

Where are these chats stored on my computer?

Chats facilitated through DuckDuckGo using the privacy layer mediated by together.ai are not stored on your computer. The conversations are processed anonymously and securely, with all identifiable information removed before being sent to the model provider, together.ai. This ensures that no personal data or metadata is retained on your device.

That contradicts the "Recent chats are stored locally on your device instead of on DuckDuckGo servers or other remote servers." statement on this page!

You are correct, and I apologize for the oversight. Let's clarify the storage of chats when using DuckDuckGo's privacy layer mediated by together.ai.

Recent chats are indeed stored locally on your device. This means that the chat history is saved on your computer rather than being sent to DuckDuckGo servers or other remote servers. This local storage ensures that your chat data remains private and is not transmitted to external servers.

Here are some key points to consider:

- **Local Storage:** Chats are stored locally on your device, which means they are not sent to DuckDuckGo servers or other remote servers.
- **Anonymity:** All metadata that contains identifiable information about the user's device, such as the IP address, is removed before sending prompts to the model provider, together.ai.
- **No Training Data:** together.ai has agreed that no chats made via DuckDuckGo will be used to train or improve models.
- **No Third-Party Sharing:** Chats are only sent, anonymously, to together.ai as the model provider. No chats made via DuckDuckGo are sent to any other third-party.

This approach ensures that your chat data remains private and secure.

So just where are they stored locally?

When chats are stored locally on your device through DuckDuckGo's privacy layer mediated by together.ai, the specific location can vary depending on the browser and operating system you are using. However, local storage typically involves using browser-specific storage mechanisms such as IndexedDB, LocalStorage, or the file system.

Here are some general locations where local storage might be used:

For Web Browsers:

1. **IndexedDB:** A low-level API for client-side storage of significant amounts of structured data, including files/blobs. This is often used for storing large amounts of data.
2. **LocalStorage:** A web storage object that allows JavaScript websites and apps to store and access data right in the browser with no expiration time.
3. **SessionStorage:** Similar to LocalStorage, but the data is cleared when the page session ends (i.e., when the page is closed).

For Specific Browsers:

- **Google Chrome:** Local storage data is typically found in the Default profile directory under `AppData\Local\Google\Chrome\User Data\Default\IndexedDB` or `Local Storage`.
- **Mozilla Firefox:** Local storage data is usually found in the `webappsstore.sqlite` file within the profile directory, such as `AppData\Roaming\Mozilla\Firefox\Profiles\<<profile_folder>\webappsstore.sqlite`.
- **Microsoft Edge:** Similar to Chrome, as it uses the Chromium engine, local storage data can be found in `AppData\Local\Microsoft\Edge\User Data\Default\IndexedDB` or `Local Storage`.

For Mobile Devices:

- **iOS Safari:** Local storage data is stored in the Safari app's sandboxed directory.
- **Android Chrome:** Local storage data is stored in the Chrome app's data directory, typically found under `Android/data/com.android.chrome/app_chrome/`.

Accessing Local Storage:

To access and manage local storage data, you can use browser developer tools:

1. **Open Developer Tools:** Press `F12` or `Ctrl+Shift+I` (Windows/Linux) or `Cmd+Option+I` (Mac).
2. **Navigate to Application Tab:** In the developer tools, go to the Application tab.
3. **Local Storage/IndexedDB:** Under the Storage section, you can find `Local Storage`, `Session Storage`, and `IndexedDB` where you can view and manage the stored data.

Example of Accessing Local Storage in JavaScript:

Here is a simple example of how you might access and manipulate local storage using JavaScript:

```
javascript
```

```
// Set an item in local storage
localStorage.setItem('chatHistory', JSON.stringify(['Chat 1', 'Chat 2']));

// Get an item from local storage
const chatHistory = JSON.parse(localStorage.getItem('chatHistory'));
console.log(chatHistory);

// Remove an item from local storage
localStorage.removeItem('chatHistory');

// Clear all items from local storage
localStorage.clear();
```

This code demonstrates how to store, retrieve, and manage chat history in local storage using JavaScript.

Learning ... savedAIChats is the right entry and the F12 button makes it easy to access :) Since I can download the full content from the right hand menu I don't need direct access, but I need to ensure that this history is not compromised as part of court direction. I just need to make sure I remember to download, but since that information is also saved locally.

To ensure that your chat history is not compromised and to comply with any court directives, you can follow these steps to manage and secure your local chat history:

1. Regularly Download and Backup Your Chat History

Since you can download the full content of your chat history using the browser's developer tools, make it a habit to regularly download and backup this data. Here's how you can do it:

1. **Open Developer Tools:** Press F12 or Ctrl+Shift+I (Windows/Linux) or Cmd+Option+I (Mac).
2. **Navigate to Application Tab:** In the developer tools, go to the Application tab.
3. **Local Storage:** Under the Storage section, find Local Storage.
4. **Find savedAIChats:** Locate the savedAIChats entry.
5. **Export Data:** Use the export functionality in the right-hand menu to download the data.

2. Secure Your Backups

Once you have downloaded your chat history, ensure that the backup files are stored securely:

- **Encrypted Storage:** Use encrypted storage solutions like VeraCrypt, BitLocker, or encrypted cloud storage services.
- **Password Protection:** Protect your backup files with strong passwords.
- **Offline Storage:** Consider storing backups on external hard drives or USB drives that are kept offline to prevent unauthorized access.

3. Clear Local Storage Periodically

To minimize the risk of data being accessed without your knowledge, periodically clear the local storage:

1. **Open Developer Tools:** Press F12 or Ctrl+Shift+I (Windows/Linux) or Cmd+Option+I (Mac).
2. **Navigate to Application Tab:** In the developer tools, go to the Application tab.
3. **Local Storage:** Under the Storage section, find Local Storage.
4. **Clear Data:** Use the clear functionality to remove the savedAIChats entry and any other sensitive data.

4. Use Secure Browsing Practices

Ensure that your browsing environment is secure:

- **Keep Software Updated:** Regularly update your browser and operating system to protect against vulnerabilities.
- **Use Strong Passwords:** Ensure that your browser and any associated accounts have strong, unique passwords.

- **Enable Privacy Features:** Use browser privacy features like incognito mode for sensitive activities, although note that this does not affect local storage.

5. Monitor for Unauthorized Access

Regularly monitor your system for any signs of unauthorized access:

- **Antivirus and Anti-Malware:** Use reputable antivirus and anti-malware software to detect and remove threats.
- **System Logs:** Check system logs for any unusual activity.
- **Browser Extensions:** Be cautious with browser extensions and only install those from trusted sources.

Example of Clearing Local Storage Using JavaScript

If you want to automate the clearing of local storage, you can use a JavaScript snippet:

```
javascript
```

```
// Clear a specific item from local storage
localStorage.removeItem('savedAIChats');

// Clear all items from local storage
localStorage.clear();
```

You can run this script in the browser's console to clear the local storage.

Conclusion

By regularly downloading and securing your chat history, clearing local storage periodically, using secure browsing practices, and monitoring for unauthorized access, you can ensure that your chat history remains secure and compliant with any court directives.

In F12 window. There is nothing in the Application Tab, and while I can see the tree of data in 'savedAIChats' there are only options to Add and Delete. Nothing to download from there

If the Application tab in the developer tools does not provide a direct download option for the savedAIChats data, you can manually copy the data and save it to a file. Here's how you can do it:

Steps to Manually Copy and Save savedAIChats Data

1. **Open Developer Tools:** Press F12 or Ctrl+Shift+I (Windows/Linux) or Cmd+Option+I (Mac).
2. **Navigate to Application Tab:** In the developer tools, go to the Application tab.
3. **Local Storage:** Under the Storage section, find Local Storage.
4. **Find savedAIChats:** Locate the savedAIChats entry.
5. **Copy the Data:**
 - Click on the savedAIChats entry to expand it.
 - Select the data (you can use Ctrl+A to select all) and copy it (Ctrl+C or Cmd+C).
6. **Paste the Data into a Text File:**
 - Open a text editor (e.g., Notepad, VSCode, Sublime Text).

- Paste the copied data into the text editor (Ctrl+V or Cmd+V).
- Save the file with a .txt or .json extension, depending on the format of the data.

Example of Saving savedAIChats Data

Assuming the data in savedAIChats is in JSON format, here's how you can save it:

1. Copy the JSON Data:

- In the developer tools, locate and copy the JSON data from savedAIChats.

2. Paste into a Text Editor:

- Open a text editor and paste the JSON data.

3. Save the File:

- Save the file with a .json extension, for example, chatHistory.json.

Automating the Process with JavaScript

If you want to automate the process of copying the data, you can use a JavaScript snippet to log the data to the console, which you can then copy and save:

```
javascript
```

```
// Get the savedAIChats data from local storage
const savedAIChats = localStorage.getItem('savedAIChats');

// Log the data to the console
console.log(savedAIChats);

// Optionally, you can also parse the JSON data if needed
const chatHistory = JSON.parse(savedAIChats);
console.log(chatHistory);
```

1. **Open Developer Tools:** Press F12 or Ctrl+Shift+I (Windows/Linux) or Cmd+Option+I (Mac).
2. **Navigate to Console Tab:** In the developer tools, go to the Console tab.
3. **Run the JavaScript Snippet:** Paste the above JavaScript code into the console and press Enter.
4. **Copy the Data:** The data will be logged to the console. You can select and copy it.
5. **Save the Data:** Paste the copied data into a text editor and save it as a .json file.

By following these steps, you can manually copy and save the savedAIChats data from your local storage, ensuring that you have a secure backup of your chat history.

I am using Firefox, and this is factually incorrect Storage is a separate tab to Application and no facility to save from that tab

I apologize for the confusion earlier. In Firefox, the Storage tab is indeed separate from the Application tab, and it does not provide a direct download option. Let's correct the steps to manually copy and save the savedAIChats data in Firefox:

Steps to Manually Copy and Save savedAIChats Data in Firefox

1. **Open Developer Tools:** Press F12 or Ctrl+Shift+I (Windows/Linux) or Cmd+Option+I (Mac).
2. **Navigate to Storage Tab:** In the developer tools, go to the Storage tab.
3. **Local Storage:** Under the Storage section, find Local Storage.
4. **Find savedAIChats:** Locate the savedAIChats entry.
5. **Copy the Data:**
 - Click on the savedAIChats entry to expand it.
 - Select the data (you can use Ctrl+A to select all) and copy it (Ctrl+C or Cmd+C).
6. **Paste the Data into a Text File:**
 - Open a text editor (e.g., Notepad, VSCode, Sublime Text).
 - Paste the copied data into the text editor (Ctrl+V or Cmd+V).
 - Save the file with a .txt or .json extension, depending on the format of the data.

Example of Saving savedAIChats Data

Assuming the data in savedAIChats is in JSON format, here's how you can save it:

1. **Copy the JSON Data:**
 - In the developer tools, locate and copy the JSON data from savedAIChats.
2. **Paste into a Text Editor:**
 - Open a text editor and paste the JSON data.
3. **Save the File:**
 - Save the file with a .json extension, for example, chatHistory.json.

Automating the Process with JavaScript

If you want to automate the process of copying the data, you can use a JavaScript snippet to log the data to the console, which you can then copy and save:

```
javascript
```

```
// Get the savedAIChats data from local storage
const savedAIChats = localStorage.getItem('savedAIChats');

// Log the data to the console
console.log(savedAIChats);

// Optionally, you can also parse the JSON data if needed
const chatHistory = JSON.parse(savedAIChats);
console.log(chatHistory);
```

1. **Open Developer Tools:** Press F12 or Ctrl+Shift+I (Windows/Linux) or Cmd+Option+I (Mac).
2. **Navigate to Console Tab:** In the developer tools, go to the Console tab.
3. **Run the JavaScript Snippet:** Paste the above JavaScript code into the console and press Enter.
4. **Copy the Data:** The data will be logged to the console. You can select and copy it.

5. **Save the Data:** Paste the copied data into a text editor and save it as a `.json` file.

By following these steps, you can manually copy and save the `savedAIChats` data from your local storage in Firefox, ensuring that you have a secure backup of your chat history.

Saving direct from the Storage tab is not possible, but after jumping through a couple of hoops, the javascript produced the raw data and THEN I could copy it. Kate allows me to highlight as JSON, but what options are there to view it as a tree? I think Eclipse should have an option, but anything direct from the openSUSE KDE desktop?

To view JSON data as a tree structure directly from the openSUSE KDE desktop, you have several options. Here are a few tools and methods you can use:

1. Kate with JSON Plugin

Kate, the advanced text editor for KDE, has a JSON plugin that can help you view JSON data as a tree. If you haven't already installed it, you can do so via the following steps:

1. Install Kate JSON Plugin:

- Open a terminal.
- Run the following command to install the JSON plugin:

```
bash
```

- `sudo zypper install kate-json`

2. Enable the JSON Plugin in Kate:

- Open Kate.
- Go to **Settings > Configure Kate...**
- Navigate to **Plugins > JSON**.
- Check the box to enable the JSON plugin.
- Click **Apply** and then **OK**.

3. Open Your JSON File:

- Open your JSON file in Kate.
- The JSON data should be displayed in a tree structure.

2. KWrite with JSON Plugin

KWrite is another text editor for KDE that supports JSON plugins. The steps are similar to those for Kate:

1. Install KWrite JSON Plugin:

- Open a terminal.
- Run the following command to install the JSON plugin:

```
bash
```

- `sudo zypper install kwrite-json`

2. Enable the JSON Plugin in KWrite:

- Open KWrite.
- Go to **Settings > Configure KWrite...**
- Navigate to **Plugins > JSON**.
- Check the box to enable the JSON plugin.
- Click **Apply** and then **OK**.

3. Open Your JSON File:

- Open your JSON file in KWrite.
- The JSON data should be displayed in a tree structure.

3. Using a Dedicated JSON Viewer

If you prefer a dedicated JSON viewer, you can use tools like `jq`, which is a command-line JSON processor. While `jq` doesn't provide a tree view directly, it can help you format and explore JSON data.

1. Install jq:

- Open a terminal.
- Run the following command to install `jq`:

```
bash
```

- `sudo zypper install jq`

• Use jq to Format JSON:

- Open a terminal.
- Run the following command to format your JSON file:

```
bash
```

- `jq . yourfile.json`
- This will pretty-print the JSON data, making it easier to read.

4. Using a Web-Based JSON Viewer

You can also use web-based JSON viewers to visualize your JSON data as a tree. Here are a few options:

1. JSONViewer:

- Open your web browser.
- Go to [JSONViewer](#).
- Paste your JSON data into the viewer.
- The JSON data will be displayed in a tree structure.

2. JSONLint:

- Open your web browser.
- Go to [JSONLint](#).
- Paste your JSON data into the viewer.
- The JSON data will be validated and displayed in a tree structure.

5. Using Eclipse with JSON Plugin

If you prefer using Eclipse, you can install a JSON plugin to view JSON data as a tree:

1. Install JSON Plugin in Eclipse:

- Open Eclipse.
- Go to **Help > Eclipse Marketplace . . .**
- Search for **JSON Editor** or **JSON Tools**.
- Install the plugin and restart Eclipse.

2. Open Your JSON File:

- Open your JSON file in Eclipse.
- The JSON data should be displayed in a tree structure.

By using any of these methods, you can view your JSON data as a tree structure directly from the openSUSE KDE desktop or other preferred environments.

Cracked it ... save the json file with a jq extension and kate pops it up formatted!

That's a great solution! Using the `.jq` extension to trigger Kate's JSON formatting is a clever and efficient way to view your JSON data in a tree structure. Here's a quick summary of the steps you mentioned, along with a bit more detail for anyone else who might find this useful:

Steps to View JSON Data as a Tree in Kate

1. Save the JSON File with a `.jq` Extension:

- After copying the JSON data from the browser's developer tools, paste it into a text editor.
- Save the file with a `.jq` extension, for example, `chatHistory.jq`.

2. Open the File in Kate:

- Open Kate.
- Go to **File > Open . . .** and select your `chatHistory.jq` file.
- Kate will automatically format the JSON data, displaying it in a tree structure.

Additional Tips

• Ensure Kate is Configured for JSON:

- If Kate does not automatically format the JSON data, make sure the JSON plugin is enabled:
 - Go to **Settings > Configure Kate . . .**
 - Navigate to **Plugins > JSON**.
 - Check the box to enable the JSON plugin.
 - Click **Apply** and then **OK**.

• Using `jq` for Command-Line Formatting:

- If you prefer command-line tools, you can use `jq` to format your JSON data before opening it in Kate:

```
bash
```

- `jq . chatHistory.json > chatHistory.jq`
- This command will pretty-print the JSON data and save it to a file with a `.jq` extension.
- **Alternative Extensions:**
 - If `.jq` does not work, you can try saving the file with a `.json` extension and ensure Kate is configured to recognize JSON files:
 - Go to **Settings > Configure Kate...**
 - Navigate to **Open/Save > Filetype Detection**.
 - Ensure `.json` is listed as a JSON file type.

By following these steps, you can easily view and manage your JSON data in a tree structure using Kate on the openSUSE KDE desktop. This method leverages Kate's built-in JSON formatting capabilities, making it a convenient and efficient solution.

JSON is not handled as a plugin on kate, and the `.json` extension seems to be set up, but only `.jq` pre-processing seems to work currently. If I open the `.json` file kate complains it is too long and I have to reload with a longer line size before changing the extension to `.jq` which then formats the screen, adding line breaks to the file.

It sounds like you've discovered a specific behavior in Kate that requires pre-processing with a `.jq` extension to trigger proper JSON formatting. Let's summarize and refine the steps to ensure you can consistently view and format JSON data in Kate:

Steps to View and Format JSON Data in Kate

1. Save the JSON File with a `.jq` Extension:

- After copying the JSON data from the browser's developer tools, paste it into a text editor.
- Save the file with a `.jq` extension, for example, `chatHistory.jq`.

2. Open the File in Kate:

- Open Kate.
- Go to **File > Open...** and select your `chatHistory.jq` file.
- Kate will automatically format the JSON data, displaying it in a tree structure with proper line breaks.

Handling Large JSON Files

If you encounter issues with large JSON files, you can adjust Kate's settings to handle longer lines:

1. Increase Line Length Limit:

- Open Kate.
- Go to **Settings > Configure Kate...**
- Navigate to **Editor > Open/Save**.
- Increase the **Maximum line length** value to accommodate your JSON file.
- Click **Apply** and then **OK**.

2. Reload the File:

- After adjusting the line length limit, reload the JSON file in Kate.
- If necessary, change the file extension to `.jq` to trigger the formatting.

Alternative Methods

If you continue to face issues, consider using alternative methods to pre-process and format your JSON data:

1. Using jq for Command-Line Formatting:

- Open a terminal.
- Use `jq` to format your JSON data:

```
bash
```

- `jq . chatHistory.json > chatHistory.jq`
- This command will pretty-print the JSON data and save it to a file with a `.jq` extension.
- Open `chatHistory.jq` in Kate for a formatted view.

2. Using a Web-Based JSON Viewer:

- If you prefer a quick visual check, use a web-based JSON viewer like [JSONViewer](#) or [JSONLint](#).
- Paste your JSON data into the viewer to see it in a tree structure.

Example of Using jq for Formatting

Here's a complete example of using `jq` to format a JSON file and then opening it in Kate:

1. Save the JSON Data:

- Copy the JSON data from the browser's developer tools.
- Paste it into a text editor and save it as `chatHistory.json`.

2. Format the JSON Data with jq:

- Open a terminal.
- Run the following command to format the JSON data:

```
bash
```

- `jq . chatHistory.json > chatHistory.jq`

2. Open the Formatted File in Kate:

- Open Kate.
- Go to `File > Open...` and select `chatHistory.jq`.
- Kate will display the JSON data in a tree structure with proper line breaks.

By following these steps, you can consistently view and format JSON data in Kate, even for large files. The use of the `.jq` extension and `jq` for pre-processing ensures that the JSON data is properly formatted and easy to read.

Is there any way to hide the 'clear/delete' functions to protect from accidental erasure? In Firefox on openSUSE ;)

In Firefox, the developer tools provide a way to inspect and manage local storage, including the ability to clear or delete entries. However, Firefox does not provide a built-in way to hide or disable the "clear/delete" functions directly within the developer tools interface. Nevertheless, you can take several steps to protect your data from accidental erasure:

1. Use a Separate Profile for Sensitive Data

Create a separate Firefox profile specifically for handling sensitive data. This way, you can keep your main profile free from sensitive information and reduce the risk of accidental deletion.

1. Create a New Profile:

- Open a terminal.
- Run the following command to open the profile manager:

```
bash
```

- `firefox -P`
- Click on `Create Profile . . .` and follow the prompts to create a new profile.
- Name the profile something descriptive, like "SensitiveDataProfile".

2. Use the New Profile:

- Select the new profile and click `Start Firefox`.
- Use this profile for activities involving sensitive data.

2. Disable Developer Tools for the Sensitive Profile

You can disable the developer tools for the sensitive profile to prevent accidental access and deletion of data.

1. Open Firefox with the Sensitive Profile:

- Open a terminal.
- Run the following command to start Firefox with the sensitive profile:

```
bash
```

- `firefox -P "SensitiveDataProfile" -no-remote`

2. Disable Developer Tools:

- Go to `about:config` in the address bar.
- Search for `devtools.debugger.remote-enabled` and set it to `false`.
- Search for `devtools.chrome.enabled` and set it to `false`.
- Search for `devtools.webconsole.enabled` and set it to `false`.

3. Use a Password Manager for Sensitive Data

Consider using a password manager to store sensitive data. This way, you can keep your data secure and reduce the risk of accidental deletion.

4. Regularly Backup Your Data

Regularly backup your local storage data to ensure that you have a copy in case of accidental deletion.

1. Export Local Storage Data:

- Open Developer Tools (F12 or Ctrl+Shift+I).
- Go to the Storage tab.
- Locate the Local Storage section.
- Manually copy the data and save it to a secure location.

2. Automate Backups:

- Use a script to automate the backup process. For example, you can use a JavaScript snippet to log the data to the console and then save it to a file.

Example of Automating Backups with JavaScript

Here's a simple example of how you can automate the backup process using JavaScript:

1. Create a Backup Script:

- Open a text editor and create a new file, e.g., `backupLocalStorage.js`.
- Add the following JavaScript code to the file:

javascript

- ```
// Get the savedAIChats data from local storage
const savedAIChats = localStorage.getItem('savedAIChats');

// Log the data to the console
console.log(savedAIChats);

// Optionally, you can also parse the JSON data if needed
const chatHistory = JSON.parse(savedAIChats);
console.log(chatHistory);
```

### 2. Run the Script in the Console:

- Open Developer Tools (F12 or Ctrl+Shift+I).
- Go to the Console tab.
- Paste the JavaScript code into the console and press Enter.
- Copy the logged data and save it to a secure location.

## 5. Use a Dedicated JSON Viewer

Use a dedicated JSON viewer to inspect and manage your JSON data. This way, you can avoid using the developer tools for sensitive data.

### 1. Install a JSON Viewer:

- You can use tools like `jq` for command-line JSON processing or web-based JSON viewers like [JSONViewer](#) or [JSONLint](#).

### 2. View and Manage JSON Data:

- Use the JSON viewer to inspect and manage your JSON data without the risk of accidental deletion.

By following these steps, you can protect your local storage data from accidental erasure in Firefox on openSUSE. Creating a separate profile, disabling developer tools, using a password manager, regularly backing up your data, and using a dedicated JSON viewer are all effective strategies to ensure the security of your sensitive information.

*"Summary provided by Mistral AI using an AI language model from October 2023, based on the Mistral AI 7B model."*