# Код примера AI ассистента Eva v2

AI_PRO_Eva_v2.php

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Eva AI Pro Fixed</title>
    <style>
        /* ОСНОВНЫЕ СТИЛИ И ГАБАРИТЫ */
        :root { --bg: #ffffff; --accent: #007bff; --bot-msg: #f1f3f5; --user-msg: #e7f3ff; --text: #212529; --eva-active: #28a745; --warn: #dc3545; }
        body { margin: 0; padding: 0; background: #e9ecef; font-family: 'Segoe UI', sans-serif; display: flex; justify-content: center; align-items: center; height: 100vh; overflow: hidden; }
        .container { width: 600px; height: 350px; background: var(--bg); border-radius: 16px; box-shadow: 0 10px 30px rgba(0,0,0,0.1); display: flex; flex-direction: column; overflow: hidden; border: 2px solid transparent; transition: 0.3s; position: relative; }
        .active-listening { border-color: var(--eva-active); box-shadow: 0 0 20px rgba(40, 167, 69, 0.2); }

        /* ШАПКА */
        .header { background: #fff; padding: 6px 15px; display: flex; justify-content: space-between; align-items: center; border-bottom: 1px solid #eee; height: 35px; }
        #monitor { font-size: 10px; font-weight: 900; color: var(--eva-active); text-transform: uppercase; letter-spacing: 1px; }
        #live-transcript { font-size: 12px; color: #adb5bd; white-space: nowrap; overflow: hidden; text-overflow: ellipsis; max-width: 350px; font-style: italic; }

        /* ОКНО ЧАТА */
        #chat-log { flex: 1; overflow-y: auto; padding: 15px; display: flex; flex-direction: column; gap: 10px; background: #fafafa; scroll-behavior: smooth; }
        .msg { padding: 8px 14px; border-radius: 14px; max-width: 80%; font-size: 13px; line-height: 1.4; color: var(--text); box-shadow: 2px 5px rgba(0,0,0,0.02); }
        .user-msg { background: var(--user-msg); align-self: flex-end; color: #004085; border-bottom-right-radius: 2px; }
        .bot-msg { background: var(--bot-msg); align-self: flex-start; border-bottom-left-radius: 2px; border: 1px solid #dee2e6; }

        /* ФУТЕР */
        .footer { padding: 10px; background: #fff; border-top: 1px solid #eee; }
```

```
        .vol-wrap { width: 100%; height: 4px; background: #eee; border-
radius: 2px; margin-bottom: 10px; overflow: hidden; }
        #vol-bar { width: 0%; height: 100%; background: var(--eva-
active); transition: 0.05s; }
        .controls { display: flex; gap: 8px; align-items: center; }
        button { border: none; border-radius: 10px; cursor: pointer;
font-weight: bold; transition: 0.2s; outline: none; }
        #main-btn { background: var(--accent); color: white; flex-grow:
1; height: 36px; font-size: 11px; text-transform: uppercase; }
        #main-btn.stop { background: var(--warn); }
        .p-btn { background: #f8f9fa; color: #495057; width: 32px;
height: 32px; border: 1px solid #dee2e6; }
        .talk-mode-ui { background: #6f42c1 !important; }
        #learning-tip { position: absolute; top: 40px; left: 50%;
transform: translateX(-50%); background: var(--warn); color: white;
padding: 4px 12px; border-radius: 20px; font-size: 10px; font-weight:
bold; z-index: 10; display: none; }
    </style>
</head>
<body>

<div class="container" id="main-box">
    <div id="learning-tip">НЕ ЗАПОМИНАЙ — ОТМЕНА</div>
    <div class="header">
        <div id="monitor">OFFLINE</div>
        <div id="live-transcript">Готов к запуску</div>
    </div>
    <div id="chat-log"></div>
    <div class="footer">
        <div class="vol-wrap"><div id="vol-bar"></div></div>
        <div class="controls">
            <button id="main-btn">Запустить систему</button>
            <div class="player-btns">
                <button class="p-btn" onclick="music.prev()">□</button>
                <button class="p-btn"
onclick="music.toggle()">□</button>
                <button class="p-btn" onclick="music.next()">□</button>
            </div>
        </div>
    </div>
    <audio id="audio-player"></audio>
</div>

<script>
/** --- ПЕРЕМЕННЫЕ --- **/
const monitor = document.getElementById('monitor'), transcriptUI =
document.getElementById('live-transcript');
const chatLog = document.getElementById('chat-log'), mainBtn =
document.getElementById('main-btn');
const volBar = document.getElementById('vol-bar'), player =
```

```
document.getElementById('audio-player');
const mainBox = document.getElementById('main-box'), tip =
document.getElementById('learning-tip');
const synth = window.speechSynthesis;

let isLive = false, isSpeaking = false, isLearning = false, freeTalk =
false, isEvaActive = false, currentContext = null;
let lastQ = "", responses = [];
const playlist = ["music/Святки.mp3", "music/Зажечь огни.mp3",
"music/Когда я стану старой бабкой.mp3" ];
let trackIdx = 0;

/** --- ПЕРЕЗАГРУЗКА БАЗЫ --- **/
    async function reloadEvaBase() {
        responses = [];
        currentContext = null;
        lastQ = "";
        isLearning = false;

        try {
            // Читаем файл base.json
            const response = await fetch('./base.json?v=' +
Date.now()); // Добавили ?v=время
            const data = await response.json();

            // Если в json есть ключ "responses", берем его, если нет — берем
весь массив
            responses = data.responses || (Array.isArray(data) ? data :
[]);

            console.log("База загружена, ОЗУ чистое");
            if(typeof updateStats === "function") updateStats(); //
Обновить счетчики, если функция есть
        } catch (e) {
            console.error("Файл base.json не найден, создана пустая база");
            responses = [];
        }
    }


/** --- 1. ЗАГРУЗКА ПРИ СТАРТЕ --- **/
async function init() {
    try {
        const res = await fetch('./base.json?v=' + Date.now());
        if (!res.ok) throw new Error("Файл не найден на сервере (404)");

        const text = await res.text();
        console.log("Сырые данные из файла:", text); // СМОТРИТЕ ЭТО В
КОНСОЛИ (F12)

        if (!text || text.trim() === "") {
```

```
            console.warn("Файл base.json пуст");
            responses = [];
        } else {
            const data = JSON.parse(text);

            // ПРОВЕРКА: если это массив — берем как есть,
            // если объект с ключом responses — берем ключ, иначе — пустой
массив
            if (Array.isArray(data)) {
                responses = data;
            } else if (data && typeof data === 'object' &&
data.responses) {
                responses = data.responses;
            } else {
                responses = [];
            }
        }

        console.log("Итоговый массив responses:", responses);
        isBaseLoaded = true; // Важный флаг для сохранения
        logUI("READY");
    } catch (e) {
        console.error("Критическая ошибка при чтении JSON:", e.message);
        responses = [];
        logUI("ERROR: JSON CORRUPT"); // Покажем ошибку в UI, чтобы вы
знали
    }
}



// Вызываем загрузку сразу
init();


/** --- 2. ФУНКЦИИ --- **/
const logUI = (txt) => { monitor.innerText = txt; tip.style.display =
isLearning ? "block" : "none"; };

function normalize(t) {
    if (!t) return "";
    return t.toLowerCase().trim().replace(/[.?!,]/g, '')
            .replace(/\b(я|мне|сейчас|уже|пришел|пришла|ева)\b/g, '')
            .replace(/\s+/g, ' ').trim();
}

function processResponse(text) {
    if (!text) return { audio: "", screen: "" };
    let t = text.trim();
    const stressMap = {"звонит": "звонит", "договор": "договор", "начала":
```

```
"начала", "ева": "Ева"};
    for (let [w, c] of Object.entries(stressMap)) t = t.replace(new
RegExp(w, "gi"), c);
    let screenT = t.replace(/\+/g, '');
    screenT = screenT.charAt(0).toUpperCase() + screenT.slice(1);
    if (!/[.!?]$/.test(screenT)) screenT += ".";
    return { audio: t, screen: screenT };
}

function addMsg(txt, cls) {
    const d = document.createElement('div'); d.className = 'msg ' +
cls; d.innerText = txt;
    chatLog.appendChild(d); chatLog.scrollTop = chatLog.scrollHeight;
}

/** --- 3. ПЛЕЕР --- **/
const music = {
    play: () => {
        if(playlist.length > 0) {
            player.src = playlist[trackIdx]; player.play();
            let name =
playlist[trackIdx].split('/').pop().replace('.mp3','');
            speak("Играет " + name); logUI("MUSIC");
        }
    },
    toggle: () => player.paused ? player.play() : player.pause(),
    next: () => { trackIdx = (trackIdx + 1) % playlist.length;
music.play(); },
    prev: () => { trackIdx = (trackIdx - 1 + playlist.length) %
playlist.length; music.play(); },
    stop: () => { player.pause(); player.currentTime = 0; log("MUSIC
STOPPED"); }
};

/** --- 4. ГОЛОС --- **/
function speak(txt, lang = 'ru-RU', onFinished = null) {
    if (synth.speaking) synth.cancel();
    const processed = processResponse(txt);
    const ut = new SpeechSynthesisUtterance(lang === 'ru-RU' ?
processed.audio : txt);
    ut.lang = lang;
    const voices = synth.getVoices();
    ut.voice = voices.find(v => v.name.includes('Google') &&
v.lang.includes(lang.split('-')[0])) || voices[0];
    ut.rate = 1.0; ut.pitch = 1.1;

    ut.onstart = () => {
        isSpeaking = true;
        logUI("ЕВА ГОВОРИТ");
        try { rec.stop(); } catch(e){}
    };
```

```
    ut.onend = () => {
        isSpeaking = false;
        logUI("ЕВА СЛУШАЕТ");

        // Сначала выполняем callback (встречный вопрос), если он есть
        if (typeof onFinished === "function") {
            onFinished();
        } else {
            // Если callback нет, просто включаем микрофон
            if (isLive) startRec();
        }
    };

    synth.speak(ut);
    return processed.screen;
}

/** --- 5. СЛУХ (ФИКС UNDEFINED И СКОРОСТИ) --- **/
const SpeechRec = window.webkitSpeechRecognition ||
window.SpeechRecognition;
const rec = new SpeechRec();
rec.lang = 'ru-RU'; rec.continuous = true; rec.interimResults = true;

function startRec() { if (isLive && !isSpeaking) try { rec.start(); }
catch(e){} }

rec.onresult = (e) => {
    let interim = '', final = '';
    for (let i = e.resultIndex; i < e.results.length; ++i) {
        if (e.results[i] && e.results[i][0]) { // ФИКС: Тройная проверка
            const txt = e.results[i][0].transcript;
            if (e.results[i].isFinal) final += txt; else interim +=
txt;
        }
    }
    const heard = (final || interim).toLowerCase().trim();
    if (heard) transcriptUI.innerText = heard;

    if (!freeTalk && !isEvaActive && !isLearning && !isSpeaking) {
        if (heard.includes("ева") || heard.includes("еву")) {
            isEvaActive = true; rec.stop(); speak("Ау?"); return;
        }
    }
    if (final.trim() !== "") handle(final.toLowerCase().trim());
};

/** --- 6. ЛОГИКА --- **/
function handle(text) {
    if (!text || isSpeaking) return;
```

```
    // Отмена обучения
    if (isLearning && text.includes("не запоминай")|| text.includes("не
отвечай")|| text.includes("забудь")) {
        isLearning = false; isEvaActive = false; speak("Отменяю.");
return;
    }

    // Запомни вопрос
    if (text.includes("запомни вопрос")) {
        let q = text.replace("запомни вопрос", "").trim();
        if(q) { responses.push({ type: "user_note", content: q });
speak("Записала."); }
        return;
    }

    // Перевод
    if (text.includes("переведи на английский")) {
        let trans = text.replace("переведи на английский", "").trim();
        if(trans) { speak(trans, 'en-US'); addMsg("Eng: " + trans,
'bot-msg'); }
        return;
    }

    // Время и дата
    if (text.includes("скажи время")) { speak(`Сейчас ${new
Date().getHours()} ${new Date().getMinutes()}`); return; }
    if (text.includes("дата")|| text.includes("какое сегодня число")) {
speak(`Сегодня ${new Date().toLocaleDateString('ru-RU')}`); return; }

    // Управление
    if (text.includes("давай поговорим")|| text.includes("даваай
поболтаем")|| text.includes("поболтаем")) { freeTalk = true;
mainBtn.classList.add('talk-mode-ui'); speak("Давай."); return; }
    if (text.includes("стоп разговор")) { freeTalk = false;
mainBtn.classList.remove('talk-mode-ui'); speak("Ок, замолкаю...");
return; }
    if (text.includes("музыку")) { music.play(); return; }
    if (text.includes("стоп") || text.includes("пауза")) {
player.pause(); speak("Пауза"); return; }

            // ОБУЧЕНИЕ
    if (isLearning) {
        // Создаем стандартизированный объект новой записи
        const newEntry = {
            type: "qa",
            questions: [lastQ],
            answers: [text],
            sub: [] // Автоматическое создание пустого массива для веток
        };
```

```
        if (currentContext) {
            // Если мы находимся внутри ветки (в контексте)
            // На всякий случай проверяем существование sub у родителя
            if (!currentContext.sub) currentContext.sub = [];

            currentContext.sub.push(newEntry);
            logUI("ДОБАВЛЕНА ВЕТКА (SUB)");
        } else {
            // Если мы в корневом уровне
            let entry = responses.find(r => r.questions &&
r.questions.includes(lastQ));
            if (entry) {
                // Если вопрос уже есть, просто добавляем новый вариант ответа
                entry.answers.push(text);
                // Проверяем, есть ли у существующей записи поле sub (для
старых баз)
                if (!entry.sub) entry.sub = [];
            } else {
                // Если вопроса нет, добавляем полностью новый объект
                responses.push(newEntry);
            }
            logUI("ЗАПИСАНО В КОРЕНЬ");
        }

        isLearning = false;
        isEvaActive = false;
        currentContext = null; // Сброс контекста после успешного обучения
        speak("Запомнила");
        addMsg(text, 'user-msg');
        if(typeof updateStats === "function") updateStats(); // Вызывать
только если она есть
        return;
    }


    if (!freeTalk && !isEvaActive) return;

// В начале функции обработки текста
const clean = normalize(text.replace(/ева|еву/g, ''));
if (!clean && isEvaActive) return;

// 1. СНАЧАЛА ИЩЕМ, ЕСТЬ ЛИ ТАКОЙ ВОПРОС В БАЗЕ
let found = responses.find(r =>
    r.questions && r.questions.some(q => normalize(q) === clean)
);

// 2. ЕСЛИ МЫ В РЕЖИМЕ ОБУЧЕНИЯ (отвечаем на "Как мне ответить?")
if (isLearning) {
    const newEntry = {
        type: "qa",
```

```
            questions: [lastQ],
            answers: [text],
            sub: []
        };

        if (currentContext) {
            if (!currentContext.sub) currentContext.sub = [];
            currentContext.sub.push(newEntry);
        } else {
            // Если такой вопрос уже был в базе, просто добавим новый вариант ответа
            if (found) {
                found.answers.push(text);
            } else {
                responses.push(newEntry);
            }
        }

        isLearning = false;
        isEvaActive = false;
        currentContext = null;
        speak("Запомнила");
        addMsg(text, 'user-msg');
        updateStats();
        return;
    }

    // 3. ЕСЛИ МЫ ПРОСТО ГОВОРИМ (ПОИСК ОТВЕТА)
    addMsg(text, 'user-msg');
    logUI("ОБРАБОТКА");

    if (found) {
        const randomAnswer = found.answers[Math.floor(Math.random() *
found.answers.length)];
        speak(randomAnswer);
        addMsg(randomAnswer, 'eva-msg');
        currentContext = found;
        isEvaActive = true;
        logUI("ОТВЕТ НАЙДЕН");
    } else {
        // Если ничего не нашли — включаем обучение
        lastQ = clean;
        isLearning = true;
        speak("Как мне отвечать на это?");
        logUI("ОБУЧЕНИЕ...");
    }




        // ПОИСК: Сначала ищем в  sub  текущего контекста, если он есть
        let entry = null;
```

```
    if (currentContext && currentContext.sub) {
        entry = currentContext.sub.find(r => r.questions &&
r.questions.some(q => normalize(q) === clean));
    }

    // Если в ветке не нашли, ищем в основном массиве
    if (!entry) {
        entry = responses.find(r => r.questions && r.questions.some(q
=> normalize(q) === clean || clean.includes(normalize(q))));
    }

        if (entry) {
        const ans = entry.answers[Math.floor(Math.random() *
entry.answers.length)];
        isEvaActive = false;

        // 1. Говорим основной ответ и используем  callback  для встречного вопроса
        const screenText = speak(ans, 'ru-RU', () => {
            // ЭТОТ БЛОК ВЫПОЛНИТСЯ, КОГДА ЕВА ДОГОВОРИТ ОСНОВНОЙ ТЕКСТ

            // Встречный вопрос (30%)
            if (Math.random() > 0.7) {
                let qNodes = responses.filter(r => r.type === "qa");
                if(qNodes.length > 0) {
                    setTimeout(() => {
                        let randEntry = qNodes[Math.floor(Math.random()
* qNodes.length)];

                        // Устанавливаем контекст на встречный вопрос, чтобы
ответ попал в его  sub
                        currentContext = randEntry;

                        // Берем первый вопрос из массива (или саму строку, если
это строка)
                        const nextQ =
Array.isArray(randEntry.questions) ? randEntry.questions[0] :
randEntry.questions;

                        addMsg(nextQ, 'bot-msg');
                        speak(nextQ); // Озвучиваем вопрос (без  callback,
чтобы не зациклить)
                        logUI("ИНИЦИАТИВА ЕВЫ");
                    }, 1000); // Короткая пауза для естественности
                }
            } else {
                // Если вопроса нет, просто возвращаемся в режим прослушивания
                if (isLive) startRec();
            }
        });
```

```
        // Выводим основной текст на экран (обработанный через processResponse внутри speak)
        addMsg(screenText, 'bot-msg');

        // 2. УСТАНОВКА КОНТЕКСТА: Если у этого ответа есть продолжение (sub), фиксируем его
        if (entry.sub && entry.sub.length > 0) {
            currentContext = entry;
            logUI("В КОНТЕКСТЕ: " + entry.questions[0]);
        } else {
            currentContext = null; // Выход из ветки, если продолжения нет
        }

    } else {
        // ... далее ваш старый блок обучения (lastQ = clean; и т.д.)

        lastQ = clean;
        isLearning = true;
        logUI("ОБУЧЕНИЕ");
        speak("Я не знаю ответа. Как мне ответить?");
    }
}

/** --- 7. ПЕРИФЕРИЯ --- **/
async function initMeter() {
    try {
        const stream = await navigator.mediaDevices.getUserMedia({ audio: true });
        const ctx = new AudioContext(), src = ctx.createMediaStreamSource(stream), ans = ctx.createAnalyser();
        src.connect(ans);
        const data = new Uint8Array(ans.frequencyBinCount);
        function update() {
            if (!isLive) return;
            ans.getByteFrequencyData(data);
            volBar.style.width = Math.min(100, (data.reduce((a,b)=>a+b, 0) / data.length) * 5) + "%";
            requestAnimationFrame(update);
        }
        update();
    } catch(e){ logUI("MIC ERROR"); }
}

mainBtn.onclick = () => {
    if (!isLive) {
        isLive = true;
        mainBtn.innerText = "ВЫЙТИ И СОХРАНИТЬ";
        mainBtn.classList.add('stop');
        mainBox.classList.add('active-listening');
        initMeter();
        startRec();
```

```
    } else {
        // ЗАЩИТА: Если база почему-то пуста, не даем сохранить
        if (!responses || responses.length === 0) {
            alert("Внимание! База пуста. Сохранение отменено, чтобы не испортить файл.");
            return;
        }

        const blob = new Blob([JSON.stringify(responses, null, 4)], {type:'application/json'});
        const url = URL.createObjectURL(blob);
        const a = document.createElement('a');
        a.href = url;
        a.download = "base.json"; // Просто имя, без точек и слэшей

        document.body.appendChild(a);
        a.click();

        // Очистка и перезагрузка
        setTimeout(() => {
            URL.revokeObjectURL(url);
            document.body.removeChild(a);
            location.reload();
        }, 1000);
    }
};


rec.onend = () => { if (isLive && !isSpeaking) startRec(); };
window.speechSynthesis.onvoiceschanged = () => synth.getVoices();
    // Вызываем очистку и загрузку сразу при старте
    reloadEvaBase();
</script>
</body>
</html>
```