API-Nutzung & Beispiele

- Javascript - Beispielhafte Anfrage
- Php - Beispielhafte Anfrage
- Windows Terminal - Beispielhafte Anfrage

Javascript - Beispielhafte Anfrage

Beschreibung:

Diese JavaScript-Funktion ruft Sensordaten von der SmartDog API v2 ab.

Die API liefert Sensordaten für einen bestimmten Zeitraum im JSON-Format zurück.

Programmiersprache: JavaScript (async/await)

API-URL: `https://apiv2.smart-dog.eu/index.php`

Code:

```
async function getSensorData() {
    const url = "https://apiv2.smart-dog.eu/index.php";

    const requestData = {
        action: "getSensorData",
        apikey: "6641d282073d76b625987af5141d3e2a",
        SensorID: "551941",
        UTC_TIMESTAMP_FROM: "1739867939",
        UTC_TIMESTAMP_TO: "1739877939"
    };

    try {
        const response = await fetch(url, {
            method: "POST",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify(requestData)
        });
```

```
        if (!response.ok) {
            throw new Error(`HTTP-Fehler! Status: ${response.status}`);
        }

        const data = await response.json();
        document.getElementById("sensorOutput").textContent = JSON.stringify(data, null, 2);
    } catch (error) {
        console.error("Fehler beim Abrufen der Sensordaten:", error);
        document.getElementById("sensorOutput").textContent = "Fehler beim Abrufen der Daten.";
    }
}
```

Beschreibung der Funktion:

1. API-Request vorbereiten:
    - Erstellt eine Anfrage an die API mit den Parametern:
        - `action`: `"getSensorData"`
        - `apikey`: `"6641d282073d76b625987af5141d3e2a"`
        - `SensorID`: `"551941"`
        - `UTC_TIMESTAMP_FROM`: `"1739867939"`
        - `UTC_TIMESTAMP_TO`: `"1739877939"`
2. Daten abrufen mit `fetch()`:
    - Sendet die Anfrage per `POST`-Methode an die API.
    - Setzt den `Content-Type`-Header auf `"application/json"`.
    - Wandelt das `requestData`-Objekt in JSON um und sendet es als `body`.
3. Fehlermanagement:
    - Falls der Server keine gültige Antwort sendet (`!response.ok`), wird eine `Error`-Exception geworfen.

- Falls ein Fehler auftritt, wird dieser in der Konsole (`console.error`) ausgegeben und eine Fehlermeldung im HTML-Element mit der ID `"sensorOutput"` angezeigt.

Beispiel für eine erfolgreiche API-Antwort:


```
{
  "sensor_id": 551941,
  "valid": 1,
  "datasets": {
    "1739868002": {
      "DATA": "251",
      "TIMESTAMP_UTC": 1739868002,
      "TIMESTAMP_LOCAL": 1739871602
    },
    "1739868302": {
      "DATA": "257",
      "TIMESTAMP_UTC": 1739868302,
      "TIMESTAMP_LOCAL": 1739871902
    }
  }
}
```

Php - Beispielhafte Anfrage

Beschreibung:

Dieses PHP-Skript ruft Sensordaten von der SmartDog API v2 ab.

Die API liefert Sensordaten für einen bestimmten Zeitraum im JSON-Format zurück.

Programmiersprache: PHP

API-URL: `https://apiv2.smart-dog.eu/index.php`

Code:

```
<?php

function getSensorData() {
    $url = "https://apiv2.smart-dog.eu/index.php";

    $requestData = [
        "action" => "getSensorData",
        "apikey" => "6641d282073d76b625987af5141d3e2a",
        "SensorID" => "551941",
        "UTC_TIMESTAMP_FROM" => "1739867939",
        "UTC_TIMESTAMP_TO" => "1739877939"
    ];

    $options = [
        "http" => [
            "header"  => "Content-Type: application/json\r\n",
            "method"  => "POST",
            "content" => json_encode($requestData)
        ]
    ];

    $context  = stream_context_create($options);
    $response = file_get_contents($url, false, $context);
```

```
    if ($response === FALSE) {
        die("Fehler beim Abrufen der Sensordaten");
    }

    $data = json_decode($response, true);
    echo "<pre>" . print_r($data, true) . "</pre>";
}

getSensorData();

?>
```

Beschreibung der Funktion:

1. API-Request vorbereiten:
    - Erstellt eine Anfrage an die API mit den Parametern:
        - `action`: `"getSensorData"`
        - `apikey`: `"6641d282073d76b625987af5141d3e2a"`
        - `SensorID`: `"551941"`
        - `UTC_TIMESTAMP_FROM`: `"1739867939"`
        - `UTC_TIMESTAMP_TO`: `"1739877939"`
2. Datenabruf mit `file_get_contents()`:
    - Sendet eine `POST`-Anfrage mit JSON-Daten an die API.
    - Nutzt `stream_context_create()` zum Setzen der `Content-Type`-Header.
    - Falls die Antwort fehlschlägt (`FALSE`), gibt das Skript eine Fehlermeldung aus.
3. Antwortverarbeitung:
    - Dekodiert die JSON-Antwort mit `json_decode()`.
    - Gibt die Sensordaten formatiert (`print_r()`) als HTML-`<pre>`-Block aus.

Beispiel für eine erfolgreiche API-Antwort:


```
{
   "sensor_id": 551941,
   "valid": 1,
   "datasets": {
      1739868002.000000: {
         "DATA": "251",
         "TIMESTAMP_UTC": 1739868002,
         "TIMESTAMP_LOCAL": 1739871602
      },
      1739868302.000000: {
         "DATA": "257",
         "TIMESTAMP_UTC": "1739868302.000000",
         "TIMESTAMP_LOCAL": 1739871902
      }
   }
}
```


Alternative mit cURL (empfohlen für größere API-Anfragen):

```
<?php

function getSensorData() {
   $url = "https://apiv2.smart-dog.eu/index.php";

   $requestData = [
      "action" => "getSensorData",
      "apikey" => "6641d282073d76b625987af5141d3e2a",
      "SensorID" => "551941",
      "UTC_TIMESTAMP_FROM" => "1739867939",
```

```
        "UTC_TIMESTAMP_TO" => "1739877939"
    ];

    $ch = curl_init($url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, ["Content-Type: application/json"]);
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($requestData));

    $response = curl_exec($ch);

    if (curl_errno($ch)) {
        die("Fehler beim Abrufen der Sensordaten: " . curl_error($ch));
    }

    curl_close($ch);

    $data = json_decode($response, true);
    echo "<pre>" . print_r($data, true) . "</pre>";
}

getSensorData();

?>
```

Diese Variante verwendet `cURL`, was für größere API-Anfragen besser geeignet ist.

Einsatz in einer HTML-Seite:

```
<button onclick="location.reload();">Sensordaten abrufen</button>
<pre><?php getSensorData(); ?></pre>
```

Beim Klicken auf den Button `"Sensordaten abrufen"` wird die PHP-Funktion ausgeführt, und die Daten erscheinen im `<pre>`-Tag.

Windows Terminal - Beispielhafte Anfrage

Beschreibung:
Dieser `cURL`-Befehl sendet eine `POST`-Anfrage an die SmartDog API v2, um Sensordaten für einen bestimmten Zeitraum abzurufen.

Tool: cURL
API-URL: `https://apiv2.smart-dog.eu/index.php`
Methode: `POST`

Code:

```
curl -X POST "https://apiv2.smart-dog.eu/index.php" \
    -H "Content-Type: application/json" \
    -d '{
        "action": "getSensorData",
        "apikey": "6641d282073d76b625987af5141d3e2a",
        "SensorID": "551941",
        "UTC_TIMESTAMP_FROM": "1739867939",
        "UTC_TIMESTAMP_TO": "1739877939"
       }'
```

Beschreibung der Parameter:

- `-X POST` → Die Anfrage wird per `POST`-Methode gesendet.
- `-H "Content-Type: application/json"` → Setzt den Header auf JSON-Daten.
- `-d '{...}'` → Enthält die zu sendenden JSON-Daten.

Beispiel für eine erfolgreiche API-Antwort:


```
{
   "sensor_id": 551941,
   "valid": 1,
   "datasets": {
      "1739868002": {
         "DATA": "251",
         "TIMESTAMP_UTC": 1739868002,
         "TIMESTAMP_LOCAL": 1739871602
      },
      "1739868302": {
         "DATA": "257",
         "TIMESTAMP_UTC": 1739868302,
         "TIMESTAMP_LOCAL": 1739871902
      }
   }
}
```


Alternative mit Ausgabe in einer Datei:

Falls du die Antwort direkt in einer Datei speichern möchtest:

```
curl -X POST "https://apiv2.smart-dog.eu/index.php" \
    -H "Content-Type: application/json" \
    -d '{
         "action": "getSensorData",
         "apikey": "6641d282073d76b625987af5141d3e2a",
         "SensorID": "551941",
         "UTC_TIMESTAMP_FROM": "1739867939",
         "UTC_TIMESTAMP_TO": "1739877939"
       }' > response.json
```

Die Antwort wird dann in `response.json` gespeichert.