“

Uncovering Browser Storage

Daniel Danielecki

Start

1.

Browser Storage aka. Client Storage

=

Storing application data in the browser

2.

Web Storage?

Web Storage aka. DOM Storage

=

localStorage + sessionStorage

Browser Storage
=
Web Storage + Cookies + IndexedDB

Browser Storage

Server

- Server-side database
- Can be shared with other users
- Important data
- Example: e-commerce products

Browser

- localStorage, sessionStorage, IndexedDB, Cookies
- Can’t be shared with other users
- Temporary data, the data isn’t always available
- Example: caching, session ID, shopping cart items

Types of Browser Storage

localStorage

sessionStorage

IndexedDB

Cookies

localStorage

Window.localStorage

The **`localStorage`** read-only property of the `window` interface allows you to access a `Storage` object for the `Document` 's origin; the stored data is saved across browser sessions.

`localStorage` is similar to `sessionStorage`, except that while `localStorage` data has no expiration time, `sessionStorage` data gets cleared when the page session ends — that is, when the page is closed. (`localStorage` data for a document loaded in a "private browsing" or "incognito" session is cleared when the last "private" tab is closed.)

localStorage

Allows to save key/value pairs in the browser

Can I use localstorage ? Settings

2 results found

☑ Caniuse (1) ☑ MDN (1)

Window API: localStorage

Usage % of all users ?

Global 97.52%

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	K Br
		3.1-3.2	2-3	10.1	6-7											
4-106	12-106	4-16.0	3.5-106	11.5-91	8-10		3.2-16.0	4-18.0		12-12.1		2.1-4.4.4				
107	107	16.1	107	92	11	107	16.1	19.0	all	72	13.4	107	106	13.1	13.18	
108-110		16.2-TP	108-109													

Can I use localstorage ? Settings

2 results found

☑ Caniuse (1) ☑ MDN (1)

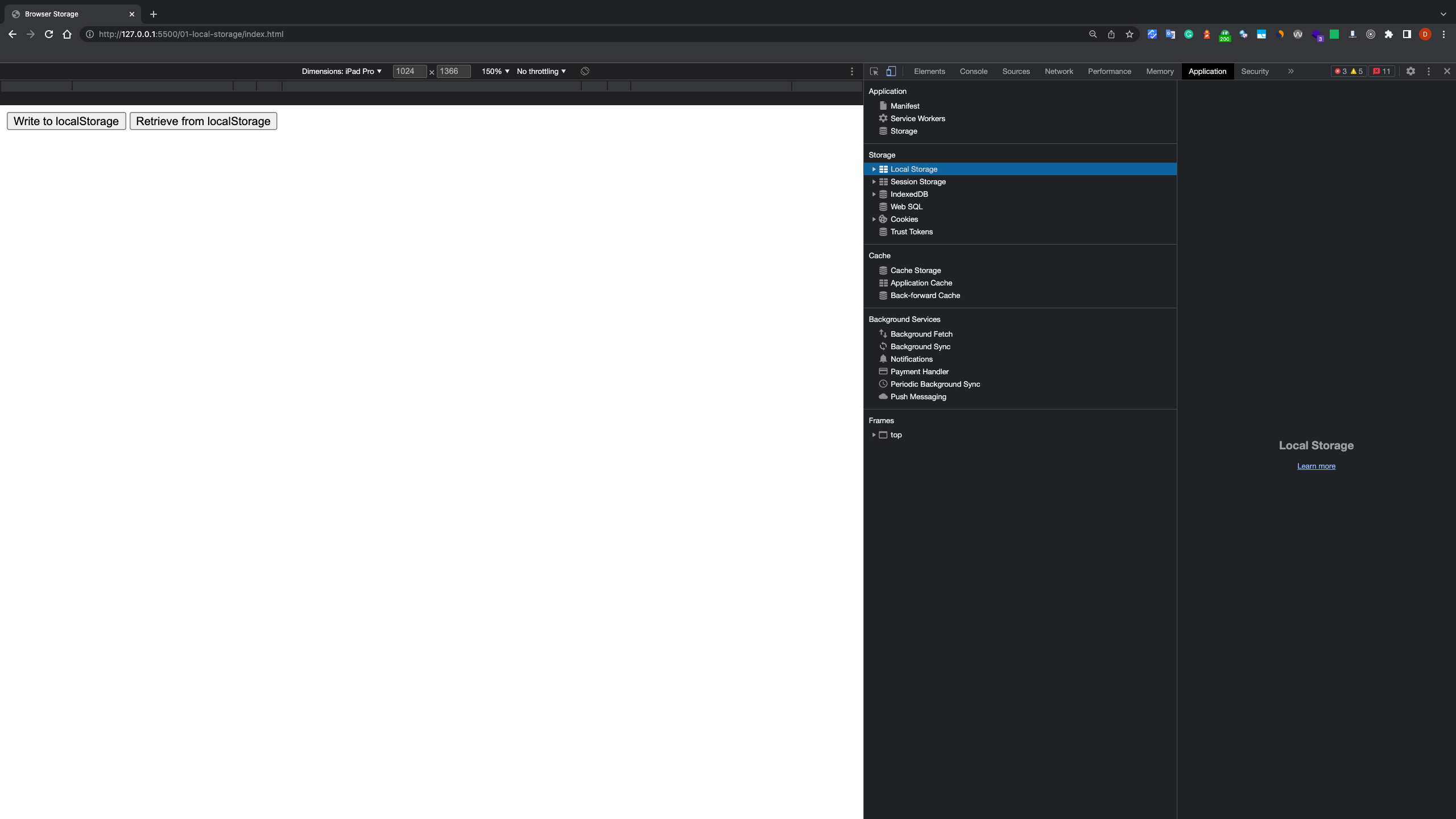
Window API: localStorage

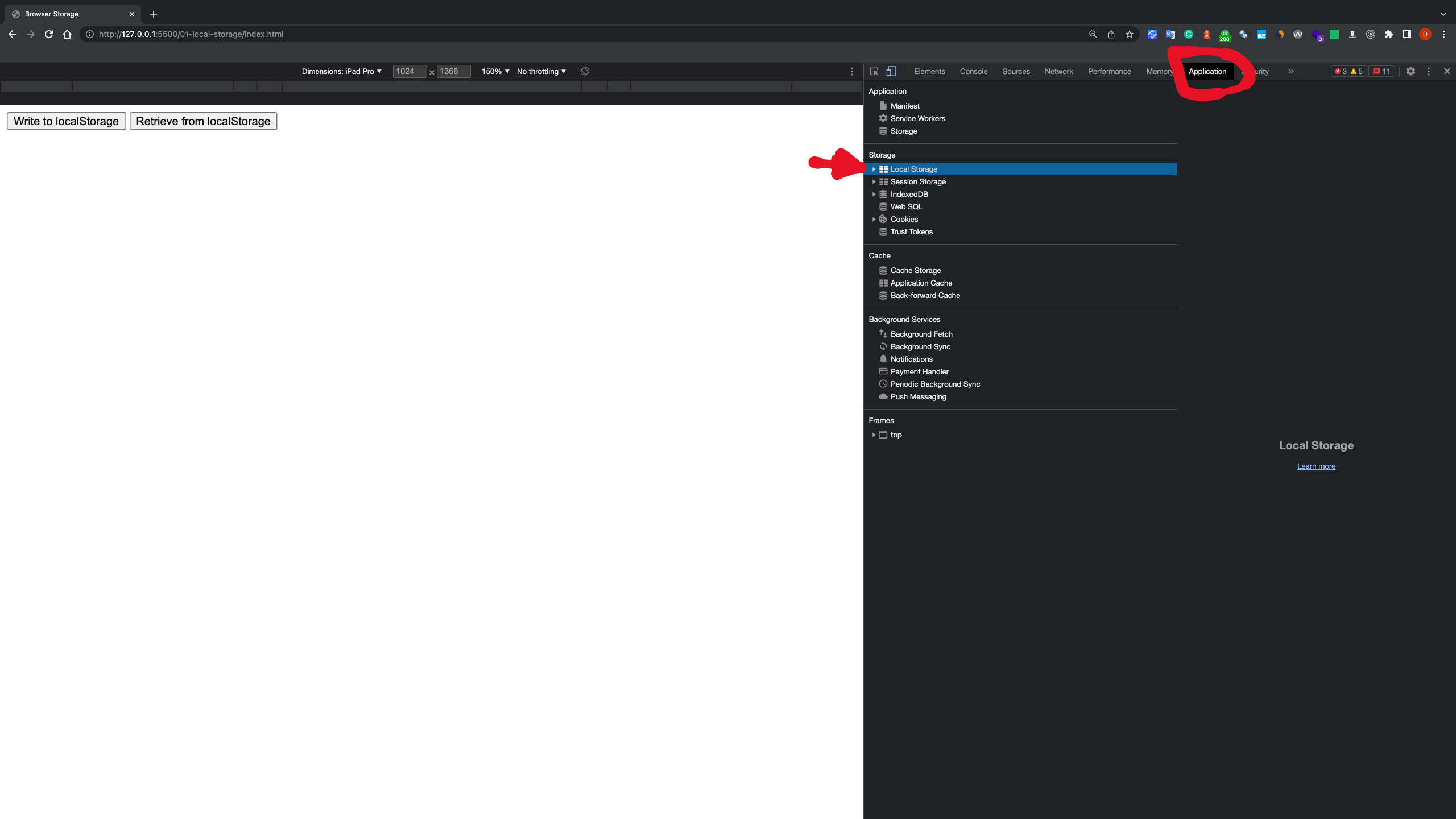
Usage % of all users ?

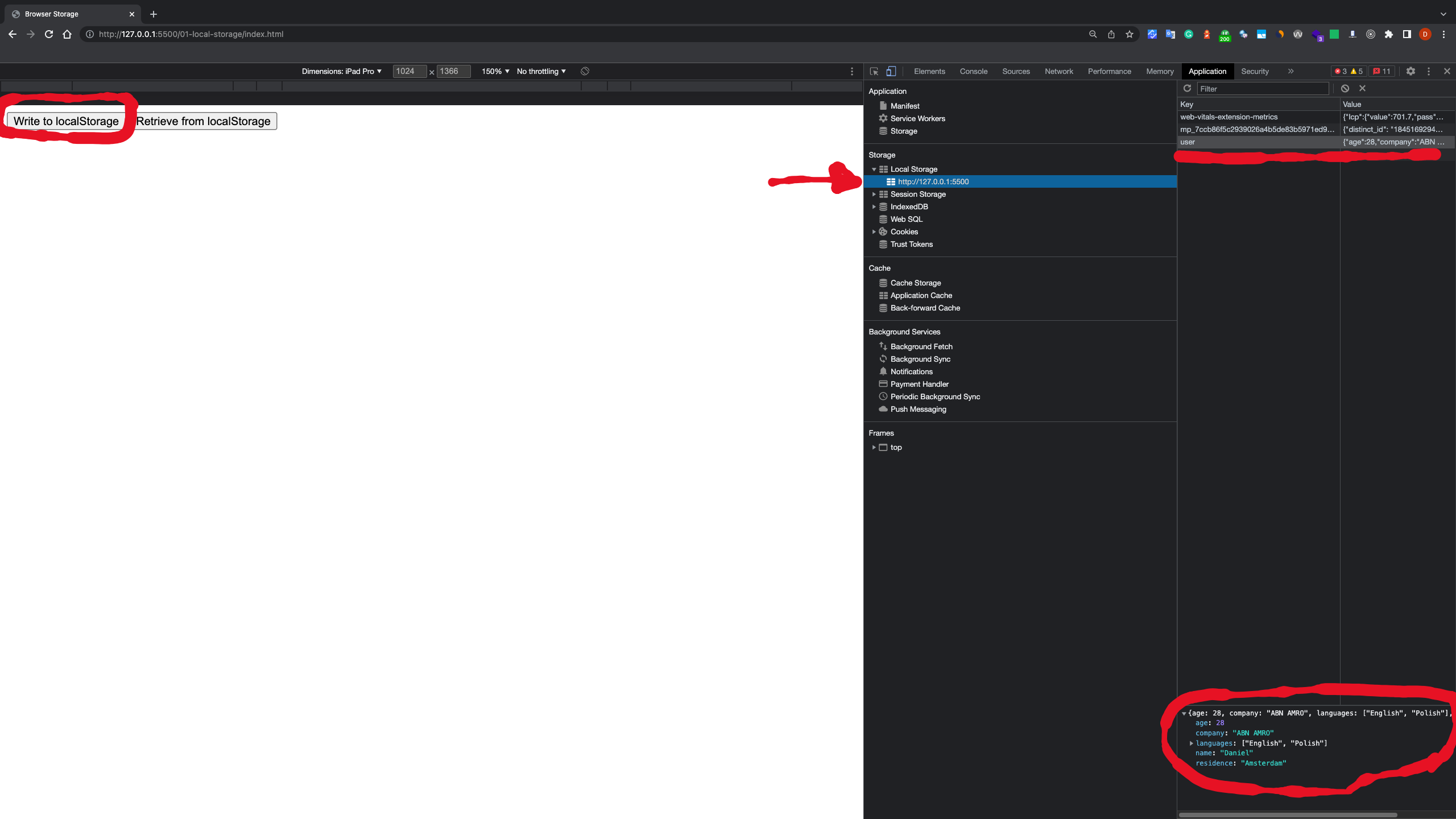
Global 97.52%

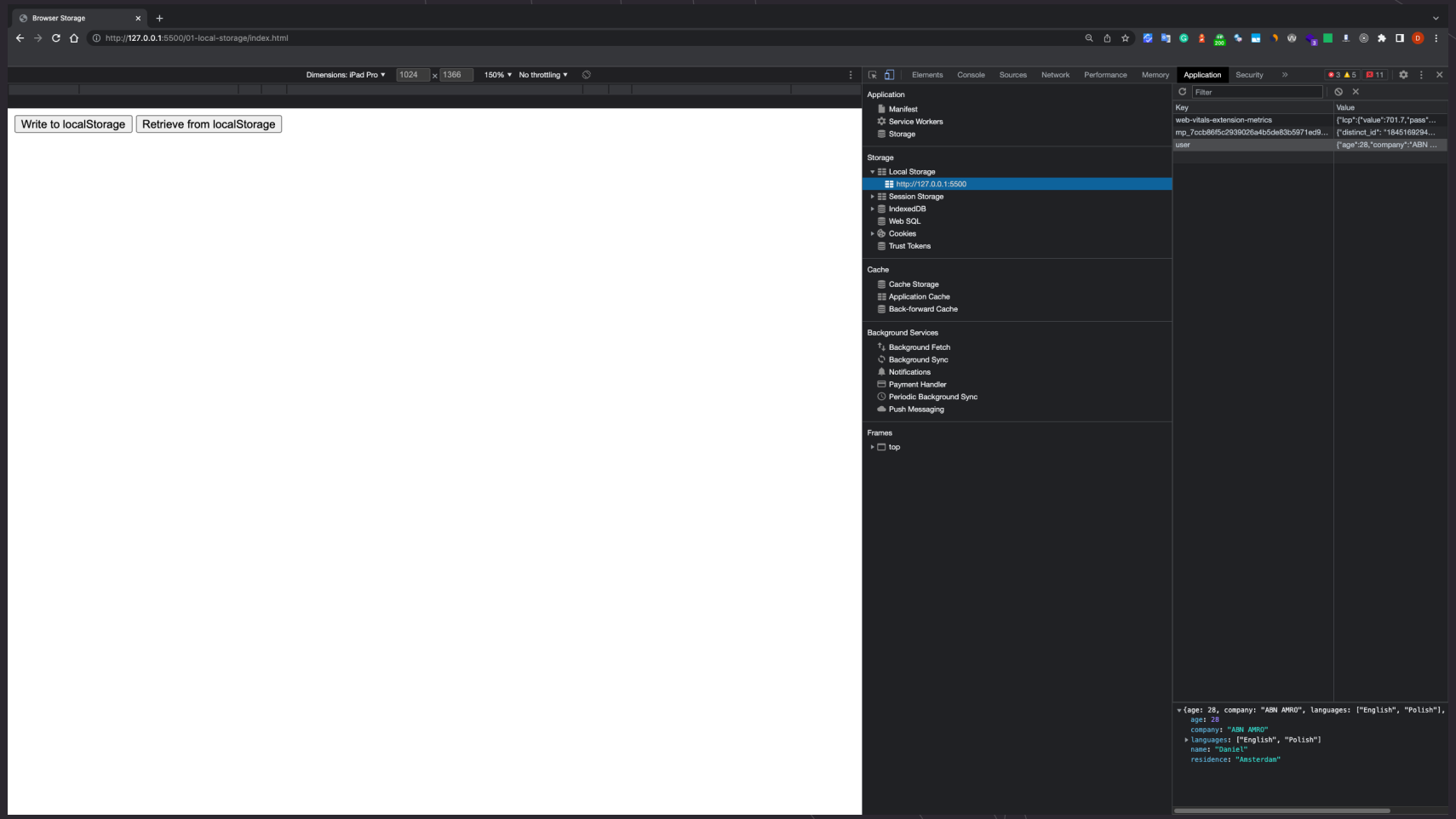
Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	K Br
		3.1-3.2	2-3	10.1	6-7											
4-106	12-106	4-16.0	3.5-106	11.5-91	8-10		3.2-16.0	4-18.0		12-12.1		2.1-4.4.4				
107	107	16.1	107	92	11	107	16.1	19.0	all	72	13.4	107	106	13.1	13.18	
108-110		16.2-TP	108-109													


Browser Storage
http://127.0.0.1:5500/01-local-storage/index.html
Dimensions: iPad Pro
1024
1366
150%
No throttling
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Write to localStorage
Retrieve from localStorage
Application
Manifest
Service Workers
Storage
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
Trust Tokens
Cache
Cache Storage
Application Cache
Back-forward Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
top
Local Storage
Learn more


Browser Storage
http://127.0.0.1:5500/01-local-storage/index.html
Dimensions: iPad Pro
1024
1366
150%
No throttling
Write to localStorage
Retrieve from localStorage
Elements
Console
Sources
Network
Performance
Memory
Application
Application
Manifest
Service Workers
Storage
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
Trust Tokens
Cache
Cache Storage
Application Cache
Back-forward Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
top
Local Storage
Learn more


Browser Storage
http://127.0.0.1:5500/01-local-storage/index.html
Dimensions: iPad Pro
1024
1366
150%
No throttling
Write to localStorage
Retrieve from localStorage
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Application
Manifest
Service Workers
Storage
Storage
Local Storage
http://127.0.0.1:5500
Session Storage
IndexedDB
Web SQL
Cookies
Trust Tokens
Cache
Cache Storage
Application Cache
Back-forward Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
top
Filter
Key
Value
web-vitals-extension-metrics
{"lcp":{"value":701.7,"pass"...
mp_7ccb86f5c2939026a4b5de83b5971ed9...
{"distinct_id": "1845169294...
user
{"age":28,"company":"ABN ...
{age: 28, company: "ABN AMRO", languages: ["English", "Polish"],
age: 28
company: "ABN AMRO"
languages: ["English", "Polish"]
name: "Daniel"
residence: "Amsterdam"



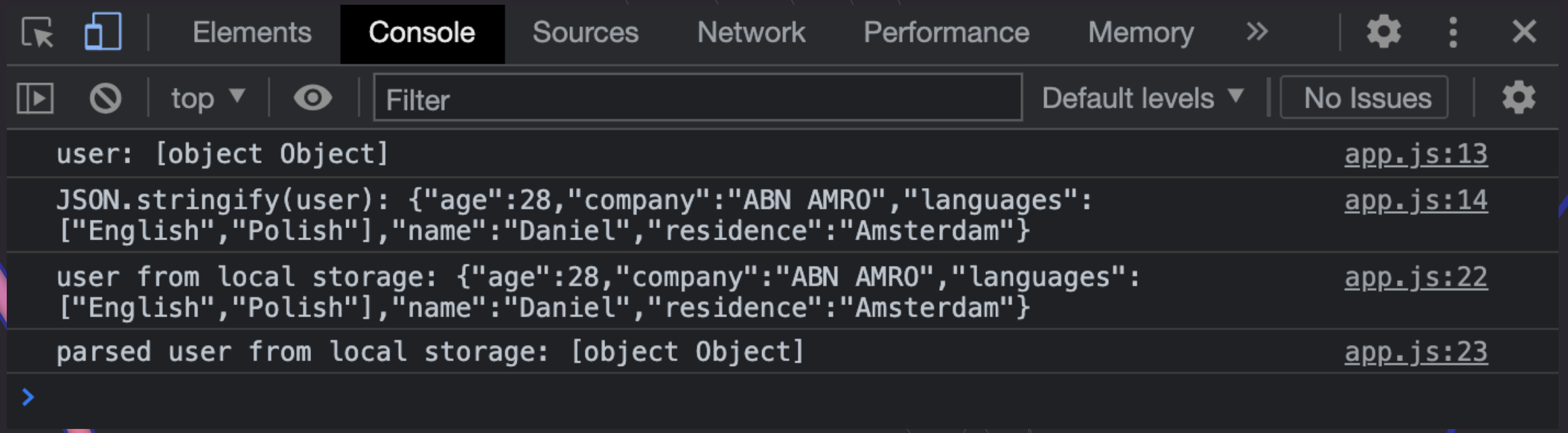
localStorage demo

Live Server & http://127.0.0.1:5500/01-local-storage/index.html

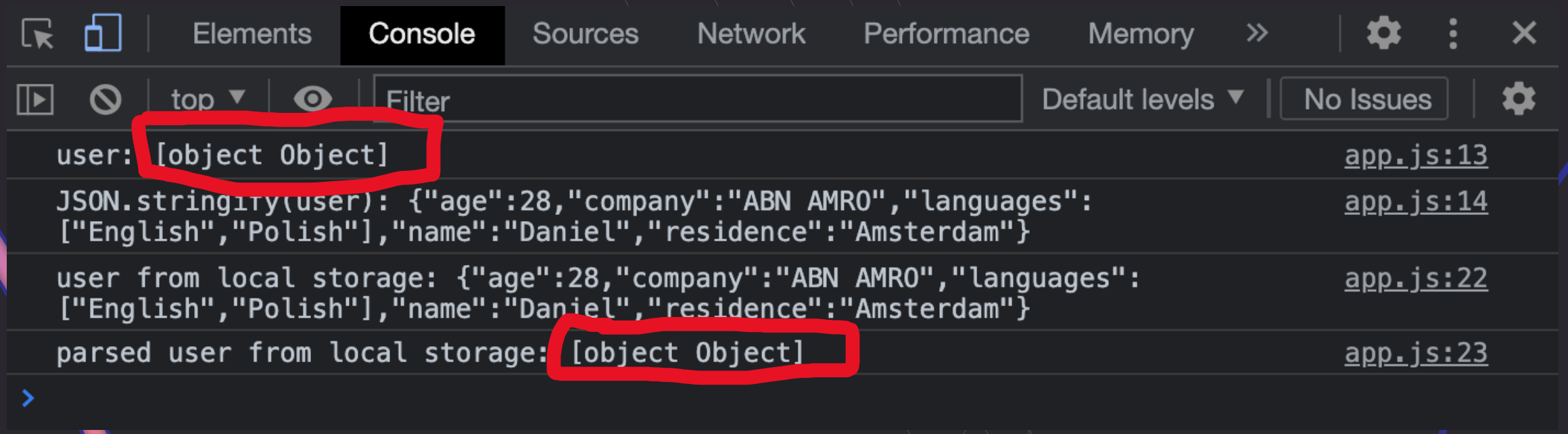
```
1  const writeToButton = document.getElementById('write-to-button');
2  const retrieveFromButton = document.getElementById('retrieve-from-button');
3
4  const user = {
5    age: 28,
6    company: 'ABN AMRO',
7    languages: ['English', 'Polish'],
8    name: 'Daniel',
9    residence: 'Amsterdam'
10 };
11
12 writeToButton.addEventListener('click', () => {
13   console.log('user: ' + user);
14   console.log('JSON.stringify(user): ' + JSON.stringify(user));
15   localStorage.setItem('user', JSON.stringify(user));
16 });
17
18 retrieveFromButton.addEventListener('click', () => {
19   const userFromStorage = localStorage.getItem('user');
20   const parsedUserFromStorage = JSON.parse(localStorage.getItem('user'));
21
22   console.log('user from local storage: ' + userFromStorage);
23   console.log('parsed user from local storage: ' + parsedUserFromStorage);
24 });
25
```

```
1  const writeToButton = document.getElementById('write-to-button');
2  const retrieveFromButton = document.getElementById('retrieve-from-button');
3
4  const user = {
5    age: 28,
6    company: 'ABN AMRO',
7    languages: ['English', 'Polish'],
8    name: 'Daniel',
9    residence: 'Amsterdam'
10 };
11
12 writeToButton.addEventListener('click', () => {
13   console.log('user: ' + user);
14   console.log('JSON.stringify(user): ' + JSON.stringify(user));
15   localStorage.setItem('user', JSON.stringify(user));
16 });
17
18 retrieveFromButton.addEventListener('click', () => {
19   const userFromStorage = localStorage.getItem('user');
20   const parsedUserFromStorage = JSON.parse(localStorage.getItem('user'));
21
22   console.log('user from local storage: ' + userFromStorage);
23   console.log('parsed user from local storage: ' + parsedUserFromStorage);
24 });
25
```

JSON.stringify()/JSON.parse() ???



JSON.stringify()/JSON.parse() ???



sessionStorage

Window.sessionStorage

The read-only `sessionStorage` property accesses a session `Storage` object for the current origin. `sessionStorage` is similar to `localStorage`; the difference is that while data in `localStorage` doesn't expire, data in `sessionStorage` is cleared when the page session ends.

sessionStorage

Allows to save key/value pairs in the browser

Can I use sessionStorage ? Settings

2 results found

Caniuse (1) MDN (1)

Window API: sessionStorage

Usage % of all users ?

Global 97.52%

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE
4		3.1-3.2		10.1	6-7
5-106	12-106	4-16.0	2-106	11.5-91	8-10
107	107	16.1	107	92	11
108-110		16.2-TP	108-109		

Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	K Br
	3.2-16.0	4-18.0		12-12.1		2.1-4.4.4				
107	16.1	19.0	all	72	13.4	107	106	13.1	13.18	

Can I use sessionStorage ? Settings

2 results found

Caniuse (1) MDN (1)

Window API: sessionStorage

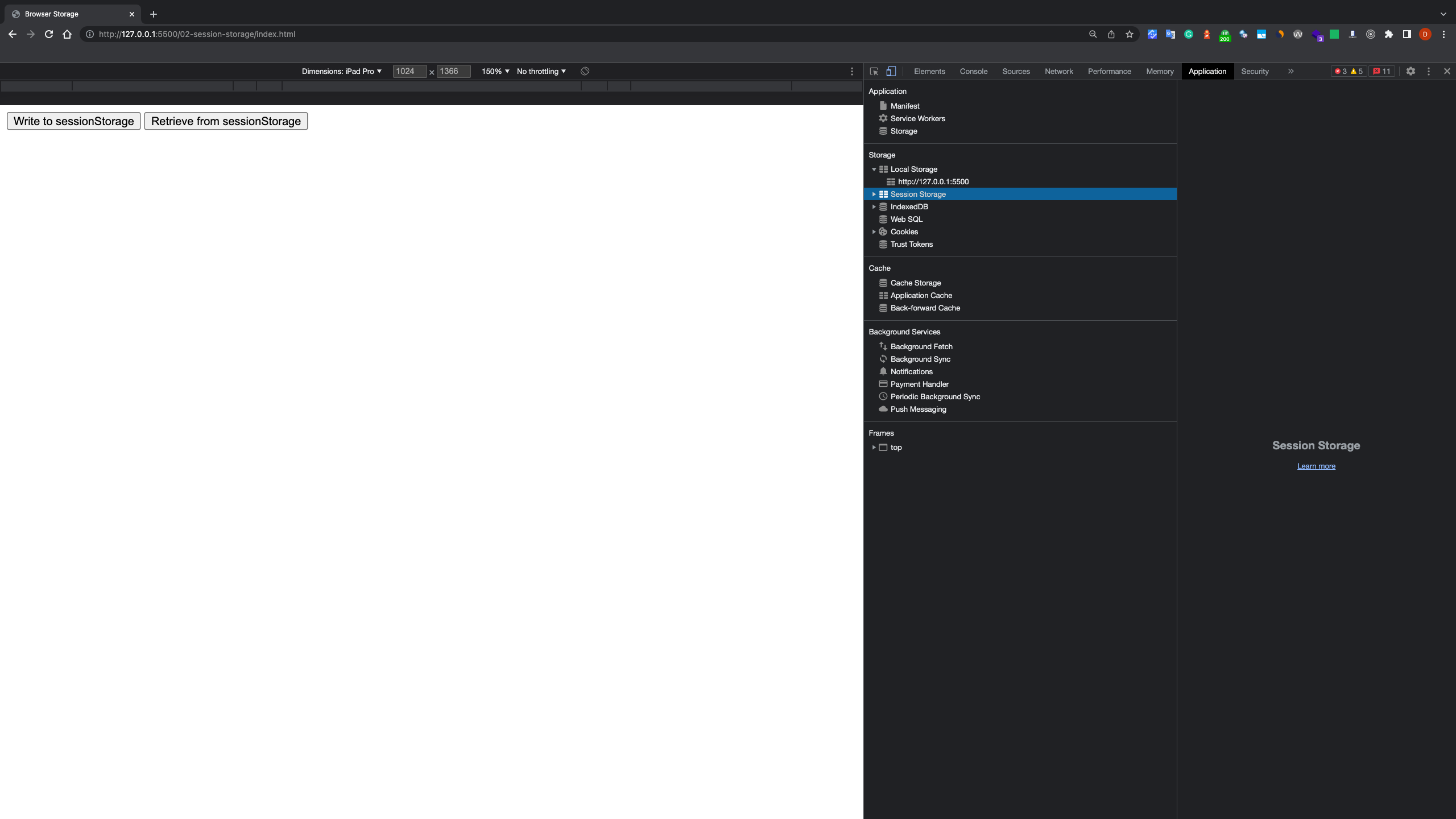
Usage % of all users ?

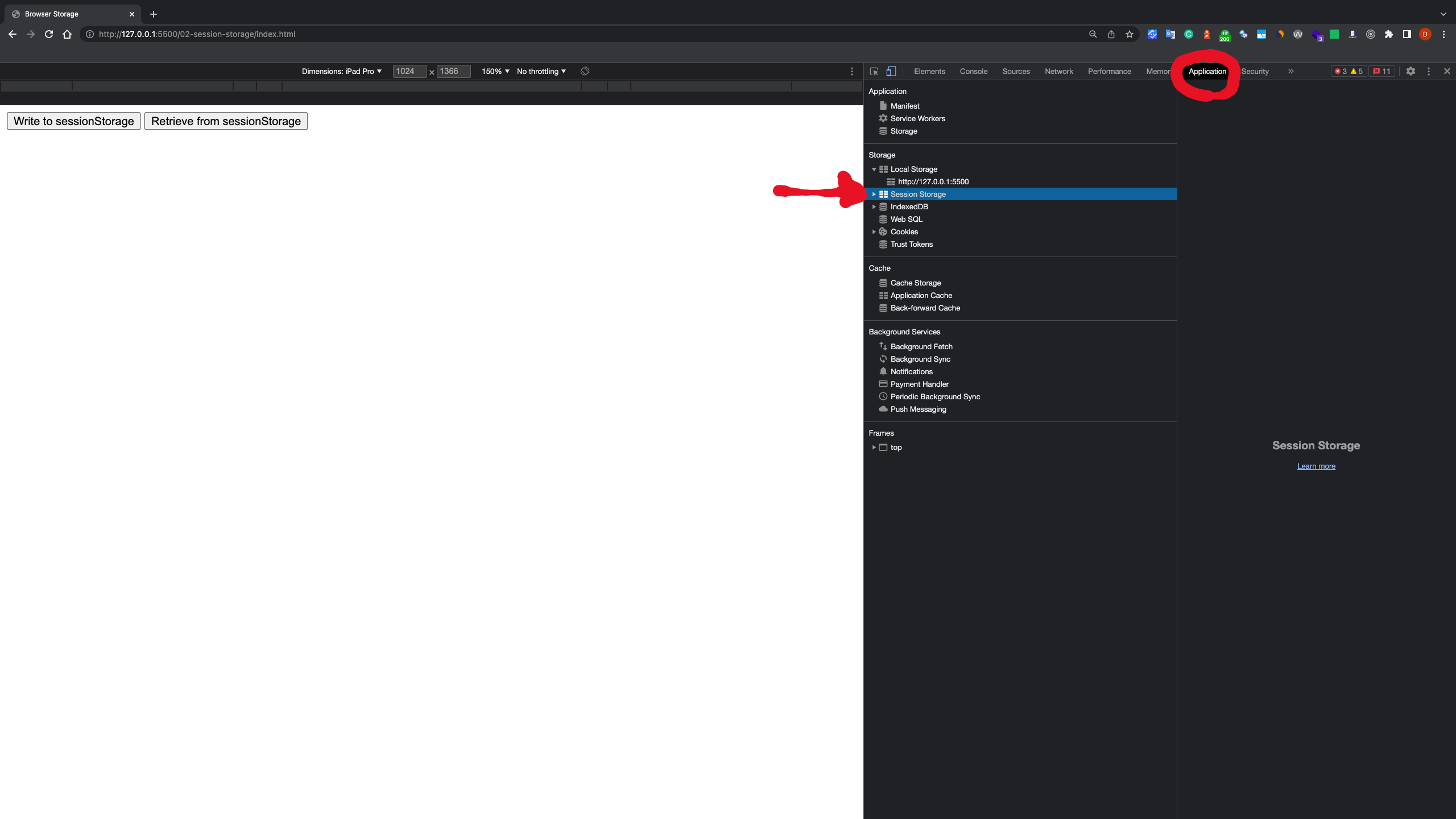
Global 97.52%

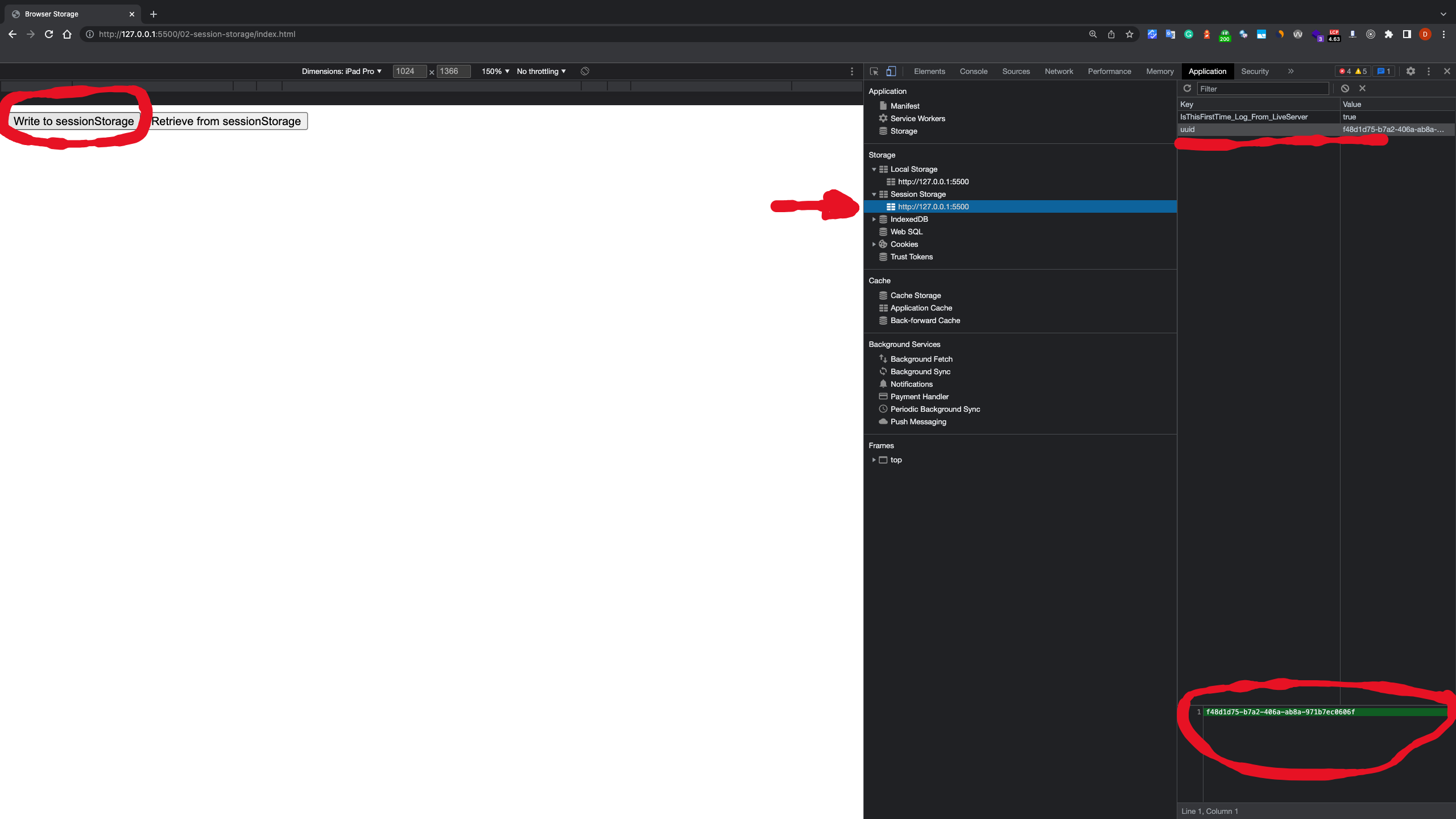
Current aligned | Usage relative | Date relative | Filtered | All

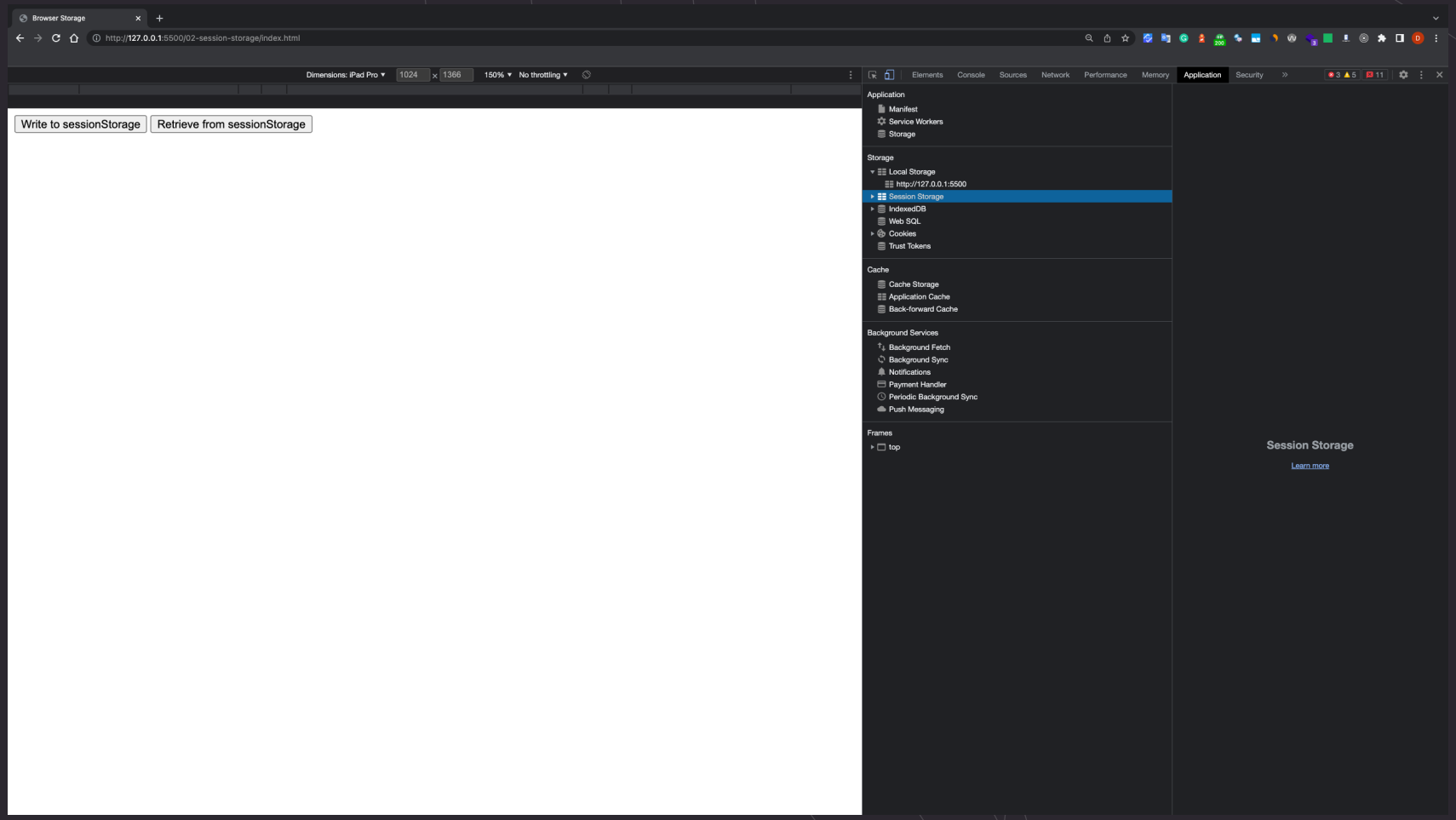
Chrome	Edge *	Safari	Firefox	Opera	IE
4		3.1-3.2		10.1	6-7
5-106	12-106	4-16.0	2-106	11.5-91	8-10
107	107	16.1	107	92	11
108-110		16.2-TP	108-109		

Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	K Br
	3.2-16.0	4-18.0		12-12.1		2.1-4.4.4				
107	16.1	19.0	all	72	13.4	107	106	13.1	13.18	


Browser Storage
http://127.0.0.1:5500/02-session-storage/index.html
Dimensions: iPad Pro
1024
1366
150%
No throttling
Write to sessionStorage
Retrieve from sessionStorage
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Application
Manifest
Service Workers
Storage
Storage
Local Storage
http://127.0.0.1:5500
Session Storage
IndexedDB
Web SQL
Cookies
Trust Tokens
Cache
Cache Storage
Application Cache
Back-forward Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
top
Session Storage
Learn more


Browser Storage
http://127.0.0.1:5500/02-session-storage/index.html
Dimensions: iPad Pro
1024
1366
150%
No throttling
Write to sessionStorage
Retrieve from sessionStorage
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Application
Manifest
Service Workers
Storage
Storage
Local Storage
http://127.0.0.1:5500
Session Storage
IndexedDB
Web SQL
Cookies
Trust Tokens
Cache
Cache Storage
Application Cache
Back-forward Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
top
Session Storage
Learn more


Browser Storage
http://127.0.0.1:5500/02-session-storage/index.html
Dimensions: iPad Pro
1024
1366
150%
No throttling
Write to sessionStorage
Retrieve from sessionStorage
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Application
Manifest
Service Workers
Storage
Storage
Local Storage
http://127.0.0.1:5500
Session Storage
http://127.0.0.1:5500
IndexedDB
Web SQL
Cookies
Trust Tokens
Cache
Cache Storage
Application Cache
Back-forward Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
top
Filter
Key
Value
IsThisFirstTime_Log_From_LiveServer
true
uuid
f48d1d75-b7a2-406a-ab8a-...
f48d1d75-b7a2-406a-ab8a-971b7ec0606f
Line 1, Column 1



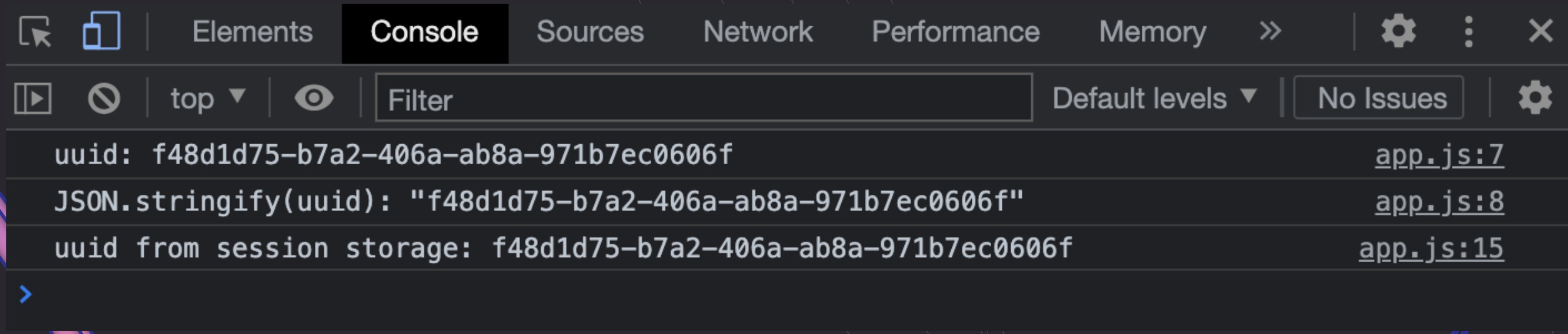
sessionStorage demo

Live Server & http://127.0.0.1:5500/02-session-storage/index.html

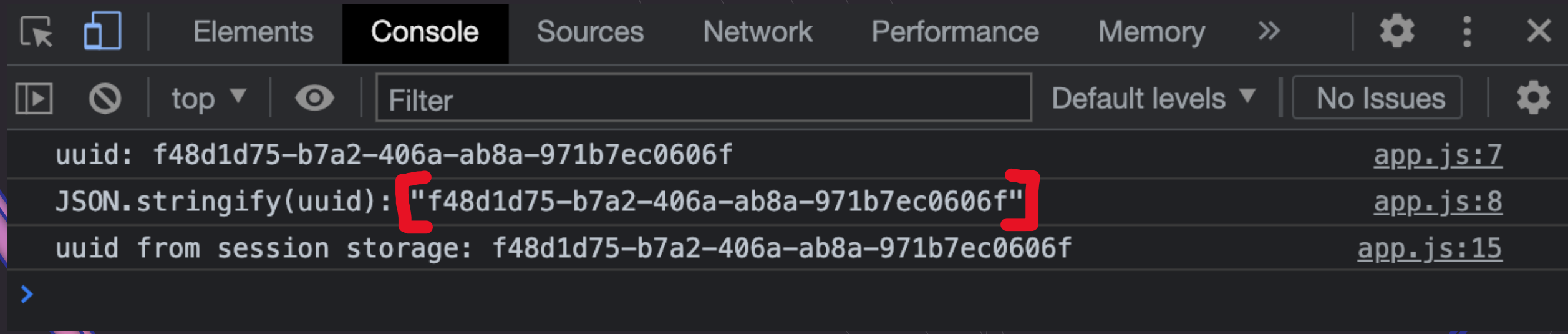
```
1  const writeToButton = document.getElementById('write-to-button');
2  const retrieveFromButton = document.getElementById('retrieve-from-button');
3
4  const uuid = 'f48d1d75-b7a2-406a-ab8a-971b7ec0606f';
5
6  writeToButton.addEventListener('click', () => {
7    console.log('uuid: ' + uuid);
8    console.log('JSON.stringify(uuid): ' + JSON.stringify(uuid));
9    sessionStorage.setItem('uuid', uuid);
10 });
11
12 retrieveFromButton.addEventListener('click', () => {
13   const uuidFromStorage = sessionStorage.getItem('uuid');
14
15   console.log('uuid from session storage: ' + uuidFromStorage);
16 });
17
```

```
1  const writeToButton = document.getElementById('write-to-button');
2  const retrieveFromButton = document.getElementById('retrieve-from-button');
3
4  const uuid = 'f48d1d75-b7a2-406a-ab8a-971b7ec0606f';
5
6  writeToButton.addEventListener('click', () => {
7    console.log('uuid: ' + uuid);
8    console.log('JSON.stringify(uuid): ' + JSON.stringify(uuid));
9    sessionStorage.setItem('uuid', uuid);
10 });
11
12 retrieveFromButton.addEventListener('click', () => {
13   const uuidFromStorage = sessionStorage.getItem('uuid');
14
15   console.log('uuid from session storage: ' + uuidFromStorage);
16 });
17
```

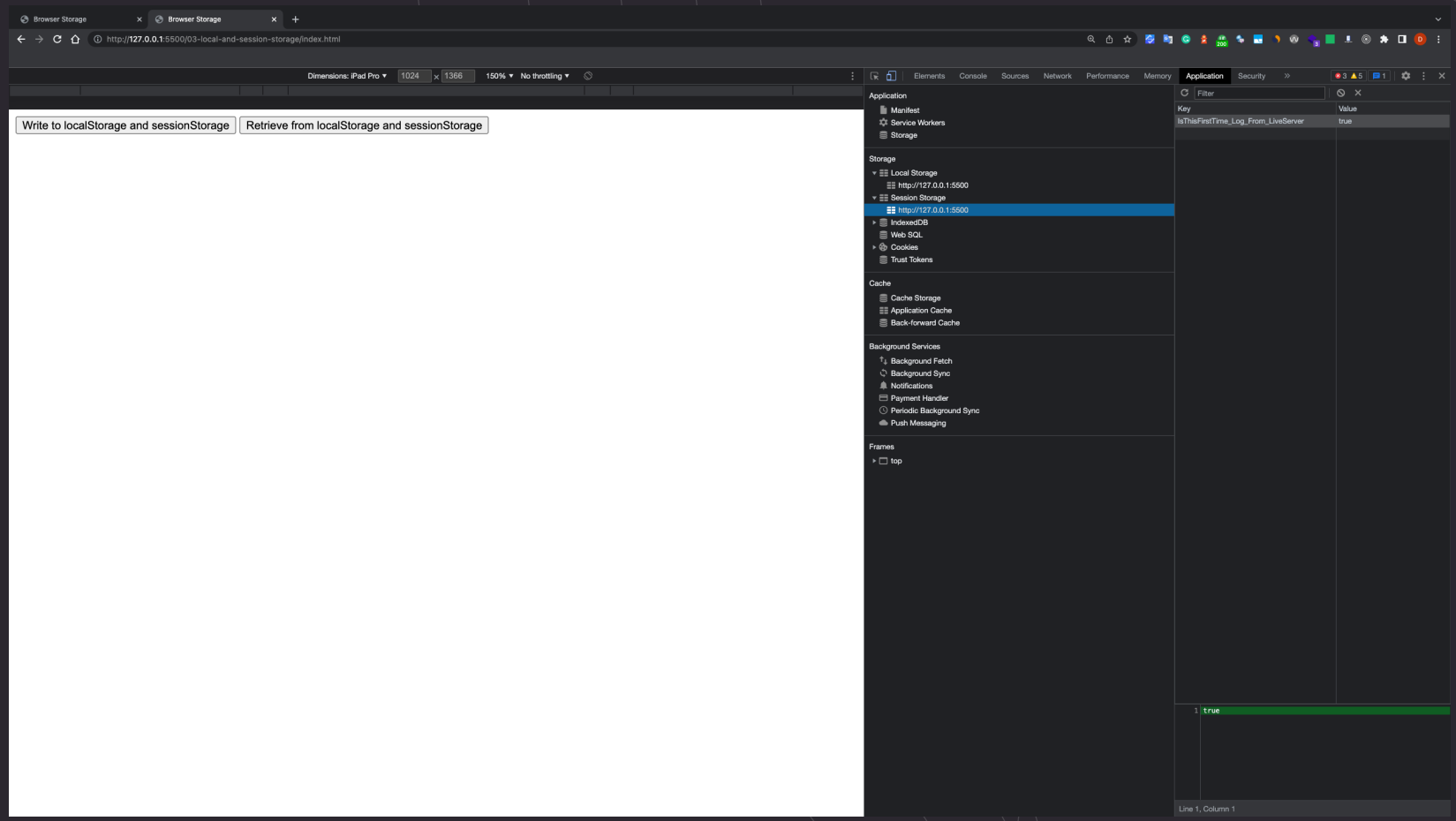
JSON.stringify()/JSON.parse() ???



JSON.stringify()/JSON.parse() ???

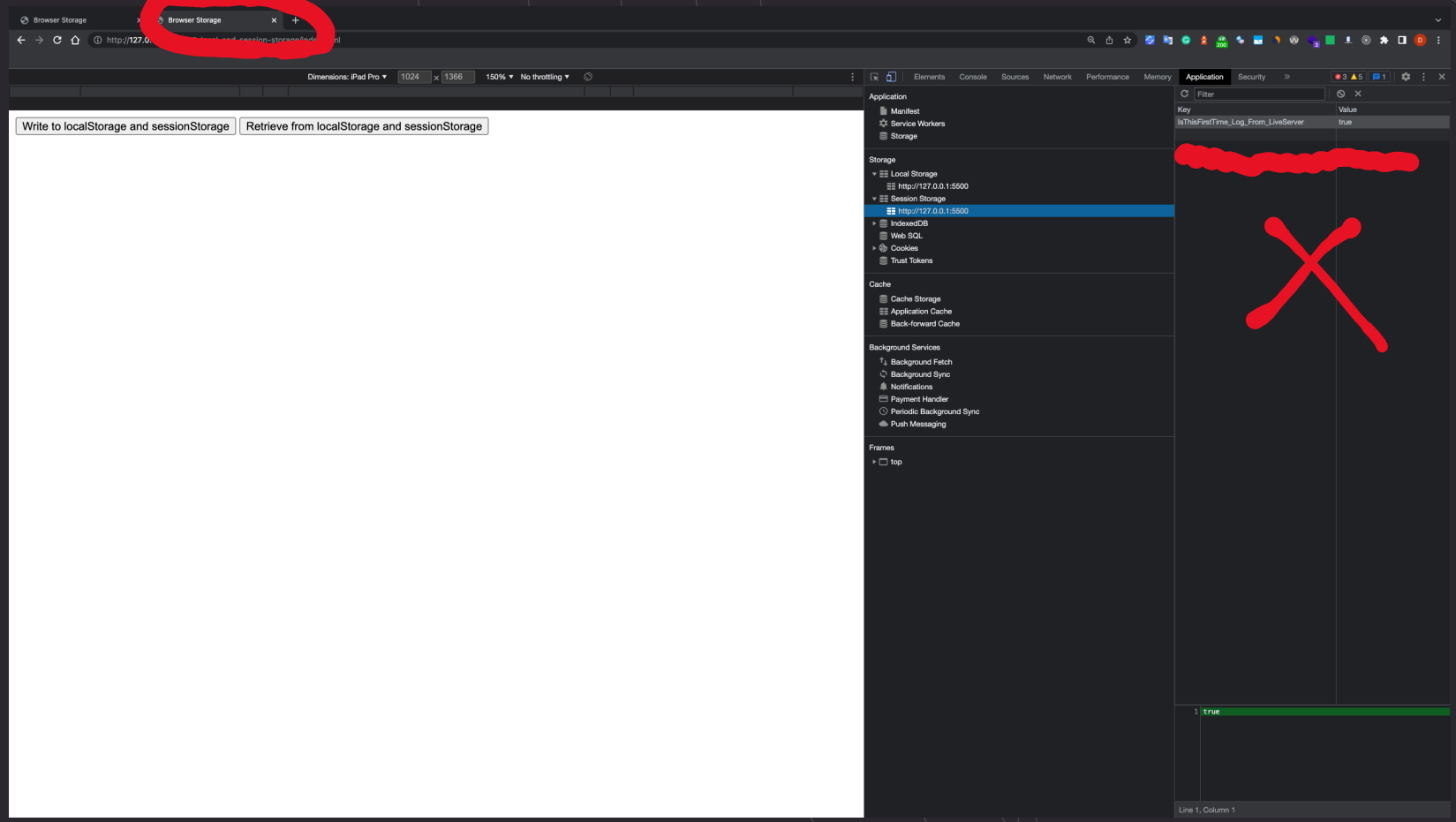


localStorage
vs
sessionStorage



localStorage vs sessionStorage

Live Server & http://127.0.0.1:5500/03-local-and-session-storage/index.html



Difference: storing data per session

localStorage vs sessionStorage

- **sessionStorage – data only in active tab, when new tab opens/browser close it’s gone**
- **localStorage – it never clears unless user/browser clears it**
- Both:
  - Are synchronous, so na promises
  - Must store a String key-value format
  - Not only user can delete it, but also browser, when, for example, running out of disk space
  - Good API
  - Size limit 5MB
- Can be stored: session ID of user, analytics key

IndexedDB

IndexedDB API

IndexedDB is a low-level API for client-side storage of significant amounts of structured data, including files/blobs. This API uses indexes to enable high-performance searches of this data. While Web Storage is useful for storing smaller amounts of data, it is less useful for storing larger amounts of structured data. IndexedDB provides a solution. This is the main landing page for MDN's IndexedDB coverage — here we provide links to the full API reference and usage guides, browser support details, and some explanation of key concepts.

IndexedDB

In-browser database for large amount of data

Can I use IndexedDB ? Settings

4 results found

☑ Caniuse (2) ☑ MDN (2)

IndexedDB - REC

Method of storing data client-side, allows indexed database queries.

Usage % of all users ?

Global 96.62% + 1.82% = 98.44%

unprefixed: 96.58% + 1.68% = 98.26%

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	K Br
		3.1-7					3.2-7.1									
4-10		2 7.1-9.1	2-3.6				2 8-9.3									
11-22		10-14	4-9				10-14.4									
23	1 12-18	3 14.1	10-15	10-12.1	6-9		3 14.8					2.1-4.3				
24-106	79-106	15-16.0	16-106	15-91	1 10		15-16.0	4-18.0		12-12.1		4.4-4.4.4				
107	107	16.1	107	92	1 11	107	16.1	19.0	all	72	13.4	107	106	13.1	13.18	
108-110		16.2-TP	108-109													

Can I use IndexedDB ? Settings

4 results found

☑ Caniuse (2) ☑ MDN (2)

IndexedDB - REC

Method of storing data client-side, allows indexed database queries.

Usage % of all users

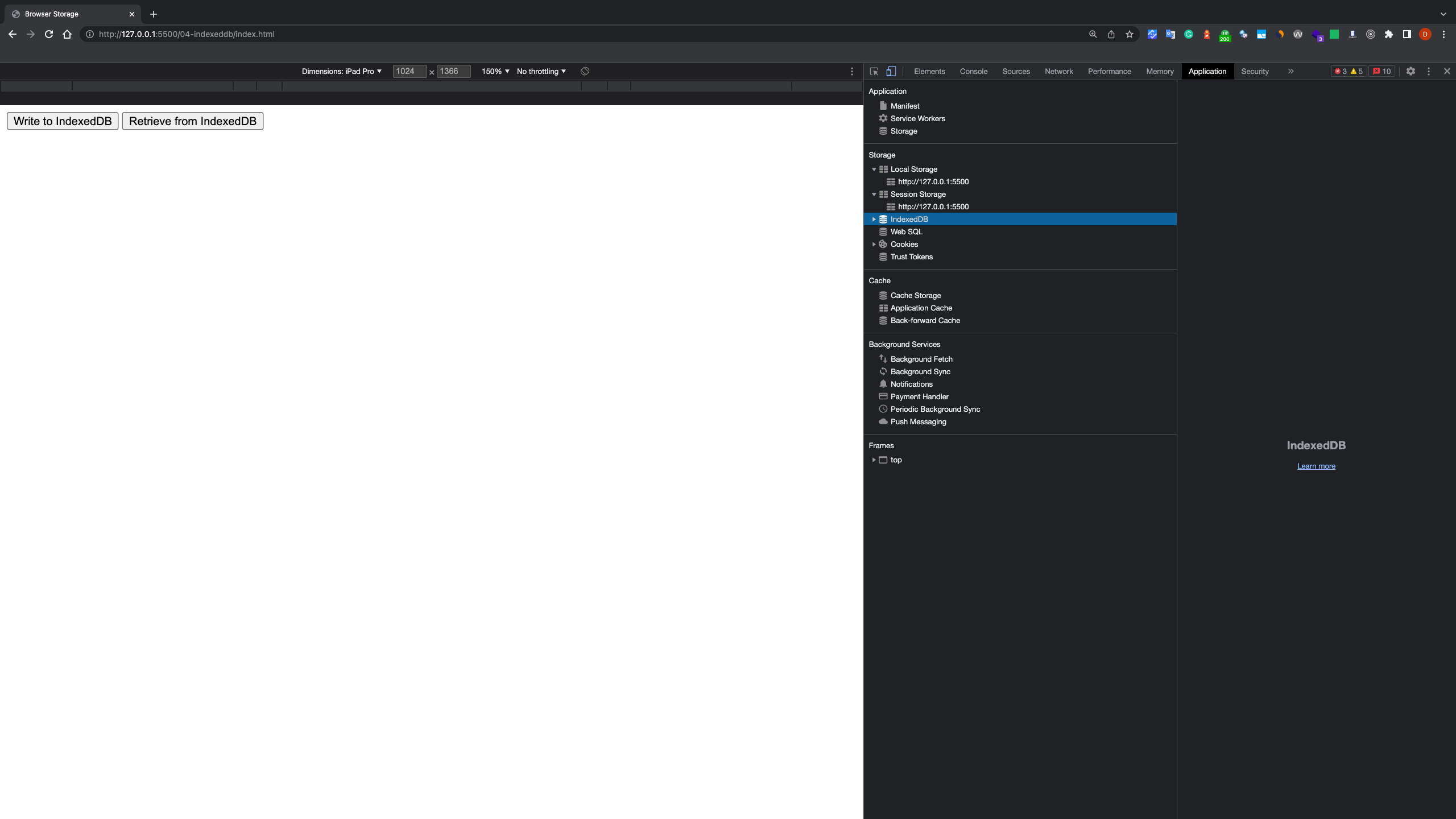
Global 96.62% + 1.82% = 98.44%

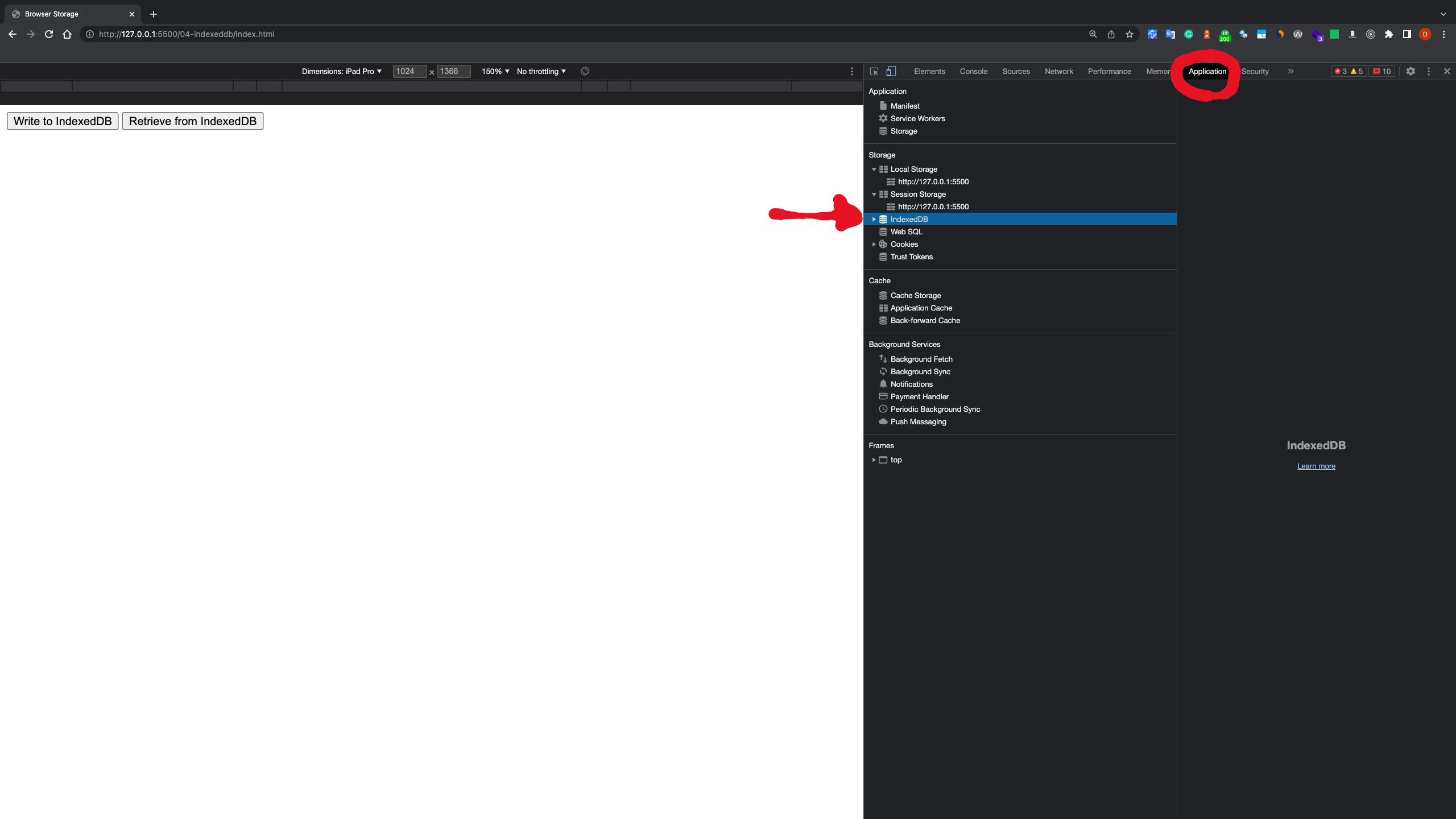
unprefixed: 96.58% + 1.68% = 98.26%

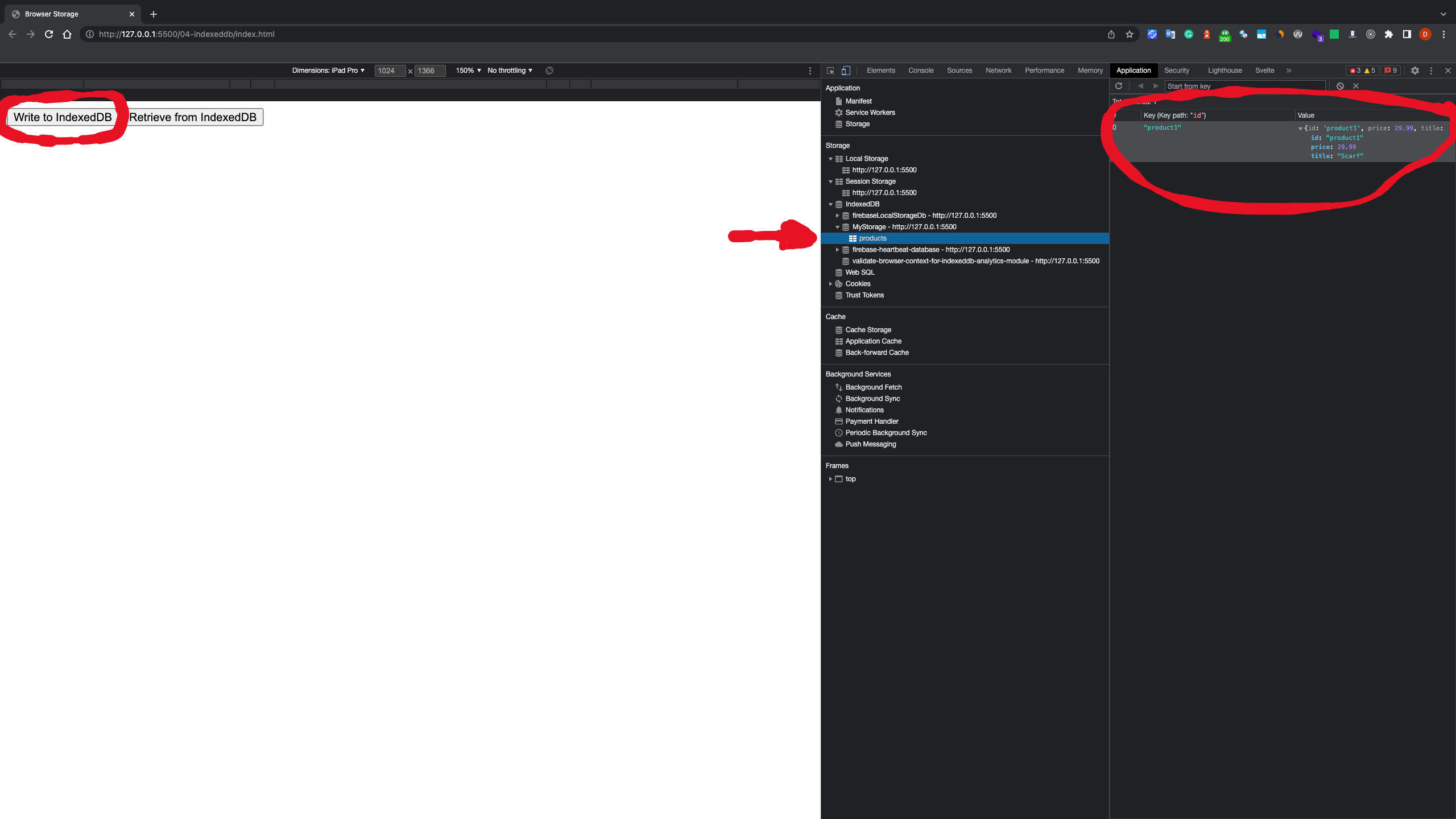
Current aligned | Usage relative | Date relative | Filtered | All

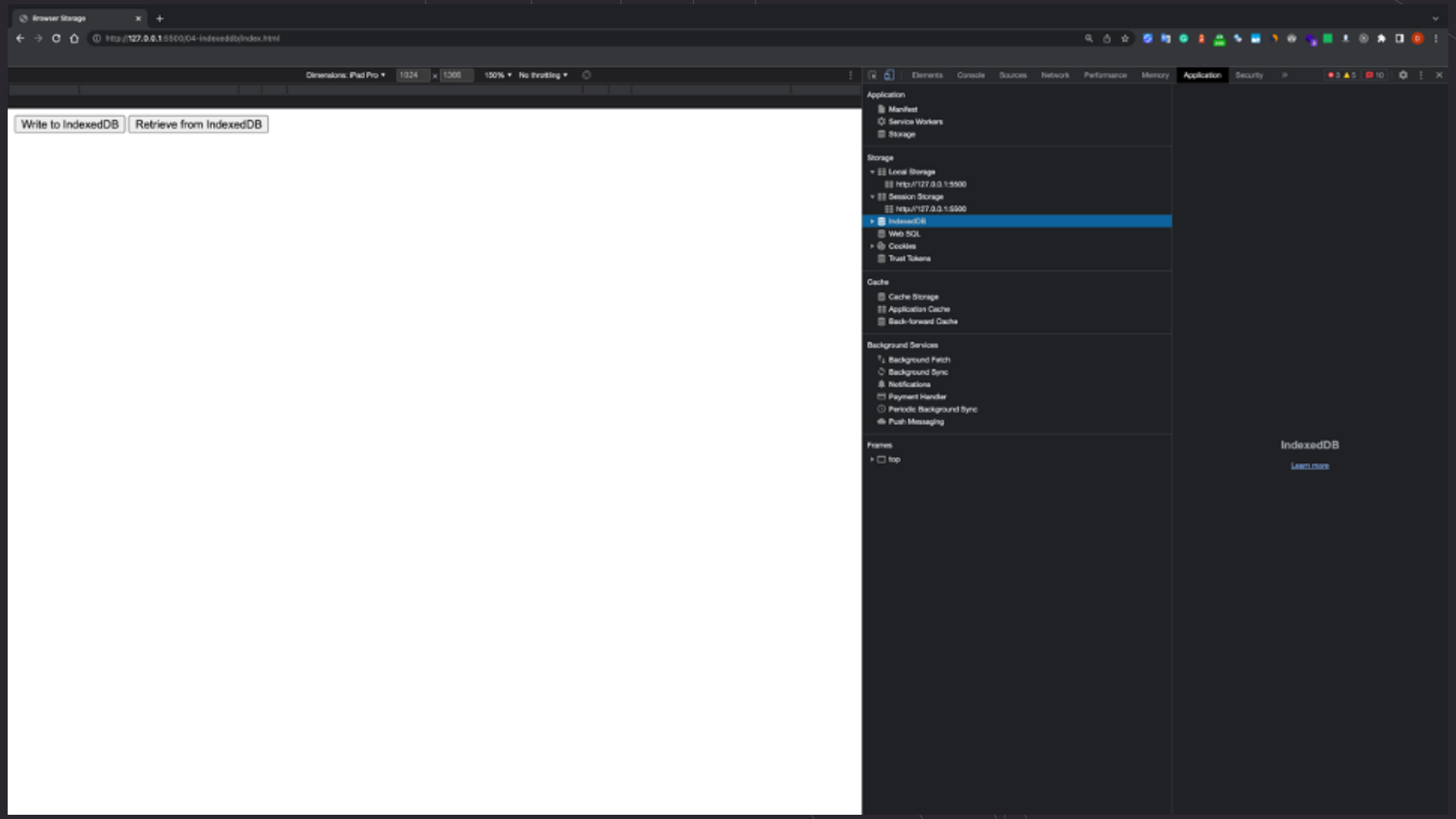
Chrome	Edge *	Safari	Firefox	Opera	IE
		3.1-7			
4-10		2 7.1-9.1	2-3.6		
11-22		10-14	4-9		
23	1 12-18	3 14.1	10-15	10-12.1	6-9
24-106	79-106	15-16.0	16-106	15-91	1 10
107	107	16.1	107	92	1 11
108-110		16.2-TP	108-109		

Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	K Br
	3.2-7.1									
	2 8-9.3									
	10-14.4									
	3 14.8					2.1-4.3				
	15-16.0	4-18.0		12-12.1		4.4-4.4.4				
107	16.1	19.0	all	72	13.4	107	106	13.1	13.18	


Browser Storage
http://127.0.0.1:5500/04-indexeddb/index.html
Dimensions: iPad Pro
1024
1366
150%
No throttling
Write to IndexedDB
Retrieve from IndexedDB
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Application
Manifest
Service Workers
Storage
Storage
Local Storage
http://127.0.0.1:5500
Session Storage
http://127.0.0.1:5500
IndexedDB
Web SQL
Cookies
Trust Tokens
Cache
Cache Storage
Application Cache
Back-forward Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
top
IndexedDB
Learn more


Browser Storage
http://127.0.0.1:5500/04-indexeddb/index.html
Dimensions: iPad Pro
1024
1366
150%
No throttling
Write to IndexedDB
Retrieve from IndexedDB
Elements
Console
Sources
Network
Performance
Application
Security
Application
Manifest
Service Workers
Storage
Storage
Local Storage
http://127.0.0.1:5500
Session Storage
http://127.0.0.1:5500
IndexedDB
Web SQL
Cookies
Trust Tokens
Cache
Cache Storage
Application Cache
Back-forward Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
top
IndexedDB
Learn more


Browser Storage
http://127.0.0.1:5500/04-indexeddb/index.html
Dimensions: iPad Pro
1024
1366
150%
No throttling
Write to IndexedDB
Retrieve from IndexedDB
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Svelte
Application
Manifest
Service Workers
Storage
Storage
Local Storage
http://127.0.0.1:5500
Session Storage
http://127.0.0.1:5500
IndexedDB
firebaseLocalStorageDb - http://127.0.0.1:5500
MyStorage - http://127.0.0.1:5500
products
firebase-heartbeat-database - http://127.0.0.1:5500
validate-browser-context-for-indexeddb-analytics-module - http://127.0.0.1:5500
Web SQL
Cookies
Trust Tokens
Cache
Cache Storage
Application Cache
Back-forward Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
top
Start from key
Key (Key path: "id")
Value
0
"product1"
{id: 'product1', price: 29.99, title:
id: "product1"
price: 29.99
title: "Scarf"

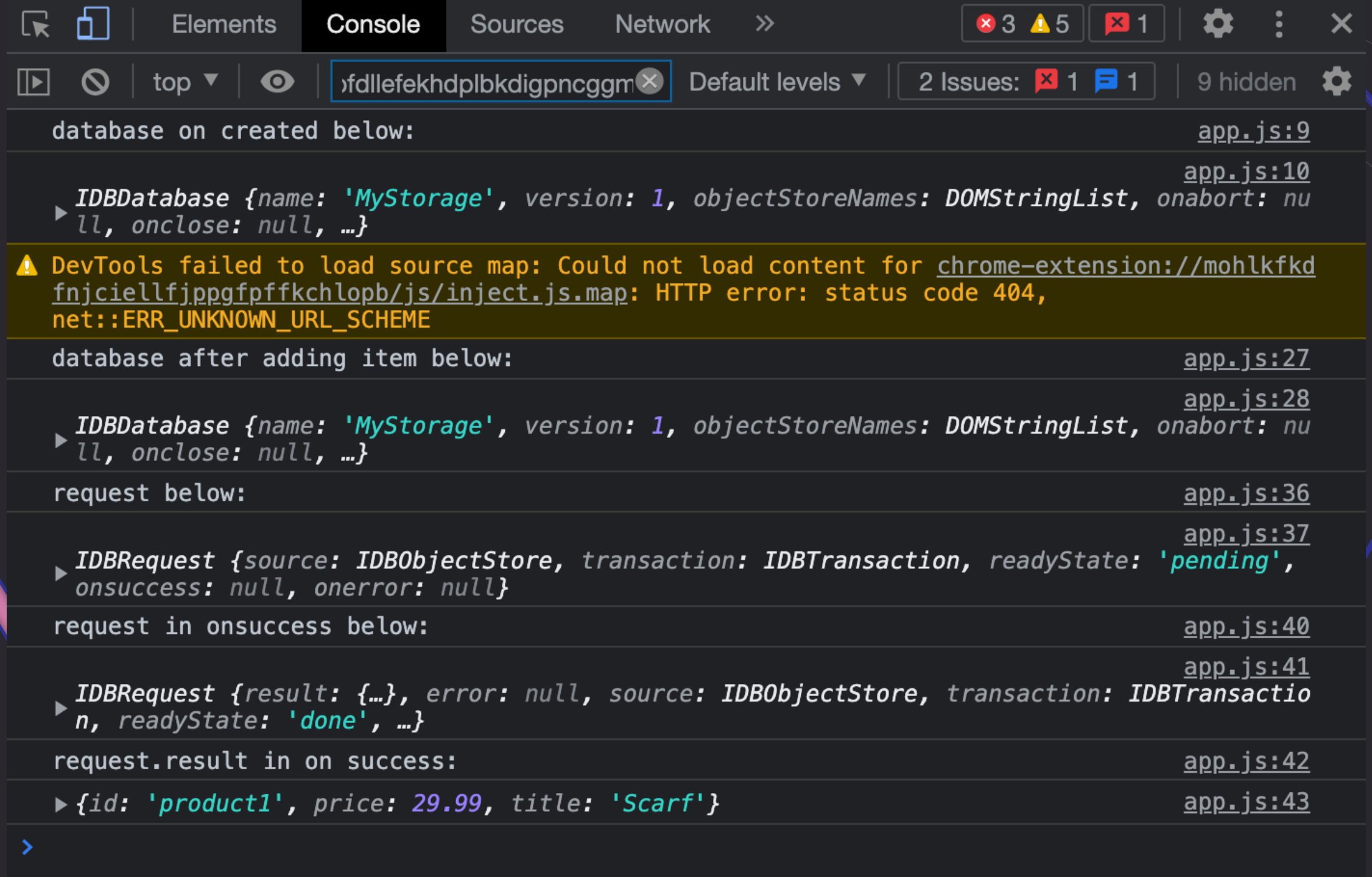


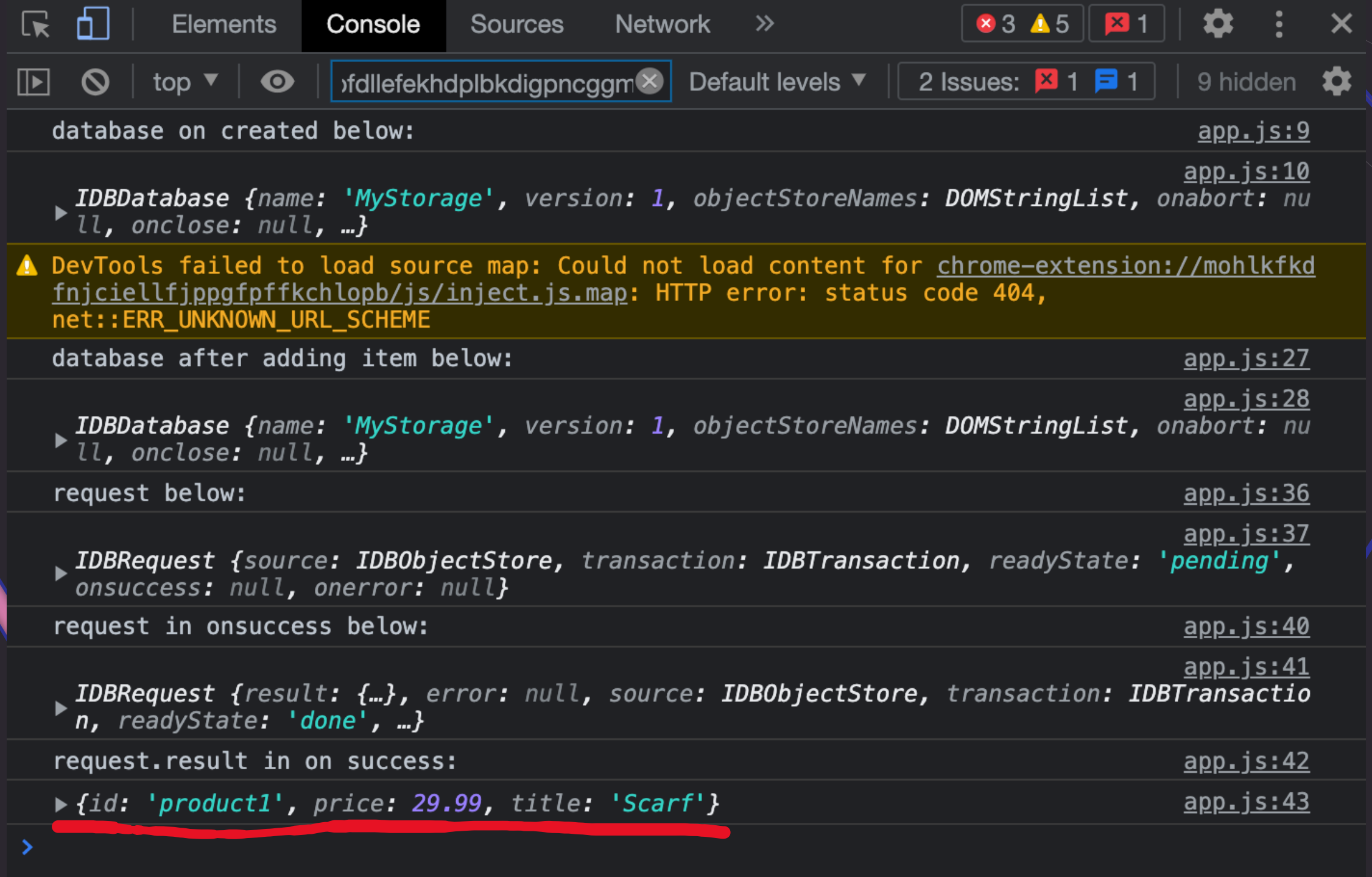
IndexedDB demo

Live Server & http://127.0.0.1:5500/04-indexeddb/index.html

```
const writeToButton = document.getElementById('write-to-button');
const retrieveFromButton = document.getElementById('retrieve-from-button');

const databaseRequest = indexedDB.open('MyStorage', 1);
let database;

databaseRequest.onsuccess = function (event) {
  database = event.target.result;
  console.log('database on created below:');
  console.log(database);
};

databaseRequest.onerror = function (event) {
  console.log('ERROR!');
  console.log(event);
};

writeToButton.addEventListener('click', () => {
  const productsStore = database
    .transaction('products', 'readwrite')
    .objectStore('products');
  productsStore.add({
    id: 'product1',
    price: 29.99,
    title: 'Scarf',
  });
  console.log('database after adding item below:');
  console.log(database);
});

retrieveFromButton.addEventListener('click', () => {
  const productsStore = database
    .transaction('products', 'readwrite')
    .objectStore('products');
  const request = productsStore.get('product1');
  console.log('request below:');
  console.log(request);

  request.onsuccess = function () {
    console.log('request in onsuccess below:');
    console.log(request);
    console.log('request.result in on success:');
    console.log(request.result);
  }
});

```

```js
const writeToButton = document.getElementById('write-to-button');
const retrieveFromButton = document.getElementById('retrieve-from-button');

const databaseRequest = indexedDB.open('MyStorage', 1);
let database;

databaseRequest.onsuccess = function (event) {
  database = event.target.result;
  console.log('database on created below:');
  console.log(database);
};

databaseRequest.onerror = function (event) {
  console.log('ERROR!');
  console.log(event);
};

writeToButton.addEventListener('click', () => {
  const productsStore = database
    .transaction('products', 'readwrite')
    .objectStore('products');
  productsStore.add({
    id: 'product1',
    price: 29.99,
    title: 'Scarf',
  });
  console.log('database after adding item below:');
  console.log(database);
});

retrieveFromButton.addEventListener('click', () => {
  const productsStore = database
    .transaction('products', 'readwrite')
    .objectStore('products');
  const request = productsStore.get('product1');
  console.log('request below:');
  console.log(request);

  request.onsuccess = function () {
    console.log('request in onsuccess below:');
    console.log(request);
    console.log('request.result in on success:');
    console.log(request.result);
  }
});

```


Elements
Console
Sources
Network
3 5
1
top
fdllefekhdplbkdigpncggm
Default levels
2 Issues: 1 1
9 hidden
database on created below:
app.js:9
app.js:10
IDBDatabase {name: 'MyStorage', version: 1, objectStoreNames: DOMStringList, onabort: null, onclose: null, …}
DevTools failed to load source map: Could not load content for chrome-extension://mohlkfkdfnjciellfjppgfpffkchlopb/js/inject.js.map: HTTP error: status code 404, net::ERR_UNKNOWN_URL_SCHEME
database after adding item below:
app.js:27
app.js:28
IDBDatabase {name: 'MyStorage', version: 1, objectStoreNames: DOMStringList, onabort: null, onclose: null, …}
request below:
app.js:36
app.js:37
IDBRequest {source: IDBObjectStore, transaction: IDBTransaction, readyState: 'pending', onsuccess: null, onerror: null}
request in onsuccess below:
app.js:40
app.js:41
IDBRequest {result: {…}, error: null, source: IDBObjectStore, transaction: IDBTransaction, readyState: 'done', …}
request.result in on success:
app.js:42
{id: 'product1', price: 29.99, title: 'Scarf'}
app.js:43


Elements
Console
Sources
Network
3 5 1
top
fdllefekhdplbkdigpncggm
Default levels
2 Issues: 1 1
9 hidden
database on created below: app.js:9
app.js:10
IDBDatabase {name: 'MyStorage', version: 1, objectStoreNames: DOMStringList, onabort: null, onclose: null, …}
DevTools failed to load source map: Could not load content for chrome-extension://mohlkfkdfnjciellfjppgfpffkchlopb/js/inject.js.map: HTTP error: status code 404, net::ERR_UNKNOWN_URL_SCHEME
database after adding item below: app.js:27
app.js:28
IDBDatabase {name: 'MyStorage', version: 1, objectStoreNames: DOMStringList, onabort: null, onclose: null, …}
request below: app.js:36
app.js:37
IDBRequest {source: IDBObjectStore, transaction: IDBTransaction, readyState: 'pending', onsuccess: null, onerror: null}
request in onsuccess below: app.js:40
app.js:41
IDBRequest {result: {…}, error: null, source: IDBObjectStore, transaction: IDBTransaction, readyState: 'done', …}
request.result in on success: app.js:42
{id: 'product1', price: 29.99, title: 'Scarf'} app.js:43

- Most sophisticated, in-browser database, the most complex
- NoSQL
- Object Store format
- Large amount of data (up to **80% of disk space** – yes, it can be in GB's)
- Asynchronous
- Access can be readonly or readwrite
- A bit clunky API = there are 3rd party libraries to work with IndexedDB to make it more pleasant, especially to have Promises

Cookies

Using HTTP cookies

An **HTTP cookie** (web cookie, browser cookie) is a small piece of data that a server sends to a user's web browser. The browser may store the cookie and send it back to the same server with later requests. Typically, an HTTP cookie is used to tell if two requests come from the same browser—keeping a user logged in, for example. It remembers stateful information for the stateless HTTP protocol.

Cookies

Small piece of data

Can I use Document API: cookie ? Settings

2 results found

☑ Caniuse (1) ☑ MDN (1)

Document API: cookie

Usage % of all users ?

Global 97.92%

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	K Br
												2.1-4.3				
4-106	12-106	3.1-16.0	1 2-106	10-91	6-10		3.2-16.0	4-18.0		12-12.1		4.4-4.4.4				
107	107	16.1	1 107	92	11	107	16.1	19.0	all	72	13.4	107	1 106	13.1	13.18	
108-110		16.2-TP	1 108-109													

Can I use Document API: cookie ? Settings

2 results found

☑ Caniuse (1) ☑ MDN (1)

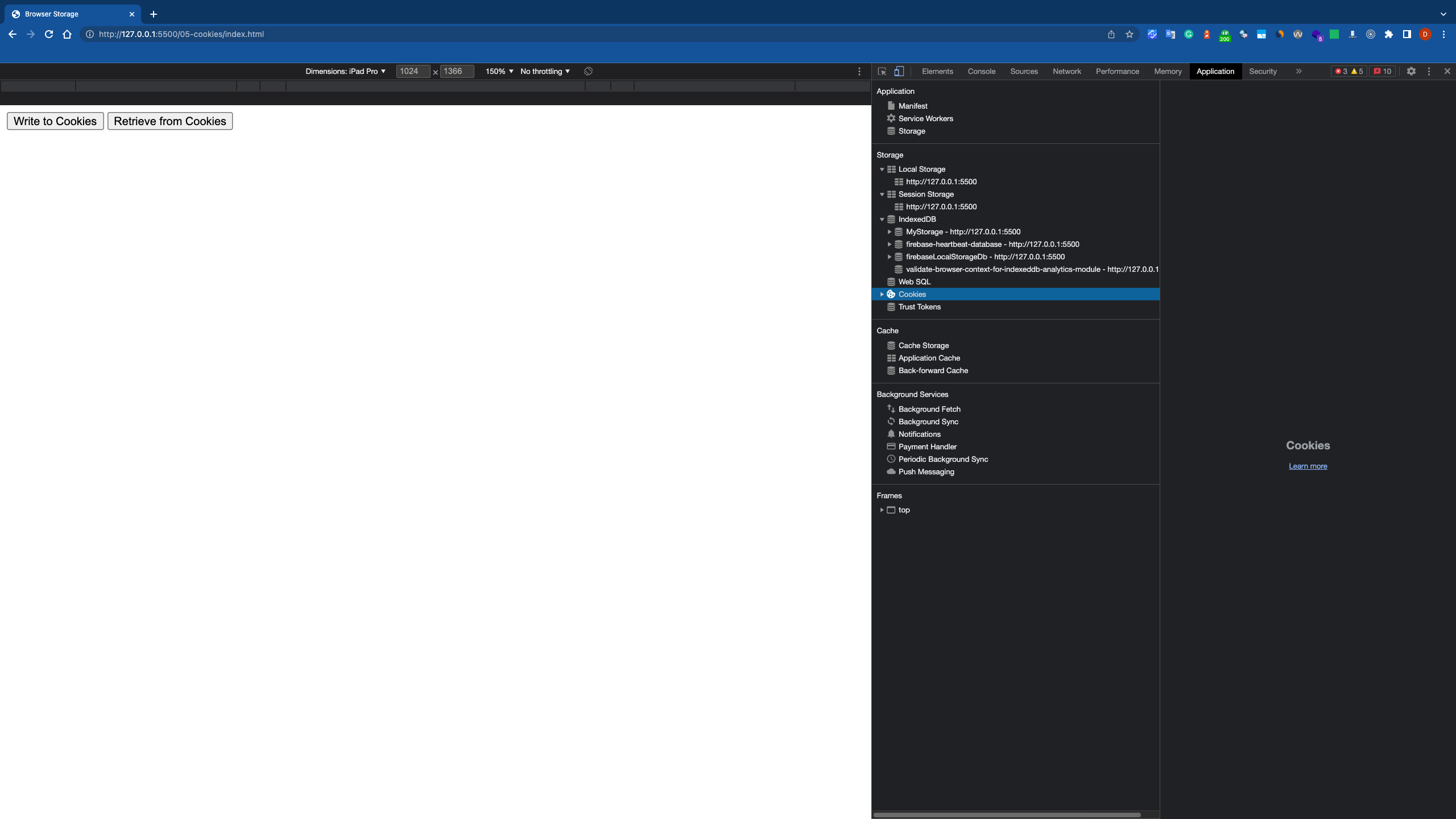
Document API: cookie

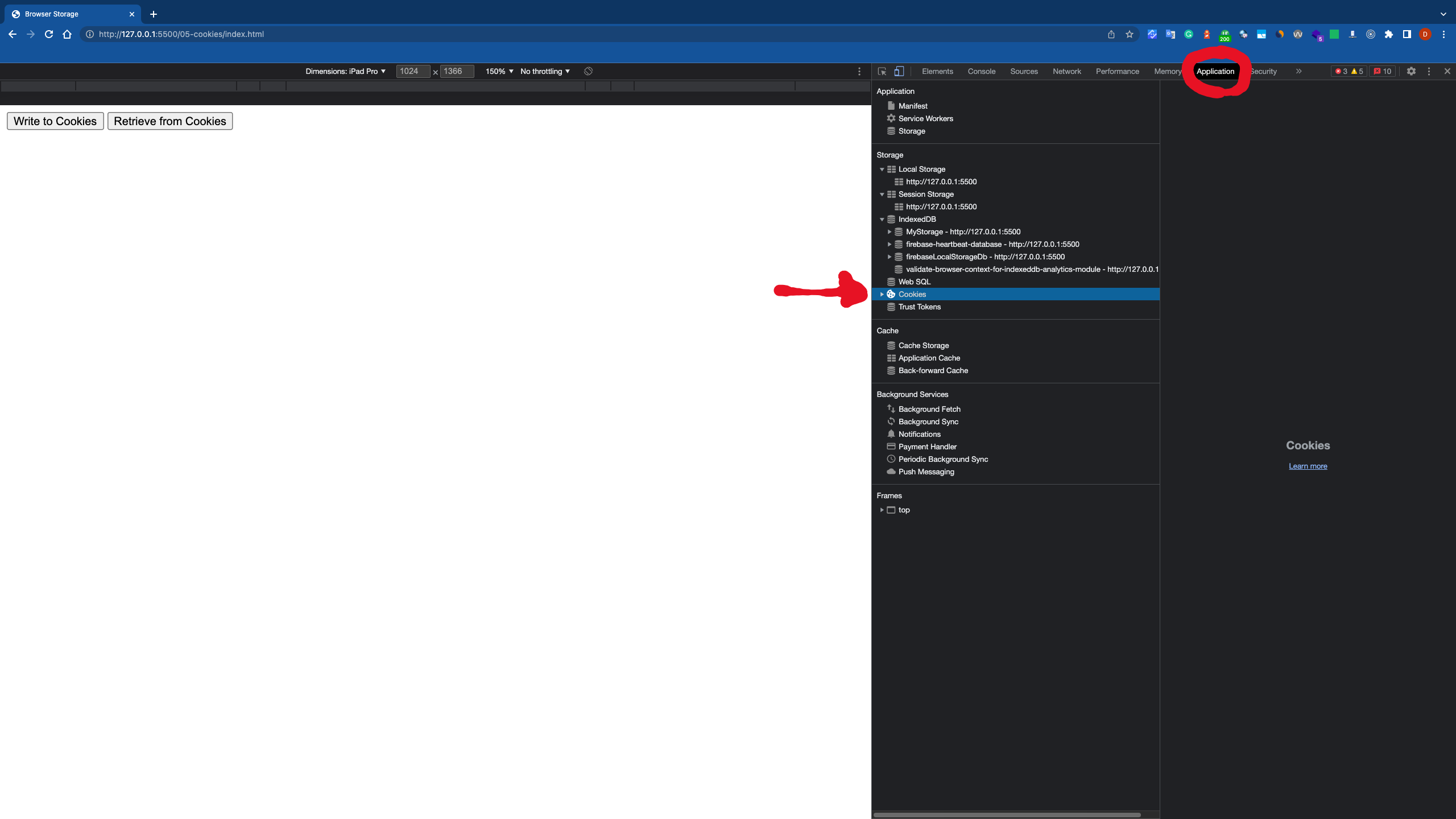
Usage % of all users

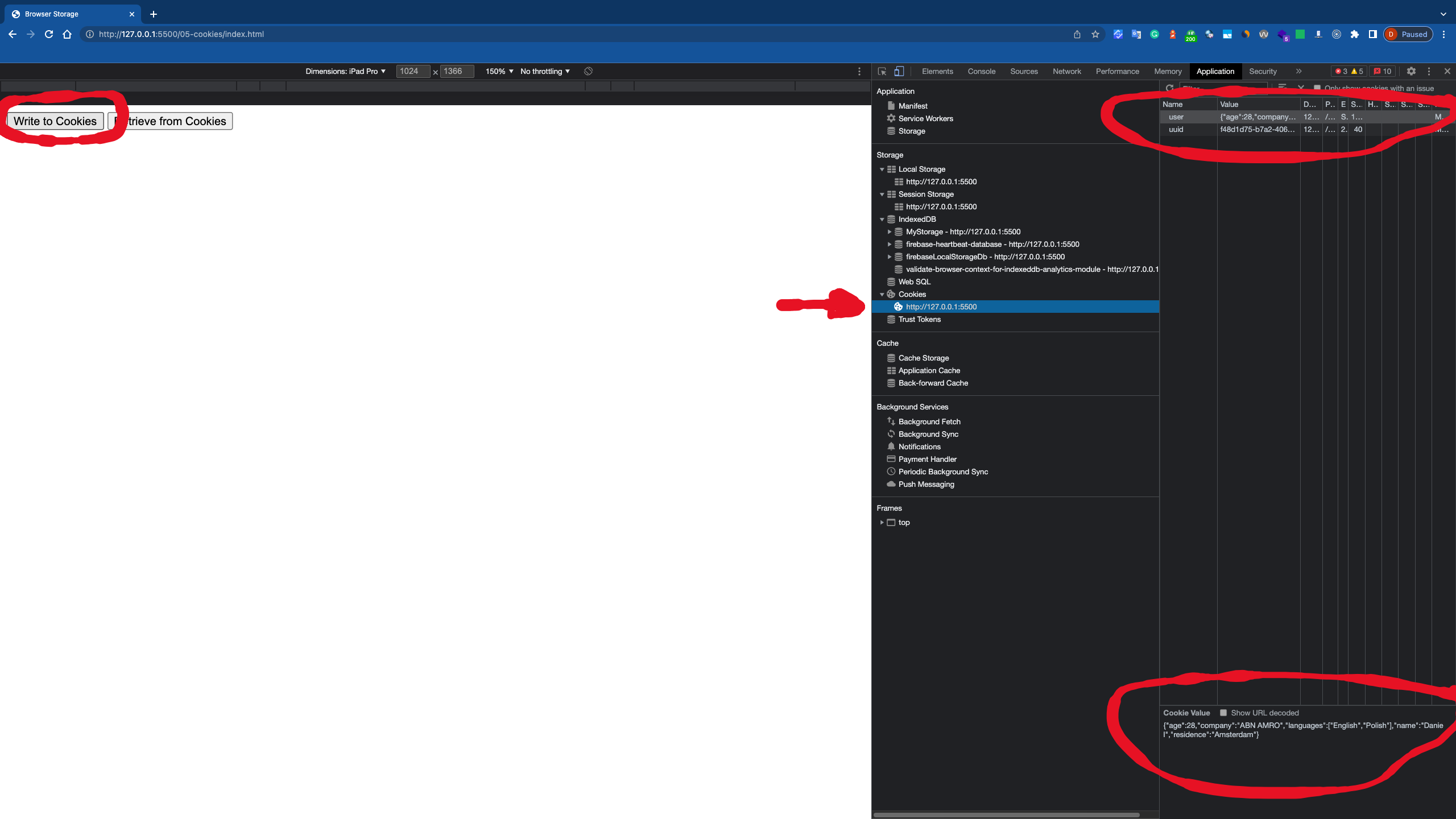
Global 97.92%

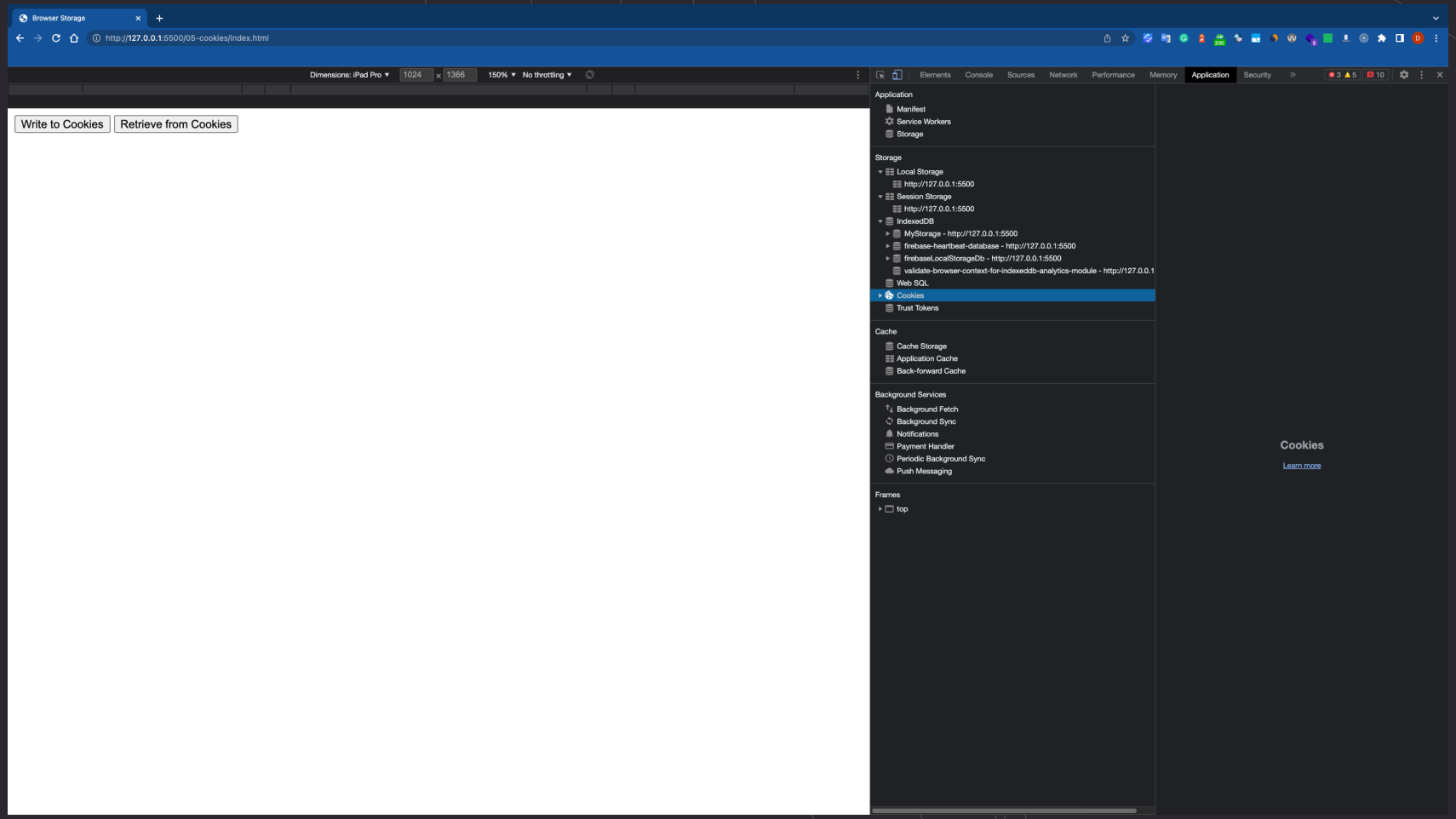
Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	K Br
												2.1-4.3				
4-106	12-106	3.1-16.0	1 2-106	10-91	6-10		3.2-16.0	4-18.0		12-12.1		4.4-4.4.4				
107	107	16.1	1 107	92	11	107	16.1	19.0	all	72	13.4	107	1 106	13.1	13.18	
108-110		16.2-TP	1 108-109													


Browser Storage
http://127.0.0.1:5500/05-cookies/index.html
Dimensions: iPad Pro 1024 x 1366 150% No throttling
Write to Cookies
Retrieve from Cookies
Elements Console Sources Network Performance Memory Application Security
Application
Manifest
Service Workers
Storage
Storage
Local Storage
http://127.0.0.1:5500
Session Storage
http://127.0.0.1:5500
IndexedDB
MyStorage - http://127.0.0.1:5500
firebase-heartbeat-database - http://127.0.0.1:5500
firebaseLocalStorageDb - http://127.0.0.1:5500
validate-browser-context-for-indexeddb-analytics-module - http://127.0.0.1
Web SQL
Cookies
Trust Tokens
Cache
Cache Storage
Application Cache
Back-forward Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
top
Cookies
Learn more


Browser Storage
http://127.0.0.1:5500/05-cookies/index.html
Dimensions: iPad Pro 1024 x 1366 150% No throttling
Elements Console Sources Network Performance Memory Application Security
Write to Cookies
Retrieve from Cookies
Application
Manifest
Service Workers
Storage
Storage
Local Storage
http://127.0.0.1:5500
Session Storage
http://127.0.0.1:5500
IndexedDB
MyStorage - http://127.0.0.1:5500
firebase-heartbeat-database - http://127.0.0.1:5500
firebaseLocalStorageDb - http://127.0.0.1:5500
validate-browser-context-for-indexeddb-analytics-module - http://127.0.0.1
Web SQL
Cookies
Trust Tokens
Cache
Cache Storage
Application Cache
Back-forward Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
top
Cookies
Learn more


Browser Storage
http://127.0.0.1:5500/05-cookies/index.html
Paused
Dimensions: iPad Pro
1024
1366
150%
No throttling
Write to Cookies
trieve from Cookies
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Application
Manifest
Service Workers
Storage
Storage
Local Storage
http://127.0.0.1:5500
Session Storage
http://127.0.0.1:5500
IndexedDB
MyStorage - http://127.0.0.1:5500
firebase-heartbeat-database - http://127.0.0.1:5500
firebaseLocalStorageDb - http://127.0.0.1:5500
validate-browser-context-for-indexeddb-analytics-module - http://127.0.0.1
Web SQL
Cookies
http://127.0.0.1:5500
Trust Tokens
Cache
Cache Storage
Application Cache
Back-forward Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
top
Only show cookies with an issue
Name
Value
user
{"age":28,"company...
uuid
f48d1d75-b7a2-406...
Cookie Value
Show URL decoded
{"age":28,"company":"ABN AMRO","languages":["English","Polish"],"name":"Daniel","residence":"Amsterdam"}

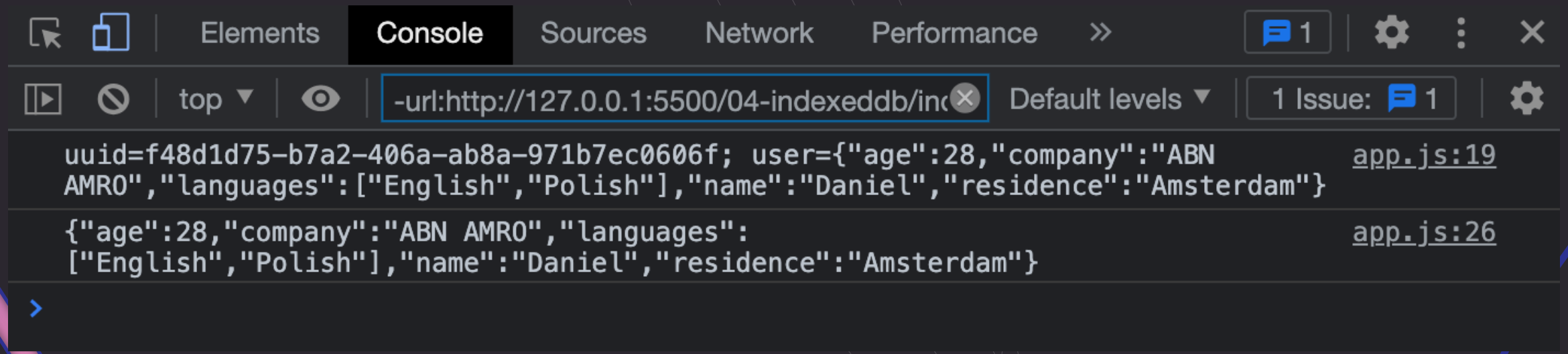


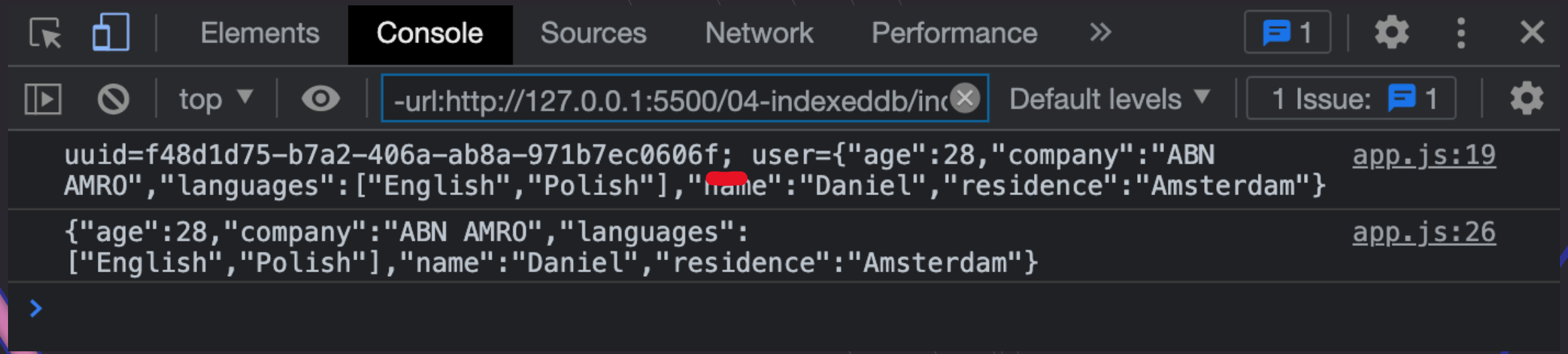
Cookies demo

Live Server & http://127.0.0.1:5500/05-cookies/index.html

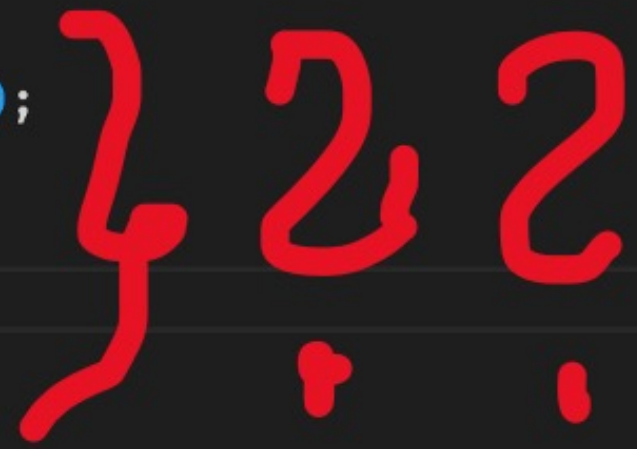
```
const writeToButton = document.getElementById('write-to-button');
const retrieveFromButton = document.getElementById('retrieve-from-button');

writeToButton.addEventListener('click', () => {
  const user = {
    age: 28,
    company: 'ABN AMRO',
    languages: ['English', 'Polish'],
    name: 'Daniel',
    residence: 'Amsterdam'
  };
  const uuid = 'f48d1d75-b7a2-406a-ab8a-971b7ec0606f';

  document.cookie = `uuid=${uuid}; max-age=360`;
  document.cookie = `user=${JSON.stringify(user)}`;
});

retrieveFromButton.addEventListener('click', () => {
  console.log(document.cookie); // Access cookies.

  // Get user's value in a "beautify" way.
  const cookieData = document.cookie.split(';');
  const data = cookieData.map(i => {
    return i.trim();
  });
  console.log(data[1].split('=')[1]);
});

```

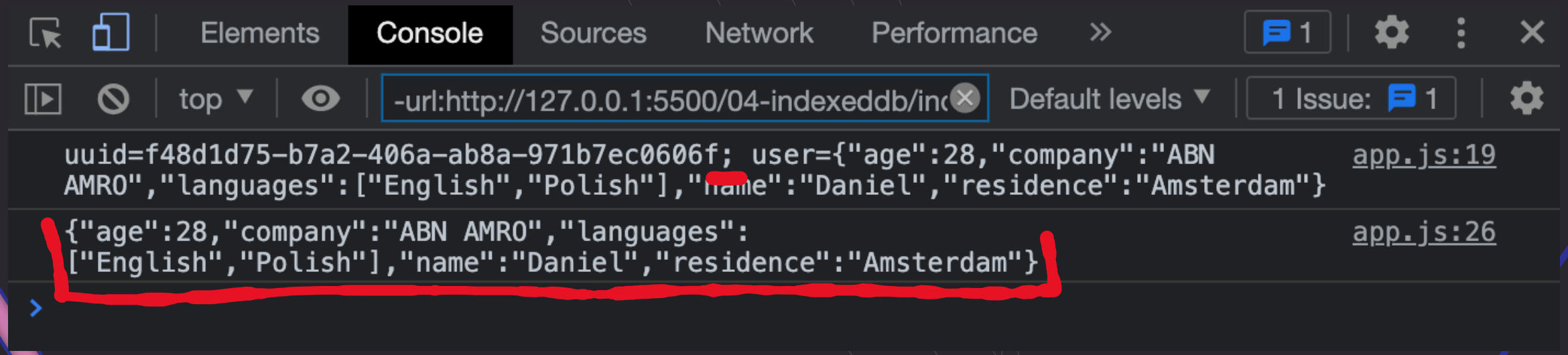
```
1  const writeToButton = document.getElementById('write-to-button');
2  const retrieveFromButton = document.getElementById('retrieve-from-button');
3
4  writeToButton.addEventListener('click', () => {
5    const user = {
6      age: 28,
7      company: 'ABN AMRO',
8      languages: ['English', 'Polish'],
9      name: 'Daniel',
10     residence: 'Amsterdam'
11   };
12   const uuid = 'f48d1d75-b7a2-406a-ab8a-971b7ec0606f';
13
14   document.cookie = `uuid=${uuid}; max-age=360`;
15   document.cookie = `user=${JSON.stringify(user)}`;
16 });
17
18 retrieveFromButton.addEventListener('click', () => {
19   console.log(document.cookie); // Access cookies.
20
21   // Get user's value in a "beautify" way.
22   const cookieData = document.cookie.split(';');
23   const data = cookieData.map(i => {
24     return i.trim();
25   });
26   console.log(data[1].split('=')[1]);
27 });
28
```


Elements
Console
Sources
Network
Performance
1
top
-url:http://127.0.0.1:5500/04-indexeddb/in
Default levels
1 Issue: 1
uuid=f48d1d75-b7a2-406a-ab8a-971b7ec0606f; user={"age":28,"company":"ABN AMRO","languages":["English","Polish"],"name":"Daniel","residence":"Amsterdam"}
app.js:19
{"age":28,"company":"ABN AMRO","languages":["English","Polish"],"name":"Daniel","residence":"Amsterdam"}
app.js:26


Elements
Console
Sources
Network
Performance
1
top
-url:http://127.0.0.1:5500/04-indexeddb/in
Default levels
1 Issue: 1
uuid=f48d1d75-b7a2-406a-ab8a-971b7ec0606f; user={"age":28,"company":"ABN AMRO","languages":["English","Polish"],"name":"Daniel","residence":"Amsterdam"}
app.js:19
{"age":28,"company":"ABN AMRO","languages":["English","Polish"],"name":"Daniel","residence":"Amsterdam"}
app.js:26

```
18  retrieveFromButton.addEventListener('click', () => {
19    console.log(document.cookie); // Access cookies.
20
21    // Get user's value in a "beautify" way.
22    const cookieData = document.cookie.split(';');
23    const data = cookieData.map(i => {
24      return i.trim();
25    });
26    console.log(data[1].split('=')[1]);
27  });
28
```



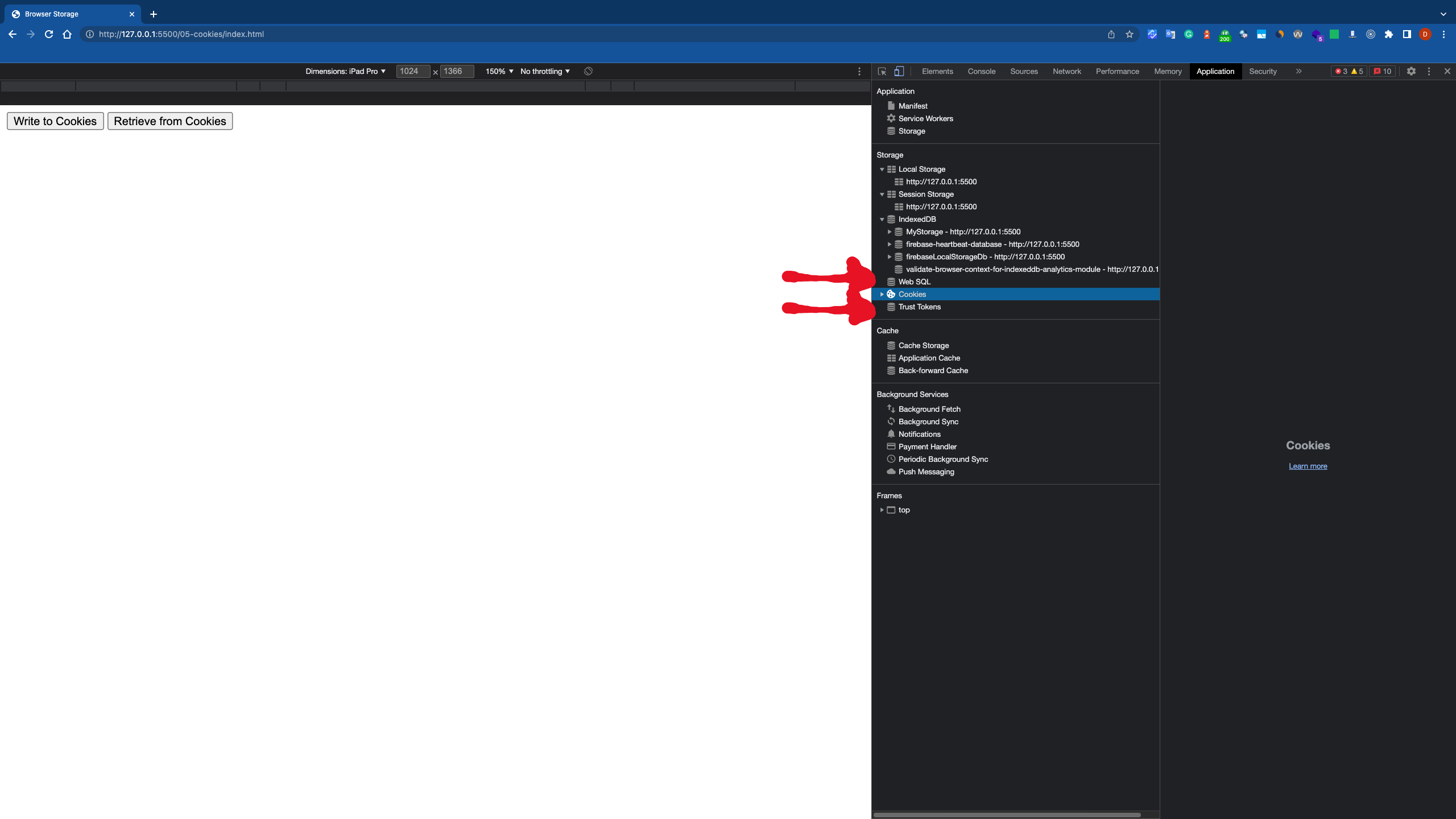

Elements
Console
Sources
Network
Performance
1
top
-url:http://127.0.0.1:5500/04-indexeddb/in
Default levels
1 Issue: 1
uuid=f48d1d75-b7a2-406a-ab8a-971b7ec0606f; user={"age":28,"company":"ABN AMRO","languages":["English","Polish"],"name":"Daniel","residence":"Amsterdam"}
app.js:19
{"age":28,"company":"ABN AMRO","languages":["English","Polish"],"name":"Daniel","residence":"Amsterdam"}
app.js:26

- Advantages of cookies: we can set them to expire and we can send them to the server with HTTP request
  - Cookies we can set to expire after some time, not possible in others
  - Cookies are typically send to the server, which is different than the others, so it makes them a bit „special” (of course, server needs to be capable of handling cookies)
  - By default, cookies expire when the browser is being closed, the user can also delete cookies - no matter if they have the expiration date or not
    - **max-age** is in seconds, alternative is **expires** which should be a date
- In general, the API isn’t as pleasant as local/session Storage
  - It’s not possible to get a single item from a cookie, it’s all or nothing
  - **document.cookie** = ”” adds a new entry to the cookies, so it’ll not overwrite or clear the existing cookies data. It’ll add something to it
  - Can be accessed via document.cookie, all cookies, not only 1, it feels a bit unintuitive
- Cookies are only available if your page is being served by a real server, so it’ll not work from file://, different than session/localStorage/IndexedDB which does
- Security:
  - **HttpOnly** – the cookie is accessible only by the server, not by the browser, it’s extra security mechanism so those won’t be visible by using console.log(document.cookie)
  - **Secure** – only serverd via HTTPS
- Synchronous
- Size 4KB
- String key-value format

Privacy Concern

Contains information about browsing history

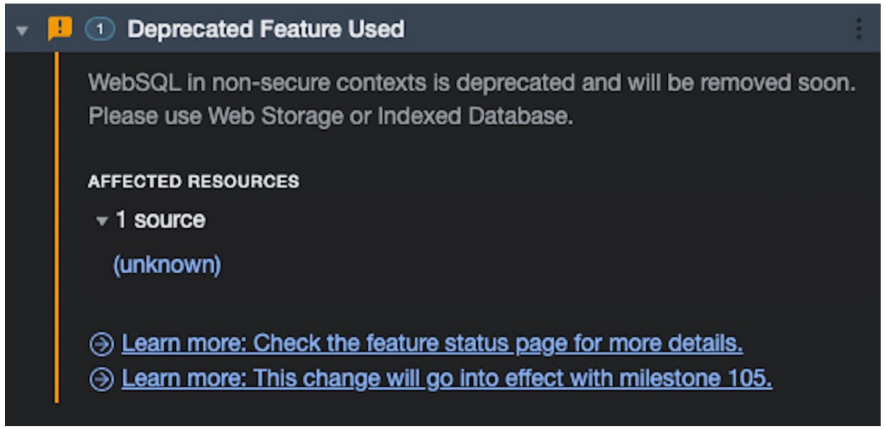
Final 2 Questions


Browser Storage
http://127.0.0.1:5500/05-cookies/index.html
Dimensions: iPad Pro 1024 x 1366 150% No throttling
Write to Cookies
Retrieve from Cookies
Elements Console Sources Network Performance Memory Application Security
Application
Manifest
Service Workers
Storage
Storage
Local Storage
http://127.0.0.1:5500
Session Storage
http://127.0.0.1:5500
IndexedDB
MyStorage - http://127.0.0.1:5500
firebase-heartbeat-database - http://127.0.0.1:5500
firebaseLocalStorageDb - http://127.0.0.1:5500
validate-browser-context-for-indexeddb-analytics-module - http://127.0.0.1
Web SQL
Cookies
Trust Tokens
Cache
Cache Storage
Application Cache
Back-forward Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
top
Cookies
Learn more

Web SQL?

Web SQL deprecation and removal steps

- [✅ Done.] Web SQL was deprecated and removed for **third-party contexts** in **Chromium 97** (Tue, Jan 4, 2022).
- [✅ Done.] Web SQL access in **insecure contexts** was deprecated as of **Chromium 105** (Tue, Aug 30, 2022) at which time a warning message was shown in the Chrome DevTools Issue panel.



- [📍 We are here.] Web SQL access in **insecure contexts** is no longer available as of **Chromium 110** (Tue, Feb 7, 2023). An **enterprise policy** to keep using the feature is available from **Chromium 110** (Tue, Feb 7, 2023) to **Chromium 111** Tue, Mar 7, 2023.
- [🔮 In the future.] The final step will be to remove Web SQL completely **in all contexts**, but no date has been set for this step yet.

Web SQL?

Deprecated

Trust Tokens?

Why do we need Trust Tokens?

The web needs ways to establish trust signals which show that a user is who they say they are, and not a bot pretending to be a human, or a malicious third-party defrauding a real person or service. Fraud protection is particularly important for advertisers, ad providers, and CDNs.

Trust Tokens?

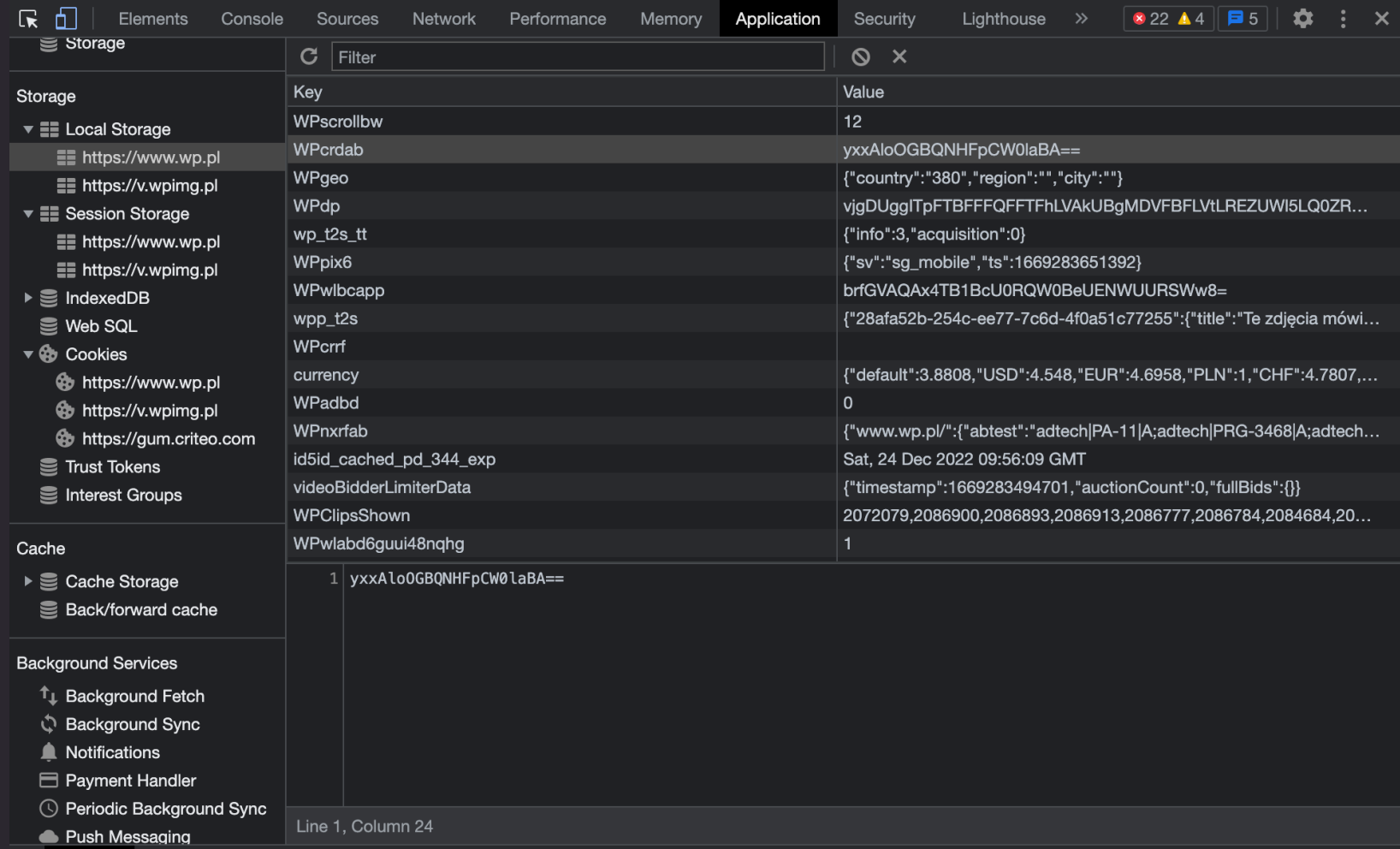
Difficult to demo + only Chromium-based browsers

Summary

Option	Cookies	localStorage	sessionStorage	IndexedDB
Goal	Manage user preferences or basic user data	Manage user preferences or basic user data	Manage user preferences or basic user data	Manage complex data (most sophisticated)
Data durability	Can be cleared by user and browser	Can be cleared by user and browser	Can be cleared by user and browser	Can be cleared by user and browser
Size	4KB	5MB	5MB	80% Disk Space
Format	String (key-value)	String (key-value)	String (key-value)	Object Store
API	A bit clunky	Easy & Intuitive	Easy & Intuitive	A bit clunky
A/synchronous	Synchronous	Synchronous	Synchronous	Asynchronous
Summary	**A bit clunky to use, quite versatile, sent to server, bad for complex data**	**Easy to use, quite versatile, bad for complex data**	**Easy to use, quite versatile, bad for complex data**	**A bit clunky to use, great for complex (non-ciritcal) data, good performance**

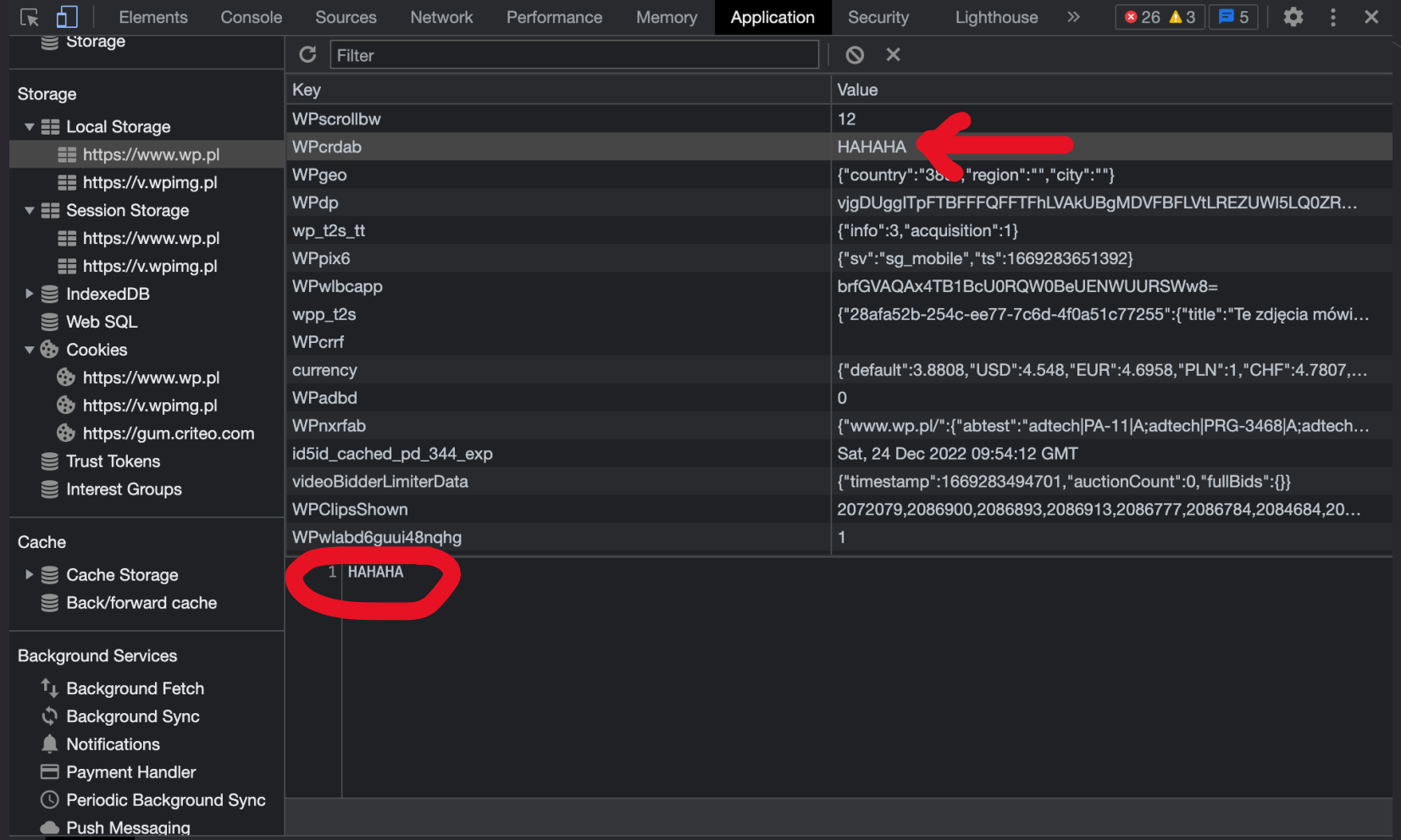
Option	Cookies	localStorage	sessionStorage	IndexedDB
Goal	Manage user preferences or basic user data	Manage user preferences or basic user data	Manage user preferences or basic user data	Manage complex data (most sophisticated)
Data durability	Can be cleared by user and browser	Can be cleared by user and browser	Can be cleared by user and browser	Can be cleared by user and browser
Size	4KB	5MB	5MB	80% Disk Space
Format	String (key-value)	String (key-value)	String (key-value)	Object Store
API	A bit clunky	Easy & Intuitive	Easy & Intuitive	A bit clunky
A/synchronous	Synchronous	Synchronous	Synchronous	Asynchronous
Summary	**A bit clunky to use, quite versatile, sent to server, bad for complex data**	**Easy to use, quite versatile, bad for complex data**	**Easy to use, quite versatile, bad for complex data**	**A bit clunky to use, great for complex (non-ciritcal) data, good performance**

Real-world Examples



Manipulating Browser Storage

Random Website



Manipulating Browser Storage

Random Website

- It doesn’t replace server side database and storing data on the server
- Preferences – Cookies/localStorage
- Tracking – Cookies
- Session management – Cookies, but shopping carts/game scores also could localStorage
- Caching – localStorage
- Separate storage across tabs (e.g., flight ticket) – sessionStorage
- UUID – sessionStorage
- Offline complex App (calendar, TODO, Game, etc.) – IndexedDB

Q&A

Thanks