DATA VISUALIZATION WITH P5.JS

With the flash cards you can recreate your own first prototype step by step.

THE SURFACE OF EDITOR.P5JS.ORG

On editor.p5js.org you get to the editor in which you can start programming directly.

A preview of your program will be shown here when you click the play button

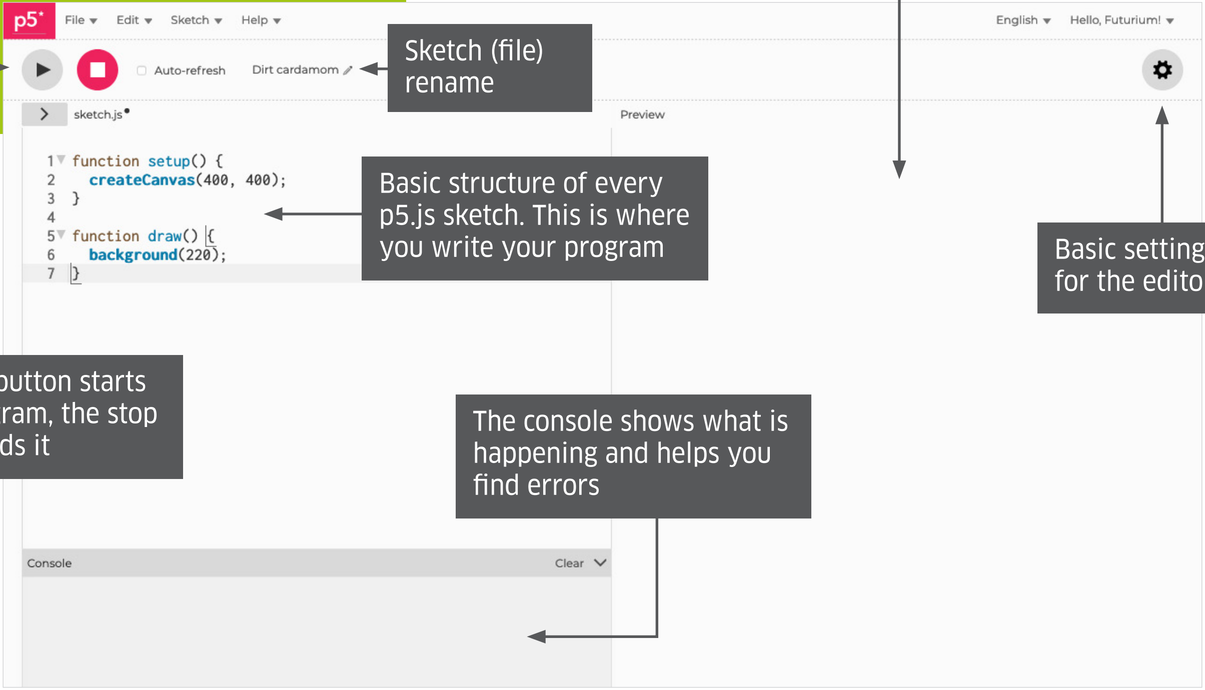
Sketch (file) rename

Basic structure of every p5.js sketch. This is where you write your program

Basic settings for the editor

The play button starts your program, the stop button ends it

The console shows what is happening and helps you find errors

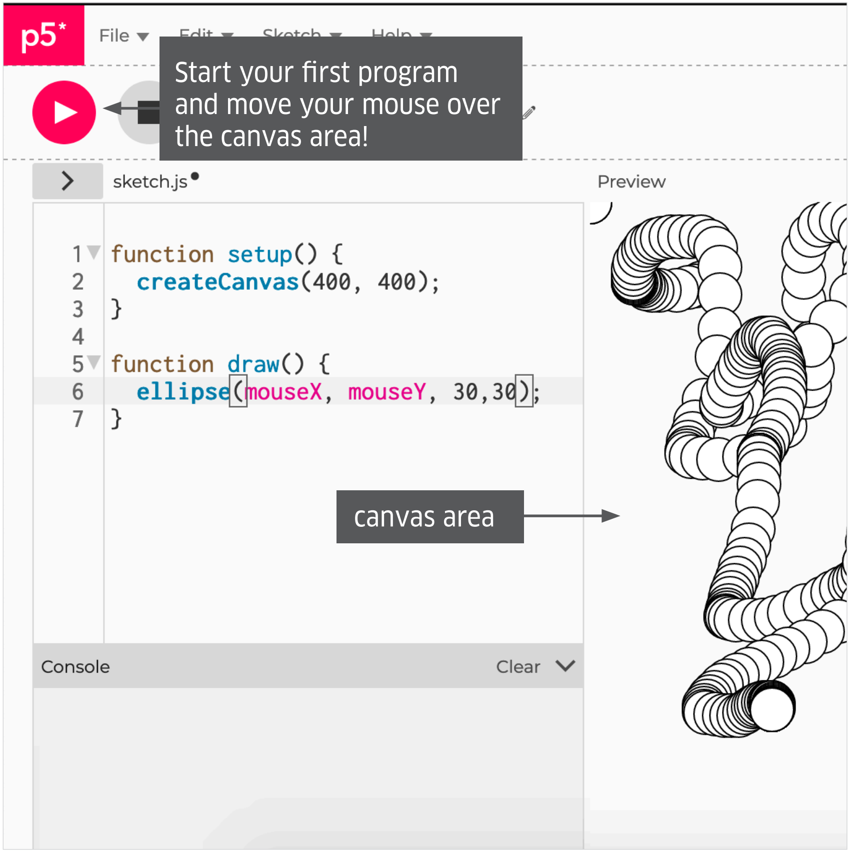


```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
}
```

HELLO ELLIPSE!

Following a small programming tradition, we also want to start with a „Hello World“ program, or more precisely with „Hello Ellipse“.

The idea behind the JavaScript library p5 is based on the open source programming language Processing. It‘s a kind of software sketchbook that makes creative design with programming easy for artists, designers and other people with ideas. You can use it to create graphic and interactive applications for the web directly in the browser! P5 is open source and was brought to the world by Lauren McCarthy.



Start your first program and move your mouse over the canvas area!

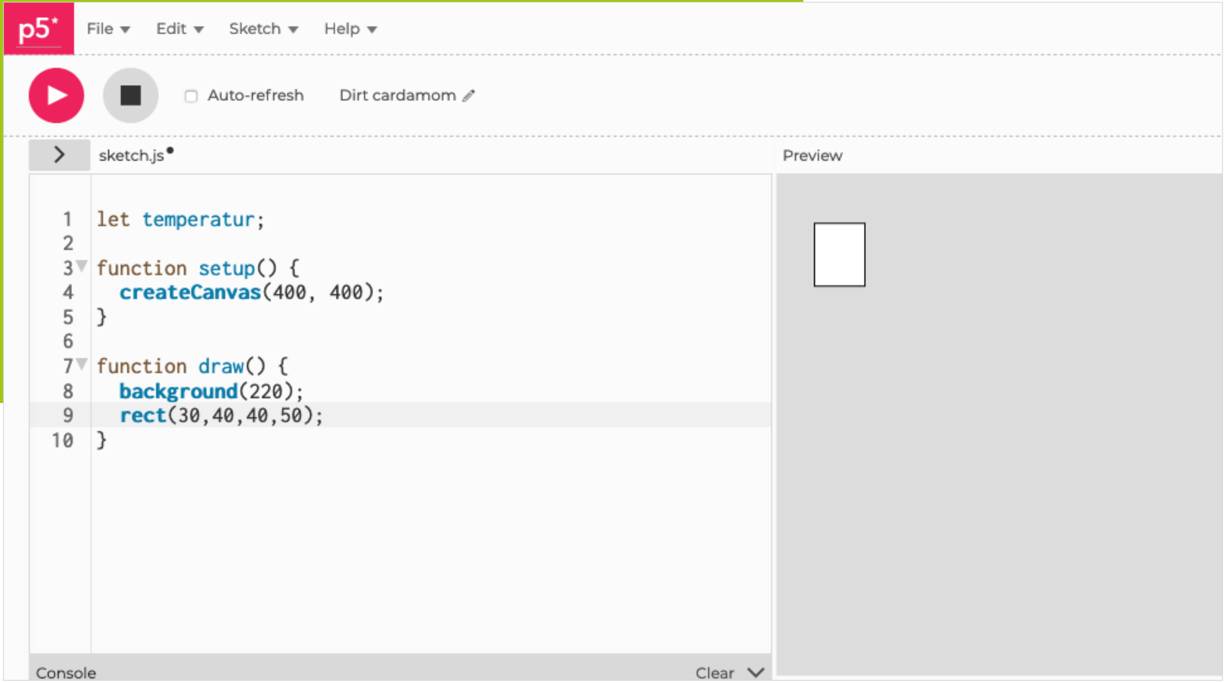
```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  ellipse(mouseX, mouseY, 30,30);
}
```

canvas area

DRAW A RECTANGLE

For the data visualization we want to color a rectangle depending on the temperature, for this we draw a rectangle on our drawing surface at the beginning.

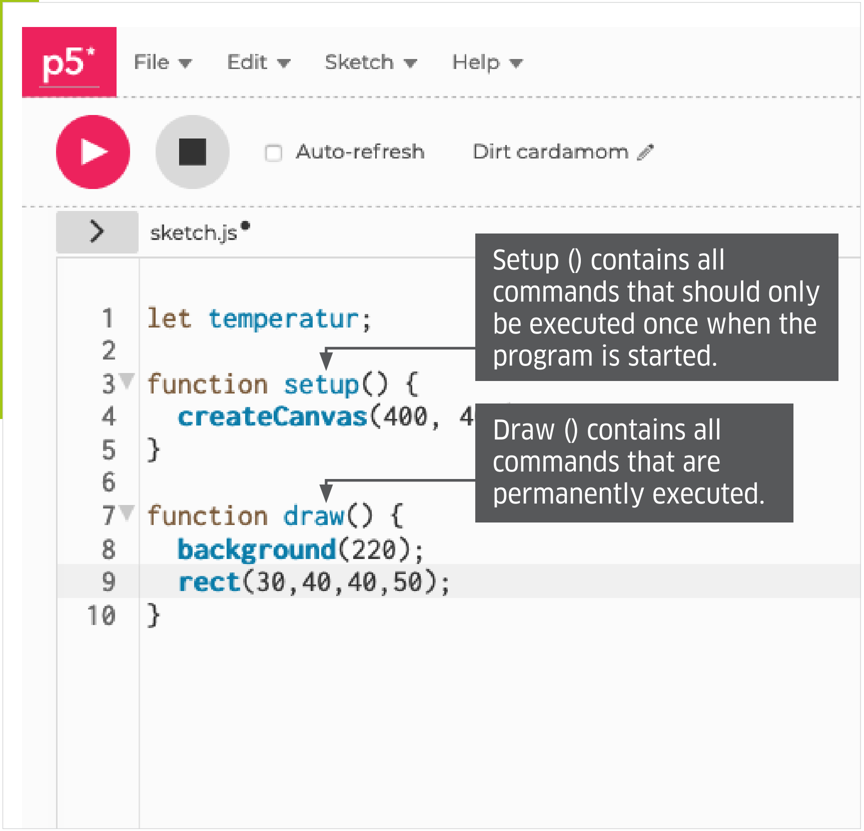
STEP 1



```
let temperatur;

function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
  rect(30,40,40,50);
}
```

SETUP & DRAW FUNCTION

p5.js provides various functions that are called automatically. The two most important are setup and draw. They can contain various commands that are processed step by step. The setup () function is executed once when the program starts, for example createCanvas () only needs to be called once. Then the function draw () repeats itself infinitely often.

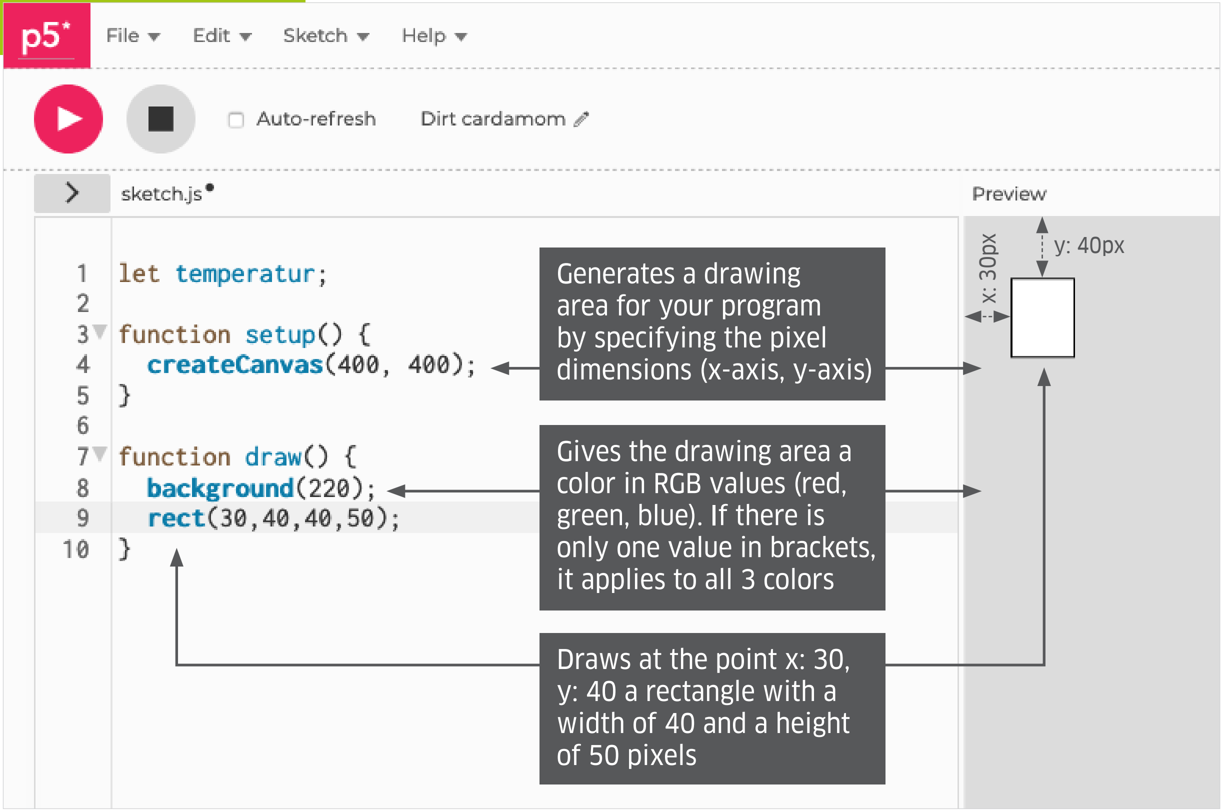


```
let temperatur;

function setup() {
  createCanvas(400, 4
}

function draw() {
  background(220);
  rect(30,40,40,50);
}
```

Setup () contains all commands that should only be executed once when the program is started.

Draw () contains all commands that are permanently executed.

CANVAS, RECT & BACKGROUND



```
let temperatur;

function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
  rect(30,40,40,50);
}
```

Generates a drawing area for your program by specifying the pixel dimensions (x-axis, y-axis)

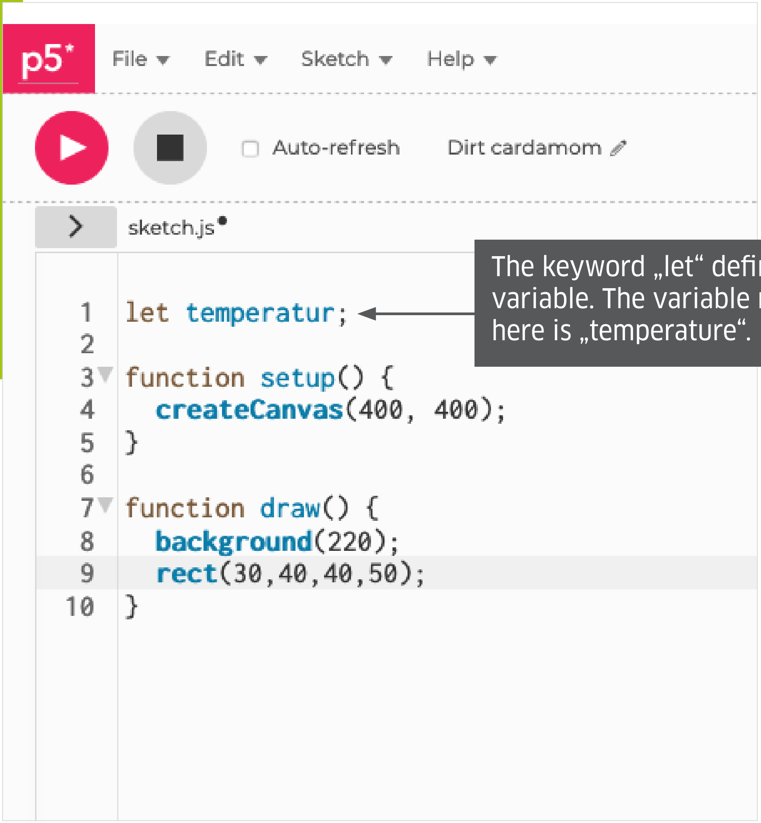
Gives the drawing area a color in RGB values (red, green, blue). If there is only one value in brackets, it applies to all 3 colors

Draws at the point x: 30, y: 40 a rectangle with a width of 40 and a height of 50 pixels

VARIABLES

A variable can store information that can be used anywhere in the code.

You can visualize a variable like a box. The box is the variable, the content the value. The value can change at will. In our example we want to save a temperature value (e.g. 14) of a senseBox in the „temperature“ box.



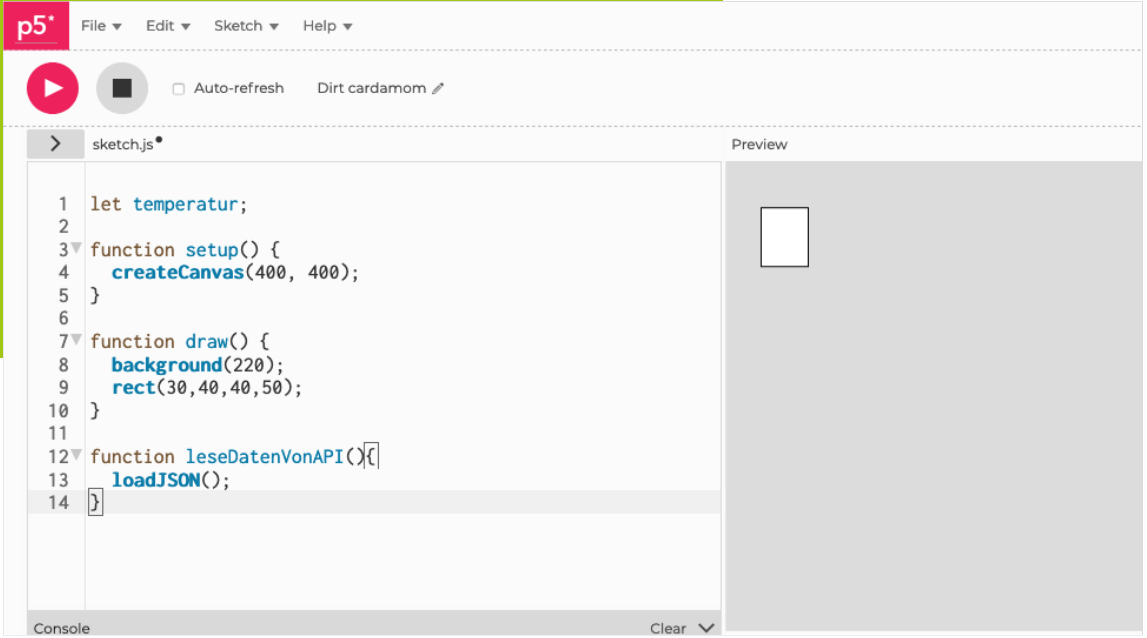
```
let temperatur;

function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
  rect(30,40,40,50);
}
```

The keyword „let“ defines a variable. The variable name here is „temperature“.

LOAD DATA FROM A SENSEBOX

With the LoadJSON function we can load data from a specific senseBox. This function here calls another function called loadJSON. The "loadJSON" function is already provided by p5.js to call up data of the "JSON" file type. You can think of it as a text file with a certain structure in order to access specific elements in the file during programming.

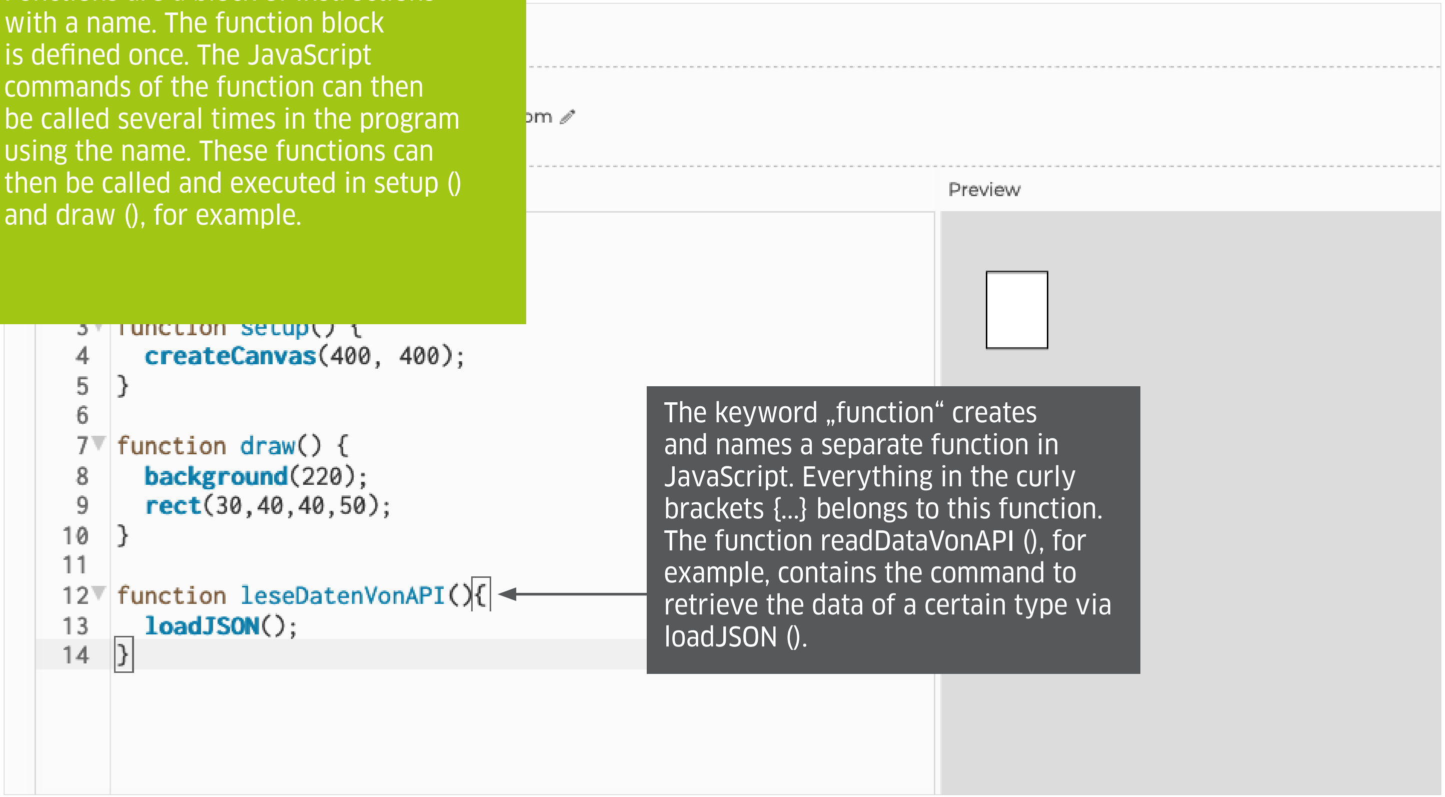
STEP 2



```
let temperatur;

function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
  rect(30,40,40,50);
}

function leseDatenVonAPI(){
  loadJSON();
}
```

OWN FUNCTIONS IN JAVASCRIPT

Functions are a block of instructions with a name. The function block is defined once. The JavaScript commands of the function can then be called several times in the program using the name. These functions can then be called and executed in setup () and draw (), for example.

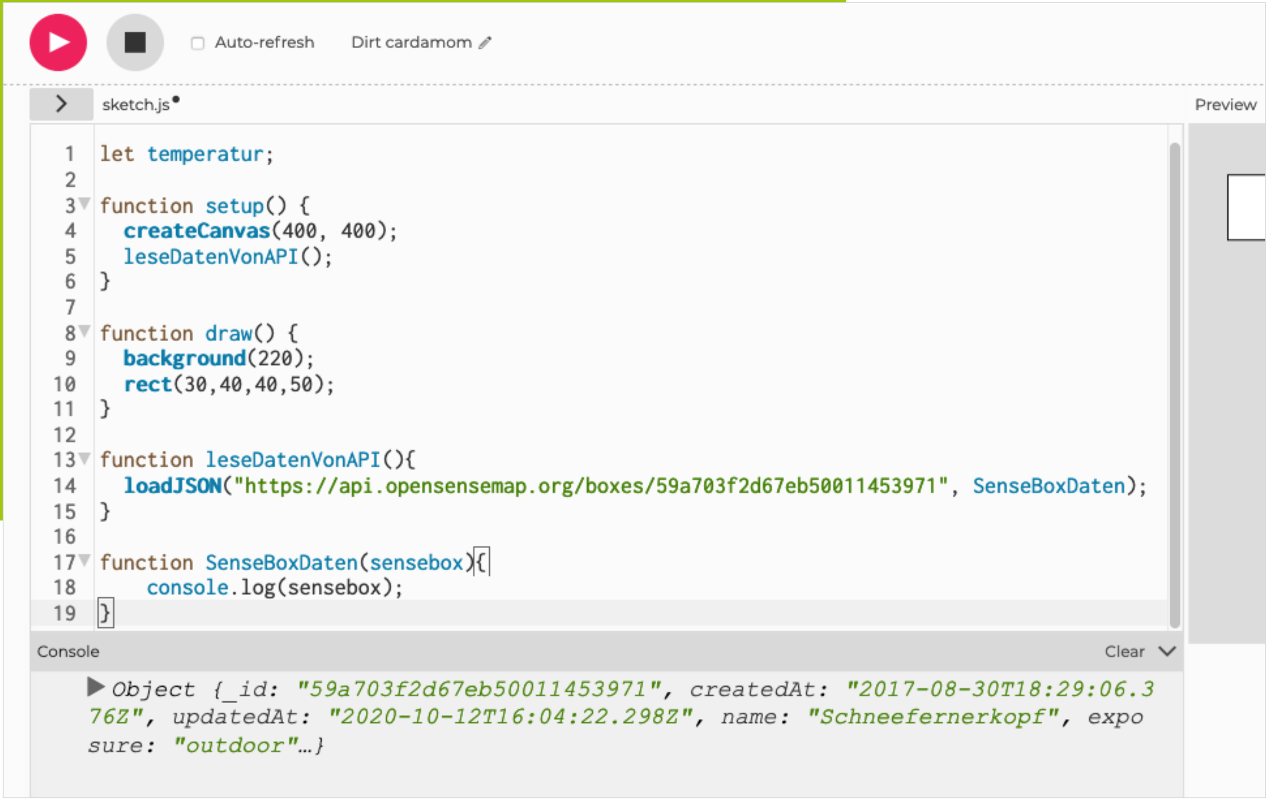


```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
  rect(30,40,40,50);
}

function leseDatenVonAPI(){
  loadJSON();
}
```

The keyword „function“ creates and names a separate function in JavaScript. Everything in the curly brackets {...} belongs to this function. The function readDataVonAPI (), for example, contains the command to retrieve the data of a certain type via loadJSON ().

OUTPUT DATA FROM A SENSEBOX TO THE CONSOLE

STEP 3

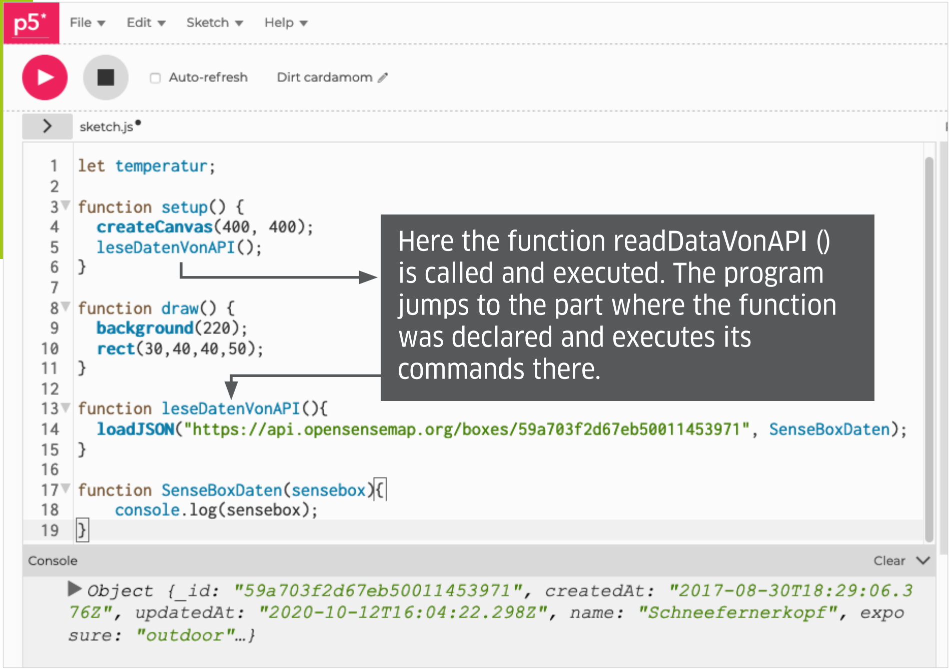


```
let temperatur;

function setup() {
  createCanvas(400, 400);
  leseDatenVonAPI();
}

function draw() {
  background(220);
  rect(30,40,40,50);
}

function leseDatenVonAPI(){
  loadJSON("https://api.opensensemap.org/boxes/59a703f2d67eb50011453971", SenseBoxDaten);
}

function SenseBoxDaten(sensebox){
    console.log(sensebox);
}
```

```
▶Object {_id: "59a703f2d67eb50011453971", createdAt: "2017-08-30T18:29:06.3
76Z", updatedAt: "2020-10-12T16:04:22.298Z", name: "Schneefernerkopf", expo
sure: "outdoor"…}
```

CALL YOUR OWN FUNCTION

The function readDataVonAPI ()
was only declared before. It is only
executed the first time when the
function is called in setup ().
The program jumps to, so to speak
where the function is written, executes
the commands and then jumps back to
set-up (). This was the last command
in setup (), so draw () is executed
afterwards.



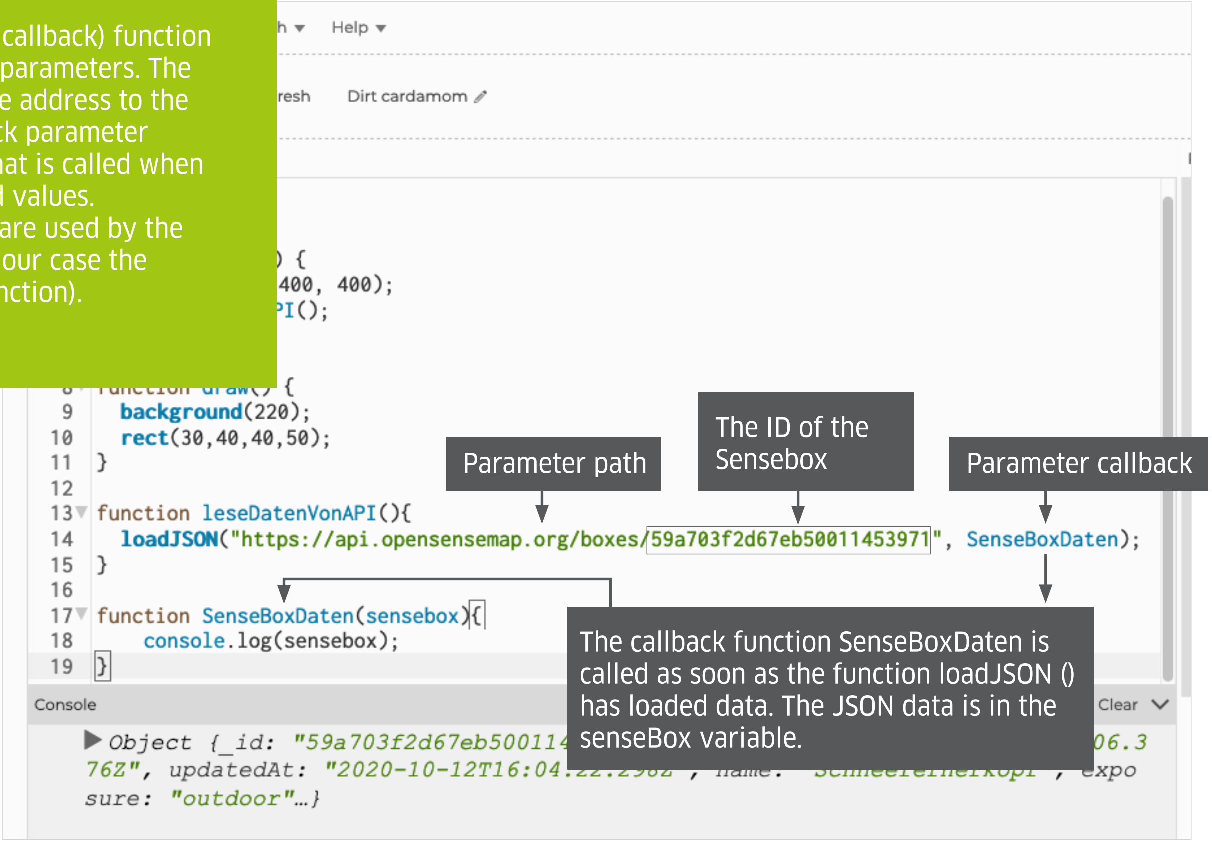
```
let temperatur;

function setup() {
  createCanvas(400, 400);
  leseDatenVonAPI();
}

function draw() {
  background(220);
  rect(30,40,40,50);
}

function leseDatenVonAPI(){
  loadJSON("https://api.opensensemap.org/boxes/59a703f2d67eb50011453971", SenseBoxDaten);
}

function SenseBoxDaten(sensebox){
    console.log(sensebox);
}
```

```
▶ Object {_id: "59a703f2d67eb50011453971", createdAt: "2017-08-30T18:29:06.3
76Z", updatedAt: "2020-10-12T16:04:22.298Z", name: "Schneefernerkopf", expo
sure: "outdoor"…}
```

Here the function readDataVonAPI () is called and executed. The program jumps to the part where the function was declared and executes its commands there.

LOADJSON (PATH,CALLBACK)

The loadJSON (path, callback) function expects at least two parameters. The path parameter is the address to the JSON file, the callback parameter expects a function that is called when loadJSON has loaded values.
The received values are used by the callback function (in our case the SenseBoxDaten () function).



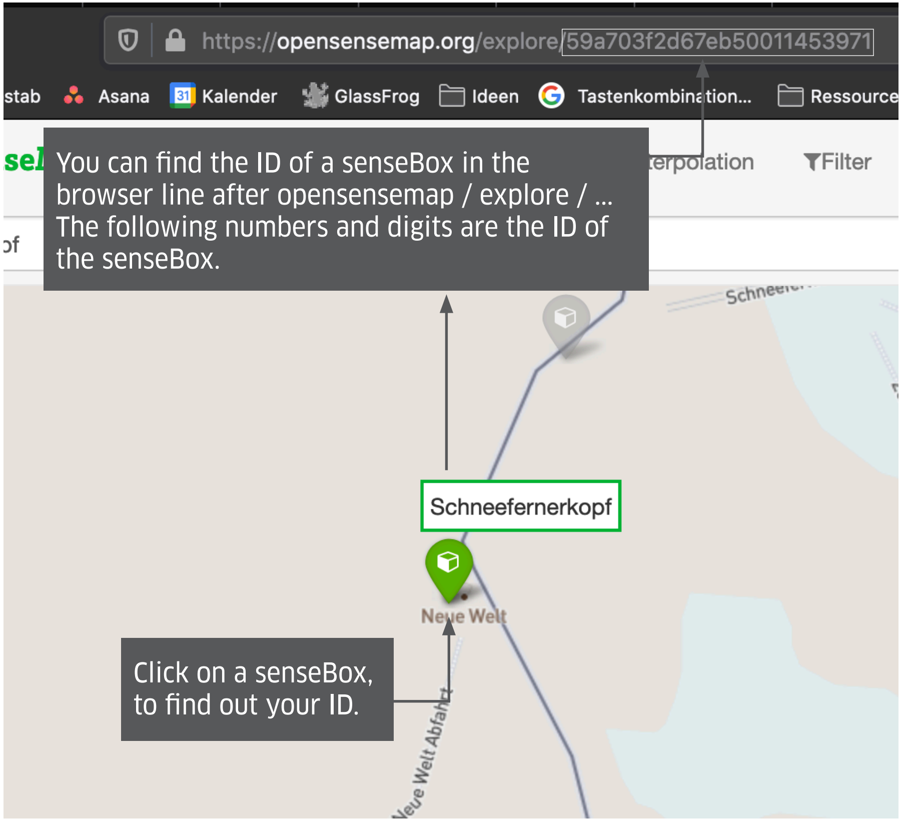
```
) {
400, 400);
PI();

function draw() {
  background(220);
  rect(30,40,40,50);
}

function leseDatenVonAPI(){
  loadJSON("https://api.opensensemap.org/boxes/59a703f2d67eb50011453971", SenseBoxDaten);
}

function SenseBoxDaten(sensebox){
    console.log(sensebox);
}
```

```
▶ Object {_id: "59a703f2d67eb500114                    06.3
76Z", updatedAt: "2020-10-12T16:04:22.296Z", name: "Schneefernerkopf", expo
sure: "outdoor"…}
```

The callback function SenseBoxDaten is called as soon as the function loadJSON () has loaded data. The JSON data is in the senseBox variable.

ID OF A SENSEBOX

You can find many senseBoxes on the openSenseMap.
So that the different data of each senseBox can be sorted and evaluated, each sensoBox needs a number or sequence of numbers that occurs only once or a combination of both. This combination of digits is then the ID (identifier) of a senseBox, which can thus be clearly identified.

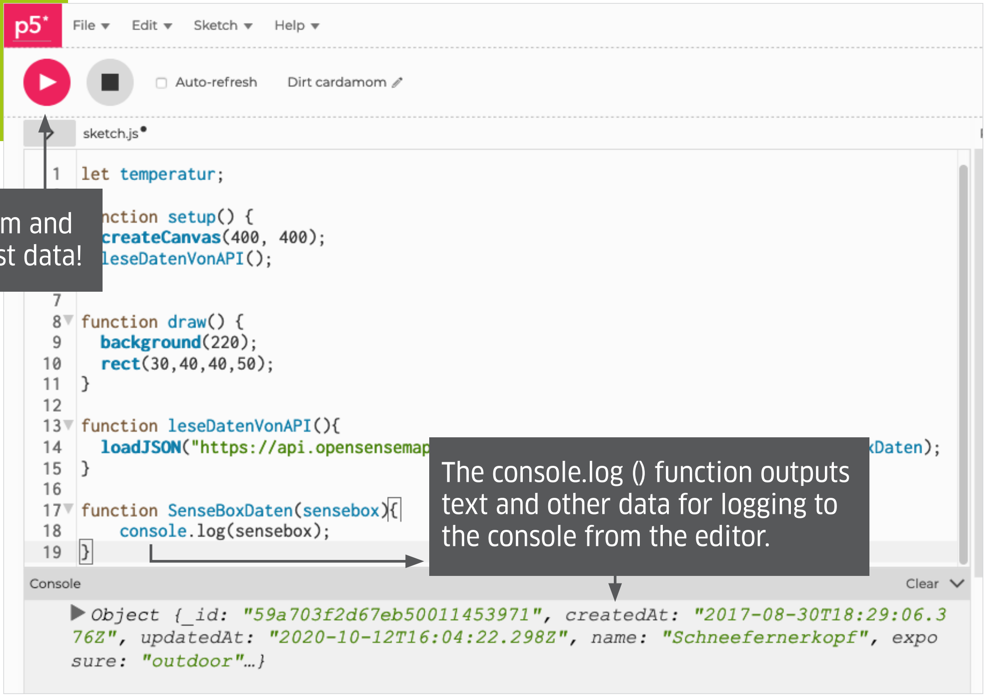


You can find the ID of a senseBox in the browser line after opensensemap / explore / ... The following numbers and digits are the ID of the senseBox.

Click on a senseBox, to find out your ID.

OUTPUT DATA FROM A SENSEBOX TO THE CONSOLE

So that we can see whether we have really received data from a SenseBox, we can output this to the console. Depending on your internet connection, this may take a while.



Start the program and receive your first data!

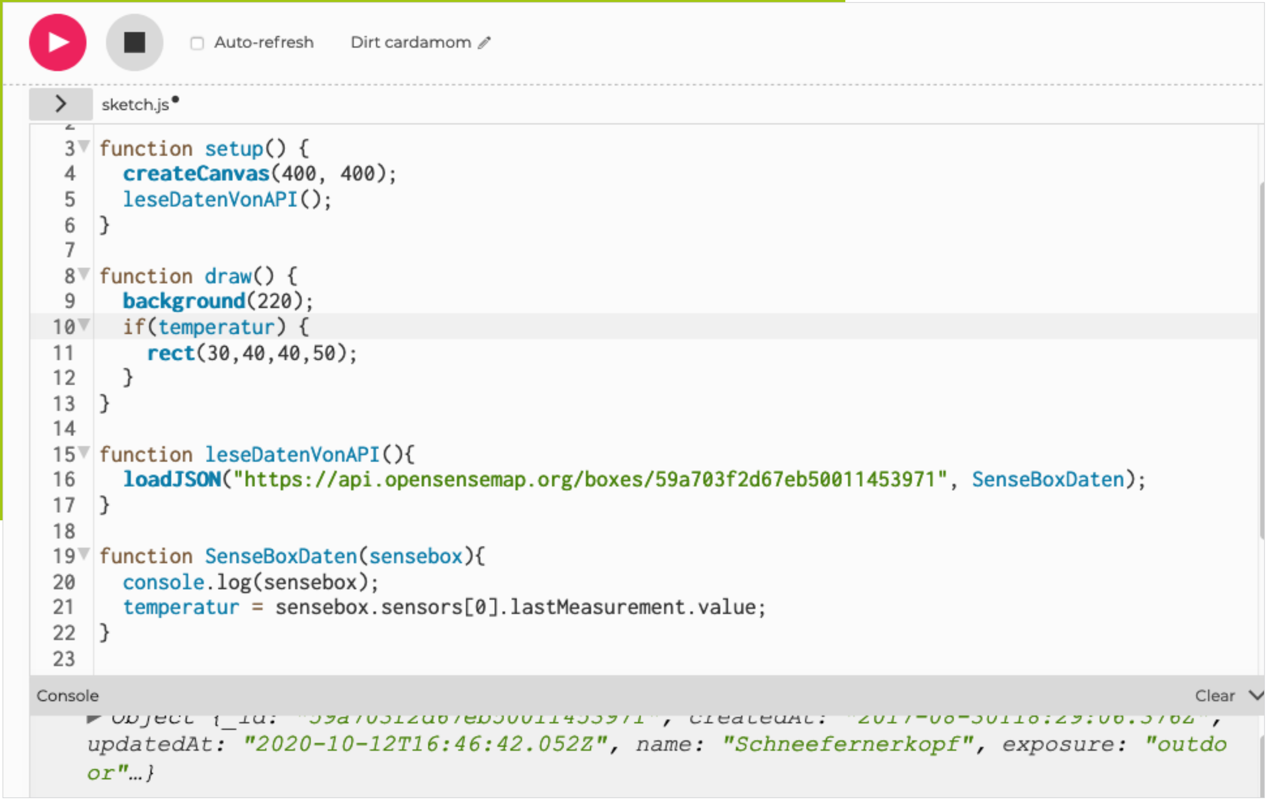
```
let temperatur;

nction setup() {
  createCanvas(400, 400);
  leseDatenVonAPI();


function draw() {
  background(220);
  rect(30,40,40,50);
}

function leseDatenVonAPI(){
  loadJSON("https://api.opensensemap                         kDaten);
}

function SenseBoxDaten(sensebox){
    console.log(sensebox);
}
```

The console.log () function outputs text and other data for logging to the console from the editor.

```
▶ Object {_id: "59a703f2d67eb50011453971", createdAt: "2017-08-30T18:29:06.3
76Z", updatedAt: "2020-10-12T16:04:22.298Z", name: "Schneefernerkopf", expo
sure: "outdoor"…}
```

READ THE TEMPERATURE OF THE SENSEBOX

STEP 4

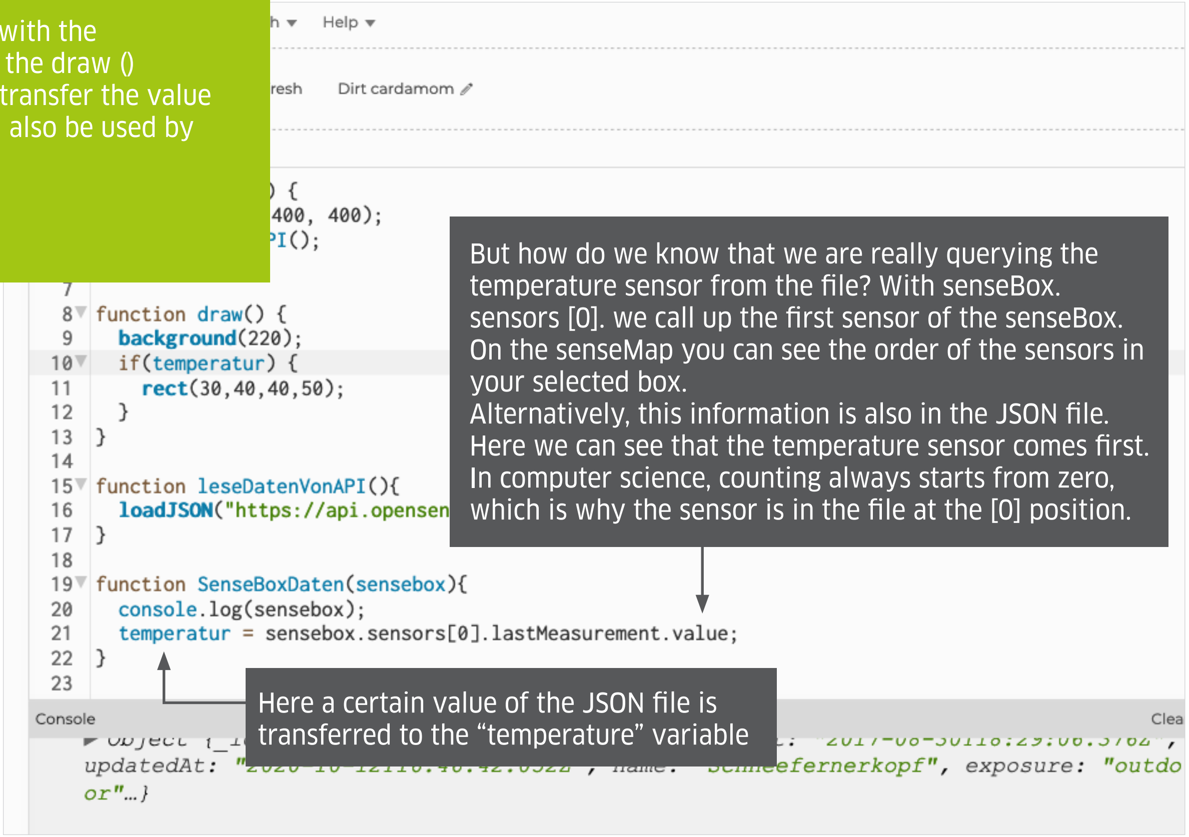


```
function setup() {
  createCanvas(400, 400);
  leseDatenVonAPI();
}

function draw() {
  background(220);
  if(temperatur) {
    rect(30,40,40,50);
  }
}

function leseDatenVonAPI(){
  loadJSON("https://api.opensensemap.org/boxes/59a703f2d67eb50011453971", SenseBoxDaten);
}

function SenseBoxDaten(sensebox){
  console.log(sensebox);
  temperatur = sensebox.sensors[0].lastMeasurement.value;
}

```

```
▶ Object {_id: "59a703f2d67eb50011453971", createdAt: "2017-08-30T18:29:06.376Z",
updatedAt: "2020-10-12T16:46:42.052Z", name: "Schneefernerkopf", exposure: "outdo
or"…}
```

TRANSFER JSON FILES IN A VARIABLE

So that we can work with the temperature value in the draw () function, we have to transfer the value to a variable that can also be used by other functions.



```
) {
400, 400);
PI();

function draw() {
  background(220);
  if(temperatur) {
    rect(30,40,40,50);
  }
}

function leseDatenVonAPI(){
  loadJSON("https://api.opensen
}

function SenseBoxDaten(sensebox){
  console.log(sensebox);
  temperatur = sensebox.sensors[0].lastMeasurement.value;
}
```

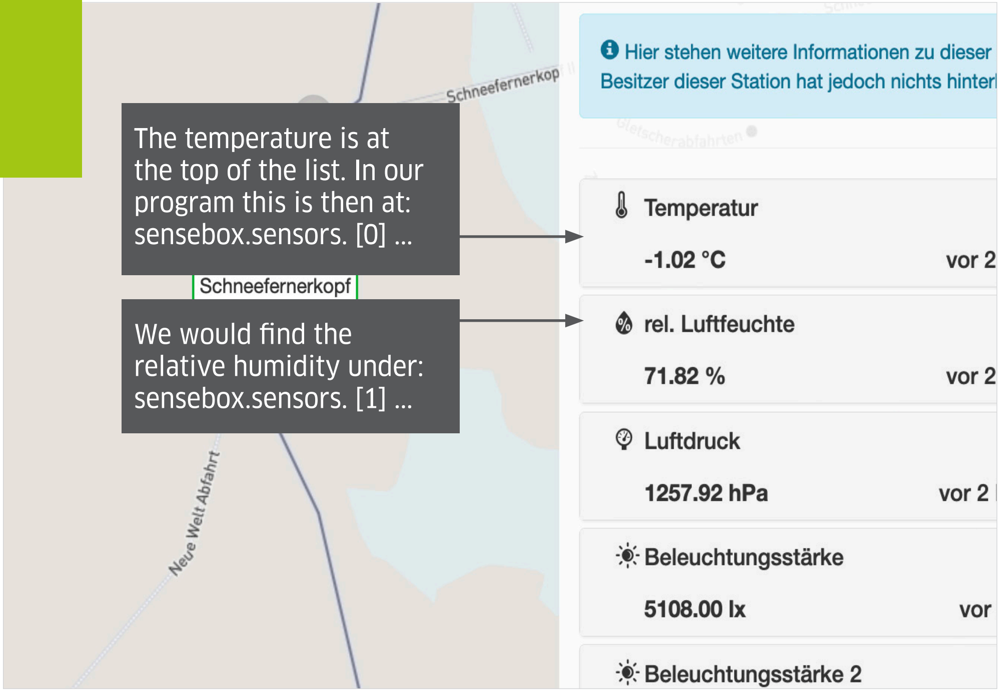
```
updatedAt: "                    ", name:  "Schneefernerkopf", exposure: "outdo
or"…}
```

But how do we know that we are really querying the temperature sensor from the file? With senseBox. sensors [0]. we call up the first sensor of the senseBox. On the senseMap you can see the order of the sensors in your selected box.
Alternatively, this information is also in the JSON file. Here we can see that the temperature sensor comes first. In computer science, counting always starts from zero, which is why the sensor is in the file at the [0] position.

Here a certain value of the JSON file is transferred to the “temperature” variable

FIND A SPECIFIC SENSOR USING THE SENSEMAP

If you are on senseMap select a senseBox, their sensors are in shown in a list. The entries in the list can you are known to count starting with 0. In this way you get the numbers of the individual sensors correlating to the structure in the JSON file.


The temperature is at the top of the list. In our program this is then at: sensebox.sensors. [0] ...

We would find the relative humidity under: sensebox.sensors. [1] ...

Hier stehen weitere Informationen zu dieser
Besitzer dieser Station hat jedoch nichts hinterl

Temperatur
-1.02 °C vor 2

rel. Luftfeuchte
71.82 % vor 2

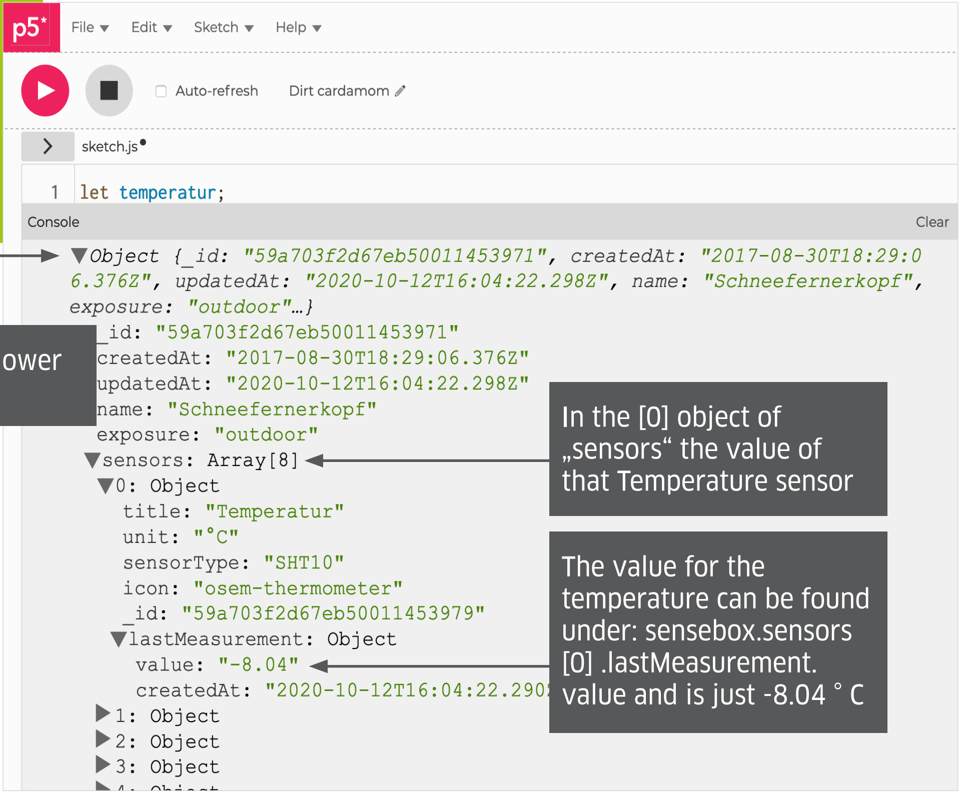
Luftdruck
1257.92 hPa vor 2

Beleuchtungsstärke
5108.00 lx vor

Beleuchtungsstärke 2

FIND A SPECIFIC SENSOR VIA THE JSON FILE

A JSON file can contain different data from a senseBox. So that we can see the data of a sensor, we navigate through the different tabs of the file. Click on the small triangles to see the lower levels.



```
let temperatur;
```

```
▼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: "59a703f2d67eb50011453979"
   ▼lastMeasurement: Object
      value: "-8.04"
      createdAt: "2020-10-12T16:04:22.290
  ▶1: Object
  ▶2: Object
  ▶3: Object
```

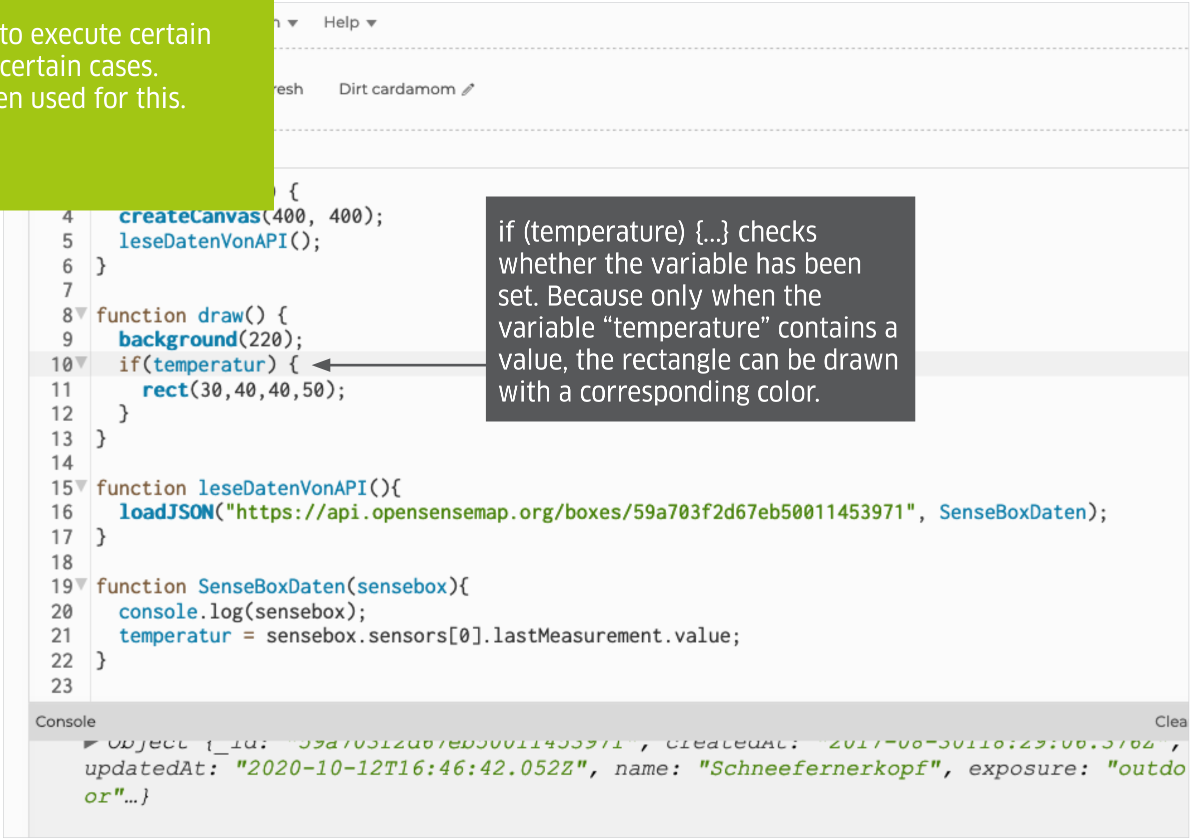
Opens the lower ones levels

In the [0] object of „sensors“ the value of that Temperature sensor

The value for the temperature can be found under: sensebox.sensors [0] .lastMeasurement. value and is just -8.04 ° C

IF-CONDITIONS

It is often necessary to execute certain lines of code only in certain cases.
If statements are often used for this.



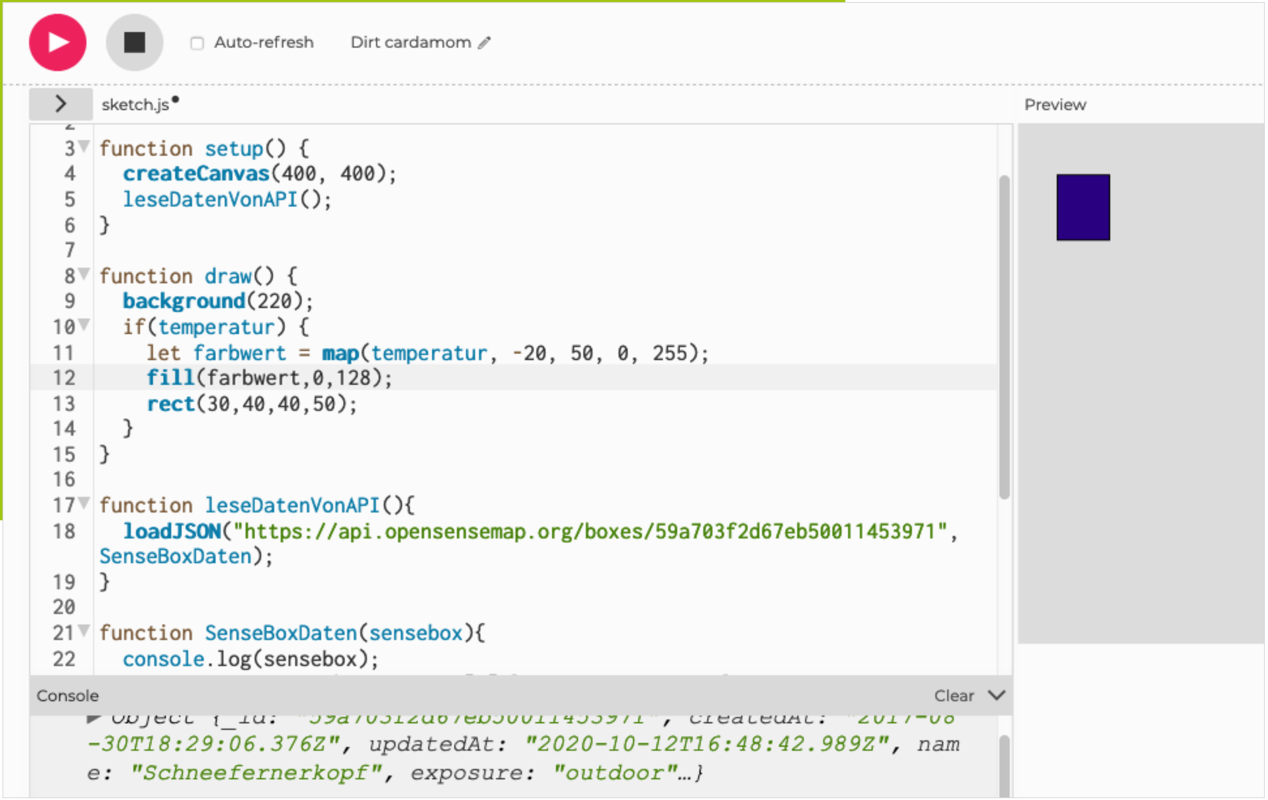
```
    createCanvas(400, 400);
    leseDatenVonAPI();
}

function draw() {
    background(220);
    if(temperatur) {
      rect(30,40,40,50);
    }
}

function leseDatenVonAPI(){
    loadJSON("https://api.opensensemap.org/boxes/59a703f2d67eb50011453971", SenseBoxDaten);
}

function SenseBoxDaten(sensebox){
    console.log(sensebox);
    temperatur = sensebox.sensors[0].lastMeasurement.value;
}
```

```
▶ Object {_id: "59a703f2d67eb50011453971", createdAt: "2017-08-30T18:29:06.376Z",
updatedAt: "2020-10-12T16:46:42.052Z", name: "Schneefernerkopf", exposure: "outdo
or"…}
```

if (temperature) {...} checks whether the variable has been set. Because only when the variable "temperature" contains a value, the rectangle can be drawn with a corresponding color.

COLOR THE RECTANGLE

STEP 5

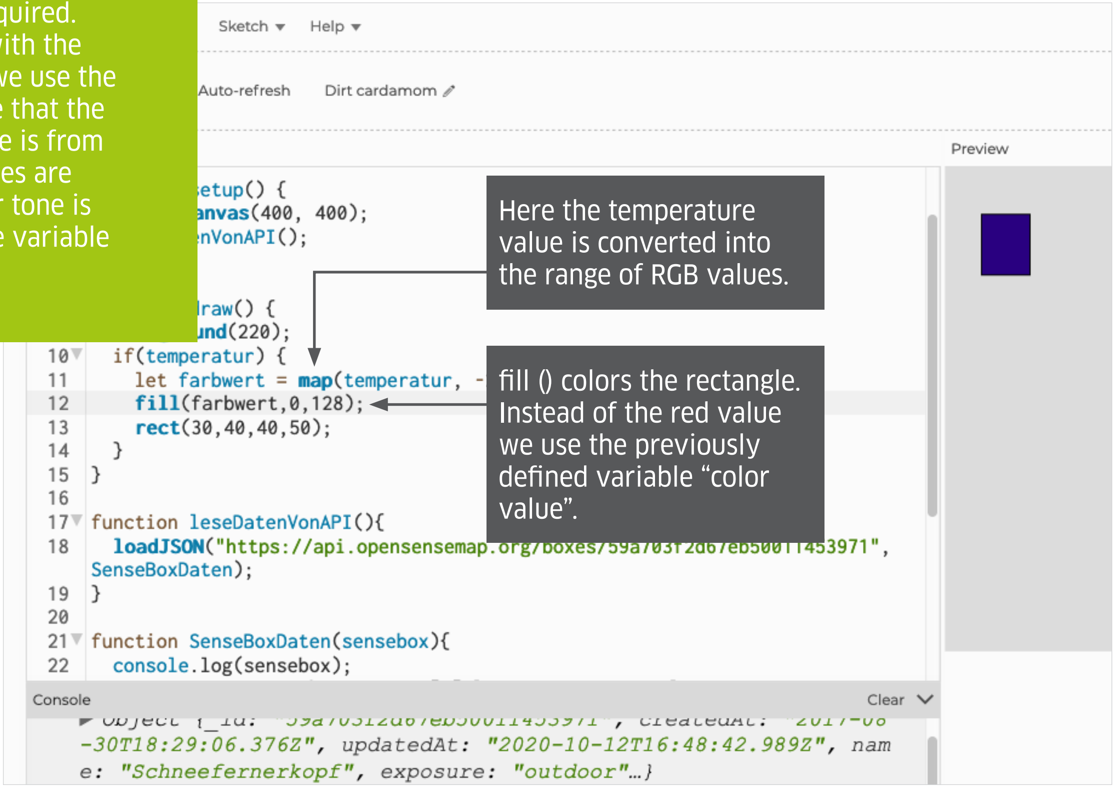


```
function setup() {
  createCanvas(400, 400);
  leseDatenVonAPI();
}

function draw() {
  background(220);
  if(temperatur) {
    let farbwert = map(temperatur, -20, 50, 0, 255);
    fill(farbwert,0,128);
    rect(30,40,40,50);
  }
}

function leseDatenVonAPI(){
  loadJSON("https://api.opensensemap.org/boxes/59a703f2d67eb50011453971",
SenseBoxDaten);
}

function SenseBoxDaten(sensebox){
  console.log(sensebox);
```

```
-30T18:29:06.376Z", updatedAt: "2020-10-12T16:48:42.989Z", nam
e: "Schneefernerkopf", exposure: "outdoor"…}
```

VARIABLE FOR THE COLOR VALUE

To color the rectangle, RGB values in a range from 0 to 255 are required. So that the color changes with the temperature, do the math we use the map () function. We assume that the temperature value im Range is from -20 and 50 count. RGB values are between 0 and 255. A color tone is calculated and stored in the variable ‚color value'.



```
etup() {
anvas(400, 400);
nVonAPI();

lraw() {
und(220);
  if(temperatur) {
    let farbwert = map(temperatur, -
    fill(farbwert,0,128);
    rect(30,40,40,50);
  }
}

function leseDatenVonAPI(){
  loadJSON("https://api.opensensemap.org/boxes/59a703f2d67eb50011453971",
SenseBoxDaten);
}

function SenseBoxDaten(sensebox){
  console.log(sensebox);
```

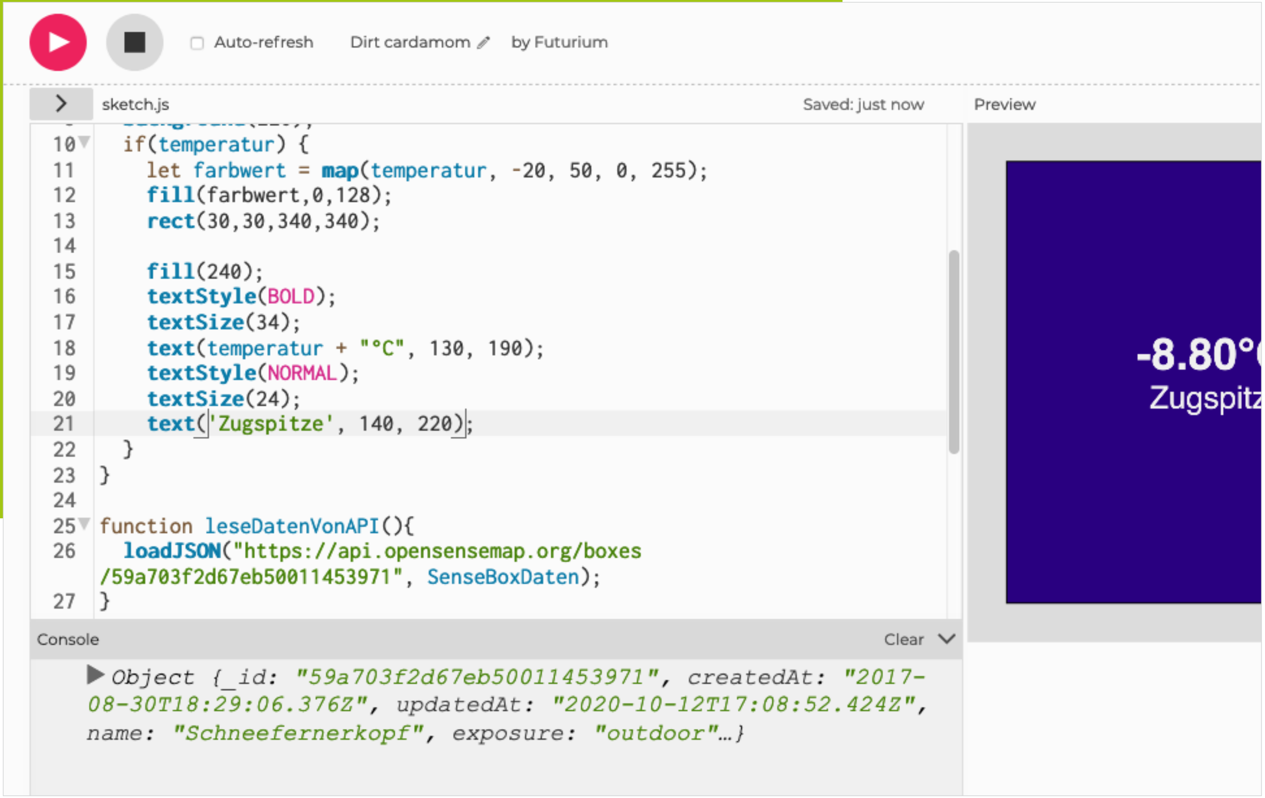
```
▶ Object {_id: "59a703f2d67eb50011453971", createdAt: "2017-08
-30T18:29:06.376Z", updatedAt: "2020-10-12T16:48:42.989Z", nam
e: "Schneefernerkopf", exposure: "outdoor"…}
```

Here the temperature value is converted into the range of RGB values.

fill () colors the rectangle. Instead of the red value we use the previously defined variable "color value".

ADJUST THE RECTANGLE AND SET THE TEXT

STEP 6

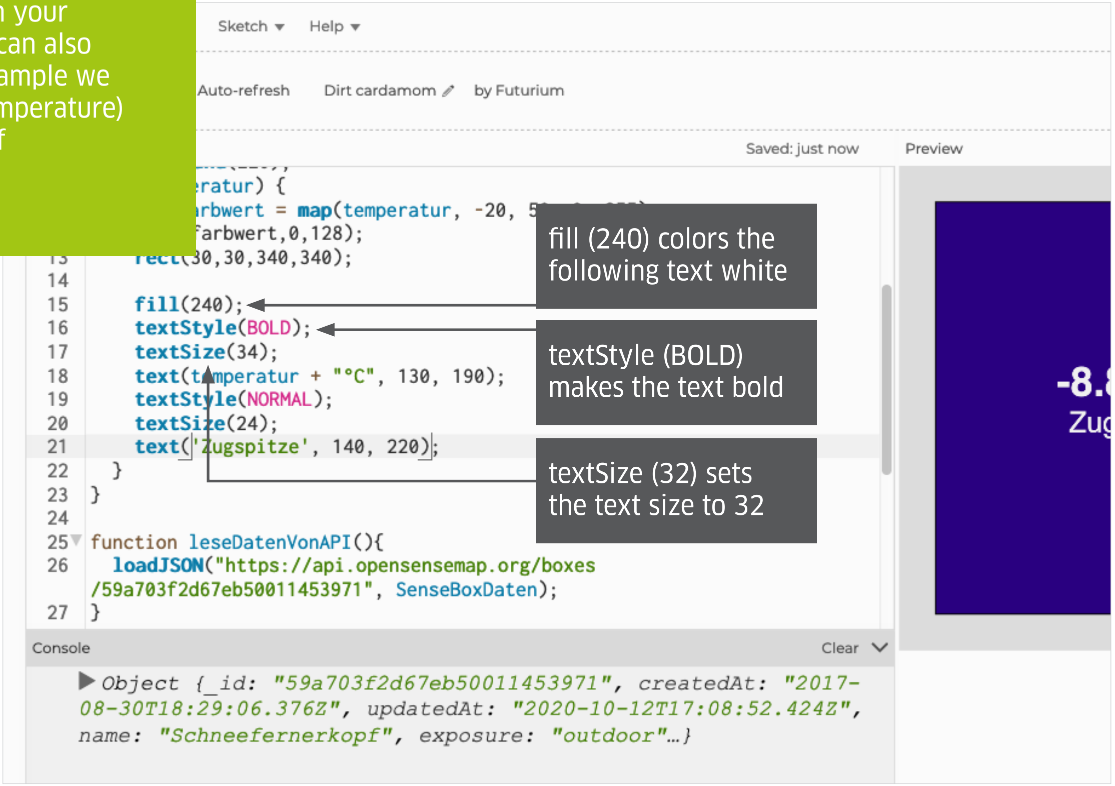


```
  if(temperatur) {
    let farbwert = map(temperatur, -20, 50, 0, 255);
    fill(farbwert,0,128);
    rect(30,30,340,340);

    fill(240);
    textStyle(BOLD);
    textSize(34);
    text(temperatur + "°C", 130, 190);
    textStyle(NORMAL);
    textSize(24);
    text('Zugspitze', 140, 220);
  }
}

function leseDatenVonAPI(){
  loadJSON("https://api.opensensemap.org/boxes
/59a703f2d67eb50011453971", SenseBoxDaten);
}
```

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

DISPLAY THE TEMPERATURE

The function text („Text“, x, y) writes text at the position (x, y) on your drawing surface. Variables can also be output as text. In our example we output the temperature (temperature) together (+) with the unit of measurement ° C („° C“).



```
ratur) {
arbwert = map(temperatur, -20, 5
farbwert,0,128);
    rect(30,30,340,340);

    fill(240);
    textStyle(BOLD);
    textSize(34);
    text(temperatur + "°C", 130, 190);
    textStyle(NORMAL);
    textSize(24);
    text('Zugspitze', 140, 220);
  }
}

function leseDatenVonAPI(){
  loadJSON("https://api.opensensemap.org/boxes
/59a703f2d67eb50011453971", SenseBoxDaten);
}
```

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

fill (240) colors the following text white

textStyle (BOLD) makes the text bold

textSize (32) sets the text size to 32

CONGRATULATIONS

You wrote your first p5.js program.