CaptureBarriereFrei - Barrierefreie Formulare mit effektivem Bot-Schutz

Übersicht

CaptureBarriereFrei bietet eine innovative, vollständig barrierefreie Alternative zu konventionellen CAPTCHA-Systemen. Die Lösung schützt Webformulare zuverlässig vor automatisierten Bot-Zugriffen und gewährleistet gleichzeitig uneingeschränkte Zugänglichkeit für alle Nutzer. Im Gegensatz zu herkömmlichen CAPTCHAs, die erhebliche Hürden für Menschen mit Behinderungen darstellen können, setzt CaptureBarriereFrei auf eine Kombination nicht-intrusiver, intelligenter Techniken zur zuverlässigen Unterscheidung zwischen menschlichen Nutzern und automatisierten Bots.

Kernfunktionen

- **Barrierefreier Bot-Schutz**: Robuste Sicherheit ohne Einschränkung der Zugänglichkeit
- **Verhaltensbasierte Analyse**: Erkennung menschlicher Interaktionsmuster durch differenzierte Auswertung von Maus-, Tastatur- und Scrollverhalten
- **Intelligente Honeypot-Mechanismen**: Strategisch platzierte, für Menschen unsichtbare Fallen für Bots
- **Präzise Zeitanalyse**: Auswertung realistischer Formular-Ausfüllzeiten
- **Barrierefreie Verifikation**: Semantisch korrekte und mit Screenreadern kompatible Bestätigungselemente
- **Zuverlässiger E-Mail-Versand**: Integrierte Kommunikationsfunktionen mit automatischen Fallback-Mechanismen
- **Modulares, erweiterbares Design**: Flexible Anpassung an unterschiedliche Anforderungen

Installation

Systemvoraussetzungen

- Webserver mit PHP-Unterstützung (für die E-Mail-Funktionalität)
- Moderner Browser mit aktiviertem JavaScript

Einrichtungsschritte

1. Projekt in Ihr Webverzeichnis integrieren:

   ```
   git clone [repository-url] /path/to/webserver/mailTest
   # alternativ: manuelle Dateiübertragung
   ```

2. PHP-Mail-Konfiguration validieren:

   ```
   http://localhost/mailTest/mail_diagnose.php
   ```

3. Konfiguration anpassen (siehe nächster Abschnitt)

Konfiguration

Die Konfiguration erfolgt über die zentrale Datei `assets/js/config.js`. Hier werden alle Einstellungen für beide Module definiert:

```
// Empfängeradresse für das Kontaktformular
const EMPFAENGER_EMAIL = 'ihre.email@domain.de';

// Betreff für die E-Mail
const EMAIL_BETREFF = 'Neue Anfrage über Ihr Kontaktformular';

// Bot-Schutz-Einstellungen
const MIN_AUSFUELLZEIT = 3000;            // Millisekunden
const SCHWELLWERT_PUNKTE = 10;            // Erforderliche Punkte für
gültige Absendung

// Dateiupload-Einstellungen
const MAX_DATEIGROESSE = 5;               // MB
const ERLAUBTE_DATEITYPEN = ['.jpg', '.jpeg', '.png', '.pdf'];

// Debug-Einstellungen
const DEBUGGING = false;                  // Debug-Meldungen in der
Konsole anzeigen
```

Die detaillierten Konfigurationsoptionen werden automatisch in der `erstelleModulKonfigurationen()`-Funktion verarbeitet und an die entsprechenden Module übergeben.

Einbindung

HTML-Grundstruktur

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <title>Kontaktformular</title>
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <!-- Formular mit der Klasse "protected" für automatischen Bot-Schutz
-->
    <form id="kontaktFormular" method="post" class="protected">
        <!-- Formularfelder -->
        <div class="form-group">
            <label for="name">Name</label>
            <input type="text" id="name" name="name" required>
```

```
        </div>

        <div class="form-group">
            <label for="email">E-Mail</label>
            <input type="email" id="email" name="email" required>
        </div>

        <!-- Dateiupload-Unterstützung -->
        <div class="form-group">
            <label for="upload">Datei-Upload</label>
            <input type="file" id="upload" name="upload" accept=".jpg,
.jpeg, .png, .pdf">
            <p class="file-size-info">Maximale Dateigröße: 5 MB</p>
        </div>

        <div class="form-group">
            <label for="message">Nachricht</label>
            <textarea id="message" name="message" required></textarea>
        </div>

        <div class="form-actions">
            <button type="submit">Nachricht senden</button>
        </div>
    </form>

    <!-- Feedback-Bereich für Statusmeldungen -->
    <div id="form-feedback" aria-live="polite"></div>

    <!-- Script-Einbindung -->
    <script src="assets/js/config.js"></script>
    <script type="module" src="assets/js/captureBarriereFrei/index.js">
</script>
    <script src="assets/js/mailSender.js"></script>
    <script>
        // Initialisierung starten, wenn das DOM geladen ist
        document.addEventListener('DOMContentLoaded', function() {
            // Konfiguration einlesen und beide Module initialisieren
            FormularKonfiguration.initialisiere();
        });
    </script>
</body>
</html>
```

Modul 1: CaptureBarriereFrei - Intelligente Bot-Erkennung

Komponenten

- **Core (core.js)**: Zentrales Steuerungsmodul mit Hauptlogik
- **BotDetection (botDetection.js)**: Fortschrittliche Verhaltensanalyse
- **FormProtection (formProtection.js)**: Implementierung mehrschichtiger Schutzstrategien
- **UI (ui.js)**: Barrierefreie Benutzeroberfläche mit ARIA-Integration
- **FormValidation (formValidation.js)**: Robuste, client-seitige Validierungsfunktionen

- **Utils (utils.js)**: Optimierte Hilfsfunktionen für verschiedene Aufgaben
- **Logger (logger.js)**: Konfigurierbares Logging- und Debugging-System

Funktionsprinzipien und Schutzmechanismen

Mehrschichtige Verhaltensanalyse

Das System erfasst und analysiert differenzierte Interaktionsmuster durch:

- **Präzise Bewegungsverfolgung**: Analyse von Mausbewegungen, Klickmustern und -sequenzen
- **Tastaturinteraktionsanalyse**: Erfassung von Eingabegeschwindigkeit, Rhythmus und Mustern
- **Scrollverhaltenserkennung**: Identifikation natürlicher Scrollmuster und -geschwindigkeiten
- **Interaktionsgewichtung**: Algorithmische Auswertung der Balance zwischen verschiedenen Interaktionsformen

Zeitliche Verhaltensanalyse

Die Lösung implementiert eine fortschrittliche temporale Analyse:

- **Zeitfensterüberwachung**: Messung der Gesamtinteraktionszeit mit dem Formular
- **Dynamische Schwellwertanpassung**: Konfigurierbare Mindestinteraktionszeiten je nach Formulartyp
- **Musteranalyse**: Erkennung unnatürlich schneller oder mechanischer Eingabesequenzen

Honeypot-Technologie

Strategisch implementierte Fallen für automatisierte Systeme:

- **Für Menschen unsichtbar**: Vollständige Verbergung durch CSS (position: absolute; left: -9999px)
- **Screenreader-kompatibel**: Korrekte Implementierung von aria-hidden="true"
- **Tab-Navigation-sicher**: Ausschluss aus der Tabreihenfolge (tabindex="-1")

Barrierefreie Verifikation

Eine innovative, zugängliche Alternative zu traditionellen CAPTCHA-Elementen:

- **Vollständige Tastaturbedienbarkeit**: Optimierte Bedienung ohne Maus
- **Semantisch korrekte ARIA-Attribute**: Präzise Spezifikation der Rolle und des Status
- **Multimodale Rückmeldung**: Visuelles, akustisches und strukturelles Feedback

Score-basierte Entscheidungslogik

Ein nuanciertes Punktesystem bewertet Interaktionen und ermöglicht eine präzise Klassifizierung:

- **Kumulative Bewertung**: Zusammenführung verschiedener Interaktionsmetriken
- **Gewichtete Faktoren**: Priorisierung besonders aussagekräftiger Verhaltensmuster
- **Dynamische Schwellenwerte**: Anpassbare Grenzen für unterschiedliche Sicherheitsanforderungen

Beispielanalyse eines menschlichen Nutzers:

- Diverse Mausinteraktionen: +8 Punkte
- Mehrere Tastatureingaben: +12 Punkte
- Natürliches Scrollverhalten: +7 Punkte
- Aktivierte Verifikation: +25 Punkte
- Ausgewogene Interaktion: +5 Punkte
- **Gesamtwert: 57 Punkte** (deutlich über dem Standardschwellwert von 5)

Der Standardschwellwert lässt sich über die `thresholdScore`-Konfiguration anpassen.

Barrierefreiheitskonzepte

Screenreader-Optimierung

- Semantisch strukturierter HTML-Code mit korrekten Elementrollen
- Strategisch platzierte ARIA-Live-Regionen für dynamische Inhalte
- Kontextbezogene, informative Fehlermeldungen

Umfassende Tastaturzugänglichkeit

- Implementierung von Skip-Links für effiziente Navigation
- Durchdachte, logische Tabulator-Sequenz
- Deutliche visuelle Fokusindikatoren für alle interaktiven Elemente

Validierung mit Fokus auf Zugänglichkeit

- Kontextbezogene, feldspezifische Fehlermeldungen
- ARIA-basierte Statuskommunikation
- Klare Handlungsanweisungen zur Fehlerkorrektur

Konfigurationsoptionen

```
const captureConfig = {
    autoProtect: true,                // Automatische Aktivierung für
alle passenden Formulare
    formSelectors: 'form.protected',    // Ziel-Selektoren
    botScoreFieldName: 'security-score', // Bezeichner des
Sicherheitsfeldes
    minTimeToFill: 3000,              // Minimale Ausfüllzeit in ms
    thresholdScore: 10,               // Erforderlicher
Mindestsicherheitswert
    enableLogging: false              // Aktivierung detaillierter
Protokollierung
};
```

Modul 2: MailSender - Zuverlässige E-Mail-Kommunikation

Übersicht

Das MailSender-Modul stellt ein professionelles System zur sicheren Verarbeitung und Übermittlung von Formulardaten per E-Mail bereit. Es ist vollständig mit dem CaptureBarriereFrei-Modul integrierbar und wird über die zentrale Konfigurationsdatei gesteuert.

Architektur

Frontend-Komponenten

- **MailSender (mailSender.js)**: Hauptklasse zur Formularverarbeitung und API-Kommunikation
- **UI-Integration**: Barrierefreie Feedback- und Statusanzeigeelemente

Backend-Komponenten

- **Mail-Prozessor (send_mail.php)**: Server-seitige Datenverarbeitung und E-Mail-Versand
- **Diagnose-Tool (mail_diagnose.php)**: Überprüfungswerkzeug für die Mail-Konfiguration

Funktionaler Ablauf

1. Initialisierung und Konfiguration

Bei der Initialisierung wird das Zielformular mit spezialisierten Event-Listenern ausgestattet:

```
const mailSender = new MailSender({
    recipient: 'empfaenger@beispiel.de',
    formSelector: '#kontaktFormular',
    fallbackToMailto: true
});
```

2. Intelligente Formularverarbeitung

Bei der Formularübermittlung durchläuft das System folgende Phasen:

1. **Event-Interception**: Kontrollierte Übernahme des Submit-Prozesses
2. **Validierung**: Optionale, anpassbare Client-seitige Datenprüfung
3. **Strukturierte Datensammlung**: Aufbereitung der Formularfelder als FormData-Objekt
4. **Sicherheitsintegration**: Nahtlose Einbindung des CaptureBarriereFrei-Sicherheitsscores
5. **Asynchrone Übermittlung**: Effiziente AJAX-Kommunikation mit dem Backend

3. Server-seitige Verarbeitung

Der Backend-Prozessor führt eine mehrstufige Verarbeitung durch:

1. **Eingabevalidierung**: Umfassende Prüfung auf Vollständigkeit und Datenintegrität
2. **Inhaltsgenerierung**: Strukturierte Aufbereitung der E-Mail-Inhalte
3. **Versandprozess**: Flexible Nutzung von PHP mail() oder alternativen Transportmethoden
4. **Strukturierte Antwort**: Generierung standardisierter JSON-Antworten mit Statusinformationen

4. Fehlertoleranz und Ausfallsicherheit

Das Modul implementiert mehrschichtige Fehlerbehandlungsstrategien:

1. **Netzwerkfehler-Erkennung**: Identifikation und Behandlung von HTTP-Fehlern
2. **Transportfehler-Management**: Erkennung von Mail-Server-Problemen
3. **Fallback-Mechanismen**: Optionale Aktivierung von mailto-Links bei Serverfehlern
4. **Nutzerkommunikation**: Transparente Statusmeldungen mit Handlungsempfehlungen

5. Sicherheitsarchitektur

Umfassende Sicherheitsmaßnahmen schützen vor Missbrauch:

1. **Header-Authentifizierung**: Validierung der Anfrage-Herkunft
2. **Eingabesanitisierung**: Umfassende Bereinigung aller Datenfelder
3. **Anti-Spam-Integration**: Nahtlose Verknüpfung mit CaptureBarriereFrei
4. **Transportverschlüsselung**: Optionale HTTPS-Erzwingung

Verfügbare Konfigurationsoptionen

In der `config.js` werden die wichtigsten Einstellungen definiert. Intern werden folgende Parameter unterstützt:

```
const mailConfig = {
    // Grundkonfiguration
    recipient: 'empfaenger@beispiel.de',   // Zieladresse
    subject: 'Neue Formularanfrage',       // E-Mail-Betreff
    formSelector: '#kontaktFormular',      // Formular-Identifikation

    // Übermittlungskonfiguration
    endpoint: 'send_mail.php',             // Backend-Endpunkt
    method: 'POST',                        // HTTP-Methode
    fallbackToMailto: true,                // Aktivierung des Fallback-
Mechanismus

    // Sicherheitskonfiguration
    requireSecurityField: true,            // Integration mit
CaptureBarriereFrei
    securityFieldName: 'security-score',   // Bezeichner des
Sicherheitsfeldes

    // Benutzererfahrung
    successMessage: 'Vielen Dank! Ihre Nachricht wurde erfolgreich
übermittelt.',
    errorMessage: 'Bei der Übermittlung ist ein Fehler aufgetreten. Bitte
versuchen Sie es später erneut.',
    mailtoErrorMessage: 'Die direkte Übermittlung ist fehlgeschlagen. Ihr
E-Mail-Programm wird geöffnet...',
    resetFormAfterSubmit: true,            // Formularrücksetzung nach
erfolgreicher Übermittlung

    // Systemkonfiguration
    debug: false,                          // Aktivierung detaillierter
```

```
Protokollierung
    preserveFormHandlers: true,           // Erhaltung vorhandener Event-
Handler
    scrollToFeedback: true,               // Automatisches Scrollen zum
Feedback-Element
    feedbackElement: '#form-feedback',    // Selektor für das Feedback-
Element
    feedbackDuration: 5000                // Anzeigedauer von
Statusmeldungen (0 = permanent)
};
```

Diese Konfiguration wird automatisch aus den Grundeinstellungen in `config.js` erzeugt. Bei speziellen Anforderungen kann die Konfiguration erweitert werden.

Erweiterungsmöglichkeiten

Alternative Transportmechanismen

Das System unterstützt die Integration spezialisierter Mail-Transport-Bibliotheken:

```php
// Integration von PHPMailer für SMTP-Versand
function sendMailSMTP($to, $subject, $message, $headers) {
    $mail = new PHPMailer(true);
    $mail->isSMTP();
    $mail->Host = 'smtp.example.com';
    $mail->SMTPAuth = true;
    $mail->Username = 'user@example.com';
    $mail->Password = 'password';
    $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
    $mail->Port = 587;

    $mail->setFrom('from@example.com', 'Formular-System');
    $mail->addAddress($to);
    $mail->Subject = $subject;
    $mail->Body = $message;

    return $mail->send();
}
```

Erweiterte Validierungsregeln

Das MailSender-Modul unterstützt benutzerdefinierte Validierungslogik:

```javascript
mailSender.addCustomValidator('field-name', function(value) {
    // Spezialisierte Validierungslogik
    return value.length >= 5 && /^[a-z0-9]+$/i.test(value);
}, 'Bitte geben Sie mindestens 5 alphanumerische Zeichen ein.');
```

Template-basierte E-Mail-Generierung

Für komplexere Anwendungsfälle steht eine Template-Engine zur Verfügung:

```
// Flexibles Template-System
function loadEmailTemplate($templateName, $variables) {
    $template = file_get_contents("templates/{$templateName}.html");
    foreach ($variables as $key => $value) {
        $template = str_replace("{{" . $key . "}}",
htmlspecialchars($value), $template);
    }
    return $template;
}
```

Integration beider Module

Implementierungsbeispiel

```
<!-- Einbindung der Stylesheets -->
<link rel="stylesheet" href="assets/css/style.css" />

<!-- Formular mit Schutzkennzeichnung -->
<form id="kontaktFormular" method="post" class="protected">
    <!-- Strukturierte Formularsektionen -->
    <div class="form-group">
        <label for="name">Name</label>
        <input type="text" id="name" name="name" required>
    </div>
    <div class="form-group">
        <label for="email">E-Mail</label>
        <input type="email" id="email" name="email" required>
    </div>
    <div class="form-group">
        <label for="message">Nachricht</label>
        <textarea id="message" name="message" required></textarea>
    </div>
    <div class="form-actions">
        <button type="submit">Nachricht senden</button>
    </div>
</form>

    <!--    Skripte für die Funktionalität
            Folgende Skripte sind erforderlich:
            - captureBarriereFrei.js: Hauptskript für die
Formularverarbeitung
            - mailSender.js: Skript für den E-Mail-Versand
            - config.js: Konfiguration und Initialisierung

            Die Skripte sind in der Reihenfolge geladen, in der sie
benötigt werden. wichtig sie müssen als module geladen werden,
```

```
            da sie ES6-Module sind.
    -->
    <!-- Module-Scripts -->
    <script type="module" src="assets/js/captureBarriereFrei/index.js">
</script>
    <script type="module" src="assets/js/mailSender/index.js"></script>

    <!-- Konfiguration und Initialisierung -->
    <script src="assets/js/config.js"></script>
    <script>
        // Einfache Initialisierung, wenn die Seite geladen ist
        document.addEventListener('DOMContentLoaded', function() {
            // Formular mit den vordefinierten Einstellungen
initialisieren
            window.FormularKonfiguration.initialisiere();
        });c
    </script>
```

Prozessablauf der integrierten Lösung

1. Systeminitialisierung

- CaptureBarriereFrei etabliert Schutzmaßnahmen und Interaktions-Tracking
- MailSender übernimmt die Formularsteuerung und bereitet Kommunikationswege vor

2. Nutzerinteraktionsphase

- CaptureBarriereFrei analysiert kontinuierlich Interaktionsmuster und aktualisiert den Sicherheitsscore
- Benutzer interagiert mit dem Formular und vervollständigt seine Eingaben

3. Übermittlungsphase

- MailSender übernimmt die Kontrolle beim Formularversand
- CaptureBarriereFrei finalisiert den Sicherheitsscore basierend auf der Gesamtinteraktion
- MailSender integriert den Score in die zu übermittelnden Daten

4. Kommunikationsphase

- MailSender überträgt die Daten inkl. Sicherheitsinformationen an den Server
- Backend-Prozessor validiert den Sicherheitsscore und die Formulardaten
- Bei positivem Ergebnis erfolgt die E-Mail-Generierung und der Versand

5. Abschlussphase

- MailSender verarbeitet die Server-Antwort und aktualisiert die Benutzeroberfläche
- Bei erfolgreicher Übermittlung wird eine Bestätigung angezeigt
- Bei Fehlern werden entsprechende Maßnahmen (z.B. Fallback) eingeleitet

Anpassung und Erweiterung

Zentrale Konfigurationsanpassung

Die sauberste Methode zur Anpassung ist die Bearbeitung der `config.js`. Für fortgeschrittene Anpassungen können Sie die folgenden Ansätze verwenden:

Visuelle Anpassungen

Die Darstellung lässt sich über CSS individualisieren:

- `assets/css/style.css` - Zentrale Styling-Datei
- UI-Komponenten - Individuelle Anpassungen der Interaktionselemente

Funktionale Erweiterungen

Die Erweiterung der Funktionalität sollte vorzugsweise über eigene JavaScript-Dateien erfolgen, die nach der Grundinitialisierung geladen werden:

```
// In einer eigenen Datei, z.B. custom-extensions.js
document.addEventListener('DOMContentLoaded', function() {
    // Warten auf Abschluss der Grundinitialisierung
    setTimeout(function() {
        // Nach der Initialisierung durch FormularKonfiguration
        // haben wir Zugriff auf die globalen Instanzen

        // Erweiterung der Validierung
        const originalValidateField =
window.captureBarriereFreiInstance.validateField;
        window.captureBarriereFreiInstance.validateField = function(input,
errorElement) {
            // Basisvalidierung
            const isValid = originalValidateField.call(this, input,
errorElement);

            // Eigene Validierung
            if (isValid && input.name === 'iban') {
                return this.validateIBAN(input.value, errorElement);
            }

            return isValid;
        };

        // Implementierung eigener Validierungsmethoden
        window.captureBarriereFreiInstance.validateIBAN = function(value,
errorElement) {
            const isValid = /^[A-Z]{2}\d{2}[A-Z0-9]{12,30}$/.test(value);
            if (!isValid) {
                this.showError(errorElement, 'Bitte geben Sie eine gültige
IBAN ein.');
            }
            return isValid;
        };
    }, 500); // Kurze Verzögerung, um sicherzustellen, dass die
Initialisierung abgeschlossen ist
});
```

Problemlösungen

Übermittlungsprobleme

Bei Schwierigkeiten mit der Datenübermittlung:

- Überprüfen Sie die Browser-Konsole auf JavaScript-Fehler
- Führen Sie eine Diagnose der Mail-Funktionalität mit mail_diagnose.php durch
- Aktivieren Sie den Debug-Modus beider Module für detaillierte Protokolle

Mail-Übermittlungsprobleme

Bei Problemen mit dem E-Mail-Versand:

- Validieren Sie die PHP mail()-Konfiguration Ihres Servers
- Erwägen Sie alternative Transportmethoden wie SMTP mit PHPMailer
- Überprüfen Sie Firewall- und Portkonfigurationen für ausgehende Mails

Barrierefreiheitsprobleme

Bei Zugänglichkeitsproblemen:

- Testen Sie mit verschiedenen Screenreadern (NVDA, JAWS, VoiceOver)
- Überprüfen Sie Farbkontraste mit WCAG-konformen Tools
- Validieren Sie die Tastaturbedienbarkeit aller interaktiven Elemente

Lizenz

Dieses Projekt steht unter der MIT-Lizenz - Details finden Sie in der LICENSE-Datei.

Mitwirkende

- JP.Böhm - Hauptentwickler und Projektmaintainer

Danksagungen

- Besonderer Dank gilt der Barrierefreiheits-Community für wertvolle Impulse
- Inspiration durch etablierte Sicherheitslösungen, neu gedacht mit Fokus auf universelle Zugänglichkeit