```
<!doctype html>
<html lang="id">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,initial-scale=1" />
<title>TTS Slideshow — Player & Export (ID/JP)</title>
<style>
:root{--accent:#3b82f6}
body{font-family:system-ui,-apple-system,Segoe UI,Roboto,"Noto
Sans",Arial;margin:0;padding:16px;background:#f7fafc;color:#111}
h1{font-size:20px;margin:0 0 12px}
.container{max-width:1100px;margin:0 auto}
.panel{background:white;border-radius:10px;padding:12px;margin-bottom:12px;box-shadow:
0 6px 18px rgba(15,23,42,0.06)}
.row{display:flex;gap:8px;align-items:center}
input[type=file]{display:block}
.clips{display:flex;flex-direction:column;gap:8px;max-height:360px;overflow:auto;padding:4px
}
.clip{display:flex;gap:8px;align-items:center;padding:8px;border:1px dashed
#e5e7eb;border-radius:8px;background:#fff}
.thumb{width:100px;height:70px;object-fit:cover;border-radius:6px;background:#eee}
.meta{flex:1}
textarea{width:100%;height:64px}
button{background:var(--accent);color:white;border:0;padding:8px
12px;border-radius:8px;cursor:pointer}
.small{font-size:13px;padding:6px 8px}
#previewCanvas{width:100%;height:360px;background:#000;border-radius:8px}
.sfx-list{display:flex;gap:8px;flex-wrap:wrap}
.sfx-item{border:1px solid #e5e7eb;padding:6px;border-radius:6px}
label.switch{display:inline-flex;align-items:center;gap:8px}
.hint{font-size:13px;color:#6b7280}
.controls{display:flex;gap:6px;align-items:center;margin-top:8px}
.progress{height:8px;background:#e6eefc;border-radius:8px;overflow:hidden;margin-top:8px
}
.progress > div{height:100%;width:0;background:var(--accent)}
.status{font-size:13px;color:#6b7280;margin-left:8px}
</style>
</head>
<body>
<div class="container">
  <h1>TTS Slideshow — Player & Export (ID / JP)</h1>

  <div class="panel">
    <div style="display:flex;gap:12px;align-items:center;flex-wrap:wrap">
      <div>
        <label class="hint">Upload gambar (bisa beberapa):</label><br>
        <input id="inputImages" type="file" accept="image/*" multiple />
      </div>
      <div>
        <label class="hint">Upload efek suara (opsional):</label><br>
        <input id="inputSfx" type="file" accept="audio/*" multiple />
      </div>
      <div>
        <label class="hint">Pilih bahasa TTS:</label><br>
        <select id="langSelect">
          <option value="id-ID">Bahasa Indonesia</option>
          <option value="ja-JP">日本語 (Jepang)</option>
        </select>
      </div>
      <div>
        <label class="hint">Pilih voice (browser):</label><br>
        <select id="voiceSelect"></select>
      </div>
      <div>
        <label class="hint">Kecepatan:</label><br>
        <input id="rate" type="range" min="0.6" max="1.6" step="0.1" value="1" /> <span
id="rateLabel">1</span>
      </div>
    </div>

    <hr />
    <div class="hint">Daftar klip (urutkan drag & drop). Untuk tiap klip: tulis narasi, atur durasi
(atau auto), pilih SFX.</div>
    <div class="clips" id="clips"></div>

    <div style="margin-top:8px">
      <div class="controls">
        <button id="btnPrev" class="small">⏮ Prev</button>
        <button id="btnRew" class="small">⏪ -5s</button>
        <button id="btnPlay" class="small">▶ Play</button>
        <button id="btnPause" class="small" disabled>⏸ Pause</button>
        <button id="btnStop" class="small" disabled>⏹ Stop</button>
        <button id="btnFwd" class="small">+5s ⏩</button>
        <button id="btnNext" class="small">Next ⏭</button>
        <div class="status" id="playerStatus">Idle</div>
      </div>
      <div class="progress"><div id="progressBar"></div></div>
    </div>

    <div style="margin-top:8px">
      <button id="exportBtn" class="small">Export WebM (Eksperimental)</button>
      <button id="downloadJson" class="small">Download JSON</button>
    </div>
  </div>

  <div class="panel">
```

```
    <div class="row" style="justify-content:space-between;align-items:flex-start">
      <div style="flex:1;margin-right:12px">
        <canvas id="previewCanvas" width="1280" height="720"></canvas>
      </div>
      <div style="width:320px">
        <h3 style="margin-top:0">Efek suara tersedia</h3>
        <div id="sfxContainer" class="sfx-list"><div class="hint">(Belum ada SFX)</div></div>
        <hr />
        <div class="hint"><strong>Note:</strong> WebM export mencoba merekam canvas +
audio dari SFX (WebAudio). Browser biasanya **tidak** merekam audio dari
speechSynthesis ke MediaRecorder. Untuk hasil yang pasti lengkap (narasi + audio),
gunakan konversi server-side/FFmpeg atau TTS service eksternal.</div>
      </div>
    </div>
  </div>
</div>

<script>
// --- state ---
let clips = []; // {imgName,imgUrl,text,duration,auto,sfx:[]}
let sfxFiles = []; // {name,url,file}
let currentIndex = 0;
let playing = false;
let clipStartTime = 0;
let clipElapsedTimer = null;
let clipResolve = null;

// UI refs
const clipsEl = document.getElementById('clips');
const inputImages = document.getElementById('inputImages');
const inputSfx = document.getElementById('inputSfx');
const sfxContainer = document.getElementById('sfxContainer');
const langSelect = document.getElementById('langSelect');
const voiceSelect = document.getElementById('voiceSelect');
const rate = document.getElementById('rate');
const rateLabel = document.getElementById('rateLabel');
const canvas = document.getElementById('previewCanvas');
const ctx = canvas.getContext('2d');
const btnPlay = document.getElementById('btnPlay');
const btnPause = document.getElementById('btnPause');
const btnStop = document.getElementById('btnStop');
const btnPrev = document.getElementById('btnPrev');
const btnNext = document.getElementById('btnNext');
const btnRew = document.getElementById('btnRew');
const btnFwd = document.getElementById('btnFwd');
const progressBar = document.getElementById('progressBar');
const playerStatus = document.getElementById('playerStatus');
```

```
rate.addEventListener('input',()=>rateLabel.textContent=rate.value);

// --- helpers ---
function readFileAsDataURL(file){ return new Promise((res,rej)=>{ const fr=new
FileReader(); fr.onload=()=>res(fr.result); fr.onerror=rej; fr.readAsDataURL(file); }); }
function fileToBlobUrl(dataUrl){ const arr = dataUrl.split(','); const mime =
arr[0].match(/:(.*?);/)[1]; const bstr = atob(arr[1]); let n=bstr.length; const u8 = new
Uint8Array(n); while(n--){ u8[n]=bstr.charCodeAt(n); } const blob = new
Blob([u8],{type:mime}); return URL.createObjectURL(blob); }
function sleep(ms){ return new Promise(r=>setTimeout(r,ms)); }
function setStatus(t){ playerStatus.textContent = t; }

// safe escape
function escapeHtml(s){ return
(s||'').replace(/&/g,'&amp;').replace(/</g,'&lt;').replace(/>/g,'&gt;'); }

// --- Clip UI ---
function makeClipElement(i, clip){
  const el = document.createElement('div'); el.className='clip'; el.draggable=true;
el.dataset.index=i;
  const thumb = document.createElement('img'); thumb.className='thumb'; thumb.src =
clip.imgUrl;
  const meta = document.createElement('div'); meta.className='meta';
  meta.innerHTML = `
    <div style="display:flex;gap:8px;align-items:center">
      <strong>#${i+1}</strong>
      <button class="small" data-action="up">▲</button>
      <button class="small" data-action="down">▼</button>
      <button class="small" data-action="remove"
style="background:#ef4444">Hapus</button>
    </div>
    <div style="margin-top:6px"><textarea placeholder="Tulis
narasi...">${escapeHtml(clip.text||'')}</textarea></div>
    <div style="display:flex;gap:8px;margin-top:6px;align-items:center">
      Durasi (detik): <input type="number" min="1" value="${clip.duration||5}"
style="width:80px" />
      <label class="switch"><input type="checkbox" ${clip.auto? 'checked':''} /> Auto (ikuti
TTS)</label>
      <button class="small" data-action="preview-clip">▶ Pratinjau</button>
    </div>
    <div style="margin-top:6px">Pilih efek suara: <div class="sfx-picker"
style="margin-top:6px"></div></div>
  `;
  el.appendChild(thumb); el.appendChild(meta);

  meta.querySelector('button[data-action="up"]').onclick = ()=>{ if(i>0){
[clips[i-1],clips[i]]=[clips[i],clips[i-1]]; renderClips(); } };
```

```
  meta.querySelector('button[data-action="down"]').onclick = ()=>{ if(i<clips.length-1){
[clips[i+1],clips[i]]=[clips[i],clips[i+1]]; renderClips(); } };
  meta.querySelector('button[data-action="remove"]').onclick = ()=>{ clips.splice(i,1);
renderClips(); };

  const ta = meta.querySelector('textarea'); ta.addEventListener('input', e=>{ clips[i].text =
e.target.value; });
  const dur = meta.querySelector('input[type=number]'); dur.addEventListener('input', e=>{
clips[i].duration = parseFloat(e.target.value)||3; });
  const autoCB = meta.querySelector('input[type=checkbox]');
autoCB.addEventListener('change', e=>{ clips[i].auto = e.target.checked; });

  meta.querySelector('button[data-action="preview-clip"]').addEventListener('click', ()=>{
playSingleClip(i); });

  // sfx picker population
  const picker = meta.querySelector('.sfx-picker');
  function refreshPicker(){
    picker.innerHTML='';
    sfxFiles.forEach((s,si)=>{
      const d = document.createElement('div'); d.className='sfx-item';
      const id = `sfx-${i}-${si}`;
      d.innerHTML = `<label><input type="checkbox" id="${id}" ${clip.sfx &&
clip.sfx.includes(si)?'checked':''}/> ${escapeHtml(s.name)}</label> <button class="small"
data-play="${si}">▶</button>`;
      picker.appendChild(d);
      d.querySelector('input').addEventListener('change', e=>{
        clips[i].sfx = clips[i].sfx||[];
        if(e.target.checked){ if(!clips[i].sfx.includes(si)) clips[i].sfx.push(si); }
        else clips[i].sfx = clips[i].sfx.filter(x=>x!==si);
      });
      d.querySelector('button').addEventListener('click', ()=>{ new
Audio(s.url).play().catch(()=>{}); });
    });
  }
  refreshPicker();

  // drag events
  el.addEventListener('dragstart', ev=>{ ev.dataTransfer.setData('text/plain', i);
el.style.opacity=0.4; });
  el.addEventListener('dragend', ()=>el.style.opacity=1);
  el.addEventListener('dragover', ev=>ev.preventDefault());
  el.addEventListener('drop', ev=>{
    ev.preventDefault();
    const from = Number(ev.dataTransfer.getData('text/plain'));
    const to = i;
    if(!isNaN(from)){ const it = clips.splice(from,1)[0]; clips.splice(to,0,it); renderClips(); }
  });
  return el;
}

function renderClips(){
  clipsEl.innerHTML='';
  clips.forEach((c,i)=>clipsEl.appendChild(makeClipElement(i,c)));
  refreshSfxUI();
}

// sfx UI
function refreshSfxUI(){
  sfxContainer.innerHTML = '';
  if(sfxFiles.length===0){ sfxContainer.innerHTML = '<div class="hint">(Belum ada
SFX)</div>'; return; }
  sfxFiles.forEach((s,si)=>{
    const d = document.createElement('div'); d.className='sfx-item';
    d.innerHTML = `${escapeHtml(s.name)} <button class="small"
data-play="${si}">▸</button> <button class="small" data-download="${si}">↓</button>`;
    sfxContainer.appendChild(d);
    d.querySelector('button[data-play]').addEventListener('click', ()=> new
Audio(s.url).play().catch(()=>{}));
    d.querySelector('button[data-download]').addEventListener('click', ()=>{ const a =
document.createElement('a'); a.href = s.url; a.download = s.name; a.click(); });
  });
}

// voices
function populateVoices(){
  const voices = speechSynthesis.getVoices();
  voiceSelect.innerHTML='';
  voices.forEach(v=>{ const opt = document.createElement('option'); opt.value=v.name;
opt.textContent = `${v.name} — ${v.lang}`; voiceSelect.appendChild(opt); });
}
speechSynthesis.onvoiceschanged = populateVoices;
langSelect.addEventListener('change', populateVoices);

// show image on canvas
function showImage(imgUrl){
  return new Promise(res=>{
    const img = new Image();
    img.onload = ()=>{
      ctx.fillStyle = 'black';
      ctx.fillRect(0,0,canvas.width,canvas.height);
      const ratio = Math.min(canvas.width/img.width, canvas.height/img.height);
      const nw = img.width*ratio; const nh = img.height*ratio;
      ctx.drawImage(img, (canvas.width-nw)/2, (canvas.height-nh)/2, nw, nh);
      res();
    };
    img.src = imgUrl;
  });
}

// --- playback engine for sequence ---
async function playAll(fromIndex=0){
  if(clips.length===0) return;
  playing = true;
  setControlsPlaying(true);
  currentIndex = fromIndex;
  for(let i = fromIndex; i < clips.length; i++){
    if(!playing) break;
    currentIndex = i;
    updateProgress(0, i);
    await showImage(clips[i].imgUrl);
    await playClip(clips[i], i);
  }
  playing = false;
  setControlsPlaying(false);
  updateProgress(1, clips.length-1);
  setStatus('Selesai');
}

function stopAll(){
  playing = false;
  if(clipResolve) clipResolve(); // resolve waiting clip promise
  clipResolve = null;
  speechSynthesis.cancel();
  clearInterval(clipElapsedTimer);
  setControlsPlaying(false);
  setStatus('Stopped');
  updateProgress(0, currentIndex);
}

function pauseAll(){
  speechSynthesis.pause();
  playing = false;
  setControlsPlaying(false);
  setStatus('Paused');
  clearInterval(clipElapsedTimer);
}

function resumeAll(){
  if(!playing){
    speechSynthesis.resume();
    playing = true;
    setControlsPlaying(true);
    setStatus('Playing');
    // continue: playAll will still be running if clip wasn't resolved; else start next
  }
}

// play single clip (used for preview button)
async function playSingleClip(idx){
  setStatus('Preview clip '+(idx+1));
  await showImage(clips[idx].imgUrl);
  await playClip(clips[idx], idx);
  setStatus('Selesai preview');
}

// play clip: plays SFX via Audio and TTS via speechSynthesis
function playClip(clip, idx){
  return new Promise(async (res)=>{
    // play SFX first (fire and forget)
    const sfxPlayers = (clip.sfx||[]).map(si=>{
      try{ const a = new Audio(sfxFiles[si].url); a.play().catch(()=>{}); return a; }catch(e){return
null;}
    });

    // compute wait time
    let waitMs = 3000;
    if(clip.auto){
      // heuristik: words / speed
      const words = (clip.text||'').split(/\s+/).filter(Boolean).length;
      const wps = 2.5 * (parseFloat(rate.value)||1); // words per second approx
      waitMs = Math.max(800, Math.round((words / wps) * 1000));
    } else waitMs = (parseFloat(clip.duration)||3) * 1000;

    // If text present, speak and wait for onend
    if(clip.text && clip.text.trim()!==''){
      const u = new SpeechSynthesisUtterance(clip.text);
      u.lang = langSelect.value;
      u.rate = parseFloat(rate.value)||1;
      const vName = voiceSelect.value;
      const voices = speechSynthesis.getVoices();
      const found = voices.find(v=>v.name===vName);
      if(found) u.voice = found;

      u.onstart = ()=>{ clipStartTime = Date.now(); startClipTimer(waitMs, idx);
setStatus(`Playing klip ${idx+1}`); };
      u.onend = ()=>{ clearInterval(clipElapsedTimer); setStatus('Menunggu...'); res(); };
      u.onerror = ()=>{ clearInterval(clipElapsedTimer); setStatus('TTS error, lanjut'); res(); };
      speechSynthesis.speak(u);
      clipResolve = res; // allow stopAll to resolve
    } else {
      // no text: just wait for duration
      startClipTimer(waitMs, idx);
      setStatus(`Menampilkan klip ${idx+1} (${Math.round(waitMs/1000)}s)`);
      setTimeout(()=>{ clearInterval(clipElapsedTimer); res(); }, waitMs);
    }
  });
}

function startClipTimer(durationMs, clipIdx){
  const start = Date.now();
  clearInterval(clipElapsedTimer);
  clipElapsedTimer = setInterval(()=>{
    const elapsed = Date.now() - start;
    const pct = Math.min(1, elapsed / durationMs);
    updateProgress(pct, clipIdx);
    if(pct>=1) clearInterval(clipElapsedTimer);
  }, 120);
}

function updateProgress(pct, clipIdx){
  // overall: simple: (clipIdx + pct) / total
  const total = Math.max(1, clips.length);
  const value = ((clipIdx) + pct) / total;
  progressBar.style.width = (value*100)+'%';
}

// control state UI
function setControlsPlaying(isPlaying){
  btnPlay.disabled = isPlaying;
  btnPause.disabled = !isPlaying;
  btnStop.disabled = !isPlaying;
}

// --- export (experimental) ---
async function exportRecording(){
  if(clips.length===0){ alert('Belum ada klip'); return; }
  if(!confirm('Export WebM eksperimen: TTS (speechSynthesis) MUNGKIN tidak direkam
oleh browser. Kalau kamu butuh narasi di file, gunakan FFmpeg (lihat petunjuk). Lanjut?'))
return;
  setStatus('Preparing export...');
  // prepare canvas stream
  const videoStream = canvas.captureStream(30);
  const audioCtx = new (window.AudioContext || window.webkitAudioContext)();
  const dest = audioCtx.createMediaStreamDestination();
  const master = audioCtx.createGain(); master.connect(dest);
  // decode SFX ahead
  const sfxBuffers = [];
  for(const s of sfxFiles){
    try{
      const ab = await fetch(s.url).then(r=>r.arrayBuffer());
      const buf = await audioCtx.decodeAudioData(ab);
      sfxBuffers.push(buf);
    }catch(e){
      sfxBuffers.push(null);
    }
  }
  // combine media tracks
  const combined = new MediaStream();
  videoStream.getVideoTracks().forEach(t=>combined.addTrack(t));
  dest.stream.getAudioTracks().forEach(t=>combined.addTrack(t));
  const recorder = new MediaRecorder(combined);
  const chunks = [];
  recorder.ondataavailable = e=>chunks.push(e.data);
  recorder.start();
  setStatus('Recording...');
  // play slideshow but route SFX into audioCtx
  for(const c of clips){
    await showImage(c.imgUrl);
    const now = audioCtx.currentTime + 0.05;
    const sDurPromises = (c.sfx||[]).map(si=>{
      const buf = sfxBuffers[si];
      if(!buf) return Promise.resolve(0);
      const src = audioCtx.createBufferSource(); src.buffer = buf; src.connect(master);
      src.start(now);
      return Promise.resolve(buf.duration*1000);
    });
    // try to speak (this will play locally but often not be captured)
    if(c.text && c.text.trim()!==''){
      const u = new SpeechSynthesisUtterance(c.text);
      u.lang = langSelect.value; u.rate = parseFloat(rate.value)||1;
      const vs = speechSynthesis.getVoices(); const found =
vs.find(v=>v.name===voiceSelect.value);
      if(found) u.voice = found;
      speechSynthesis.speak(u);
    }
    // wait heuristic
    let waitMs = 2000;
    if(c.auto){ const words = (c.text||'').split(/\s+/).filter(Boolean).length; waitMs =
Math.max(1200, (words/2.5)*(1000/(parseFloat(rate.value)||1))); } else waitMs =
(parseFloat(c.duration)||3)*1000;
    const sfxDur = await Promise.all(sDurPromises);
    const maxSfx = sfxDur.length? Math.max(...sfxDur):0;
    const finalWait = Math.max(waitMs, maxSfx + 200);
    await sleep(finalWait);
  }
  recorder.stop();
  await new Promise(r=> recorder.onstop = r);
  const blob = new Blob(chunks, {type:'video/webm'});
  const url = URL.createObjectURL(blob);
  const a = document.createElement('a'); a.href = url; a.download = 'slideshow.webm';
a.click();
  URL.revokeObjectURL(url);
  audioCtx.close();
  setStatus('Export selesai (WebM).');
}

// --- wiring UI events ---
inputImages.addEventListener('change', async e=>{
  const files = Array.from(e.target.files);
  for(const f of files){
    const url = await readFileAsDataURL(f);
    clips.push({imgName:f.name,imgUrl:url,text:'',duration:4,auto:false,sfx:[]});
  }
  renderClips();
  if(clips.length>0) showImage(clips[0].imgUrl);
});

inputSfx.addEventListener('change', async e=>{
  const files = Array.from(e.target.files);
  for(const f of files){
    const url = await readFileAsDataURL(f);
    sfxFiles.push({name:f.name,url:fileToBlobUrl(url),file:f});
  }
  refreshSfxUI();
  renderClips();
});

btnPlay.addEventListener('click', ()=>{ if(!playing) { playAll(currentIndex || 0);
setStatus('Playing'); } });
btnPause.addEventListener('click', ()=>{ pauseAll(); });
btnStop.addEventListener('click', ()=>{ stopAll(); });
btnPrev.addEventListener('click', ()=>{ if(currentIndex>0) currentIndex--;
showImage(clips[currentIndex].imgUrl); updateProgress(0, currentIndex); });
btnNext.addEventListener('click', ()=>{ if(currentIndex<clips.length-1) currentIndex++;
showImage(clips[currentIndex].imgUrl); updateProgress(0, currentIndex); });
btnRew.addEventListener('click', ()=>{ // cannot seek TTS; just adjust UI progress heuristic
  // just re-show current image
  showImage(clips[currentIndex].imgUrl); setStatus('Rewind -5s (heuristic)');
});
btnFwd.addEventListener('click', ()=>{ showImage(clips[currentIndex].imgUrl);
setStatus('Forward +5s (heuristic)'); });

document.getElementById('exportBtn').addEventListener('click', ()=> exportRecording() );
document.getElementById('downloadJson').addEventListener('click', ()=>{ const
a=document.createElement('a'); a.href = URL.createObjectURL(new
Blob([JSON.stringify({clips, sfxFiles:
sfxFiles.map(s=>({name:s.name}))},null,2)],{type:'application/json'}));
a.download='slideshow.json'; a.click(); });

populateVoices();
renderClips();
</script>
</body>
</html>
```