

```
// Store an object instead of a string
let friend = {
    name: "Maria",
    age: 33
}

localStorage.setItem("myFriend", friend)
console.log(typeof localStorage.getItem("myFriend"));   // string

friend = localStorage.getItem("myFriend")
console.log(friend);          // "[object Object]"
console.log(friend.name);     // undefined
```

Web Data Storage

2. Local Storage

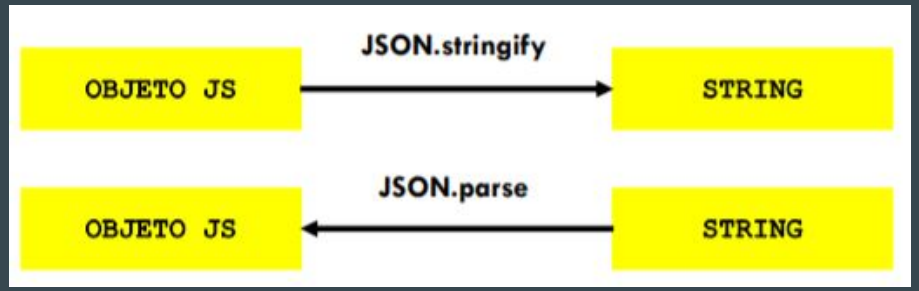
- Web Storage API
  - How can I store an object in LocalStorage and, then, get back the object from LocalStorage?
  - Solution: using the JSON object

  - What is JSON?
    - JavaScript Object Notation
    - Open standard file format
    - Uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value)
    - Replacement for XML in AJAX systems

Web Data Storage

2. Local Storage

- Web Storage API
  - How can I store an object in LocalStorage and, then, get back the object from LocalStorage?
  - Solution: using 2 methods from JSON object
    - Stringify - takes a JavaScript object and transforms it into a JSON string
    - Parse - takes a JSON string and transforms it into a JavaScript object



Web Data Storage

2. Local Storage

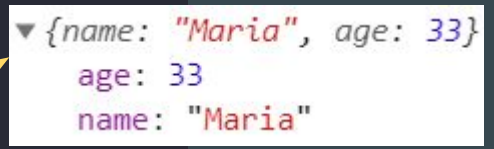
- Web Storage API

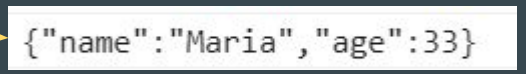
Web Data Storage

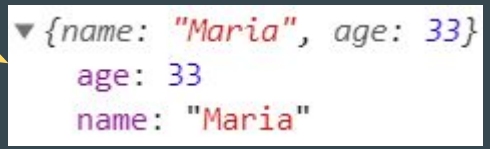
2. Local Storage

- Web Storage API

```
// Set a new object
let friend = {
    name: "Maria",
    age: 33
}
console.log(friend);

// Stringify the object
const friendStr = JSON.stringify(friend)
console.log(friendStr);

console.log(JSON.parse(friendStr));
```



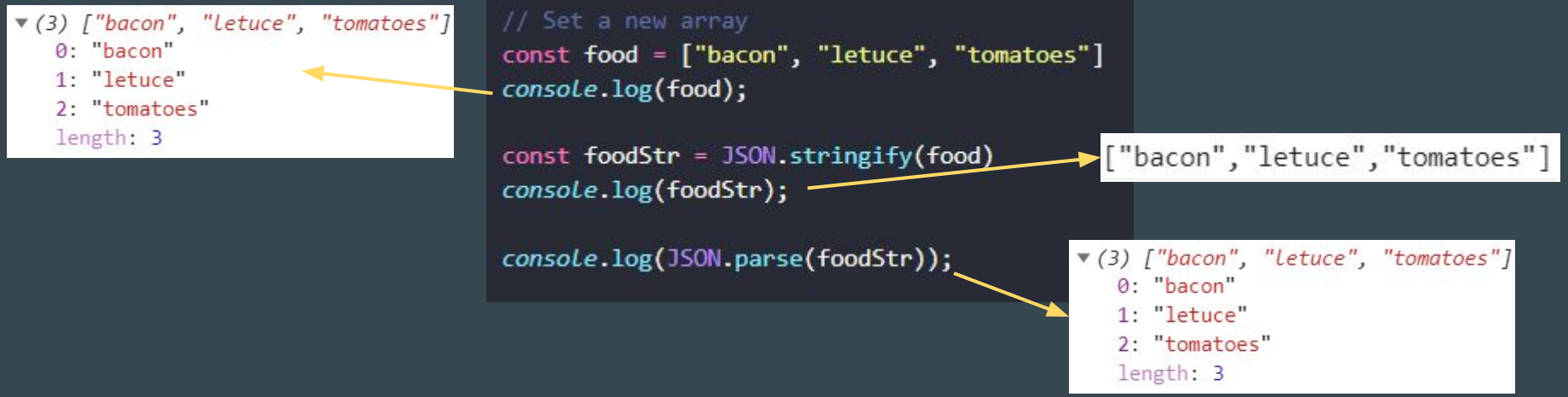




Web Data Storage

2. Local Storage

- Web Storage API
  - And although the methods are usually used on objects, they can also be used on arrays
  - Since an array is a special object

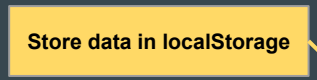


Web Data Storage

2. Local Storage

- Web Storage API
  - Example (store data)

```
// Class creation
class User {
  constructor(name, login, pass) {
    this.name = name;
    this.login = login;
    this.pass = pass;
  }
}

// Array creation
const users = [];

// Objects creation
const user1 = new User("Ricardo", "ricky", "12345");
const user2 = new User("Maria", "mary", "54321");
const user3 = new User("Pedro", "peter", "15243");

// Add objects to array
users.push(user1, user2, user3);

// Storage of the array in the Local Storage
localStorage.setItem("usersList", JSON.stringify(users));
```



Web Data Storage

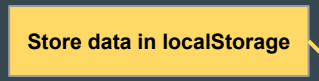
2. Local Storage

- Web Storage API
  - Example (store data)

```
// Class creation
class User {
  constructor(name, login, pass) {
    this.name = name;
    this.login = login;
    this.pass = pass;
  }
}

// Array creation
const users = [];
```

Key	Value
usersList	[{"name":"Ricardo","login":"ricky","pass":"12345"},{"name":"Maria","login":"mary","pass":"54321"},{"name":"Pedro","login":"p...

```
const user3 = new User("Pedro", "peter", "15243");

// Add objects to array
users.push(user1, user2, user3);

// Storage of the array in the Local Storage
localStorage.setItem("usersList", JSON.stringify(users));
```



Web Data Storage

2. Local Storage

- Web Storage API
  - Example (get data)

Get data from localStorage

```
// Array creation
let users = [];

// Verification of a usersList key in the LocalStorage
if (localStorage.getItem("usersList")) {
  // Conversion of the value from string to object (array)
  users = JSON.parse(localStorage.getItem("usersList"));
}

// Show in the console de array users
console.log(users);
```

Web Data Storage

2. Local Storage

- Web Storage API
  - Example (get data)

Get data from localStorage

```
// Array creation
let users = [];

// Verification of a usersList key in the LocalStorage
if (localStorage.getItem("usersList")) {
  // Conversion of the value from string to object (array)
  users = JSON.parse(localStorage.getItem("usersList"));
}

// Show in the console
console.log(users);
```

```
▼ (3) [{…}, {…}, {…}]
  ▶ 0: {name: "Ricardo", login: "ricky", pass: "12345"}
  ▶ 1: {name: "Maria", login: "mary", pass: "54321"}
  ▶ 2: {name: "Pedro", login: "peter", pass: "15243"}
    length: 3
```

Web Data Storage

2. Local Storage

- 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
- Other Solutions
  - IndexedDB API

Web Data Storage

2. Local Storage

- 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 Data Storage

2. Local Storage

- 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, LocalForage and Dexie

Web Data Storage

2. Local Storage

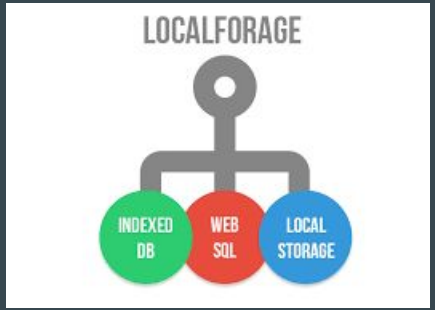
- Libraries (Lockr)
  - A wrapper for LocalStorage
  - Redis-like API
  - 2.5 kb
  - Free, Open Source
  - https://github.com/tsironis/lockr


```
<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 Data Storage

2. Local Storage

- 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 Data Storage

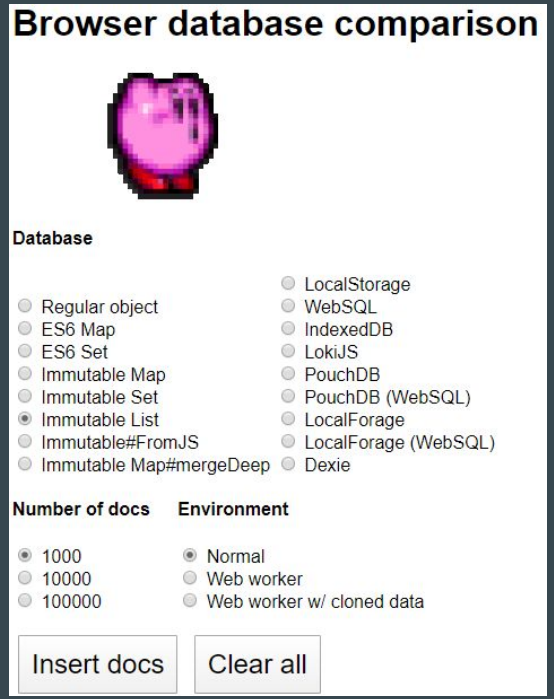
2. Local Storage

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

Web Data Storage

2. Local Storage

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



Web Data Storage

References

- W3Schools: http://www.w3schools.com/
- Mozilla Developer Network: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API
- Alligator: https://alligator.io/js/

Web Data Storage

Questions?

ricardoqueiros@esmad.ipp.pt