

Weblydarkiv and Voicegram: Implementing Web-Based Audio Archiving with W3C WebAudio and WebStorage

Ole Aamot

2025

Abstract

This research paper explores the implementation of Weblydarkiv and Voicegram, two web-based audio archiving systems developed using W3C WebAudio and WebStorage APIs. The primary focus is on how these technologies can be used to create scalable, client-side audio storage and metadata management systems suitable for archival and retrieval purposes. The paper also discusses the potential for future iterations of these systems, including the integration of more advanced features and their compatibility with standards like Noark5 for long-term digital archiving.

1 Introduction

The advent of web technologies such as W3C WebAudio and WebStorage APIs has opened up new possibilities for building sophisticated web applications. In this paper, we investigate the potential of these technologies in the context of audio archiving. Specifically, we examine two systems, Weblydarkiv and Voicegram, which leverage these APIs for audio data storage and management. These systems aim to provide a web-based solution for archiving audio in formats such as FLAC, ensuring both high fidelity and accessibility.

2 Background

Weblydarkiv and Voicegram are part of ongoing research and development into more efficient and accessible audio archiving methods. Inspired by discussions on audio file storage formats, particularly FLAC, these systems were designed to facilitate the preservation and retrieval of audio content, incorporating the W3C WebAudio API for playback and the WebStorage API for metadata storage. This section outlines the technical foundations of these APIs and their relevance to the project.

2.1 W3C WebAudio API

The WebAudio API is a powerful tool for manipulating audio in the browser. It allows for sophisticated audio processing and control, making it an ideal choice for an archiving solution where high-quality playback is essential.

2.2 W3C WebStorage API

WebStorage provides a mechanism for storing data locally on a user's device. This feature is crucial for Weblydarkiv and Voicegram, as it allows for the offline storage of metadata and audio files, making the systems more resilient to connectivity issues.

3 Implementation

The implementation of Weblydarkiv and Voicegram is primarily based on HTML5, utilizing the WebAudio and WebStorage APIs. Audio files are encoded in FLAC format, a lossless compression format that preserves audio quality while maintaining manageable file sizes. The systems use IndexedDB for efficient local storage, and localStorage ensures that metadata is preserved between sessions.

3.1 Weblydarkiv System

Weblydarkiv is a web-based audio archiving tool that allows users to upload, edit, and store metadata associated with audio files. The system supports the FLAC format and is compatible with the Noark5 standard for digital

archiving. The metadata editor allows users to input crucial information such as title, creator, creation date, and provenance. All metadata is stored in IndexedDB, ensuring persistence across sessions.

3.2 Voicegram System

Voicegram is another web-based system that focuses on managing multiple-location audio recordings (MLAR). The system is capable of handling audio files from different locations and supports the encoding of audio data in base64 format for storage within web applications.

4 Noark5 Audio Recording HTML5 Code

Below is the HTML5 source code from the Noark5 audio recording implementation on the Aamot Research website:

Listing 1: HTML5 Source Code from Noark5 Audio Recording

```
<!DOCTYPE html>
<html lang="no">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
  <title>Flere Filer og Lydopptak</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    #fileList {
      margin-top: 20px;
    }
    button {
      padding: 10px;
      margin: 10px;
      cursor: pointer;
    }
    #recorderControls {
      margin-top: 20px;
    }
    #audioPlayer {
      margin-top: 20px;
```

```

    }
  </style>
</head>
<body>
  <h1>Flere Filer og Lydopptak</h1>

  <button id="loadFilesButton">Last inn lydfiler fra
    WebStorage</button>
  <button id="clearStorageButton">Tøm WebStorage</button>

  <h2>Lydfiler:</h2>
  <div id="fileList"></div>

  <h2>Lydopptaker:</h2>
  <div id="recorderControls">
    <button id="startRecordingButton">Start opptak</button>
    <button id="stopRecordingButton" disabled>Stopp opptak</
      button>
    <audio id="audioPlayer" controls></audio>
  </div>

  <input type="file" id="fileInput" accept=".flac" multiple /
    >

  <script>
    // JavaScript-kode for håndtering av flere .flac filer
    og opptak

    // Funksjon for å hente og vise lagrede filer fra
    WebStorage
    function loadFilesFromStorage() {
      const files = JSON.parse(localStorage.getItem('
        flacFiles')) || [];
      const fileList = document.getElementById('fileList');
      fileList.innerHTML = '';

      if (files.length > 0) {
        files.forEach((fileData, index) => {
          const fileElement = document.createElement('div');
          fileElement.innerHTML = `<strong>Fil ${index + 1}:<
            /strong>
            <audio controls>
              <source src="${fileData.audioUrl}" type="audio/
                flac">
            Din nettleser støtter ikke FLAC-formatet.

```

```

        </audio>`;
        fileList.appendChild(fileElement);
    });
} else {
    fileList.innerHTML = 'Ingen lydfiler lagret.';
}
}

// Funksjon for laste opp og lagre flac-filer i
// WebStorage
function handleFileUpload(event) {
    const files = event.target.files;
    const fileArray = JSON.parse(localStorage.getItem('
        flacFiles')) || [];

    Array.from(files).forEach(file => {
        if (file.type === 'audio/flac') {
            const reader = new FileReader();

            reader.onloadend = function() {
                // Legg til filens base64-streng i WebStorage
                fileArray.push({
                    fileName: file.name,
                    audioUrl: reader.result
                });

                // Lagre de nye filene i WebStorage
                localStorage.setItem('flacFiles', JSON.stringify(
                    fileArray));

                // Last inn og vis de lagrede filene p nytt
                loadFilesFromStorage();
            };

            // Les filen som data-URL (base64)
            reader.readAsDataURL(file);
        } else {
            alert('Kun .flac-filer er tillatt.');
        }
    });
}

// Funksjon for tømme WebStorage og filene
function clearStorage() {
    localStorage.removeItem('flacFiles');
}

```

```

    loadFilesFromStorage();
}

// Lydopptak
let mediaRecorder;
let audioChunks = [];

// Start opptak
function startRecording() {
    navigator.mediaDevices.getUserMedia({ audio: true })
        .then(stream => {
            mediaRecorder = new MediaRecorder(stream);
            mediaRecorder.ondataavailable = event => {
                audioChunks.push(event.data);
            };
            mediaRecorder.onstop = () => {
                const audioBlob = new Blob(audioChunks, { type: 'audio/wav' });
                const audioUrl = URL.createObjectURL(audioBlob);
                document.getElementById('audioPlayer').src = audioUrl;

                // Lagre opptaket i WebStorage som en data-URI
                const fileArray = JSON.parse(localStorage.getItem('flacFiles')) || [];
                fileArray.push({
                    fileName: 'recorded_audio.wav',
                    audioUrl: audioUrl
                });

                // Lagre de nye filene i WebStorage
                localStorage.setItem('flacFiles', JSON.stringify(fileArray));

                // Last inn og vis de lagrede filene p  nytt
                loadFilesFromStorage();
            };
            mediaRecorder.start();

            // Deaktiver start-knappen og aktiver stopp-knappen
            document.getElementById('startRecordingButton').disabled = true;
            document.getElementById('stopRecordingButton').disabled = false;
        })
}

```

```

        .catch(error => {
            console.error('Feil ved tilgang til mikrofonen:',
                error);
            alert('Kunne ikke starte opptaket.');
```

```
        });
    }

```

```

// Stopp opptak
function stopRecording() {
    mediaRecorder.stop();

```

```

    // Deaktiver stopp-knappen og aktiver start-knappen
    document.getElementById('startRecordingButton').
        disabled = false;
    document.getElementById('stopRecordingButton').disabled
        = true;
}

```

```

// Koble til knapper og filinput
document.getElementById('loadFilesButton').
    addEventListener('click', loadFilesFromStorage);
document.getElementById('clearStorageButton').
    addEventListener('click', clearStorage);
document.getElementById('fileInput').addEventListener('
    change', handleFileUpload);
document.getElementById('startRecordingButton').
    addEventListener('click', startRecording);
document.getElementById('stopRecordingButton').
    addEventListener('click', stopRecording);

```

```

    // Last inn lagrede filer ved oppstart
    loadFilesFromStorage();
</script>
</body>
</html>

```

5 Noark5 Audio Archiving HTML5 Code

Below is the HTML5 source code from the Noark5 audio archiving implementation on the Aamot Research website:

Listing 2: HTML5 Source Code from Noark5 Audio Archiving

```
<!DOCTYPE html>
```

```

<html lang="no">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
  <title>Noark5 Metadata Editor</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"><
    /script>
  <style>
    body { font-family: Arial, sans-serif; margin: 20px; }
    input, textarea { width: 100%; padding: 8px; margin: 5px
      0; }
    button { padding: 10px; margin: 10px; cursor: pointer; }
  </style>
</head>
<body>
  <h1>Noark5 Metadata Editor</h1>

  <h3>Last opp lydfil:</h3>
  <input type="file" id="fileInput" accept=".flac" />

  <h3>Rediger Metadata:</h3>
  <label for="metadataTitle">Tittel:</label>
  <input type="text" id="metadataTitle" placeholder="Tittel_
    p _lydfil" />

  <label for="metadataCreator">Skaper:</label>
  <input type="text" id="metadataCreator" placeholder="Skaper
    _av_lydfilen" />

  <label for="metadataDate">Dato for opprettelse:</label>
  <input type="datetime-local" id="metadataDate" />

  <label for="metadataSubject">Emne:</label>
  <input type="text" id="metadataSubject" placeholder="Emne_
    for_lydfilen" />

  <label for="metadataKeywords">Stikkord:</label>
  <input type="text" id="metadataKeywords" placeholder="
    Stikkord, _adskilt _med _komma" />

  <label for="metadataProvenance">Proveniens:</label>
  <textarea id="metadataProvenance" placeholder="Opprinnelse_
    til _filen"></textarea>

```



```

<button id="saveMetadataButton" disabled>Lagre Metadata</button>
<button id="clearMetadataButton">Tøm Metadata</button>

<h3>Lagrede filer:</h3>
<div id="fileList"></div>

<script>
  const dbName = "audioArchiveDB";
  let db;

  // pne IndexedDB
  function openDB() {
    let request = indexedDB.open(dbName, 1);
    request.onsuccess = (e) => { db = e.target.result;
      loadFilesFromDB(); };
    request.onerror = (e) => { console.error("Database□
      error:", e.target.error); };
    request.onupgradeneeded = (e) => {
      db = e.target.result;
      let store = db.createObjectStore("files", { keyPath:
        "id", autoIncrement: true });
      store.createIndex("fileName", "fileName", { unique:
        false });
    };
  }

  // Lagrer metadata i IndexedDB
  function storeFileInDB(file, metadata) {
    let transaction = db.transaction(["files"], "readwrite"
    );
    let store = transaction.objectStore("files");
    store.add({ fileName: file.name, fileData: file,
      metadata: metadata });
    loadFilesFromDB();
  }

  // Henter filer fra IndexedDB
  function loadFilesFromDB() {
    $("#fileList").empty();
    let transaction = db.transaction(["files"], "readonly"
    );
    let store = transaction.objectStore("files");
    let request = store.getAll();
  }
</script>

```

```

request.onsuccess = (e) => {
  let files = e.target.result;
  if (files.length > 0) {
    files.forEach((file, index) => {
      let fileElement = $(`
        <div>
          <strong>Fil ${index + 1}</strong> ${file.
            metadata.title} <br>
          <strong>Skaper:</strong> ${file.metadata.
            creator} <br>
          <strong>Dato:</strong> ${file.metadata.
            creationDate} <br>
          <audio controls>
            <source src="${URL.createObjectURL(file.
              fileData)}" type="audio/flac">
            Nettleseren din st tter ikke FLAC.
          </audio>
          <button onclick="editMetadata(${file.id})">
            Rediger</button>
        </div>
        <hr>
      `);
      $("#fileList").append(fileElement);
    });
  } else {
    $("#fileList").html("Ingen lydfiler lagret.");
  }
};
}

// Fyll ut metadata-redigereren
function editMetadata(fileId) {
  let transaction = db.transaction(["files"], "readonly");
  ;
  let store = transaction.objectStore("files");
  let request = store.get(fileId);

  request.onsuccess = (e) => {
    let file = e.target.result;
    $("#metadataTitle").val(file.metadata.title);
    $("#metadataCreator").val(file.metadata.creator);
    $("#metadataDate").val(file.metadata.creationDate);
    $("#metadataSubject").val(file.metadata.subject);
    $("#metadataKeywords").val(file.metadata.keywords.
      join(", "));
  }
}

```

```

        $("#metadataProvenance").val(file.metadata.provenance
        );

        $("#saveMetadataButton").prop("disabled", false).off(
            "click").on("click", () => saveMetadata(fileId));
    };
}

// Lagre metadata i IndexedDB
function saveMetadata(fileId) {
    let transaction = db.transaction(["files"], "readwrite"
    );
    let store = transaction.objectStore("files");
    let request = store.get(fileId);

    request.onsuccess = (e) => {
        let file = e.target.result;
        file.metadata = {
            title: $("#metadataTitle").val(),
            creator: $("#metadataCreator").val(),
            creationDate: $("#metadataDate").val(),
            subject: $("#metadataSubject").val(),
            keywords: $("#metadataKeywords").val().split(",").
                map(keyword => keyword.trim()),
            provenance: $("#metadataProvenance").val()
        };

        store.put(file);
        alert("Metadata lagret!");
        loadFilesFromDB();
    };
}

// H ndter filopplastning
$("#fileInput").on("change", function(event) {
    let file = event.target.files[0];
    if (file && file.type === "audio/flac") {
        let metadata = {
            title: file.name,
            creator: "Ukjent",
            creationDate: new Date().toISOString(),
            subject: "Lydarkiv",
            keywords: ["audio", "archive"],
            provenance: "Opplattet av bruker"
        };
    };
}

```

```

        storeFileInDB(file, metadata);
    } else {
        alert("Kun flac-filer er tillatt.");
    }
});

// Lagrer metadata til LocalStorage
$("input, textarea").on("input", function() {
    localStorage.setItem($(this).attr("id"), $(this).val())
    ;
});

// Gjenoppretter metadata fra LocalStorage
$(document).ready(() => {
    $("input, textarea").each(function() {
        let savedValue = localStorage.getItem($(this).attr("id"));
        if (savedValue) $(this).val(savedValue);
    });

    openDB();
});

// Nullstill metadata
$("#clearMetadataButton").on("click", function() {
    $("input, textarea").val("");
    localStorage.clear();
});
</script>
</body>
</html>

```

6 Future Work

There are several potential improvements and extensions for Weblydarkiv and Voicegram. One possibility is the integration of these systems with existing archival standards, such as Noark5, to enhance their usability in official archival and documentation settings. Furthermore, additional file formats and features, such as the ability to process and store metadata for multiple file types, could be incorporated into future versions.

7 Conclusion

The implementation of Weblydarkiv and Voicegram demonstrates the potential of web-based technologies for creating robust, scalable audio archiving systems. By utilizing the W3C WebAudio and WebStorage APIs, these systems are capable of offering high-quality audio storage and metadata management without the need for external servers. This paper provides a foundation for further research into the application of web technologies in the field of digital archiving.

8 References

- Aamot, O. (2025). Weblydarkiv and Voicegram: Implementing Web-Based Audio Archiving with W3C WebAudio and WebStorage. Aamot Research. <https://www.oleaamot.com/research/Aamot,2026.pdf>