

1. JavaScript-Datei: **src/KameraBlock.js**

(Code für den Gutenberg-Block mit Kamerawechsel-Logik)

Diese Version implementiert den Wechsel zwischen der **environment** (Rückkamera) und **user** (Frontkamera) über einen Button. Der Kameramodus wird im React-State (**facingMode**) gespeichert und der Media-Stream bei jeder Änderung neu geladen.

JavaScript

None

```
import React, { useRef, useState, useEffect } from "react";
import { registerBlockType } from "@wordpress/blocks";

registerBlockType("kamera/foto-block", {
  title: "Kamera Foto Block",
  icon: "camera",
  category: "widgets",
  edit: () => <KameraBlock />,
  save: () => null,
});

function KameraBlock() {
  const videoRef = useRef(null);
  const canvasRef = useRef(null);
  const [stream, setStream] = useState(null);
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [photoTaken, setPhotoTaken] = useState(false);
  const [consent, setConsent] = useState(false);
  const [photoData, setPhotoData] = useState(null);
  // NEU: State für den Kameramodus (Standard: Rückkamera)
  const [facingMode, setFacingMode] = useState("environment");

  useEffect(() => {
    async function getCamera() {
      // Vorhandenen Stream stoppen, bevor ein neuer gestartet
      wird
      stream?.getTracks().forEach((track) => track.stop());

      try {
```

```

        const s = await navigator.mediaDevices.getUserMedia({
            // VERWENDET NEUEN STATE: Bestimmt Front- oder
Rückkamera
            video: { facingMode: facingMode },
            audio: false,
        });
        setStream(s);
        if (videoRef.current) videoRef.current.srcObject = s;
    } catch (err) {
        console.error("Fehler beim Zugriff auf Kamera", err);
        alert("Kamerazugriff fehlgeschlagen: " + err.name);
    }
}
getCamera();
// CLEANUP: Stoppt den Stream beim Entfernen der
Komponente
return () => {
    stream?.getTracks().forEach((track) => track.stop());
};
// ABHÄNGIGKEIT HINZUGEFGÜGT: Startet bei Wechsel des
facingMode neu
}, [facingMode]);

const takePhoto = () => {
    const context = canvasRef.current.getContext("2d");
    // Sicherstellen, dass das Canvas die korrekte Größe hat
    const videoWidth = videoRef.current.videoWidth;
    const videoHeight = videoRef.current.videoHeight;
    canvasRef.current.width = videoWidth;
    canvasRef.current.height = videoHeight;

    context.drawImage(videoRef.current, 0, 0, videoWidth,
videoHeight);
    const dataUrl = canvasRef.current.toDataURL("image/png");
    setPhotoData(dataUrl);
    setPhotoTaken(true);
};

const sendPhoto = async () => {
    if (!name || !email || !photoData || !consent)

```

```

        return alert("Bitte alle Felder ausfüllen.");

// Stream stoppen, sobald das Foto gesendet wird
stream?.getTracks().forEach((track) => track.stop());
setStream(null);

try {
    await fetch("/wp-json/kamera-api/v1/upload", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ name, email, photo: photoData
    })),
    });
    alert("Foto erfolgreich gesendet!");
} catch (err) {
    console.error(err);
    alert("Fehler beim Senden des Fotos");
}
};

// Funktion zum Wechseln des Kameramodus
const switchCamera = () => {
    setFacingMode(prevMode =>
        prevMode === 'environment' ? 'user' : 'environment'
    );
    // Foto-Status zurücksetzen, um zur Live-Vorschau
    zurückzukehren
    setPhotoTaken(false);
    setPhotoData(null);
};

return (
    <div className="p-4 border rounded-xl shadow-xl max-w-md
mx-auto">
        <h2>Kundenfoto aufnehmen</h2>
        {stream && (
            // Button zum Wechseln der Kamera, nur sichtbar,
            wenn Stream aktiv

```

```

        <button onClick={switchCamera} disabled={photoTaken}
style={{marginBottom: '10px'}}>
            {facingMode === 'environment' ? 'Zu Selfie-Kamera
wechseln' : 'Zur Rückkamera wechseln'}
        </button>
    )}

    {/* VIDEO-Element ist nur sichtbar, solange kein Foto
gemacht wurde */}
    {!photoTaken && <video ref={videoRef} autoPlay
playsInline width="320" height="240" style={{border: '1px
solid #ccc'}} />}

    <canvas ref={canvasRef} width="320" height="240"
hidden></canvas>
    <div>
        <input
            type="text"
            placeholder="Name"
            value={name}
            onChange={(e) => setName(e.target.value)}
        />
        <input
            type="email"
            placeholder="E-Mail"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
        />
        <label>
            <input
                type="checkbox"
                checked={consent}
                onChange={(e) => setConsent(e.target.checked)}
            />
            Ich bin einverstanden mit der Nutzung meines Fotos.
        </label>

        {/* Logik für Foto aufnehmen / Vorschau / Absenden */}
        {!photoTaken ? (
            <button onClick={takePhoto} disabled={!stream}>

```

```

        Foto machen
    </button>
) : (
    <>
        <img src={photoData} alt="Vorschau" style={{width:
'320px', height: '240px', objectFit: 'cover', border: '1px
solid black'}} />
        <button onClick={sendPhoto}
disabled={!consent}>Absenden</button>
        <button onClick={() => setPhotoTaken(false)}
style={{marginLeft: '10px'}}>Neues Foto</button>
    </>
    </div>
</div>
);
}

```

2. PHP-Datei: **kamera-foto-block.php**

(Plugin-Logik mit Mediathek-Integration und Sicherheitshinweisen)

Diese Version der PHP-Datei sorgt dafür, dass die Base64-Daten als reguläre WordPress-Uploads behandelt, in der **Mediathek** registriert und Metadaten erzeugt werden.

PHP

```

None
<?php
/*
Plugin Name: Kamera Foto Block
Description: Ein Gutenberg-Block mit Kamera-Funktion für
Kundenfotos.
Version: 1.0
Author: Dein Name
*/

// Block-Assets registrieren
function kamera_foto_block_register_block() {
    wp_register_script(

```

```

        'kamera-foto-block-editor',
        plugins_url('src/KameraBlock.js', __FILE__),
        array('wp-blocks', 'wp-element', 'wp-editor',
'wp-components', 'wp-i18n'),
        filemtime(plugin_dir_path(__FILE__) .
'src/KameraBlock.js')
    );

    register_block_type('kamera/foto-block', array(
        'editor_script' => 'kamera-foto-block-editor',
    ));
}
add_action('init', 'kamera_foto_block_register_block');

// REST API Endpoint
add_action('rest_api_init', function () {
    register_rest_route('kamera-api/v1', '/upload', array(
        'methods' => 'POST',
        'callback' => 'kamera_api_handle_upload',
        // ACHTUNG: __return_true bedeutet, dass JEDER (auch
nicht eingeloggte Nutzer) hochladen kann.
        // Für öffentlich zugängliche Formulare erforderlich,
aber SPAM-Schutz im Frontend wird empfohlen.
        'permission_callback' => '__return_true',
    ));
});

/**
 * Verarbeitet den Base64-Upload und speichert das Bild in der
WordPress Mediathek.
 */
function kamera_api_handle_upload($request) {
    // Stellen Sie sicher, dass die notwendigen
WordPress-Funktionen geladen sind
    if ( ! function_exists( 'wp_handle_sideload' ) || !
function_exists( 'wp_insert_attachment' ) ) {
        require_once( ABSPATH . 'wp-admin/includes/file.php'
    );

        require_once( ABSPATH . 'wp-admin/includes/image.php'
    );

```

```

    }

    $params = $request->get_json_params();
    $name = sanitize_text_field($params['name']);
    $email = sanitize_email($params['email']);
    $photo = $params['photo'];

    if (!$photo || !$name || !$email) {
        return new WP_Error('missing_data', 'Fehlende Daten',
array('status' => 400));
    }

    // 1. Base64-Daten dekodieren und temporär speichern
    $photo_data =
base64_decode(preg_replace('#^data:image/\w+;base64,#i', '',
$photo));
    $tmp_name = wp_tempnam();

    if (!file_put_contents($tmp_name, $photo_data)) {
        return new WP_Error('temp_save_error', 'Temporäre
Datei konnte nicht gespeichert werden.', array('status' =>
500));
    }

    // 2. WordPress-konformen Upload-Array erstellen (fälscht
einen Datei-Upload vor)
    $file = array(
        'name'      => 'kundenfoto_' . time() . '.png',
        'type'      => 'image/png',
        'tmp_name'  => $tmp_name,
        'error'     => 0,
        'size'      => filesize($tmp_name),
    );

    // 3. Datei durch WordPress verarbeiten lassen
    $upload_overrides = array( 'test_form' => false );
    $uploaded_file_info = wp_handle_sideload( $file,
$upload_overrides );

    // Temporäre Datei löschen

```

```

@unlink( $tmp_name );

if ( ! empty( $uploaded_file_info['error'] ) ) {
    return new WP_Error( 'upload_error', 'Fehler beim
Hochladen: ' . $uploaded_file_info['error'], array( 'status'
=> 500 ) );
}

$file_url = $uploaded_file_info['url'];
$file_path = $uploaded_file_info['file'];
$file_type = $uploaded_file_info['type'];

// 4. Datei in der WordPress Mediathek registrieren
$attachment = array(
    'post_mime_type' => $file_type,
    'post_title'      => 'Kundenfoto von ' . $name,
    'post_content'    => 'E-Mail: ' . $email,
    'post_status'     => 'inherit',
);

$attachment_id = wp_insert_attachment( $attachment,
$file_path );

// Metadaten für Bildgrößen generieren
$attach_data = wp_generate_attachment_metadata(
$attachment_id, $file_path );
wp_update_attachment_metadata( $attachment_id,
$attach_data );

// 5. E-Mail senden mit URL
wp_mail(
    get_option('admin_email'),
    'Neues Kundenfoto ( ' . $name . ' )',
    "Name: $name\nE-Mail: $email\n\nDas Foto wurde
erfolgreich in die Mediathek hochgeladen.\nLink zum Foto:
$file_url"
);

return new WP_REST_Response(array(

```

```
'success' => true,  
'attachment_id' => $attachment_id,  
'url' => $file_url,  
, 200);  
}
```



Kurzanleitung zur Installation

1. Erstellen Sie in Ihrem WordPress-Installationsverzeichnis unter `wp-content/plugins/` einen neuen Ordner (z.B. `kamera_foto_block`).
2. Erstellen Sie im Ordner `kamera_foto_block` die Datei `kamera-foto-block.php` und fügen Sie den Inhalt aus Abschnitt 2 ein.
3. Erstellen Sie im Ordner `kamera_foto_block` einen Unterordner `src`.
4. Erstellen Sie im Ordner `src` die Datei `KameraBlock.js` und fügen Sie den Inhalt aus Abschnitt 1 ein.
5. Melden Sie sich in WordPress an, gehen Sie zu **Plugins** und aktivieren Sie den **"Kamera Foto Block"**.
6. Auf jeder Seite oder in jedem Beitrag können Sie nun den neuen **Gutenberg-Block** hinzufügen.

Wichtig: Progressive Web App (PWA)

Um die App-Funktionalität (wie das Starten vom Homescreen, Offline-Fähigkeit) optimal zu nutzen, installieren Sie zusätzlich ein **PWA-Plugin** für WordPress (z.B. "PWA for WP & AMP" oder "WebPushr PWA"). Dieses Plugin generiert die notwendigen Service Worker und das Manifest, um Ihre Webseite zu einer installierbaren App zu machen.