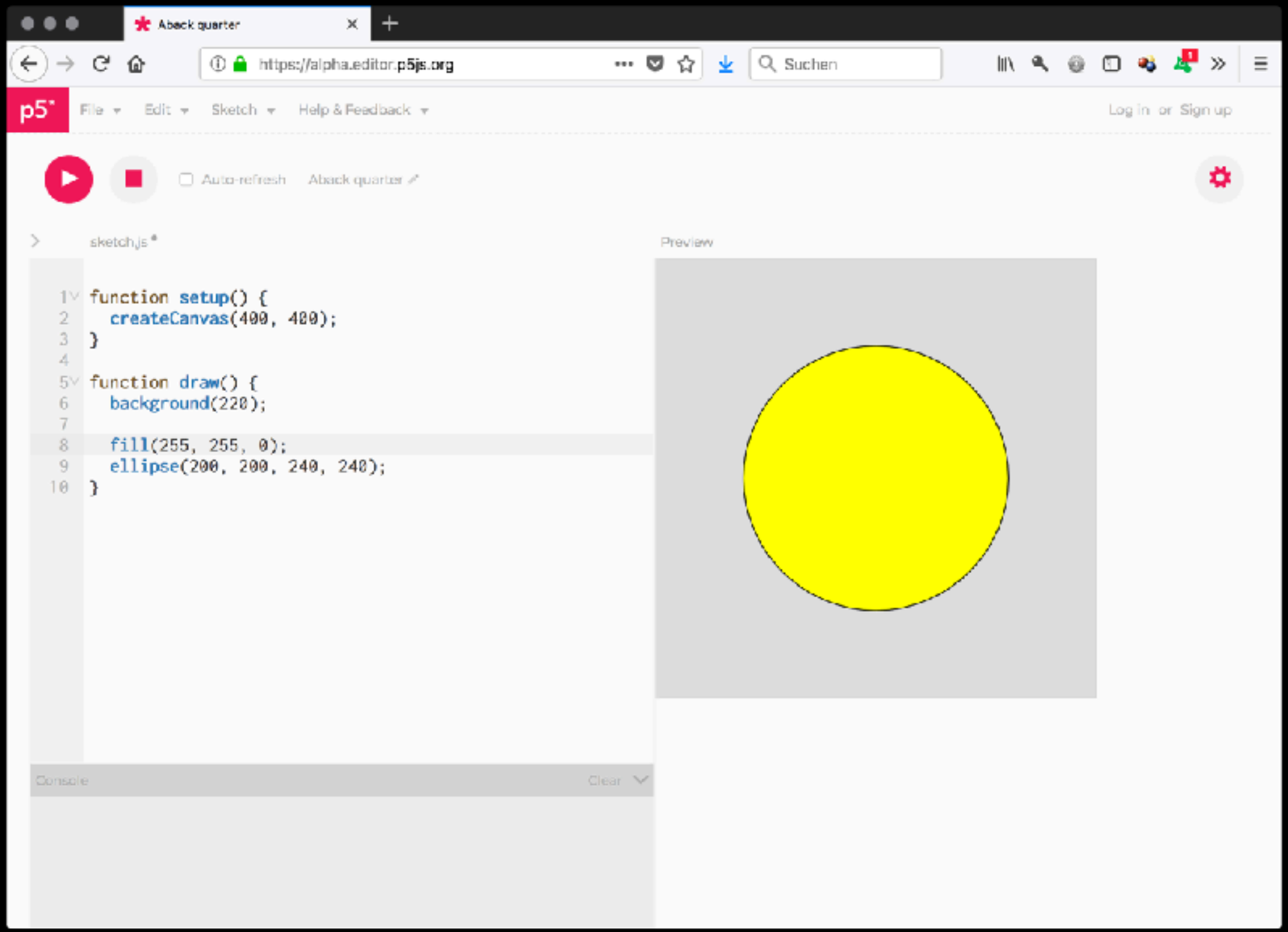
Hello.

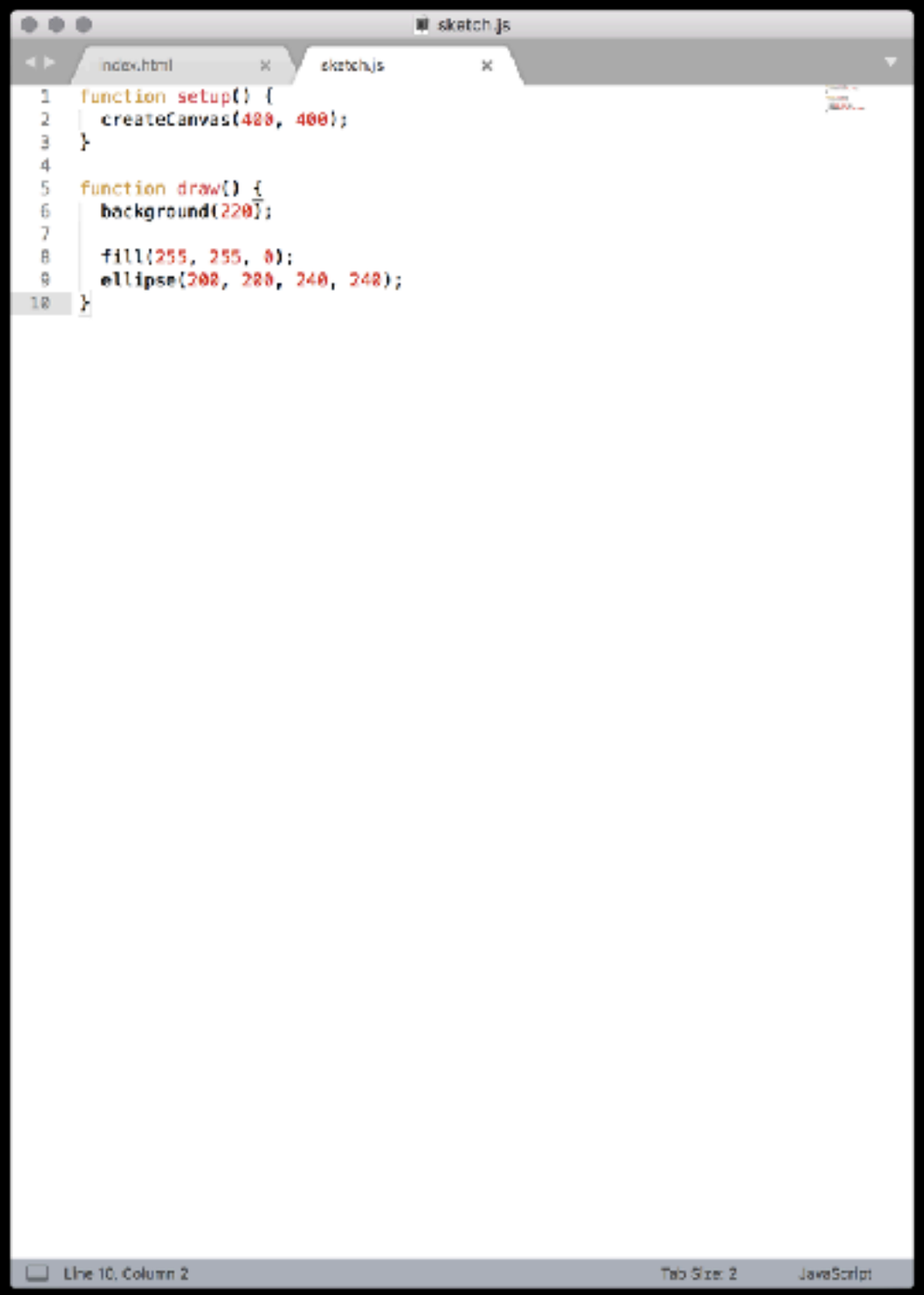
Introduction to **p5.js**

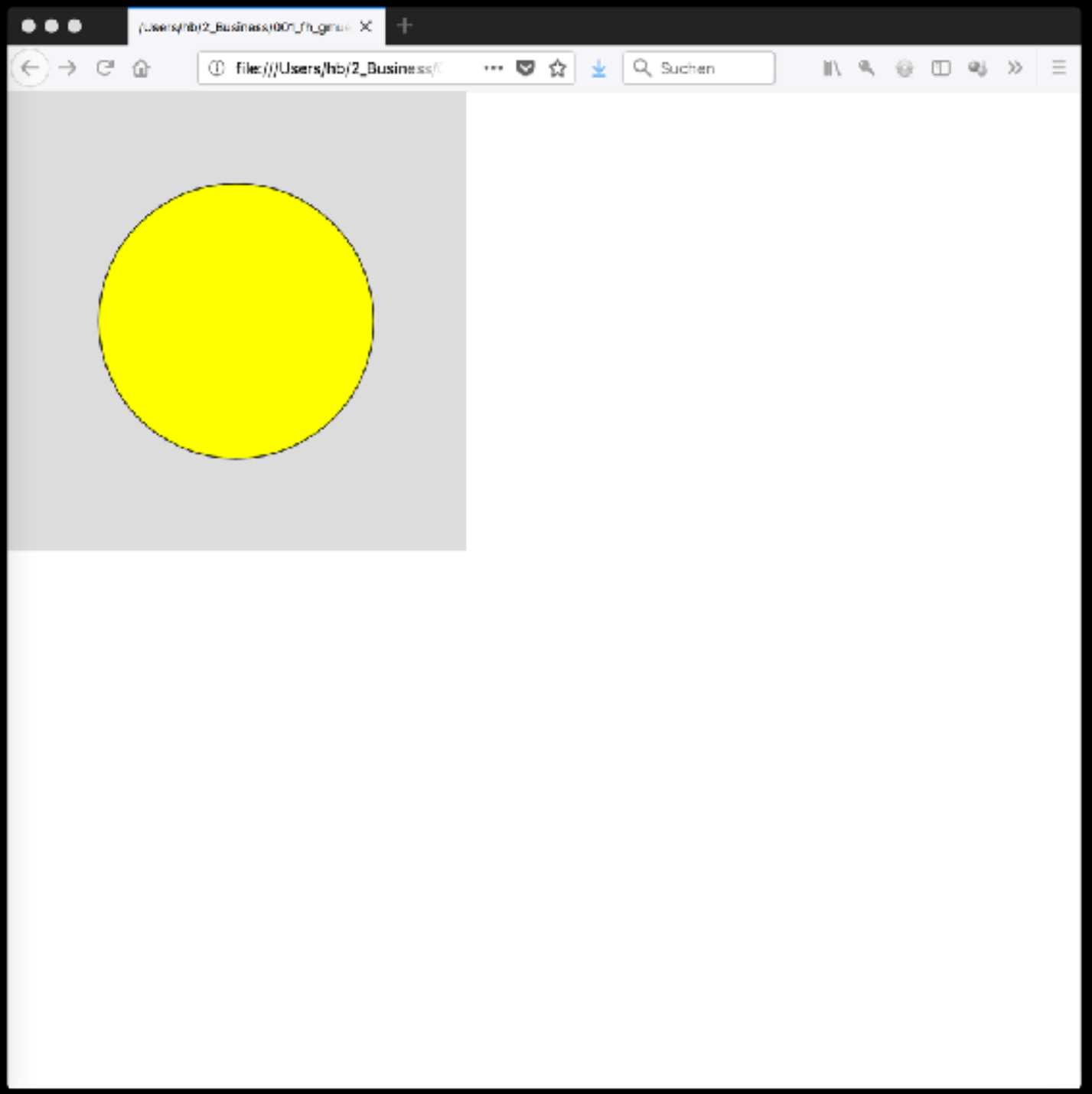
p5.js

p5.js


Aback quarter
https://alpha.editor.p5js.org
Suchen
p5*
File
Edit
Sketch
Help & Feedback
Log in or Sign up
Auto-refresh
Aback quarter
sketch.js *
Preview
function setup() {
createCanvas(400, 400);
}
function draw() {
background(220);
fill(255, 255, 0);
ellipse(200, 200, 240, 240);
}
Console
Clear

p5.js


sketch.js
index.html
sketch.js
function setup() {
createCanvas(400, 400);
}
function draw() {
background(220);
fill(255, 255, 0);
ellipse(200, 200, 240, 240);
}
Line 10, Column 2
Tab Size: 2
JavaScript


Suchen

Syntax

Syntax

The principles by which sentences are constructed in a particular language.

Syntax

Draw a rectangle at the position x:20 y:30 with the size of 50 x 80 pixels.

Syntax

```
rect(20, 30, 50, 80);
```

Syntax

```
rect(20, 30, 50, 80);
```

Syntax

```
rect(20, 30, 50, 80);
```

Syntax

```
rect(20, 30, 50, 80);
```

Name of the function

Parameters

Syntax

```
rect(20, 30, 50, 80);
```

x y w h

Name of the function

Parameters

Syntax

```
rect(20, 30, 50, 80);
```

Let's try that, shall we?

There are lots of functions.
Some are for drawing something, ...

Syntax

```
rect( x, y, b, h );
ellipse( x, y, b, h );
line( x1, y1, x2, y2 );
point( x, y );
```

... some are for changing the style of drawing.

Syntax

stroke(greyvalue);

fill(greyvalue);

strokeWeight(w);

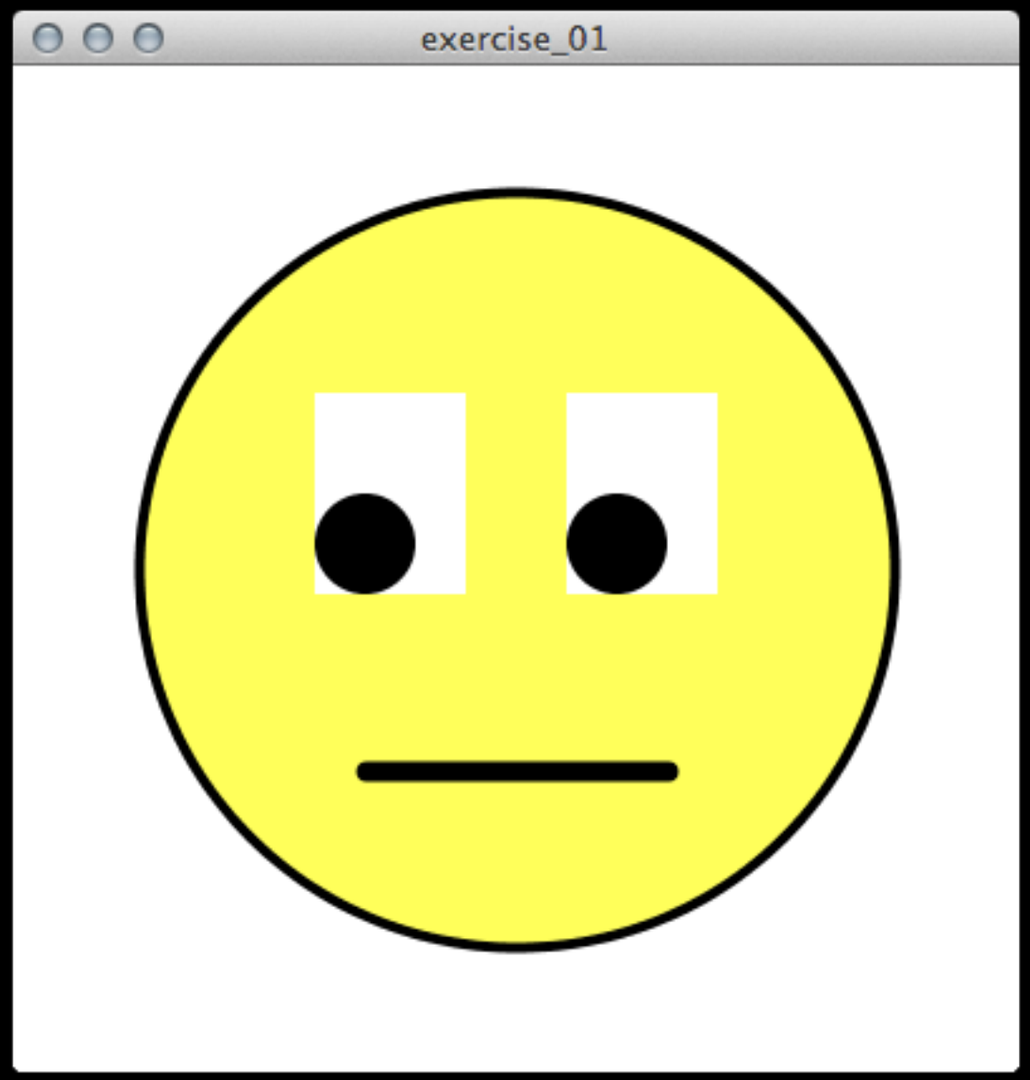
Syntax

```
createCanvas( w, h );
```

Syntax

p5js.org/reference/

Let’s try this:



Comments

Comments

```
rect(20, 30, 50, 80);
```

Comments

draw rectangle

rect(20, 30, 50, 80);

Comments

```
// draw rectangle
rect(20, 30, 50, 80);
```

Comments

```
// draw rectangle
rect(20, 30, 50, 80);
```

Comment Indicator (for one line)

Comments

```
/* draw
rectangle */
rect(20, 30, 50, 80);
```

Comments

```
/* draw
rectangle */
rect(20, 30, 50, 80);
```

Comment Indicator (for multiple lines)

Variables

Variables

```
rect(20, 30, 50, 80);
```

Variables

```
var h = 80;
rect(20, 30, 50, h);
```

Variables

```
var h = 80;
```

Variables

```
var h = 80;
```

Keyword for defining a variable

Name of the variable

Value

Variables

```
var h = 100;
rect( 0, 0, 100, h );
h = 80;
rect( 0, 0, 100, h );
```

Keyword „var“ is necessary only
at the first usage of the variable.

Variables

```
var h = 100;
h = h * 2;
rect( 0, 0, 100, h );
```

Calculation with variables

Tip

Print variables

```
var x = 100;
console.log( x );
console.log( “x is “ + x );
```

Operators

Operators

+ Addition

- Subtraction

* Multiplication

/ Division

% Modulo (Rest of a divison)

= Assignment

Operators

++ Plus one

-- Minus one

Operators

++ Plus one

-- Minus one

```
var number = 10;
number++;  // number = 11
number--;  // number = 10
```

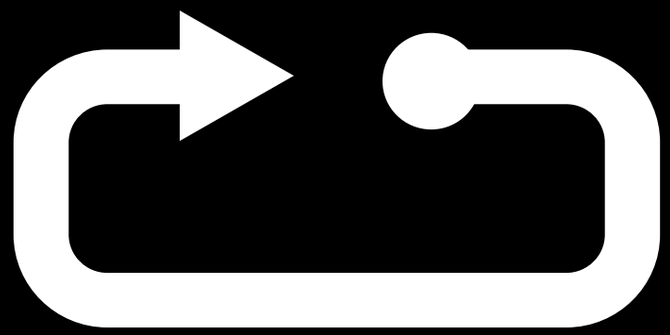
Two nice variables

// Width of the canvas
width

// Height of the canvas
height

Loops

Loops



Loops

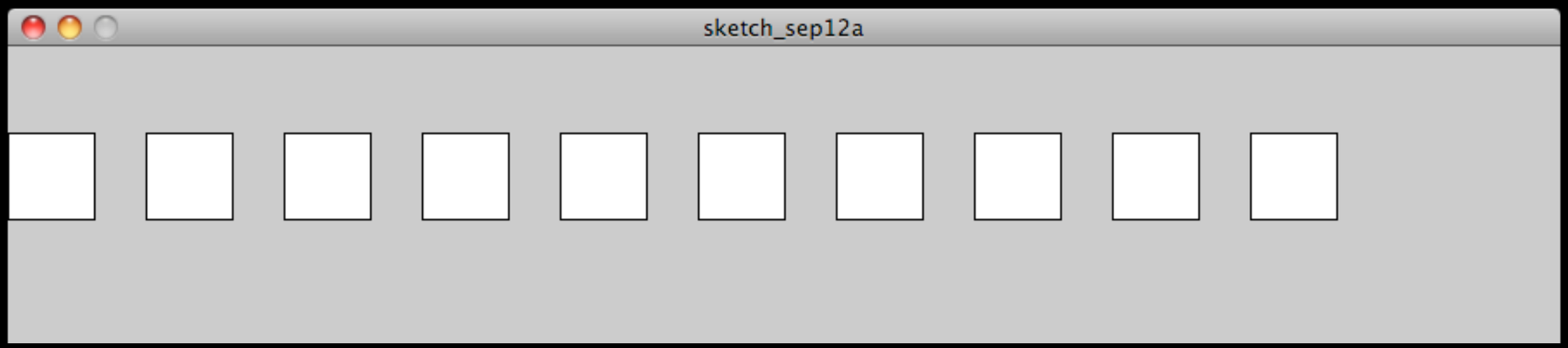
while loop

Loops

```
var i = 0;

while (i < 10) {
    rect(i * 80, 50, 50, 50);
    i = i + 1;
}
```

Loops

```
var i = 0;
while (i < 10) {
  rect(i * 80, 50, 50, 50);
  i = i + 1;
}
```



Loops

```
var i = 0;
while (i < 10) {
    rect(i * 80, 50, 50, 50);
    i = i + 1;
}
```

Condition

Command block

Loops

for loop

Loops

```
var i=0;
while ( i<10 ) {
    rect( i * 80, 50, 50, 50 );
    i=i+1;
}
```

Initializing the counter

Condition for the loop

Operation with the counter

Loops

```
var i=0;

while ( i<10 ) {
    rect( i * 80, 50, 50, 50 );
    i=i+1;
}
```

```
for ( var i=0; i<10; i=i+1 ) {
    rect( i * 80, 50, 50, 50 );
}
```

Initializing the counter

Condition for the loop

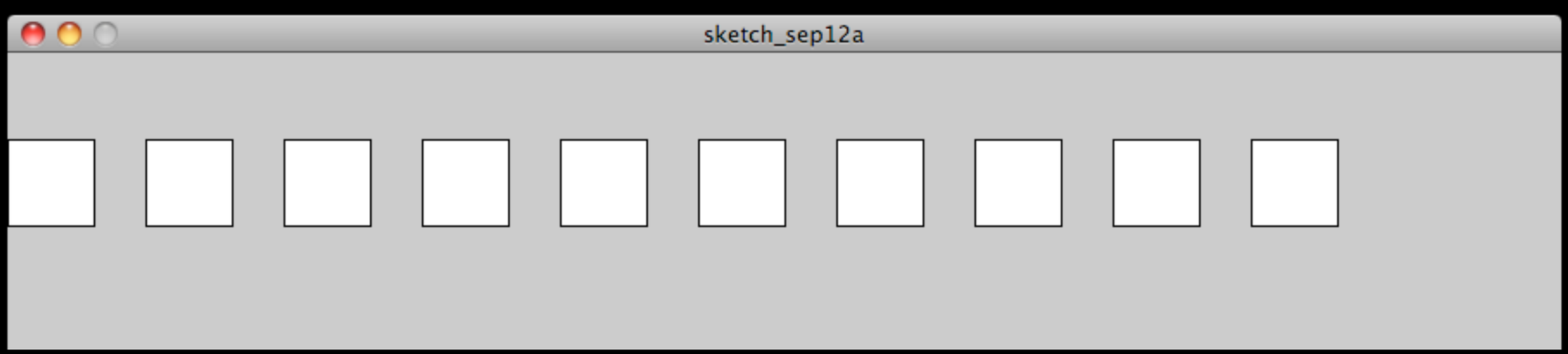
Operation with the counter

Loops

```
for ( var i=0; i<10; i=i+1 ) {
    rect( i * 80, 50, 50, 50 );
}
```

Loops

```
for ( var i=0; i<10; i=i+1 ) {
    rect( i * 80, 50, 50, 50 );
}
```



Loops

```
for ( var i=0; i<10; i=i+1 ) {
    rect( i * 80, 50, 50, 50 );
}
```

Initializing the counter

Condition for the loop

Operation with the counter

Loops

```
for ( var i=0; i<10; i=i+1 ) {
    rect( i * 80, 50, 50, 50 );
}
```

Code block

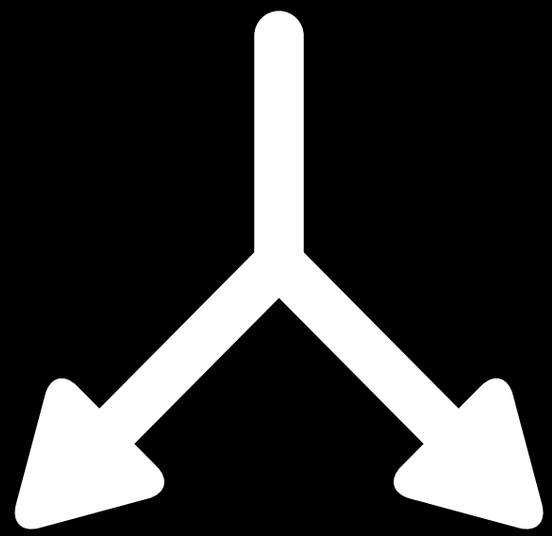
Loops

```
for ( var i=0; i<10; i=i+1 ) {
    rect( i * 80, 50, 50, 50 );
}
```

Using the counter

Branches

Branches



Branches

```
if (Condition) {
    Commands
}
```

Branches

```
var x = 10;
if (x > 5) {
    rect(0, 0, 50, 50);
}
```

Branches

```
var x = 10;
if (x > 5) {
    rect(0, 0, 50, 50);
}
```

Condition

Commands

Branches

if (a < b)	less
if (a <= b)	less or equal
if (a == b)	equal
if (a >= b)	greater or equal
if (a > b)	greater
if (a != b)	not equal

Branches

```
if (a < b && c < d)          and
if (a < b || c < d)          or
```

Animation

Animation

A movie is a sequence of frames

Animation

setup and draw

Animation

setup

is called once at the beginning of the sketch.

Animation

draw

is called in every frame.

Animation

```
function setup() {
    size(500, 500);
}

function draw() {
    ellipse(0, 0, 50, 50);
}
```

An interesting Variable

/* Frame counter
is incremented by 1 in
every frame */

frameCount

Interactivity

Interactivity

Mouse and Keyboard

Interactivity

Mouse

Interactivity

mouseX and mouseY

Interactivity

mouseX and mouseY

```
function draw() {
   rect( mouseX, mouseY, 50, 50 );
}
```

Interactivity

mouseIsPressed

```
function draw() {
   if ( mouseIsPressed == true ) {
      rect( mouseX, mouseY, 50, 50 );
   }
}
```

Interactivity

Keyboard

Interactivity

keyIsPressed

```
function draw() {
  if ( keyIsPressed == true ) {
    rect( mouseX, mouseY, 50, 50 );
  }
}
```

Interactivity

key

```
function draw() {
    if ( keyIsPressed && key == ‘r‘ ) {
       rect( mouseX, mouseY, 50, 50 );
    }
}
```

Transformations

Transformations

Manipulating the coordinate system

Transformations

translate()
rotate()
scale()

push()
pop()

Transformations

rotate()

rotates the coordinate system around the actual origin point

Transformations

rotate(angle)

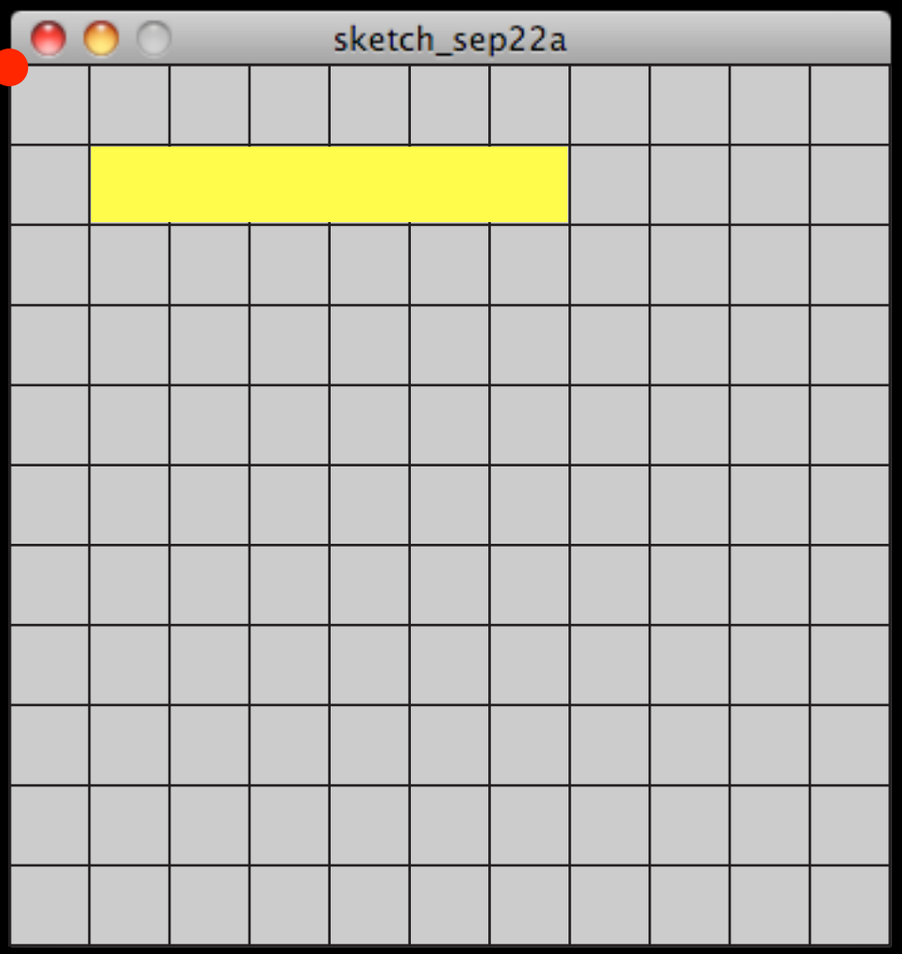
Transformations

rotate(angle)

angle value in radians
(0 – TWO_PI)

Transformations

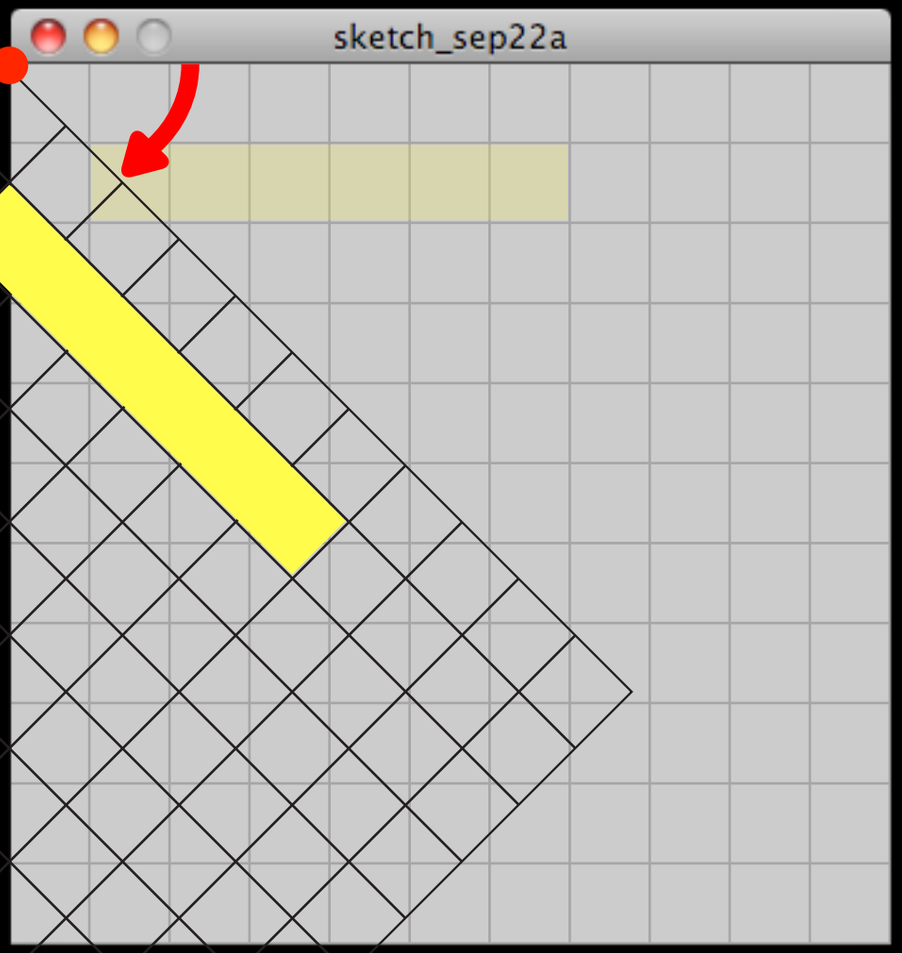
rotate(angle)



rect(25,25,150,25);

Transformations

rotate(angle)




```
rotate(radians(45));

rect(25,25,150,25);
```

Transformations

translate()

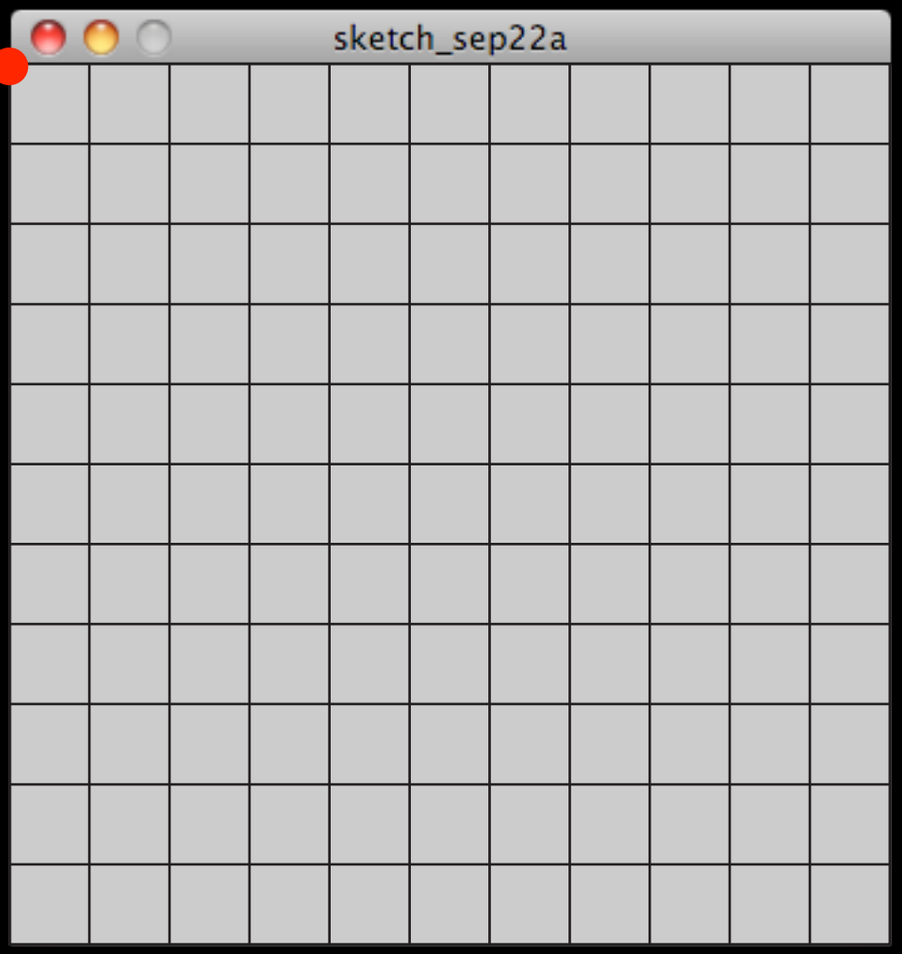
moves the coordinate system

Transformations

translate(x, y)

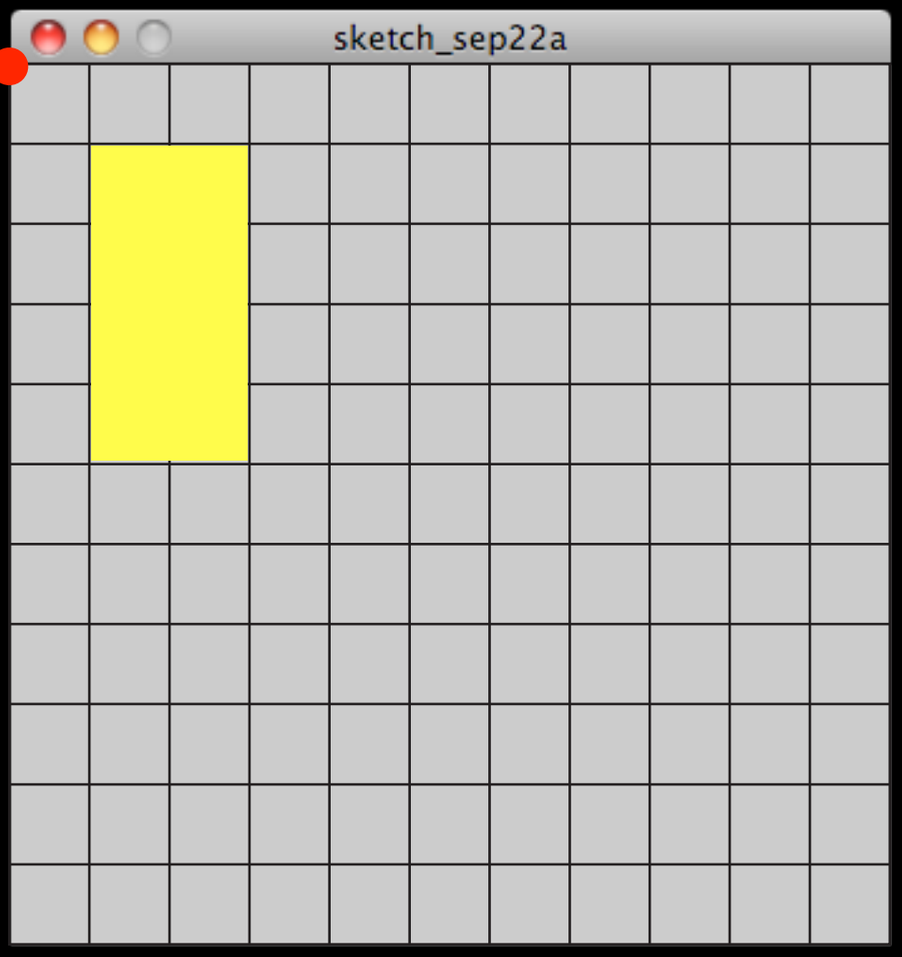
Transformations

translate(x, y)



Transformationen

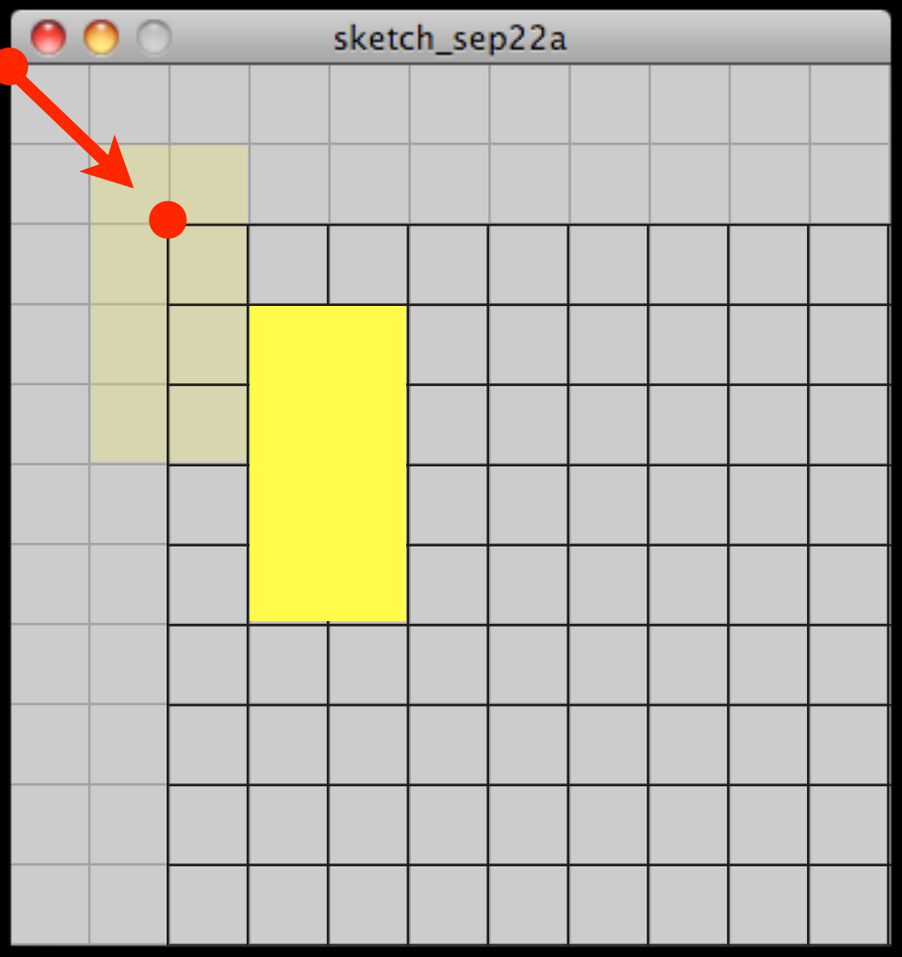
translate(x, y)



rect(25,25,50,100);

Transformations

translate(x, y)

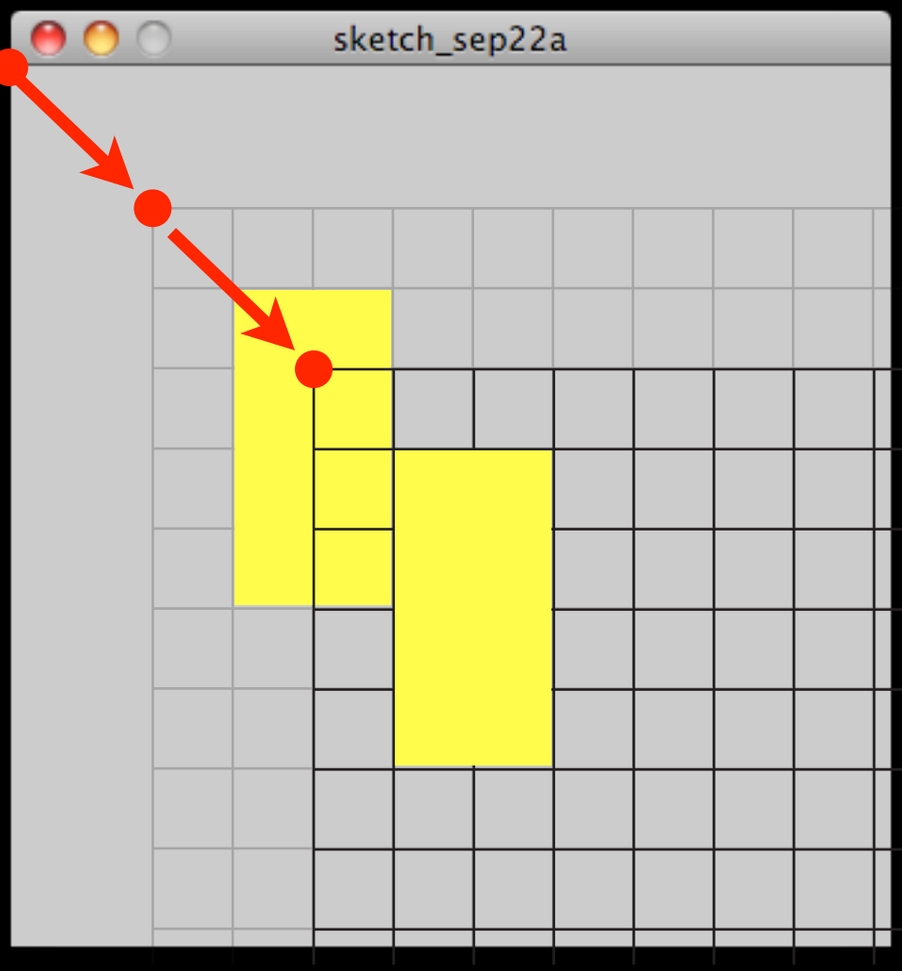


translate(50, 50);

rect(25,25,50,100);

Transformations

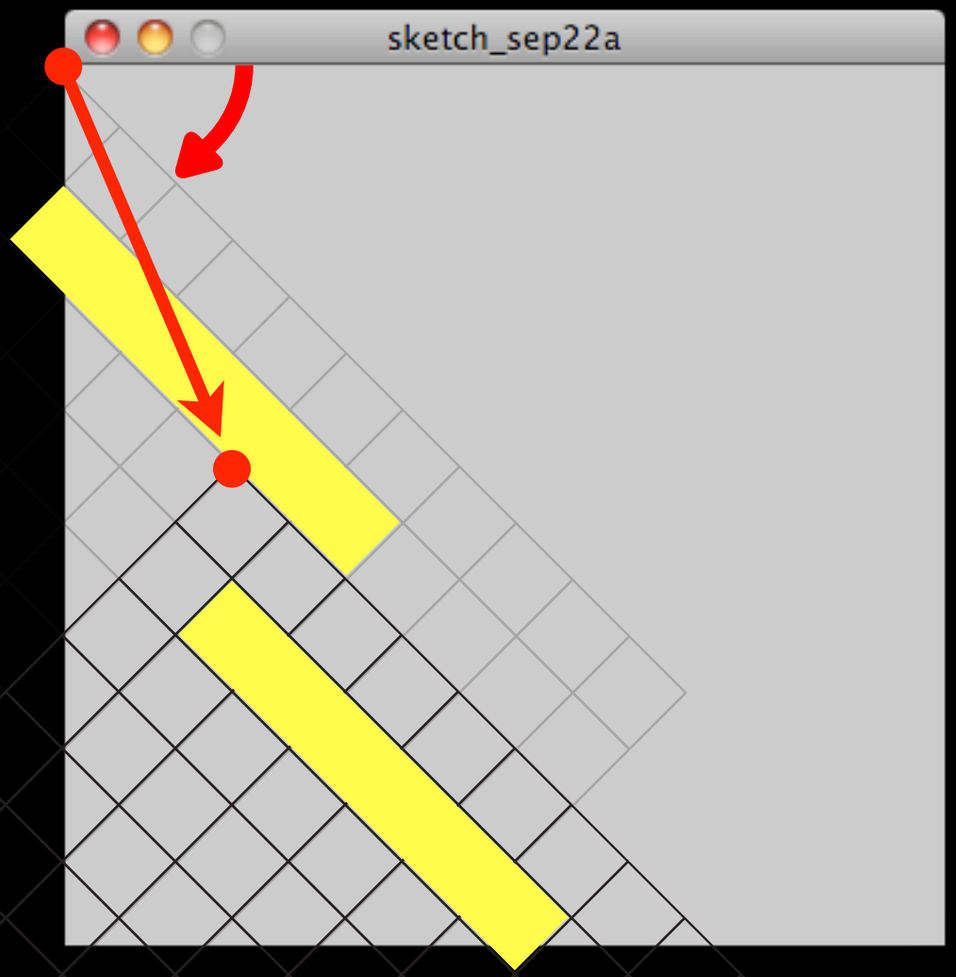
translate(x, y)




```
translate(50, 50);
rect(25,25,50,100);
translate(50, 50);
rect(25,25,50,100);
```

Transformations

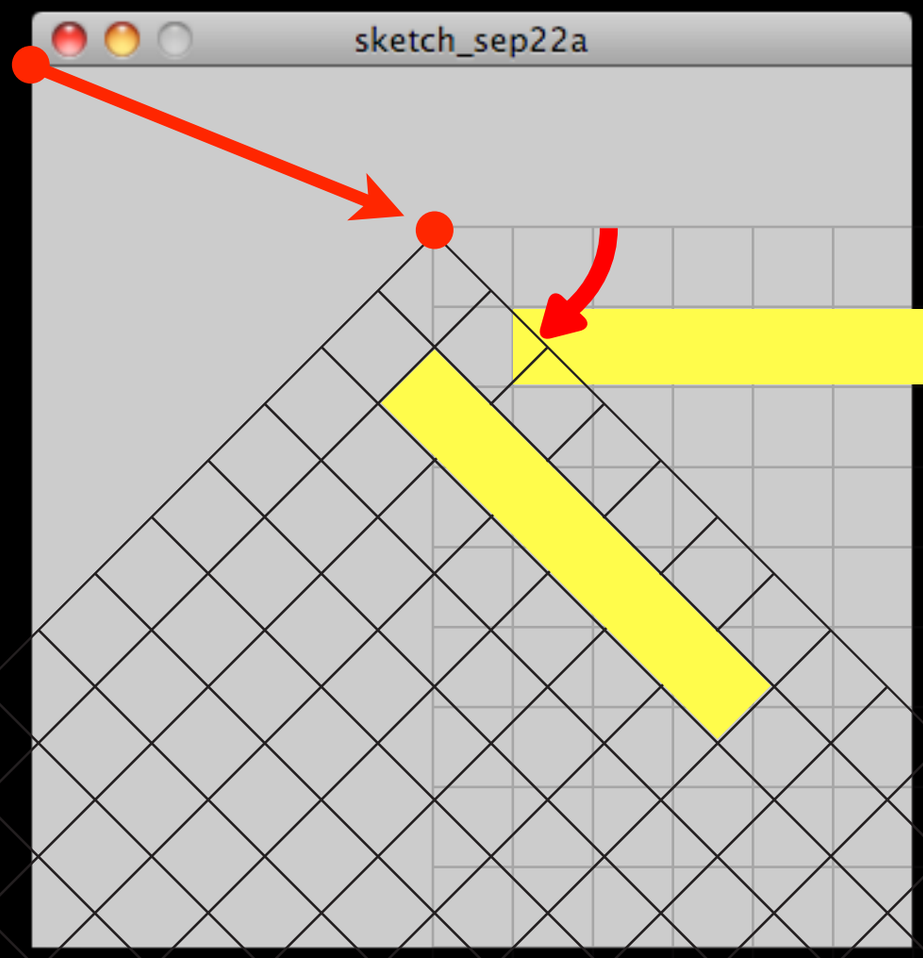
rotate(winkel)




```
rotate(radians(45));
rect(25,25,150,25);
translate(125, 50);
rect(25,25,150,25);
```

Transformations

note the difference when changing the order:




```
translate(125, 50);
rect(25,25,150,25);
rotate(radians(45));
rect(25,25,150,25);
```

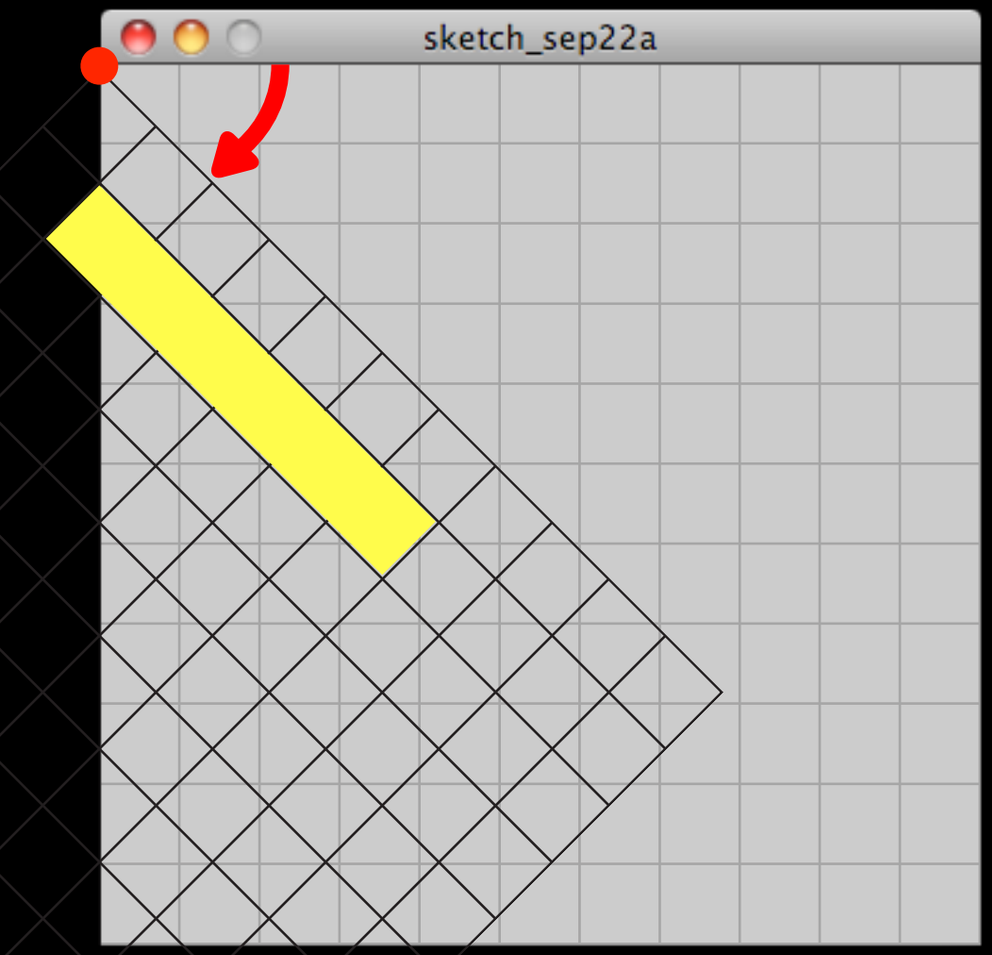
Transformations

push()

pop()

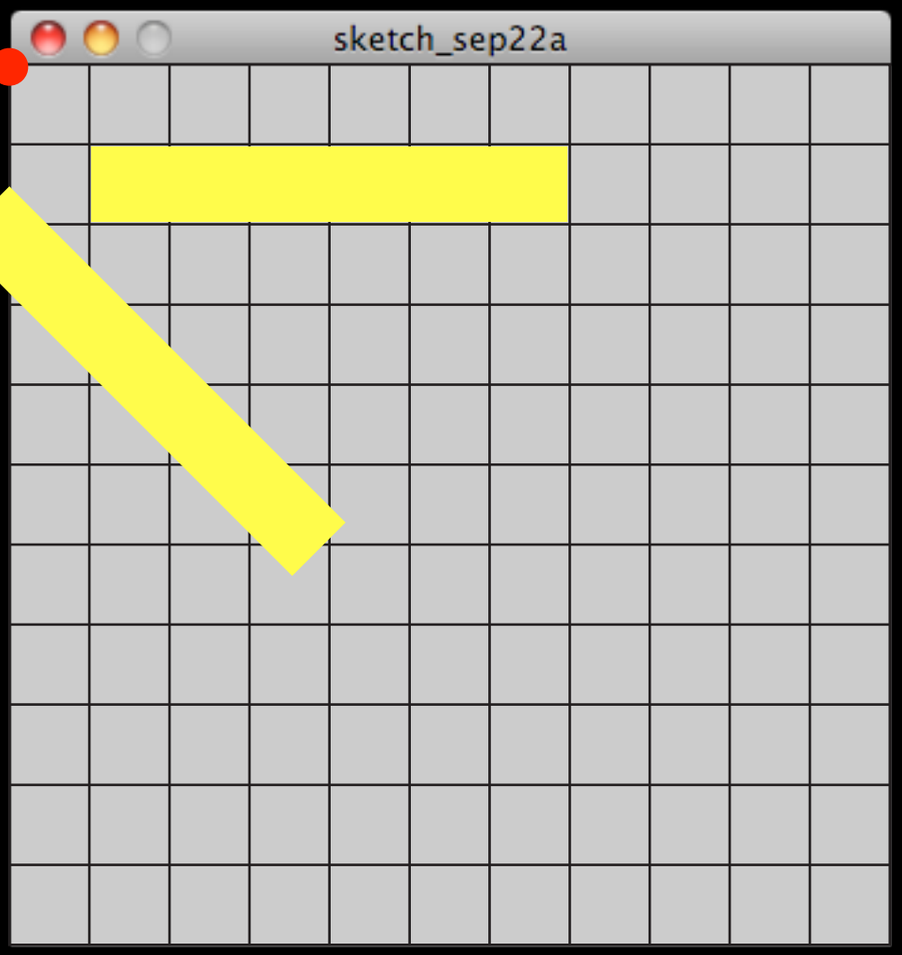
save and load the
coordinate systems

Transformations




```
push();
rotate(radians(45));
rect(25,25,150,25);
pop();
```

Transformations




```
push();
rotate(radians(45));
rect(25,25,150,25);
pop();
rect(25,25,150,25);
```

Functions

Functions

Reusing parts of the code

Functions

```
function abc() {
    // some lines of code
}
```

```
abc();
abc();
```

Name of the function

Code block

Calling the function

Functions

```
var abc = function() {
    // some lines of code
}

abc();
abc();
```

Functions

```
function abc(x, y) {
    circle(x, y, 100, 100);
}

abc(20, 50);
abc(40, 40);
```

Defining parameters

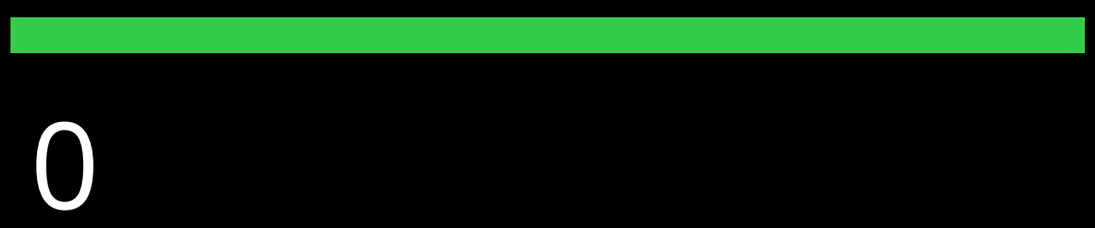
Parameter values

Functions

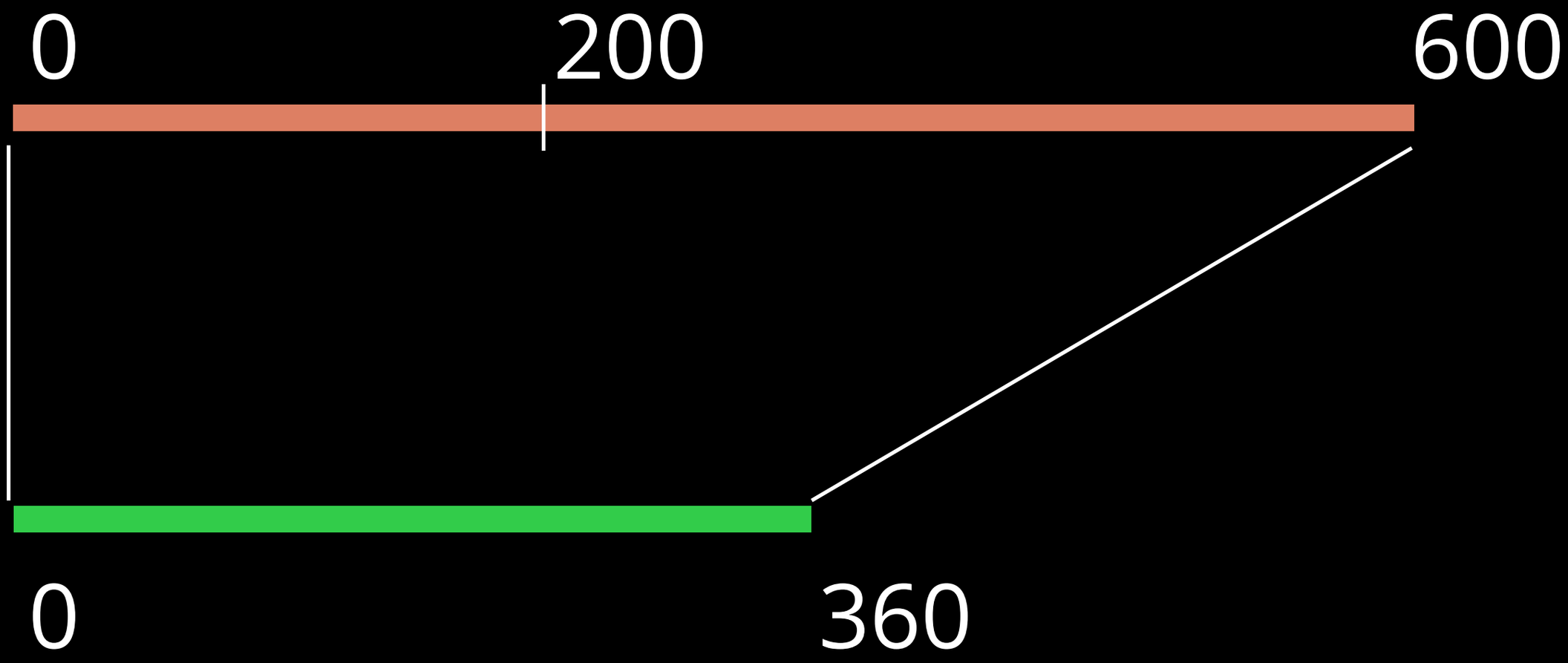
```
function average(a, b) {
    return (a + b) / 2;
}

var result = average(40, 30);
// result will be 35
```

Mapping

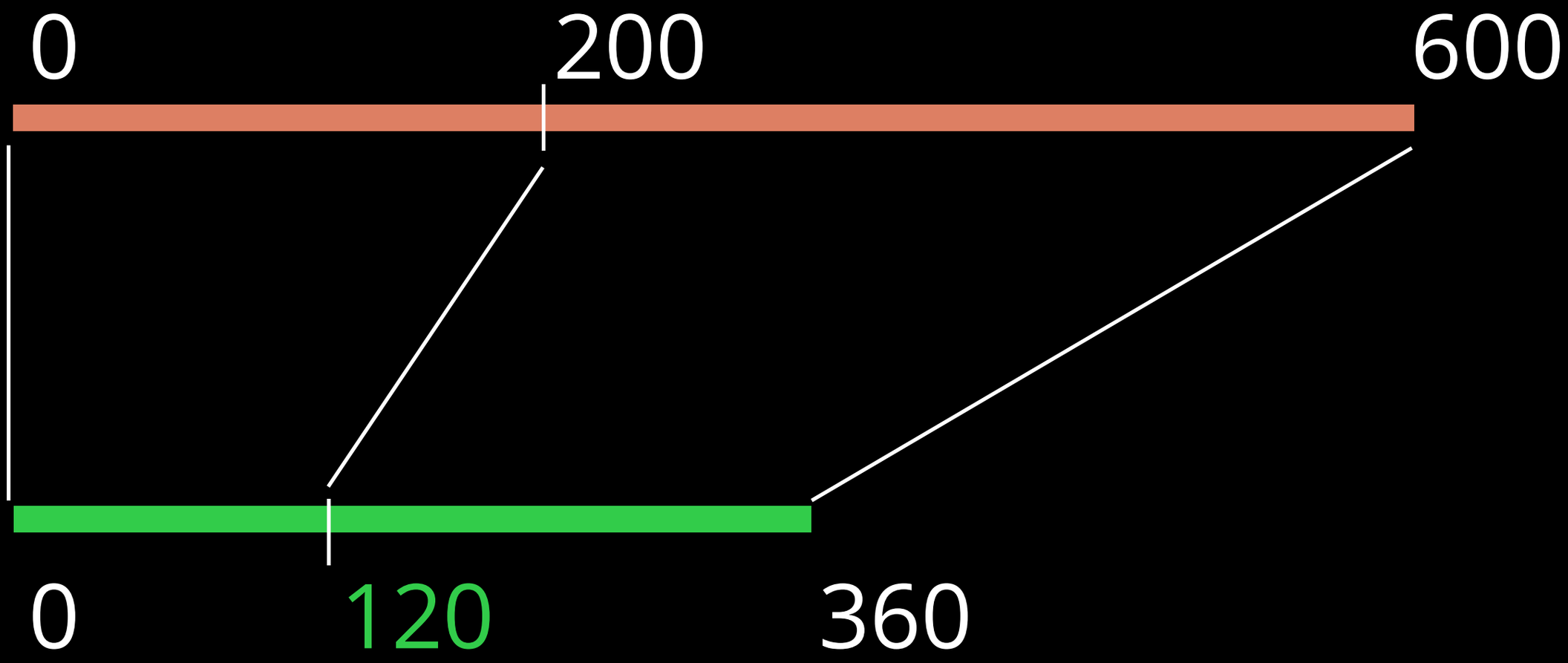
Mapping



Mapping



Mapping



Mapping

```
void draw() {
   var a = map( mouseX, 0, 600, 0, 360);
   rotate( radians(a) );
   rect( 300, 300, 50, 50 );
}
```

Value in the source range

Minimum and maximum of source range

Minimum and maximum of target range

Arrays

Arrays are lists of values. You can create them in several ways.

Arrays

```
float[] a = {4, 7, 3};
```

Data type

Name of the variable

Value

Arrays

```
float[] a = {4, 7, 3};
```

Get values:

a[0] → 4

a[1] → 7

a[2] → 3

a[3] → Error!

Arrays

```
float[] a = new float[100];
```

Data type

Name of the variable

Create an empty array with 100 values

Arrays

```
float[] a = new float[100];
```

Set values:

```
a[51] = 17;
println(a[51]); → 17.0
println(a[0]); → 0.0
```