Technischer Bericht

Projekt: Digital & Mobil in Deutschland

18. April 2026

Deutsche Version

Teil 1. Ersetzen von Emojis durch SVG-Icons im Wetterblock

1.1 Aufgabe

Im dynamischen Wetterblock der Website wurden Emojis (☀️ ☁️ ❄️ 🌧️ 🍃 💨) zur Darstellung der Wetterbedingungen und des Luftqualitätsindex (AQI) verwendet. Ziel war es, Emojis durch SVG-Icons über CSS-Klassen zu ersetzen, um eine bessere Kontrolle über das Erscheinungsbild auf allen Geräten, einschließlich Smart TV, zu gewährleisten.

1.2 Problem mit Emojis

Emojis werden auf verschiedenen Plattformen und Browsern unterschiedlich dargestellt. Auf einigen Smart TVs werden sie gar nicht unterstützt. SVG-Icons bieten volle Kontrolle über Größe, Farbe und Stil.

1.3 Lösung — CSS-Klassen für Icons

Eine separate Datei icons.css wurde erstellt mit einem Basisstil für alle Icons und spezifischen Klassen für jedes Icon:

```
/* Basisstil */
.icon-emoji {
    display: inline-block;
    width: 22px; height: 22px;
    background-repeat: no-repeat;
    background-size: contain;
    background-position: center;
    vertical-align: middle;
    margin-right: 8px;
}
```

```
.icon-sun    { background-image: url('../img/icon/2600.svg'); }
.icon-cloud  { background-image: url('../img/icon/2601.svg'); }
.icon-snow   { background-image: url('../img/icon/2744.svg'); }
.icon-rain   { background-image: url('../img/icon/1f327.svg'); }
.icon-leaf   { background-image: url('../img/icon/1f343.svg'); }
.icon-wind   { background-image: url('../img/icon/1f4a8.svg'); }
.icon-fog    { background-image: url('../img/icon/1f32b.svg'); }
.icon-warning { background-image: url('../img/icon/26a0.svg'); }
.icon-mask   { background-image: url('../img/icon/1f637.svg'); }
```

1.4 Änderungen in weather.js

In der Funktion applyWeatherData wurde innerText durch innerHTML ersetzt. Emojis wurden durch CSS-Klassen ersetzt:

```
// Vorher:
let icon = '☁';
if (code === 800) icon = '☀';
tempEl.innerText = `${city} ${icon} ${temp}°C`;
```

```
// Nachher:
let iconClass = 'icon-cloud';
if (code === 800)     iconClass = 'icon-sun';
else if (code > 800)  iconClass = 'icon-cloud';
else if (code >= 600) iconClass = 'icon-snow';
else if (code >= 300) iconClass = 'icon-rain';
```

```
tempEl.innerHTML = `${city} <span class="icon-emoji ${iconClass}"></span>
${temp}°C`;
```

1.5 Änderungen in der Funktion updateAQIUI

Anstatt innerHTML mit verschachteltem span wird className des vorhandenen Elements direkt geändert:

```
// Vorher:
icoEl.innerText = icon;
icoEl.style.color = color;
```

```
// Nachher:
icoEl.className = `icon-emoji ${iconClass}`;
icoEl.style.filter = `drop-shadow(0 0 4px ${color})`;
```

1.6 Änderungen in header.js (AppHeader-Komponente)

```
// Vorher:
<span id="aqi-icon">🍃</span>
```

```
// Nachher:
<span id="aqi-icon" class="icon-emoji icon-leaf"></span>
```

1.7 Diagnose und Problemlösung

- Icons wurden nicht angezeigt — Überprüfung durch background-color: red. Das rote Quadrat erschien, d.h. die Größen funktionieren korrekt.
- Klassenname-Konflikt — in JS wurden icon-wind und icon-warning verwendet, im CSS aber icon-air und icon-attention. CSS-Klassen wurden an JS-Namen angepasst.
- SVG-Dateien behielten ihre ursprünglichen Namen (air.svg, attention.svg) — kein Umbenennen der Dateien erforderlich.

1.8 Erweiterung der SVG-Icons auf alle HTML-Dateien

Nach erfolgreichem Test der Icon-Darstellung auf Smart TV und PC wurde entschieden, alle verwendeten Emojis in den HTML-Dateien durch SVG zu ersetzen.

Da das Projekt weiter aktualisiert wird, wurden zur Vereinfachung der Inhaltserstellung alle Emojis als SVG-Dateien mit Standardnamen von GitHub heruntergeladen. Beispiel: 2600.svg (☀), 1f1e9-1f1ea.svg (DE). Dies ermöglicht die schnelle Suche nach dem gewünschten Icon über den Unicode-Code anhand der öffentlich verfügbaren Emoji-Liste.

Alle Emojis in allen HTML-Dateien wurden nach einheitlichem Prinzip ersetzt:

```
<h1 style="margin-top: 0; margin-bottom: 10px; display: flex; align-items:
center;">
    Willkommen!
    <span class="icon-emoji icon-1f1e9-1f1ea" style="margin-left: 8px;"></span>
</h1>
```

Teil 2. Implementierung des Blocks für geomagnetische Aktivität

2.1 Aufgabe

Der Wert der geomagnetischen Aktivität in der Top-Bar war als Zahl 2 fest einprogrammiert und wurde nie aktualisiert. Ziel war es, echte Kp-Index-Daten über eine API abzurufen.

```
// Vorher — statische Zahl:
<span class="w-label">${texts.geo}</span> <span id="geo">2</span>
```

```
// Nachher — leer, wird vom Skript befüllt:
<span class="w-label">${texts.geo}</span> <span id="geo">--</span>
```

2.2 Versuch 1 — GFZ Potsdam (durch Safari blockiert)

GFZ Potsdam (Deutschland) ist seit 1997 die offizielle Quelle für den Kp-Index. Die API ist kostenlos unter der CC BY 4.0 Lizenz verfügbar. Es wurde ein Skript magnet.js mit einem GFZ-Request geschrieben:

```
const res = await fetch(
    `https://kp.gfz.de/app/json/?start=${startStr}&end=${endStr}&index=Kp`
);
```

Der Test im Safari-Browser auf dem iPhone zeigte einen Fehler:

```
ERROR: Load failed
```

Ursache: Safari auf iOS blockiert fetch()-Anfragen an Drittanbieter-Domains strenger als Desktop-Browser (CORS-Richtlinie). Eine Lösung über Vercel Serverless Function (Proxy) wurde erwogen, würde aber die Architektur der Website verkomplizieren. Diese Option wurde verworfen.

2.3 Versuch 2 — NOAA SWPC (funktioniert)

Das NOAA Space Weather Prediction Center stellt den planetarischen Kp-Index mit offenem CORS für alle Browser einschließlich Safari auf iOS bereit. Daten werden jede Minute aktualisiert:

```
const res = await fetch(
    'https://services.swpc.noaa.gov/json/planetary_k_index_1m.json'
);
const kp = data[data.length - 1].kp_index;
```

2.4 Finaler Code magnet.js

```
const MAGNET_CACHE_TTL = 30 * 60 * 1000; // 30 Minuten
```

```
async function getGeomagneticActivity() {
    const cached = localStorage.getItem('magnetCache');
    if (cached) {
        const parsed = JSON.parse(cached);
        if (Date.now() - parsed.timestamp < MAGNET_CACHE_TTL) {
            applyMagnetData(parsed.kp); return;
        }
    }
    const res = await fetch(
        'https://services.swpc.noaa.gov/json/planetary_k_index_1m.json'
    );
    const data = await res.json();
    const kp = data[data.length - 1].kp_index;
    localStorage.setItem('magnetCache', JSON.stringify({
```

```
            kp, timestamp: Date.now()
        }));
        applyMagnetData(kp);
    }
```

```
function applyMagnetData(kp) {
    const el = document.getElementById('geo');
    if (!el) return;
    el.innerText = Math.round(kp * 10) / 10;
    let color;
    if (kp <= 2)      color = '#2ecc71'; // ruhig
    else if (kp <= 4) color = '#f1c40f'; // moderat
    else if (kp <= 6) color = '#e67e22'; // aktiv
    else              color = '#e74c3c'; // Sturm
    el.style.color = color;
    el.style.fontWeight = 'bold';
}
```

2.5 Reihenfolge der Script-Einbindung

Wichtig: magnet.js muss NACH components.js eingebunden werden, da die AppHeader-Komponente das Element id="geo" im DOM erstellt. Wird magnet.js früher geladen, gibt getElementById null zurück.

```
<script src="js/components.js"></script>
<script src="js/weather.js"></script>
<script src="js/magnet.js"></script>  <!-- nach components.js! -->
<script src="js/main.js"></script>
```

2.6 Diagnose auf iPhone (ohne DevTools)

```
// Test 1 — Überprüfung der Dateieinbindung:
document.getElementById('geo').innerText = 'TEST';
// TEST erschien nicht → Datei nicht eingebunden
```

```
// Nach Verschieben von magnet.js unter components.js:
// TEST erschien → Datei funktioniert
```

```
// Test 2 — fetch()-Überprüfung:
alert('ERROR: ' + e.message);
// ERROR: Load failed → CORS durch Safari blockiert
```

2.7 Vergleich der Datenquellen

GFZ Potsdam: lokaler Index für Europa, wird alle 3 Stunden aktualisiert, CORS in Safari iOS blockiert.

NOAA SWPC: planetarischer Index (Durchschnitt von 13 Observatorien), wird jede Minute aktualisiert, CORS für alle Browser offen. Als Hauptquelle gewählt.

Fazit

Beide Verbesserungen wurden erfolgreich implementiert und über Vercel aus dem Branch compact des Repositories fin325/digital-mobil-deutschland deployed.

- SVG-Icons haben Emojis im Wetterblock und AQI über CSS background-image ersetzt
- Alle Emojis in allen HTML-Dateien wurden durch SVG ersetzt — Icons wurden von GitHub mit Standard-Unicode-Dateinamen heruntergeladen
- Geomagnetische Aktivität erhält echte Kp-Index-Daten von NOAA alle 30 Minuten
- Farbliche Anzeige: Grün (0-2) / Gelb (3-4) / Orange (5-6) / Rot (7-9)
- Caching in localStorage verhindert unnötige API-Anfragen

> **Die Website Digital & Mobil in Deutschland funktioniert jetzt korrekt nicht nur auf iOS und Android, sondern auch auf PC und Smart TV — alle Icons werden lokal aus der SVG-Datenbasis geladen und sind unabhängig von der Plattform oder dem systeminternen Emoji-Rendering.**

Технический отчёт

Проект: Digital & Mobil in Deutschland

Апрель 2026

Русская версия

Часть 1. Замена эмодзи на SVG-иконки в погодном блоке

1.1 Задача

В динамическом погодном блоке сайта использовались эмодзи (☀️ ☁️ ❄️ 🌧️ 🍃 💨) для отображения погодных условий и индекса качества воздуха (AQI). Задача — заменить эмодзи на SVG-иконки через CSS-классы для лучшего контроля над внешним видом на всех устройствах, включая Smart TV.

1.2 Проблема с эмодзи

Эмодзи отображаются по-разному на разных платформах и браузерах. На некоторых Smart TV они не поддерживаются вообще. SVG-иконки дают полный контроль над размером, цветом и стилем.

1.3 Решение — CSS-классы для иконок

Создан отдельный файл icons.css с базовым стилем для всех иконок и конкретными классами для каждой иконки:

```
/* Базовый стиль */
.icon-emoji {
    display: inline-block;
    width: 22px; height: 22px;
    background-repeat: no-repeat;
    background-size: contain;
    background-position: center;
    vertical-align: middle;
    margin-right: 8px;
}
```

```
.icon-sun   { background-image: url('../img/icon/2600.svg'); }
.icon-cloud { background-image: url('../img/icon/2601.svg'); }
.icon-snow  { background-image: url('../img/icon/2744.svg'); }
.icon-rain  { background-image: url('../img/icon/1f327.svg'); }
.icon-leaf  { background-image: url('../img/icon/1f343.svg'); }
.icon-wind  { background-image: url('../img/icon/1f4a8.svg'); }
.icon-fog   { background-image: url('../img/icon/1f32b.svg'); }
.icon-warning { background-image: url('../img/icon/26a0.svg'); }
.icon-mask  { background-image: url('../img/icon/1f637.svg'); }
```

1.4 Изменения в weather.js

В функции applyWeatherData заменён innerText на innerHTML. Эмодзи заменены на CSS-классы:

```
// Было:
let icon = '☁';
if (code === 800) icon = '☀';
tempEl.innerText = `${city} ${icon} ${temp}°C`;
```

```
// Стало:
let iconClass = 'icon-cloud';
if (code === 800)     iconClass = 'icon-sun';
else if (code > 800)  iconClass = 'icon-cloud';
else if (code >= 600) iconClass = 'icon-snow';
else if (code >= 300) iconClass = 'icon-rain';
```

```
tempEl.innerHTML = `${city} <span class="icon-emoji ${iconClass}"></span>
${temp}°C`;
```

1.5 Изменения в функции updateAQIUI

Вместо innerHTML с вложенным span используется прямое изменение className существующего элемента:

```
// Было:
icoEl.innerText = icon;
icoEl.style.color = color;
icoEl.style.textShadow = `0 0 8px ${color}66`;
```

```
// Стало:
icoEl.className = `icon-emoji ${iconClass}`;
icoEl.style.filter = `drop-shadow(0 0 4px ${color})`;
```

1.6 Изменения в header.js (компонент AppHeader)

Элемент aqi-icon изначально задаётся с нужными классами — без эмодзи по умолчанию:

```
// Было:
<span id="aqi-icon">🍃</span>
```

```
// Стало:
<span id="aqi-icon" class="icon-emoji icon-leaf"></span>
```

1.7 Диагностика и решение проблем

- Иконки не отображались — добавлена проверка через background-color: red. Красный квадрат появился, значит размеры работают корректно.
- Несоответствие имён классов — в JS использовались icon-wind и icon-warning, а в CSS были icon-air и icon-attention. Классы в CSS переименованы под JS-названия.
- Файлы SVG остались с прежними именами (air.svg, attention.svg) — переименование файлов не потребовалось.

1.8 Расширение SVG-иконок на все HTML-файлы

После успешного тестирования отображения иконок на платформах Smart TV и ПК было принято решение заменить все используемые эмодзи в файлах HTML на SVG.

Так как проект будет обновляться далее, для упрощения создания нового контента с SVG были скачаны все эмодзи в формате SVG со стандартными именами файлов с GitHub. Пример: 2600.svg (🌞), 1f1e9-1f1ea.svg (🇩🇪). Это позволяет использовать общедоступный список названий эмодзи для быстрого поиска нужной иконки по коду.

Все эмодзи во всех HTML-файлах заменены по единому принципу:

```
<h1 style="margin-top: 0; margin-bottom: 10px; display: flex; align-items:
center;">
    Willkommen!
    <span class="icon-emoji icon-1f1e9-1f1ea" style="margin-left: 8px;"></span>
</h1>
```

Часть 2. Реализация блока геомагнитной активности

2.1 Задача

Значение геомагнитной активности в топ-баре было захардкожено как цифра 2 и никогда не обновлялось. Задача — получать реальные данные Kp-индекса через API.

```
// Было в header.js — статичная цифра:
<span class="w-label">${texts.geo}</span> <span id="geo">2</span>
```

```
// Стало — пустое значение, заполняется скриптом:
<span class="w-label">${texts.geo}</span> <span id="geo">--</span>
```

2.2 Попытка 1 — GFZ Potsdam (заблокировано Safari)

GFZ Potsdam (Германия) — официальный источник Kp-индекса с 1997 года. API предоставляется бесплатно под лицензией CC BY 4.0. Был написан скрипт magnet.js с запросом к GFZ:

```
const res = await fetch(
    `https://kp.gfz.de/app/json/?start=${startStr}&end=${endStr}&index=Kp`
);
const kp = data.Kp[data.Kp.length - 1];
```

Проверка в браузере Safari на iPhone показала ошибку:

```
ERROR: Load failed
```

Причина: Safari на iOS блокирует fetch()-запросы к сторонним доменам строже чем десктопный браузер (CORS-политика). Рассматривалось решение через Vercel Serverless Function (прокси), но это усложнило бы архитектуру сайта. От этого варианта отказались.

2.3 Попытка 2 — NOAA SWPC (работает)

NOAA Space Weather Prediction Center предоставляет планетарный Kp-индекс с открытым CORS для всех браузеров включая Safari на iOS. Данные обновляются каждую минуту:

```
const res = await fetch(
    'https://services.swpc.noaa.gov/json/planetary_k_index_1m.json'
);
const kp = data[data.length - 1].kp_index;
```

2.4 Итоговый код magnet.js

```
const MAGNET_CACHE_TTL = 30 * 60 * 1000; // 30 Minuten
```

```
async function getGeomagneticActivity() {
    const cached = localStorage.getItem('magnetCache');
    if (cached) {
        const parsed = JSON.parse(cached);
        if (Date.now() - parsed.timestamp < MAGNET_CACHE_TTL) {
            applyMagnetData(parsed.kp); return;
        }
    }
    const res = await fetch(
        'https://services.swpc.noaa.gov/json/planetary_k_index_1m.json'
    );
    const data = await res.json();
    const kp = data[data.length - 1].kp_index;
```

```
    localStorage.setItem('magnetCache', JSON.stringify({
        kp, timestamp: Date.now()
    }));
    applyMagnetData(kp);
}
```

```
function applyMagnetData(kp) {
    const el = document.getElementById('geo');
    if (!el) return;
    el.innerText = Math.round(kp * 10) / 10;
    let color;
    if (kp <= 2)      color = '#2ecc71';
    else if (kp <= 4) color = '#f1c40f';
    else if (kp <= 6) color = '#e67e22';
    else              color = '#e74c3c';
    el.style.color = color;
    el.style.fontWeight = 'bold';
}
```

2.5 Порядок подключения скриптов

Критически важно: magnet.js должен подключаться ПОСЛЕ components.js, так как компонент AppHeader создаёт элемент id="geo" в DOM. Если magnet.js загружается раньше — getElementById возвращает null.

```
<script src="js/components.js"></script>
<script src="js/weather.js"></script>
<script src="js/magnet.js"></script>  <!-- после components.js! -->
<script src="js/main.js"></script>
```

2.6 Диагностика на iPhone (без DevTools)

```
// Тест 1 — проверка подключения файла:
document.getElementById('geo').innerText = 'TEST';
// TEST не появился → файл не подключён
```

```
// После перемещения magnet.js ниже components.js:
// TEST появился → файл работает
```

```
// Тест 2 — проверка fetch():
alert('ERROR: ' + e.message);
// ERROR: Load failed → CORS заблокирован Safari
```

2.7 Сравнение источников данных

GFZ Potsdam: локальный индекс для Европы, обновляется каждые 3 часа, CORS заблокирован в Safari iOS.

NOAA SWPC: планетарный индекс (среднее по 13 обсерваториям), обновляется каждую минуту, CORS открыт для всех браузеров. Выбран как основной источник.

Итог

Оба улучшения успешно реализованы и задеплоены через Vercel из ветки compact репозитория fin325/digital-mobil-deutschland.

- SVG-иконки заменили эмодзи в погодном блоке и AQI через CSS background-image
- Все эмодзи во всех HTML-файлах заменены на SVG — иконки скачаны с GitHub со стандартными именами по Unicode-кодам
- Геомагнитная активность получает реальные данные Kp-индекса от NOAA каждые 30 минут
- Цветовая индикация: зелёный (0-2) / жёлтый (3-4) / оранжевый (5-6) / красный (7-9)
- Кэширование в localStorage исключает лишние API-запросы

> **Сайт Digital & Mobil in Deutschland теперь корректно работает не только на iOS и Android, но и на ПК и Smart TV — все иконки хранятся локально в базе SVG-файлов и не зависят от платформы или системного рендеринга эмодзи.**