Malen mit Fourier

Mit mathematischen Funktionen und Processing/p5.js Figuren zeichnen

Vorstellen

InfoLabSaar

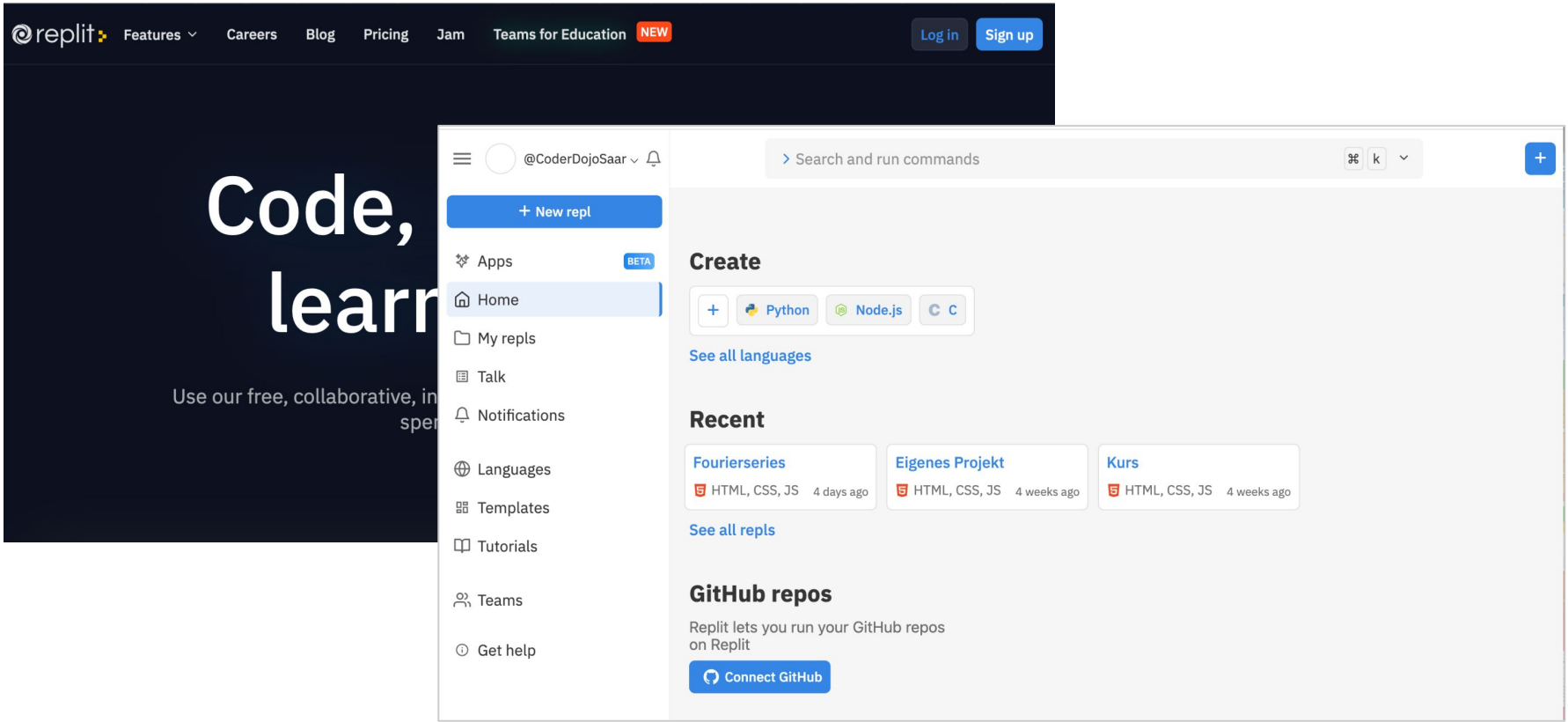
Wer sind wir?

Wer seid ihr?

Was wir heute machen

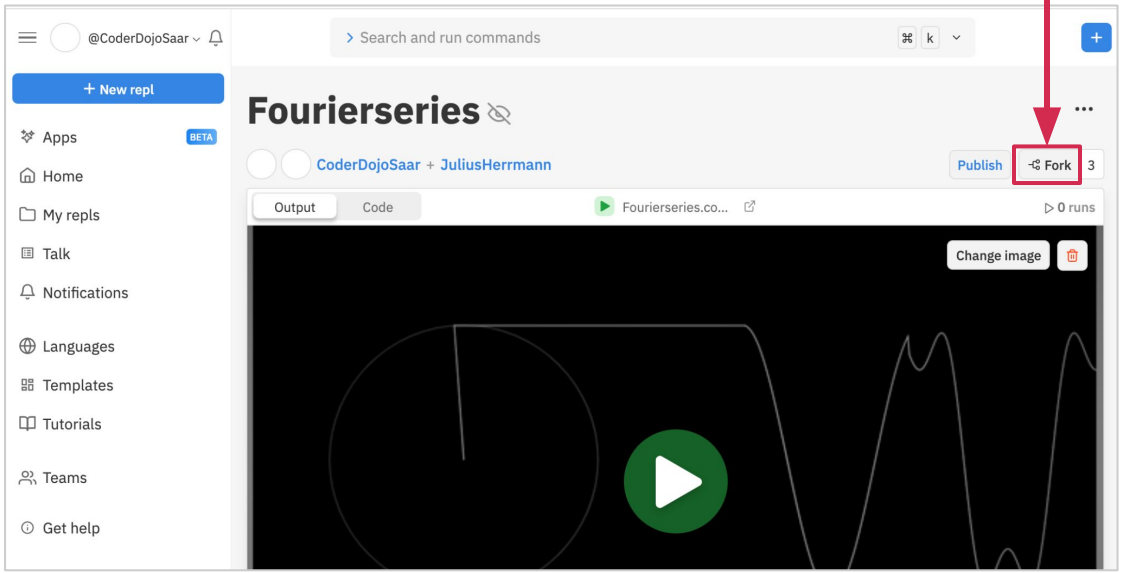
1. Vorstellen
2. Bei repl.it anmelden
3. Animation einer Fourierreihe forken, ausprobieren + anschauen
4. Signalquelle für den Algorithmus hinzufügen
5. **Diskrete Fouriertransformation (DFT) implementieren**
6. Zwei Signale gleichzeitig als Quelle nutzen
   -> für den X-Wert -> für den Y-Wert
7. Alles hübsch auf dem Canvas anordnen und die Signale in der Quelle so setzen, dass eine Figur gezeichnet wird.

Anmelden bei repl.it

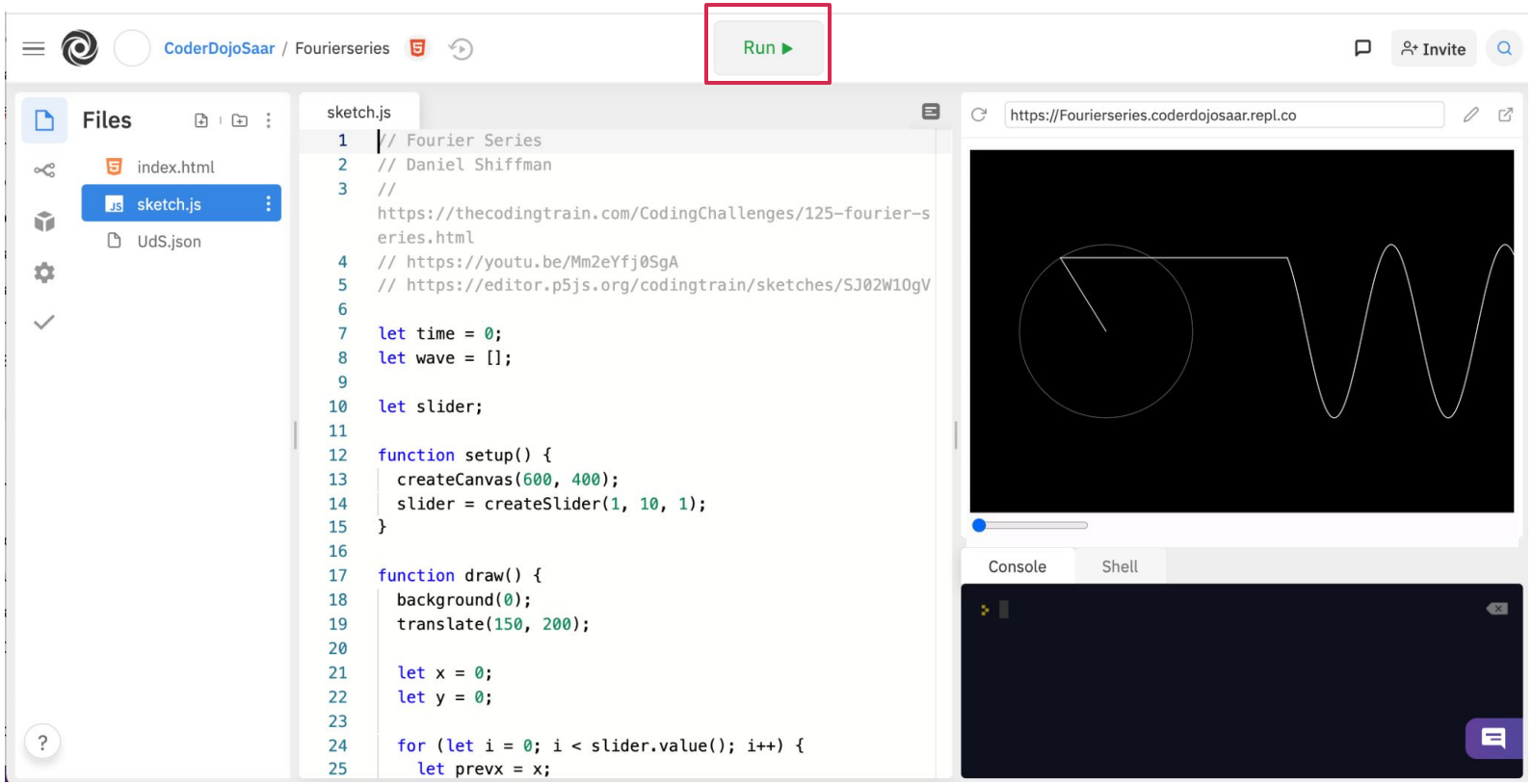


Fourierreihen-Animation forken

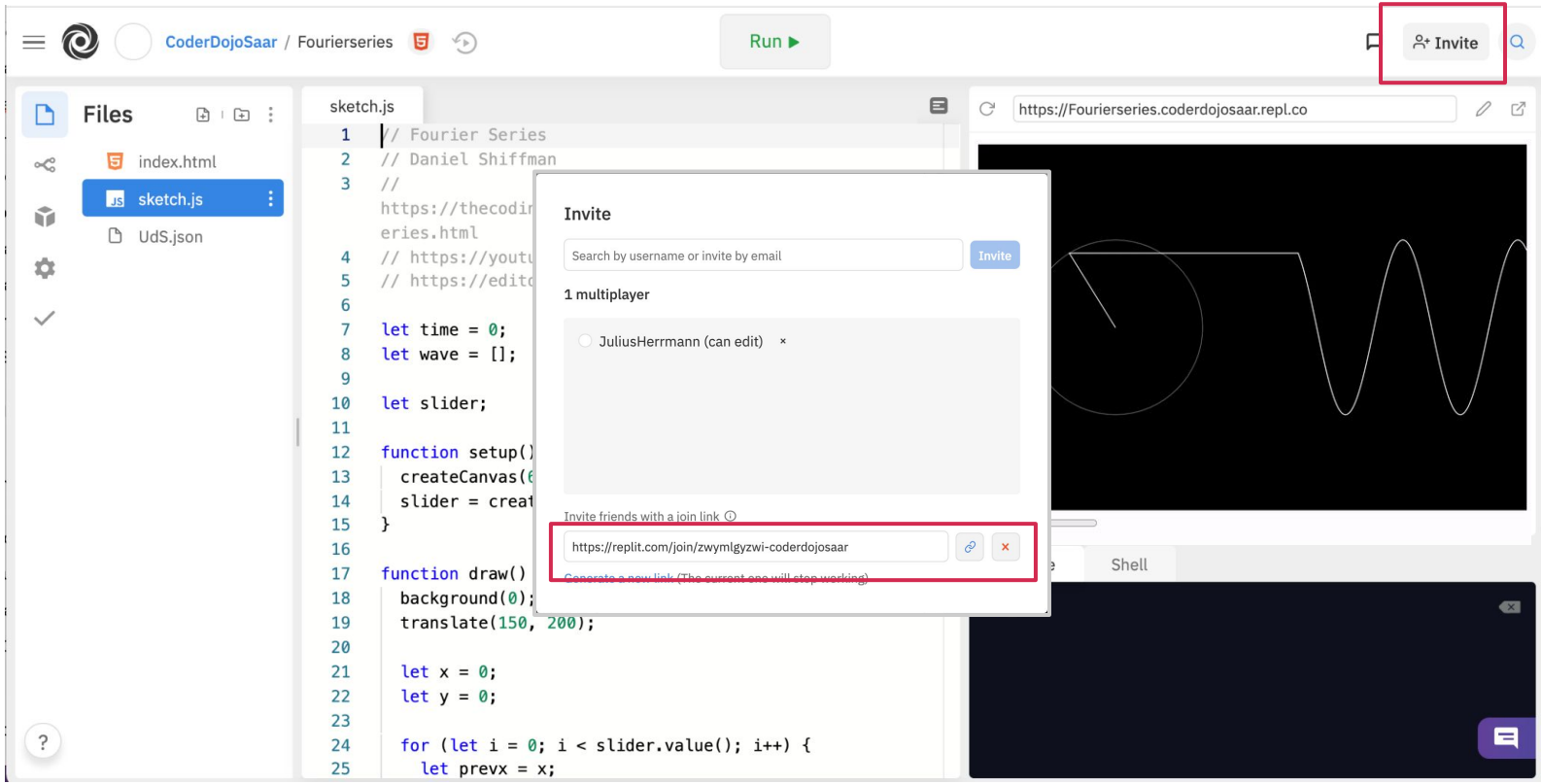
1. Angemeldet bei repl.it
2. Fourierreihen-Animation finden:
   https://replit.com/@CoderDojoSaar/Fourierseries?v=1
3. Animation forken



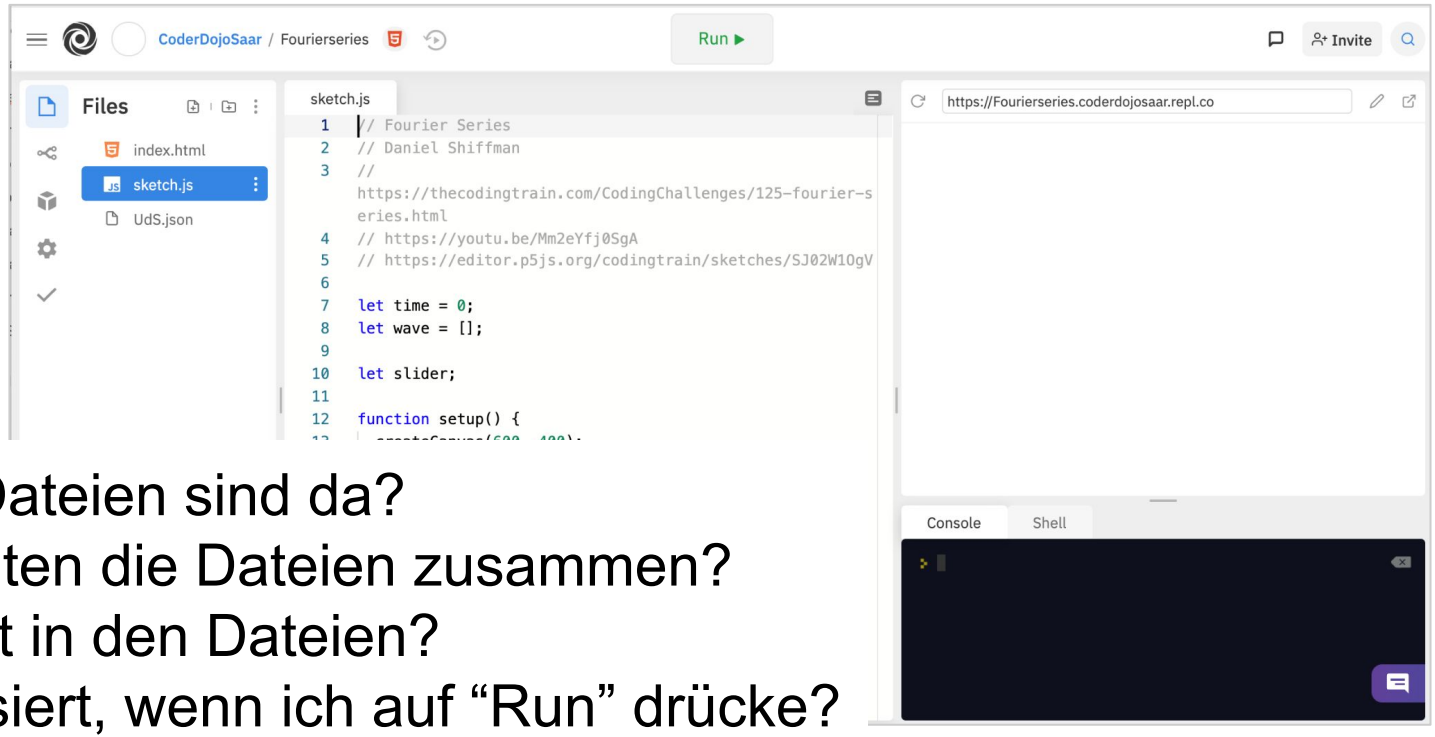
Fourierreihen-Animation ausführen



Repl teilen


CoderDojoSaar / Fourierseries
Run
Invite
Files
index.html
sketch.js
UdS.json
sketch.js
// Fourier Series
// Daniel Shiffman
let time = 0;
let wave = [];
let slider;
function setup()
createCanvas(
slider = creat
}
function draw()
background(0);
translate(150, 200);
let x = 0;
let y = 0;
for (let i = 0; i < slider.value(); i++) {
let prevx = x;
Invite
Search by username or invite by email
Invite
1 multiplayer
JuliusHerrmann (can edit)
Invite friends with a join link
https://replit.com/join/zwymlgyzwi-coderdojosaar
https://Fourierseries.coderdojosaar.repl.co
Shell

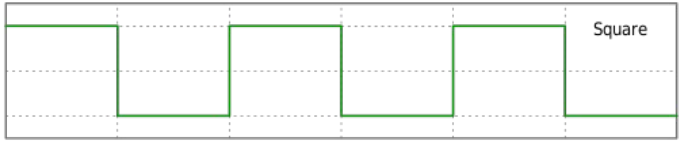
Fourierreihen-Animation anschauen

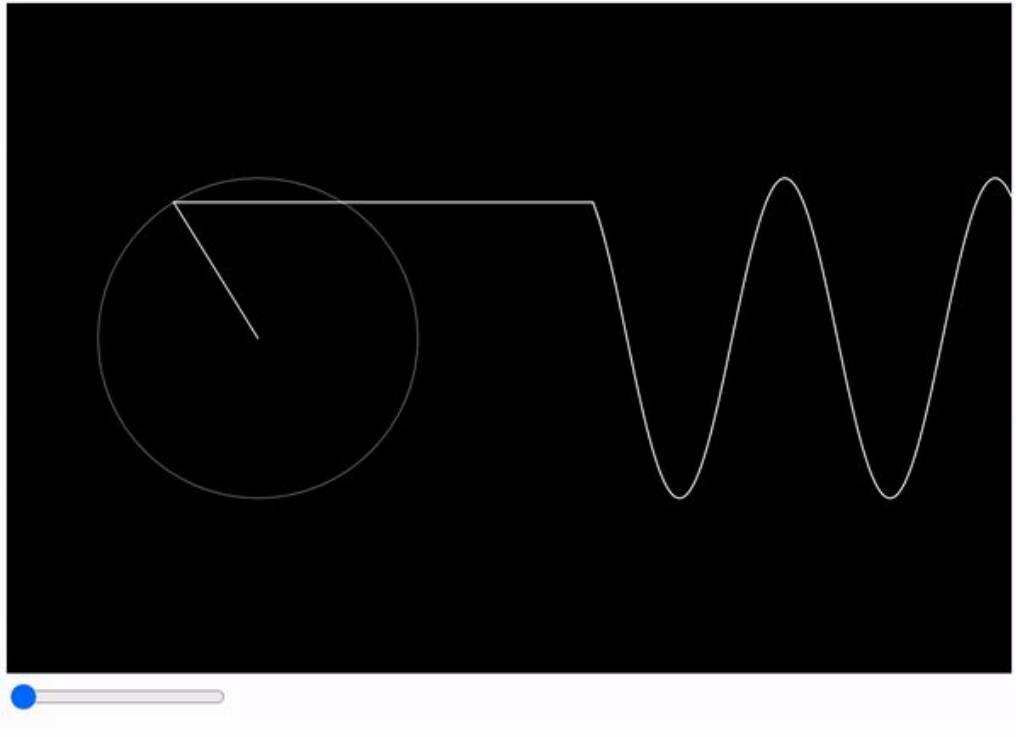


1. Welche Dateien sind da?
2. Wie arbeiten die Dateien zusammen?
3. Was steht in den Dateien?
4. Was passiert, wenn ich auf “Run” drücke?
5. Was passiert bei “Open in a new tab” ?

Animation der Fourierreihe

- Fourierreihe = Überlagerung von Sinus- und Kosinusfunktionen
- Ziel = Annäherung an eine periodischen Funktion
- In der Animation: Annäherung an eine Rechteckschwingung





replit.com/@CoderDojoSaar/Fourierseries?v=1

thecodingtrain.com/CodingChallenges/125-fourier-series.html

Was ist Processing? Was ist p5.js?

InfoLabSaar

- **Processing** ist eine anfängerfreundliche Programmiersprache + IDE von 2001, die auf Grafik, Simulation und Animation spezialisiert ist.
- **p5.js** ist eine Javascript-Bibliothek von 2013. p5.js ist eine Interpretation von Porcessing für das Internet.
- Das Ziel von Processing/p5.js ist **“creative coding”**.

p5.js

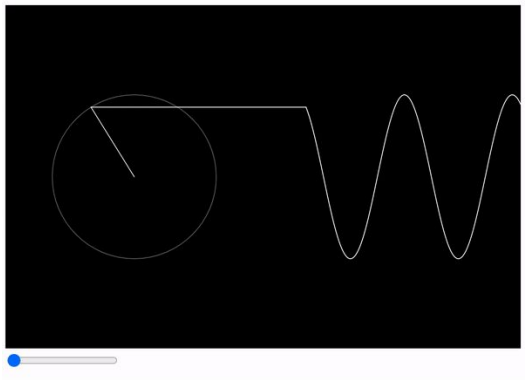
Animation der Fourierreihe in Processing/p5.js

```
let time = 0;
let wave = [];

let slider;

function setup() {
  createCanvas(600, 400);
  slider = createSlider(1, 10, 1);
}

```



```
function draw() {
  background(0);
  translate(150, 200);

  let x = 0;
  let y = 0;

  for (let i = 0; i < slider.value(); i++) {
    let prevx = x;
    let prevy = y;

    let n = i * 2 + 1;
    let radius = 75 * (4 / (n * PI));
    x += radius * cos(n * time);
    y += radius * sin(n * time);

    stroke(255, 100);
    noFill();
    ellipse(prevx, prevy, radius * 2);

    //fill(255);
    stroke(255);
    line(prevx, prevy, x, y);
    //ellipse(x, y, 8);
  }
  wave.unshift(y);
```

```

  translate(200, 0);
  line(x - 200, y, 0, wave[0]);
  beginShape();
  noFill();
  for (let i = 0; i < wave.length; i++) {
    vertex(i, wave[i]);
  }
  endShape();

  time += 0.05;

  if (wave.length > 250) {
    wave.pop();
  }
}
```

https://replit.com/@CoderDojoSaar/Fourierseries?v=1

1. Slider entfernen

```
11  let slider;
12
13  function setup() {
14    createCanvas(600, 400);
15    slider = createSlider(1, 10, 1);
16  }
17
18  function draw() {
19    background(0);
20    translate(150, 200);
21
22    let x = 0;
23    let y = 0;
24
25    for (let i = 0; i < slider.value(); i++) {
```

```
11  function setup() {
12    createCanvas(600, 400);
13  }
14
15  function draw() {
16    background(0);
17    translate(150, 200);
18
19    let x = 0;
20    let y = 0;
21
22    for (let i = 0; i < 5; i++) {
```

2. Arrays für Eingabesignal und Fourierergebnisse

```
4   let y = []; // Eingabewerte
5   let fourierY; // berechnete Fourierwerte
6
7   let time = 0;
8   let wave = [];
9
10  function setup() {
11    createCanvas(600, 400);
12    fourierY = dft(y); // Funktion zur Berechnung der diskreten Fouriertransformation
13  }
```

3. Eingabewerte hardcodieren

```
10  function setup() {
11    createCanvas(600, 400);
12    y = [100, 100, 100, -100, -100,  -100, 100, 100, 100, -100, -100,  100];
13
14    fourierY = dft(y); // Funktion zur Berechnung der diskreten Fouriertransformation
15  }
```

4. Iteration über berechnete Werte

```
24    for (let i = 0; i < fourierY.length; i++) {
25      let prevx = x;
26      let prevy = y;
```

5. Zeichnen erstmal auskommentieren

InfoLabSaar

```
23    for (let i = 0; i < fourierY.length; i++) {
24      let prevx = x;
25      let prevy = y;
26
27    //    let radius = 75 * (4 / (n * PI));
28    //    x += radius * cos(n * time);
29    //    y += radius * sin(n * time);
30    //
31    //    stroke(255, 100);
32    //    noFill();
33    //    ellipse(prevx, prevy, radius * 2);
34    //
35    //    stroke(255);
36    //    line(prevx, prevy, x, y);
37    }
```

6. dft schreiben

```
56  function dft (vals) {
57      return null;
58  }
```

Die Diskrete Fouriertransformation

Fouriertransformation (Ausgabe)

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i \cdot \frac{2\pi}{N} kn}$$

$$e^{ix} = \cos x + i \sin x$$

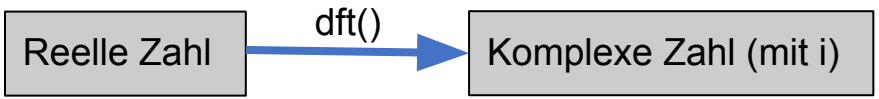
Eulersche Formel

$$= \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N} kn\right) - i \cdot \sin\left(\frac{2\pi}{N} kn\right) \right]$$

Eingabesignal

Realteil (re)

Imaginärteil (im)



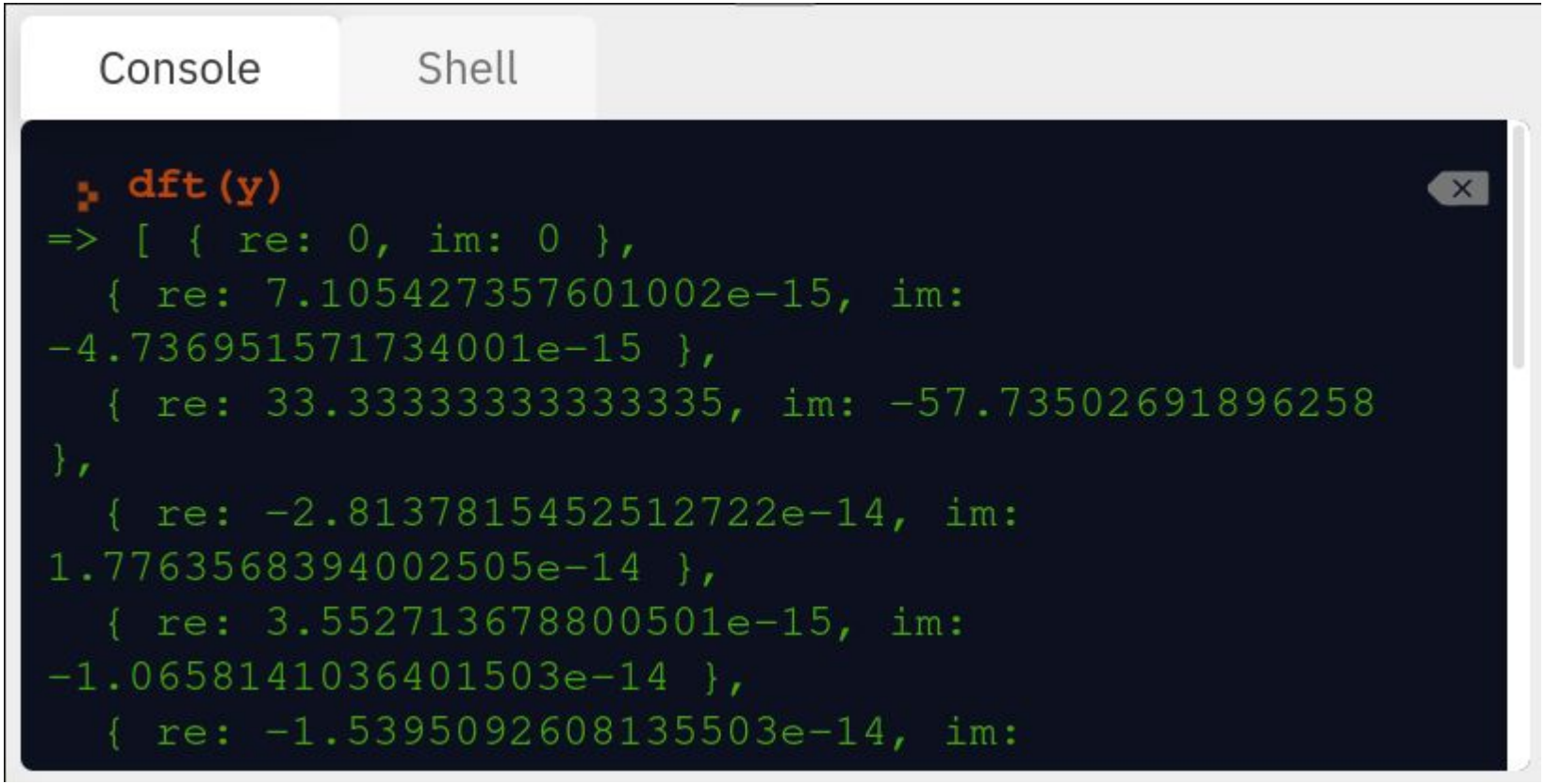
7. dft schreiben

```
63  function dft(x){
64    let X=[];
65    let N=x.length;
66    for(let k = 0; k < N; k++){
67      let re = 0;
68      let im = 0;
69      for(let n = 0; n < N;n++){
70        const phi = (TWO_PI * k * n) / N;
71        re += x[n] * cos(phi);
72        im -= x[n] * sin(phi);
73      }
74      re = re / N;
75      im = im / N;
76      X[k] = {re,im};
77    }
78   return X;
79  }
```

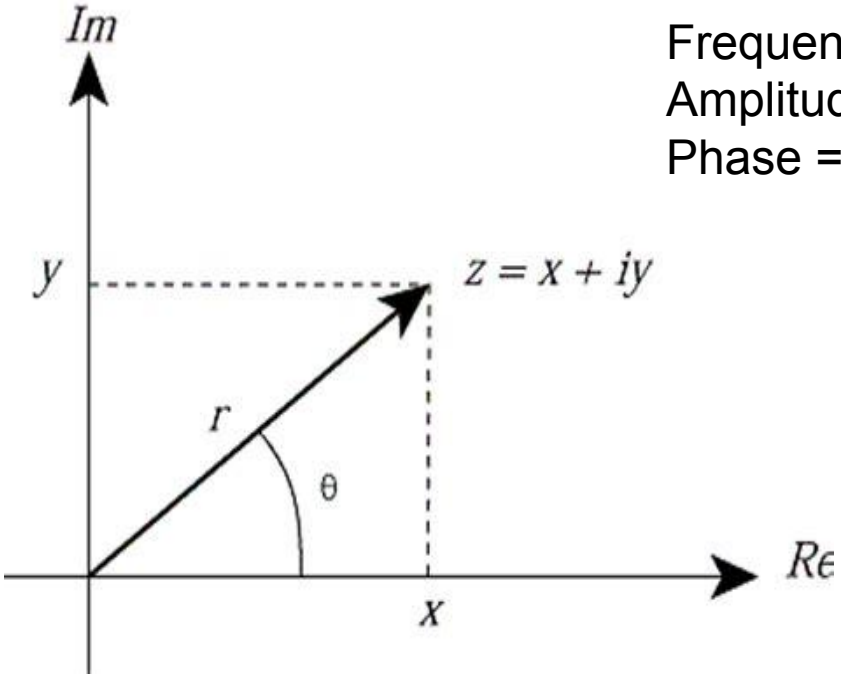
Jede einzelne Welle berechnen.

Summenzeichen

Funktion dft() testen



Wie stelle ich das in einem Kreis da?



Frequenz => k
Amplitude => Länge des Pfeils => sqrt(re*re + im*im)
Phase => θ => atan2(im, re)

8. dft() erweitern

```
function dft(x){
  let X=[];
  let N=x.length;
  for(let k = 0; k < N; k++){
    let re = 0;
    let im = 0;
    for(let n = 0; n < N;n++){
      const phi = (TWO_PI * k * n) / N;
      re += x[n] * cos(phi);
      im -= x[n] * sin(phi);
    }
    re = re / N;
    im = im / N;

    let freq = k;
    let amp = sqrt(re*re + im*im);
    let phase = atan2(im, re);
    X[k] = {re,im,freq,amp,phase};
  }
 return X;
}
```

9. Neue draw() Funktion

```
27    for (let i = 0; i < fourierY.length; i++) {
28      let prevx = x;
29      let prevy = y;
30
31      let n = fourierY[i].freq;
32      let radius = fourierY[i].amp;
33      let phase = fourierY[i].phase;
34      x += radius * cos(n * time + phase + HALF_PI);
35      y += radius * sin(n * time + phase + HALF_PI);
36
37      stroke(255, 100);
38      noFill();
39      ellipse(prevx, prevy, radius * 2);
40
41      fill(255);
42      stroke(255);
43      line(prevx, prevy, x, y);
44      ellipse(x, y, 8);
45    }
```

10. Neuen Timestep berechnen

```
57    const dt = TWO_PI / fourierY.length;
58    time += dt;
```

11. Neuen Funktion -> epiCycles

```
20  function epiCycles(x, y, rotation, fourier) {
21    for (let i = 0; i < fourier.length; i++) {
22      let prevx = x;
23      let prevy = y;
24      let freq = fourier[i].freq;
25      let radius = fourier[i].amp;
26      let phase = fourier[i].phase;
27      x += radius * cos(freq * time + phase + rotation);
28      y += radius * sin(freq * time + phase + rotation);
29
30      stroke(255, 100);
31      noFill();
32      ellipse(prevx, prevy, radius * 2);
33      stroke(255);
34      line(prevx, prevy, x, y);
35    }
36    return createVector(x, y);
37  }
```

12. Neue draw() Funktion

```
54  function draw() {
55    strokeWeight(1);
56    translate(300,500)
57    //scale(1/2, 1/2)
58    background(0);
59
60    let vx = epiCycles(width / 2 + 100, 100, 0, fourierX);
61    let vy = epiCycles(100, height / 2 + 100, HALF_PI, fourierY);
62    let v = createVector(vx.x, vy.y);
63    path.unshift(v);
64    line(vx.x, vx.y, v.x, v.y);
65    line(vy.x, vy.y, v.x, v.y);
66
67    beginShape();
68    stroke('blue');
69    strokeWeight(4);
70    noFill();
71    for (let i = 0; i < path.length; i++) {
72      vertex(path[i].x, path[i].y);
73    }
74    endShape();
75
76    const dt = TWO_PI / fourierY.length;
77    time += dt;
78
79    if (time > TWO_PI) {
80      time = 0;
81      path = [];
82    }
83  }
```

X-koordinaten

y-koordinaten

13. Neue Variablen für x und y

```
7    let y = [];
8    let fourierY;
9
10   let time = 0;
11   let wave = [];
```

```
7    let x = [];
8    let y = [];
9    let fourierX;
10   let fourierY;
11   let time = 0;
12   let path = [];
```

14. Neue setup() Funktion

```
15  function setup() {
16    createCanvas(800, 800);
17    const skip = 2;
18    for (let i = 0; i < data.drawing.length; i += skip) {
19      x.push(data.drawing[i].x);
20      y.push(data.drawing[i].y);
21    }
22    fourierX = dft(x);
23    fourierY = dft(y);
24
25    fourierX.sort((a, b) => b.amp - a.amp);
26    fourierY.sort((a, b) => b.amp - a.amp);
27  }
```

→ Nach größe sortieren

15. Das Bild laden

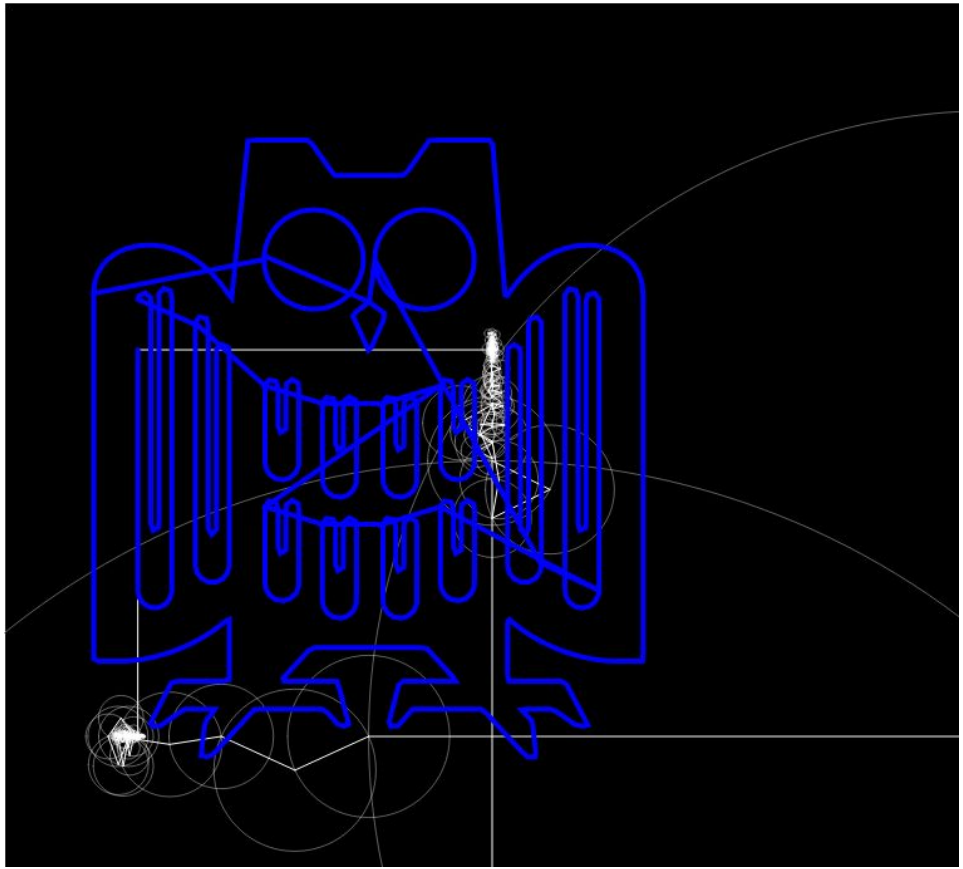
```
8   function preload() {
9   //  data = loadJSON('codingtrain.json');
10    data = loadJSON('UdS.json');
11  }
```

16. Bild am Ende neu malen

```
79  if (wave.length > 250) {
80    wave.pop();
81  }
```

```
79  if (time > TWO_PI) {
80    time = 0;
81    path = [];
82  }
```

Das Ergebnis 🎉



Jeweils 1491 einzelne Kreise für X- und Y-Koordinaten!!!!