

# **THE TUNE**

# **MP3-Player**

## **- Installationsanleitung -**

### **Wichtiger Hinweis:**

Den Code aus diesem Dokument immer erst in eine leere Textdatei einfügen und hinterher in einen Editor einfügen, damit keine verfremdeten „HTML/<span>- und <p>-Codes“ eingefügt werden - da dies den Playercode und somit den Player zerstören würde.

## Einbindung der Player-Bibliothek im Head

Füge im <head> deiner HTML-Datei den Wavesurfer-Script-Link ein. Das ist die zentrale Bibliothek für die Audiovisualisierung.

In meinem Fall liegt die `wavesurfer_v.7.8.1.7.js` direkt im Root des Servers. Jeder kann sie von dort direkt frei runterladen, z. B. mit „JDownloader“.

Wichtig: Neuere Versionen von „Wavesurfer“ sind **nicht** mit dem hier angezeigten Code kompatibel; versucht es erst gar nicht, eine neuere Version mit diesem Code kompatibel zu machen; spart euch die Arbeit lieber.

## Code:

```
<script src="/wavesurfer_v.7.8.1.7.js"></script>
```

## Einfügen der Styles für den Player

Kopiere untenstehenden Code direkt nach dem (obigen stehend) Script (<script src="/wavesurfer\_V.7.8.1.7.js"></script>) in den Head.

Dieser Code definiert das Layout, die Farben, die Waveform-Höhen, Button-Abstände, Volumen- und Powermeter sowie die Visualisierungsbalken.

Hier könnt ihr später natürlich eure eigenen Farben und Einstellungen nach Belieben ändern. Hier gibt es Feinjustierung für Jedermann! Ich persönlich habe für meinen Player aber eigentlich schon alles super eingestellt.

## Code:

```
<style>
:root {
  --wave-width: 320px;
  --wave1-height: 50px;
  --wave2-height: 20px;
  --wave-gap: 2px;
  --wave1-color: #FF3000;
  --wave1-progress: #444444;
  --wave2-color: #FFFFFF;
  --wave2-progress: #555555;
  --wave2-type: 1;
  --placeholder-line-color: #CCCCCC;
  --button-gap-top: 15px;
  --button-gap-between: 15px;
  --meter1-update-interval: 30;
  --meter1-sensitivity: 3.5;
  --meter2-update-interval: 700;
  --meter2-sensitivity: 0.75;
  --meter3-update-interval: 50;
  --meter3-smooth-factor: 0.6;
  --meter3-sensitivity: 1.5;
  --meter3-start-angle: -50;
  --meter3-max-swing: 55;
  --waveform-bg: transparent;
}
.player-container { margin-bottom: 20px; }
.duration {
  font-family: Calibri;
  font-size: 12pt;
  color: #FFFFFF;
}
```

```

        position: absolute;
        right: 10px;
        top: 5px;
        background: rgba(51, 51, 51, 0.5);
        padding: 2px 6px;
        border-radius: 3px;
        border: 1px solid #eaeaea;
        z-index: 4;
    }
    .current-time {
        font-family: Calibri;
        font-size: 12pt;
        color: #FFFFFF;
        position: absolute;
        top: -20px;
        background: rgba(51, 51, 51, 0.5);
        padding: 2px 6px;
        border-radius: 3px;
        z-index: 4;
    }
    .audioplayer {
        position: relative;
        width: var(--wave-width);
        height: calc(var(--wavel-height) + var(--wave-gap) + var(--wave2-
height) + var(--button-gap-top) + 28px);
        background: var(--waveform-bg);
    }
    .waveform-container {
        position: absolute;
        width: var(--wave-width);
        height: var(--wavel-height);
        top: 0;
        left: 0;
        z-index: 2;
    }
    .waveform-main { width: 100%; height: 100%; }
    .waveform-shadow-container {
        position: absolute;
        width: var(--wave-width);
        height: var(--wave2-height);
        top: calc(var(--wavel-height) + var(--wave-gap));
        left: 0;
        z-index: 1;
    }
    .waveform-shadow { width: 100%; height: 100%; }
    .waveform-overlay {
        width: var(--wave-width);
        height: calc(var(--wavel-height) + var(--wave-gap) + var(--wave2-
height));
        position: absolute;
        top: 0;
        left: 0;
        z-index: 3;
        pointer-events: auto;
    }
    .button-container {
        position: absolute;
        width: var(--wave-width);
        height: 28px;
        top: calc(var(--wavel-height) + var(--wave-gap) + var(--wave2-height) +
var(--button-gap-top));
        left: 0;
        z-index: 4;
    }
    .placeholder-line {
        position: absolute;
        top: 50%;
        transform: translateY(-50%);
        width: var(--wave-width);
        height: 1px;
    }

```

```

        background-color: var(--placeholder-line-color);
        z-index: 5;
        transition: opacity 0.3s ease;
        pointer-events: none;
    }
    .volume-meter-1 {
        width: 30px;
        height: 100px;
        background:
url('https://www.thetuneproductions.com/wavesurfer/VolumeLevel3.png') no-repeat;
        background-size: 30px 100px;
        position: relative;
        overflow: hidden;
        margin: 0 auto;
    }
    .volume-level-1 {
        width: 30px;
        height: 100px;
        background: #242424;
        position: absolute;
        top: 0;
        transition: height 0s ease;
    }
    .volume-meter-2 {
        width: 3px;
        height: 100px;
        background:
url('https://www.thetuneproductions.com/wavesurfer/VolumeLevel2.png') no-repeat;
        background-size: 3px 100px;
        position: relative;
        overflow: hidden;
        margin: 0 auto;
    }
    .volume-level-2-bottom {
        width: 3px;
        height: 100px;
        background: rgba(36, 36, 36, 0.9);
        position: absolute;
        bottom: 0;
        transition: height 0.1s ease;
    }
    .volume-level-2-top {
        width: 3px;
        height: 100px;
        background: rgba(36, 36, 36, 0.9);
        position: absolute;
        top: 0;
        transition: height 0.1s ease;
    }
    .volume-peak-2 {
        width: 3px;
        height: 3px;
        background: #FFFFFF;
        position: absolute;
        bottom: 0;
        transition: bottom 0.1s ease;
        z-index: 1;
    }
    .powermeter-container {
        position: relative;
        width: 250px;
        height: 150px;
        overflow: hidden;
    }
    .powermeter-needle {
        position: absolute;
        bottom: 4px;
        left: 50%;
        transform-origin: bottom center;
        transition: transform 0.05s ease;
    }

```

```

    }
    .needle-glow {
        position: absolute;
        bottom: 31px;
        left: 50%;
        transform-origin: bottom center;
        width: 3px;
        pointer-events: none;
        transition: transform 0.05s ease, opacity 0.2s ease, height 0.2s ease;
        z-index: 0;
    }
    .restart-button {
        width: 28px;
        height: 28px;
        background:
url('https://www.thetuneproductions.com/wavesurfer/Arrow_left.png') no-repeat;
        border: none;
        position: absolute;
        left: 5px;
        cursor: pointer;
    }
    .rewind-button {
        width: 28px;
        height: 28px;
        background:
url('https://www.thetuneproductions.com/wavesurfer/10Secback.png') no-repeat;
        border: none;
        position: absolute;
        left: calc(5px + 28px + var(--button-gap-between));
        cursor: pointer;
    }
    .forward-button {
        width: 28px;
        height: 28px;
        background:
url('https://www.thetuneproductions.com/wavesurfer/10Secforward.png') no-repeat;
        border: none;
        position: absolute;
        left: calc(5px + 56px + 2 * var(--button-gap-between));
        cursor: pointer;
    }
    .col1-button {
        width: 28px;
        height: 28px;
        background:
url('https://www.thetuneproductions.com/wavesurfer/Col1.png') no-repeat;
        border: none;
        position: absolute;
        left: calc(5px + 84px + 3 * var(--button-gap-between));
        cursor: pointer;
    }
    .col2-button {
        width: 28px;
        height: 28px;
        background:
url('https://www.thetuneproductions.com/wavesurfer/Col2.png') no-repeat;
        border: none;
        position: absolute;
        left: calc(5px + 112px + 4 * var(--button-gap-between));
        cursor: pointer;
    }
    .backcol-button {
        width: 28px;
        height: 28px;
        background:
url('https://www.thetuneproductions.com/wavesurfer/BackCol.png') no-repeat;
        border: none;
        position: absolute;
        left: calc(5px + 140px + 5 * var(--button-gap-between));
        cursor: pointer;
    }

```

```

    }
    .waveform-main, .waveform-shadow {
        background: #222222;
        height: var(--wave1-height);
    }
    .waveform-shadow {
        background: #333333;
        height: var(--wave2-height);
    }
    .preload-images {
        position: absolute;
        width: 0;
        height: 0;
        overflow: hidden;
        background-image:
url('https://www.thetuneproductions.com/wavesurfer/Button01_Play_100p_mouseover.png
');
    }
</style>
<!-- # NEUE VISUALISATION-BALKEN (10 Stück) # -->
<style>
    :root {
        --vis-bars-opacity: 0.5;                /* Gesamt-Opazität der Balken-
Ebene (0.0 bis 1.0) */
        --vis-bar-gradient-from: #FFFFFF;        /* Unten: Startfarbe des
Gradienten */
        --vis-bar-gradient-mid: #888888;         /* Mitte: Übergang */
        --vis-bar-gradient-to: #000000;         /* Oben: Endfarbe des Gradienten
*/
        --vis-bar-width: 12px;                  /* Breite jedes Balkens */
        --vis-bar-gap: 6px;                     /* Abstand zwischen den Balken */
        --vis-bar-max-height: 75px;             /* Maximale Höhe der Balken
(passt zur Ebene) */
        --vis-container-width: 220px;           /* Breite der gesamten
Visualisationsebene */
        --vis-container-height: 75px;          /* Höhe der Visualisationsebene
*/
    }
    .vis-bars-container {
        position: absolute;
        bottom: 0;
        left: 50%;
        transform: translateX(-50%);
        width: var(--vis-container-width);
        height: var(--vis-container-height);
        pointer-events: none;
        opacity: var(--vis-bars-opacity);
        display: flex;
        justify-content: center;
        align-items: flex-end;
        gap: var(--vis-bar-gap);
        z-index: 2;                            /* Über Needle, aber unter
anderen Elementen wenn nötig */
        /* NEU: Die Balken beginnen 5 px über dem unteren Rand */
        padding-bottom: 5px; /* &#8592; das ist die wichtigste Zeile */
        box-sizing: border-box; /* damit die Höhe nicht aufgebläht wird */
    }
    .vis-bar {
        width: var(--vis-bar-width);
        height: 0;                             /* Start bei 0, wird per JS
gesetzt */
        background: linear-gradient(
            to top,
            var(--vis-bar-gradient-from) 0%,
            var(--vis-bar-gradient-mid) 50%,
            var(--vis-bar-gradient-to) 100%
        );
        border-radius: 2px 2px 0 0;
        transition: height 0.08s ease-out;
        box-shadow: 0 0 6px rgba(255,255,255,0.3) inset;

```

```
}  
</style>
```



## Erstellen des Player-Containers im Body

An jeder gewünschten Stelle im Body kannst du den modularen Player-Container einfügen. Passe dabei die data-mp3-URL auf die gewünschte MP3-Datei an, den Songtitel usw. Der Container enthält bereits alle Buttons, Waveforms, Volumen- und Powermeter sowie die neue Visualisierungs-Ebene mit 10 Balken.

## Code:

```
<!-- = KOPIERBARER PLAYER-CONTAINER = -->

<div class="player-container" data-
mp3="https://www.thetuneproductions.com/music/Die_Panikplatte_Collection/Die%20P
anikplatte%201%20(Englisch).mp3">

    <div align="center" style="margin-bottom: -20px;">

        <table border="0" cellpadding="0" cellspacing="0" style="border-
collapse: collapse" bordercolor="#E8E8E8" width="790">

            <tr>

                <td align="center" height="40" width="110"
style="background: url('wavesurfer/Cellgraphics1.png') no-repeat; background-
size: 110px 40px;">

                </td>

                <td colspan="5" align="center" height="40" width="430"
style="background: url('wavesurfer/Cellgraphics4.png') no-repeat; background-
size: 430px 40px;">

                    <span class="the-name player-meta--songname">

                        <span style="font-weight: 700">

                            <font style="font-size: 16pt"
color="#FFFFFF">

                                THE PANIC PLATE #1 (English)</font>

                            </span>

                        </span>

                    </span>

                </td>

            </tr>

        </table>

    </div>

</div>
```

```

        </td>

        <td align="center" height="40" width="250"
style="background: url('wavesurfer/Cellgraphics8.png') no-repeat; background-
size: 250px 40px;">

        </td>

    </tr>

    <tr>

        <td height="150" width="110" align="center" colspan="2"
style="background: url('wavesurfer/Cellgraphics2.png') no-repeat; background-
size: 110px 150px;">

            <button class="play-button" style="background:
url('https://www.thetuneproductions.com/wavesurfer/Button01_Play.png') no-
repeat; width: 100px; height: 100px; border: none;"></button>

        </td>

        <td height="150" width="350" align="center"
style="background: url('wavesurfer/Cellgraphics3.png') no-repeat; background-
size: 350px 150px;">

            <div class="audioplayer">

                <div class="waveform-container">

                    <div class="placeholder-line"></div>

                    <div class="waveform-main"></div>

                </div>

                <div class="waveform-shadow-container">

                    <div class="waveform-shadow"></div>

                </div>

                <div class="waveform-overlay"></div>

                <div class="button-container">

                    <button class="restart-button"></button>

                    <button class="rewind-button"></button>

                    <button class="forward-button"></button>

                    <button class="col1-button"></button>

                    <button class="col2-button"></button>

                    <button class="backcol-button"></button>

                </div>

```

```

duration="0:00">&nbsp;</span>
<span class="duration" data-
time="0:00">&nbsp;</span>

</div>

</td>

<td height="150" width="40" align="center"
style="background: url('wavesurfer/Cellgraphics5.png') no-repeat; background-
size: 40px 150px;">

<div class="volume-meter-1">

<div class="volume-level-1"></div>

</div>

</td>

<td height="150" width="40" align="left"
style="background: url('wavesurfer/Cellgraphics6.png') no-repeat; background-
size: 40px 150px;">

<div class="volume-meter-2">

<div class="volume-level-2-bottom"></div>

<div class="volume-peak-2"></div>

<div class="volume-level-2-top"></div>

</div>

</td>

<td height="150" width="250" align="left" colspan="2" style="background:
url('wavesurfer/Cellgraphics7.png') no-repeat; background-size: 250px 150px;
position: relative;">

<div class="powermeter-container">

<font color="#C5C5C5">



</font>

<div class="needle-glow" style="background: #FEB885;"></div>

</div>

<!-- NEUE VISUALISATION-EBENE: 10 Balken, unten mittig -->

<div class="vis-bars-container">

<div class="vis-bar" data-bar-index="0"></div>

```

```

        <div class="vis-bar" data-bar-index="1"></div>

        <div class="vis-bar" data-bar-index="2"></div>

        <div class="vis-bar" data-bar-index="3"></div>

        <div class="vis-bar" data-bar-index="4"></div>

        <div class="vis-bar" data-bar-index="5"></div>

        <div class="vis-bar" data-bar-index="6"></div>

        <div class="vis-bar" data-bar-index="7"></div>

        <div class="vis-bar" data-bar-index="8"></div>

        <div class="vis-bar" data-bar-index="9"></div>

    </div>

</td>

    </tr>

    <tr>

        <td height="25" width="790" align="center" colspan="7"
style="background-position: 0% 0%; background-size: 110px 150px; background-
color:transparent; background-repeat:no-repeat; background-attachment:scroll;
font-size: 1px; line-height: 1px;" valign="top">

            <div align="center">

                <center>

                    <table border="0" cellpadding="0"
cellspacing="0" style="border-collapse: collapse" width="790">

                        <tr>

                            <td align="center"
width="35">&nbsp;</td>

                            <td align="center" width="140">

                                

                                </td>

                            <td align="center"
width="440">&nbsp;</td>

                            <td align="center" width="140">

                                

                                </td>

                        </tr>

```

```
width="35">&nbsp;</td>
                                <td align="center"
                                </tr>
                                </table>
                                </center>
                                </div>
                                </td>
                                </tr>
                                </table>
                                </div>
                                </div>
```

## Einfügen des Scripts ganz am Ende des Body.

Kopiere den kompletten JavaScript-Code direkt vor dem schließenden `</body>`-Tag. Dieses Script initialisiert jeden Player, lädt Peaks, erstellt AudioContext + Analysers und bindet alle Buttons und Visualisierungen an. Es sorgt auch für Lazy-Loading, sodass der Player erst startet, wenn er sichtbar ist.

## Code:

```
!-- = SCRIPT IMMER GANZ AM ENDE DES BODY (!) = -->

<script>

// =====

// JEDER Player hat eigenen AudioContext + Analyser (stabil gegen Wechsel)

// =====

let currentlyPlayingContainer = null;

async function initializePlayer(container) {

  const mp3Url = container.dataset.mp3;

  const peakUrl = mp3Url.replace('.mp3', '_peaks.json');

  function formatTime(seconds) {

    const minutes = Math.floor(seconds / 60);

    const secs = Math.floor(seconds % 60);

    return `${minutes}:${secs < 10 ? '0' + secs : secs}`;

  }

  // DOM-Elemente

  const waveformMain      = container.querySelector('.waveform-main');

  const waveformShadow    = container.querySelector('.waveform-shadow');

  const overlay           = container.querySelector('.waveform-overlay');

  const playButton        = container.querySelector('.play-button');
```

```

const duration          = container.querySelector('.duration');

const currentTime       = container.querySelector('.current-time');

const volumeLevel1      = container.querySelector('.volume-level-1');

const volumeLevel2Bottom = container.querySelector('.volume-level-2-bottom');

const volumePeak2       = container.querySelector('.volume-peak-2');

const volumeLevel2Top    = container.querySelector('.volume-level-2-top');

const needle            = container.querySelector('.powermeter-needle');

const needleGlow         = container.querySelector('.needle-glow');

const restartButton      = container.querySelector('.restart-button');

const rewindButton       = container.querySelector('.rewind-button');

const forwardButton      = container.querySelector('.forward-button');

const col1Button         = container.querySelector('.col1-button');

const col2Button         = container.querySelector('.col2-button');

const backcolButton      = container.querySelector('.backcol-button');

const audioplayer        = container.querySelector('.audioplayer');

const placeholderLine    = container.querySelector('.placeholder-line');

// NEU: Visualisation-Balken

const visBarsContainer   = container.querySelector('.vis-bars-container');

const visBars            = container.querySelectorAll('.vis-bar');

// CSS-Variablen

const wave1Color         = getComputedStyle(document.documentElement).getPropertyValue('--wave1-color').trim();

const wave1Progress      = getComputedStyle(document.documentElement).getPropertyValue('--wave1-progress').trim();

const wave2Color         = getComputedStyle(document.documentElement).getPropertyValue('--wave2-color').trim();

const wave2Progress      = getComputedStyle(document.documentElement).getPropertyValue('--wave2-progress').trim();

const wave2Type          =
parseInt(getComputedStyle(document.documentElement).getPropertyValue('--wave2-type').trim() || '1');

const waveWidth          =
parseInt(getComputedStyle(document.documentElement).getPropertyValue('--wave-width') || '800');

// Meter-Konstanten

```

```

    const meter1Sensitivity      =
parseFloat(getComputedStyle(document.documentElement).getPropertyValue('--meter1-
sensitivity')) || 3.5;

    const meter1UpdateInterval =
parseFloat(getComputedStyle(document.documentElement).getPropertyValue('--meter1-update-
interval')) || 30;

    const meter2Sensitivity      =
parseFloat(getComputedStyle(document.documentElement).getPropertyValue('--meter2-
sensitivity')) || 0.75;

    const meter2UpdateInterval =
parseFloat(getComputedStyle(document.documentElement).getPropertyValue('--meter2-update-
interval')) || 700;

    const meter3SmoothFactor     =
parseFloat(getComputedStyle(document.documentElement).getPropertyValue('--meter3-smooth-
factor')) || 0.6;

    const meter3Sensitivity      =
parseFloat(getComputedStyle(document.documentElement).getPropertyValue('--meter3-
sensitivity')) || 1.5;

    const meter3StartAngle       =
parseFloat(getComputedStyle(document.documentElement).getPropertyValue('--meter3-start-
angle')) || -50;

    const meter3MaxSwing         =
parseFloat(getComputedStyle(document.documentElement).getPropertyValue('--meter3-max-
swing')) || 55;

    const glowColor              = '#FEB885';

    const glowFactor             = 3.4;

    const glowHeightFactor       = 88.0;

    const glowSpreadFactor       = 7.5;

    const meterSmoothFactor      = 0.1;

    let wavesurferMain = null;

    let wavesurferShadow = null;

    let peaksLoaded = false;

    let isReady = false;

    let peakRms = 0;

    let lastUpdateMeter1 = 0;

    let lastUpdateMeter2 = 0;

    let lastUpdateMeter3 = 0;

    let currentAngle = meter3StartAngle;

    let audioContext = null;

    let analyser = null;

    let playerSource = null;

```



```

// Preload Mouseover-Bilder

[

  'https://www.thetuneproductions.com/wavesurfer/Button01_Play_100p_mouseover.png',

  'https://www.thetuneproductions.com/wavesurfer/Button01_pause_100p_mouseover.png',

  'https://www.thetuneproductions.com/wavesurfer/Arrow_left_mouseover.png',

  'https://www.thetuneproductions.com/wavesurfer/10Secbackmouseover.png',

  'https://www.thetuneproductions.com/wavesurfer/10Secforwardmouseover.png',

  'https://www.thetuneproductions.com/wavesurfer/Col1_mouseover.png',

  'https://www.thetuneproductions.com/wavesurfer/Col2_mouseover.png',

  'https://www.thetuneproductions.com/wavesurfer/BackCol_mouseover.png'

].forEach(src => { new Image().src = src; });

async function loadPeaks() {

  if (peaksLoaded) return;

  try {

    const response = await fetch(peakUrl);

    if (!response.ok) throw new Error('Peaks nicht gefunden');

    const json = await response.json();

    const peaks = json.data;

    wavesurferMain.load(mp3Url, [peaks]);

    wavesurferShadow.load(mp3Url, [peaks]);

    peaksLoaded = true;

  } catch {

    wavesurferMain.load(mp3Url);

    wavesurferShadow.load(mp3Url);

  }

}

async function ensureLoaded() {

  await loadPeaks();

  if (!isReady) {

    await new Promise(resolve => wavesurferMain.once('ready', resolve));

  }

}

```

```

}

async function setupAudioContextAndConnect() {

  if (!audioContext) {

    audioContext = new (window.AudioContext || window.webkitAudioContext)();

    analyser = audioContext.createAnalyser();

    analyser.fftSize = 2048;

    const media = wavesurferMain.getMediaElement();

    playerSource = audioContext.createMediaElementSource(media);

    playerSource.connect(analyser);

    analyser.connect(audioContext.destination);

  }

  if (audioContext.state === 'suspended') {

    await audioContext.resume();

  }

}

function rmsToDb(rms) {

  return Math.max(-60, 20 * Math.log10(rms) + 1);

}

function dbToAngle(db) {

  const minDb = -60;

  const rangeDb = meter3MaxSwing - meter3StartAngle;

  const normalizedDb = Math.max(minDb, db * meter3Sensitivity);

  const angleRange = (normalizedDb - minDb) / (0 - minDb);

  return meter3StartAngle + (angleRange * rangeDb);

}

// WaveSurfer Instanzen

wavesurferMain = WaveSurfer.create({

  container: waveformMain,

  waveColor: wave1Color,

  progressColor: wave1Progress,

  barWidth: 1.2,

```

```

    barGap: 1.8,

    height: parseInt(getComputedStyle(document.documentElement).getPropertyValue('--wave1-
height')) || '80'),

    cursorWidth: 1,

    cursorColor: '#FFFFFF',

    normalize: true,

    mediaControls: false,

    barAlign: 'bottom',

    backgroundColor: 'transparent'

});

const shadowOpts = {

    container: waveformShadow,

    waveColor: wave2Color,

    progressColor: wave2Progress,

    barWidth: 1.2,

    barGap: 1.8,

    height: parseInt(getComputedStyle(document.documentElement).getPropertyValue('--wave2-
height')) || '60'),

    cursorWidth: 0,

    interact: false,

    normalize: true,

    barAlign: 'center',

    backgroundColor: 'transparent'

};

wavesurferShadow = WaveSurfer.create(wave2Type === 2 ? { ...shadowOpts, renderer:
'Canvas' } : shadowOpts);

wavesurferMain.on('ready', () => {

    isReady = true;

    placeholderLine.style.opacity = '0';

    setTimeout(() => { placeholderLine.style.display = 'none'; }, 300);

    duration.textContent = formatTime(wavesurferMain.getDuration());

    duration.style.display = 'block';

    needle.style.transform = `rotate(${meter3StartAngle}deg)`;

```

```

        needleGlow.style.transform = `rotate(${meter3StartAngle}deg)`;

        needleGlow.style.opacity = 0;

    });

    wavesurferMain.on('finish', () => {

        wavesurferMain.stop();

        wavesurferMain.seekTo(0);

        wavesurferShadow.seekTo(0);

        playButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Button01_Play.png') no-repeat";

        currentlyPlayingContainer = null;

    });

    wavesurferMain.on('play', () => {

        playButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Button01_Pause.png') no-repeat";

        playButton.style.pointerEvents = 'auto';

        currentlyPlayingContainer = container;

    });

    wavesurferMain.on('pause', () => {

        currentAngle = meter3StartAngle;

        needle.style.transform = `rotate(${currentAngle}deg)`;

        needleGlow.style.transform = `rotate(${currentAngle}deg)`;

        needleGlow.style.opacity = 0;

        needleGlow.style.height = '0px';

        playButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Button01_Play.png') no-repeat";

        currentlyPlayingContainer = null;

    });

    wavesurferMain.on('loading', (percent) => {

        if (percent < 100) {

            playButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Button01_Pause.png') no-repeat";

            playButton.style.pointerEvents = 'none';

        } else {

            playButton.style.pointerEvents = 'auto';

```

```

    }

    });

    wavesurferMain.on('timeupdate', (currentTimeValue) => {

        currentTime.textContent = formatTime(currentTimeValue);

        currentTime.style.display = 'block';

        const cursorPos = (currentTimeValue / wavesurferMain.getDuration()) * waveWidth;

        currentTime.style.left = `${Math.min(cursorPos, waveWidth - 20)}px`;

        if (currentlyPlayingContainer === container && analyser && wavesurferMain.isPlaying())
        {

            const bufferLength = analyser.frequencyBinCount;

            const dataArray = new Uint8Array(bufferLength);

            analyser.getBytesTimeDomainData(dataArray);

            let sum = 0;

            for (let i = 0; i < bufferLength; i++) {

                const value = (dataArray[i] - 128) / 128;

                sum += value * value;

            }

            const rms = Math.sqrt(sum / bufferLength);

            if (Date.now() - lastUpdateMeter1 >= meter1UpdateInterval) {

                const smoothed = volumeLevel1.dataset.smoothedRms

                ? (parseFloat(volumeLevel1.dataset.smoothedRms) * meterSmoothFactor) + (rms * (1
- meterSmoothFactor))

                : rms;

                volumeLevel1.dataset.smoothedRms = smoothed;

                const adj = Math.min(smoothed * meter1Sensitivity, 1);

                volumeLevel1.style.height = `${(1 - adj) * 100}px`;

                lastUpdateMeter1 = Date.now();

            }

            peakRms = Math.max(peakRms, rms * meter1Sensitivity);

            if (Date.now() - lastUpdateMeter2 >= meter2UpdateInterval) {

                const peakPos = Math.min(peakRms * meter2Sensitivity * 100, 100);

                volumeLevel2Bottom.style.height = `${peakPos}px`;

```

```

    volumePeak2.style.bottom = `${peakPos}px`;

    volumeLevel2Top.style.height = `${100 - peakPos - 3}px`;

    peakRms = 0;

    lastUpdateMeter2 = Date.now();
}

if (Date.now() - lastUpdateMeter3 >= meter1UpdateInterval) {

    const adjRms = rms * meter3Sensitivity;

    const db = rmsToDb(adjRms);

    const targetAngle = dbToAngle(db);

    currentAngle += (targetAngle - currentAngle) * meter3SmoothFactor;

    needle.style.transform = `rotate(${currentAngle}deg)`;

    needleGlow.style.transform = `rotate(${currentAngle}deg)`;

    const h = 9 * (glowHeightFactor / 10);

    needleGlow.style.height = `${h}px`;

    const op = Math.min(0.8, rms * glowFactor);

    const blur = 10 + 20 * (glowSpreadFactor / 10);

    needleGlow.style.boxShadow = `0 0 ${blur}px ${op} ${glowColor}`;

    needleGlow.style.opacity = op;

    lastUpdateMeter3 = Date.now();
}

// NEU: 10 Visualisation-Balken updaten (FFT-Daten nutzen)

if (visBarsContainer && visBars.length === 10 && analyser &&
wavesurferMain.isPlaying()) {

    const bufferLength = analyser.frequencyBinCount;

    const dataArray = new Uint8Array(bufferLength);

    analyser.getByteFrequencyData(dataArray); // Frequency statt TimeDomain für
bessere Balken

    // Teile die Frequenzen in 10 Gruppen (niedrig &#8594; hoch)

    const binStep = Math.floor(bufferLength / 20); // Etwas mehr Bins &#8594; bessere
Auflösung

    for (let i = 0; i < 10; i++) {

        let sum = 0;

        for (let j = 0; j < binStep; j++) {

```

```

        const idx = i * binStep + j;

        if (idx < bufferLength) sum += dataArray[idx];
    }

    const avg = sum / binStep;

    // === Dämpfung der tiefen Frequenzen ===

    let heightPercent = avg / 255; // Basiswert 0-1

    if      (i === 0) heightPercent *= 0.60;    // tiefster Bass -62%

    else if (i === 1) heightPercent *= 0.68;    // tiefer Bass   -45%

    else if (i === 2) heightPercent *= 0.88;    // Bass-Mitten  -22%

    // Optional: Höhen etwas anheben, damit sie besser sichtbar sind

    else if (i >= 6) heightPercent *= 1.25;    // obere Mitten + Höhen +25%

    heightPercent = Math.min(1, heightPercent);

    const heightPx = heightPercent *
parseFloat(getComputedStyle(document.documentElement).getPropertyValue('--vis-bar-max-height')) || '75';

    // Umdrehen: niedrige i &#8594; hoher Balken-Index (rechts)

    const barIndex = 9 - i;    // <--- Bars EQ-Freq. umkehren - links: niedrige Frq,
rechts: hohe Frq

    visBars[i].style.height = `${heightPx}px`;
}

}

}

const shadowSeek = currentTimeValue / wavesurferMain.getDuration();

if (isFinite(shadowSeek)) wavesurferShadow.seekTo(shadowSeek);

});

// Play/Pause (bereits korrekt)

playButton.addEventListener('click', async () => {

```

```

    if (currentlyPlayingContainer && currentlyPlayingContainer !== container) {

        const otherWs = currentlyPlayingContainer.querySelector('.waveform-
main')?.__wavesurfer;

        if (otherWs) otherWs.pause();

    }

    await ensureLoaded();

    await setupAudioContextAndConnect();

    if (wavesurferMain.isPlaying()) {

        wavesurferMain.pause();

    } else {

        wavesurferMain.play();

    }

});

// Mouseover Play/Pause

playButton.addEventListener('mouseover', () => {

    const isPlaying = wavesurferMain?.isPlaying();

    playButton.style.background = isPlaying

    ?

"url('https://www.thetuneproductions.com/wavesurfer/Button01_pause_100p_mouseover.png')
no-repeat"

    :

"url('https://www.thetuneproductions.com/wavesurfer/Button01_Play_100p_mouseover.png') no-
repeat";

});

playButton.addEventListener('mouseout', () => {

    const isPlaying = wavesurferMain?.isPlaying();

    playButton.style.background = isPlaying

    ? "url('https://www.thetuneproductions.com/wavesurfer/Button01_Pause.png') no-
repeat"

    : "url('https://www.thetuneproductions.com/wavesurfer/Button01_Play.png') no-
repeat";

});

// Overlay Seek

overlay.addEventListener('click', async (e) => {

    await ensureLoaded();

```



```

const rect = overlay.getBoundingClientRect();

const x = e.clientX - rect.left;

const seek = x / waveWidth;

if (isFinite(seek)) {

    wavesurferMain.seekTo(seek);

    wavesurferShadow.seekTo(seek);

    if (wavesurferMain.isPlaying()) {

        await setupAudioContextAndConnect();

    }

}

});

// === FIX: Restart Button - alten Player stoppen ===

restartButton.addEventListener('click', async () => {

    // WICHTIG: Alten Player immer pausieren, wenn ein anderer läuft

    if (currentlyPlayingContainer && currentlyPlayingContainer !== container) {

        const otherWs = currentlyPlayingContainer.querySelector('.waveform-
main')?.__wavesurfer;

        if (otherWs) otherWs.pause();

    }

    await ensureLoaded();

    wavesurferMain.seekTo(0);

    wavesurferShadow.seekTo(0);

    await setupAudioContextAndConnect();

    wavesurferMain.play();

    playButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Button01_Pause.png') no-repeat";

});

restartButton.addEventListener('mouseover', () => restartButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Arrow_left_mouseover.png') no-
repeat");

restartButton.addEventListener('mouseout', () => restartButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Arrow_left.png') no-repeat");

// === FIX: Rewind 10s ===

rewindButton.addEventListener('click', async () => {

```

```

    if (currentlyPlayingContainer && currentlyPlayingContainer !== container) {

        const otherWs = currentlyPlayingContainer.querySelector('.waveform-
main')?.__wavesurfer;

        if (otherWs) otherWs.pause();

    }

    await ensureLoaded();

    const cur = wavesurferMain.getCurrentTime();

    const newT = Math.max(0, cur - 10);

    wavesurferMain.seekTo(newT / wavesurferMain.getDuration());

    wavesurferShadow.seekTo(newT / wavesurferMain.getDuration());

    await setupAudioContextAndConnect();

    wavesurferMain.play();

    playButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Button01_Pause.png') no-repeat";

});

rewindButton.addEventListener('mouseover', () => rewindButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/10Secbackmouseover.png') no-repeat");

rewindButton.addEventListener('mouseout', () => rewindButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/10Secback.png') no-repeat");

// === FIX: Forward 10s ===

forwardButton.addEventListener('click', async () => {

    if (currentlyPlayingContainer && currentlyPlayingContainer !== container) {

        const otherWs = currentlyPlayingContainer.querySelector('.waveform-
main')?.__wavesurfer;

        if (otherWs) otherWs.pause();

    }

    await ensureLoaded();

    const cur = wavesurferMain.getCurrentTime();

    const dur = wavesurferMain.getDuration();

    const newT = Math.min(dur, cur + 10);

    wavesurferMain.seekTo(newT / dur);

    wavesurferShadow.seekTo(newT / dur);

    await setupAudioContextAndConnect();

    wavesurferMain.play();

```

```

        playButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Button01_Pause.png') no-repeat";

    });

    forwardButton.addEventListener('mouseover', () => forwardButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/10Secforwardmouseover.png') no-
repeat");

    forwardButton.addEventListener('mouseout', () => forwardButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/10Secforward.png') no-repeat");

    // Farbenreihenfolge ändern

    // Farbenreihenfolge ändern

    // Farbenreihenfolge ändern

    // Farbenreihenfolge ändern

    // Farbenreihenfolge ändern

    // Farbenreihenfolge ändern

    // Farbenreihenfolge ändern

    // Farbenreihenfolge ändern

    // Farbenreihenfolge - Wellenform 1 (obere Haupt-Welle)

    const collColors = ['#F0FF00', '#FF3000', '#FFB400', '#FF007E', '#0084FF', '#00FF36'];

    let collIndex = 2; // Start mit Index 0 & #8594; #F0FF00 (helles Gelb-Orange)

    // WICHTIG: Sofort beim Laden die Startfarbe setzen (überschreibt den alten CSS-Wert)

    document.documentElement.style.setProperty('--wave1-color', collColors[collIndex]);

    wavesurferMain.setOptions({ waveColor: collColors[collIndex] });

    // Mouseover/Mouseout wie bisher (unverändert)

    collButton.addEventListener('mouseover', () => collButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Coll_mouseover.png') no-repeat");

    collButton.addEventListener('mouseout', () => collButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Coll.png') no-repeat");

    // Click-Handler: Beim Klicken weiter wechseln

    collButton.addEventListener('click', () => {

        collIndex = (collIndex + 1) % collColors.length;

        document.documentElement.style.setProperty('--wave1-color', collColors[collIndex]);

        wavesurferMain.setOptions({ waveColor: collColors[collIndex] });

```

```

});

    col1Button.addEventListener('mouseover', () => col1Button.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Col1_mouseover.png') no-repeat");

    col1Button.addEventListener('mouseout', () => col1Button.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Col1.png') no-repeat");

    const col2Colors = ['#FFFFFF', '#444444', '#777777', '#AAAAAA'];

    let col2Index = 0;

    col2Button.addEventListener('click', () => {

        col2Index = (col2Index + 1) % col2Colors.length;

        document.documentElement.style.setProperty('--wave2-color', col2Colors[col2Index]);

        wavesurferShadow.setOptions({ waveColor: col2Colors[col2Index] });

    });

    col2Button.addEventListener('mouseover', () => col2Button.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Col2_mouseover.png') no-repeat");

    col2Button.addEventListener('mouseout', () => col2Button.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/Col2.png') no-repeat");

    const backcolColors = ['transparent', '#000000', '#111111', '#222222'];

    let backcolIndex = 0;

    // Initial beim Laden auf transparent setzen (das fehlte!)

    audioplayer.style.background = 'transparent';

    waveformMain.style.background = 'transparent';

    waveformShadow.style.background = 'transparent';

    backcolButton.addEventListener('click', () => {

        backcolIndex = (backcolIndex + 1) % backcolColors.length;

        const bg = backcolColors[backcolIndex];

        audioplayer.style.background = bg;

        waveformMain.style.background = bg;

        waveformShadow.style.background = bg;

    });

    backcolButton.addEventListener('mouseover', () => backcolButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/BackCol_mouseover.png') no-repeat");

    backcolButton.addEventListener('mouseout', () => backcolButton.style.background =
"url('https://www.thetuneproductions.com/wavesurfer/BackCol.png') no-repeat");

    waveformMain.__wavesurfer = wavesurferMain;

```

```
}

// Lazy Loading

const observer = new IntersectionObserver((entries) => {

  entries.forEach(entry => {

    if (entry.isIntersecting) {

      initializePlayer(entry.target);

      observer.unobserve(entry.target);

    }

  });

}, { threshold: 0.1 });

document.querySelectorAll('.player-container').forEach(container => {

  observer.observe(container);

});

</script>
```

## **- Letzter Schritt -**

### **Bilder und Ressourcen prüfen**

Stelle sicher, dass alle im CSS und Script referenzierten Bilder (Buttons, Backgrounds, Needle, Volume-Level usw.) erreichbar sind, entweder auf deinem Server oder über die angegebenen URLs. Hierzu dann einfach die hier im Code angegebenen Pfade und Bilddateien umbenennen.