Data Storage

Agenda

JavaScript Object Notation

Data serialization

Local data storage

JavaScript Object Notation

JSON abbreviation

Lightweight data-interchange format

Introducing JSON **http://json.org**

JSON Tutorial

JSON valid data types

string

number

object

array

boolean (true,false)

null

JSON document

```
{
  "name": "Adam",
  "age": 23,
  "student": true,
  "interests": [
    "shopping",
    "swimming"
  ]
}
```

```
{
  "students": {
    "student": [
      {
        "name": "Adam",
        "age": 23
      },
      {
        "name": "Alice",
        "age": 18
      }
    ]
  }
}
```

JSON online tools

[JSON Editor]

[JSON viewer]

[JSON formatter]

Data serialization

Process of translating objects / data structures into a format for data storing / transmission

Usage

storing data in databases / disk drives

transferring data through wires

Serialization in JS

Serialization `JSON.stringify(value);`

Deserialization `JSON.parse(text);`

Mozilla Developer Network Description

Local data storage

Web Storage API

- **Local / Session Storage objects**
  - **Window.localStorage; Window.sessionStorage;**
- **Storing key / value pairs**
- **String values only**
- **Methods setItem / getItem for data storing / retrieving**
- **tutorials: w3schools MDN**

Local databases

- Partially supported by web browsers
  - IndexedDB
  - Web SQL
- Open-source / commercial
  - pouchDB
  - LocalForage

To do

Convert JSON

1. **Convert the customers.json document to XML.**
2. **To convert a document, use an online JSON to XML converter.**

Convert to JSON

1. **Convert any XML document to JSON format.**
2. **As an XML document, you can use your university schedule.**
3. **Display the university schedule for your course.**
4. **Next, at the end of the URL, add the string** &xml **to display the schedule in XML format.**
5. **Then, save the xml data to the schedule.xml document.**
6. **Finally, convert the xml file to the JSON format. Compare those two files.**

Create a JSON document

1. **Create a json document for describing a single person with the following information: name, surname, age, gender, email, phone (landline, mobile), address (street and home number, postal code, city), interests (list of interests), family (spouse (name, surname), children list (child name, surname, gender, age, child’s interests))**
2. **To create a document, use an online JSON editor, e.g. http://www.jsoneditoronline.org/**
3. **Format the JSON data with proper indentation.**
4. **Save the document with the name person.json.**

Create a list of objects

1. Based on the person.json document, create a json document containing a list of at least three people.
2. To create a document, use an online JSON editor, e.g. http://www.jsoneditoronline.org/
3. Format the JSON data with proper indentation.
4. Save the document with the name people.json.

Store data

1. **Create the datastorage.html document with a JavaScript code to save your name and surname to the local storage.**
2. **Then run the Chrome DevTools to check the local storage for your personal data.**
3. **In the Chrome DevTools, remove your data from the local storage**

Translate JS object

1. Create the translate.html document, and then create a JavaScript object with personal information. You can use the person.json document created in one of the previous exercises as an example.
2. Translate data structure of the JavaScript object into a string (serialization) and save data to the local storage.
3. Check in the Chrome DevTools whether the object has been saved in the storage.

Store form data

1. **Create the form.html document with two input fields (name, surname) and two buttons (save, restore).**
2. **After clicking on the save button, your program should save the input fields data to the local storage.**
3. **Check in the Chrome DevTools whether the name and surname have been saved in the storage.**
4. **Close and reopen the html document. The input fields should be empty.**
5. **After clicking on the restore button, your program should retrieve data from the local storage and place them in the input fields.**

Log in to an app

1. Create the login.html document with two input fields: Username and Password and a button Log in.
2. By using the Chrome DevTools, add the following data to the local storage: usename=annie, password=ax123
3. Add in the html document a JavaScript code to create a program.
4. After clicking on the button, the program checks whether the username and password are available in the local storage. If not, tries to save the data in the storage.
5. If the username and password are available in the storage, the program tries to compare them with the input fields values. If the data have been typed correctly, the message ‘User logged in successfully’ is displayed in a popup window.

Connect with a local sql database

1. **Familiarise yourself with the use of Web SQL available in the Google Chrome browser:**
   - HTML5 - Web SQL Database
   - Web SQL Database: Creating and Querying Databases
   - Web SQL Database Mini Tutorial
   - Web SQL Database
2. **Then create the database.html document to store data in the Web SQL database.**
3. **Analyse the data contained in the users.sql file. Then create proper database structure do store the list of users.**
4. **Open the Chrome DevTools and check whether the database has been created successfully.**
5. **Display the list of students ordered by the surname and name in the console.**

Check local storage performance

1. Create the teststorage.html document to measure local storage performance.
2. Try to save in the local storage 1000 random integer numbers. How much time does it take? Display the execution time in the console (difference between start and end time in milisec.).
3. Repeat the task for 10000 and 100000 numbers respectively and compare results.

Check sql database performance

1. Create the testsql.html document to measure Web SQL database performance.
2. Create a database table with only one column to store integer numbers.
3. Try to save in the database 1000 random integer numbers. How much time does it take? Display the execution time in the console.
4. Repeat the task for 10000 and 100000 numbers respectively and compare results.