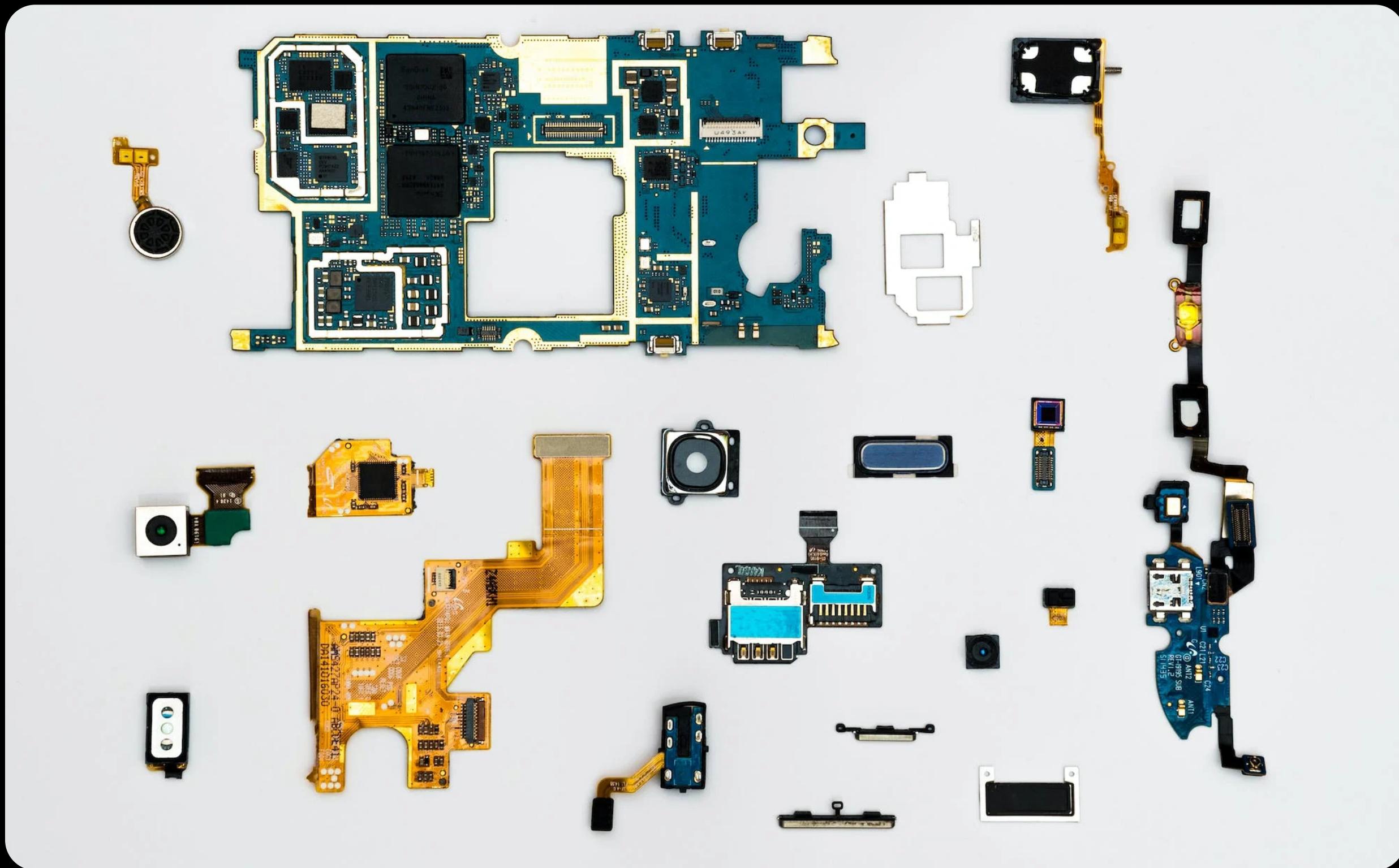


Introduction to p5.js

00

A word on p5.js



p5.js is open
source software
designed to
serve the
underserved.



Our mission is to promote software learning within the arts, artistic learning within technology-related fields, and to celebrate the diverse communities that make these fields vibrant, liberatory, and innovative. Our goal is to support people of all backgrounds in learning how to program and make creative work with code, especially those who might not otherwise have access to tools and resources. We also believe that some of the most radical futures and innovative technologies are being built by communities that have been pushed to the margins by dominant tech. We hope to support those who have been marginalized by technology in continued self-determination by providing time, space, and resources.

01

The p5.js Lifecycle



```
setup()  
draw()
```

The `setup()` function runs once, at the beginning of your program. Usually you use it to create the canvas to draw onto, or load images. It's a special reserved name that p5.js knows how to look for, and p5 runs it for you.

setup()
draw()

The draw() function runs once every frame, by default 60 frames per second. Because of this, it's a kind of *infinite loop*. This is where the magic happens. It's also a reserved name that p5.js knows how to look for and run.

1

setup()

2

draw()

1 setup()

2 3 draw()

1 setup()

2 3 4 draw()

1

setup()

∞

draw()

Refresher Variables

What is a Variable?

Variables are names which you assign values to, and can reassign value to later (hence the name). They're essential to a program's ability to "remember" values for later, or to pass values around within the program, and to your own ability to remembering what your code does and what each piece of data is for.

Untyped

Can store any kind of data.

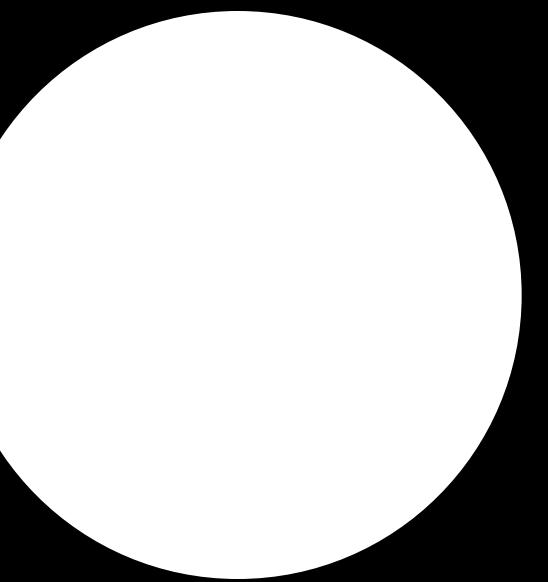
Typed

Can only store specific value types.

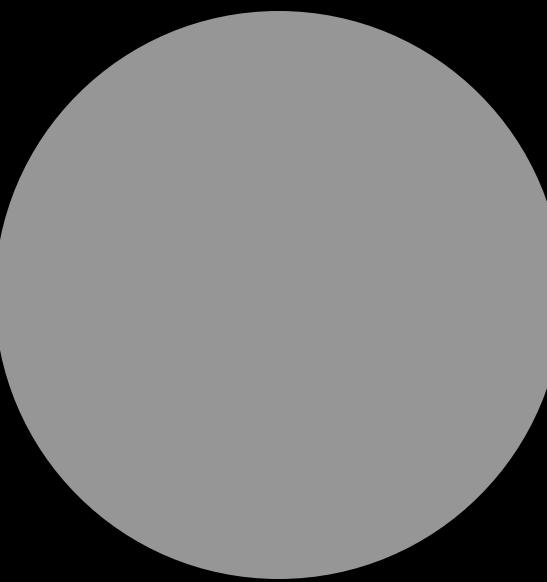
Constants

Cannot be changed later.

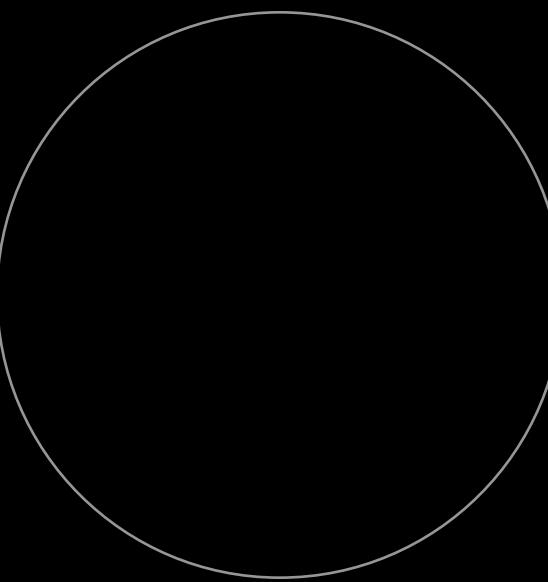
Nouns



Constants

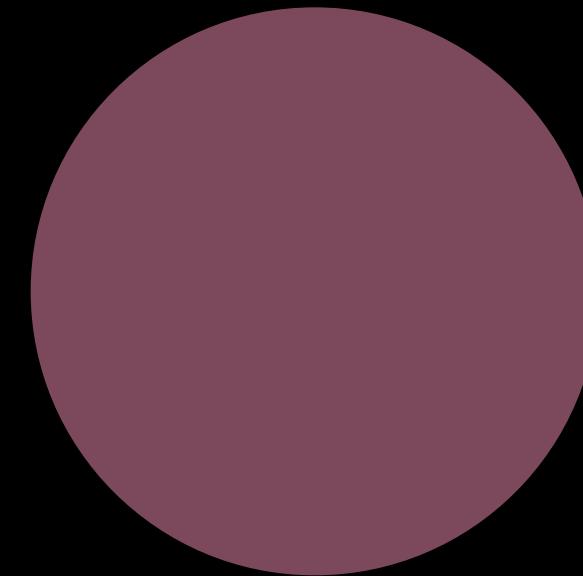


Typed

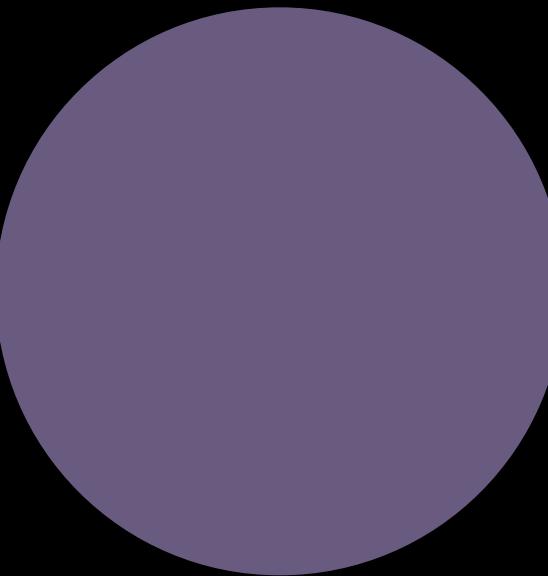


Untyped

Verbs

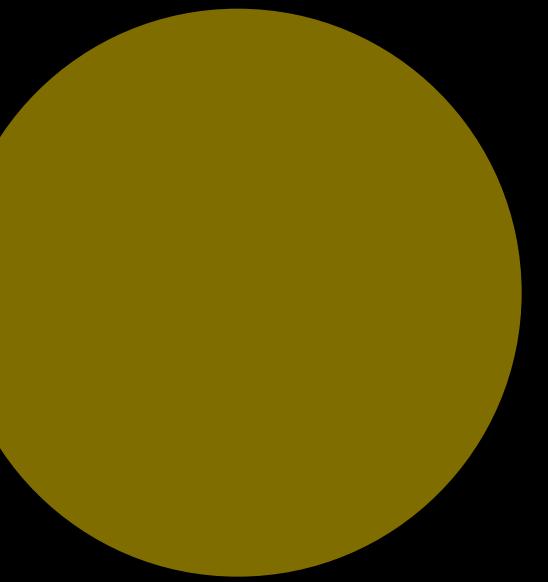


Imperative functions

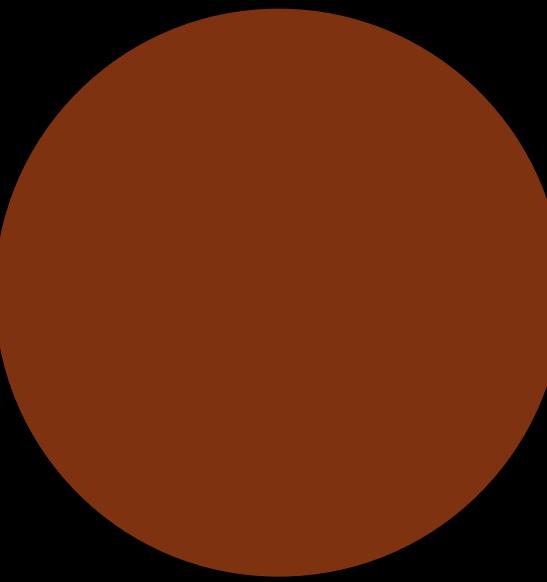


Event-driven functions

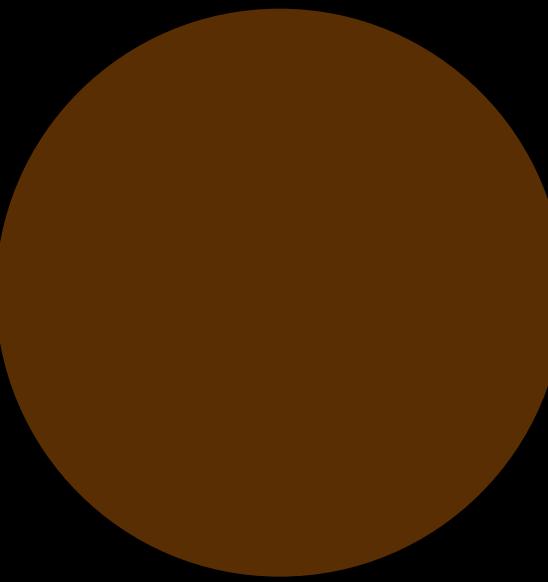
Grammar



Conditionals



Loops



Functions

What is a Variable?

Variables are names which you assign values to, and can reassign value to later (hence the name). They're essential to a program's ability to "remember" values for later, or to pass values around within the program, and to your own ability to remembering what your code does and what each piece of data is for.

Untyped

Can store any kind of data.

Typed

Can only store specific value types.

Constants

Cannot be changed later.

```
2
3 let GRID_ROWS = 9;
4 let GRID_COLS = 9;
5 const CANVAS_W = 400;
6 const CANVAS_H = 400;
7 const BORDER = 40;
8 const CANVAS_W_TRUE = CANVAS_W + 2 * BORDER;
9 const CANVAS_H_TRUE = CANVAS_H + 2 * BORDER;
10 let GRID_W = CANVAS_W / (GRID_ROWS - 1);
11 let GRID_H = CANVAS_H / (GRID_COLS - 1);
12
```

Refresher Functions

What is a Function?

If variables are the nouns of your program, the functions are the verbs: they are what your program does. In many languages, functions are also objects, meaning they can be stored inside variables. Functions are so important to programming that an entire class of languages, functional, exist in which the whole paradigm of programming is oriented around functions.

Imperative
Run on command.

Event-driven
Run in response to events.



What is a Function?

If variables are the nouns of your program, the functions are the verbs: they are what your program does. In many languages, functions are also objects, meaning they can be stored inside variables. Functions are so important to programming that an entire class of languages, functional, exist in which the whole paradigm of programming is oriented around functions.

Imperative
Run on command.

Event-driven
Run in response to events.

```
81
82 function setup() {
83   createCanvas(CANVAS_W + BORDER * 2, CANVAS_H + BORDER * 2);
84 }
85
86 function drawGrid() {
87   for (let i = 0; i < GRID_ROWS; i++) {
88     for (let j = 0; j < GRID_COLS; j++) {
89       stroke(255);
90       point(...gridToDrawCoord(i, j));
91     }
92   }
93 }
```

Live code demo

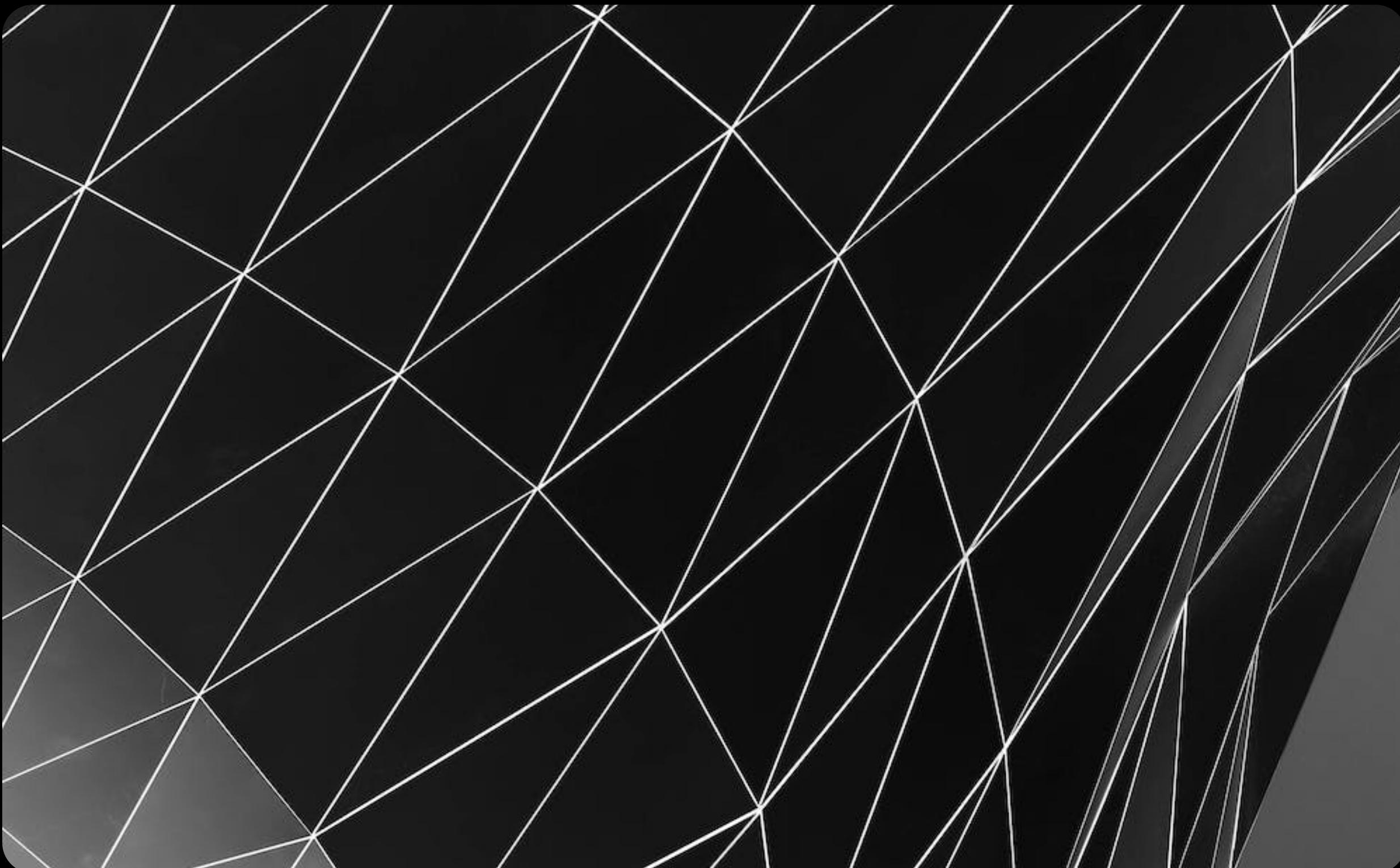
Variables

<https://editor.p5js.org/mngyuan/sketches>,
click on “CC W4 1”



02

Geometric Primitives



The p5.js site has documentation, examples, and resources, all free to use.



Home

Editor

Download

Donate

Get Started

Reference

Libraries

Learn

Teach

Examples

Contribute

Books

Community

Hello!

p5.js is a JavaScript library for creative coding, with a focus on coding accessible and inclusive for artists, designers, educators and anyone else! p5.js is free and open-source because we believe software, and the tools to learn it, should be accessible to everyone.

Using the metaphor of a sketch, p5.js has a full set of drawing functionality. However, you're not limited to your drawing canvas; you can think of your whole browser page as your sketch, including objects for text, input, video, webcam, and sound.

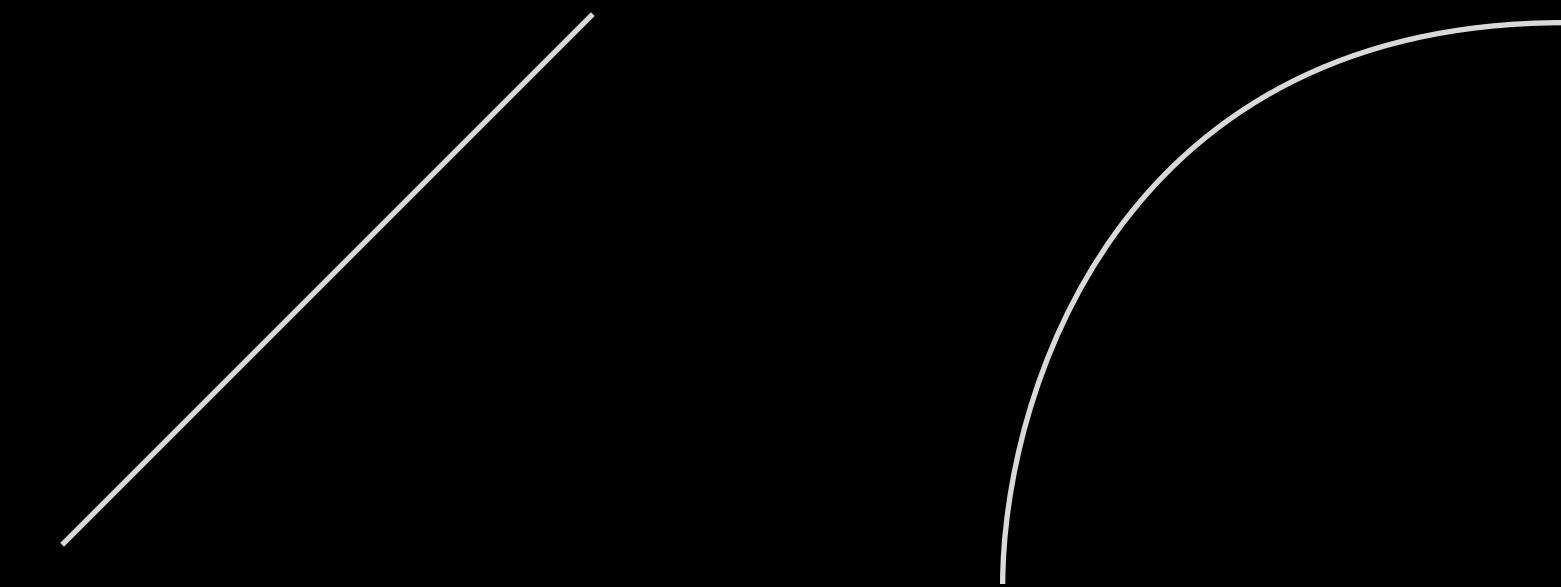
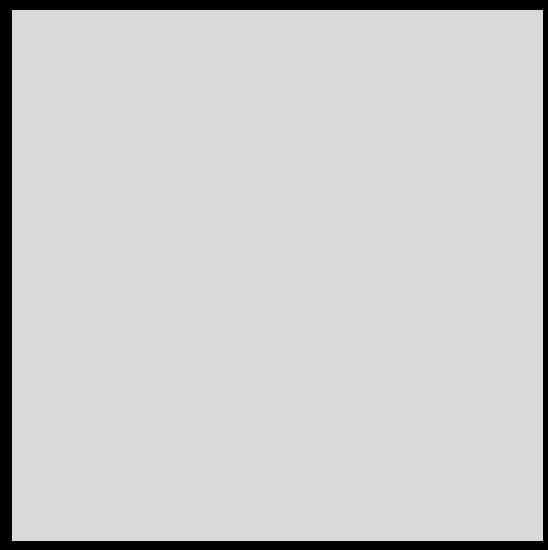
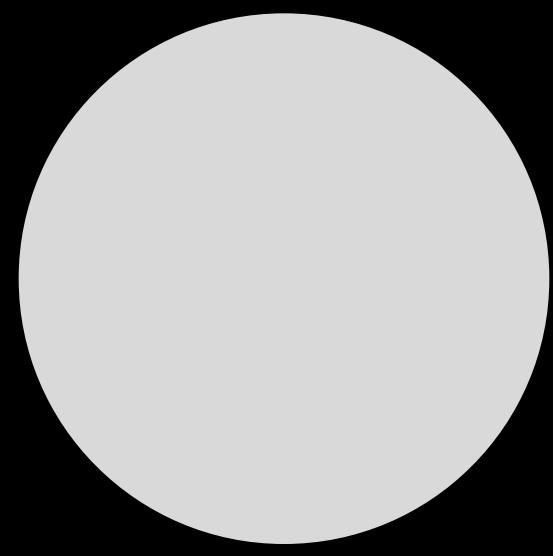
Start creating with the p5 Editor!

Community

We are a community of, and in solidarity with, people from every

Search p5js.org

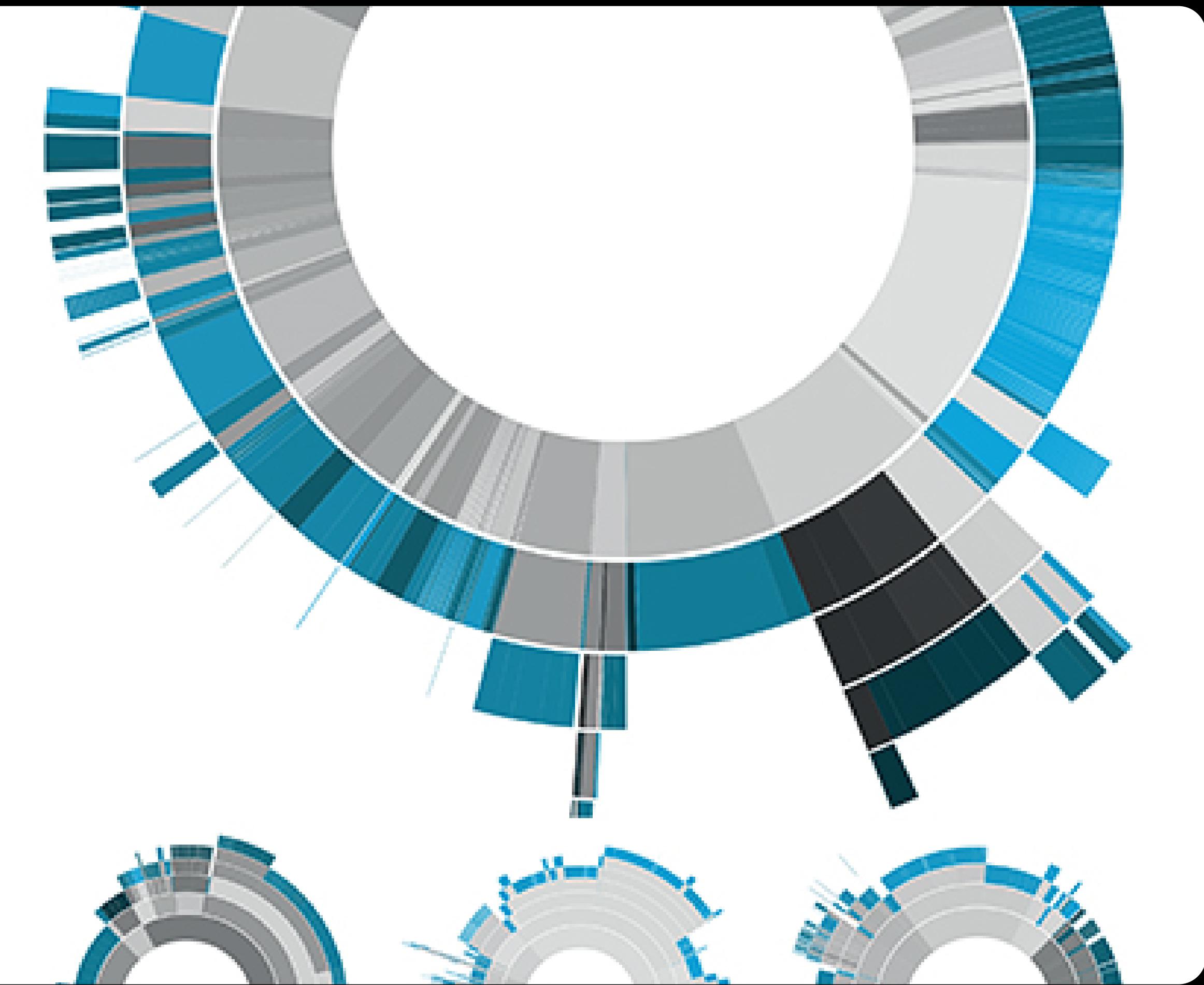
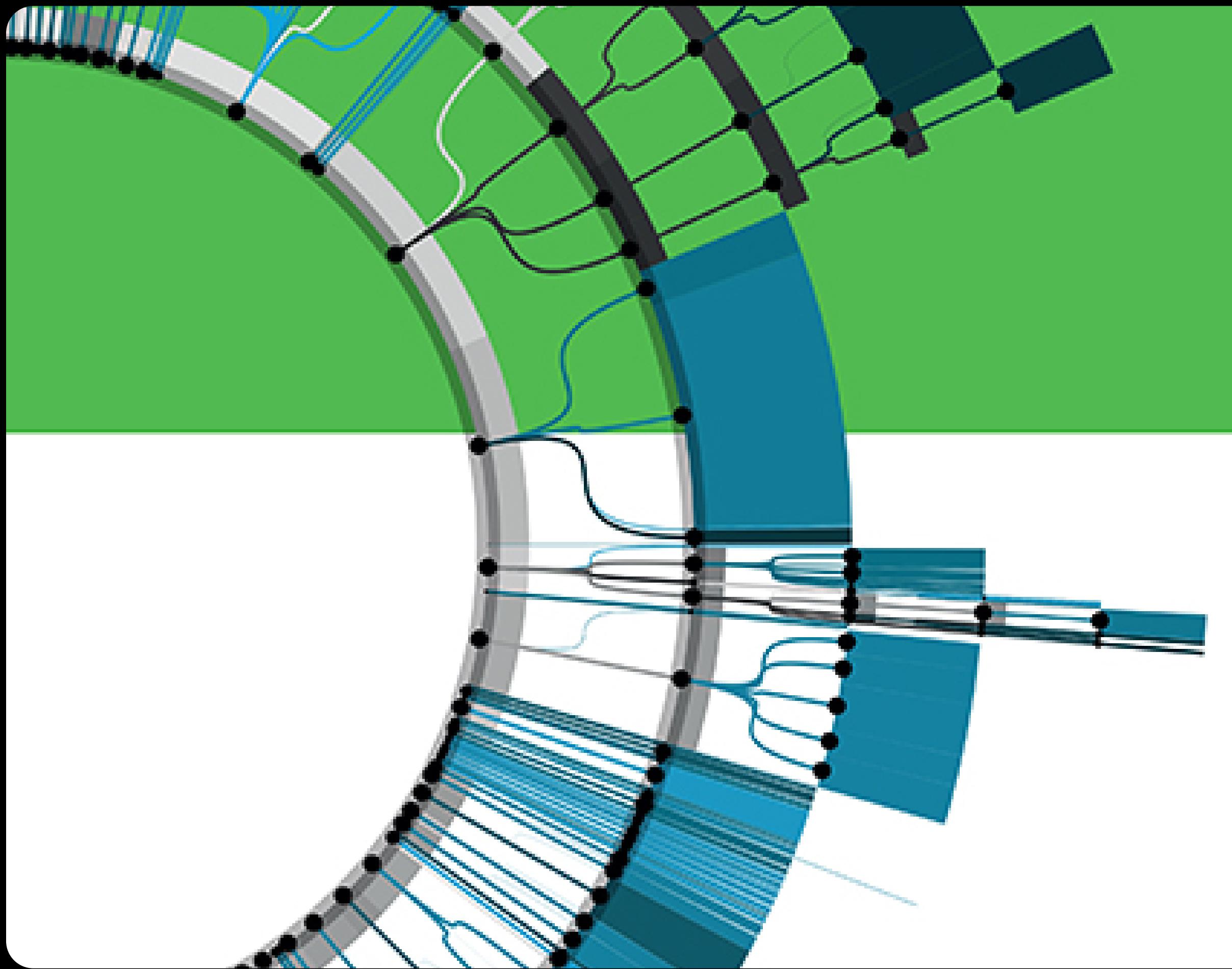
Geometric primitives



“Analog Algorithm”



“Generative Design”

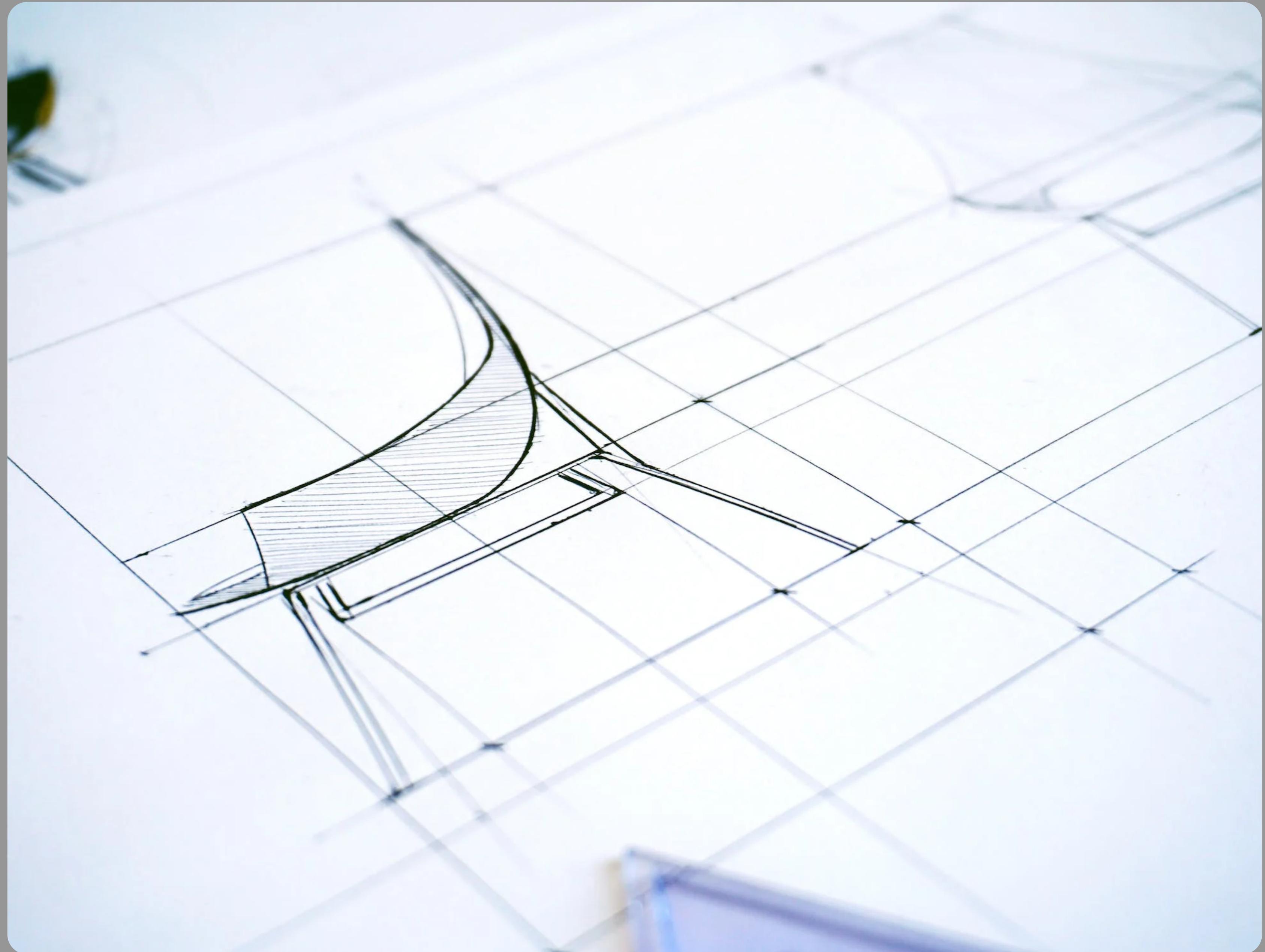


Live paper demo

Primitives

Using the 4 geometric primitives we talked about - the ellipse, the rectangle, the line, the arc - in groups, quietly

1. give each person one primitive. only they can use that primitive. once you've agreed,
2. slowly and without talking, draw the most interesting drawing you can as a group



Live code demo Primitives

<https://editor.p5js.org/mngyuan/sketches>,
click on “CC W4 2”

