* Mit "Barrierefreiheit verbessern" ist gemeint, den Chatbot zugänglicher und benutzerfreundlicher für Menschen mit Behinderungen zu machen. Dies umfasst:
* 1. ARIA-Labels: Das sind zusätzliche Attribute in HTML-Elementen, die Screenreadern helfen, die Funktion und den Inhalt von Elementen zu verstehen und vorzulesen.
*  Tastaturnavigation: Sicherstellen, dass alle Funktionen des Chatbots auch ohne Maus, nur mit der Tastatur bedient werden können.
* Weitere Aspekte könnten sein:
*  Ausreichende Farbkontraste für sehbehinderte Nutzer
*  Anpassbare Schriftgrößen
*  Alternativtexte für Bilder
*  Klare und einfache Sprache
* Diese Verbesserungen helfen nicht nur Menschen mit Behinderungen, sondern machen den Chatbot insgesamt benutzerfreundlicher und zugänglicher für alle Nutzer.

<!DOCTYPE html>

<html lang="de">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Kontaktformular mit Kalender und Zeitslots</title>

<script type="text/javascript" src="https://cdn.jsdelivr.net/npm/@emailjs/browser@3/dist/email.min.js"></script>

<script type="text/javascript">

(function() {

emailjs.init("DMCuGfYNM8iEwMkFd");

})();

</script>

/\* Grundlegende Styles \*/

body {

font-family: Arial, sans-serif;

line-height: 1.6;

color: #333;

max-width: 1200px;

margin: 0 auto;

padding: 20px;

background-color: transparent;

}

.container {

display: flex;

flex-wrap: wrap;

gap: 40px;

padding: 0 20px;

}

/\* Form Styles \*/

.form-section {

flex: 1;

min-width: 300px;

}

.calendar-section {

flex: 1;

min-width: 300px;

max-width: 320px;

padding-right: 80px;

}

input, textarea {

width: 100%;

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ddd;

border-radius: 4px;

background-color: rgba(255, 255, 255, 0.9);

}

label {

display: block;

margin-bottom: 5px;

color: white;

text-shadow: 1px 1px 2px rgba(0,0,0,0.7);

}

/\* Kalender Styles \*/

.calendar {

background-color: rgba(255, 255, 255, 0.95);

border-radius: 8px;

padding: 15px;

box-shadow: 0 4px 6px rgba(0,0,0,0.1);

width: 320px;

margin-bottom: 20px;

}

.calendar-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 15px;

}

.calendar-grid {

display: grid;

grid-template-columns: repeat(7, 1fr);

gap: 5px;

padding-top: 5px;

}

.calendar-date {

width: 100%;

aspect-ratio: 1;

display: flex;

align-items: center;

justify-content: center;

border: 1px solid #e9ecef;

background-color: white;

border-radius: 50%;

font-size: 0.9em;

cursor: pointer;

padding: 0;

transition: all 0.2s ease-in-out;

}

/\* Zeitslots Styles \*/

.timeslots-container {

background-color: rgba(255, 255, 255, 0.95);

border-radius: 8px;

padding: 15px;

box-shadow: 0 4px 6px rgba(0,0,0,0.1);

margin-top: 20px;

max-height: 300px;

overflow-y: auto;

}

.timeslots-grid {

display: grid;

grid-template-columns: repeat(3, 1fr);

gap: 10px;

padding: 10px;

}

.timeslot {

background-color: #f8f9fa;

border: 1px solid #dee2e6;

border-radius: 4px;

padding: 8px;

text-align: center;

cursor: pointer;

transition: all 0.2s ease-in-out;

font-size: 0.9em;

}

.timeslot:hover {

background-color: #e9ecef;

transform: scale(1.05);

}

.timeslot.selected {

background-color: #ffebee;

border-color: #ef9a9a;

color: #c62828;

font-weight: bold;

}

/\* Nachrichten Styles \*/

.success-message, .error-message {

display: none;

padding: 10px;

margin-top: 10px;

border-radius: 4px;

text-align: center;

}

.success-message {

background-color: #d4edda;

color: #155724;

}

.error-message {

background-color: #f8d7da;

color: #721c24;

} <link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="container">

<div class="form-section">

<h2>Kontaktieren Sie uns</h2>

<form id="contactForm">

<label for="name">Ihr Name:</label>

<input type="text" id="name" name="name" required>

<label for="email">Ihre E-Mail:</label>

<input type="email" id="email" name="email" required>

<label for="phone">Ihre Telefonnummer:</label>

<input type="tel" id="phone" name="phone" required>

<label for="message">Nachricht:</label>

<textarea id="message" name="message" rows="5" required></textarea>

<button type="submit">Nachricht senden</button>

</form>

<div id="successMessage" class="success-message">

Ihre Nachricht wurde erfolgreich gesendet!

</div>

<div id="errorMessage" class="error-message">

Es gab einen Fehler beim Senden Ihrer Nachricht. Bitte versuchen Sie es später erneut.

</div>

</div>

<div class="calendar-section">

<h2>Termin auswählen</h2>

<div id="calendar" class="calendar"></div>

<div id="selectedDateDisplay" style="margin-top: 10px; color: white; text-shadow: 1px 1px 2px rgba(0,0,0,0.7);"></div>

<div id="timeslots" class="timeslots-container">

<h3>Verfügbare Zeiten</h3>

<div class="timeslots-grid"></div>

</div>

</div>

</div>

// Calendar-Klasse für die Verwaltung des Kalenders und der Zeitslots

class Calendar {

constructor(element) {

this.element = element;

this.currentDate = new Date();

this.selectedDate = null;

this.selectedTime = null;

this.today = new Date();

this.daysOfWeek = ['Mo', 'Di', 'Mi', 'Do', 'Fr', 'Sa', 'So'];

this.render();

this.attachEventListeners();

this.renderTimeSlots();

}

render() {

const year = this.currentDate.getFullYear();

const month = this.currentDate.getMonth();

const firstDay = new Date(year, month, 1);

const lastDay = new Date(year, month + 1, 0);

const daysInMonth = lastDay.getDate();

const startingDay = (firstDay.getDay() + 6) % 7; // Anpassung für Montag als ersten Tag

let calendarHTML = `

<div class="calendar-header">

<button class="prev-month">&lt;</button>

<h3>${this.currentDate.toLocaleString('de-DE', { month: 'long', year: 'numeric' })}</h3>

<button class="next-month">&gt;</button>

</div>

<div class="calendar-grid">

${this.daysOfWeek.map(day => `<div class="calendar-weekday">${day}</div>`).join('')}

`;

// Leere Zellen für Tage vor Monatsbeginn

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

calendarHTML += '<div></div>';

}

// Kalendertage

for (let day = 1; day <= daysInMonth; day++) {

const date = new Date(year, month, day);

const isToday = date.toDateString() === this.today.toDateString();

const isSelected = this.selectedDate && date.toDateString() === this.selectedDate.toDateString();

const isPast = date < this.today;

const isWeekend = date.getDay() === 0 || date.getDay() === 6;

calendarHTML += `

<button class="calendar-date${isToday ? ' today' : ''}${isSelected ? ' selected' : ''}"

data-date="${date.toISOString()}"

${isPast || isWeekend ? 'disabled' : ''}>

${day}

</button>

`;

}

calendarHTML += '</div>';

this.element.innerHTML = calendarHTML;

this.updateSelectedDateDisplay();

}

renderTimeSlots() {

const timeslotsGrid = document.querySelector('.timeslots-grid');

if (!timeslotsGrid) return;

timeslotsGrid.innerHTML = '';

// Zeitslots nur anzeigen, wenn ein Datum ausgewählt ist

if (!this.selectedDate) {

timeslotsGrid.innerHTML = '<p>Bitte wählen Sie zuerst ein Datum aus.</p>';

return;

}

// Generiere Zeitslots von 8:00 bis 18:00 im 30-Minuten-Takt

for (let hour = 8; hour < 18; hour++) {

for (let minutes of ['00', '30']) {

const timeString = `${hour.toString().padStart(2, '0')}:${minutes}`;

const slot = document.createElement('div');

slot.className = 'timeslot';

if (this.selectedTime === timeString) {

slot.classList.add('selected');

}

slot.textContent = timeString;

slot.addEventListener('click', () => this.selectTime(timeString));

timeslotsGrid.appendChild(slot);

}

}

}

selectTime(time) {

this.selectedTime = time;

document.querySelectorAll('.timeslot').forEach(slot => {

slot.classList.remove('selected');

if (slot.textContent === time) {

slot.classList.add('selected');

}

});

this.updateSelectedDateDisplay();

}

attachEventListeners() {

this.element.addEventListener('click', (event) => {

if (event.target.classList.contains('calendar-date') && !event.target.disabled) {

this.selectDate(new Date(event.target.dataset.date));

} else if (event.target.classList.contains('prev-month')) {

this.changeMonth(-1);

} else if (event.target.classList.contains('next-month')) {

this.changeMonth(1);

}

});

}

selectDate(date) {

this.selectedDate = date;

this.selectedTime = null; // Zurücksetzen der ausgewählten Zeit bei neuem Datum

this.render();

this.renderTimeSlots();

this.updateSelectedDateDisplay();

}

changeMonth(delta) {

this.currentDate.setMonth(this.currentDate.getMonth() + delta);

this.render();

}

getSelectedDateTime() {

if (!this.selectedDate) return 'Kein Termin ausgewählt';

const dateStr = this.selectedDate.toLocaleDateString('de-DE');

return this.selectedTime ? `${dateStr} um ${this.selectedTime} Uhr` : dateStr;

}

updateSelectedDateDisplay() {

const displayElement = document.getElementById('selectedDateDisplay');

if (displayElement) {

displayElement.textContent = `Ausgewählter Termin: ${this.getSelectedDateTime()}`;

}

}

}

// Initialisierung und Event Handling

document.addEventListener('DOMContentLoaded', function() {

const calendarElement = document.getElementById('calendar');

let calendar;

if (calendarElement) {

calendar = new Calendar(calendarElement);

} else {

console.error('Calendar element not found');

}

const contactForm = document.getElementById('contactForm');

const successMessage = document.getElementById('successMessage');

const errorMessage = document.getElementById('errorMessage');

contactForm.addEventListener('submit', function(e) {

e.preventDefault();

const templateParams = {

from\_name: document.getElementById('name').value,

from\_email: document.getElementById('email').value,

phone: document.getElementById('phone').value,

message: document.getElementById('message').value,

selected\_datetime: calendar ? calendar.getSelectedDateTime() : 'Kein Termin ausgewählt',

to\_email: 'Info@alphainformatik.ch'

};

emailjs.send('service\_xklufw9', 'template\_8qyruy4', templateParams)

.then(function(response) {

console.log('SUCCESS!', response.status, response.text);

successMessage.style.display = 'block';

errorMessage.style.display = 'none';

contactForm.reset();

if (calendar) {

calendar.selectedDate = null;

calendar.selectedTime = null;

calendar.render();

calendar.renderTimeSlots();

}

setTimeout(() => {

successMessage.style.display = 'none';

}, 5000);

}, function(error) {

console.log('FAILED...', error);

errorMessage.style.display = 'block';

successMessage.style.display = 'none';

setTimeout(() => {

errorMessage.style.display = 'none';

}, 5000);

});

});

}); <script src="calendar.js"></script>

</body>

</html>