The background features a complex, abstract pattern. The upper portion is filled with dense, overlapping, wavy lines in shades of pink and magenta, creating a sense of depth and movement. The lower portion transitions into a solid pink background with a regular grid of small black dots, resembling a halftone or dot-matrix pattern. Two black rectangular boxes are overlaid on the image, containing white text.

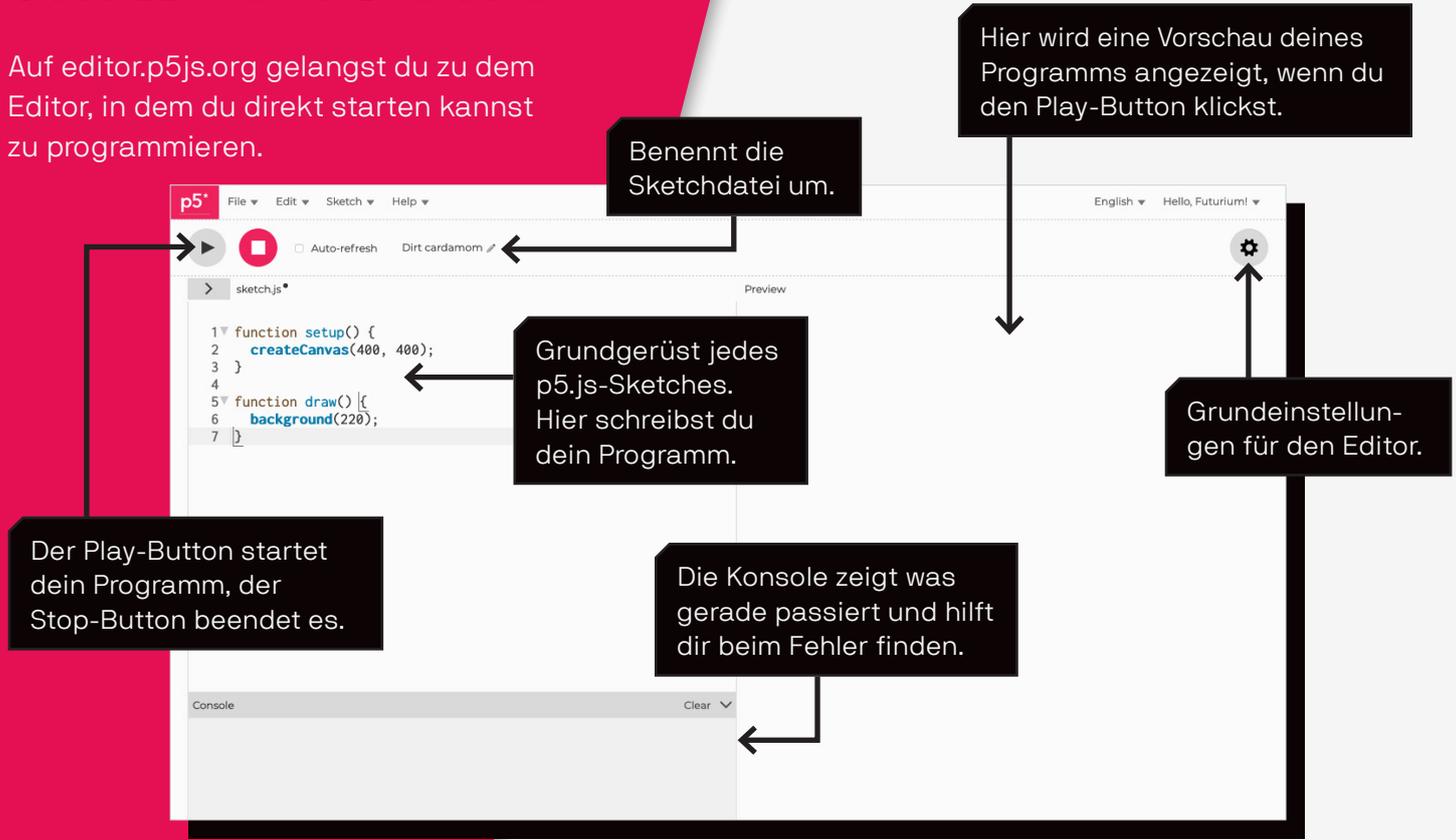
# DATENVISUALISIERUNG

## MIT P5.JS

Mit den Lernkarten kannst du dir deinen eigenen ersten Prototypen Schritt für Schritt nachbauen.

# DIE OBERFLÄCHE VON EDITOR.P5JS.ORG

Auf editor.p5js.org gelangst du zu dem Editor, in dem du direkt starten kannst zu programmieren.



# HELLO ELLIPSE!

Einer kleinen Programmier-tradition folgend, wollen wir auch mit einem „Hello World“-Programm starten, oder genauer gesagt mit „Hello Ellipse“.

Die Idee der JavaScript-Bibliothek p5 basiert auf der quelloffenen Programmiersprache Processing. Es ist eine Art Software-Skizzenbuch, mit dem Künstler\*innen, Designer\*innen und anderen Menschen mit Ideen das kreative Gestalten mit Programmierung einfach gemacht wird.

Man kann damit grafische und interaktive Anwendungen für das Web direkt im Browser erschaffen! P5 ist Open Source und wurde von Lauren McCarthy in die Welt gebracht.

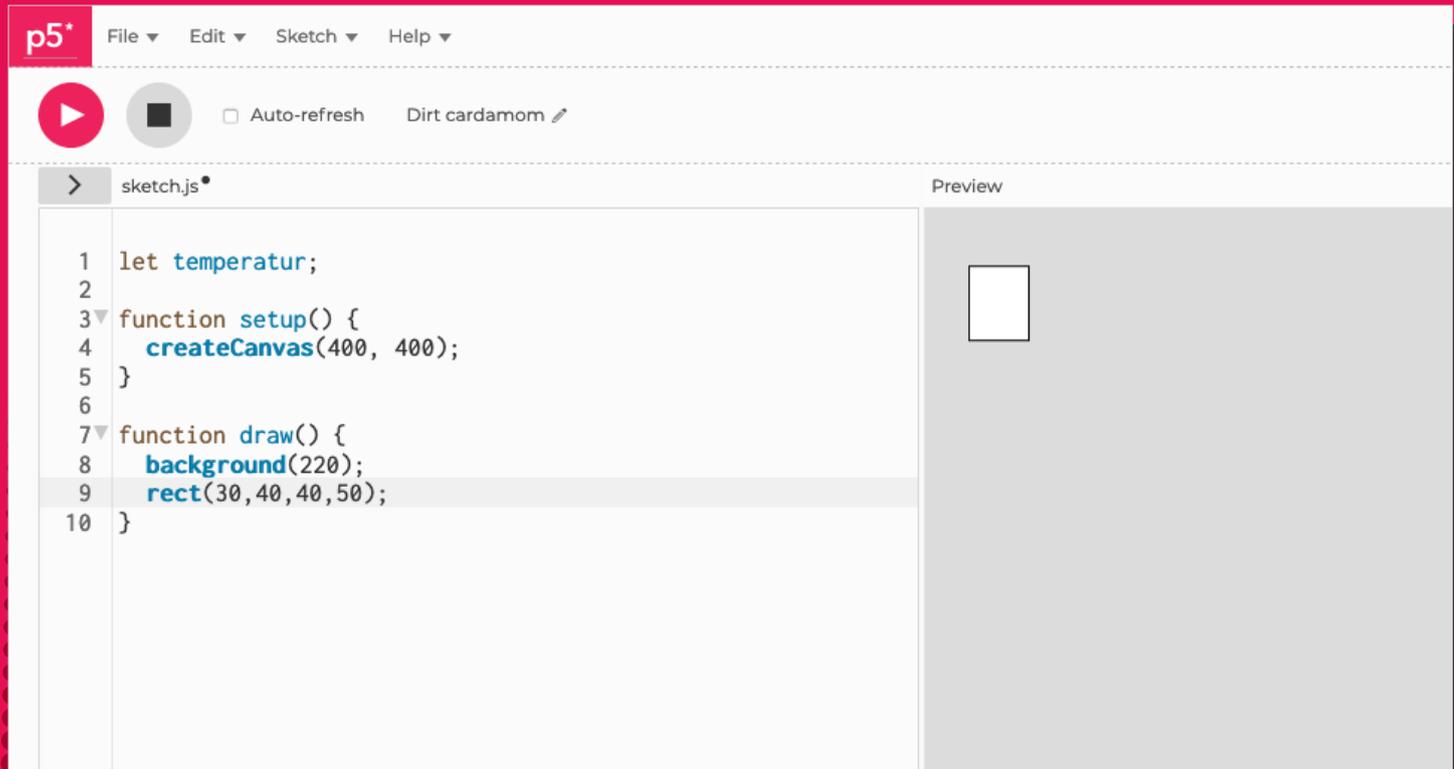
The screenshot shows the p5.js web editor interface. At the top, there is a navigation bar with the p5 logo and menu items: File, Edit, Sketch, and Help. Below the navigation bar, there are control buttons: a red play button (run), a grey square button (stop), and a grey square button (clear). A text box with an arrow points to the play button, containing the text: "Starte dein erstes Programm und gehe mit deiner Maus über den Canvas-Bereich!". Below the control buttons, the file name "sketch.js" is displayed, along with the text "Saved: 14 minutes ago". The main area is divided into two sections: a code editor on the left and a preview window on the right. The code editor contains the following code:

```
1 function setup() {  
2   createCanvas(400, 400);  
3 }  
4  
5 function draw() {  
6   ellipse(mouseX, mouseY, 30, 30);  
7 }
```

The preview window shows a black and white drawing of a complex, looping, ribbon-like shape. A text box with an arrow points to the preview window, containing the text: "Canvas-Bereich (Zeichenfläche)". At the bottom right of the preview window, there is a "Clear" button with a dropdown arrow.

# EIN RECHTECK ZEICHNEN

Für die Datenvisualisierung wollen wir ein Rechteck je nach Temperatur einfärben, dafür programmieren wir zu Beginn ein Rechteck auf unsere Zeichenfläche.



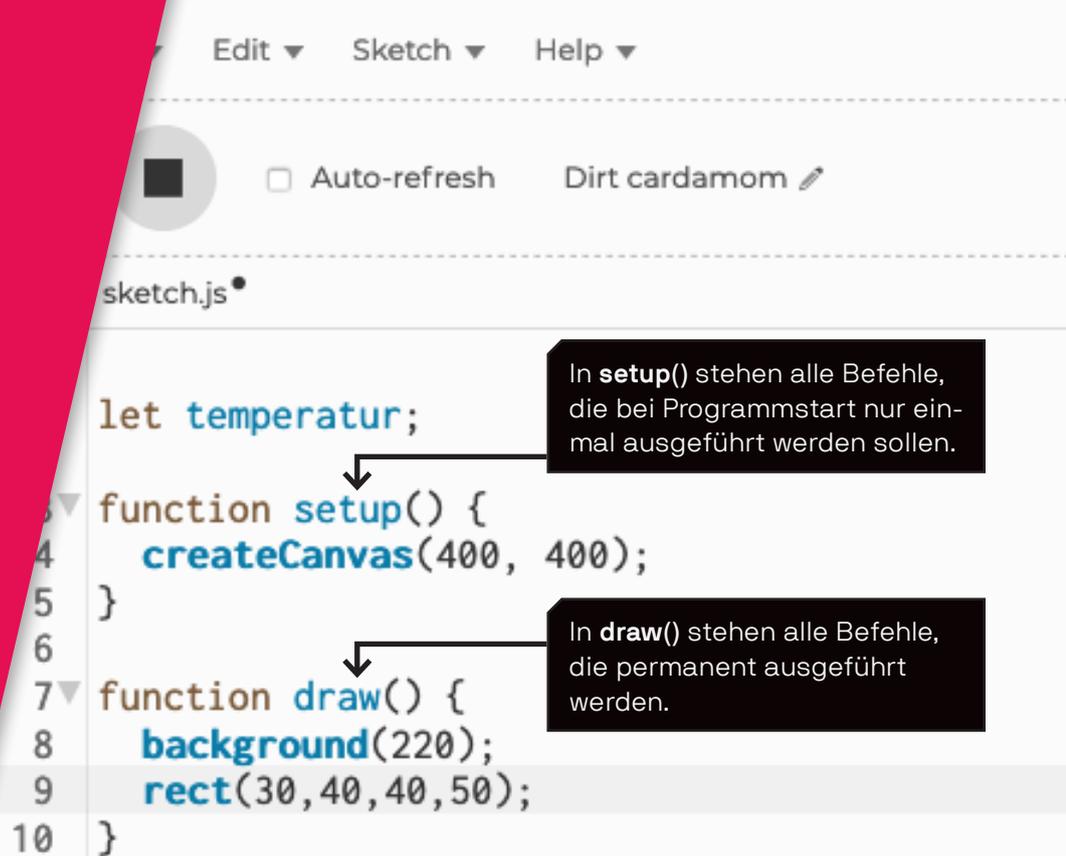
The screenshot shows the p5.js IDE interface. The top menu bar includes 'File', 'Edit', 'Sketch', and 'Help'. Below the menu bar, there is a play button, a square icon, and a checkbox for 'Auto-refresh'. The user's name 'Dirt cardamom' is visible. The main workspace is divided into two panels: 'sketch.js' on the left and 'Preview' on the right. The 'sketch.js' panel contains the following code:

```
1 let temperatur;
2
3 function setup() {
4   createCanvas(400, 400);
5 }
6
7 function draw() {
8   background(220);
9   rect(30,40,40,50);
10 }
```

The 'Preview' panel shows a white rectangle with a black border on a black background, representing the output of the code.

# SETUP & DRAW FUNKTION

p5.js stellt verschiedene Funktionen zur Verfügung, die automatisch aufgerufen werden. Die beiden wichtigsten sind **setup()** und **draw()**. Sie können verschiedene Befehle enthalten, die Schritt für Schritt abgearbeitet werden. Die Funktion **setup()** wird bei Programmstart einmal ausgeführt, zum Beispiel muss nur einmal **createCanvas()** aufgerufen werden. Danach wiederholt sich die Funktion **draw()** unendlich oft.



The image shows a screenshot of a code editor interface. At the top, there are menu items: 'Edit', 'Sketch', and 'Help'. Below the menus, there is a dark square icon, a checkbox labeled 'Auto-refresh', and the text 'Dirt cardamom' with an edit icon. The main area shows a code file named 'sketch.js' with the following code:

```
let temperatur;  
function setup() {  
  createCanvas(400, 400);  
}  
function draw() {  
  background(220);  
  rect(30, 40, 40, 50);  
}
```

Two callout boxes with arrows pointing to the code:

- The first callout box points to the **setup()** function and contains the text: "In **setup()** stehen alle Befehle, die bei Programmstart nur einmal ausgeführt werden sollen."
- The second callout box points to the **draw()** function and contains the text: "In **draw()** stehen alle Befehle, die permanent ausgeführt werden."

# CANVAS, RECT & BACKGROUND

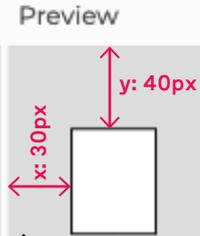
damom ✎

```
2  
3 function setup() {  
4   createCanvas(400, 400);  
5 }  
6  
7 function draw() {  
8   background(220);  
9   rect(30, 40, 40, 50);  
10 }
```

Erzeugt eine Zeichenfläche für dein Programm durch Angabe der Pixelmaße (x-Achse, y-Achse).

Gibt der Zeichenfläche eine Farbe in RGB-Werten (Rot, Grün, Blau). Wenn nur ein Wert in der Klammer steht, gilt er für alle 3 Farben.

Zeichnet an der Stelle x: 30, y: 40, ein Rechteck mit der Breite 40 und der Höhe 50 Pixel.



# VARIABLEN

Eine Variable kann Informationen speichern, die an beliebigen Stellen im Code verwendet werden.

Bildlich kann man sich eine Variable wie eine Box vorstellen. Die Box ist die Variable, der Inhalt der Wert. Der Wert kann sich beliebig verändern. In unserem Beispiel wollen wir einen Temperaturwert (z. B. 14) einer senseBox in der Box „temperatur“ speichern.



File Edit Sketch Help

Auto-refresh Dirt cardamom

sketch.js

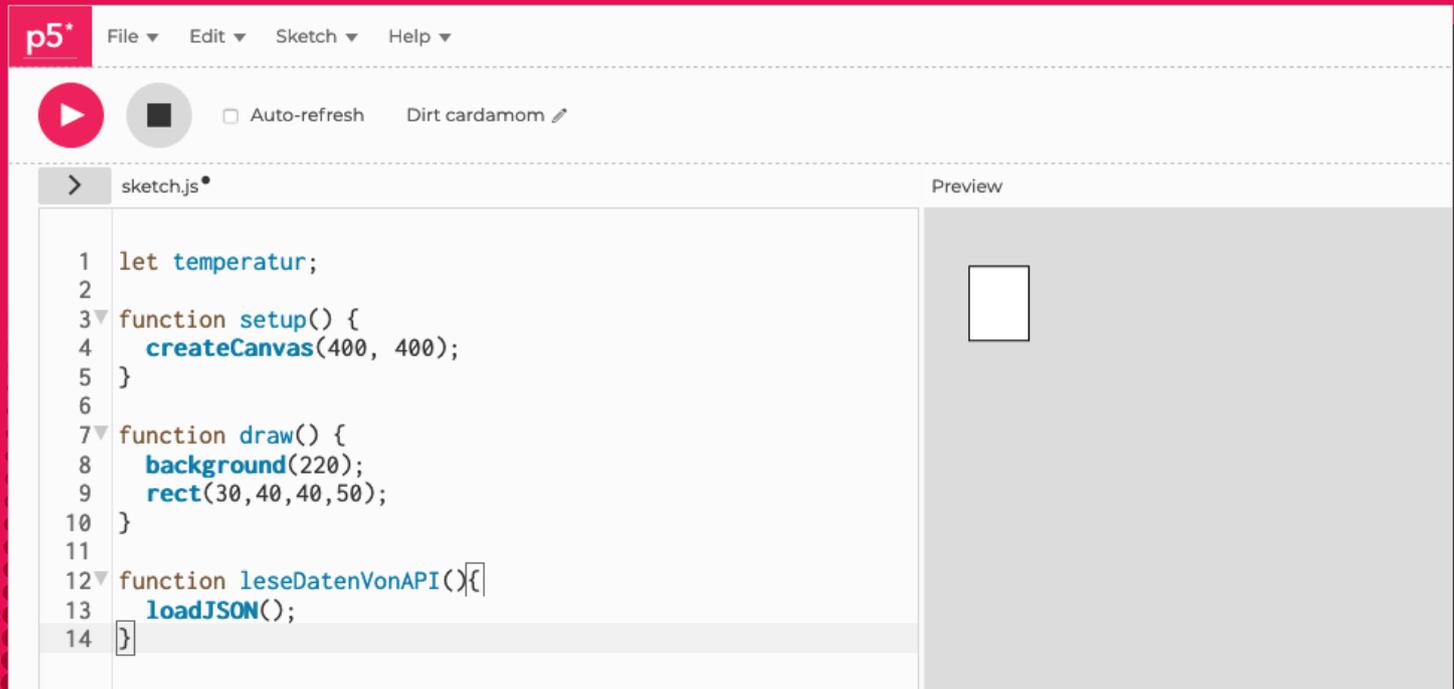
```

1 let temperatur;
2
3 function setup() {
4   createCanvas(400, 400);
5 }
6
7 function draw() {
8   background(220);
9   rect(30, 40, 40, 50);
10 }
  
```

Das Schlüsselwort „let“ legt eine Variable fest. Der Variablenname ist hier „temperatur“.

# DATEN VON EINER SENSEBOX LADEN

Die Funktion **leseDatenVonAPI()** ruft eine andere Funktion namens **loadJSON()** auf. Die Funktion **loadJSON()** wird schon von p5.js bereit gestellt, um Daten des Datei-Typs "JSON" abzurufen. Das kannst du dir wie eine Text-Datei vorstellen mit einer gewissen Struktur, um beim Programmieren gezielt auf Elemente in der Datei zuzugreifen.



The screenshot shows the p5.js IDE interface. The top menu bar includes 'File', 'Edit', 'Sketch', and 'Help'. Below the menu bar, there are control buttons: a play button, a stop button, an 'Auto-refresh' checkbox, and the user name 'Dirt cardamom'. The main workspace is split into two panes. The left pane, titled 'sketch.js', contains the following code:

```
1 let temperatur;
2
3 function setup() {
4   createCanvas(400, 400);
5 }
6
7 function draw() {
8   background(220);
9   rect(30, 40, 40, 50);
10 }
11
12 function leseDatenVonAPI() {
13   loadJSON();
14 }
```

The right pane, titled 'Preview', shows a white rectangular canvas with a black border, representing the visual output of the sketch.

# EIGENE FUNKTIONEN IN JAVASCRIPT

Funktionen sind ein Block von Anweisungen mit einem Namen. Der Funktionsblock wird einmal definiert. Danach können die JavaScript-Befehle der Funktion über den Namen mehrfach im Programm aufgerufen werden. Diese Funktionen können dann zum Beispiel in **setup()** und **draw()** aufgerufen und ausgeführt werden.

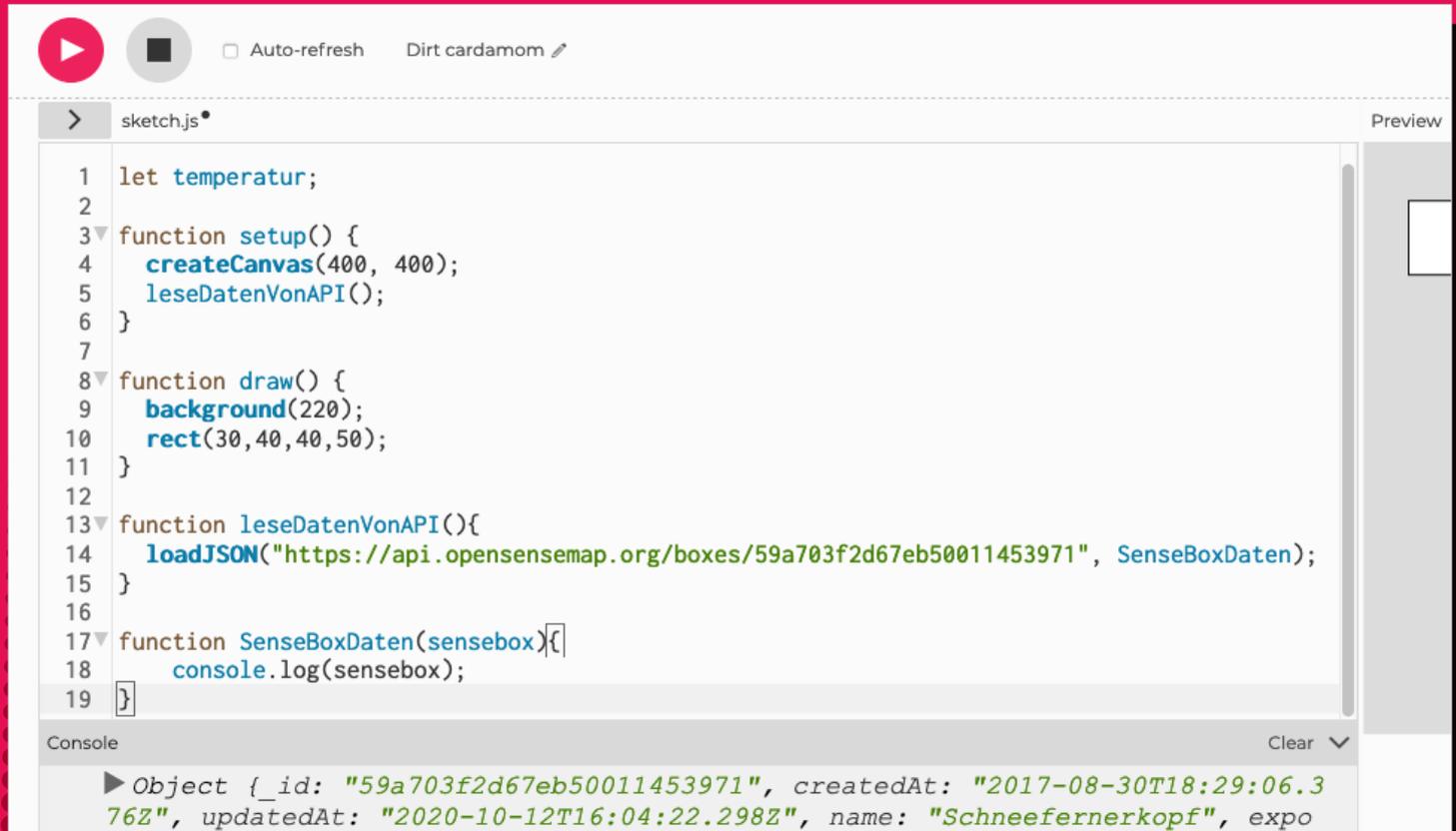
```
8   background(220);  
9   rect(30, 40, 40, 50);  
10  }  
11  
12  function leseDatenVonAPI() {  
13    loadJSON();  
14  }
```

Preview



Das Schlüsselwort „function“ erschafft und benennt eine eigene Funktion in JavaScript. Alles was in den geschweiften Klammern {...} steht, gehört zu dieser Funktion.

# DATEN VON EINER SENSEBOX IN DIE KONSOLE AUSGEBEN



The screenshot shows a code editor with the following JavaScript code:

```
1 let temperatur;
2
3 function setup() {
4   createCanvas(400, 400);
5   leseDatenVonAPI();
6 }
7
8 function draw() {
9   background(220);
10  rect(30,40,40,50);
11 }
12
13 function leseDatenVonAPI(){
14   loadJSON("https://api.opensensemap.org/boxes/59a703f2d67eb50011453971", SenseBoxDaten);
15 }
16
17 function SenseBoxDaten(sensebox){
18   console.log(sensebox);
19 }
```

The console output shows the following object:

```
Object {_id: "59a703f2d67eb50011453971", createdAt: "2017-08-30T18:29:06.376Z", updatedAt: "2020-10-12T16:04:22.298Z", name: "Schneefernerkopf", expo
```

 Auto-refresh

Dirt cardamom

&gt; sketch.js

```
1 let temperatur;
2
3 function setup() {
4   createCanvas(400, 400);
5   leseDatenVonAPI();
6 }
7
8 function draw() {
9   background(220);
10  rect(30,40,40,50);
11 }
12
13 function leseDatenVonAPI(){
14   loadJSON("https://api.opensensemap.org/boxes/59a703f2d67eb50011453971");
15 }
16
17 function SenseBoxDaten(sensebox){
18   console.log(sensebox);
19 }
```

Hier wird die Funktion **leseDatenVonAPI()** aufgerufen und ausgeführt. Das Programm springt zu dem Teil, wo die Funktion deklariert wurde und führt dort ihre Befehle aus.

SEITE 11 - INFORMATION

## EIGENE FUNKTION AUFRUFEN

Die Funktion **leseDatenVonAPI()** wurde zuvor nur deklariert. Erst durch Aufrufen der Funktion in **setup()**, wird sie das erste Mal ausgeführt.

Das Programm springt sozusagen zu der Stelle, an der die Funktion geschrieben ist, führt die Befehle aus und springt dann wieder zurück zu **setup()**. In **setup()** ist das der letzte Befehl gewesen, sodass danach **draw()** ausgeführt wird.

Console

Clear

```
▶ Object {_id: "59a703f2d67eb50011453971", createdAt: "2017-08-30T18:29:06.376Z", updatedAt: "2020-10-12T16:04:22.298Z", name: "Schneefernerkopf", exposure: "outdoor"...
```

# LOADJSON(PATH,CALLBACK)

Die Funktion **loadJSON (path, callback)**, erwartet mindestens zwei Parameter. Der Parameter „path“ ist die Adresse zu der JSON-Datei, der Parameter „callback“ erwartet eine Funktion, die aufgerufen wird, wenn **loadJSON()** Werte geladen hat. Die empfangenen Werte werden der Callback-Funktion (in unserem Fall der Funktion **SenseBoxDaten()**) übermittelt.

Parameter path

Die ID der senseBox

Parameter callback

```

12
13 function leseDatenVonAPI(){
14   loadJSON("https://api.opensensemap.org/boxes/59a703f2d67eb50011453971", SenseBoxDaten);
15 }
16
17 function SenseBoxDaten(sensebox){
18   console.log(sensebox);
19 }

```

Die Callback-Funktion SenseBoxDaten() wird aufgerufen, sobald die Funktion loadJSON() Daten geladen hat. Die JSON-Daten befinden sich in der Variablen sensebox.

Console

```

▶ Object {_id: "59a703f2d67eb50011453971", updatedAt: "2020-10-12T16:04:22...", type: "outdoor", exp

```

## ID VON EINER SENSEBOX

Auf der openSenseMap findest du viele senseBoxes. Damit die unterschiedlichen Daten jeder senseBox sortierbar und auswertbar sind, braucht jede Sensobox eine nur einmal vorkommende Nummer oder Zahlenfolge oder eine Kombination aus beidem. Diese Ziffernkombination ist dann die ID (Identifikator) einer Sensobox, die damit eindeutig bestimmt werden kann.

The image shows a browser window with the URL `https://opensensemap.org/explore/59a703f2d67eb50011453971`. The page displays a map with a green location pin labeled 'Schneefernerkopf' and 'Neue Welt'. A text box explains that the ID is found in the browser address bar after `opensensemap/explore/...`. Another text box instructs the user to click on a senseBox to find its ID. A third text box warns not to confuse the ID with the API address `https://api.opensensemap.org/boxes/...` used by the `loadJSON()` function.

Die ID einer senseBox findest du in der Browserzeile nach `opensensemap/explore/...`. Die folgenden Zahlen und Ziffern sind die ID der senseBox.

Klicke eine senseBox an, um ihre ID herauszufinden.

**Nicht Verwechseln!** Die Funktion `loadJSON()` braucht die Adresse zu der API (`https://api.opensensemap.org/boxes/...`).



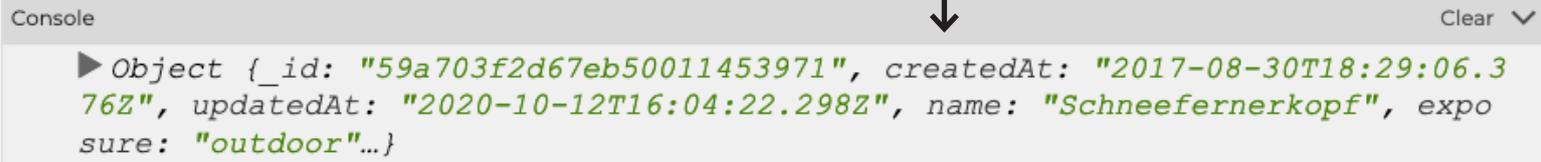
Starte das Programm und empfang deine ersten Daten!

```

4   createCanvas(400, 400);
5   leseDatenVonAPI();
6 }
7
8 function draw() {
9   background(220);
10  rect(30,40,40,50);
11 }
12
13 function leseDatenVonAPI(){
14   loadJSON("https://api.opensensemap.org/boxes/59a703f2d67eb50011453971", SenseBoxDaten);
15 }
16
17 function SenseBoxDaten(sensebox){
18   console.log(sensebox);
19 }

```

Die Funktion **console.log()** gibt Text und andere Daten zum Protokollieren in die Konsole vom Editor aus.

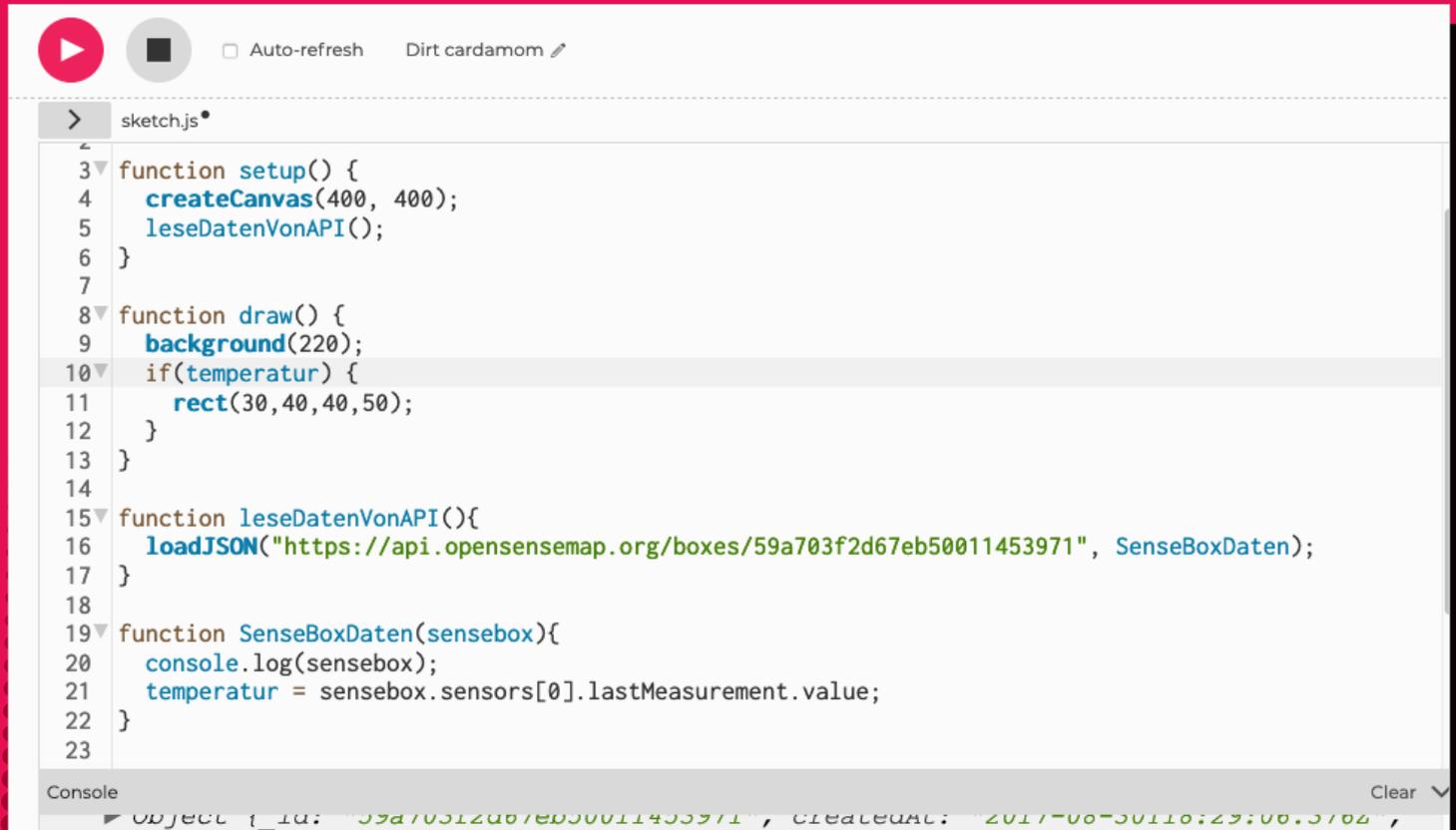


SEITE 14 - INFORMATION

# DATEN VON EINER SENSEBOX IN DIE KONSELE AUSGEBEN

Damit wir sehen können, ob wir wirklich Daten von einer senseBox erhalten haben, lassen wir uns diese in die Konsole ausgeben. Das kann je nach Internetverbindung etwas dauern.

# DIE TEMPERATUR VON DER SENSEBOX LESEN



```
sketch.js •
3 function setup() {
4   createCanvas(400, 400);
5   leseDatenVonAPI();
6 }
7
8 function draw() {
9   background(220);
10  if(temperatur) {
11    rect(30,40,40,50);
12  }
13 }
14
15 function leseDatenVonAPI(){
16   loadJSON("https://api.opensensemap.org/boxes/59a703f2d67eb50011453971", SenseBoxDaten);
17 }
18
19 function SenseBoxDaten(sensebox){
20   console.log(sensebox);
21   temperatur = sensebox.sensors[0].lastMeasurement.value;
22 }
23
```

Console

Object { id: "59a70312d67eb50011453971", createdAt: "2017-06-30T16:29:06.370Z", ...}

# JSON-DATEIEN IN EINE VARIABLE ÜBERFÜHREN

Damit wir in der **draw()**-Funktion mit dem Temperaturwert arbeiten können, müssen wir den Wert in eine Variable überführen, die auch von anderen Funktionen genutzt werden kann.

Doch woher wissen wir, dass wir wirklich gerade den Temperatursensor aus der Datei abfragen? Mit `sensebox.sensors[0]` rufen wir den ersten Sensor der `senseBox` auf. Auf der `openSenseMap` kannst du die Reihenfolge der Sensoren deiner ausgewählten Box einsehen. Alternativ liegen diese Informationen auch in der JSON-Datei. Hier können wir sehen, dass der Temperatursensor an erster Stelle steht. In der Informatik wird immer von der Null an gezählt, darum ist der Sensor in der Datei an der `[0]` Stelle.

```
15 function leseDatenVonAPI(){
16   loadJSON("https://api.opensensemap.org/boxes/59a70
17 }
18
19 function SenseBoxDaten(sensebox){
20   console.log(sensebox);
21   temperatur = sensebox.sensors[0].lastMeasurement.value;
22 }
23
```

Hier wird ein bestimmter Wert der JSON-Datei in die Variable „temperatur“ übergeben.

Console

```
Object {
  updatedAt:
  or"..."}
```

```
5971", createdAt: "2017-06-30T16:29:06.376Z",
2", name: "Schneefernerkopf", exposure: "outdo
```

Clear

# EINEN BESTIMM- TEN SENSOR ÜBER DIE OPEN- SENSEMAP FINDEN

Wenn du auf der openSenseMap eine senseBox auswählst, werden die Sensoren in einer Liste aufgezeigt. Die Einträge der Liste kannst du gedanklich mit der Null startend durchzählen. So kommst du zu der richtigen Nummer entsprechend der Struktur in der JSON-Datei.

The screenshot shows the openSenseMap interface for a station named 'Schneefernerkopf'. The station is located in the 'Gletscherabfahrten' area. The interface displays the following information:

- Station Name:** Schneefernerkopf
- Group:** ifgl
- Location:** draußen
- Last Update:** Aktualisieren in 44 Sek.
- Information:** Hier stehen weitere Informationen zu dieser senseBox, der Besitzer dieser Station hat jedoch nichts hinterlegt.
- Sensors:**
  - Temperatur: -7.70 °C (updated vor einer Minute)
  - rel. Luftfeuchte: 93.01 % (updated vor einer Minute)
  - Luftdruck: 1257.92 hPa (updated vor einem Monat)
  - Beleuchtungsstärke: 5108.00 lx (updated vor 3 Jahren)
  - Beleuchtungsstärke 2: 7193.00 lx (updated vor 3 Jahren)
  - UV-Intensität: (updated vor 3 Jahren)

Two callout boxes provide additional context:

- Callout 1:** Die Temperatur ist auf der ersten Stelle der Liste. In unserem Programm ist das dann bei: `sensebox.sensors.[0]...`
- Callout 2:** Die relative Luftfeuchte würden wir finden unter: `sensebox.sensors.[1]...`

# EINEN BESTIMM- TEN SENSOR ÜBER DIE JSON- DATEI FINDEN

Eine JSON-Datei kann unterschiedliche Daten einer senseBox enthalten. Damit wir die Daten eines Sensors einsehen können, navigieren wir uns durch die verschiedene Reiter der Datei. Klicke auf die kleinen Dreiecke, um die tieferliegenden Ebenen ansehen zu können.

p5\* File ▾ Edit ▾ Sketch ▾ Help ▾

▶ ◼ □ Auto-refresh Dirt cardamom ✎

Öffnet die tieferliegenden Ebenen.

```

Object {_id: "59a703f2d67eb50011453971", createdAt: "2017-08-30T18:29:06.376Z",
updatedAt: "2020-10-12T16:04:22.298Z", name: "Schneefernerkopf",
exposure: "outdoor"...}
  _id: "59a703f2d67eb50011453971"
  createdAt: "2017-08-30T18:29:06.376Z"
  updatedAt: "2020-10-12T16:04:22.298Z"
  name: "Schneefernerkopf"
  exposure: "outdoor"
▼sensors: Array[8]
  ▼0: Object
    title: "Temperatur"
    unit: "°C"
    sensorType: "SHT10"
    icon: "osem-thermometer"
    _id: "59a703f2d67eb50011453971"
  ▼lastMeasurement: Object
    value: "-8.04"
    createdAt: "2020-10-12T16:04:22.298Z"
  ▶ 1: Object
  ▶ 2: Object
  ▶ 3: Object

```

In dem [0] - Objekt von „sensors“ liegt der Wert von dem Temperatursensor.

Den Wert für die Temperatur finden wir unter: sensebox.sensors[0].lastMeasurement.value und beträgt gerade -8.04°C.

 Auto-refresh

Dirt cardamom

&gt; sketch.js

```

2
3 function setup() {
4   createCanvas(400, 400);
5   leseDatenVonAPI();
6 }
7
8 function draw() {
9   background(220);
10  if(temperatur) {
11    rect(30, 40, 40, 50);
12  }
13 }
14
15 function
16 loadJSC
17 }
18
19 function
20 console.log(sensebox);
21 temperatur = sensebox.sensors[0].lastMeasurement.value;
22 }
23

```

`if(temperatur){...}` prüft, ob die Variable gesetzt wurde. Denn erst wenn die Variable „temperatur“ einen Wert enthält, kann das Rechteck mit einer entsprechenden Farbe gezeichnet werden.

## IF-BEDINGUNGEN

Oft ist es notwendig bestimmte Codezeilen nur in bestimmten Fällen auszuführen.

Häufig werden dafür if-Anweisungen genutzt.

```
g/boxes/59a703f2d67eb50011453971", SenseBoxDaten);
```

Console

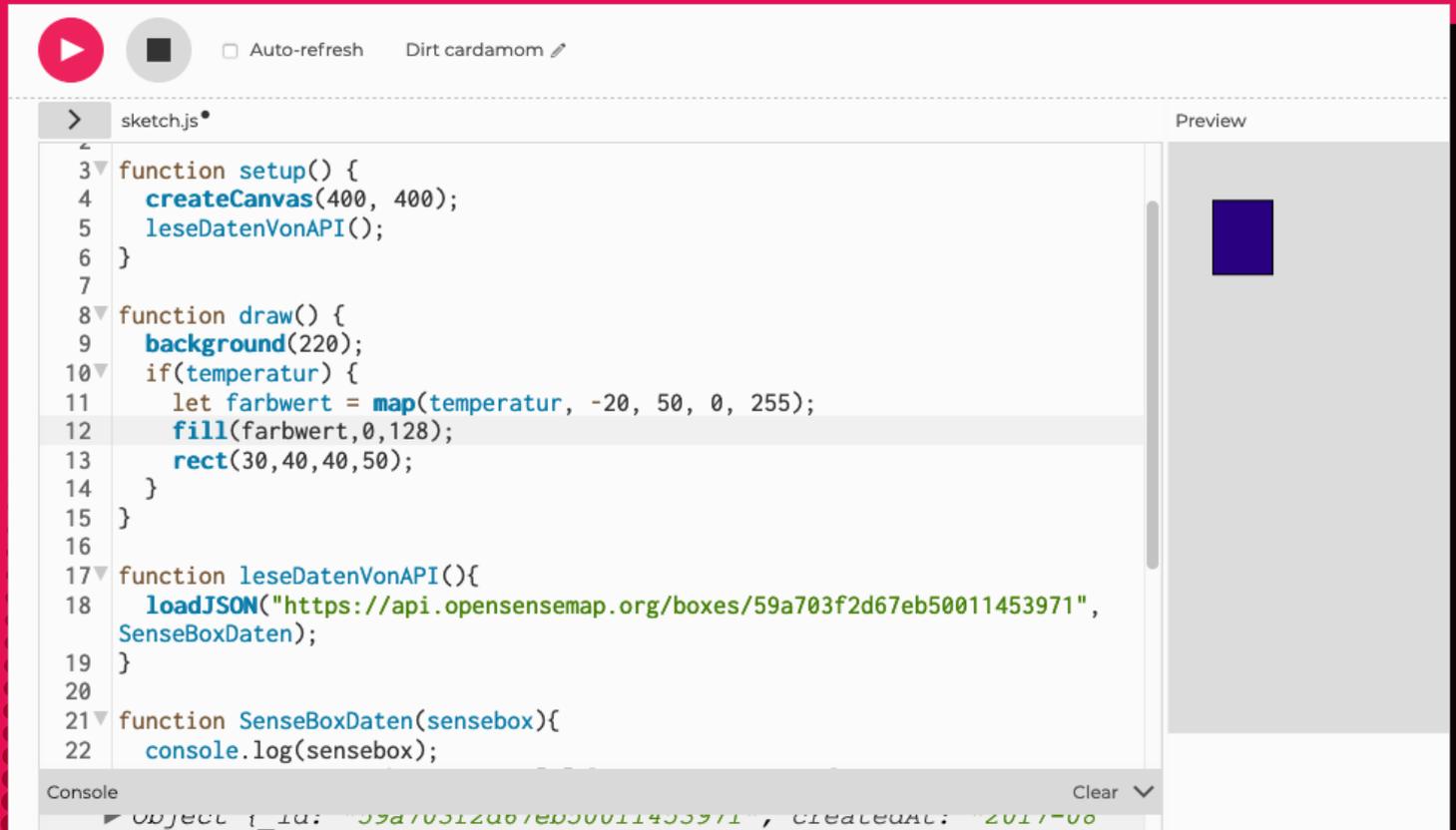
Clea

```

▶ Object {_id: "59a703f2d67eb50011453971", createdAt: "2017-08-30T16:29:08.378Z",
updatedAt: "2020-10-12T16:46:42.052Z", name: "Schneefernerkopf", exposure: "outdoor"...}

```

# DAS RECHTECK EINFÄRZEN



The screenshot shows a code editor with the following JavaScript code:

```
3 function setup() {
4   createCanvas(400, 400);
5   leseDatenVonAPI();
6 }
7
8 function draw() {
9   background(220);
10  if(temperatur) {
11    let farbwert = map(temperatur, -20, 50, 0, 255);
12    fill(farbwert,0,128);
13    rect(30,40,40,50);
14  }
15 }
16
17 function leseDatenVonAPI(){
18   loadJSON("https://api.opensensemap.org/boxes/59a703f2d67eb50011453971",
19   SenseBoxDaten);
20 }
21 function SenseBoxDaten(sensebox){
22   console.log(sensebox);
```

The preview window on the right shows a blue rectangle on a gray background.

Console

```
Object { id: "59a703f2d67eb50011453971", createdAt: "2017-06"
```

 Auto-refresh

Dirt cardamom

&gt; sketch.js

```

3 function setup() {
4   createCanvas(400, 400);
5   leseDatenVonAPI();
6 }
7
8 function draw() {
9   background(220);
10  if(temperatur) {
11    let farbwert = map(temperatur, -20, 50, 0, 255);
12    fill(farbwert, 0, 128);
13    rect(30, 40, 40, 50);
14  }
15 }
16
17 function leseDatenVon
18   loadJSON("https://
SenseBoxDaten);
19 }
20
21 function SenseBoxDaten(sensebox){
22   console.log(sensebox);

```

Hier wird der Temperaturwert in den Bereich von RGB-Werten umgerechnet.

fill() färbt das Rechteck ein. Anstatt des Rotwertes verwenden wir hier die vorher definierte Variable „farbwert“.

9a703

SEITE 21 - INFORMATION

## VARIABLE FÜR DEN FARBWERT

Zum Einfärben des Rechteckes werden RGB-Werte in einem Bereich von 0 bis 255 benötigt. Temperaturwerte liegen aber in einem anderen Bereich (meist zwischen -20 und 50 Grad). Damit der Temperaturwert in dem RGB-Wertebereich liegt, kann mit der Funktion `map()` der Temperaturwert im neuen Bereich umgerechnet werden. Nach Umrechnung entspricht zum Beispiel eine Temperatur von -20 Grad dann im RGB-Wertebereich 0 und 50 Grad entspräche dann dem Wert 255.

Console

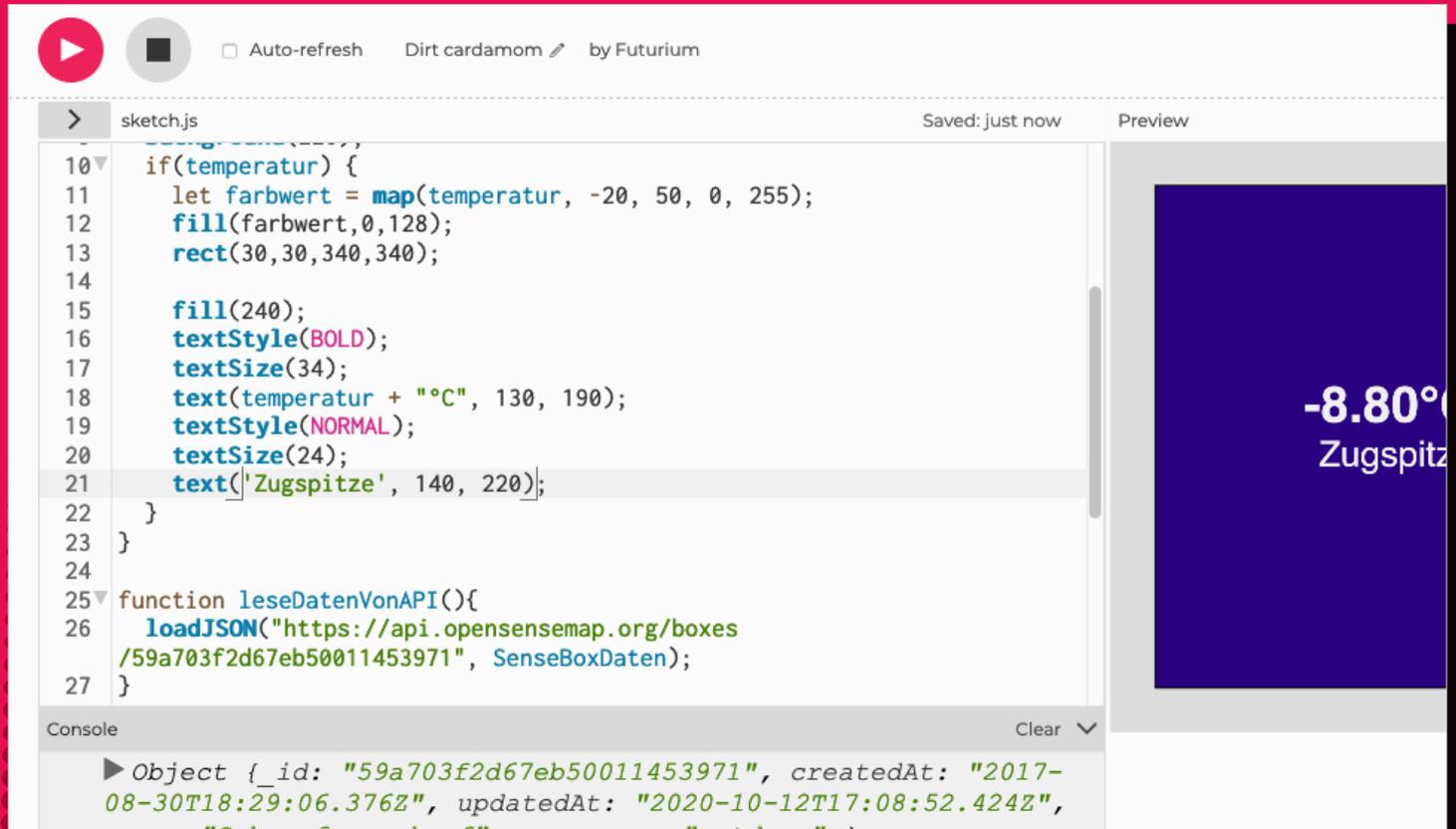
Clear

```

Object {_id: "9a70312a67eb50011453971", createdAt: "2017-06-30T18:29:06.376Z", updatedAt: "2020-10-12T16:48:42.989Z", name: "Schneefernerkopf", exposure: "outdoor"...}

```

# DAS RECHTECK ANPASSEN UND DEN TEXT SETZEN



Auto-refresh   Dirt cardamom by Futurium

```
10 if(temperatur) {
11   let farbwert = map(temperatur, -20, 50, 0, 255);
12   fill(farbwert,0,128);
13   rect(30,30,340,340);
14
15   fill(240);
16   textStyle(BOLD);
17   textSize(34);
18   text(temperatur + "°C", 130, 190);
19   textStyle(NORMAL);
20   textSize(24);
21   text('Zugspitze', 140, 220);
22 }
23 }
24
25 function leseDatenVonAPI(){
26   loadJSON("https://api.opensensemap.org/boxes
27   /59a703f2d67eb50011453971", SenseBoxDaten);
28 }
```

Saved: just now   Preview

-8.80°C  
Zugspitze

Console   Clear

```
Object {_id: "59a703f2d67eb50011453971", createdAt: "2017-08-30T18:29:06.376Z", updatedAt: "2020-10-12T17:08:52.424Z",
```



Auto-refresh

Dirt cardamom by Futurium

sketch.js

```
10 if (temperatur) {
11   fill(240);
12   textStyle(BOLD);
13   textSize(32);
14   text(temperatur + "°C", 130, 190);
15   textStyle(NORMAL);
16   textSize(24);
17   text('Zugspitze', 140, 220);
18 }
19 }
20 }
21 }
22 }
23 }
24 }
25 function leseDatenVonAPI(){
26   loadJSON("https://api.opensensemap.org/boxes
27   /59a703f2d67eb50011453971", SenseBoxDaten);
28 }
```

`fill(240)` färbt den nachfolgenden Text weiß ein.

`textStyle(BOLD)` macht den Text fett (bold).

`textSize(32)` setzt die Textgröße auf 32 Pixel.

## TEMPERATUR ANZEIGEN

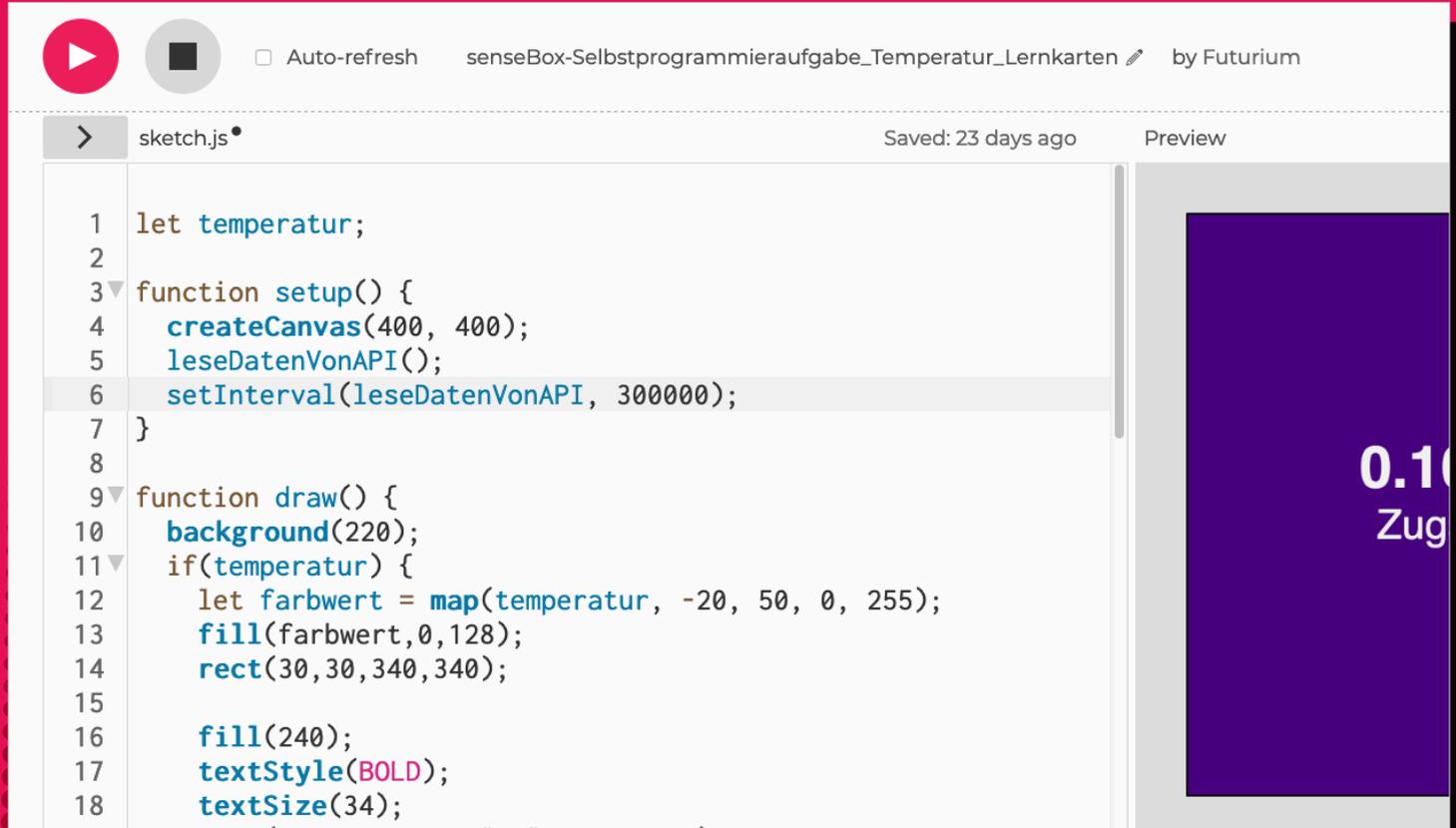
Die Funktion `text(„Text“, x, y)` schreibt auf deiner Zeichenfläche Text an der Stelle (x, y). Auch Variablen können als Text ausgegeben werden. In unserem Beispiel geben wir die Temperatur (temperatur) zusammen (+) mit der Maßeinheit °C („°C“) aus.

Console

Clear

```
▶ Object {_id: "59a703f2d67eb50011453971", createdAt: "2017-08-30T18:29:06.376Z", updatedAt: "2020-10-12T17:08:52.424Z", name: "Schneefernerkopf", exposure: "outdoor..."}
```

# SENSORDATEN NACH 5 MINUTEN AKTUALISIEREN



The screenshot shows a code editor interface with a play button, a square icon, and an "Auto-refresh" checkbox. The file name is "senseBox-Selbstprogrammieraufgabe\_Temperatur\_Lernkarten" by "Futurium". The code in "sketch.js" is as follows:

```
1 let temperatur;
2
3 function setup() {
4   createCanvas(400, 400);
5   leseDatenVonAPI();
6   setInterval(leseDatenVonAPI, 300000);
7 }
8
9 function draw() {
10  background(220);
11  if(temperatur) {
12    let farbwert = map(temperatur, -20, 50, 0, 255);
13    fill(farbwert,0,128);
14    rect(30,30,340,340);
15
16    fill(240);
17    textStyle(BOLD);
18    textSize(34);
```

On the right, a preview window shows a dark purple rectangle with the text "0.10 Zug" in white.

## WIEDERHOLENDE AUSFÜHRUNGEN

Damit das Programm eine Temperaturänderung registriert, fragen wir alle 5 Minuten die aktuelle Temperatur der senseBox ab. Das geschieht im Code mit `setInterval()`.

```
1 let temperatur;  
2  
3 function setup() {  
4   createCanvas(400, 400);  
5   leseDatenVonAPI();  
6   setInterval(leseDatenVonAPI, 300000);  
7 }  
8
```

**setInterval(leseDatenVonAPI, 300000)**

Setzt einen Timer, der die Funktion **leseDatenVonAPI()** alle 5 Minuten aufruft, um die Sensordaten immer aktuell zu halten. **setInterval()** misst die Zeit in Millisekunden (300000 Millisekunden entsprechen 5 Minuten).

```
16 fill(240);  
17 textStyle(BOLD);  
18 textSize(34);
```

Console Clear ▾

```
► Object {_id: "59a703f2d67eb50011453971", createdAt: "2017-08-30T18:29:06.376Z", updatedAt: "2020-11-11T13:21:51.308Z", name: "Schneefernerkopf", exposure: "outdoor"...}
```



HERZLICHEN GLÜCKWUNSCH!

DU HAST DEIN ERSTES P5.JS

PROGRAMM GESCHRIEBEN!