KØBENHAVNS
UNIVERSITET

Welcome to Kickstart

Welcome to Computer Science and the programming kickstart course! The exercises below will quickly get you up and running with programming P5 JavaScript. We have intentionally left some things unexplained as we want you to want you to experiment on your own.

Resources

There is a world of information and resources to help you learn p5.js. Check out the following:

Book McCarthy, Lauren, Casey Reas, and Ben Fry. Getting started with P5. js: Making interactive graphics in JavaScript and processing. Maker Media, Inc., 2015.

p5.js Web site The official website Click to go to the URL

Happy Coding A fun set of tutorials and examples for p5.js Happy Coding. Click

Kickstart Absalon You can find the book and other materials on the LMS.

Have Fun

Daniel Spikol & Martin Dybdal
ds@di.ku.dk

First Program

Type the following example in the Processing editor and press ▶-button:

```
function setup() {
  //set the canvas and background color
  createCanvas(400, 400);
  background(220);

  // Tree
  rect(55, 50, 10, 20);
  ellipse(60, 35, 30, 40);
}

function draw() {}
```
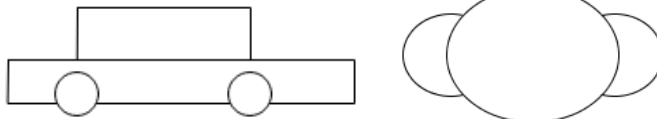
Then add:

```
  // powerplant
  rect(120, 50, 60, 30);
  rect(160, 20, 10, 30);
  triangle(120, 50, 136, 40, 136, 50);
  triangle(136, 50, 152, 40, 152, 50);
```

And:

```
# Vindmølle
  // windmill
  line(300, 50, 320, 51);
  line(300, 50, 289, 67);
  line(300, 50, 291, 32);
  line(300, 50, 300, 90);
```

Now try drawing a car and a cloud:

Colors

You are now going to colour the shapes. To do this, use the `fill(r, g, b)` function, which selects which colour to use for fill and text colour. It's all about calling `fill` in the right places! As arguments, you specify the amount of red (0-255), blue (0-255) and green (0-255).

Her er nogle grundfarver:

```
fill(255, 0, 0);   // red          fill(0, 0, 0);       // black
fill(0, 255, 0);   // green        fill(255, 255, 255); // white
fill(0, 0, 255);   // blue         fill(255, 255, 0);   // yellow
```
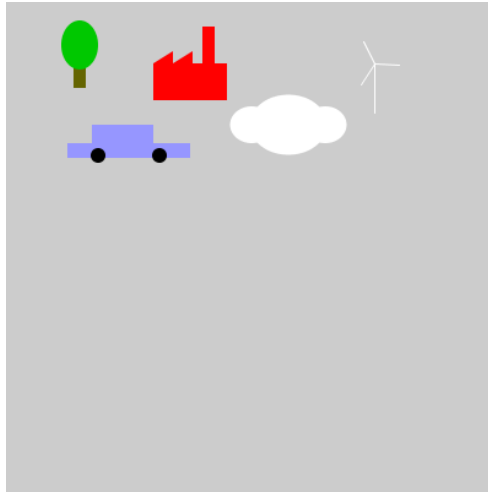
Optionally, find colours using an online color picker or RGB colour table. For example, search for "rgb color picker".

Lines and outlines

To specify the colour of strokes (e.g. `line`) and outlines, use `stroke(r, g, b)`. Also try the `noStroke()` function, to turn off outline drawing.

Daniel Spikol & Martin Dybdal
ds@di.ku.dk

Example

Here's an example of what it might look like after colouring:

Daniel Spikol & Martin Dybdal
ds@di.ku.dk

KØBENHAVNS
UNIVERSITET

Green City

Use what you've learned to make it a slightly nicer scene, with background, fore-ground and the extra details you think should be there. For example, I've drawn a road and moved the characters around.

Tip: To change the background color from gray you can use `rect` which you already know, but you can also use the `background(r, g, b)` command. It will delete everything and fill the screen with the specified colour.
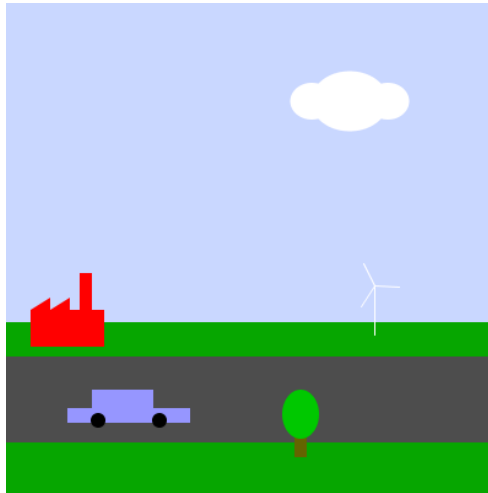
Remember to use comments to find your way around the code easily! Commenting code is good because it improves code readability and maintainability by explaining the purpose and functionality of the code to others and to your future self.

Variables

Create a new project ("File" -> "New") and immediately save it the project. Call it "Aquarium".

Type in this piece of code:

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

  //basic fish shape
  fishX = 150;
  ellipse(fishX, 200, 120, 75);
  triangle(fishX - 60, 200, fishX - 90, 170, fishX - 90, 230);
}

function draw() {

}
```

Try changing 150 to a different number in the `fishX` specification.
Now add the following:

```
  eyeSize = 15;
  ellipse(fishX + 30, 190, eyeSize, eyeSize);
```

Try changing the value of `eyeSize`.

Tasks:

You have now added a fish that can be moved just by changing one value.

- Color the fish

- Give the fish a fin that moves when you change `fishX`.

- Give the fish a pupil that moves when you change `fishX`.

- Create a new variable, `fishY`, that controls the y-position of the fish

Remember to save the project. We'll be working on it later.

KØBENHAVNS
UNIVERSITET

Green City continues

Switch to the electric car project and introduce variables to specify the location of the objects so that we can later animate these objects.

- Create a variable `carX` so the car can move forward and back

- Create a variable `cloudX` so the cloud can move back and forth

- Create a variable `treeX` so the tree can be moved horizontally

- Create a variable `treeY` so the tree can be moved vertically
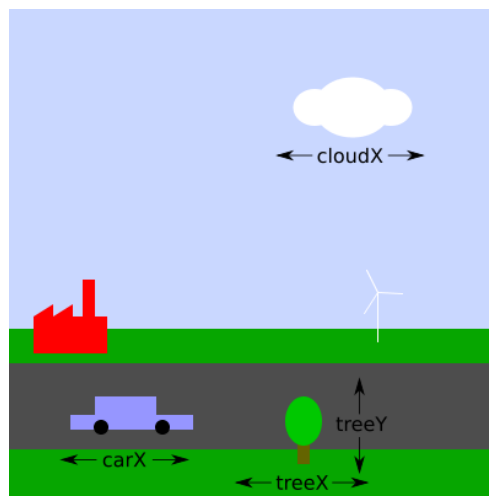
Aquarium Continues

Use what you've learned to expand your aquarium project; here's an example, but feel free to use your imagination! For example, one variable is used for the x-coordinate of the seaweed plant and another variable for the x-coordinate of the entire group of rocks (as a unit). An additional set of variables `fish2X`/`fish2Y` to control the location of the additional fish.
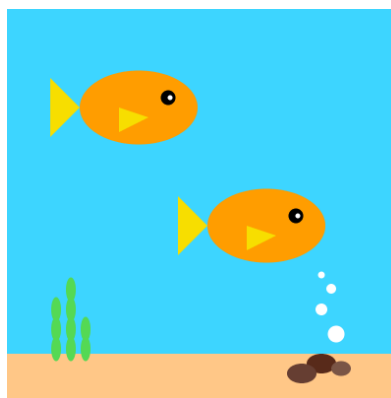
Functions

Functions allow you to reuse the same code in multiple places, name entire blocks, and add structure to code.

Open the Aquarium project. Add the following "fish-draw-function":

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

  // basic fish shape
  fishX = 150;
  ellipse(fishX, 200, 120, 75);
  triangle(fishX - 60, 200, fishX - 90, 170, fishX - 90, 230);

  // eye
  eyeSize = 15;
  ellipse(fishX + 30, 190, eyeSize, eyeSize);
}

function draw() {}
```

Now it's much faster to fill the aquarium with fish and we avoid copying code.

Task:

Create your own `drawFish(x, y)` function that draws your entire fish with colour, fins and eyes.

Green City continued

Over in the electric car project, you can also try writing a function to draw trees:

```
function setup() {
  createCanvas(400, 400);
  background(220);
  drawTree(160);
}

function draw() {
  // we are going to use the draw function soon!
}

function drawTree(treeX) {
  fill(100, 100, 0);
  rect(treeX - 5, 350, 10, 20);
  fill(0, 200, 0);
  ellipse(treeX, 335, 40, 50);
}
```

Structure your electric car project code with functions:

- Write a `drawCloud(x)` function that draws a cloud

- Extend the `drawTree(x)` function to also accept a y-coordinate

- Write a `drawCar(x)` function that draws a car

- Write a `drawPowerplant()` function and a `drawWindmill()` function that draws the power plant and the windmill, respectively. We won't need to move them around, so they don't need to not take coordinates as an argument.

Call all the functions in the `setup()` for now. For example:

```
drawTree(150, 235);
drawTree(240, 335);
drawPowerplant();
drawWindmill();
drawCar(50);
drawCloud(280);
```

Animation and Functions

Please create a new temporary project (you don't need to save it). Type in this piece of code:

```
// declare global variable accessible from any part of the code
let globalVarX = 50;

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

function draw() {
  background(220);
  background(255, 255, 255);
  fill(255, 0, 0);
  ellipse(globalVarX, 100, 30, 30);
  globalVarX = globalVarX + 1;
}
```

The `draw` function is automatically called 60 times per second!

Tasks:

- Try changing 50 in `globalVarX` to different number.

- Try changing the line x = x + 1 to x = x - 1 or to x = x + 5

- Try moving the call to `background` from `draw` to `setup` - what happens?

NOTE!!:

When using `setup`/`draw` it is not allowed to call drawing functions outside of `setup` and `draw`, everything must be moved into the two functions.

Aquarium continues

- *Rewriting the aquarium project to use `setup`/`draw`:*

  – Add empty `setup` and `draw` functions to the bottom of the program

  – Call `size(400, 400)` in `setup`

  – Call all the drawing functions in `draw`, incl. drawing of the background

- *Make the fish swim:*

  – Create two global variables `fish1X` and `fish2X` (before `setup`/`draw`)

  – Use the new variables as x-argument when you call `drawFish()`

  – *REMEMBER* the lines: `global fish1X` and `global fish2X`

  – Update the variables with $+1/-1$ inside the `draw` function


KØBENHAVNS
UNIVERSITET

Green City continues

- Create two global variables `carX` and `cloudX`

- Make the car drive to the right

- Make the cloud start outside the image on the right side and move to the left

Randomness

Create a brand new project and save it as "random_circles". Add the following code:

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

function draw() {
  // create 2 variables for different x y coordinates on the canvas
  let x = random(0, width);
  let y = random(0, height);

  // color the circle with a random monochrome shade
  fill(random(255));

  // draw the ellipse every frame
  ellipse(x, y, 30, 30);
}
```
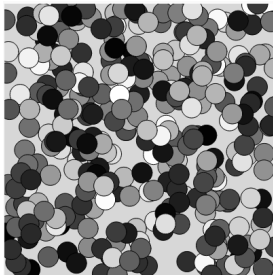

Tasks:

- Make the circles change size randomly

- Make the circles be drawn in random colours. Experiment until you find a colour scale that you think is nice.

Interaction with the Mouse

The position of the mouse can be read with the variables `mouseX` and `mouseY`. A kind of drawing program can be written like this:

```
function setup() {
  createCanvas(800, 800);
  background(255, 204, 0);
}

function draw() {
  fill(0, 0, 0);
  ellipse(mouseX, mouseY, 5, 5);
}
```

Keyboard input

Åbn tegneprograms-projektet og prøv at tilføje følgende nye funktion:

```
function setup() {
  createCanvas(400, 400);
  background(0);
}

function draw() {

}

function keyPressed(){
  background(0,0,255,255);
}
```

Start the program and press any key on the keyboard. To check for a specific key, the variable "`key`" can be read.

```
if (key == 'c') { // Check if the 'c' key is pressed
  background(255, 204, 0); // Set background color to a blue hue
```

New Project:Creative Programming

The task is now to create a more elaborate drawing program. Sign lines from all corners and midpoints on the sides. Use the variables `width` and `height`.

Remember to save the project!
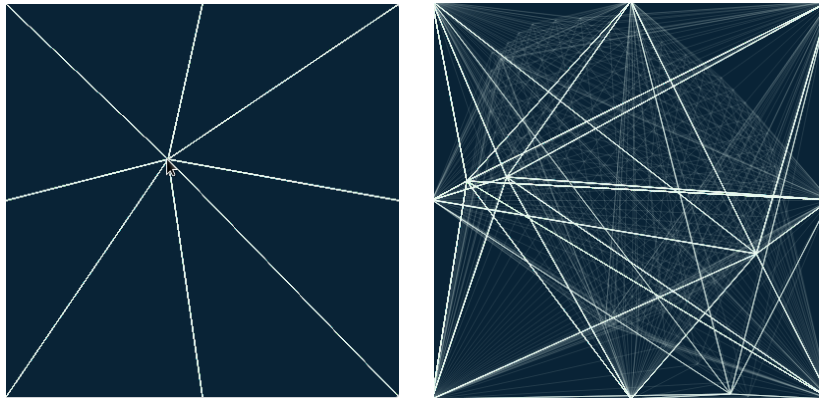
KØBENHAVNS
UNIVERSITET

Conditionals

With conditions, we can make things happen when special criteria are met met. Open the aquarium project and try inserting the following into it The `draw` function:

```
if (fish1X > 500) {
    fish1X = -40;
  }

  if (fish2X < -50) {
    fish2X = 200;
  }
```

What happens?

Change direction

By making a variable that contains the direction in which the fish swims, we can change the direction when it reaches the sides.

- Define a global variable `fish1XVelocity` and set it to 1

- Change `fish1X = fish1X + 1` to `fish1X = fish1X + fish1XVelocity`

- Remove the previous `fish1X` condition

- Add these conditions:

```
if (fish1X > 400){
    fish1XVelocity = -1
    }
if (fish1X < 0){
    fish1XVelocity = 1
    }
```
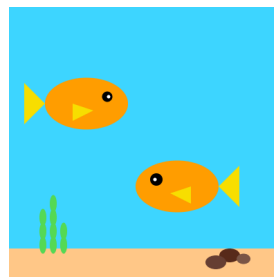
Change appearance

Try changing the `drawFish` function to strip the direction with the fish as an argument and use a condition to draw the fins and eyes differently depending on which direction the fish is swimming.

```
function drawFish(x, y, eyeSize, velocity){
    ... daw body ...

    # Swim to the right
    if (velocity >= 0){
        ...draw fins and eyes ...
        }

    # Swim to the left
    if (velocity < 0){
        ...draw fins and eyes ...
        }
}
```

Finish the aquarium project

Now, we are almost done with the aquarium project. Add, if necessary, more elements. E.g. bubbles that emerge from the bottom and swim towards the surface.

More exercises about conditions

Open the Green City project and do the following:

- Make the cloud move back to the start and pass by again and again, each time it hits the edge

- Make the car turn over at both ends.

Tasks:

The car must stop when the battery is empty

- Define a global variable `carBattery` and set it to 100

- Decrease it by 0.1 each time `draw` is called

- Display the battery status using the `text(string, x, y)` function. Remember to convert the number to a string using `str()`.

- If the battery is empty (`<= 0`) the car must stop (set velocity to 0)

Add a `keyPressed()` function and make it possible to charge the electric car when you press 'C':

- Add `0.3` to `carBattery` every time 'C' is pressed on the keyboard

- If `carBattery` exceeds 100, set it to 100 so you can't charge more than 100%

Changing wind speed

The variable `frameCount` counts how many times `draw` is run since the program started. Enter this in the draw function i the electric car project:

```
fill(0, 0, 0);
text(frameCount, 350, 20):
if (frameCount % 60 == 0){
    print(frameCount)
    }
```

Since `draw` is executed at *60 frames per second*, will The `print` function will be executed every second.

We can also use that in the electric car project to update the wind speed every second.

- Create a global windSpeed variable

- Update windSpeed with a new random value every second

- Display the wind speed using `text()`

- Create a global variable `powerplantOn`

- Make it possible to turn the power plant on and off with a press of 'p'

- Draw clouds of smoke over the power plant when it is on

- Create a variable `co2emission` and add a bit as long the power plant is on. Display the value with `text()`

Introduction to Lists and For Loops

With `for loops`, we can avoid duplicated code and iteratively perform an operation, such as drawing 3 circles on a screen. Start a new project and insert the following code:

```
// Initialize Lists
let xs = [30, 50, 70, 90, 110, 130];
let ys = [30, 50, 70, 90, 110, 130];

function setup() {
  createCanvas(400, 400);
  background(0);
}

function drawCircle(x, y) {
  noStroke();
  bubble_R = random(256);
  bubble_G = random(256);
  bubble_B = random(256);
  fill(bubble_R, bubble_G, bubble_B);
  circle(x, y, 25);
}

function draw() {
  for (var i = 0; i < xs.length; i++) {
    drawCircle(xs[i], ys[i]);
  }
}
```

What happens when you run the code? Explain here how a loop produces the result, and give yourself time to understand the logic.

More with Lists and For Loops

The following code snippet has a few changes: We now use a loop to generate our list!
Paste the code into a new project and consider the differences in the various loops.

```javascript
// Initialize Lists
let circleList = [];

function setup() {
  createCanvas(400, 400);
  background(0);
}

function drawCircle(x, y) {
  stroke(255, 255, 255);
  noStroke();
  bubble_R =random(255);
  bubble_G =random(255);
  bubble_B =random(255);
  fill(bubble_R, bubble_G, bubble_B);
  circle(x, 200, 50);
}

function createCircleList(n) {
  for (var i=0; i < n; i++) {
    append(circleList, i*60);
  }
}

function draw() {
  createCircleList(10);
  for (var i = 0; i < circleList.length; i++) {
    drawCircle(i);
  }
}
```
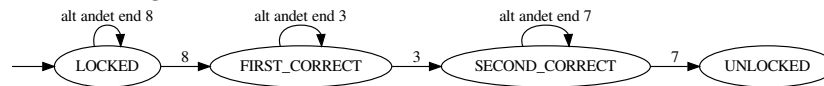
- Use loops and lists to move your fish or make more fish in different locations. Plus in Green City if you want.

KØBENHAVNS
UNIVERSITET

Finite state machines

As the first example of a state machine, we will create one electronic combination lock, such as those used for access control doors.
Tilstandsdiagram:

Download the Processing.py code for the combination lock here: `http://kortlink.dk/ufdh` and copy it into a new Processing project.

- Test the combination lock and watch as the state changes. If you did not know the ransom (password), how many attempts does it require? to guess?

- Change the code so that the default is 524 instead

- Try changing the code to follow this diagram instead, where incorrect presses reset the state to start:
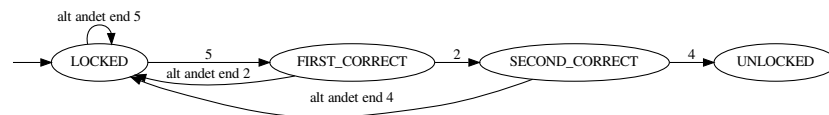
- Draw an extended state diagram of the combination lock with an extra mode, so 4 digits are required. Next, add the extra state in the code.

Daniel Spikol & Martin Dybdal
ds@di.ku.dk

Automatic relock after 2 seconds

Let's extend the lock so that the door automatically locks again after 2 seconds, which corresponds to 120 frames:
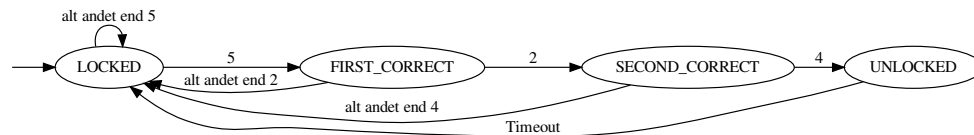
- Create a global variable "`timer`" and set it to 0

- Set the `timer` variable to 120 as soon as the lock is unlocked, it will say when it changes `lockState` to `"UNLOCKED"`.

- Count down with the timer in each frame (add the following to the `draw` function):

```
global timer
if timer > 0:
    timer = timer - 1
```

- Når timeren er talt helt ned, skal låsen åbnes. I `draw`-funktionen skal I nu tjekke om vi er i tilstanden `"UNLOCKED"` og timeren samtidig er talt ned til 0. Indsæt følgende i `draw`:

```
if lockState == "UNLOCKED":
    if timer == 0:
        lockState = "LOCKED"
```