Everyone loves cache, right? Why wouldn’t you! In that case, let’s see how we can cache data with Svelte & Appwrite!

We're going to explore how we can create a Svelte store that caches our data, and gets dynamically updated.

Lucky for us, it’s pretty easy.

The TL;DR is:

* Create a store with set, subscribe, and load
* Set the data to the cache
* Update the cache by subscribing to the data changes
* Update the data dynamically by calling the load function in your component

## Let's set everything up 📦

Let’s start by creating an appwrite.js store to make our life easier:

import { Appwrite } from 'appwrite';

const server = {

 endpoint: import.meta.env.VITE\_APP\_ENDPOINT.toString(),

 project: import.meta.env.VITE\_APP\_PROJECT.toString(),

 collection: import.meta.env.VITE\_APP\_COLLECTION\_ID.toString()

};

const sdk = new Appwrite();

sdk.setEndpoint(server.endpoint).setProject(server.project);

export { sdk, server };

You can check a complete example [here](https://github.com/appwrite/demo-todo-with-svelte)

This allows us to access Appwrite's sdk from wherever we want.

(remember to add all the endpoints in the .env file at your project's root)

## The fun begins! 🥳

Now let's initialize another store where we will implement our caching:

import { sdk, server } from './appwrite'; //this is the store we created previously

import { writable } from 'svelte/store';

import { browser } from '$app/env'; // we'll use this to check if we are in the browser

function createDocumentsListStore(){

//Magic unicorns 🦄

}

export const documentsList = createDocumentsListStore();

Great! Let's work a little bit on that function. We want to create a writable store that can fetch our cached data, and is easy to update.

function createDocumentsListStore(){

 const { subscribe, set } = writable({

/\*Check if we're in the browser.

If we are check the session storage for a item named documents, and parse it. This is our response.

If we're not in the browser or if there is no item stored the response is null\*/

 response: browser ? JSON.parse(sessionStorage.getItem('documents')) : null

 });

 return {

 subscribe,

 set,

 load: async ( queries ,limit, offset) => {

 try {

 const response = sdk.database.listDocuments(

 server.collection, //we imported server from the appwrite.js store

 queries,

 limit,

 offset,

 undefined,

 undefined,

 ['created\_at'],

 ['DESC']

 );

 set({

 response

 });

 } catch (error) {

 //Handle the error

 }

 }

 };

}

So, with this function, we check if there is data in the session storage and set the response to it.

## Where's my cache? 😕

We have a small problem: we're yet to save anything in our cache!

Lucky for us it's pretty straight forward:

if (browser) {

 documentsList.subscribe((n) => sessionStorage?.setItem('documents', JSON.stringify(n.response ?? '')));

}

We simply subscribe to the store and save the data in the session storage every time it changes.

## ✅ set, ✅ subscribe, 🔧 load

Now we have another problem! The data is set, and our cache is always up to date... but the data never changes!

That's what "load" is for ;)

For example, in our component we can do something like this:

<script>

import { documentsList } from './store';

let queries, limit, offset

$: documentsList.load(queries, limit, offset);

</script>

{#if $documentsList?.response?.total}

 {#each $documenstList.response.document as document}

 <pre>{document}</pre>

 {/each}

{/if}

This way, every time one of the arguments passed to "load" changes the data will be updated!

And now we have an Appwrite store with cached data!

## Let's put it all together 🙌

In the end our store looks like this:

import { sdk, server } from './appwrite';

import { writable } from 'svelte/store';

import { browser } from '$app/env';

function createDocumentsListStore(){

 const { subscribe, set } = writable({

 response: browser ? JSON.parse(sessionStorage.getItem('documents')) : null

 });

 return {

 subscribe,

 set,

 load: async ( queries ,limit, offset) => {

 try {

 const response = sdk.database.listDocuments(

 server.collection,

 queries,

 limit,

 offset,

 undefined,

 undefined,

 ['created\_at'],

 ['DESC']

 );

 set({

 response

 });

 } catch (error) {

 //Handle the error

 }

 }

 };

}

export const documentsList = createDocumentsListStore();

if (browser) {

 documentsList.subscribe((n) => sessionStorage?.setItem('documents', JSON.stringify(n.response ?? '')));

}