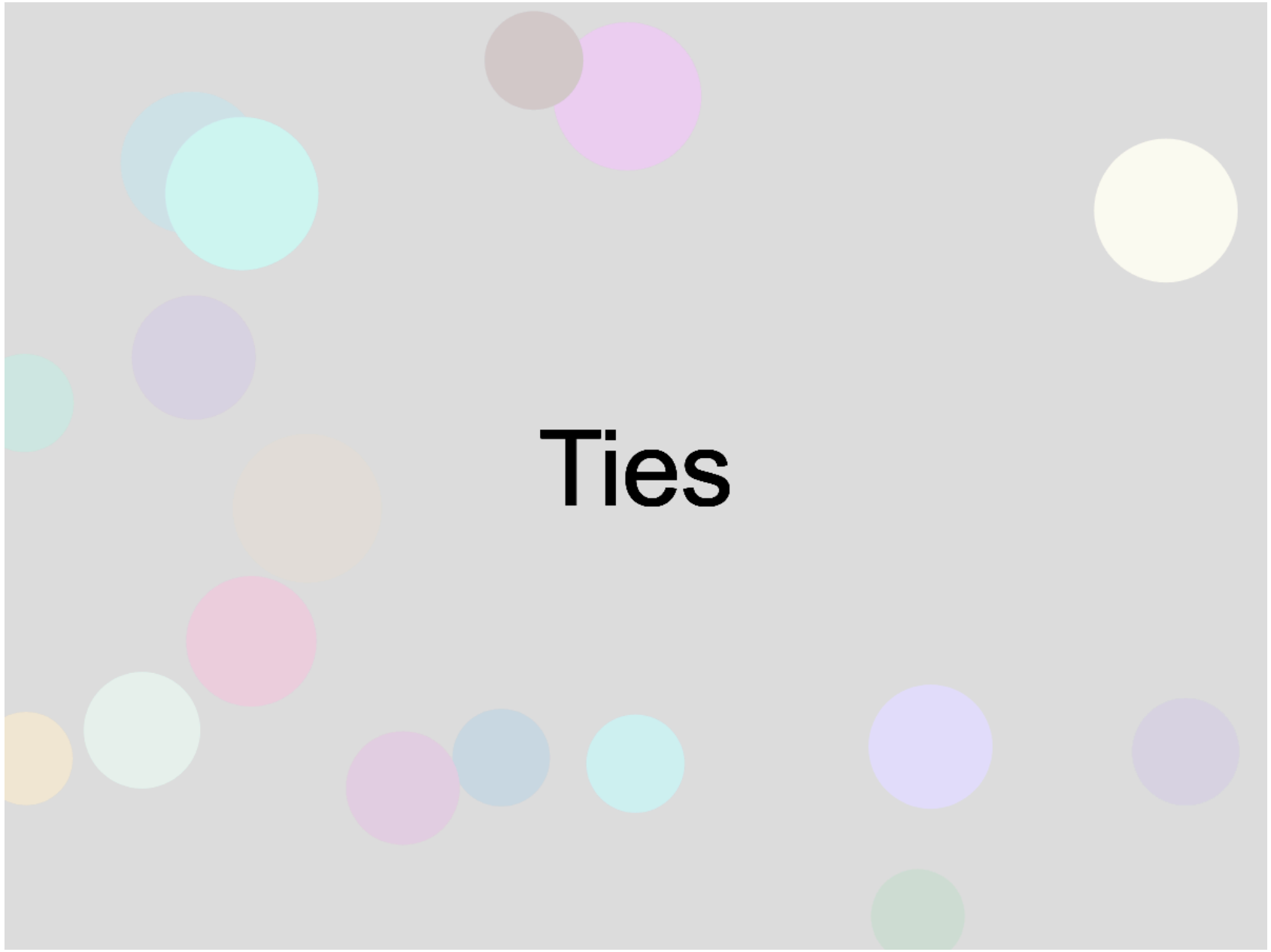
Creative Programming: Hello World!

Lecture 1 (27 Jan)

Narration

Table of contents

Narration



Narration

Why Creative Programming?

Narration

Creative Programming

- Programming is a communication tool in modern society
- Creativity allows us explore this and express ourselves through limitations

Narration

Creative Programming

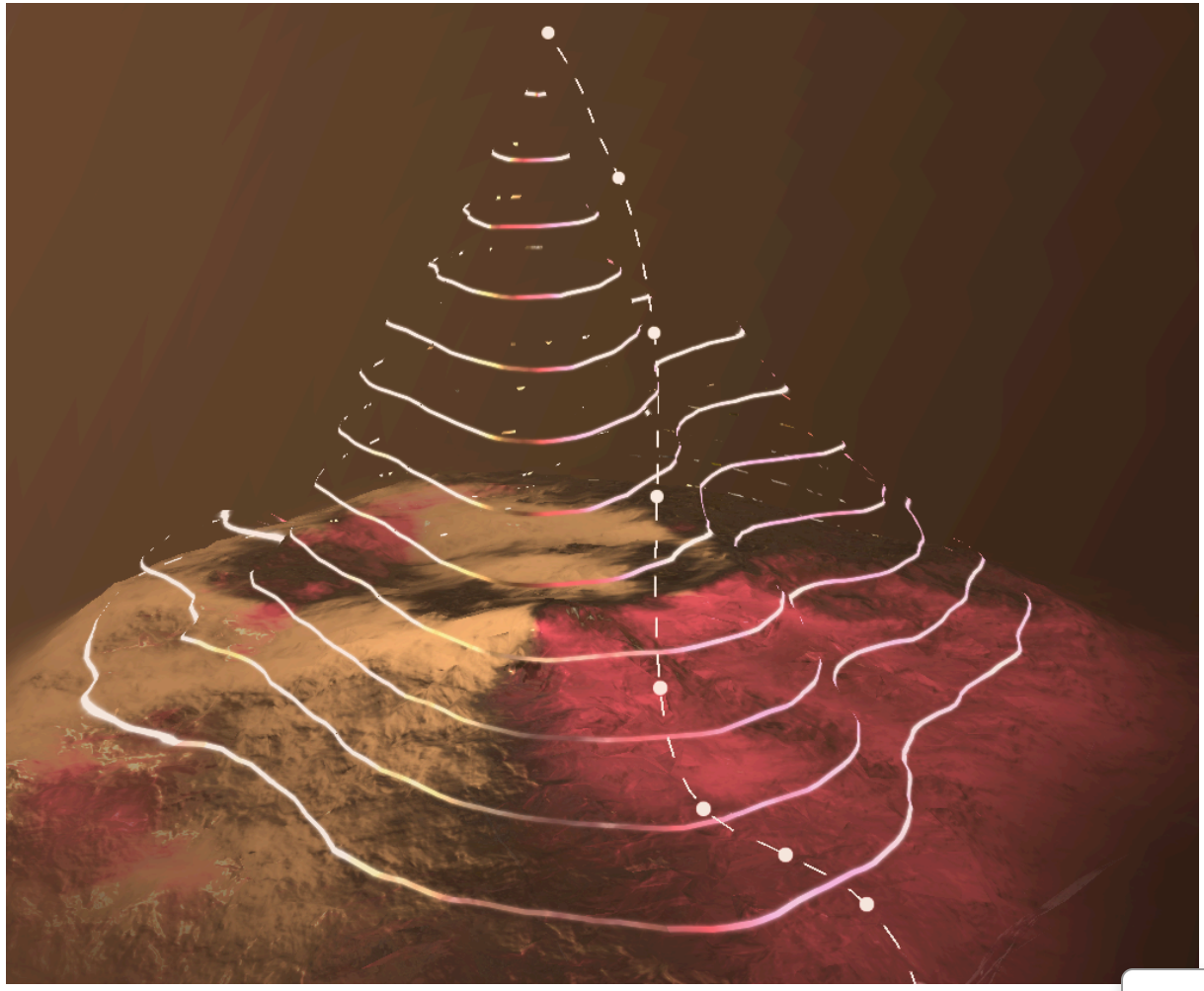
> Creative coders are artists, designers, architects, musicians, and poets who use computer programming as their chosen media. These practitioners blur the distinction between art and design and science and engineering […].

~Golan Levin and Tega Brain

Narration

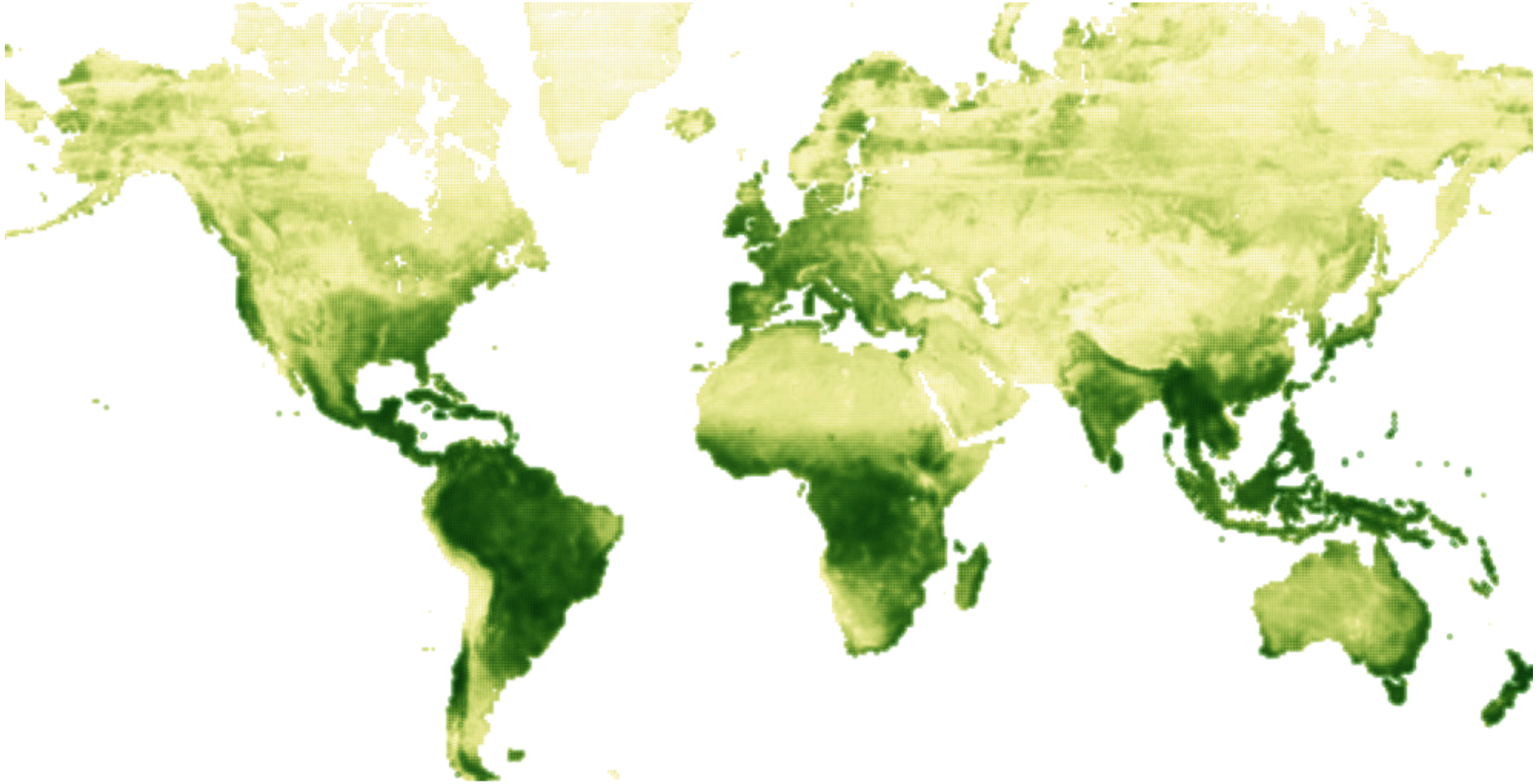
Insidious Rising

Hyphen Labs' Insidious Rising



Narration

Breathing Earth

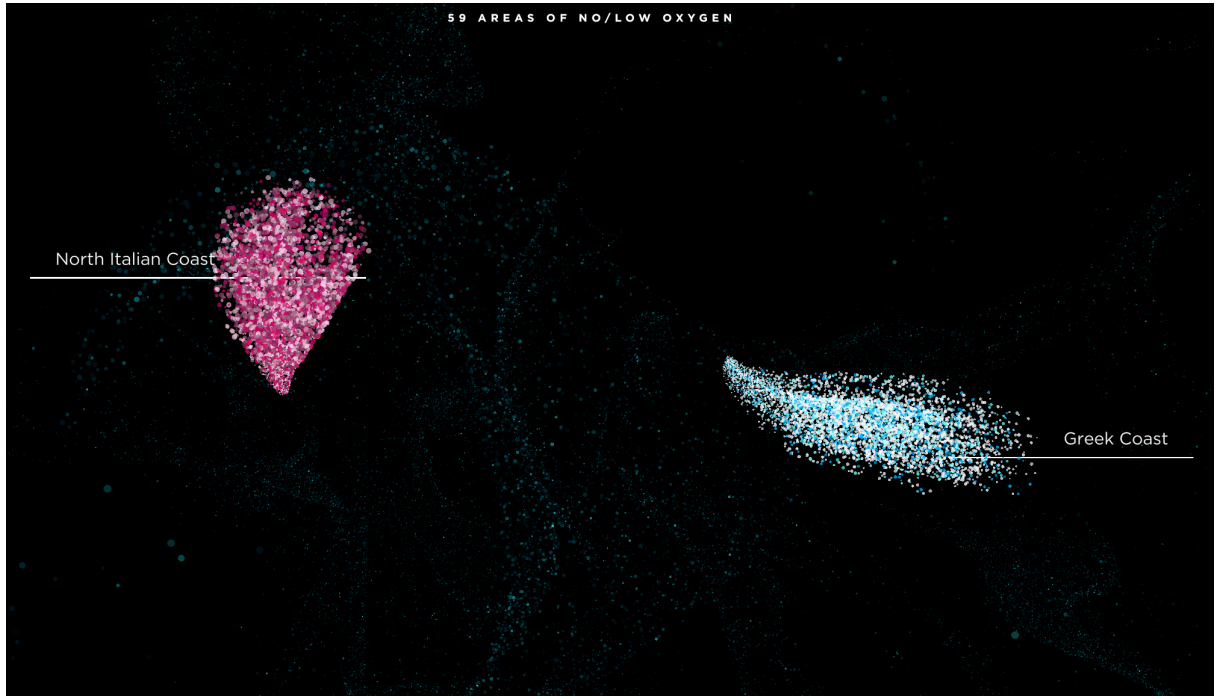


Narration

Breathing Earth by Sherley Wu

Medusae

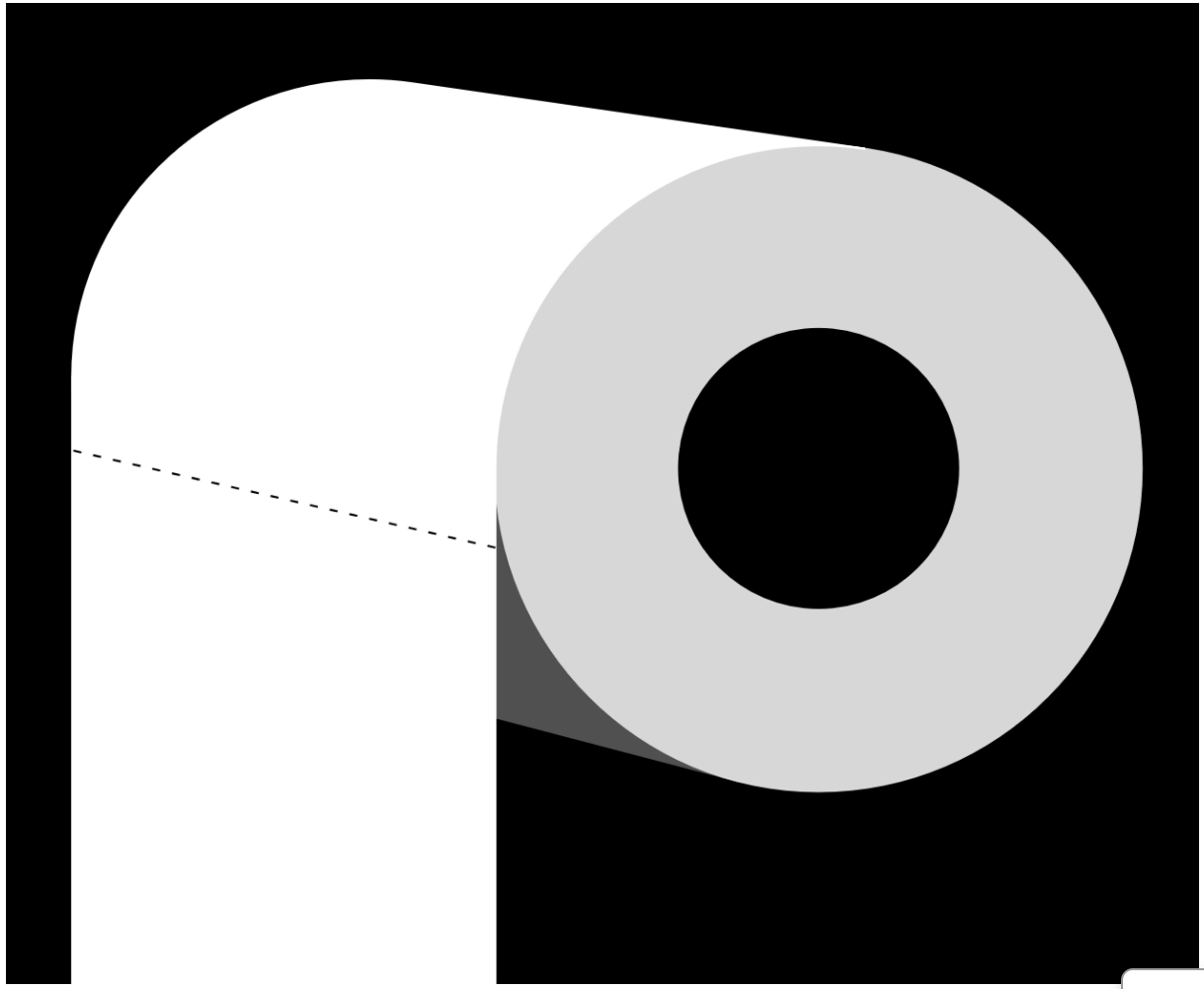
Medusae by Cristina Tarquini



Narration

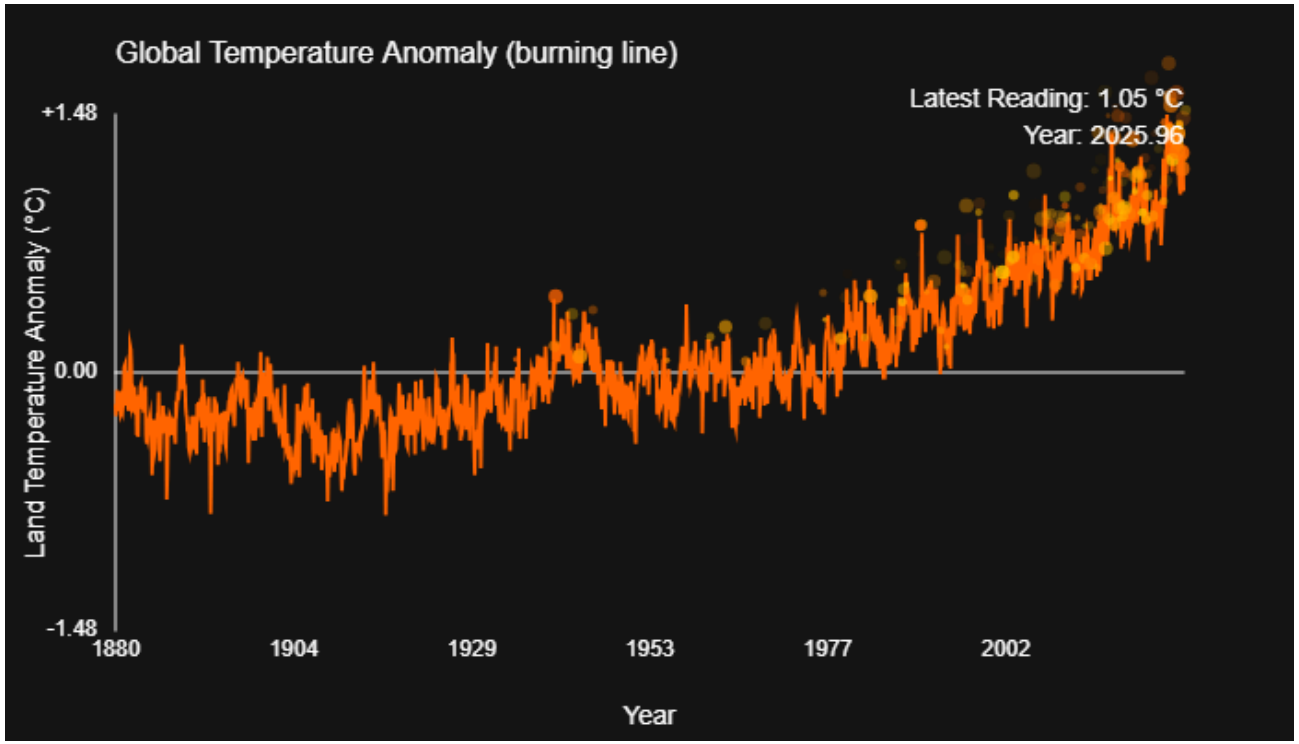
Paper Toilet

Paper Toilet by Rafaël Rozendaal



Narration

Temperature



Narration

Course Plan

Narration

Who are we?

Teachers:

- Ties Robroek (Lectures)
- Thomas Sandahl Christensen (Project)

Teaching Assistants:

- Axel Døring
- Kirstine Lund Hansen
- Line Larsen

Narration

Ties Robroek

- Hit the local news building a pancake robot
- Spent too much money in board games
- Hobbyd on Pokémon fangames for a couple of years

Narration

Thomas Sandahl Christensen

- Bachelor in what is now DDIT. Master in design from Goldsmiths
- Works as designer and fabricator in my own practice
- Founding member of the interactive art collective Circuit Circus

Fun facts:

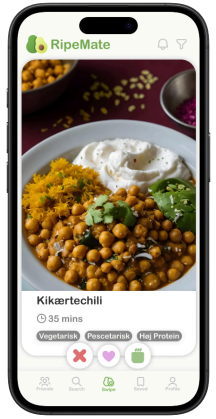
- No warm weather; enjoys long distance hiking and cross country skiing.
- I have a NFC chip embedded in my hand - so I’m technically a cyborg.

Narration

Axel Lolle Døring

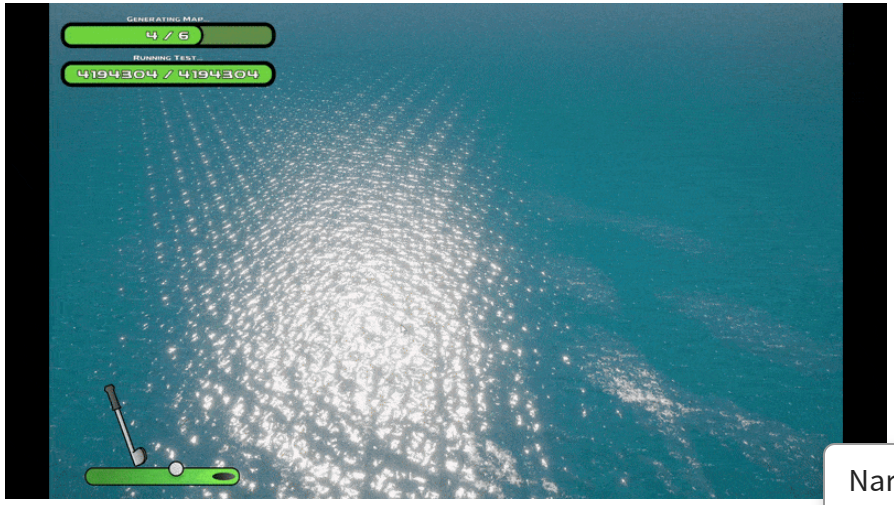
Why I love programming

- Finding novel solutions to problems



Fun facts

- I wear a dress shirt every day
- I have a christmas movie spreadsheet



Narration

Kirstine Lund Hansen

- Studied BDDIT and beginning my KDDIT thesis
- Coding can be thought of as the moldable material we are working with as interaction designers
  - I’m looking forwards to sharing my experiences and seeing your projects unfold!

Narration

Line Larsen

- 2nd year BDDIT
- DDIT is my third started bachelor's degree (SWU and psychology)
- I am OBSESSED with Baldur's Gate 3

Narration

When, What, Where, and Who

Generally (but check the course plan!)

When	What	Where	Who
Tuesday 8:00-12:00	Lectures	5A14-16	Ties
Thursday 12:00-16:00	Exercises	5A14-16	TAs

Narration

Lectures

- Lectures are *long* 4-hour sessions; we will try to keep it interactive
- Lectures include both background and live coding
  - It will be very hands on during the lectures. You need a working coding environment so you can follow
- The exercises are introduced but not elaborated upon in the lectures

Narration

Exercise Sessions

- Exercise Sessions are mandatory, TA-led sessions
- Important for getting started with bi-weekly assignments, getting help, and working with your groups
- There are four exercises to introduce you to programming, after which we start spending more time on the project

Narration

Course Plan

Creative Programming

Spring 2026

Course Plan	Lectureno	Weekno	Dates	Topics	Hand-in*	Instructor
COURSE INTRODUCTION	1	Week 5	27-Jan	Hello World!	Introduce Yourself	Ties
			29-Jan	[EXERCISE] Introduce Yourself		
INTRODUCTION TO CREATIVE PROGRAMMING	2	Week 6	03-Feb	Fundamentals		Ties
			05-Feb	[EXERCISE] Introduce Yourself		
	3	Week 7	10-Feb	Colour & Project Introductions	Interactive Drawing	Ties
			12-Feb	[EXERCISE] Interactive Drawing		
	4	Week 8	17-Feb	Interaction		Ties
			19-Feb	[EXERCISE] Interactive Drawing		
	5	Week 9	24-Feb	Sound	Make Some Noise	Ties
			26-Feb	[EXERCISE] Make Some Noise		

Narration

PROJECT START	6	Week 10	03-Mar	Milestone: Project Presentations	Milestone 1: Project Proposals	Thomas
			05-Mar	[EXERCISE] Technical Supervision		
DATA + PROJECT	7	Week 11	10-Mar	Data Representation	Handling Data	Ties
			12-Mar	[EXERCISE] Handling Data (Data Collection)		
	8	Week 12	17-Mar	Scaling Data		Ties
			19-Mar	[EXERCISE] Handling Data		
	9	Week 13	24-Mar	Visualisation		Ties
			26-Mar	[PROJECT WORK]		
BREAK		Week 14		Easter Break - No Classes		
DATA + PROJECT	10	Week 15	07-Apr	Resources	Milestone 2: Technical Deliverable	Ties
			09-Apr	Milestone 2 Project Presentations		
PROJECT		Week 16	14-Apr	Project Supervision (Thomas)		Thomas
			16-Apr	[PROJECT WORK]		
		Week 17	21-Apr	Project Supervision (Thomas)		Thomas
			23-Apr	[PROJECT WORK]		
		Week 18	28-Apr	Reflections, Report Overview & Supervision (Thomas)		Thomas
			30-Apr	[PROJECT WORK]		
	11	Week 19	05-May	Show & Tell	Milestone 3: Project Demonstration	Thomas Ti
			07-May	[EXPO] Public Design Demonstration		

Narration

The Project

- This course is built around a group project with 3-4 people.
- Throughout the semester there are several milestones
- As the semester goes on you spend less time on exercises and more on the project

Narration

Project Timeline

When	What
Tuesday Week 7	Group and Project introductions
Tuesday Week 10	Milestone 1: Present Project Concept
Thursday Week 15	Milestone 2: Present Project Progress
Tuesday Week 19	Milestone 3: Final Presentation
Friday Week 22	Report Hand-in

Narration

Project Deliverables

- Description of the creative programming process
- Technical details of your interactive data representation
- Audiovisual documentation of the final project
- Presentation at a closing event

Please note: during the design process, the only evaluations that you will do are by you yourself as designers in your creative programming experiments!

Narration

Examination

Exam type	**C1G: Submission of written work, internal (7-trinsskala)** *How well you have attained the ILO's of the course*
Report	**12 pages group report** *Documenting the group project* *Including 1-page individual reflection per member* *Including individual reflection on the process*
Hand-in	**Fri, May 29, 2026 (14:00)**

Narration

Course Goal

Narration

Programming Fundamentals

After the course, the student should be able to:

Apply programming fundamentals in a creative programming environment.

Narration

Programming Creatively

After the course, the student should be able to:

Program creative and interactive artifacts using P5.JS (in JavaScript)

Narration

Understanding Code & Applications

After the course, the student should be able to:

Analyze and describe creative programming projects.

Narration

A Critical Approach

After the course, the student should be able to:

Use programming as a tool to communicate your perspectives on a pressing issue in a broader society.

Narration

Individual Reflection

After the course, the student should be able to:

Reflect on the principles and issues underlying a creative programming process.

Narration

Why should we care?

- Develop your programming skills – work more with code.
- Develop your creative programming skills – explore and experiment through code.
- Approach data & code as design materials.
- Nurture your ability to develop a creative practice.
- Articulate your creative design process and reflect

Narration

Classroom Culture

- Be creative with programming and data.
- Use programming as something to explore, not just a final tool.
- Focus on the process, not skill levels.
- Support each other and learn together.
  - Try to solve problems together!

Narration

Artificial Intelligence

- There are a *lot* of AI tools out there, also for programming
- The issue is that when you use AI, you do not *fail*, which hinders learning
- Do not fear to make mistakes!
- If you need to use AI, limit it to using it as a search engine

Narration

Programming Experience

- Programming is a skill that requires a lot of “miles” and practice.
- Have you programmed before?
- The menti results are available on LearnIT.

Narration

Course Structure

Narration

LearnIT

- Main channel for communication
- Course program and planning
- Every lecture/exercise has its own section
- Direct links to slides other resources

Narration

GitHub

- GitHub is a place where people can put their code projects
- For this course: Lecture slides and resources are hosted there (but LearnIT has links)
- https://github.com/Sipondo/creative-programming-2026

Narration

Slides

This course uses quarto slides instead of PowerPoint.

Creative Programming is all about *interactivity*, and the slides reflect this. Lectures are hands-on!

You can open the slides on any device; they are *websites*.

Narration

(Re)visiting slides?

While there is no audio/video recording, these slide are *narrated with text*.

Did you miss a slide or want to revisit?
Open the narration tab while
studying to get an explanation of
difficult slides.

Narration

Coding Examples (Static)

Throughout this course we will have many slides with coding examples.

Sometimes the code is shown without displaying what it does:

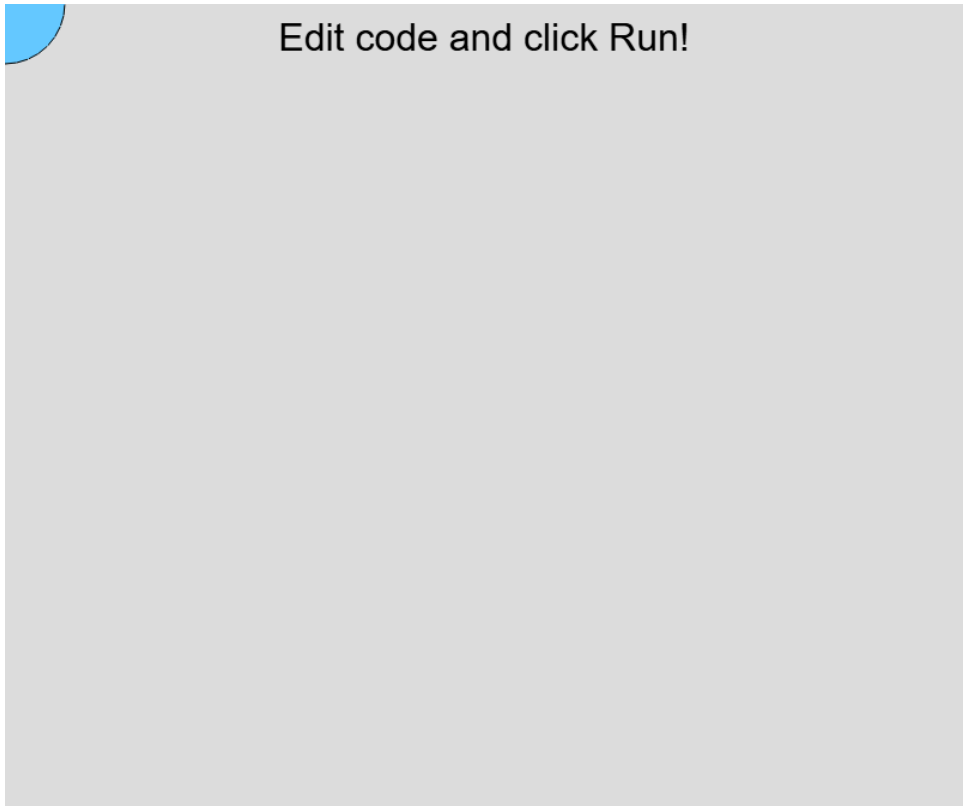
```
function setup() {
    createCanvas(400, 400);
    background(220);

    // Draw a circle
    fill(255, 0, 0);
    circle(100, 100, 80);
}
```

Narration

Coding Examples (Interactive)

Other times we have interactive coding blocks, just like the editor on your computer.

```
1  function setup() {
2      createCanvas(800, 800);
3  }
4
5  function draw() {
6      background(220);
7
8      // Draw a circle that follows the mouse
9      fill(100, 200, 255);
10     circle(mouseX, mouseY, 100);
11
12     // Draw some text
13     fill(0);
14     textAlign(CENTER, CENTER);
15     textSize(32);
16     text('Edit code and click Run!', width/2, 30);
```



Narration

Let’s get started!

Narration

Exercise 1: Hello World!

> Hello World: a simple computer program that emits (or displays) to the screen a message similar to “Hello, World!”.

These two weeks we are going to make a creative version of a “Hello World!” program.

First time we are expressing ourselves in programming - make a program that displays your name / favourite colour / hobbies / mood.

Narration

Introduce Yourself!

Brainstorm together in groups of 5-6 - who are you and how do *you* want to say “Hello World!”?

This info will be really useful for the first exercise!

The tldraw results are available on LearnIT.

Narration

p5.js

- We learn programming in this course in **p5.js**.
- p5.js is a framework written in Javascript, one of the most widely used programming languages in the world.
- Nearly every website on the internet uses Javascript!

Narration

Downloading and Installing

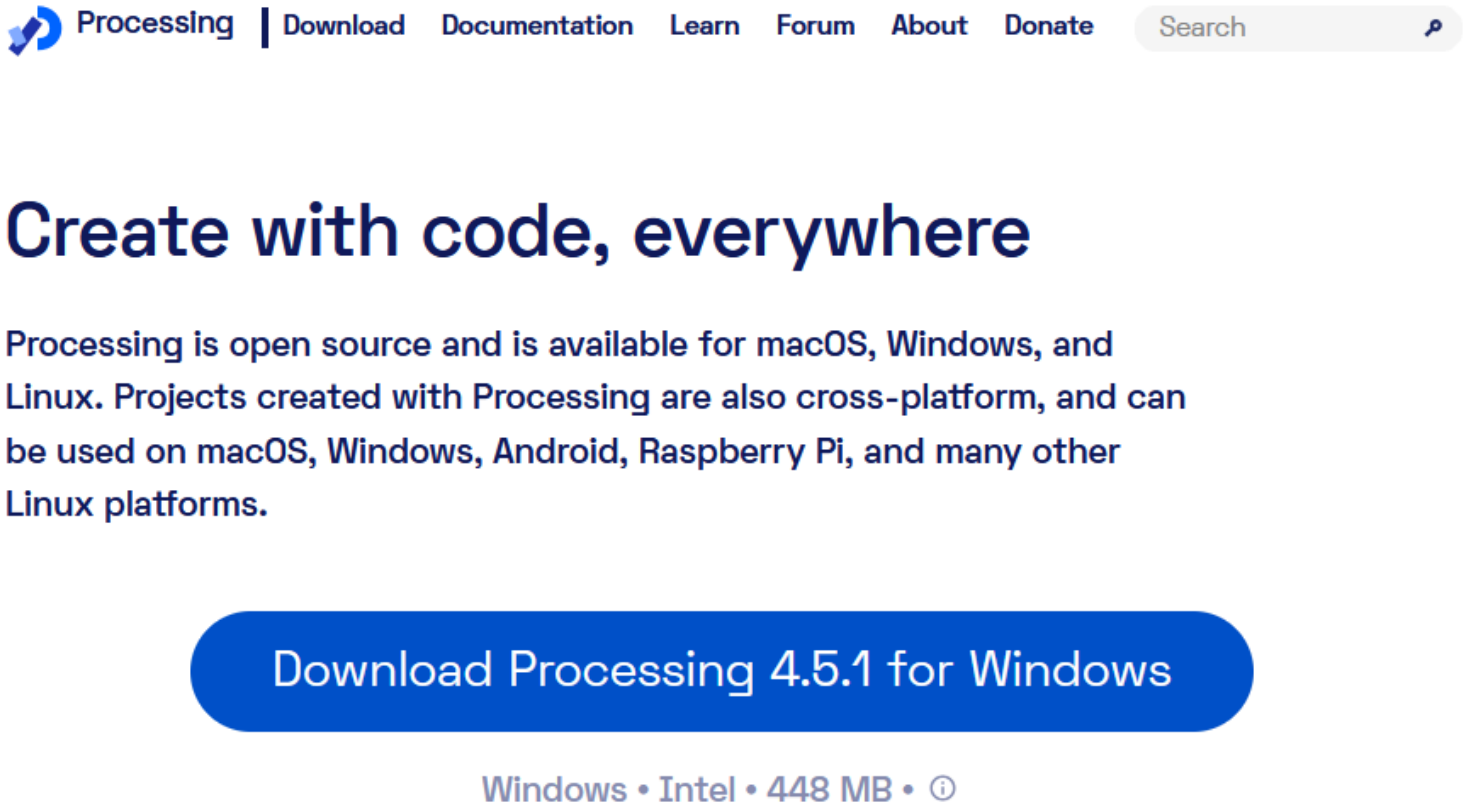
- Similar to editing text documents in Word, we need a program to edit our code in.
- For this course, we will write our code in Processing.
- How many have already downloaded processing?

Processing

Narration

Step 1: Download Processing

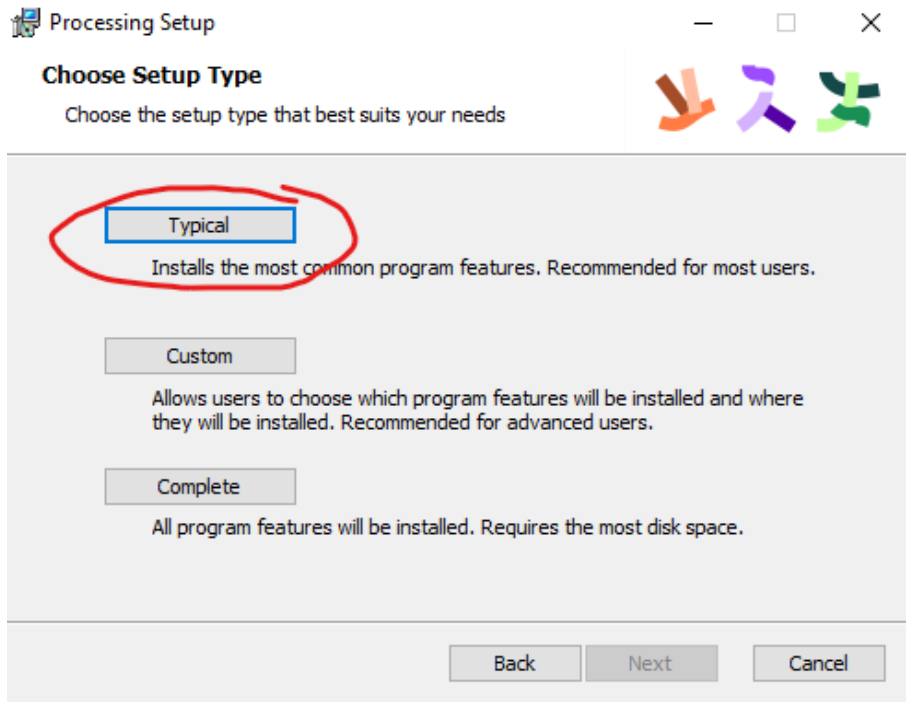
Go to https://processing.org/download

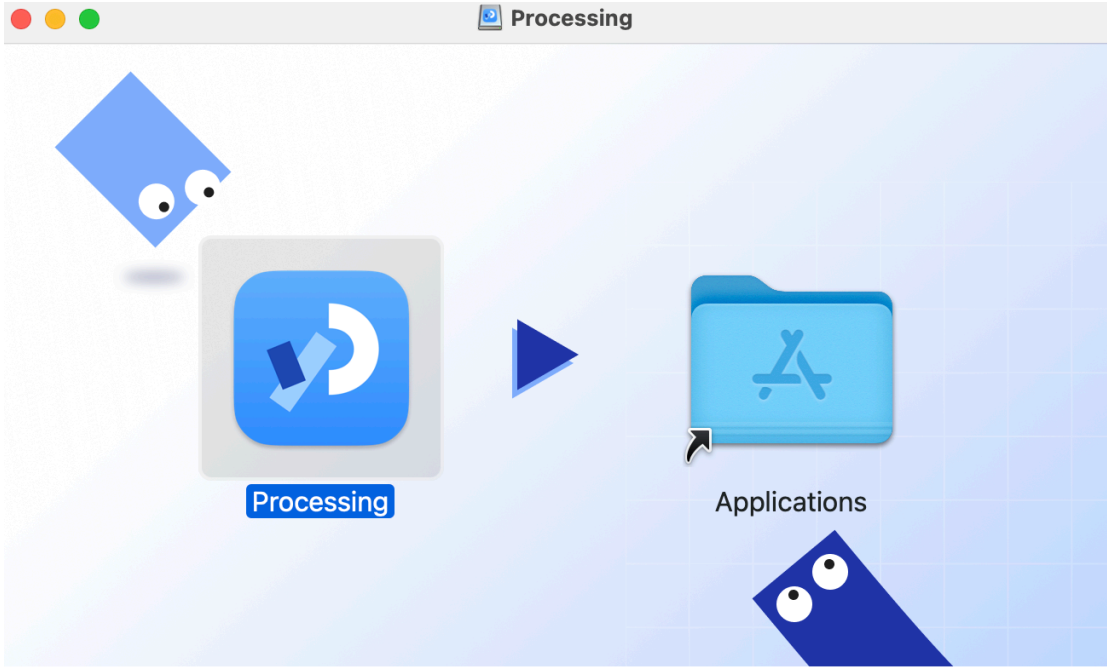


Narration

Step 2: Install Processing

Install Processing.

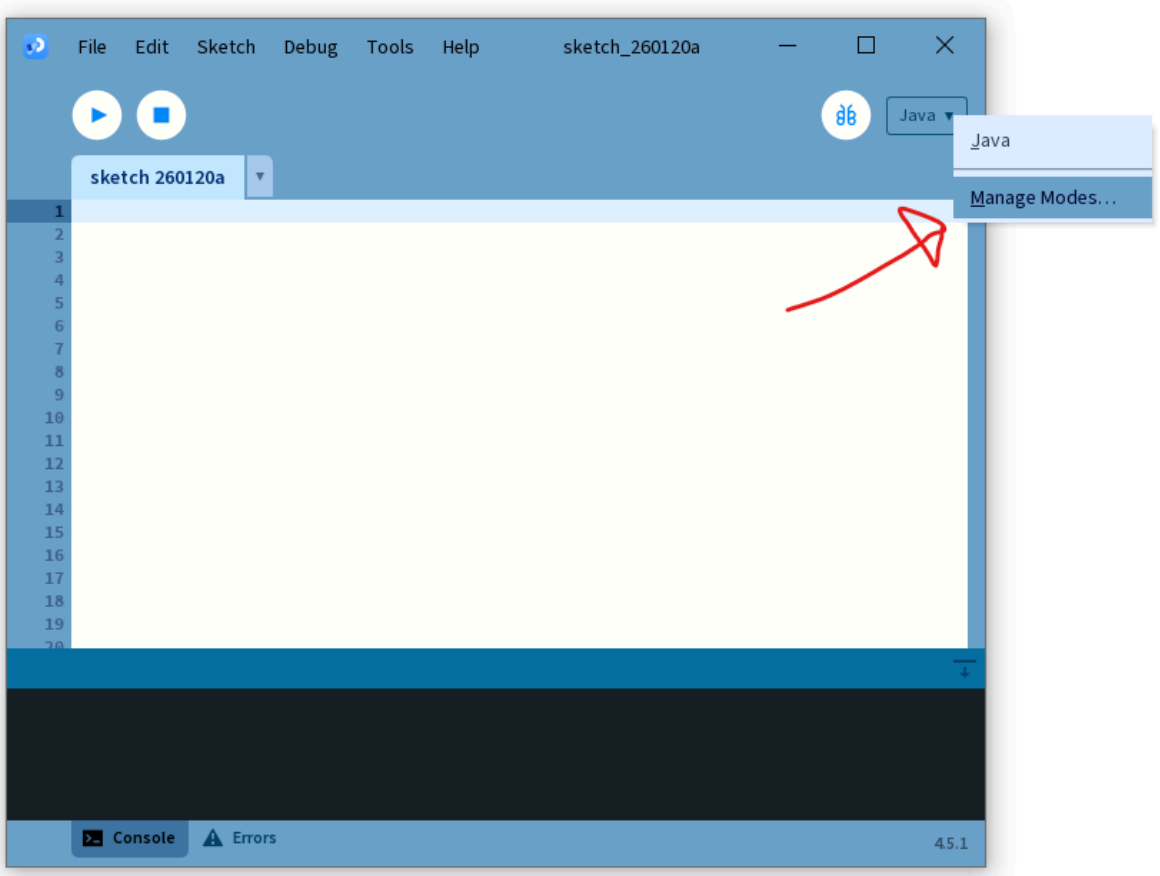




Narration

Step 3: Manage Modes

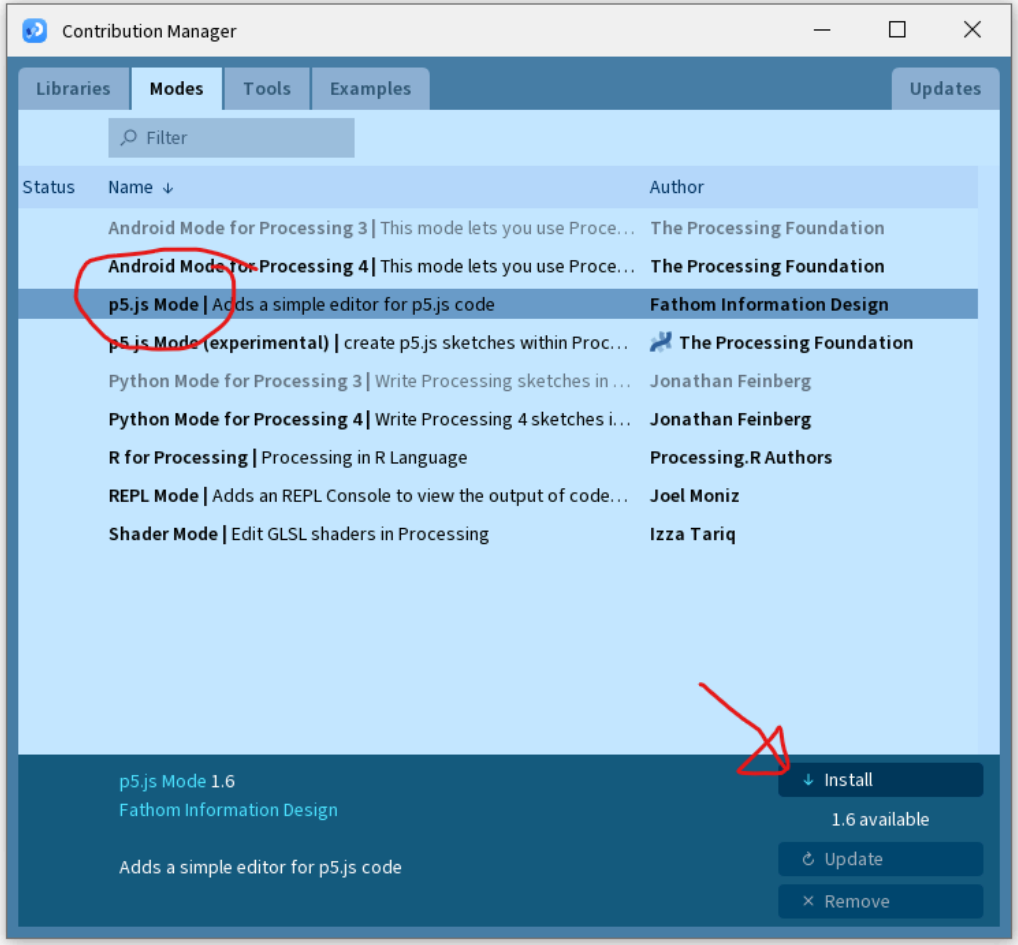
Open Processing, click away the welcome message, and select “Manage Modes” in the top right.



Narration

Step 4: Install Mode

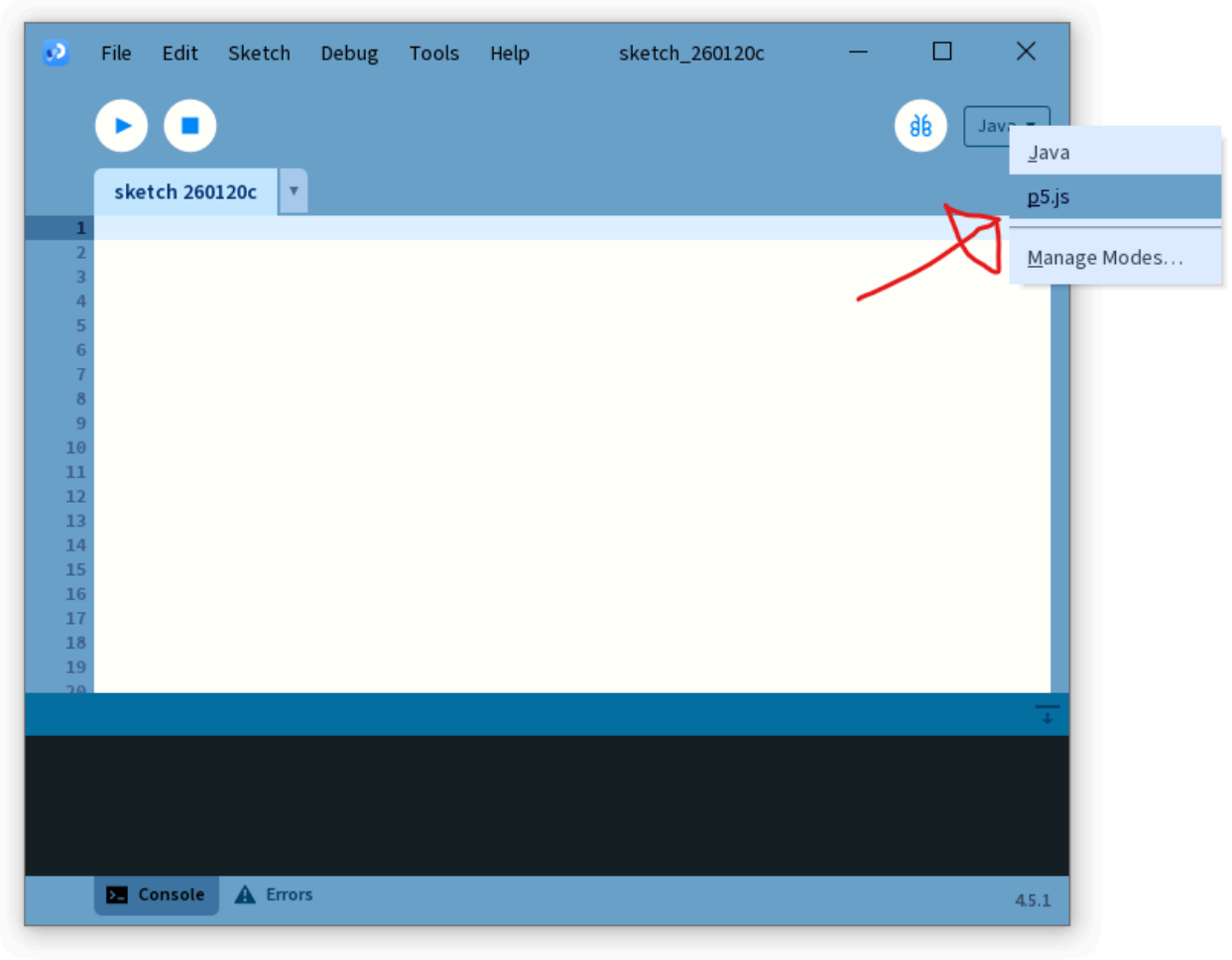
Select and install the p5.js mode.



Narration

Step 5: Select Mode

Back in the editor, select the p5.js mode.



Narration

Creative Programming: Let’s get coding!

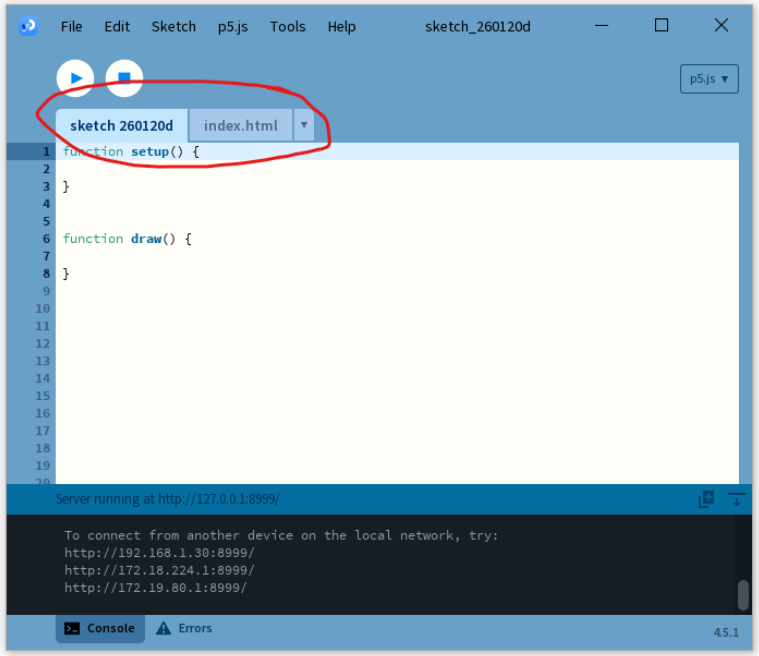
Make sure you have your Processing editor running and ready.

Narration

Sketch and index.html

Our p5.js sketch has a sketch (javascript) file and an index.html file.

We will just focus on the sketch file (for now).



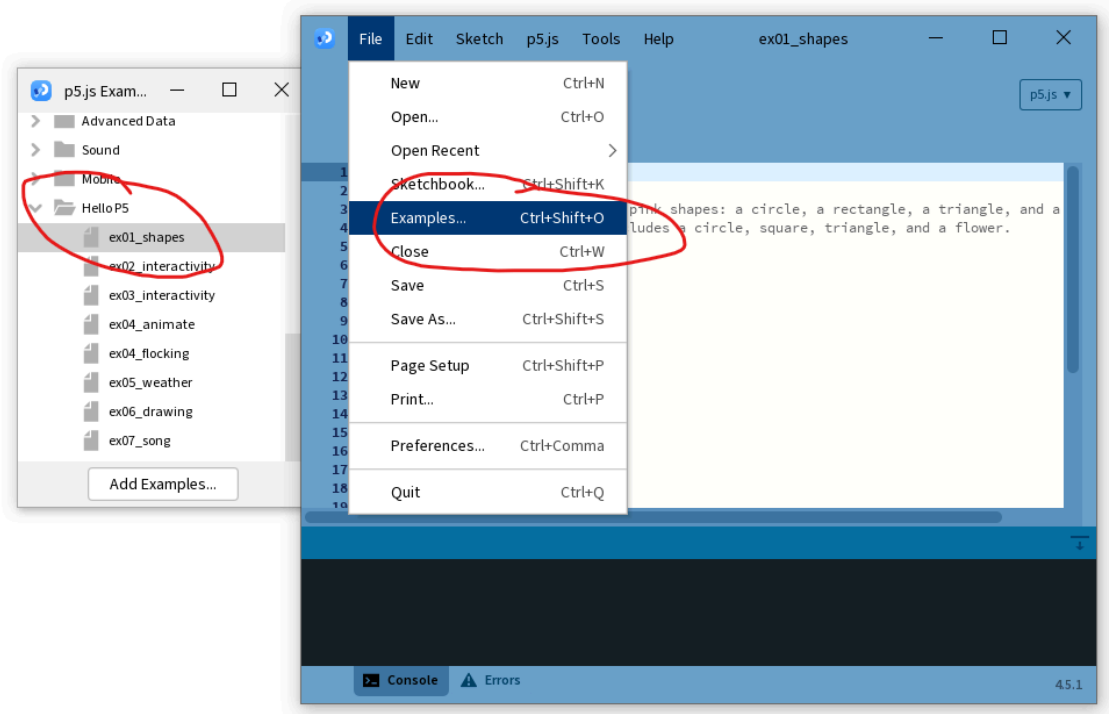
Saving a sketch will result in a folder with both these files.

Narration

Let’s open an example

There are a bunch of examples available inside of the editor.

Double click an example to open it up.

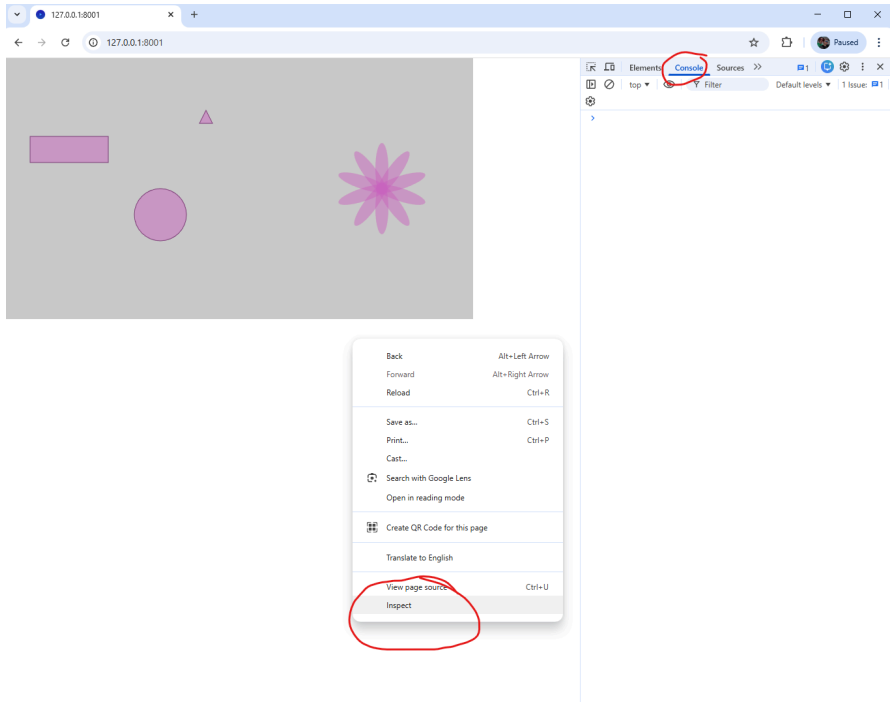


Narration

Running your code

Running the code (▶) opens up your web browser.

Javascript is a website language.



Right mouse button -> Inspect to see any errors or other console output.

Narration

What runs your code

Every Sketch starts with two functions: setup and draw.

```
function setup() {
    // initialize variables
    // create canvas
}

function draw(){
    // execute repeatedly (in a loop)
}
```

`setup()` runs *once* at the start of the sketch.

Narration

What runs your code

Every Sketch starts with two functions: setup and draw.

```
function setup() {
    // initialize variables
    // create canvas
}

function draw(){
    // execute repeatedly (in a loop)
}
```

`draw()` runs *forever*, repeating continuously.

Narration

A single message

`console.log()` allows us to write a message to the debugging console.

```
function setup() {
    console.log("Hello! This executes once.");
}
```

This is very useful for when you are trying to get rid of issues (bugs) in your code.

Narration

Let’s break this down

`console.log()` is the *function* that writes a message. The message it writes is the *argument* that is within `( ... )`.

`"Hello! This executes once."` is a *string*, a sequence of characters. Strings are surrounded by `"` to signify their start and end.

Narration

Message waterfall

Let us now move our `console.log()` to the `draw()` function instead.

```
1 function draw() {
2     console.log("Hello! This executes multiple times.");
3 }
```

Narration

Javascript syntax

- Lines end with a semicolon (`;`) to indicate the end of the instruction.
- Comments (`//`) are invaluable to explain the inner workings of the code to the reader.
- Multiple lines can be commented as well, via (`/*  */`)

Narration

Comments

```
1  function setup() {
2      // this should send a single message
3      console.log("Hello! This executes once.");
4  }
5
6  function draw(){
7      /* this executes repeatedly (in a loop)
8      TODO: maybe change this because it might be too much! */
9      console.log("Hello! This executes multiple times.");
10 }
```

Remember to comment your code to make it legible to someone else (and your future self!).

Narration

Making mistakes

Making mistakes is human. What happens if we forget the second `"` in our code?

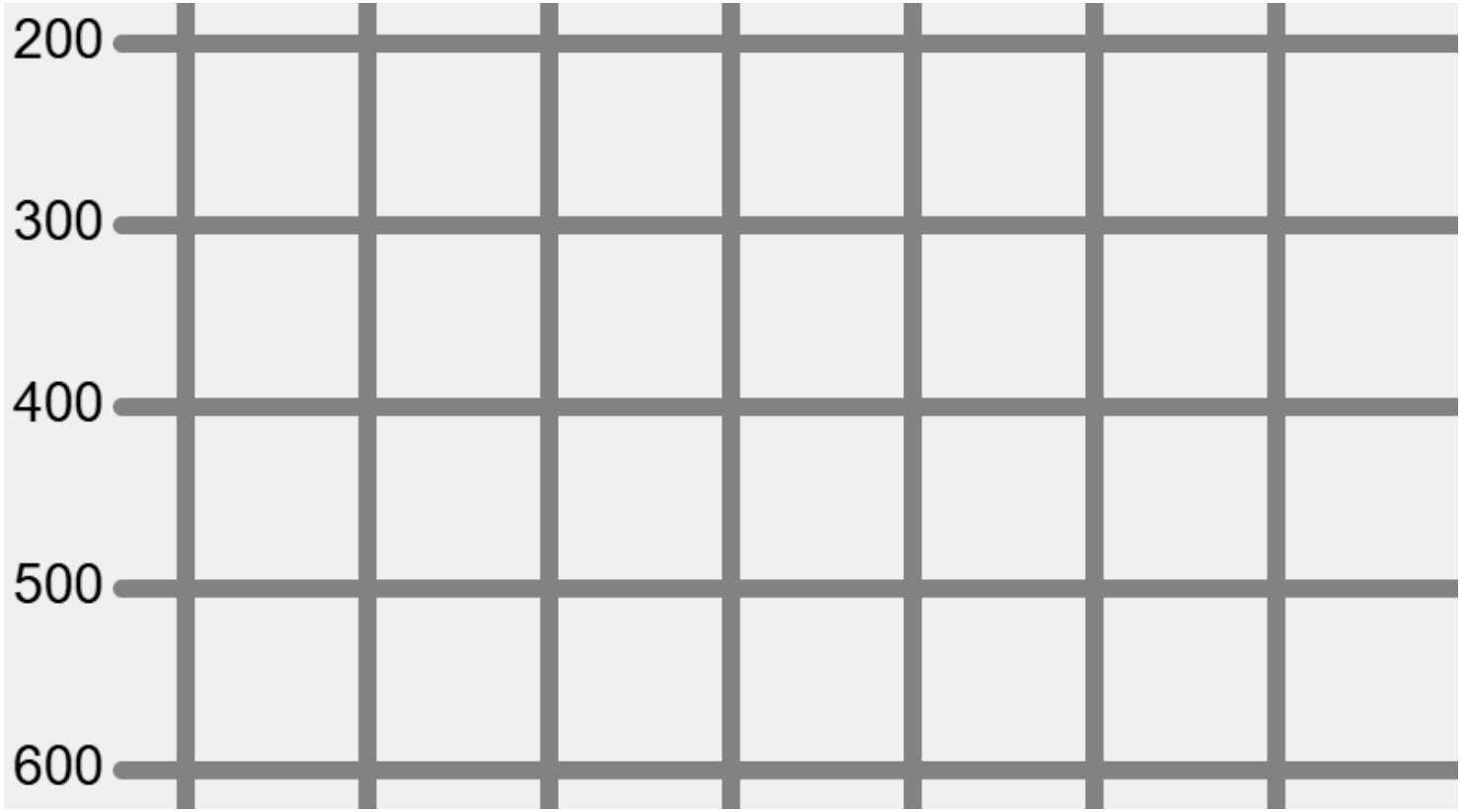
```
1 function setup() {
2     console.log("Hello! This executes once.);
3 }
```

The code does not know where the string ends, resulting in a `syntax error`.

Narration

The canvas

- Things are positioned on the canvas using *coordinates*, top-left is `(0, 0)`
- `x` is the horizontal position and `y` is the vertical position

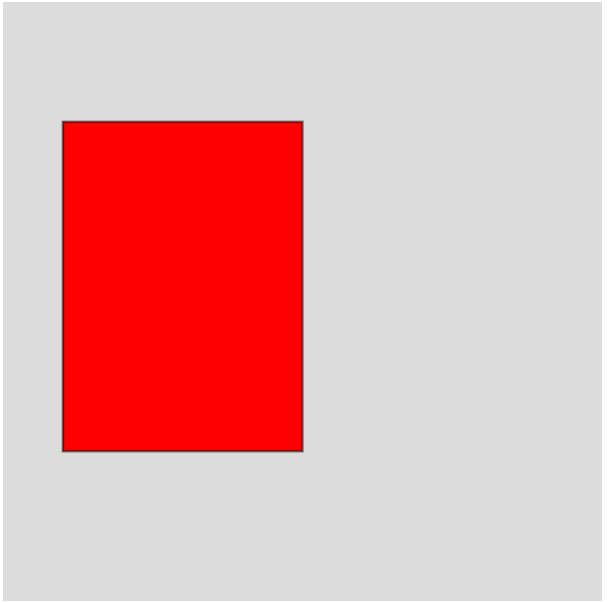


Narration

Defining the canvas

- In our `setup()` (only *once*), we define the size of the canvas
- `createCanvas()` requires *two* arguments:
  - `w` (width horizontal) and `h` (height vertical) specify the size of the canvas

```
1  function setup() {
2      createCanvas(500, 500);
3  }
4
5  function draw() {
6      background(220); // grey background
7      fill(255, 0, 0); // red colour
8      rect(50, 100, 200, 275); // rectangle from x=50 y=100 to x=200 y=275
9  }
10
```



Narration

Making a mark

- We can draw a point with the `point()` function with two arguments
- The point will be tiny; use `strokeWeight()` to define the thickness

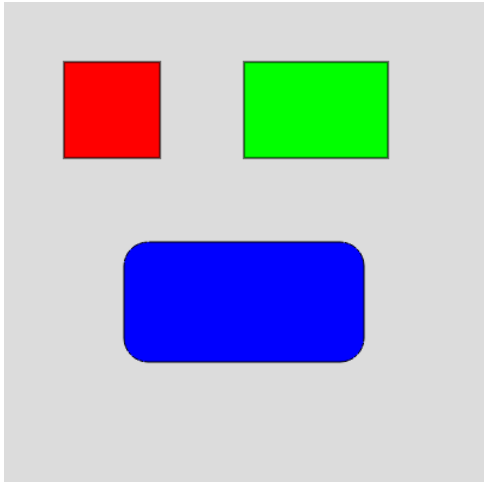
```
function setup() {
    createCanvas(400, 400);
    background(220); // grey background
    strokeWeight(10); // how thick the point is
    point(200, 200);
}

function draw() {
    // Static drawing
}

```


Narration

Drawing a rectangle

- Drawing rectangles with `rect()` requires 4 arguments
- You need to specify the top left `x` and `y` as well as the width `w` and height `h`

```
1  function setup() {
2      createCanvas(400, 400);
3      background(220); // grey background
4
5      // Draw a square
6      fill(255, 0, 0); // red colour
7      rect(50, 50, 80, 80);
8
9      // Draw a rectangle
10     fill(0, 255, 0); // green colour
11     rect(200, 50, 120, 80);
12
13     // Draw a rounded rectangle
14     fill(0, 0, 255); // blue colour
15     rect(100, 200, 200, 100, 20); // The 5th argument is *optional* and defines the corner
radius
```

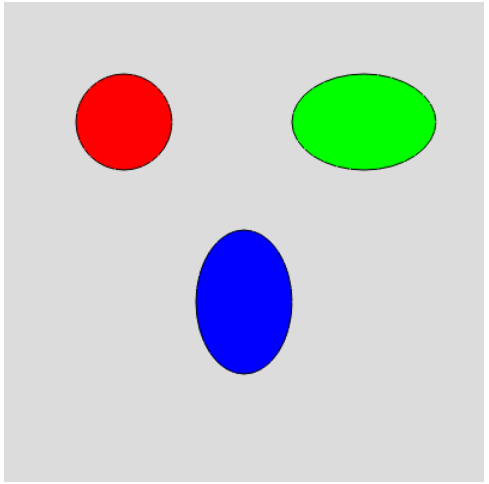


Narration

Drawing round shapes

- `circle()` draws a circle with 3 arguments: `x`, `y` and `size`
- `ellipse()` draws an ellipse with 4 arguments: `x`, `y`, `w`, `h`

```
1  function setup() {
2      createCanvas(400, 400);
3      background(220); // grey background
4
5      // Draw a circle
6      fill(255, 0, 0); // red colour
7      circle(100, 100, 80);
8
9      // Draw an ellipse
10     fill(0, 255, 0); // green colour
11     ellipse(300, 100, 120, 80);
12
13     // Draw another ellipse
14     fill(0, 0, 255); // blue colour
15     ellipse(200, 250, 80, 120);
16 }
```



Narration

A brief note about colour

- We have used `background()` and `fill()` so far to add colour
- These take three arguments to define the colour we want to draw

```
function setup() {
    background(red, green, blue); // Alter the colour of the background
    fill(red, green, blue); // Alter the colour of new shapes
}
```

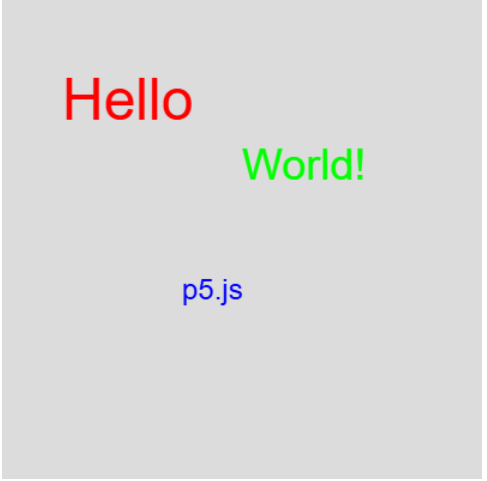
- The order is `red, green, blue`; a higher value indicates a stronger intensity of that color
- We will more thoroughly discuss colour in a future lecture

Narration

Moving our text to the canvas

- `text()` puts our text on the canvas, with the args being the text `string`, and position `x` and `y`
- We use `textSize()` before drawing to set the size of the text

```
function setup() {
    createCanvas(400, 400);
    background(220); // grey background

    // Set text properties
    textSize(48); // big text
    fill(255, 0, 0); // red colour
    text("Hello", 50, 100);

    // Different size and colour
    textSize(36); // medium text
    fill(0, 255, 0); // green colour
    text("World!", 200, 150);

```



Narration

What about interactivity?

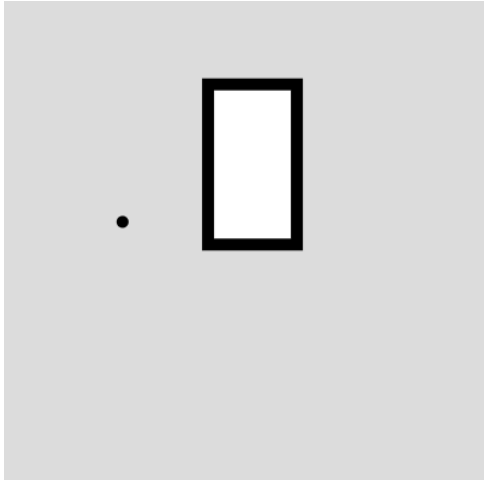
- So far we have discussed ways of drawing static shapes
- In preparation for the first exercise we will briefly discuss two ways of creating dynamic drawings
- We will use `random()` to generate random numbers
- `mouseX` and `mouseY` allows us to use the position of the mouse

Narration

Random

- `random()` generates a random number that we can use e.g. as an argument for drawing
- Two arguments: the first is the lowest number and the second the highest number it can generate

```
1  function setup() {
2      createCanvas(400, 400);
3      background(220); // grey background
4      strokeWeight(10); // how thick the lines are
5
6      point(random(50, 100), random(100, 200)); // Random point between 50-100 on x axis and
   100-200 on y axis
7      rect(random(150, 350), random(50, 150), random(50, 150), random(50, 150)); // Random
   rectangle position and size
8  }
9
10 function draw() {
```

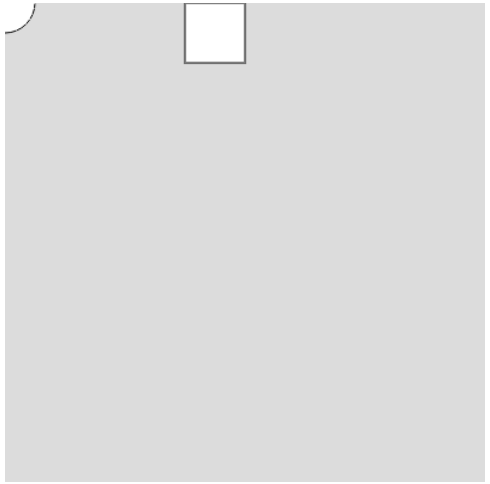


Narration

Mouse

- `mouseX` returns the horizontal `x` position of the mouse and `mouseY` the vertical `y` position
- As with `random()`, we can also use these numbers in math

```
1  function setup() {
2      createCanvas(400, 400);
3  }
4
5  function draw() {
6      background(220); // Grey background each frame. What happens if we don't do this here?
7      circle(mouseX, mouseY, 50); // Draw a circle at the mouse position
8      rect(mouseX + 150, mouseY, 50, 50); // Draw a rectangle 150 pixels to the right of the
   mouse
9  }
10
```



Narration

Resources

- p5.js reference documentation: https://p5js.org/reference/
- Coding Train: https://www.youtube.com/@TheCodingTrain/playlists
- Inspiration: http://www.generative-gestaltung.de/2/

Practice and do not be afraid to make mistakes :)

Narration