



“

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 + IndexedDB
+ Cookie

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

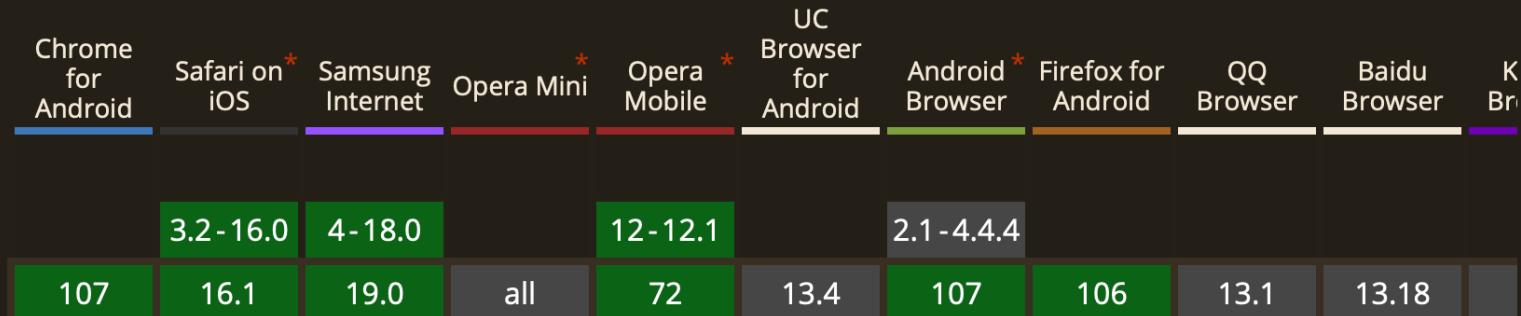
Usage
Global

% of all users
97.52%

Current aligned

Usage relative Date relative

Filtered All



Can I use

localStorage



Settings

2 results found

Caniuse (1) MDN (1)

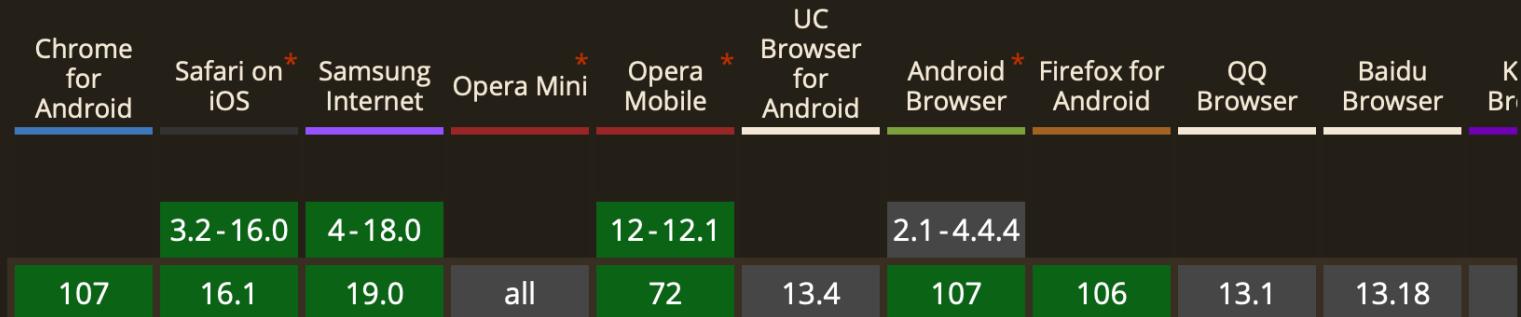
Window API: localStorage

Current aligned Usage relative Date relative Filtered All

Usage
Global

% of all users

97.52%



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

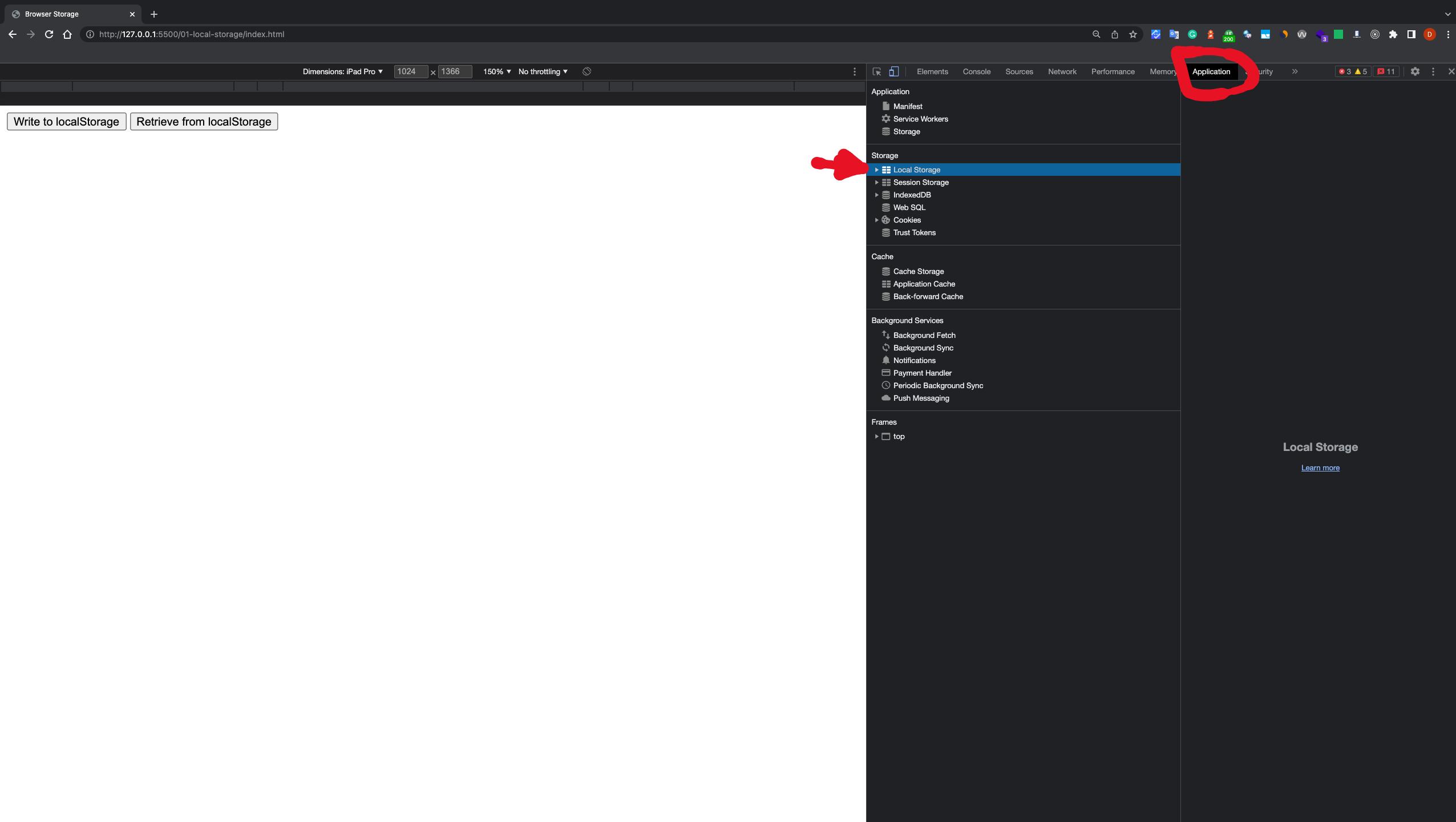
- ↑ Background Fetch
 - ⟳ Background Sync
 - 🔔 Notifications
 - ✉ Payment Handler
 - ⌚ Periodic Background Sync
 - 推送 Push Messaging

frames

- > top

Local Storage

[Learn more](#)



Browser Storage

Dimensions: iPad Pro ▾ 1024 x 1366 150% No throttling ▾

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

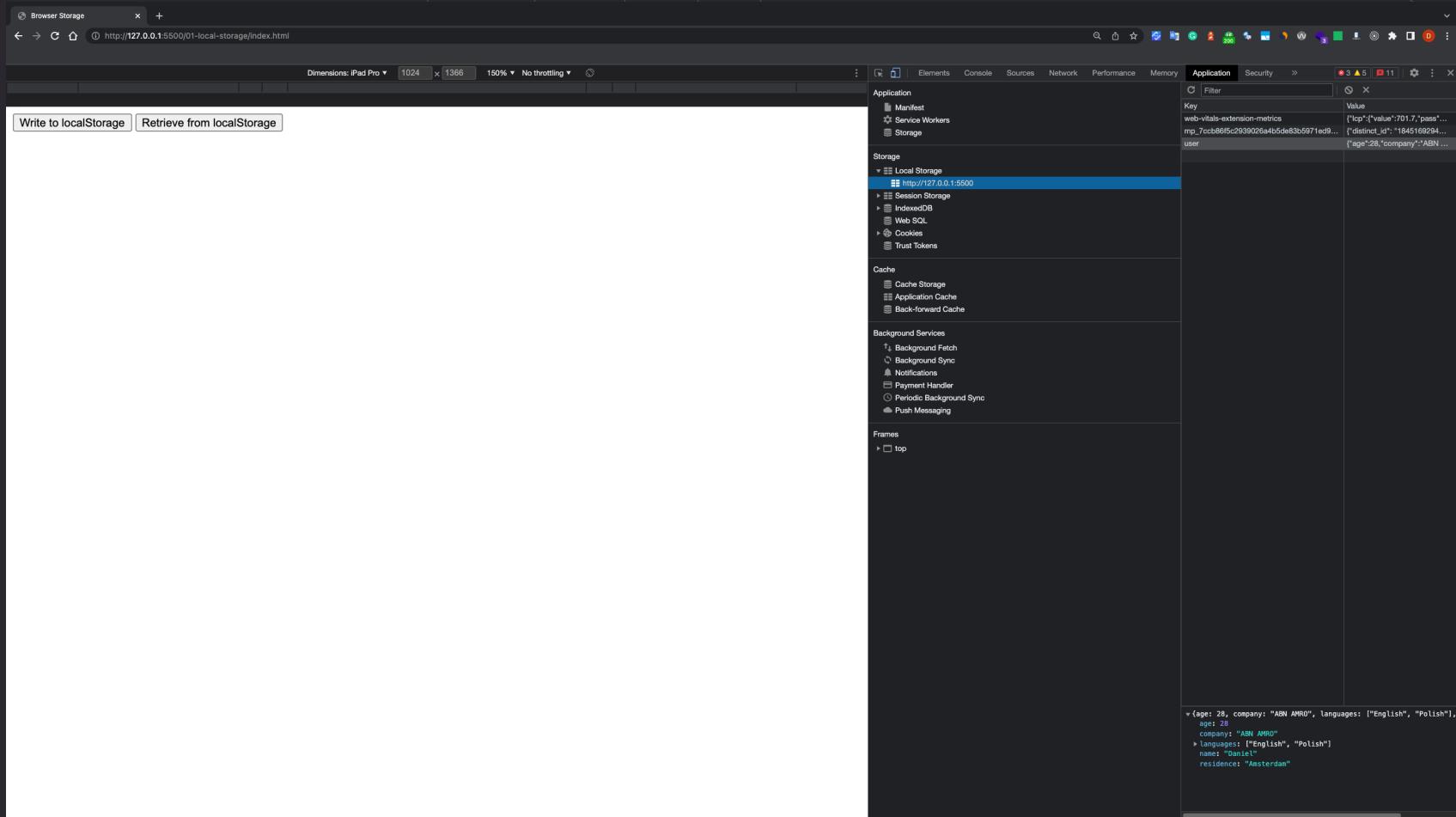
Frames

- top

Key Value

Key	Value
web-vitals-extension-metrics	{"lcp": {"value": 701.7, "pass": ...}}
mp_7ccb86f5c2939026a4b5de83b5971ed9...	{"distinct_id": "1845169294..."} {"age": 28, "company": "ABN ..."} {"name": "Daniel", "residence": "Amsterdam"}
user	

age: 28, company: "ABN AMRO", languages: ["English", "Polish"],
age: 28
company: "ABN AMRO"
languages: ["English", "Polish"]
name: "Daniel"
residence: "Amsterdam"



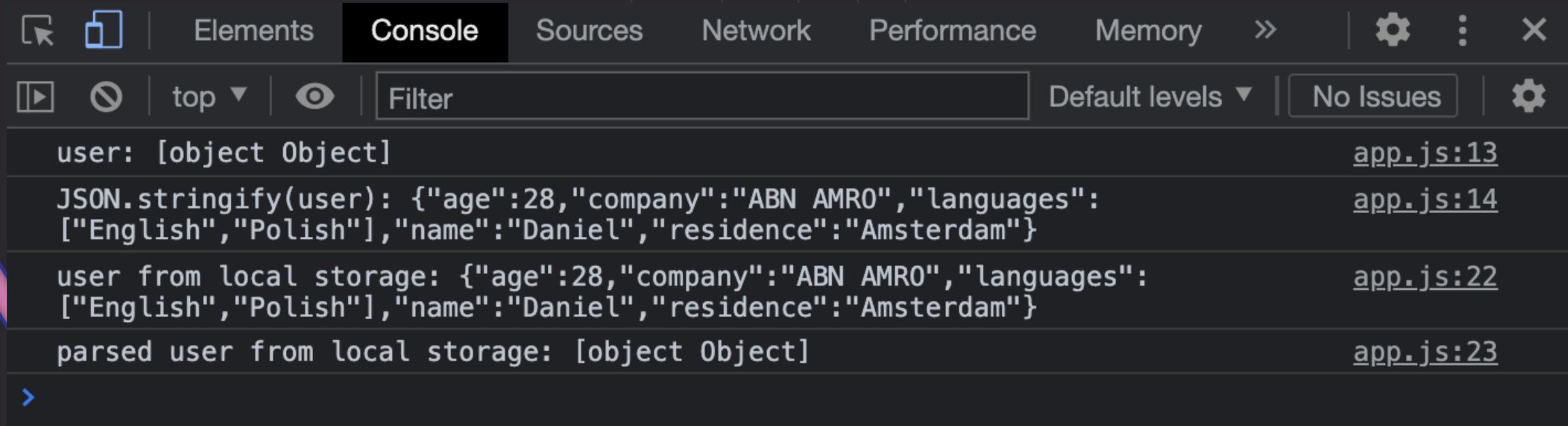
localStorage demo

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() ???



The screenshot shows the Chrome DevTools Console tab. The console output is as follows:

```
user: [object Object]                                         app.js:13
JSON.stringify(user): {"age":28,"company":"ABN AMRO","languages": ["English","Polish"],"name":"Daniel","residence":"Amsterdam"}   app.js:14
user from local storage: {"age":28,"company":"ABN AMRO","languages": ["English","Polish"],"name":"Daniel","residence":"Amsterdam"}   app.js:22
parsed user from local storage: [object Object]                app.js:23
```

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

The screenshot shows a browser's developer tools console tab active. The console output is as follows:

```
user: [object Object] app.js:13
JSON.stringify(user): {"age":28,"company":"ABN AMRO","languages": ["English","Polish"],"name":"Daniel","residence":"Amsterdam"} app.js:14
user from local storage: {"age":28,"company":"ABN AMRO","languages": ["English","Polish"],"name":"Daniel","residence":"Amsterdam"} app.js:22
parsed user from local storage: [object Object] app.js:23
```

Two specific lines of output are highlighted with red boxes: "user: [object Object]" at line 13 and "parsed user from local storage: [object Object]" at line 23.

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

97.52%

Global

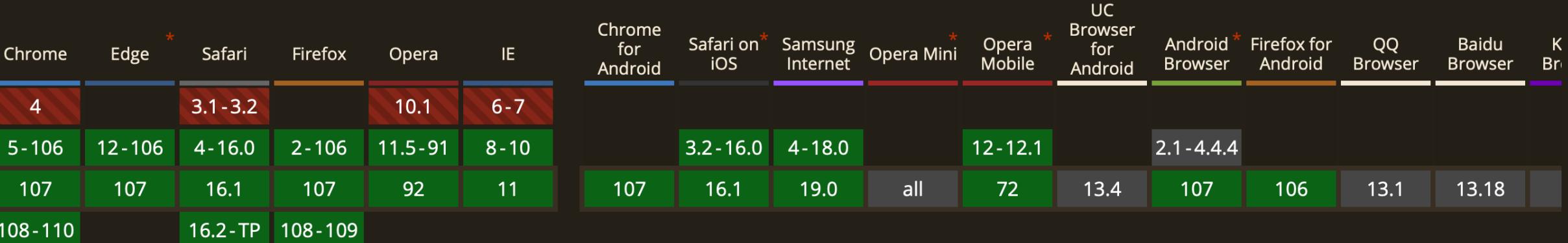
Current aligned

Usage relative

Date relative

Filtered

All



Can I use

sessionStorage

[?](#)[Settings](#)

2 results found

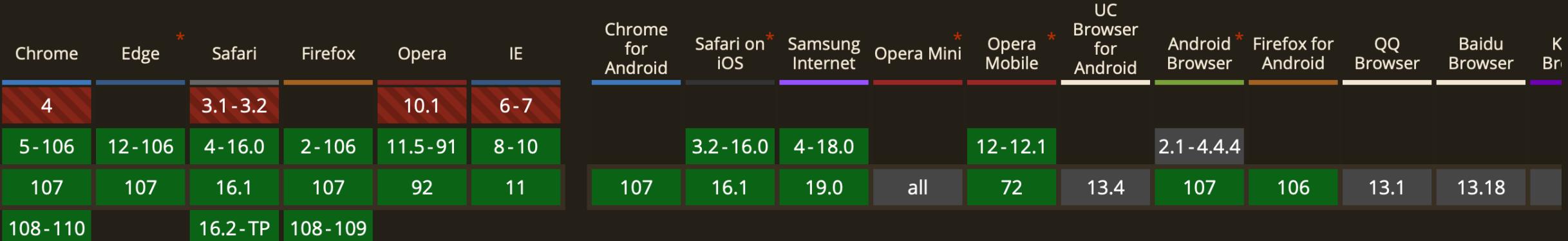
 Caniuse (1) MDN (1)

Window API: sessionStorage

Usage
Global

% of all users

97.52%

[Current aligned](#) [Usage relative](#) [Date relative](#) [Filtered](#) [All](#) [⚙️](#)

Write to sessionStorage

Retrieve from sessionStorage

pplication

- 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

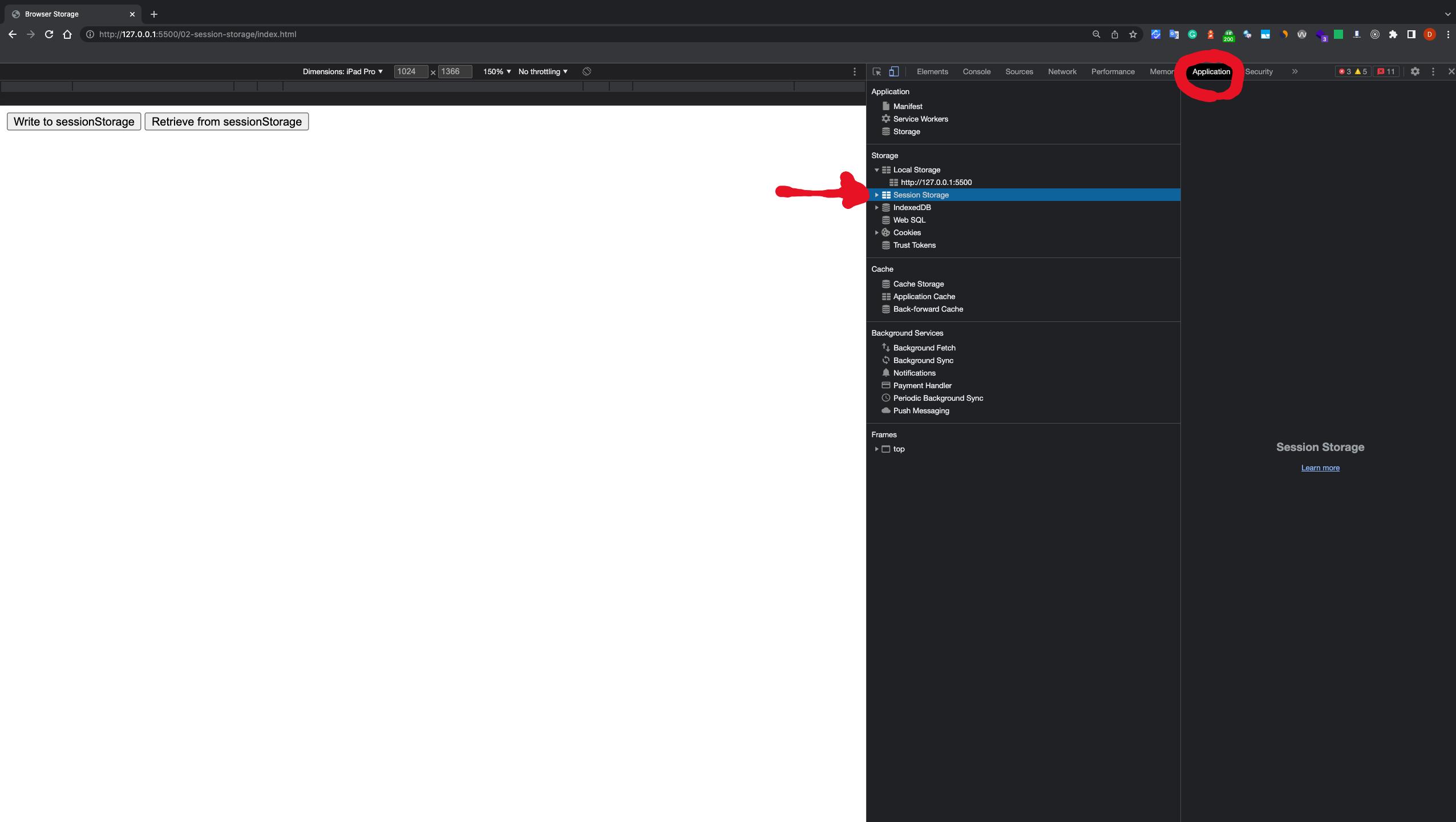
- Background Fetch
- Background Sync
- Notifications
- Payment Handler
- Periodic Background Sync
- Push Messaging

rames

> top

Session Storage

[Learn more](#)



Browser Storage

Dimensions: iPad Pro ▾ 1024 x 1366 150% No throttling ▾

Write to sessionStorage Retrieve from sessionStorage

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

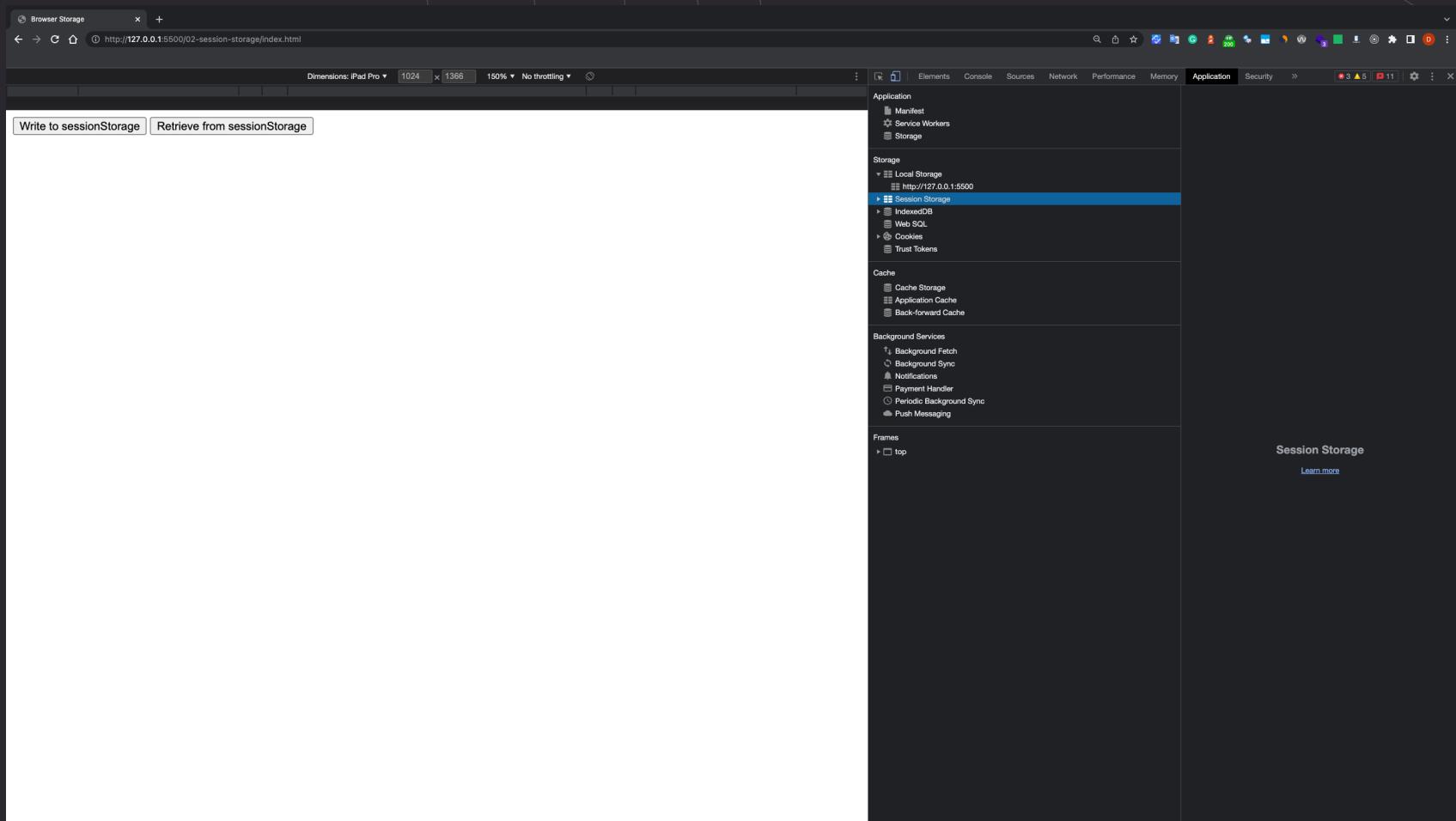
- top

Filter

Key	Value
IsThisFirstTime_Log_From_LiveServer	true
uuid	f48d1d75-b7a2-406a-ab8a-971b7ec0606f

Line 1, Column 1

The screenshot shows the Chrome DevTools Application panel. On the left, there's a toolbar with various icons like Refresh, Stop, and Reload. Below that, a header bar displays 'Dimensions: iPad Pro ▾ 1024 x 1366 150% No throttling ▾'. The main area has two tabs: 'Write to sessionStorage' and 'Retrieve from sessionStorage', with 'Write to sessionStorage' currently active. The left sidebar has sections for Application, Storage, Cache, Background Services, and Frames. The 'Storage' section is expanded, showing Local Storage and Session Storage. Under Session Storage, there's an entry for 'http://127.0.0.1:5500'. The right panel shows a table of stored data with columns for Key and Value. Two specific rows are highlighted with red circles: 'IsThisFirstTime_Log_From_LiveServer' with the value 'true' and 'uuid' with the value 'f48d1d75-b7a2-406a-ab8a-971b7ec0606f'. A red arrow points from the 'Session Storage' entry in the sidebar to the 'uuid' row in the table.



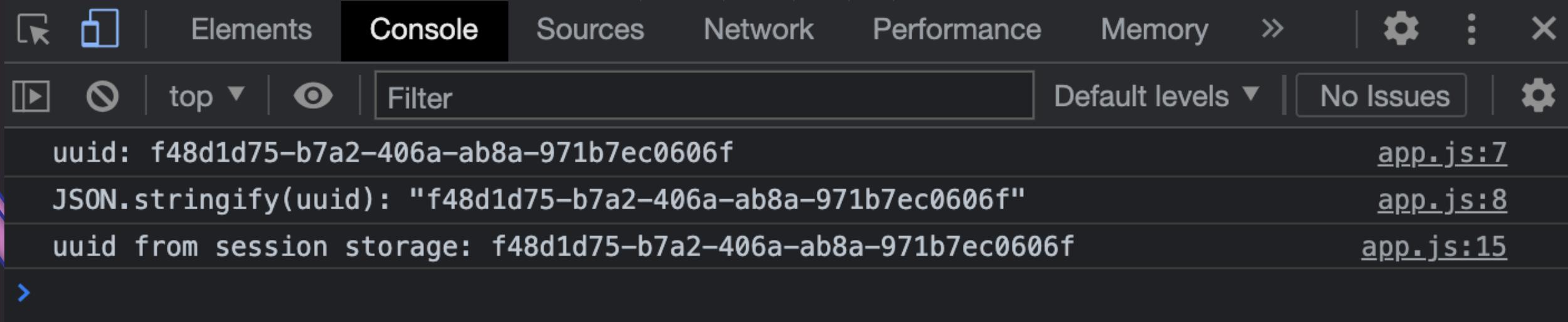
sessionStorage demo

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     console.log(`uuid from session storage: ${uuidFromStorage}`);
15 });
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     console.log('uuid from session storage: ' + uuidFromStorage);
15 });
16
17
```

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

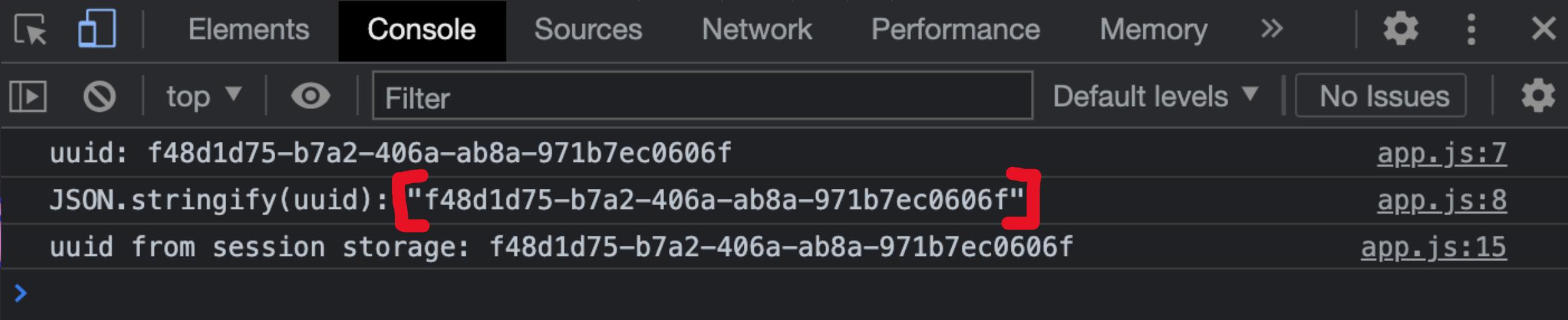


A screenshot of a browser's developer tools Console tab. The tab bar includes Elements, Console (which is selected), Sources, Network, Performance, Memory, and other options. Below the tab bar are controls for play/pause, step, and filter, along with dropdowns for 'Default levels' and 'No Issues'. The main area displays three lines of console output:

```
uuid: f48d1d75-b7a2-406a-ab8a-971b7ec0606f app.js:7
JSON.stringify(uuid): "f48d1d75-b7a2-406a-ab8a-971b7ec0606f" app.js:8
uuid from session storage: f48d1d75-b7a2-406a-ab8a-971b7ec0606f app.js:15
```

The output shows that the string representation of a UUID remains identical when converted to a string using `JSON.stringify()` and parsed back using `JSON.parse()`, which is expected for primitive types like strings.

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



A screenshot of a browser's developer tools Console tab. The tab bar includes Elements, Console (which is active), Sources, Network, Performance, Memory, and other options. Below the tab bar are controls for play/pause, stop, and filter, along with dropdowns for Default levels and No Issues. The main area displays three lines of console output:

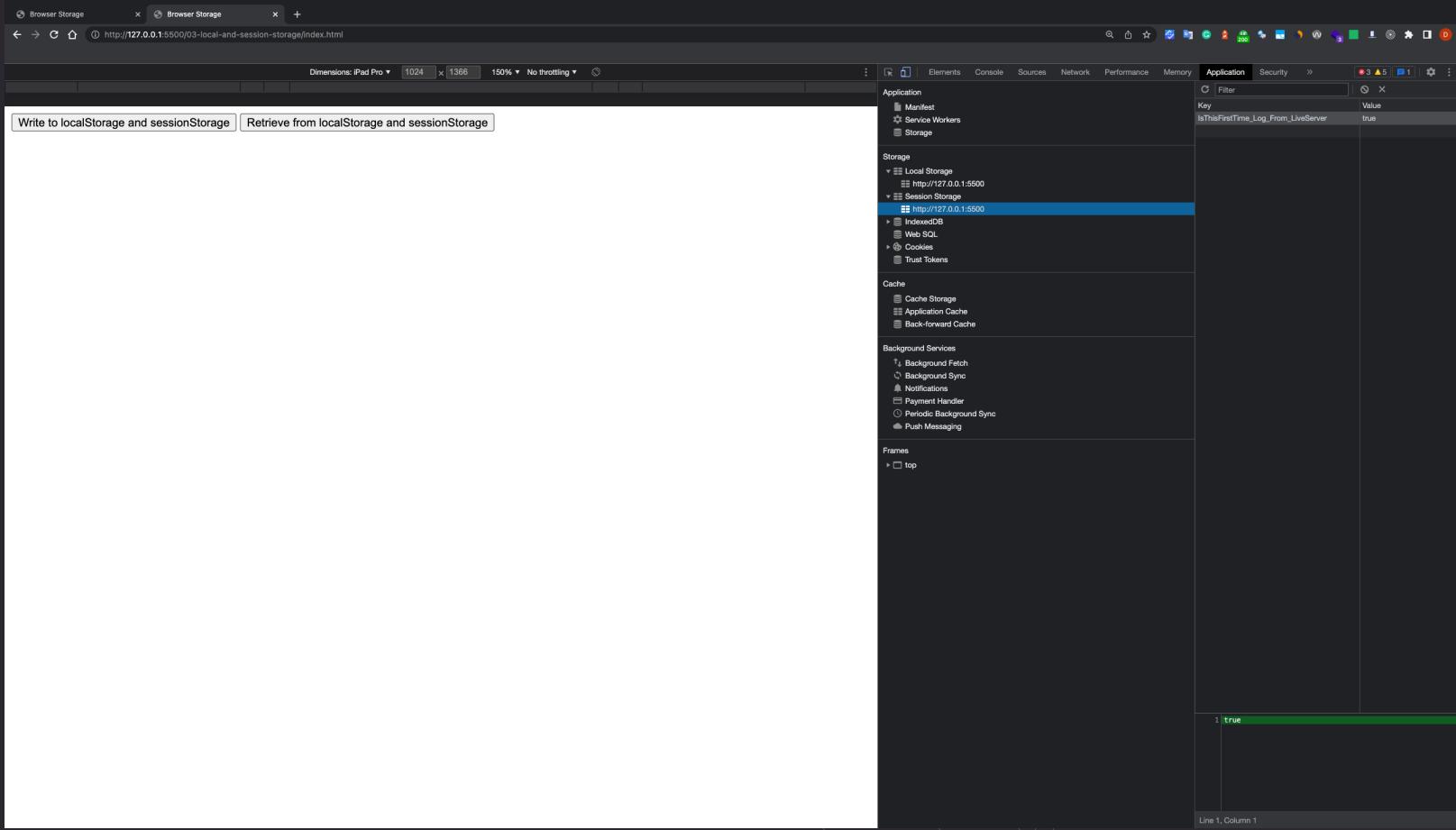
```
uuid: f48d1d75-b7a2-406a-ab8a-971b7ec0606f
JSON.stringify(uuid): ["f48d1d75-b7a2-406a-ab8a-971b7ec0606f"]
uuid from session storage: f48d1d75-b7a2-406a-ab8a-971b7ec0606f
```

The string "JSON.stringify(uuid): ["f48d1d75-b7a2-406a-ab8a-971b7ec0606f"]" is highlighted with a large red bracket underneath it.

localStorage

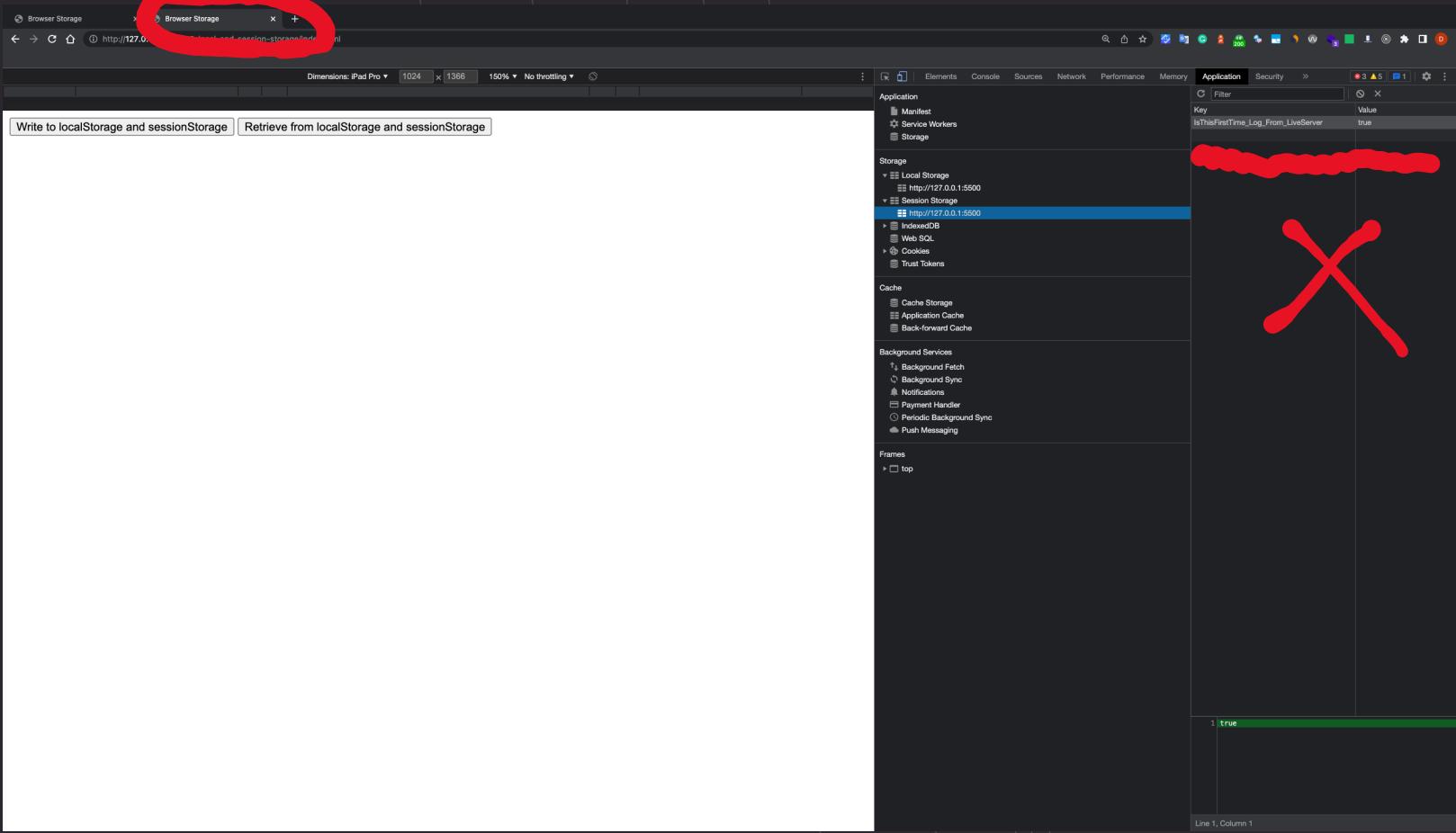
vs

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

- **localStorage – it never clears unless user/browser clears it**
- **sessionStorage – data only in active tab, when new tab opens/browser close it's gone**
- Both:
 - Are synchronous, so no 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 store: user preference, session ID of user

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)



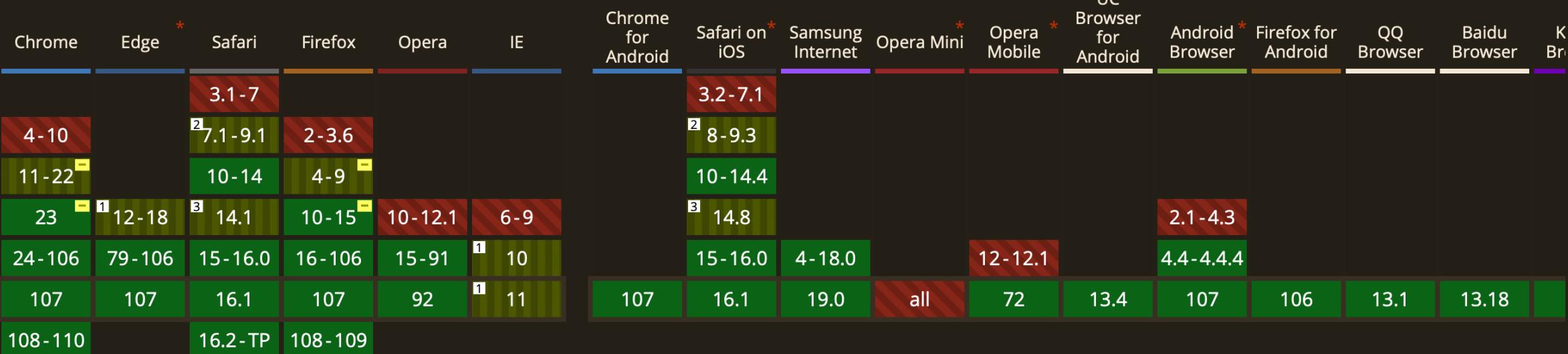
IndexedDB

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 



Can I use

IndexedDB

?  Settings

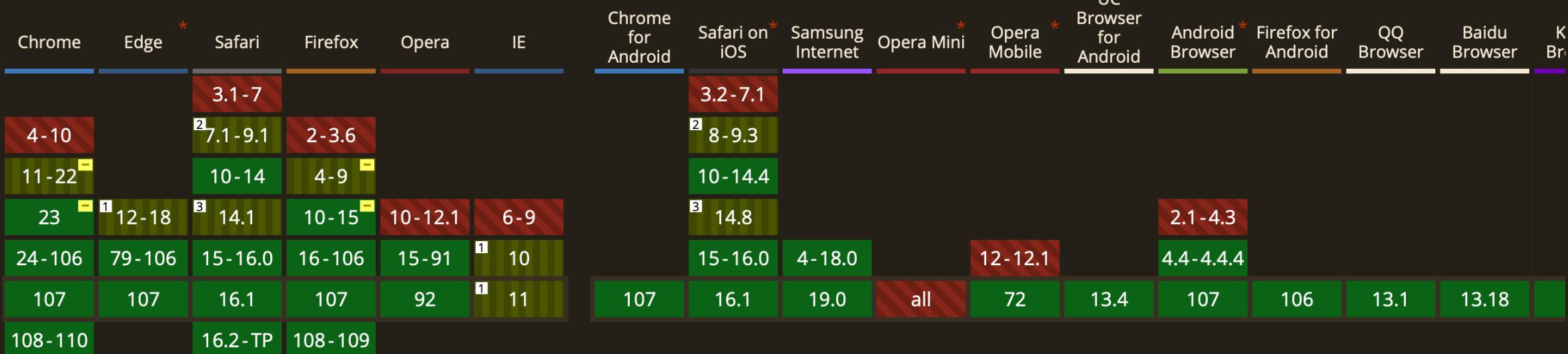
4 results found

Caniuse (2) MDN (2)

IndexedDB

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

Current aligned Usage relative Date relative Filtered All 



Write to IndexedDB Retrieve from IndexedDB

pplication

- 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

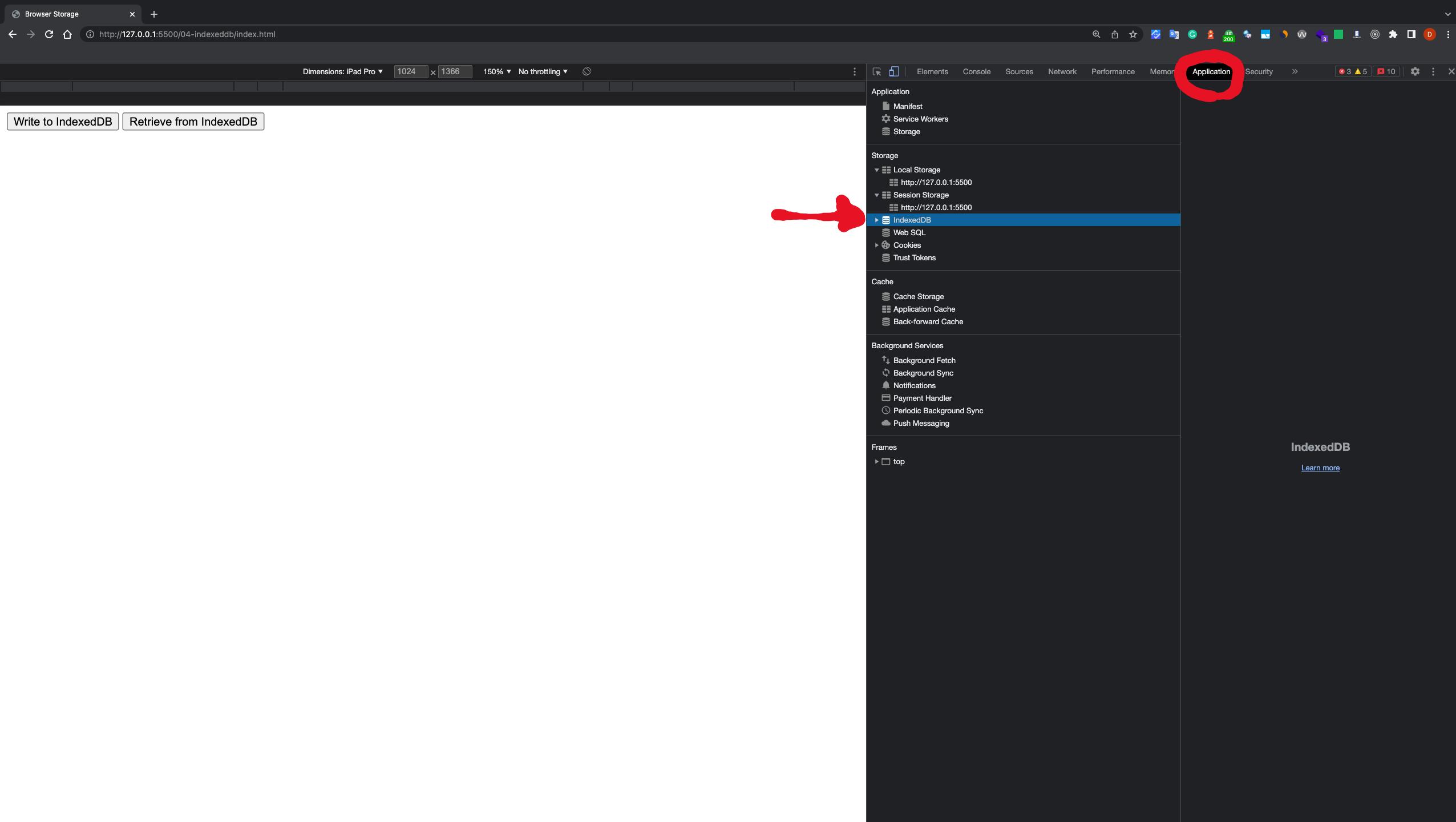
-  Background Fetch
-  Background Sync
-  Notifications
-  Payment Handler
-  Periodic Background Sync
-  Push Messaging

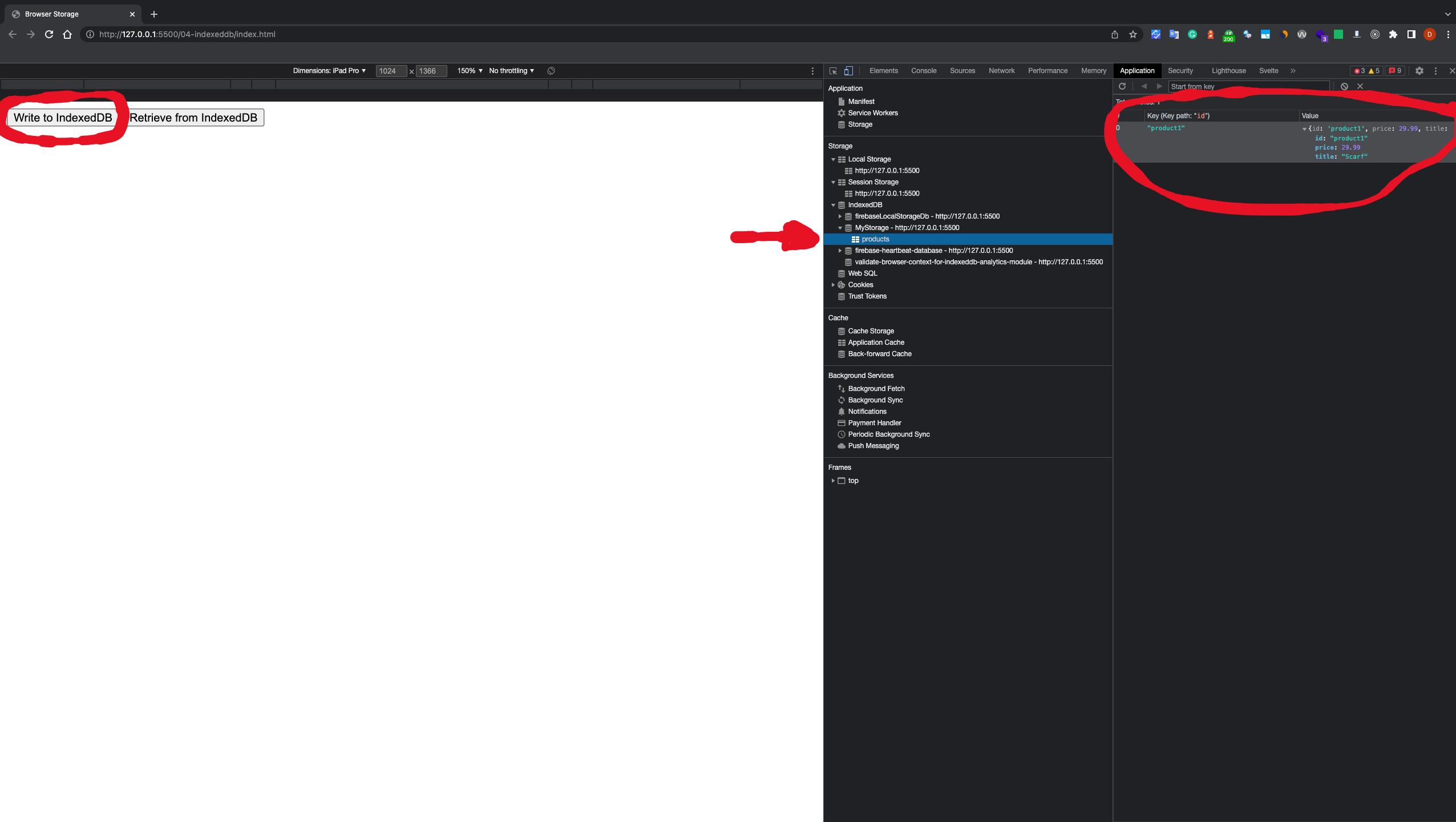
rames

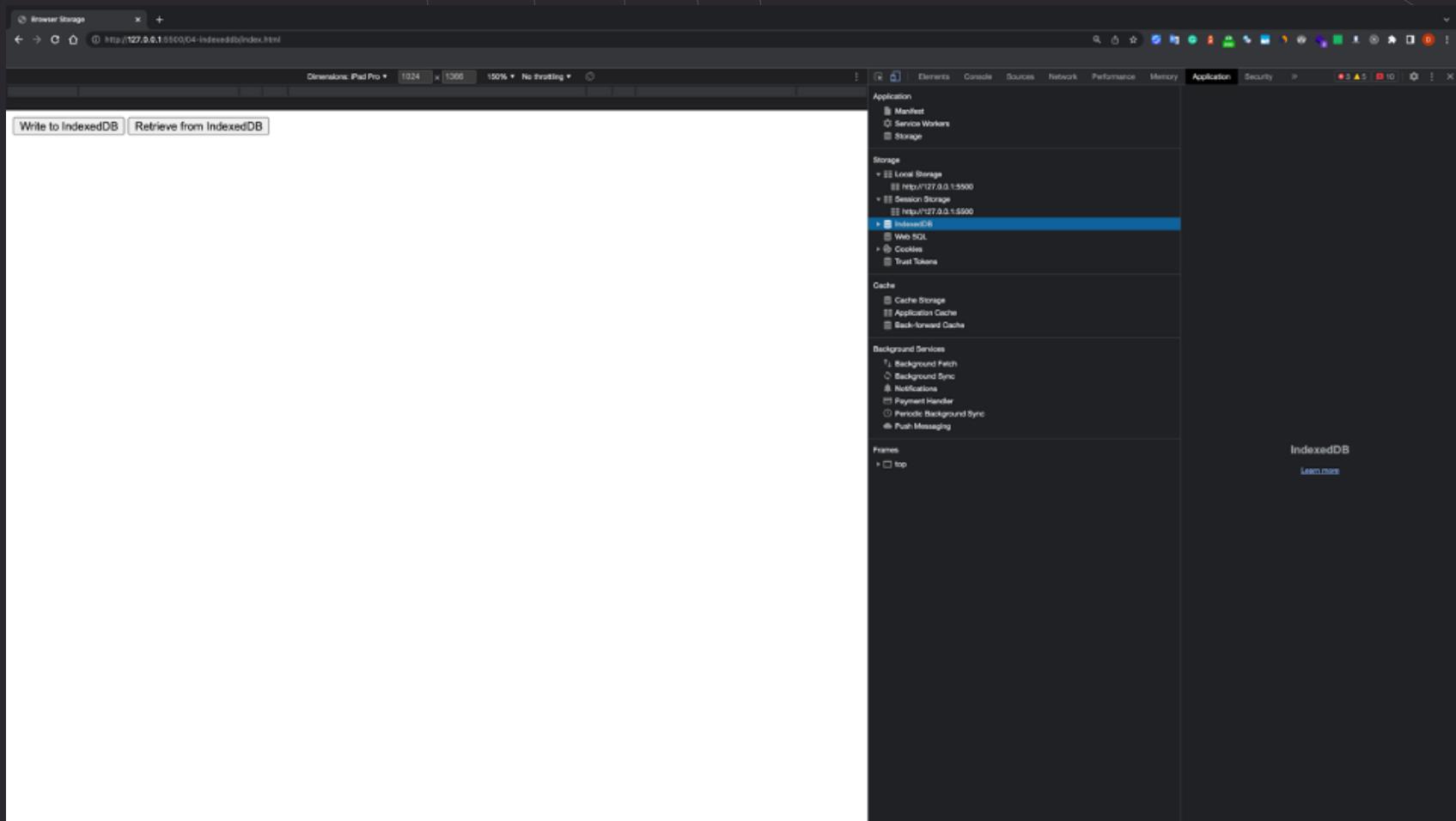
> top

IndexedDB

[Learn more](#)







IndexedDB demo

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

```
1 const writeToButton = document.getElementById('write-to-button');
2 const retrieveFromButton = document.getElementById('retrieve-from-button');
3
4 const databaseRequest = indexedDB.open('MyStorage', 1);
5 let database;
6
7 databaseRequest.onsuccess = function (event) {
8     database = event.target.result;
9     console.log('database on created below:');
10    console.log(database);
11 };
12
13 databaseRequest.onupgradeneeded = function (event) {
14     database = event.target.result;      You, now . Uncommitted changes
15     database.createObjectStore('products', { keyPath: 'id' });
16 }
17
18 databaseRequest.onerror = function (event) {
19     console.log('ERROR!');
20     console.log(event);
21 };
22
23 writeToButton.addEventListener('click', () => {
24     let productsStore = database
25         .transaction('products', 'readwrite')
26         .objectStore('products');
27     productsStore.add({
28         id: 'product1',
29         price: 29.99,
30         title: 'Scarf',
31     });
32     console.log('database after adding item below:');
33     console.log(database);
34 });
35
36 retrieveFromButton.addEventListener('click', () => {
37     const productsStore = database
38         .transaction('products', 'readwrite')
39         .objectStore('products');
40     const request = productsStore.get('product1');
41     console.log('request below:');
42     console.log(request);
43
44     request.onsuccess = function () {
45         console.log('request in onsuccess below:');
46         console.log(request);
47         console.log('request.result in on success:');
48         console.log(request.result);
49     }
50 });
```

```
1 const writeToButton = document.getElementById('write-to-button');
2 const retrieveFromButton = document.getElementById('retrieve-from-button');
3
4 const databaseRequest = indexedDB.open('MyStorage', 1);
5 let database;
6
7 databaseRequest.onsuccess = function (event) {
8     database = event.target.result;
9     console.log('database on created below:');
10    console.log(database);
11 };
12
13 databaseRequest.onupgradeneeded = function (event) {
14     database = event.target.result;          You, now . Uncommitted changes
15     database.createObjectStore('products', { keyPath: 'id' });
16 }
17
18 databaseRequest.onerror = function (event) {
19     console.log('ERROR! ');
20     console.log(event);
21 };
22
23 writeToButton.addEventListener('click', () => {
24     let productsStore = database
25         .transaction('products', 'readwrite')
26         .objectStore('products');
27     productsStore.add({
28         id: 'product1',
29         price: 29.99,
30         title: 'Scarf',
31     });
32     console.log('database after adding item below:');
33     console.log(database);
34 });
35
36 retrieveFromButton.addEventListener('click', () => {
37     const productsStore = database
38         .transaction('products', 'readwrite')
39         .objectStore('products');
40     const request = productsStore.get('product1');
41     console.log('request below:');
42     console.log(request);
43
44     request.onsuccess = function () {
45         console.log('request in onsuccess below:');
46         console.log(request);
47         console.log('request.result in on success:');
48         console.log(request.result);
49     }
50 });
```

database on created below: app.js:9

▶ *IDBDatabase {name: 'MyStorage', version: 1, objectStoreNames: DOMStringList, onabort: null, onclose: null, ...}* app.js:10

⚠ DevTools failed to load source map: Could not load content for [chrome-extension://mohlkfkdfnjciellfjppgfppfkchlobp/js/inject.js.map](#): HTTP error: status code 404,
net::ERR_UNKNOWN_URL_SCHEME

database after adding item below: app.js:27

▶ *IDBDatabase {name: 'MyStorage', version: 1, objectStoreNames: DOMStringList, onabort: null, onclose: null, ...}* app.js:28

request below: app.js:36

▶ *IDBRequest {source: IDBObjectStore, transaction: IDBTransaction, readyState: 'pending', onsuccess: null, onerror: null}* app.js:37

request in onsuccess below: app.js:40

▶ *IDBRequest {result: {...}, error: null, source: IDBObjectStore, transaction: IDBTransaction, readyState: 'done', ...}* app.js:41

request.result in on success: app.js:42

▶ *{id: 'product1', price: 29.99, title: 'Scarf'}* app.js:43

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://mohlkfkdfnjciellfjppgfppfkchlobp/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

97.92% 

Current aligned Usage relative Date relative

Filtered All 



Can I use

Document API: cookie

?  Settings

2 results found

Caniuse (1) MDN (1)

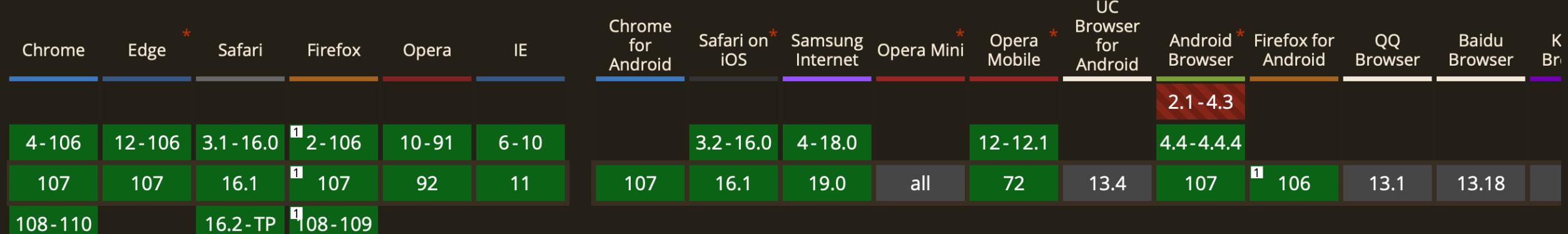
Document API: cookie

Usage
Global

% of all users

97.92% 

Current aligned Usage relative Date relative Filtered All 





Write to Cookies Retrieve from Cookies

Dimensions: iPad Pro ▾ 1024 × 1366 150% ▾ No throttling ▾

Application Elements Console Sources Network Performance Memory Application Security > [3 5] [10] [200]

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://127.0.0.1:5500
- Web SQL
- Cookies
- Trust Tokens

Cache

- Cache Storage
- Application Cache
- Back-forward Cache

Background Services

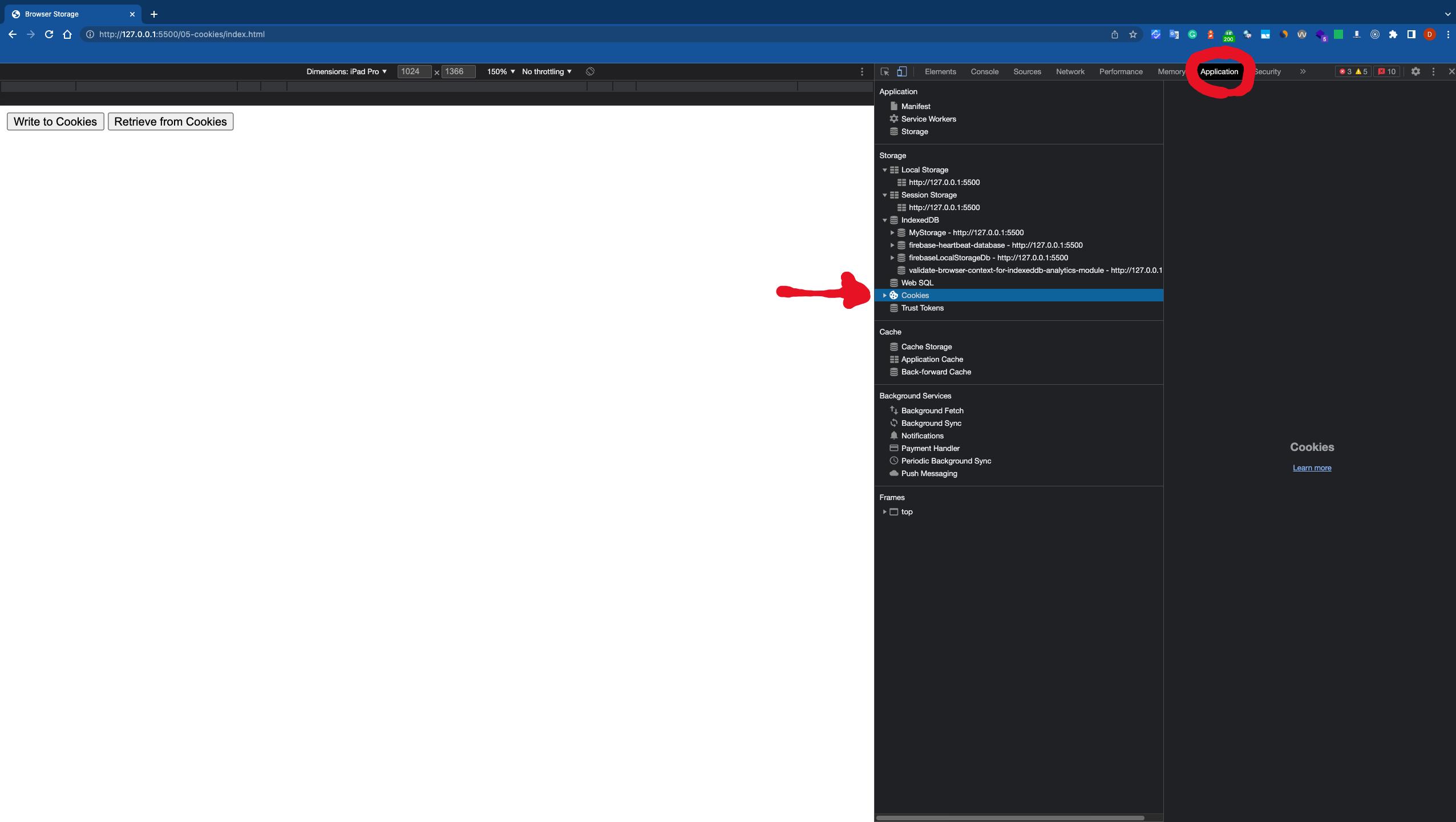
- Background Fetch
- Background Sync
- Notifications
- Payment Handler
- Periodic Background Sync
- Push Messaging

Frames

- top

Cookies

[Learn more](#)



Browser Storage

http://127.0.0.1:5500/05-cookies/index.html

Dimensions: iPad Pro 1024 x 1366 150% No throttling

Application Security

Only show cookies with an issue

Write to Cookies

Retrieve from Cookies

Name Value D... P... E... S... H... S... S... M...

user {"age":28,"company": "ABN AMRO","languages":["English","Polish"],"name": "Daniel","residence": "Amsterdam"} 12... /... S. 1...

uuid f48d1d75-b7a2-406... 12... /... 2. 40

Manifest

Service Workers

Storage

Local Storage

Session Storage

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://127.0.0.1:5500

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

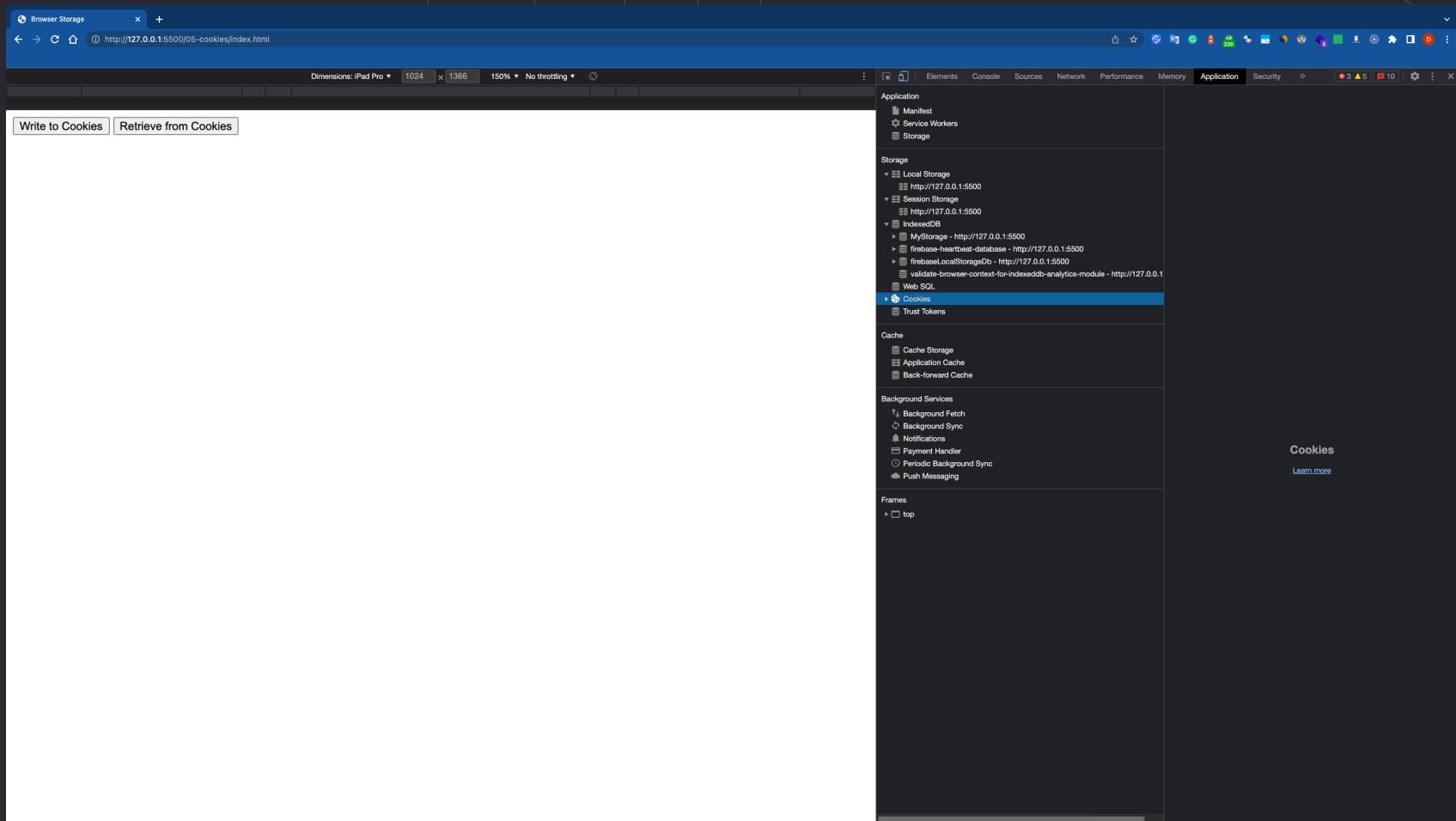
Push Messaging

Frames

top

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>

```
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
```

```
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
```

}

} 22

Elements **Console** Sources Network Performance » 1 | 1 | 1 | X

-url:http://127.0.0.1:5500/04-indexeddb/index.html Default levels ▾ | 1 Issue: 1 | 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
```

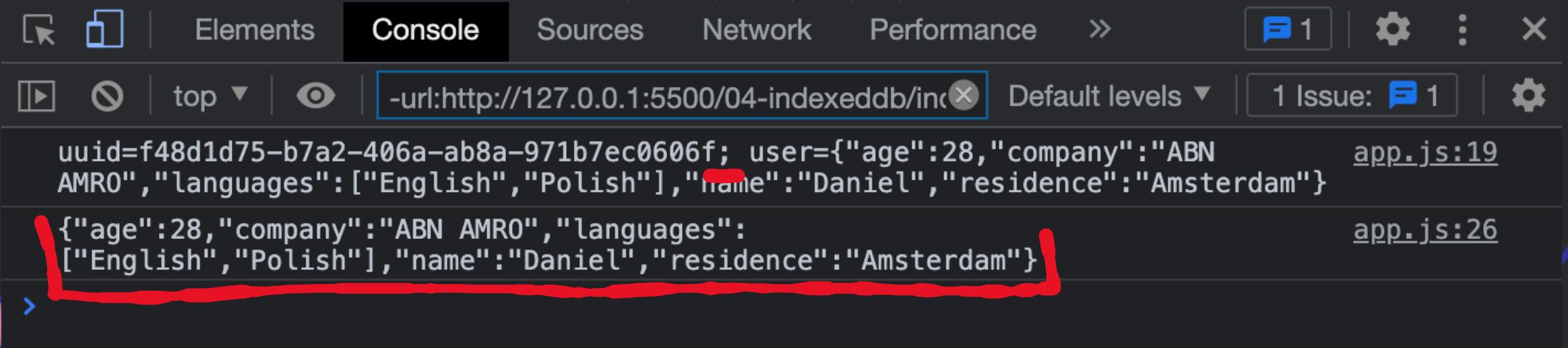
Elements Console Sources Network Performance » 1 Issue: 1

-url:http://127.0.0.1:5500/04-indexeddb/index.html

```
uuid=f48d1d75-b7a2-406a-ab8a-971b7ec0606f; user={"age":28,"company":"ABN AMRO","languages":["English","Polish"],"name":"Daniel","residence":"Amsterdam"}  
{"age":28,"company":"ABN AMRO","languages":["English","Polish"],"name":"Daniel","residence":"Amsterdam"}
```

```
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
```

222



The screenshot shows a browser's developer tools console tab selected. The URL bar contains `-url:http://127.0.0.1:5500/04-indexeddb/index.html`. The console output displays two JSON objects:

```
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
```

A red box highlights the second JSON object in the console output.

- 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/IndexdB 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 served via HTTPS
- Synchronous
- Size 4KB
- String key-value format

COOKIELESS FUTURE OF ADVERTISING

1ST PARTY



VS

3RD PARTY



Privacy Concern

Contains information about browsing history

Final 2 Questions

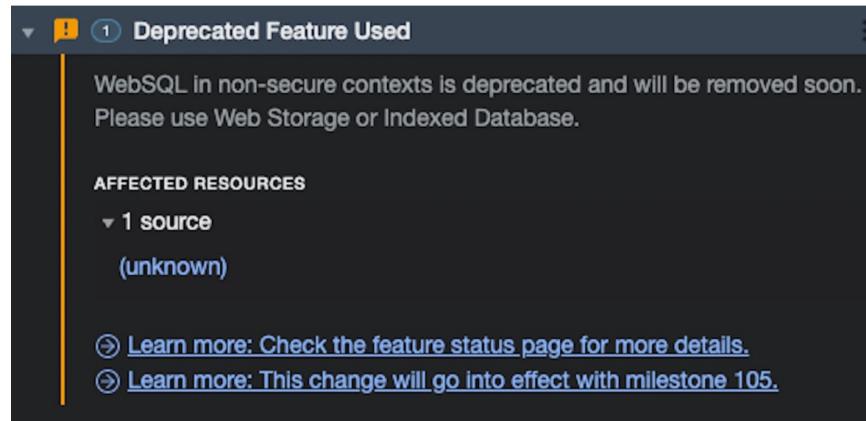


Cookies
[Learn more](#)

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-critical) 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-critical) data, good performance

Real-world Examples

The screenshot shows the Chrome DevTools interface with the Application tab selected. On the left, the Storage panel is open, displaying a tree view of storage types and their associated domains. The Local Storage section shows entries for 'https://www.wp.pl' and 'https://v.wpimg.pl'. The Session Storage section shows entries for both domains. The Cookies section shows entries for all three domains. Below these are sections for Cache, Background Services, and a list of issues.

Key	Value
WPscrollbw	12
WPcrdab	yxxAlo0GBQNHFPcW0laBA==
WPgeo	{"country":"380","region":"","city":""}
WPdp	vjgDUggITpFTBFFFQFFTfLVAkUBgMDVFBFLVtLREZUWI5LQ0ZR...
wp_t2s_tt	{"info":3,"acquisition":0}
WPpix6	{"sv":"sg_mobile","ts":1669283651392}
WPwlbcapp	brfGVAQAx4TB1BcU0RQW0BeUENWUUURSWw8=
wpp_t2s	{"28afa52b-254c-ee77-7c6d-4f0a51c77255":{"title":"Te zdjęcia mówią...}}
WPcrff	
currency	{"default":3.8808,"USD":4.548,"EUR":4.6958,"PLN":1,"CHF":4.7807,...}
WPadbd	0
WPnrxfab	{"www.wp.pl":{"abtest":"adtech PA-11 A;adtech PRG-3468 A;adtech ...}}
id5id_cached_pd_344_exp	Sat, 24 Dec 2022 09:56:09 GMT
videoBidderLimiterData	{"timestamp":1669283494701,"auctionCount":0,"fullBids":{}}
WPclipsShown	2072079,2086900,2086893,2086913,2086777,2086784,2084684,20...
WPwlabd6guui48nqhg	1

Manipulating Browser Storage

Random Website

Storage	Filter	
Storage		
▼ Local Storage		
https://www.wp.pl	WPscrollbw	12
https://v.wpimg.pl	WPcrdab	HAHAHA
▼ Session Storage		
https://www.wp.pl	WPgeo	{"country":"3b0","region":"","city":""}
https://v.wpimg.pl	WPdp	vjgDUggITpFTBFFFQFFTfLVAkUBgMDVFBFLVtLREZUWI5LQ0ZR...
▶ IndexedDB	wp_t2s_tt	{"info":3,"acquisition":1}
Web SQL	WPpix6	{"sv":"sg_mobile","ts":1669283651392}
Cookies	WPwbcapp	brfGVAQAx4TB1BcJ0RQW0BeUENWUUURSWw8-
https://www.wp.pl	wpp_t2s	{"28afa52b-254c-ee77-7c6d-4f0a51c77255":{"title":"Te zdjęcia mówią...}}
https://v.wpimg.pl	WPcrrf	
https://gum.criteo.com	currency	{"default":3.8808,"USD":4.548,"EUR":4.6958,"PLN":1,"CHF":4.7807,...}
Trust Tokens	WPadbd	0
Interest Groups	WPnxrfab	{"www.wp.pl":("abtest":"adtech PA-11 A;adtech PRG-3468 A;adtech...")}
Cache	id5id_cached_pd_344_exp	Sat, 24 Dec 2022 09:54:12 GMT
▶ Cache Storage	videoBidderLimiterData	{"timestamp":1669283494701,"auctionCount":0,"fullBids":{}}
Back/forward cache	WPClipsShown	2072079,2086900,2086893,2086913,2086777,2086784,2084684,20...
	WPwlabd6guui48nqhg	1
	1 HAHAHA	
Background Services		
Background Fetch		
Background Sync		
Notifications		
Payment Handler		
Periodic Background Sync		
Push Messaging		
⋮	Console	
⋮	Issues	

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 (also Cache Storage...)
- Separate storage across tabs (e.g., flight ticket) – sessionStorage
- UUID – sessionStorage
- Offline complex App (calendar, TODO, Game, etc.) – IndexedDB

Q&A

Thanks

