HELIKON VOICE v6.7
SYSBOOT
API...
VOICE...
PROVIDER...
MODEL...
CREDITS...
TTSOFF

?Helikon anrufen
??Pause
0:00
Text→Sprache Sprache→Sprache Sprache→Text Wiki-Suche
Stimme:Stimme 1Stimme 2
● Aufnehmen 0s

```
(function() {

    /* ============================================================
     * HELIKON VOICE v6.7 — 2026-06-19
     * Provider-Mapping slot-basiert ('1'/'2' aus buddy.voice.1/.2).
     * Queue-Fix: Voice-ID pro Item (qItem.voiceId), kein Klobber.
     * splitIntoChunks: Satzweise Aufteilung (max 2500 chars).
     * data-tts-queued: Observer-Doppelfire-Schutz.
     * ============================================================ */

    var PF = '"Press Start 2P","Courier New",Courier,monospace';
    var CRISP = '-webkit-font-smoothing:none;
-moz-osx-font-smoothing:unset; font-smooth:never;
text-rendering:optimizeSpeed;';

    var css = ''

    + '#helikonTTS { margin:0; font-family:' + PF + '; ' + CRISP + '
border:2px solid #ccc; border-radius:8px; background:#f8f8f8;
padding:14px; position:relative; overflow:hidden; }'

    + '#ttsDiag { background:#f0f0f0; border:2px solid #ddd;
border-radius:6px; padding:10px 12px; margin:0 0 10px 0; }'
    + '#diagTitle { font-size:8px; color:#0077aa; letter-spacing:2px;
text-transform:uppercase; margin:0 0 8px 0; padding:0 0 6px 0;
border-bottom:1px solid #ddd; }'

    + '#diagGrid { display:flex; flex-wrap:wrap; gap:0; }'
    + '.diagCell { display:flex; align-items:center; gap:6px; padding:5px
12px 5px 0; min-width:100px; flex:1; }'
    + '.diagLed { display:inline-block; width:8px; height:8px;
background:#ccc; border:1px solid #aaa; flex-shrink:0; transition:all
0.3s ease; }'
    + '.diagKey { font-size:7px; color:#999; letter-spacing:1px;
```

```
text-transform:uppercase; min-width:42px; }'
    + '.diagVal { font-size:7px; color:#666; letter-spacing:0.5px;
text-transform:uppercase; white-space:nowrap; overflow:hidden;
text-overflow:ellipsis; max-width:120px; }'

    + '.diagCell.ok .diagLed { background:#3a8f3a; border-color:#2a6f2a;
box-shadow:0 0 4px rgba(58,143,58,0.4); }'
    + '.diagCell.ok .diagVal { color:#3a8f3a; }'
    + '.diagCell.warn .diagLed { background:#cc8800;
border-color:#aa6600; box-shadow:0 0 4px rgba(204,136,0,0.4); }'
    + '.diagCell.warn .diagVal { color:#cc8800; }'
    + '.diagCell.err .diagLed { background:#cc3333; border-color:#aa2222;
box-shadow:0 0 6px rgba(204,51,51,0.5); animation:ledBlink 1s infinite; }'
    + '.diagCell.err .diagVal { color:#cc3333; }'
    + '.diagCell.off .diagLed { background:#ccc; border-color:#aaa; }'
    + '.diagCell.off .diagVal { color:#999; }'
    + '.diagCell.boot .diagLed { background:#0077aa;
border-color:#005580; animation:ledBoot 0.6s infinite; }'
    + '.diagCell.boot .diagVal { color:#0077aa; }'
    + '@keyframes ledBlink { 0%,100% { opacity:1; } 50% { opacity:0.3; }
}'
    + '@keyframes ledBoot { 0%,100% { opacity:0.3; } 50% { opacity:1; } }'

    + '#diagLog { margin:6px 0 0 0; padding:4px 0 0 0; border-top:1px
dashed #ddd; font-size:7px; color:#999; line-height:1.8; max-height:80px;
overflow:hidden; cursor:context-menu; }'
    + '#diagLog .dlog { opacity:0; transition:opacity 0.3s ease; }'
    + '#diagLog .dlog.vis { opacity:1; }'
    + '#diagLog .dlog .dts { color:#cc8800; margin-right:4px; }'
    + '#diagLog .dlog.derr { color:#cc3333; }'

    + '#ttsWaveWrap { height:40px; margin:0 0 10px 0; overflow:hidden;
border-radius:4px; background:#eee; border:1px solid #ddd; }'
    + '#ttsWave { display:flex; align-items:flex-end;
justify-content:center; gap:3px; height:100%; padding:4px 8px; }'
    + '.ttsBar { display:inline-block; width:7px; background:#0077aa;
border-radius:2px 2px 0 0; height:4px; opacity:0.2; transition:height
0.15s ease, opacity 0.15s ease; }'
    + '#helikonTTS.tts-on .ttsBar { opacity:0.5; }'
    + '@keyframes waveIdle { 0%,100% { height:4px; opacity:0.3; } 50% {
height:12px; opacity:0.6; } }'
    + '#helikonTTS.tts-on .ttsBar { animation:waveIdle 1.5s ease-in-out
infinite; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(1) { animation-delay:0s; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(2) { animation-delay:0.1s; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(3) { animation-delay:0.2s; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(4) { animation-delay:0.3s; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(5) { animation-delay:0.4s; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(6) { animation-delay:0.5s; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(7) { animation-delay:0.6s; }'
```

```
    + '#helikonTTS.tts-on .ttsBar:nth-child(8) { animation-delay:0.7s; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(9) { animation-delay:0.6s; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(10) { animation-delay:0.5s; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(11) { animation-delay:0.4s; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(12) { animation-delay:0.3s; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(13) { animation-delay:0.2s; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(14) { animation-delay:0.1s; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(15) { animation-delay:0s; }'
    + '#helikonTTS.tts-on .ttsBar:nth-child(16) { animation-delay:0.1s; }'

    + '@keyframes waveTalk { 0% { height:4px; } 15% { height:28px; } 30%
{ height:8px; } 45% { height:22px; } 60% { height:6px; } 75% {
height:30px; } 90% { height:12px; } 100% { height:4px; } }'
    + '#helikonTTS.tts-speaking .ttsBar { opacity:1; animation:waveTalk
0.8s ease-in-out infinite; background:#0077aa; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(1) {
animation-delay:0.05s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(2) {
animation-delay:0.1s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(3) {
animation-delay:0.15s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(4) {
animation-delay:0.2s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(5) {
animation-delay:0.25s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(6) {
animation-delay:0.3s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(7) {
animation-delay:0.35s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(8) {
animation-delay:0.4s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(9) {
animation-delay:0.35s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(10) {
animation-delay:0.3s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(11) {
animation-delay:0.25s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(12) {
animation-delay:0.2s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(13) {
animation-delay:0.15s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(14) {
animation-delay:0.1s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(15) {
animation-delay:0.05s; }'
    + '#helikonTTS.tts-speaking .ttsBar:nth-child(16) {
animation-delay:0s; }'

    + '#helikonTTS.tts-call .ttsBar { opacity:0.5; animation:waveIdle
1.2s ease-in-out infinite; background:#3a8f3a; }'
```

```
    + '#helikonTTS.tts-call.tts-speaking .ttsBar { opacity:1;
animation:waveTalk 0.8s ease-in-out infinite; background:#3a8f3a; }'

    + '#callHero { border:2px solid #ddd; border-radius:8px;
background:#fff; padding:14px; margin:0 0 10px 0; display:flex;
align-items:center; gap:10px; flex-wrap:wrap; transition:all 0.3s ease; }'
    + '#callHeroBtn { display:flex; align-items:center; gap:10px;
padding:14px 22px; border:3px solid #3a8f3a; border-radius:8px;
background:#f0fff0; color:#2a6f2a; cursor:pointer; font-family:' + PF +
'; font-size:12px; transition:all 0.25s ease; flex:1; min-width:200px;
justify-content:center; }'
    + '#callHeroBtn:hover { background:#e0ffe0; border-color:#2a6f2a;
transform:translateY(-1px); box-shadow:0 4px 12px rgba(58,143,58,0.2); }'
    + '#callIcon { font-size:20px; display:inline-block;
transition:transform 0.3s ease; }'
    + '#callBtnLabel { letter-spacing:0.5px; }'
    + '@keyframes phoneRing { 0% { transform:rotate(0deg); } 5% {
transform:rotate(15deg); } 10% { transform:rotate(-15deg); } 15% {
transform:rotate(12deg); } 20% { transform:rotate(-12deg); } 25% {
transform:rotate(0deg); } 100% { transform:rotate(0deg); } }'
    + '@keyframes heroPulse { 0%,100% { box-shadow:0 0 0 0
rgba(58,143,58,0.3); } 50% { box-shadow:0 0 0 10px rgba(58,143,58,0); } }'
    + '#callHero.active { border-color:#3a8f3a; background:#f0fff0; }'
    + '#callHero.active #callHeroBtn { background:#3a8f3a; color:#fff;
border-color:#2a6f2a; animation:heroPulse 2s infinite; }'
    + '#callHero.active #callHeroBtn:hover { background:#2a6f2a; }'
    + '#callHero.active #callIcon { animation:phoneRing 2s ease-in-out
infinite; }'
    + '#callHero.paused { border-color:#cc8800; background:#fffbf0; }'
    + '#callHero.paused #callHeroBtn { background:#3a8f3a; color:#fff;
border-color:#2a6f2a; animation:none; opacity:0.7; }'
    + '#callHero.paused #callIcon { animation:none; }'
    + '#callPauseBtn { display:none; align-items:center; gap:6px;
padding:14px 16px; border:2px solid #cc8800; border-radius:8px;
background:#fffbf0; color:#996600; cursor:pointer; font-family:' + PF +
'; font-size:10px; transition:all 0.25s ease; }'
    + '#callPauseBtn:hover { background:#fff0d0; border-color:#996600; }'
    + '#pauseIcon { font-size:14px; }'
    + '#callHero.active #callPauseBtn { display:flex; }'
    + '#callHero.paused #callPauseBtn { display:flex; background:#cc8800;
color:#fff; border-color:#996600; }'
    + '#callPhaseDisplay { display:none; align-items:center; gap:8px;
width:100%; padding:8px 0 0 0; }'
    + '#callHero.active #callPhaseDisplay { display:flex; }'
    + '#callHero.paused #callPhaseDisplay { display:flex; }'
    + '#callPhaseDot { display:inline-block; width:12px; height:12px;
border-radius:50%; background:#999; transition:all 0.3s ease;
flex-shrink:0; }'
    + '#callPhaseText { font-size:9px; color:#666; flex:1;
letter-spacing:0.5px; }'
```

```
    + '#callTimer { font-size:9px; color:#999; min-width:40px;
text-align:right; }'
    + '@keyframes listenBreathe { 0%,100% { transform:scale(1);
opacity:1; } 50% { transform:scale(1.4); opacity:0.5; } }'
    + '@keyframes procSpin { 0% { opacity:1; } 50% { opacity:0.3; } 100%
{ opacity:1; } }'
    + '@keyframes speakWave { 0%,100% { transform:scale(1); } 30% {
transform:scale(1.3); } 60% { transform:scale(0.8); } }'
    + '#callPhaseDot.listening { background:#4CAF50;
animation:listenBreathe 1.5s ease-in-out infinite; }'
    + '#callPhaseDot.processing { background:#ff9800; animation:procSpin
0.6s ease-in-out infinite; }'
    + '#callPhaseDot.speaking { background:#0077aa; animation:speakWave
0.8s ease-in-out infinite; }'
    + '#callPhaseDot.waiting { background:#ff9800; animation:procSpin 1s
ease-in-out infinite; }'
    + '#callPhaseDot.paused { background:#cc8800; animation:none; }'

    + '#ttsControls { display:flex; align-items:center; gap:6px;
padding:8px 0; flex-wrap:wrap; }'
    + '.voice-btn { background:#fff; border:2px solid #ccc; color:#555;
padding:8px 12px; cursor:pointer; font-family:' + PF + '; font-size:9px;
display:inline-flex; align-items:center; gap:6px; border-radius:4px;
transition:all 0.2s ease; position:relative; }'
    + '.voice-btn:hover { border-color:#0077aa; color:#0077aa; }'
    + '.voice-btn::before { content:""; display:inline-block; width:8px;
height:8px; border:2px solid #ccc; background:#eee; border-radius:0;
transition:all 0.15s ease; }'
    + '.voice-btn.active { background:#e8f4ff; border-color:#0077aa;
color:#0077aa; }'
    + '.voice-btn.active::before { background:#0077aa;
border-color:#005580; box-shadow:0 0 6px rgba(0,119,170,0.5); }'
    + '.voice-btn.disabled { background:#f5f5f5; border-color:#ddd;
color:#bbb; cursor:default; }'
    + '.voice-btn.disabled:hover { border-color:#ddd; color:#bbb; }'
    + '.voice-btn.disabled::before { background:#ddd; border-color:#ccc;
}'
    + '#searchToggle:hover { border-color:#cc8800; color:#cc8800; }'
    + '#searchToggle.active { background:#fff8e8; border-color:#cc8800;
color:#cc8800; }'
    + '#searchToggle.active::before { background:#cc8800;
border-color:#aa6600; box-shadow:0 0 6px rgba(204,136,0,0.5); }'

    + '#ttsVoiceSel { display:flex; align-items:center; gap:6px;
padding:6px 0; border-top:1px solid #eee; }'
    + '#voiceLabel { font-size:7px; color:#999; text-transform:uppercase;
letter-spacing:0.5px; margin-right:2px; }'
    + '.voice-opt { background:#fff; border:2px solid #ddd; color:#888;
padding:6px 10px; cursor:pointer; font-family:' + PF + '; font-size:8px;
border-radius:4px; transition:all 0.2s ease; }'
```

```
    + '.voice-opt:hover { border-color:#0077aa; color:#0077aa; }'
    + '.voice-opt.active { background:#e8f4ff; border-color:#0077aa;
color:#0077aa; }'

    + '#voiceRecBar { display:none; align-items:center; gap:10px;
padding:8px 0; border-top:1px solid #eee; }'
    + '#voiceRecBar.visible { display:flex; }'
    + '#voiceRecBtn { background:#fff; border:2px solid #cc3333;
color:#cc3333; padding:10px 14px; cursor:pointer; font-family:' + PF + ';
font-size:9px; border-radius:4px; transition:all 0.2s ease; }'
    + '#voiceRecBtn:hover { background:#cc3333; color:#fff; }'
    + '#voiceRecBtn.recording { background:#cc3333; color:#fff;
animation:recPulse 1s infinite; box-shadow:0 0 10px rgba(204,51,51,0.4);
}'
    + '@keyframes recPulse { 0%,100% { opacity:1; } 50% { opacity:0.6; }
}'
    + '#voiceRecTime { color:#cc3333; font-size:9px; min-width:30px; }'
    + '#voiceRecStatus { color:#999; font-size:7px; }'

    + '.tts-play-btn { background:#fff; border:1px solid #ddd;
color:#999; padding:3px 8px; cursor:pointer; font-size:8px;
margin-left:8px; font-family:' + PF + '; display:none; border-radius:3px;
transition:all 0.2s ease; }'
    + '.tts-play-btn:hover { border-color:#0077aa; color:#0077aa; }'
    + '.tts-play-btn.playing { border-color:#0077aa; color:#0077aa;
display:inline-block !important; box-shadow:0 0 6px rgba(0,119,170,0.3);
}'
    + '.tts-active .tts-play-btn { display:inline-block; }'

    + '.chatMsg.call-status { background:transparent !important;
border:none !important; text-align:center !important; padding:6px 10px
!important; margin:4px auto !important; max-width:100% !important;
color:#3a8f3a !important; font-size:9px !important; opacity:0.8; }'
    + '.chatMsg.call-status::before { content:none !important; }'
    + '.chatMsg.call-status.phase-listening { color:#4CAF50 !important; }'
    + '.chatMsg.call-status.phase-processing { color:#ff9800 !important;
}'
    + '.chatMsg.call-status.phase-speaking { color:#0077aa !important; }'
    + '.chatMsg.call-status.phase-paused { color:#cc8800 !important; }'
    + '.chatMsg.call-status.phase-end { color:#cc3333 !important; }'

    + '@media (max-width:600px) {'
    + '  #helikonTTS { padding:10px; }'
    + '  #diagGrid { flex-direction:column; gap:0; }'
    + '  .diagCell { min-width:0; padding:3px 0; }'
    + '  #callHero { flex-direction:column; }'
    + '  #callHeroBtn { min-width:0; width:100%; font-size:11px;
padding:16px 14px; }'
    + '  #callPauseBtn { width:100%; justify-content:center;
padding:12px; }'
```

```
    + '  #ttsControls { gap:4px; }'
    + '  .voice-btn { font-size:8px; padding:6px 8px; flex:1;
min-width:0; justify-content:center; }'
    + '  #ttsVoiceSel { flex-wrap:wrap; }'
    + '  .voice-opt { flex:1; text-align:center; }'
    + '  #callPhaseText { font-size:8px; }'
    + '}'

    + 'body.dark-mode #helikonTTS { background:#0a0a1a;
border-color:#1a1a3a; }'
    + 'body.dark-mode #ttsDiag { background:#0d0d1e;
border-color:#1a1a3a; }'
    + 'body.dark-mode #diagTitle { color:#0077aa; border-color:#1a1a3a; }'
    + 'body.dark-mode .diagKey { color:#555; }'
    + 'body.dark-mode .diagVal { color:#888; }'
    + 'body.dark-mode .diagCell.ok .diagVal { color:#5ab85a; }'
    + 'body.dark-mode .diagCell.warn .diagVal { color:#cc8800; }'
    + 'body.dark-mode .diagCell.err .diagVal { color:#ff4444; }'
    + 'body.dark-mode .diagCell.boot .diagVal { color:#00ccff; }'
    + 'body.dark-mode #diagLog { border-color:#1a1a3a; color:#444; }'
    + 'body.dark-mode #ttsWaveWrap { background:#0d0d24;
border-color:#1a1a3a; }'
    + 'body.dark-mode .ttsBar { background:#0077aa; }'
    + 'body.dark-mode #helikonTTS.tts-speaking .ttsBar {
background:#00ccff; }'
    + 'body.dark-mode #helikonTTS.tts-call .ttsBar { background:#66ff66;
}'
    + 'body.dark-mode #callHero { background:#0d0d20;
border-color:#1a2a1a; }'
    + 'body.dark-mode #callHeroBtn { background:#0a1a0a;
border-color:#3a8f3a; color:#5ab85a; }'
    + 'body.dark-mode #callHeroBtn:hover { background:#0f2a0f; }'
    + 'body.dark-mode #callHero.active #callHeroBtn { background:#3a8f3a;
color:#fff; }'
    + 'body.dark-mode #callPauseBtn { background:#1a1500;
border-color:#cc8800; color:#cc8800; }'
    + 'body.dark-mode #callHero.paused #callPauseBtn {
background:#cc8800; color:#000; }'
    + 'body.dark-mode #callPhaseText { color:#888; }'
    + 'body.dark-mode #callTimer { color:#666; }'
    + 'body.dark-mode .voice-btn { background:#111; border-color:#333;
color:#888; }'
    + 'body.dark-mode .voice-btn:hover { border-color:#0077aa;
color:#0077aa; }'
    + 'body.dark-mode .voice-btn::before { background:#222;
border-color:#444; }'
    + 'body.dark-mode .voice-btn.active { background:#0a1a2a;
border-color:#0077aa; color:#0077aa; }'
    + 'body.dark-mode .voice-btn.active::before { background:#0077aa;
border-color:#005580; }'
```

```
    + 'body.dark-mode .voice-btn.disabled { background:#0a0a0a;
border-color:#222; color:#444; }'
    + 'body.dark-mode #searchToggle.active { background:#1a1500;
border-color:#cc8800; color:#cc8800; }'
    + 'body.dark-mode #searchToggle.active::before { background:#cc8800;
border-color:#aa6600; }'
    + 'body.dark-mode #ttsVoiceSel { border-color:#1a1a3a; }'
    + 'body.dark-mode .voice-opt { background:#111; border-color:#333;
color:#666; }'
    + 'body.dark-mode .voice-opt:hover { border-color:#0077aa;
color:#0077aa; }'
    + 'body.dark-mode .voice-opt.active { background:#0a1a2a;
border-color:#00ccff; color:#00ccff; }'
    + 'body.dark-mode #voiceRecBar { border-color:#1a1a3a; }'
    + 'body.dark-mode #voiceRecBtn { background:#111; }'
    + 'body.dark-mode .tts-play-btn { background:#111; border-color:#333;
color:#666; }'
    + 'body.dark-mode .tts-play-btn:hover { border-color:#0077aa;
color:#0077aa; }'
    + 'body.dark-mode .chatMsg.call-status { opacity:0.7; }';

    var styleEl = document.createElement('style');
    styleEl.type = 'text/css';
    if (styleEl.styleSheet) { styleEl.styleSheet.cssText = css; }
    else { styleEl.appendChild(document.createTextNode(css)); }
    document.getElementsByTagName('head')[0].appendChild(styleEl);

    var fontLink = document.createElement('link');
    fontLink.rel = 'stylesheet';
    fontLink.href = 'https://fonts.googleapis.com/
css2?family=Press+Start+2P';
    document.head.appendChild(fontLink);

    /* === DOM === */
    var ttsWrap      = document.getElementById('helikonTTS');
    var ttsBtn       = document.getElementById('ttsToggle');
    var stsBtn       = document.getElementById('stsToggle');
    var sttBtn       = document.getElementById('sttToggle');
    var searchBtn    = document.getElementById('searchToggle');
    var recBar       = document.getElementById('voiceRecBar');
    var recBtn       = document.getElementById('voiceRecBtn');
    var recTime      = document.getElementById('voiceRecTime');
    var recStatus    = document.getElementById('voiceRecStatus');
    var voiceOpt1    = document.getElementById('voiceOpt1');
    var voiceOpt2    = document.getElementById('voiceOpt2');
    var callHero     = document.getElementById('callHero');
    var callHeroBtn  = document.getElementById('callHeroBtn');
    var callPauseBtn = document.getElementById('callPauseBtn');
    var callPhaseDot = document.getElementById('callPhaseDot');
    var callPhaseText= document.getElementById('callPhaseText');
```

```
    var callTimerEl  = document.getElementById('callTimer');
    var pauseBtnLabel= document.getElementById('pauseBtnLabel');
    var pauseIcon    = document.getElementById('pauseIcon');
    var dSys         = document.getElementById('dSys');
    var dApi         = document.getElementById('dApi');
    var dVoice       = document.getElementById('dVoice');
    var dModel       = document.getElementById('dModel');
    var dProvider    = document.getElementById('dProvider');
    var dCredits     = document.getElementById('dCredits');
    var dTts         = document.getElementById('dTts');
    var dSysVal      = document.getElementById('dSysVal');
    var dApiVal      = document.getElementById('dApiVal');
    var dVoiceVal    = document.getElementById('dVoiceVal');
    var dModelVal    = document.getElementById('dModelVal');
    var dProviderVal = document.getElementById('dProviderVal');
    var dCreditsVal  = document.getElementById('dCreditsVal');
    var dTtsVal      = document.getElementById('dTtsVal');
    var diagLog      = document.getElementById('diagLog');
    if (!ttsBtn) return;

    ttsBtn.className = 'voice-btn';
    stsBtn.className = 'voice-btn disabled';
    sttBtn.className = 'voice-btn';
    searchBtn.className = 'voice-btn active';

    /* === DIAGNOSTICS === */
    function setDiag(cell, valEl, state, text) {
        if (!cell || !valEl) return;
        cell.className = 'diagCell ' + state;
        valEl.textContent = text;
    }

    var diagLogLines = [];
    var diagLogInited = false;
    function logDiag(msg, isError) {
        if (!diagLog) return;
        if (!diagLogInited) { diagLog.innerHTML = ''; diagLogInited =
true; }
        var now = new Date();
        var ts = ('0' + now.getHours()).slice(-2) + ':' + ('0' +
now.getMinutes()).slice(-2) + ':' + ('0' + now.getSeconds()).slice(-2);
        var line = document.createElement('div');
        line.className = isError ? 'dlog derr' : 'dlog';
        var tsSpan = document.createElement('span');
        tsSpan.className = 'dts';
        tsSpan.textContent = ts;
        line.appendChild(tsSpan);
        line.appendChild(document.createTextNode('> ' + msg));
        diagLog.appendChild(line);
        setTimeout(function() { line.className = (isError ? 'dlog derr' :
```

```
'dlog') + ' vis'; }, 30);
        diagLogLines.push(line);
        while (diagLogLines.length > 4) {
            var old = diagLogLines.shift();
            if (old.parentNode) old.parentNode.removeChild(old);
        }
    }

    if (diagLog) {
        diagLog.addEventListener('contextmenu', function(e) {
            e.preventDefault();
            var allLines = diagLog.querySelectorAll('.dlog');
            var text = '';
            for (var li = 0; li < allLines.length; li++) { text +=
allLines[li].textContent + '\n'; }
            if (navigator.clipboard && navigator.clipboard.writeText) {
                navigator.clipboard.writeText(text).then(function() {
logDiag('DiagLog in Zwischenablage kopiert'); });
            } else {
                var ta = document.createElement('textarea');
                ta.value = text; ta.style.position = 'fixed';
ta.style.left = '-9999px';
                document.body.appendChild(ta); ta.select();
                try { document.execCommand('copy'); logDiag('DiagLog
kopiert (Fallback)'); } catch(ex) {}
                document.body.removeChild(ta);
            }
        });
    }

    function updateDiagTts() {
        if (callActive) { setDiag(dTts, dTtsVal, 'ok', 'CALL'); }
        else if (ttsActive) { setDiag(dTts, dTtsVal, 'ok', 'ON'); }
        else if (sttActive) { setDiag(dTts, dTtsVal, 'ok', 'STT'); }
        else { setDiag(dTts, dTtsVal, 'off', 'OFF'); }
    }

    /* === VOICES === */
    var VOICES = new Array(2);
    VOICES[0] = new Object(); VOICES[0].id = ''; VOICES[0].label =
'Stimme 1';
    VOICES[1] = new Object(); VOICES[1].id = 'rKiu7lQ4c5P3az3745s3';
VOICES[1].label = 'Stimme 2';
    var VOICE_ICON = '\u266A ';
    var activeVoiceIdx = 0;
    /* --- firnCore customization v6.7: slot-basiertes Mapping ('1'/'2')
--- */
    var providerVoiceMap = new Object();
    providerVoiceMap['1'] = 1;
    providerVoiceMap['2'] = 0;
```

```
    /* === STATE === */
    var ttsActive = false, stsActive = false, sttActive = false,
callActive = false;
    var callPaused = false, searchActive = true;
    var ttsApiKey = '', voiceId = '', modelId = 'eleven_multilingual_v2';
    var currentAudio = null, isSpeaking = false, isRecording = false;
    var mediaRecorder = null, audioChunks = [], recTimer = null,
recSeconds = 0;
    var creditsOk = true;
    var callStream = null, callAnalyser = null, callAudioCtx = null;
    var callPhase = 'idle';
    var callSeconds = 0, callTimerId = null;
    /* --- firnCore customization v6.7: empfindlicher Threshold, kuerzere
Silence, Max-Wartezeit --- */
    var SILENCE_THRESHOLD = 10, SILENCE_DURATION = 1200;
    var MAX_WAIT_MS = 8000;
    var CALL_TEXT_MAX = 250;

    /* [A] v6.6: TTS Queue */
    var speakQueue = [];
    var isProcessingQueue = false;

    var pageId = 0;
    if (typeof Deki !== 'undefined' && Deki.PageId) { pageId =
Deki.PageId; }

    window.helikonSearchEnabled = true;
    window.helikonCallMode = false;

    setDiag(dSys, dSysVal, 'boot', 'BOOT');
    setDiag(dApi, dApiVal, 'boot', '...');
    setDiag(dVoice, dVoiceVal, 'boot', '...');
    setDiag(dProvider, dProviderVal, 'boot', '...');
    setDiag(dModel, dModelVal, 'boot', '...');
    setDiag(dCredits, dCreditsVal, 'boot', '...');
    setDiag(dTts, dTtsVal, 'off', 'OFF');
    logDiag('HELIKON VOICE v6.7 booting...');

    /* === FETCH ALL PROPERTIES === */
    function fetchAllProperties(callback) {
        if (!pageId) { callback(new Object()); return; }
        var url = '/@api/deki/pages/' + pageId + '/properties';
        var xhr = new XMLHttpRequest();
        xhr.open('GET', url, true);
        xhr.onload = function() {
            var props = new Object();
            if (xhr.status === 200) {
                try {
                    var parser = new DOMParser();
```

```
                    var xml = parser.parseFromString(xhr.responseText,
'text/xml');
                    var entries = xml.getElementsByTagName('property');
                    for (var i = 0; i < entries.length; i++) {
                        var nameEl = entries[i].getAttribute('name');
                        if (!nameEl) continue;
                        var short =
nameEl.replace('urn:custom.mindtouch.com#', '');
                        var contentsEl =
entries[i].getElementsByTagName('contents');
                        if (contentsEl.length > 0) {
                            var href = contentsEl[0].getAttribute('href');
                            if (href) { props['_href_' + short] = href; }
                            var textNode = contentsEl[0].textContent;
                            if (textNode) { props[short] =
textNode.replace(/^\s+|\s+$/g, ''); }
                        }
                    }
                } catch(e) {}
            }
            var hrefKeys = [];
            for (var k in props) { if (k.indexOf('_href_') === 0) {
hrefKeys.push(k.replace('_href_', '')); } }
            if (hrefKeys.length === 0) { callback(props); return; }
            var done = 0;
            for (var h = 0; h < hrefKeys.length; h++) {
                (function(propName) {
                    var xh = new XMLHttpRequest();
                    xh.open('GET', props['_href_' + propName], true);
                    xh.onload = function() {
                        if (xh.status === 200) { props[propName] =
xh.responseText.replace(/^\s+|\s+$/g, ''); }
                        done++; if (done === hrefKeys.length)
callback(props);
                    };
                    xh.onerror = function() { done++; if (done ===
hrefKeys.length) callback(props); };
                    xh.send();
                })(hrefKeys[h]);
            }
        };
        xhr.onerror = function() { callback(new Object()); };
        xhr.send();
    }

    /* === CREDIT STATE === */
    function setCreditState(state) {
        if (state === 'empty') { setDiag(dCredits, dCreditsVal, 'err',
'LEER!'); creditsOk = false; logDiag('CREDITS: Kein Guthaben!', true); }
        else { setDiag(dCredits, dCreditsVal, 'ok', 'OK'); creditsOk =
```

```
true; }
    }

    /* === UI HELPERS === */
    function setRecStatus(msg) { if (recStatus) recStatus.textContent =
msg; }
    function getActiveVoiceId() { return VOICES[activeVoiceIdx].id ||
voiceId; }

    function formatTime(sec) {
        var m = Math.floor(sec / 60);
        var s = sec % 60;
        return m + ':' + (s < 10 ? '0' : '') + s;
    }

    function setCallPhaseUI(phase) {
        callPhase = phase;
        if (callPhaseDot) callPhaseDot.className = phase;
        var labels = new Object();
        labels.idle = '';
        labels.listening = '\u{1F3A4} Sprich jetzt...';
        labels.processing = '\u{1F504} Verarbeite...';
        labels.speaking = '\u{1F50A} Helikon spricht...';
        labels.waiting = '\u{23F3} Helikon denkt nach...';
        labels.paused = '\u{23F8} Pausiert';
        if (callPhaseText) callPhaseText.textContent = labels[phase] ||
'';
    }

    function updateWrapClass() {
        var cls = '';
        if (ttsActive || sttActive) cls = 'tts-on';
        if (isSpeaking) cls = 'tts-speaking';
        if (callActive && !isSpeaking) cls = 'tts-call';
        if (callActive && isSpeaking) cls = 'tts-call tts-speaking';
        ttsWrap.className = cls;
    }

    function updateButtons() {
        ttsBtn.className = 'voice-btn' + (ttsActive ? ' active' : '');
        stsBtn.className = 'voice-btn disabled';
        sttBtn.className = 'voice-btn' + (sttActive ? ' active' : '');
        searchBtn.className = 'voice-btn' + (searchActive ? ' active' :
'');
        window.helikonSearchEnabled = searchActive;
        if (callActive && callPaused) { callHero.className = 'paused'; }
        else if (callActive) { callHero.className = 'active'; }
        else { callHero.className = ''; }
        if (sttActive) { recBar.className = 'visible'; } else {
recBar.className = ''; }
```

```
        var chatLog = document.getElementById('chatLog');
        if (chatLog) {
            if (ttsActive || callActive) { chatLog.className =
chatLog.className.replace(' tts-active', '') + ' tts-active'; }
            else { chatLog.className = chatLog.className.replace('
tts-active', ''); stopAudio(); }
        }
        updateWrapClass();
        updateDiagTts();
    }

    /* === VOICE SELECTOR === */
    function selectVoice(idx) {
        activeVoiceIdx = idx;
        voiceOpt1.textContent = VOICE_ICON + VOICES[0].label;
        voiceOpt2.textContent = VOICE_ICON + VOICES[1].label;
        voiceOpt1.className = 'voice-opt' + (idx === 0 ? ' active' : '');
        voiceOpt2.className = 'voice-opt' + (idx === 1 ? ' active' : '');
        logDiag('Voice: ' + VOICES[idx].label);
    }
    voiceOpt1.onclick = function() { selectVoice(0); };
    voiceOpt2.onclick = function() { selectVoice(1); };

    /* === CALL TIMER === */
    function startCallTimer() {
        callSeconds = 0; callTimerEl.textContent = '0:00';
        callTimerId = setInterval(function() { callSeconds++;
callTimerEl.textContent = formatTime(callSeconds); }, 1000);
    }
    function stopCallTimer() { if (callTimerId) {
clearInterval(callTimerId); callTimerId = null; } }

    /* === CHAT MESSAGES === */
    function addChatMsg(text, type) {
        var chatLog = document.getElementById('chatLog');
        if (!chatLog) return;
        var div = document.createElement('div');
        div.className = 'chatMsg ' + type;
        div.textContent = text;
        chatLog.appendChild(div);
        chatLog.scrollTop = chatLog.scrollHeight;
    }

    function addCallStatus(text, phase) {
        var chatLog = document.getElementById('chatLog');
        if (!chatLog) return;
        var old = chatLog.querySelectorAll('.chatMsg.call-status');
        for (var o = 0; o < old.length; o++) {
            var cls = old[o].className || '';
            if (cls.indexOf('phase-end') === -1) {
```

```
old[o].parentNode.removeChild(old[o]); }
        }
        var div = document.createElement('div');
        div.className = 'chatMsg call-status phase-' + phase;
        div.textContent = text;
        chatLog.appendChild(div);
        chatLog.scrollTop = chatLog.scrollHeight;
    }

    /* === TEXT HELPERS === */
    function cleanText(text) {
        var LT = String.fromCharCode(60);
        var GT = String.fromCharCode(62);
        var LBRACE = String.fromCharCode(123);
        var RBRACE = String.fromCharCode(125);
        var htmlTagRe = new RegExp(LT + '[^' + GT + ']*' + GT, 'g');
        var headingRe = new RegExp('#' + LBRACE + '1,6' + RBRACE + '\\s',
'g');
        var linkRe = new RegExp('\\[([^\\]]+)\\]\\([^)]+\\)', 'g');
        var clean = text.replace(htmlTagRe,
'').replace(/\*\*([^*]+)\*\*/g, '$1').replace(/\*([^*]+)\*/g,
'$1').replace(/`([^`]+)`/g, '$1').replace(headingRe, '').replace(linkRe,
'$1').replace(/\s+/g, ' ').replace(/^\s+|\s+$/g, '');
        return clean;
    }

    /* [B] v6.6: splitIntoChunks — satzweise Aufteilung, max 2500 chars */
    function splitIntoChunks(text, maxLen) {
        if (text.length <= maxLen) { return [text]; }
        var chunks = [];
        var remaining = text;
        while (remaining.length > 0) {
            if (remaining.length <= maxLen) { chunks.push(remaining);
break; }
            var cut = remaining.substring(0, maxLen);
            var lastDot  = cut.lastIndexOf('. ');
            var lastExcl = cut.lastIndexOf('! ');
            var lastQ    = cut.lastIndexOf('? ');
            var lastNl   = cut.lastIndexOf('\n');
            var best = Math.max(lastDot, lastExcl, lastQ, lastNl);
            if (best > maxLen * 0.3) {
                chunks.push(remaining.substring(0, best + 1));
                remaining = remaining.substring(best + 1).replace(/^\s+/,
'');
            } else {
                var lastSpace = cut.lastIndexOf(' ');
                if (lastSpace > maxLen * 0.5) {
                    chunks.push(remaining.substring(0, lastSpace));
                    remaining = remaining.substring(lastSpace + 1);
                } else {
```

```
                    chunks.push(cut);
                    remaining = remaining.substring(maxLen);
                }
            }
        }
        return chunks;
    }

    /* [C] v6.7: drainQueue — reicht item.voiceId an speak() durch */
    function drainQueue() {
        if (speakQueue.length === 0) {
            isProcessingQueue = false;
            updateWrapClass();
            return;
        }
        isProcessingQueue = true;
        var item = speakQueue.shift();
        speak(item.text, function() {
            if (item.onDone) { item.onDone(); }
            drainQueue();
        }, item.voiceId);
    }

    /* [D] v6.7: speakChunked — Voice-ID pro Queue-Item aufgeloest,
       kein globaler activeVoiceIdx-Klobber mehr (Dual-Mode korrekt). */
    function speakChunked(text, provider, onDone) {
        if (!ttsActive || !ttsApiKey) { if (onDone) { onDone(); } return;
}
        var useIdx = activeVoiceIdx;
        if (provider && providerVoiceMap[provider] !== undefined) {
            useIdx = providerVoiceMap[provider];
            logDiag('TTS: ' + provider + ' \u2192 ' +
VOICES[useIdx].label);
        }
        var resolvedVoiceId = VOICES[useIdx].id || voiceId;
        var chunks = splitIntoChunks(text, 2500);
        var total = chunks.length;
        for (var c = 0; c < total; c++) {
            var qItem = new Object();
            qItem.text = chunks[c];
            qItem.voiceId = resolvedVoiceId;
            qItem.onDone = (c === total - 1) ? onDone : null;
            speakQueue.push(qItem);
        }
        if (!isProcessingQueue) { drainQueue(); }
    }

    function truncateForCall(text) {
        if (!callActive || text.length <= CALL_TEXT_MAX) return text;
        var cut = text.substring(0, CALL_TEXT_MAX);
```

```
        var lastDot = cut.lastIndexOf('.');
        var lastExcl = cut.lastIndexOf('!');
        var lastQ = cut.lastIndexOf('?');
        var best = Math.max(lastDot, lastExcl, lastQ);
        if (best > CALL_TEXT_MAX * 0.3) return cut.substring(0, best + 1);
        var lastSpace = cut.lastIndexOf(' ');
        if (lastSpace > CALL_TEXT_MAX * 0.5) return cut.substring(0,
lastSpace) + '...';
        return cut + '...';
    }

    /* === TTS (speak) — v6.7: optionale fixedVoiceId fuer Queue-Pfad ===
*/
    function speak(text, onDone, fixedVoiceId) {
        var activeId = fixedVoiceId || getActiveVoiceId();
        if (!ttsApiKey || !activeId) { logDiag('TTS: nicht konfiguriert',
true); if (onDone) onDone(); return; }
        if (!creditsOk) { logDiag('TTS: Keine Credits!', true); if
(onDone) onDone(); return; }
        var clean = cleanText(text);
        if (!clean) { if (onDone) onDone(); return; }
        clean = truncateForCall(clean);
        if (clean.length > 5000) { clean = clean.substring(0, 5000); }
        isSpeaking = true; updateWrapClass();
        if (callActive) { setCallPhaseUI('speaking');
addCallStatus('\u{1F50A} Helikon spricht...', 'speaking'); }
        logDiag('TTS: spreche (' + clean.length + ' chars)');
        var url = 'https://api.elevenlabs.io/v1/text-to-speech/' +
encodeURIComponent(activeId) + '?output_format=mp3_44100_128';
        var bodyObj = new Object(); bodyObj.text = clean;
bodyObj.model_id = modelId;
        var xhr = new XMLHttpRequest();
        xhr.open('POST', url, true); xhr.responseType = 'blob';
        xhr.setRequestHeader('Content-Type', 'application/json');
        xhr.setRequestHeader('xi-api-key', ttsApiKey);
        xhr.onload = function() {
            if (xhr.status === 200) { playBlob(xhr.response, onDone);
setCreditState('ok'); logDiag('TTS: OK'); }
            else if (xhr.status === 401 || xhr.status === 403) {
isSpeaking = false; updateWrapClass(); setCreditState('empty');
logDiag('TTS: 401/403 \u2014 Credits leer?', true); if (onDone) onDone();
}
            else { isSpeaking = false; updateWrapClass(); logDiag('TTS:
Fehler ' + xhr.status, true); setDiag(dTts, dTtsVal, 'warn', 'ERR ' +
xhr.status); if (onDone) onDone(); }
        };
        xhr.onerror = function() { isSpeaking = false; updateWrapClass();
logDiag('TTS: Netzwerkfehler', true); setDiag(dTts, dTtsVal, 'err',
'OFFLINE'); if (onDone) onDone(); };
        xhr.send(JSON.stringify(bodyObj));
```

```
    }

    /* === TTS WITH PROVIDER VOICE (nur fuer Call Mode) === */
    function speakAsProvider(text, provider, onDone) {
        var savedIdx = activeVoiceIdx;
        if (provider && providerVoiceMap[provider] !== undefined) {
            activeVoiceIdx = providerVoiceMap[provider];
            logDiag('TTS: ' + provider + ' \u2192 ' +
VOICES[activeVoiceIdx].label);
        }
        speak(text, function() {
            activeVoiceIdx = savedIdx;
            if (onDone) onDone();
        });
    }

    /* [E] v6.6: playBlob — kein stopAudio() mehr intern, Queue bleibt
intakt */
    function playBlob(blob, onDone) {
        var audioUrl = URL.createObjectURL(blob);
        if (currentAudio) { currentAudio.pause();
currentAudio.currentTime = 0; currentAudio = null; }
        currentAudio = new Audio(audioUrl);
        isSpeaking = true; updateWrapClass();
        currentAudio.onended = function() { isSpeaking = false;
updateWrapClass(); URL.revokeObjectURL(audioUrl); currentAudio = null; if
(onDone) onDone(); };
        currentAudio.onerror = function() { isSpeaking = false;
updateWrapClass(); logDiag('Audio: Wiedergabefehler', true);
URL.revokeObjectURL(audioUrl); currentAudio = null; if (onDone) onDone();
};
        currentAudio.play().catch(function() { logDiag('Audio: Autoplay
blockiert', true); isSpeaking = false; updateWrapClass(); if (onDone)
onDone(); });
    }

    /* [E] v6.6: stopAudio — cleart Queue komplett */
    function stopAudio() {
        speakQueue = [];
        isProcessingQueue = false;
        if (currentAudio) { currentAudio.pause();
currentAudio.currentTime = 0; currentAudio = null; }
        isSpeaking = false; updateWrapClass();
    }

    /* === RECORDING === */
    function startRecording(onComplete) {
        if (isRecording) return;
        if (!navigator.mediaDevices ||
!navigator.mediaDevices.getUserMedia) { setRecStatus('Mikrofon nicht
```

```
verfuegbar'); return; }
        audioChunks = []; recSeconds = 0; recTime.textContent = '0s';
        var constraints = new Object(); constraints.audio = true;
constraints.video = false;

navigator.mediaDevices.getUserMedia(constraints).then(function(stream) {
            mediaRecorder = new MediaRecorder(stream); isRecording = true;
            recBtn.className = 'recording'; recBtn.textContent = '\u25A0
Stopp'; setRecStatus('aufnahme...');
            mediaRecorder.ondataavailable = function(e) { if (e.data.size
> 0) { audioChunks.push(e.data); } };
            mediaRecorder.onstop = function() {
                isRecording = false; recBtn.className = '';
recBtn.textContent = '\u25CF Aufnehmen';
                clearInterval(recTimer); recTimer = null;
                var tracks = stream.getTracks(); for (var t = 0; t <
tracks.length; t++) { tracks[t].stop(); }
                if (audioChunks.length > 0) { var bt = new Object();
bt.type = 'audio/webm'; var blob = new Blob(audioChunks, bt); if
(onComplete) onComplete(blob); }
            };
            mediaRecorder.start();
            recTimer = setInterval(function() { recSeconds++;
recTime.textContent = recSeconds + 's'; if (recSeconds >= 30) {
stopRecording(); } }, 1000);
        }).catch(function() { setRecStatus('Mikrofon verweigert'); });
    }
    function stopRecording() { if (mediaRecorder && isRecording) {
mediaRecorder.stop(); } if (recTimer) { clearInterval(recTimer); recTimer
= null; } }

    /* === STT === */
    function transcribeAudio(blob, callback) {
        var formData = new FormData(); formData.append('file', blob,
'recording.webm'); formData.append('model_id', 'scribe_v2');
formData.append('language_code', 'de');
        var xhr = new XMLHttpRequest(); xhr.open('POST',
'https://api.elevenlabs.io/v1/speech-to-text', true);
xhr.setRequestHeader('xi-api-key', ttsApiKey);
        xhr.onload = function() {
            if (xhr.status === 200) { try { var r =
JSON.parse(xhr.responseText); callback(r.text || ''); } catch(e) {
callback(''); } }
            else { setRecStatus('STT-Fehler ' + xhr.status);
callback(''); }
        };
        xhr.onerror = function() { setRecStatus('STT nicht erreichbar');
callback(''); };
        xhr.send(formData);
    }
```

```
    /* === STS (disabled) === */
    function transformVoice(blob) {
        setRecStatus('transformiere...'); var formData = new FormData();
formData.append('audio', blob, 'recording.webm');
formData.append('model_id', 'eleven_english_sts_v2');
        var url = 'https://api.elevenlabs.io/v1/speech-to-speech/' +
encodeURIComponent(getActiveVoiceId()) + '?output_format=mp3_44100_128';
        var xhr = new XMLHttpRequest(); xhr.open('POST', url, true);
xhr.responseType = 'blob'; xhr.setRequestHeader('xi-api-key', ttsApiKey);
        xhr.onload = function() { if (xhr.status === 200) {
playBlob(xhr.response, function() { setRecStatus('bereit'); }); } else {
setRecStatus('STS-Fehler ' + xhr.status); } };
        xhr.onerror = function() { setRecStatus('STS nicht erreichbar');
}; xhr.send(formData);
    }

    /* ============================================================
       CALL MODE
       ============================================================ */
    function callStartListening() {
        if (!callActive || callPaused) return;
        if (!navigator.mediaDevices ||
!navigator.mediaDevices.getUserMedia) { addCallStatus('\u274C Mikrofon
nicht verfuegbar', 'end'); return; }
        setCallPhaseUI('listening'); audioChunks = [];
        addCallStatus('\u{1F3A4} Hoere zu... sprich jetzt', 'listening');
        var constraints = new Object(); constraints.audio = true;
constraints.video = false;

navigator.mediaDevices.getUserMedia(constraints).then(function(stream) {
            callStream = stream;
            callAudioCtx = new (window.AudioContext ||
window.webkitAudioContext)();
            var source = callAudioCtx.createMediaStreamSource(stream);
            callAnalyser = callAudioCtx.createAnalyser();
callAnalyser.fftSize = 512; source.connect(callAnalyser);
            mediaRecorder = new MediaRecorder(stream); isRecording = true;
            mediaRecorder.ondataavailable = function(e) { if (e.data.size
> 0) { audioChunks.push(e.data); } };
            mediaRecorder.onstop = function() { isRecording = false; };
            mediaRecorder.start(250);
            /* --- firnCore customization v6.7: listenStart fuer MAX_WAIT
Fallback --- */
            var silenceStart = null, hasSpoken = false, listenStart =
Date.now();
            function checkVolume() {
                if (!callActive || callPhase !== 'listening' ||
callPaused) return;
                var dataArray = new
```

```
Uint8Array(callAnalyser.frequencyBinCount);
callAnalyser.getByteFrequencyData(dataArray);
                var sum = 0; for (var vi = 0; vi < dataArray.length;
vi++) { sum += dataArray[vi]; } var avg = sum / dataArray.length;
                if (avg > SILENCE_THRESHOLD) { hasSpoken = true;
silenceStart = null; }
                else if (hasSpoken) { if (!silenceStart) { silenceStart =
Date.now(); } else if (Date.now() - silenceStart > SILENCE_DURATION) {
callStopListening(); return; } }
                /* Fallback: wenn nach MAX_WAIT_MS nix gesprochen →
StopListening (leerer Chunk → neu zuhoeren) */
                else if (!hasSpoken && Date.now() - listenStart >
MAX_WAIT_MS) { callStopListening(); return; }
                if (callActive && callPhase === 'listening' &&
!callPaused) { setTimeout(checkVolume, 100); }
            }
            setTimeout(checkVolume, 500);
        }).catch(function() { addCallStatus('\u274C Mikrofon verweigert',
'end'); callEndAll(); });
    }

    function callStopListening() {
        if (mediaRecorder && isRecording) { mediaRecorder.stop(); }
        if (callStream) { var tracks = callStream.getTracks(); for (var t
= 0; t < tracks.length; t++) { tracks[t].stop(); } callStream = null; }
        if (callAudioCtx) { callAudioCtx.close(); callAudioCtx = null; }
        setTimeout(function() {
            if (audioChunks.length === 0) { if (callActive &&
!callPaused) { callStartListening(); } return; }
            setCallPhaseUI('processing');
            addCallStatus('\u{1F504} Verarbeite deine Sprache...',
'processing');
            var bt = new Object(); bt.type = 'audio/webm'; var blob = new
Blob(audioChunks, bt);
            transcribeAudio(blob, function(text) {
                if (!text || !callActive) { if (callActive &&
!callPaused) { callStartListening(); } return; }
                var chatInput = document.getElementById('chatInput'); var
chatSend = document.getElementById('chatSend');
                if (chatInput && chatSend) {
                    chatInput.value = text;
                    setCallPhaseUI('waiting');
                    addCallStatus('\u{23F3} Helikon denkt nach...',
'processing');
                    chatSend.click();
                }
            });
        }, 300);
    }
```

```
    function callReleaseMic() {
        if (mediaRecorder && isRecording) { try { mediaRecorder.stop(); }
catch(e) {} }
        if (callStream) { var tracks = callStream.getTracks(); for (var t
= 0; t < tracks.length; t++) { tracks[t].stop(); } callStream = null; }
        if (callAudioCtx) { try { callAudioCtx.close(); } catch(e) {}
callAudioCtx = null; }
    }

    function callPause() {
        callPaused = true; callReleaseMic();
        setCallPhaseUI('paused');
        addCallStatus('\u{23F8} Anruf pausiert \u2014 Mikrofon aus',
'paused');
        pauseBtnLabel.textContent = 'Weiter';
        pauseIcon.innerHTML = '?';
        logDiag('CALL: Pausiert');
        updateButtons();
    }

    function callResume() {
        callPaused = false;
        pauseBtnLabel.textContent = 'Pause';
        pauseIcon.innerHTML = '??';
        updateButtons();
        addCallStatus('\u{1F3A4} Anruf fortgesetzt', 'listening');
        logDiag('CALL: Fortgesetzt');
        setTimeout(function() { if (callActive && !callPaused &&
!isSpeaking) { callStartListening(); } }, 300);
    }

    function callEndAll() {
        callActive = false; callPaused = false; callPhase = 'idle';
        window.helikonCallMode = false;
        callReleaseMic();
        stopAudio(); stopCallTimer(); updateButtons();
        setCallPhaseUI('idle');
        callTimerEl.textContent = '0:00';
        pauseBtnLabel.textContent = 'Pause';
        pauseIcon.innerHTML = '??';
        logDiag('CALL: Beendet (' + formatTime(callSeconds) + ')');
    }

    /* [G] v6.6: addPlayButton — Queue-Clear + speakChunked */
    function getBuddyText(el) {
        var clone = el.cloneNode(true);
        var btns = clone.querySelectorAll('.tts-play-btn');
        for (var r = 0; r < btns.length; r++) {
btns[r].parentNode.removeChild(btns[r]); }
        var badges = clone.querySelectorAll('.msgProvider');
```

```
        for (var b = 0; b < badges.length; b++) {
badges[b].parentNode.removeChild(badges[b]); }
        return clone.textContent || clone.innerText || '';
    }

    function addPlayButton(msgEl) {
        var playBtn = document.createElement('span'); playBtn.className =
'tts-play-btn';
        playBtn.textContent = '\u25B6 vorlesen'; playBtn.title =
'Nachricht vorlesen';
        playBtn.onclick = function() {
            if (isSpeaking && playBtn.className.indexOf('playing') !==
-1) {
                speakQueue = []; isProcessingQueue = false;
                stopAudio(); playBtn.textContent = '\u25B6 vorlesen';
playBtn.className = 'tts-play-btn'; return;
            }
            speakQueue = []; isProcessingQueue = false;
            stopAudio();
            var allBtns =
document.querySelectorAll('.tts-play-btn.playing');
            for (var b = 0; b < allBtns.length; b++) {
allBtns[b].textContent = '\u25B6 vorlesen'; allBtns[b].className =
'tts-play-btn'; }
            playBtn.textContent = '\u25A0 stopp'; playBtn.className =
'tts-play-btn playing';
            var prov = msgEl.getAttribute('data-provider') || '';
            speakChunked(getBuddyText(msgEl), prov, function() {
playBtn.textContent = '\u25B6 vorlesen'; playBtn.className =
'tts-play-btn'; });
        };
        msgEl.appendChild(playBtn);
    }

    function isBuddyMessage(node) {
        if (!node.className) return false;
        var cls = node.className;
        return (cls.indexOf('chatMsg') !== -1 && cls.indexOf('buddy') !==
-1 && cls.indexOf('typing') === -1);
    }

    /* [F] v6.6: watchChat — data-tts-queued + speakChunked im Autoplay */
    function watchChat() {
        var chatLog = document.getElementById('chatLog');
        if (!chatLog) { setTimeout(watchChat, 500); return; }
        var existing = chatLog.querySelectorAll('.chatMsg.buddy');
        for (var e = 0; e < existing.length; e++) { if
(!existing[e].querySelector('.tts-play-btn')) {
addPlayButton(existing[e]); } }
        if (typeof MutationObserver !== 'undefined') {
```

```
            var observer = new MutationObserver(function(mutations) {
                for (var m = 0; m < mutations.length; m++) {
                    var added = mutations[m].addedNodes;
                    for (var n = 0; n < added.length; n++) {
                        var node = added[n];
                        if (node.nodeType !== 1) continue;
                        /* [F1] Doppel-Trigger-Schutz */
                        if (isBuddyMessage(node) &&
!node.getAttribute('data-tts-queued')) {
                            node.setAttribute('data-tts-queued', '1');
                            (function(el) {
                                setTimeout(function() {
                                    if
(!el.querySelector('.tts-play-btn')) { addPlayButton(el); }
                                    var msgText =
getBuddyText(el).replace(/^\s+|\s+$/g, '');
                                    var msgProvider =
el.getAttribute('data-provider') || '';
                                    /* Call Mode — unveraendert */
                                    if (callActive && (callPhase ===
'waiting' || callPhase === 'processing')) {
                                        if (msgText) {
                                            setCallPhaseUI('speaking');
                                            speakAsProvider(msgText,
msgProvider, function() {
                                                if (callActive &&
!callPaused) {

addCallStatus('\u{1F3A4} Dein Turn...', 'listening');
                                                    setTimeout(function()
{ if (callActive && !callPaused) { callStartListening(); } }, 400);
                                                } else if (callActive &&
callPaused) {

addCallStatus('\u{23F8} Pausiert \u2014 dr\u00FCcke Weiter', 'paused');

setCallPhaseUI('paused');
                                                }
                                            });
                                        }
                                        else { if (callActive &&
!callPaused) { callStartListening(); } }
                                        return;
                                    }
                                    /* [F2] Autoplay — speakChunked statt
speakAsProvider */
                                    if (ttsActive && ttsApiKey &&
getActiveVoiceId()) {
                                        if (msgText) {
speakChunked(msgText, msgProvider); }
```

```
                                    }
                                }, 200);
                            })(node);
                        }
                    }
                }
            });
            var obsConfig = new Object(); obsConfig.childList = true;
obsConfig.subtree = false;
            observer.observe(chatLog, obsConfig);
        }
    }

    /* === GLOBAL KILL SWITCH === */
    function stopAllVoice() {
        stopAudio();
        if (isRecording) stopRecording();
        if (callActive) callEndAll();
        var allBtns = document.querySelectorAll('.tts-play-btn.playing');
        for (var ab = 0; ab < allBtns.length; ab++) {
allBtns[ab].textContent = '\u25B6 vorlesen'; allBtns[ab].className =
'tts-play-btn'; }
        logDiag('STOP ALL: Alles gestoppt');
    }
    window.helikonStopAll = stopAllVoice;
    window.helikonStopAudio = function() { stopAudio(); };

    /* === TOGGLE HANDLERS === */
    ttsBtn.onclick = function() {
        if (!ttsApiKey || !getActiveVoiceId()) { logDiag('TTS: API-Key/
Voice fehlt!', true); return; }
        if (callActive) return;
        ttsActive = !ttsActive; updateButtons();
        if (!ttsActive) {
            stopAudio();
            var allBtns =
document.querySelectorAll('.tts-play-btn.playing');
            for (var ab = 0; ab < allBtns.length; ab++) {
allBtns[ab].textContent = '\u25B6 vorlesen'; allBtns[ab].className =
'tts-play-btn'; }
            logDiag('TTS: deaktiviert \u2014 Audio gestoppt');
        } else {
            logDiag('TTS: aktiviert');
        }
    };
    stsBtn.onclick = function() {
        stopAllVoice();
        ttsActive = false; sttActive = false; stsActive = false;
        updateButtons();
        logDiag('STS: deaktiviert \u2014 Alles gestoppt');
```

```
    };
    sttBtn.onclick = function() {
        if (!ttsApiKey) { logDiag('STT: API-Key fehlt!', true); return; }
        if (callActive) return;
        sttActive = !sttActive; updateButtons();
        if (!sttActive) {
            if (isRecording) stopRecording();
            logDiag('STT: deaktiviert \u2014 Mikrofon aus');
        } else {
            setRecStatus('bereit');
            logDiag('STT: aktiviert');
        }
    };
    searchBtn.onclick = function() {
        searchActive = !searchActive; window.helikonSearchEnabled =
searchActive; updateButtons();
        logDiag('Wiki-Suche: ' + (searchActive ? 'an' : 'aus'));
    };

    callHeroBtn.onclick = function() {
        if (!ttsApiKey || !getActiveVoiceId()) { logDiag('CALL: API-Key/
Voice fehlt!', true); return; }
        if (callActive) {
            callEndAll();
            addCallStatus('\u260E Anruf beendet (' +
formatTime(callSeconds) + ')', 'end');
            addChatMsg('[Anruf beendet \u2014 ' + formatTime(callSeconds)
+ ']', 'system');
            return;
        }
        callActive = true; callPaused = false;
        ttsActive = false; sttActive = false; stsActive = false;
        window.helikonCallMode = true;
        updateButtons(); startCallTimer();
        addChatMsg('[Anruf gestartet \u2014 sprich jetzt...]', 'system');
        addCallStatus('\u260E Verbunden! Sprich jetzt...', 'listening');
        logDiag('CALL: Gestartet');
        callStartListening();
    };

    callPauseBtn.onclick = function() {
        if (!callActive) return;
        if (callPaused) { callResume(); } else { callPause(); }
    };

    recBtn.onclick = function() {
        if (isRecording) { stopRecording(); return; }
        if (sttActive) {
            startRecording(function(blob) {
                setRecStatus('transkribiere...');
```

```
                transcribeAudio(blob, function(text) {
                    if (text) { var ci =
document.getElementById('chatInput'); if (ci) { ci.value = text;
ci.focus(); } addChatMsg('[STT] ' + text, 'system');
setRecStatus(text.substring(0, 40) + (text.length > 40 ? '...' : '')); }
                    else { setRecStatus('kein Text erkannt'); }
                });
            });
        }
    };

    /* ================================================================
       BOOT SEQUENCE
       ================================================================ */
    logDiag('Lade Page Properties...');

    fetchAllProperties(function(props) {
        ttsApiKey = props['buddy.ttsapikey'] || '';
        voiceId   = props['buddy.voice1.id'] || props['buddy.ttsvoiceid']
|| '';
        modelId   = props['buddy.ttsmodel'] || 'eleven_multilingual_v2';
        VOICES[0].id = voiceId;
        VOICES[0].label = props['buddy.voice1.name'] || 'Stimme 1';
        VOICES[1].id = props['buddy.voice2.id'] || 'rKiu7lQ4c5P3az3745s3';
        VOICES[1].label = props['buddy.voice2.name'] || 'Stimme 2';
        voiceOpt1.textContent = VOICE_ICON + VOICES[0].label;
        voiceOpt2.textContent = VOICE_ICON + VOICES[1].label;

        /* --- firnCore customization v6.7: voice.1/voice.2 -> '1'/'2'
--- */
        var pv1 = parseInt(props['buddy.voice.1'] || '2', 10);
        var pv2 = parseInt(props['buddy.voice.2'] || '1', 10);
        providerVoiceMap['1'] = (pv1 >= 1 && pv1 <= 2) ? pv1 - 1 : 1;
        providerVoiceMap['2'] = (pv2 >= 1 && pv2 <= 2) ? pv2 - 1 : 0;
        logDiag('VOICE MAP: 1\u2192' +
VOICES[providerVoiceMap['1']].label + ', 2\u2192' +
VOICES[providerVoiceMap['2']].label);

        setDiag(dSys, dSysVal, 'ok', 'ONLINE');
        logDiag('SYS: Template geladen');

        setTimeout(function() {
            if (ttsApiKey) {
                var keyId = ttsApiKey.substring(0,4) + '..' +
ttsApiKey.substring(ttsApiKey.length - 4);
                setDiag(dApi, dApiVal, 'ok', keyId);
                logDiag('API: Key ' + keyId + ' geladen');
            } else {
                setDiag(dApi, dApiVal, 'err', 'FEHLT');
                logDiag('API: KEIN KEY! buddy.ttsapikey setzen');
```

```
            }
        }, 200);

        setTimeout(function() {
            if (voiceId) {
                setDiag(dVoice, dVoiceVal, 'ok',
VOICES[0].label.toUpperCase());
                logDiag('VOICE: ' + VOICES[0].label + ' (' +
voiceId.substring(0,6) + '..)');
            } else {
                setDiag(dVoice, dVoiceVal, 'err', 'FEHLT');
                logDiag('VOICE: KEINE ID! buddy.voice1.id setzen');
            }
        }, 400);

        setTimeout(function() {
            setDiag(dProvider, dProviderVal, 'ok', 'ELEVENLABS');
            logDiag('PROVIDER: ElevenLabs API v1');
        }, 500);

        setTimeout(function() {
            var shortModel = modelId.replace('eleven_',
'').replace('multilingual_', 'multi_');
            setDiag(dModel, dModelVal, 'ok', shortModel);
            logDiag('MODEL: ' + modelId);
        }, 600);

        setTimeout(function() {
            if (ttsApiKey && voiceId) {
                setDiag(dCredits, dCreditsVal, 'warn', 'UNGETESTET');
                logDiag('CREDITS: Wird bei erstem TTS-Call geprueft');
            } else {
                setDiag(dCredits, dCreditsVal, 'err', 'NICHT PRUEFBAR');
                logDiag('CREDITS: Kann nicht pruefen ohne Key/Voice');
            }
        }, 800);

        setTimeout(function() {
            if (ttsApiKey && voiceId) {
                logDiag('\u2705 System bereit. Queue aktiv (v6.7).');
            } else {
                logDiag('\u26A0 System teilweise bereit. Konfiguration
pruefen.');
            }
        }, 1000);

        if (typeof console !== 'undefined') {
            console.log('=== HELIKON VOICE v6.7 ===');
            console.log('Key: ' + (ttsApiKey ? ttsApiKey.substring(0,8) +
'...' : 'FEHLT'));
```

```
            console.log('Voice 1: ' + VOICES[0].label + ' (' +
(VOICES[0].id ? VOICES[0].id.substring(0,8) + '...' : 'FEHLT') + ')');
            console.log('Voice 2: ' + VOICES[1].label + ' (' +
VOICES[1].id.substring(0,8) + '...)');
            console.log('Model: ' + modelId);
            console.log('Voice Map: 1=' +
VOICES[providerVoiceMap['1']].label + ', 2=' +
VOICES[providerVoiceMap['2']].label);
            console.log('Queue: aktiv (max 2500 chars/Chunk, voiceId pro
Item)');
        }

        window.helikonSearchEnabled = searchActive;
        window.helikonCallMode = false;
        window.helikonSpeak = function(text, provider, onDone) {
speakAsProvider(text, provider, onDone); };
        window.helikonProviderVoice = providerVoiceMap;
        updateButtons(); selectVoice(0); watchChat();
    });

})();
```