Introduction to p5.js

p5.js

p5.js

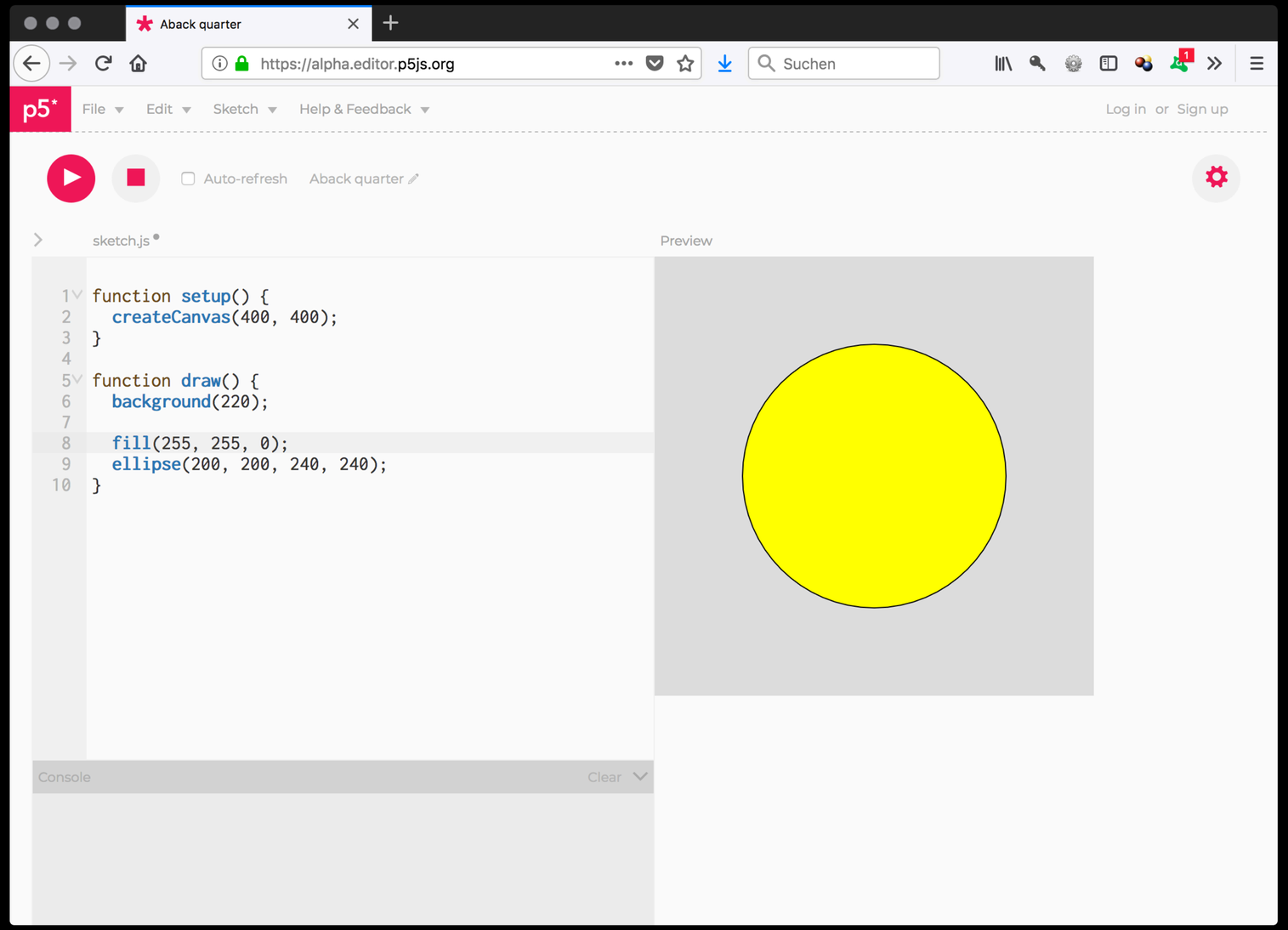
Javascript is a scripting language mainly used in a web browser.

p5.js

Javascript is a scripting language mainly used in a web browser.

p5 is a library for Javascript to make programming of graphical stuff easier.

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

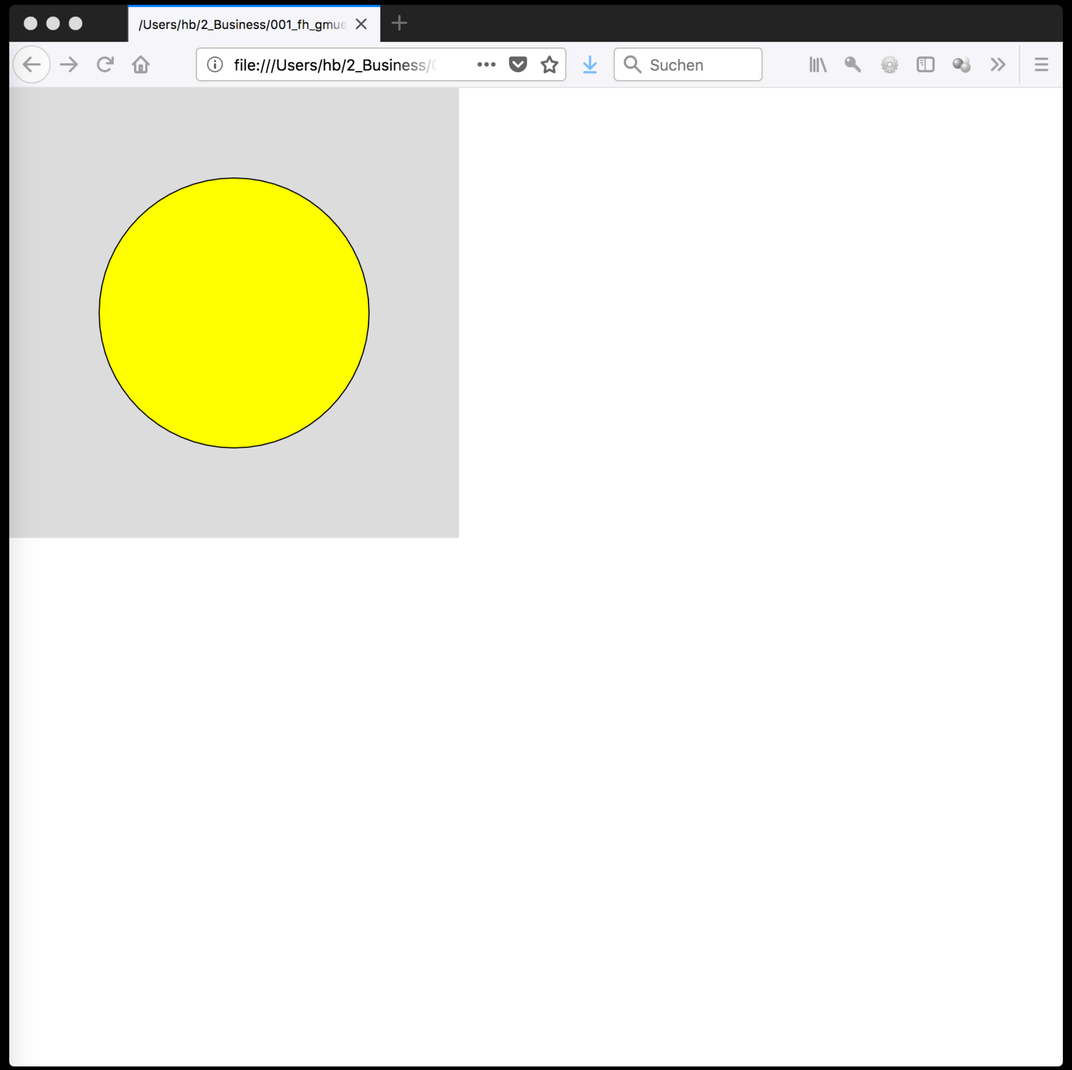
p5.js

sketch.js

index.html | sketch.js

```
1  function setup() {
2    createCanvas(400, 400);
3  }
4
5  function draw() {
6    background(220);
7
8    fill(255, 255, 0);
9    ellipse(200, 200, 240, 240);
10 }
```

Line 10, Column 2 Tab Size: 2 JavaScript



Syntax

Syntax

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

Syntax

Draw a rectangle at the position x:20 and 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?

Let’s try that, shall we?

But first we have to set up our working environment . . .

Syntax

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 );

background( greyvalue );
background( r, g, b );
```

Syntax

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

Syntax

```
stroke( greyvalue );
fill( greyvalue );
strokeWeight( w );
```

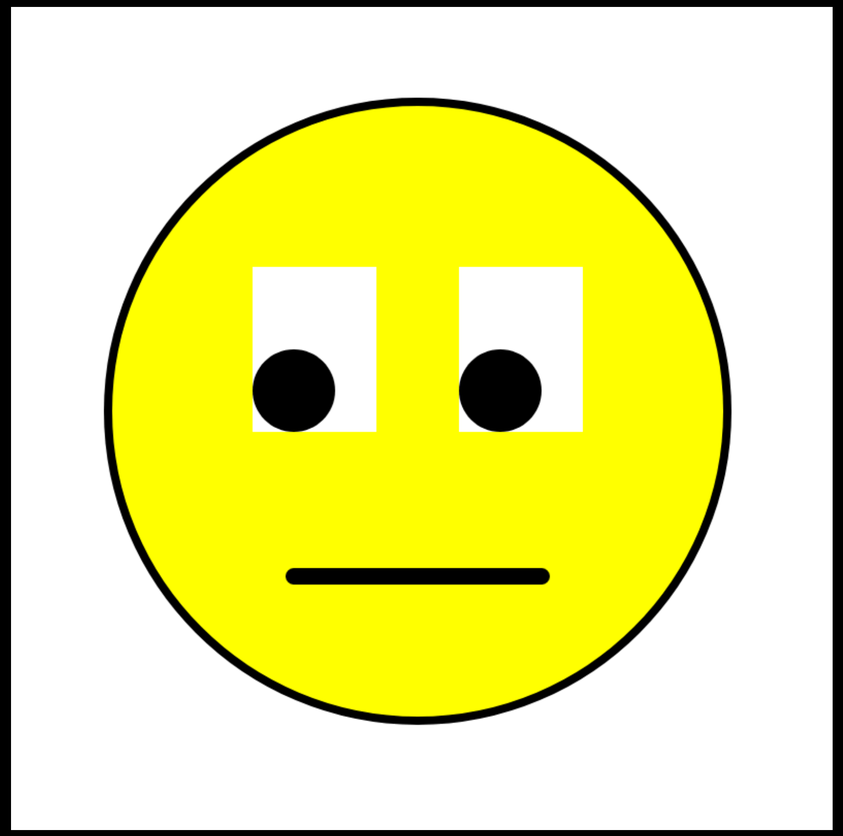
Syntax

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

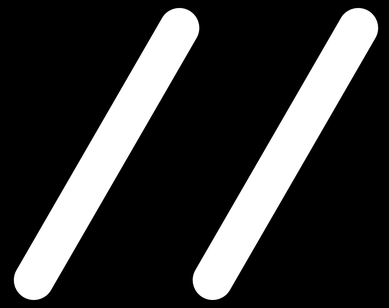
Syntax

p5js.org/reference/

Let’s try this:



Comments



Comments

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

Comments

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

This wouldn’t work

Comments

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

Comments

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

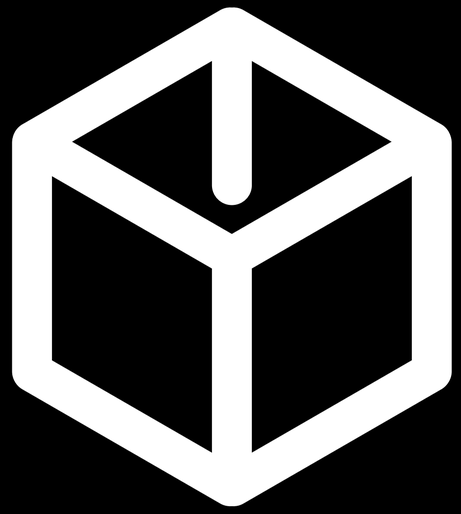
Comment Indicator (for one line)

Comments

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

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 );
```

Use the keyword „var“ only at the first appearance of the variable.

Variables

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

Calculation with variables

Variables

Tip: printing variables

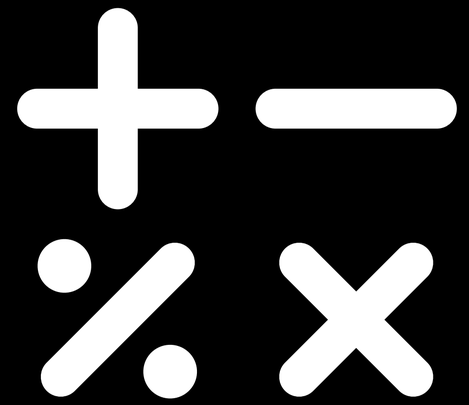
```
var x = 100;
console.log( x );
console.log( “x is “ + x );
```

p5js Variables

Width of the canvas:
`width`

Height of the canvas:
`height`

Operators



Operators

+	Addition
−	Subtraction
*	Multiplication
/	Division
%	Modulo (Rest of a divison)
=	Assignment

Operators

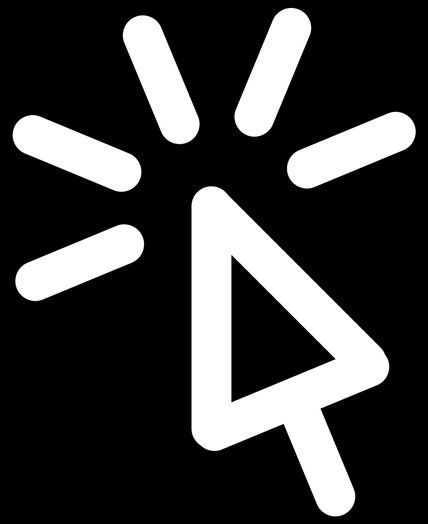
`++` Add 1 to a variable

`--` Subtract 1 from a variable

Operators

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

Interactivity



Interactivity

Mouse

Interactivity

Getting the actual coordinate of the mouse:

```
mouseX
mouseY
```

Interactivity

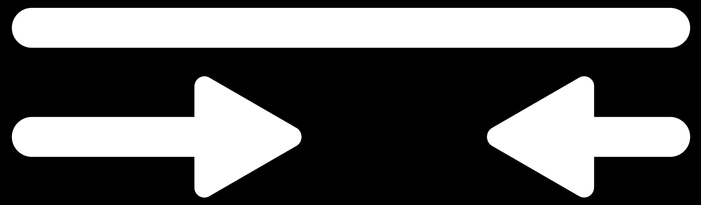
```
function draw() {
  rect( mouseX, mouseY, 50, 50 );
}
```

Interactivity

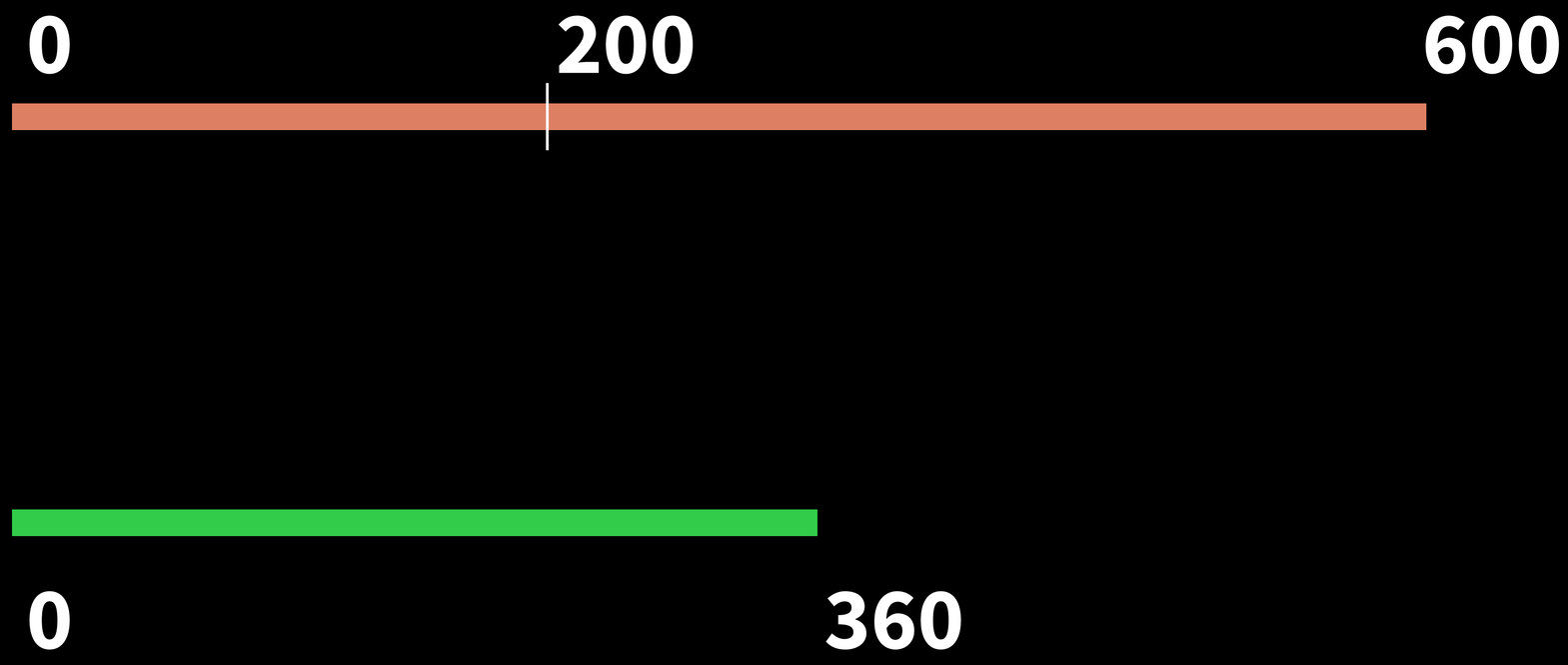
```
function draw() {
}

function mouseClicked() {
  ellipse(mouseX, mouseY, 20, 20);
}
```

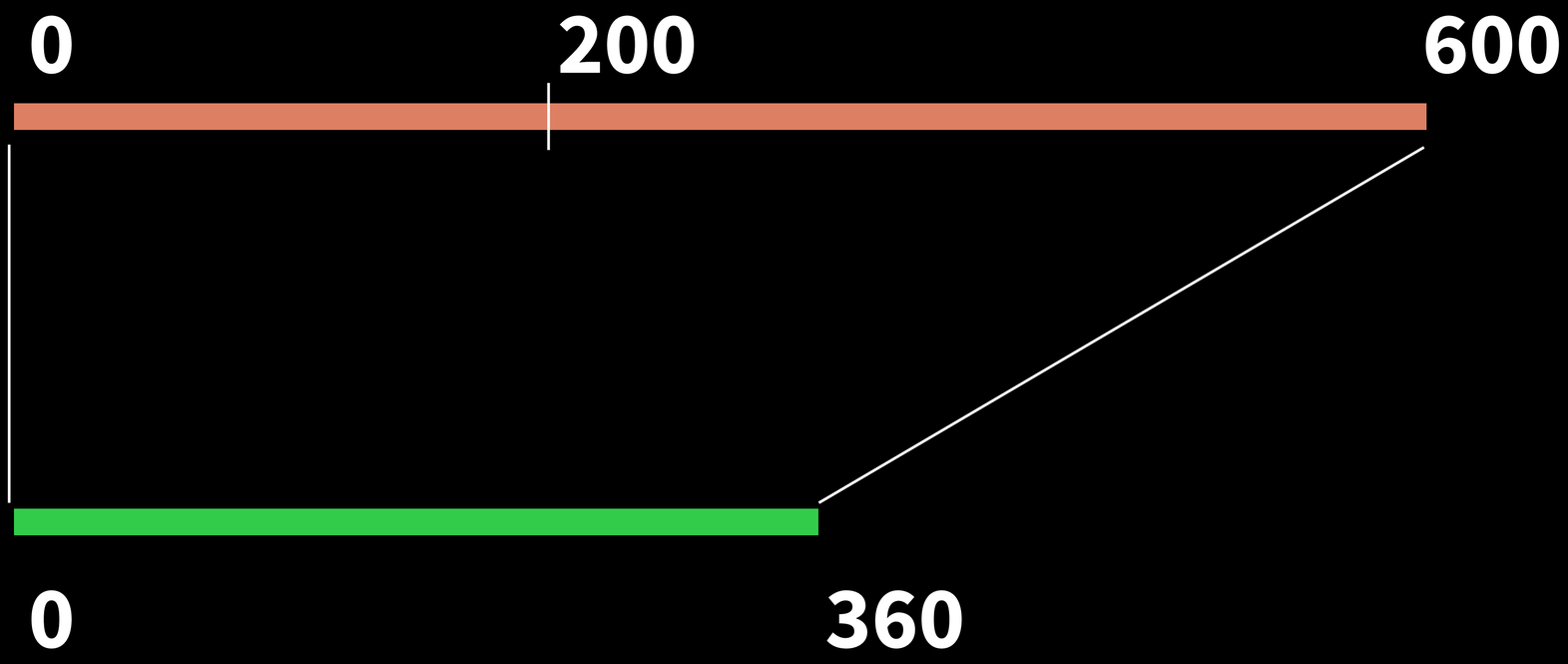
Mapping



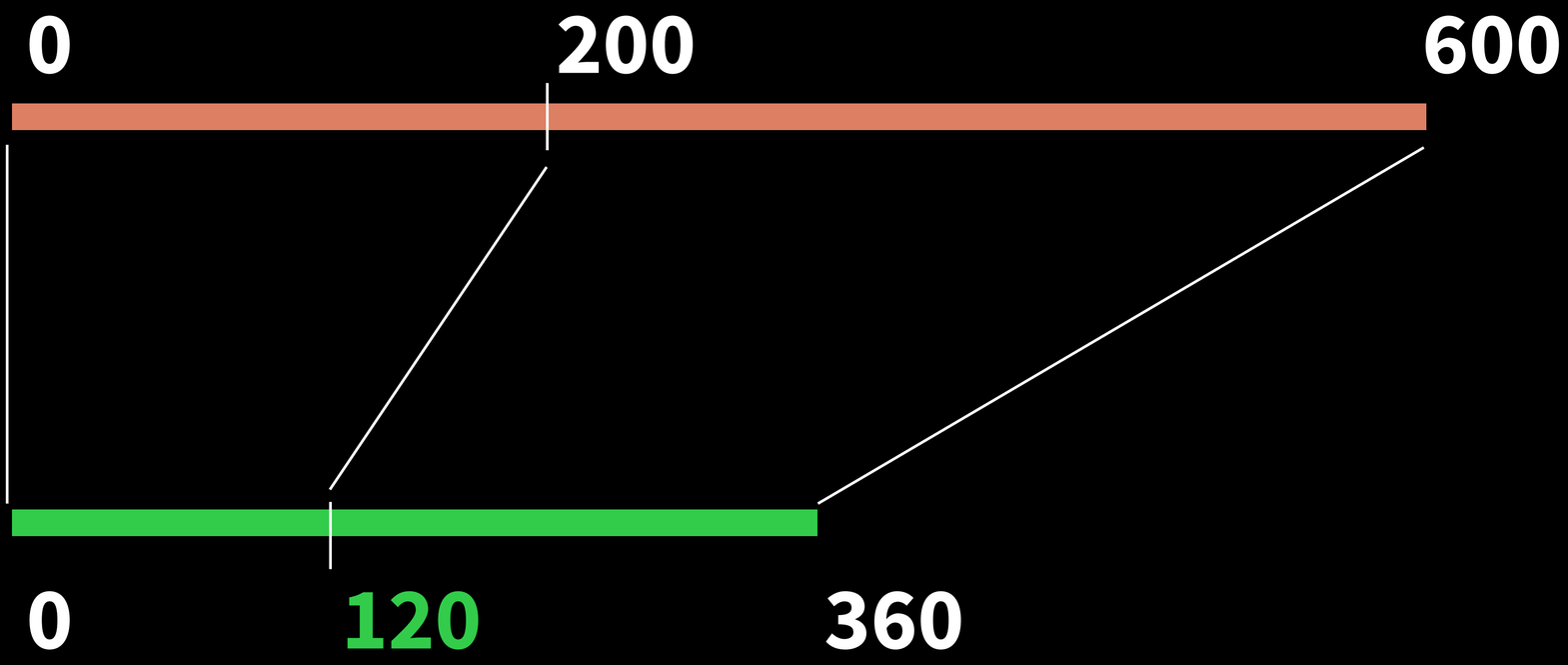
Mapping



Mapping



Mapping



Mapping

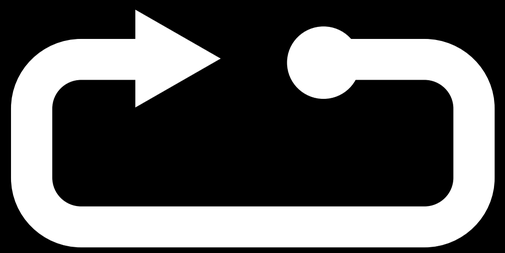
```
void draw() {
    var a = map( mouseX, 0, 600, 0, 360);
    rotate( radians(a) );
    rect( 300, 300, 50, 50 );
}
```

Value in the source range

Minimum and maximum of source range

Minimum and maximum of target range

Loops



Loops

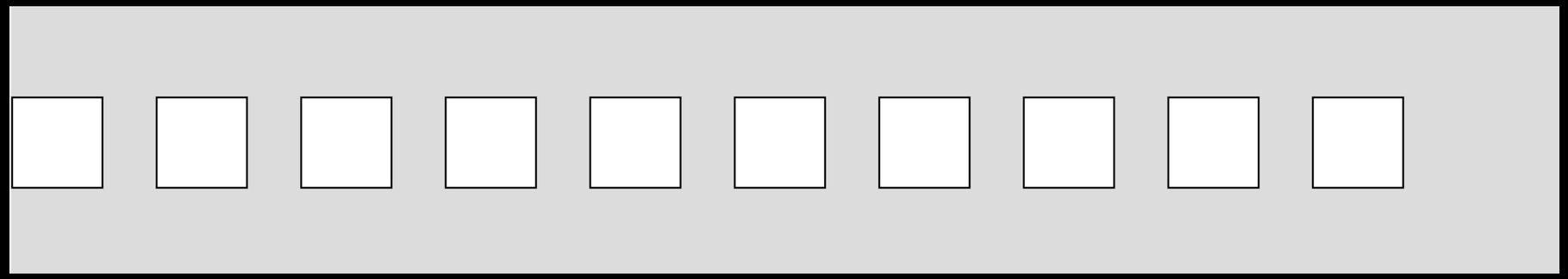
while loop

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);
}
```

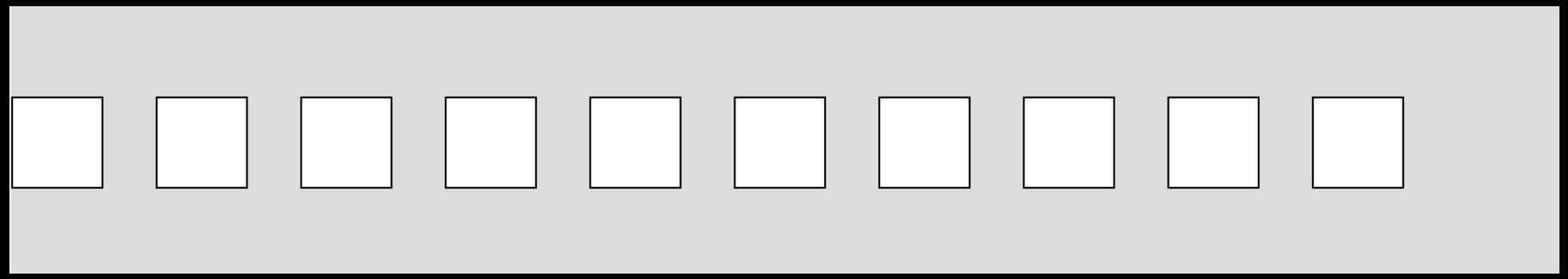
Initializing the counter

Condition for the loop

Operation with the counter

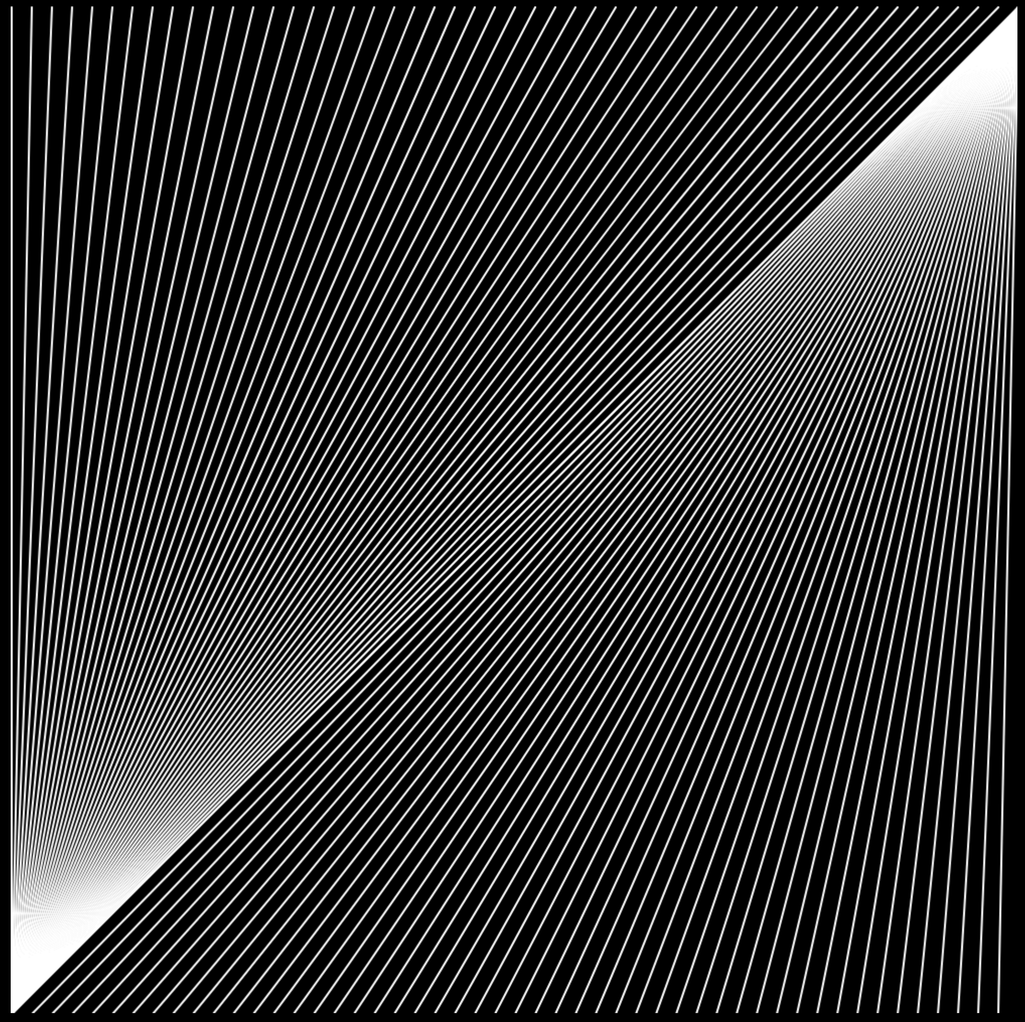
Loops

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



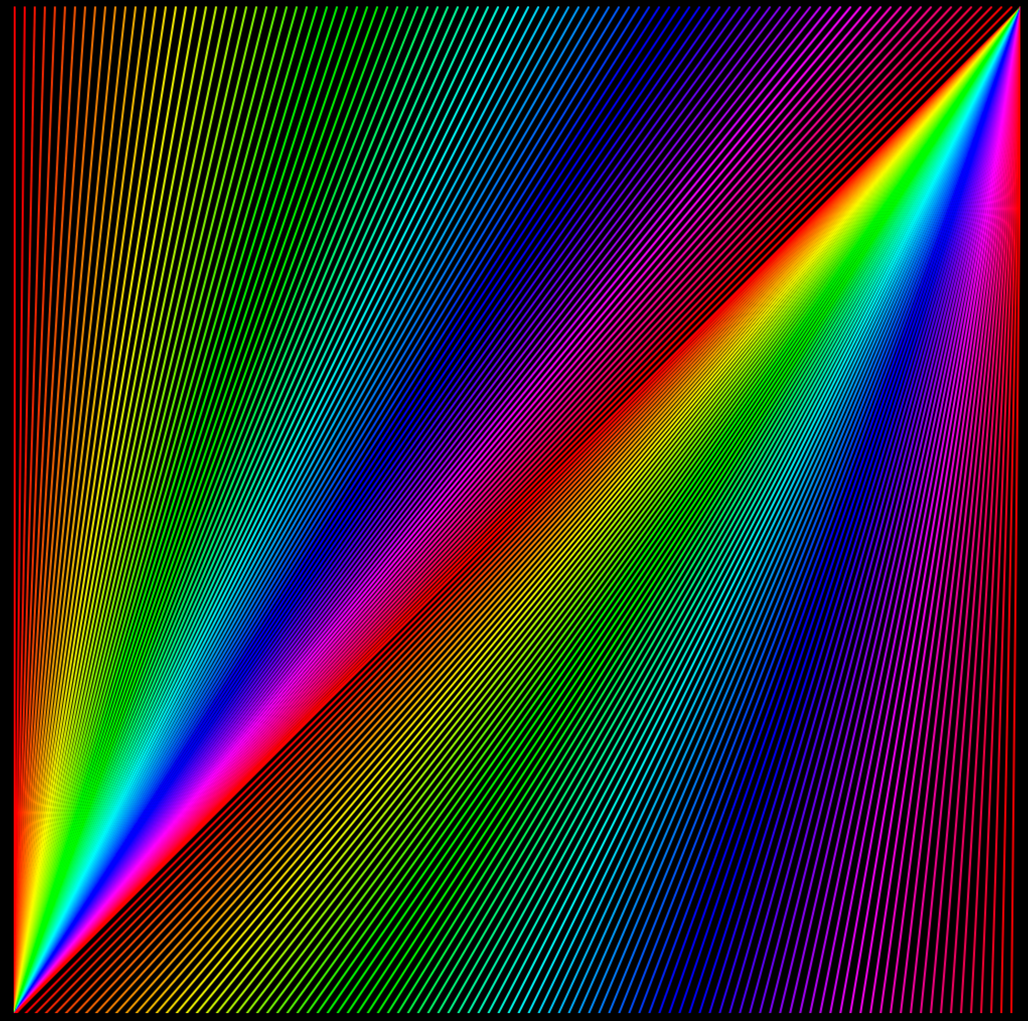
Loops

Let’s try this:



Loops

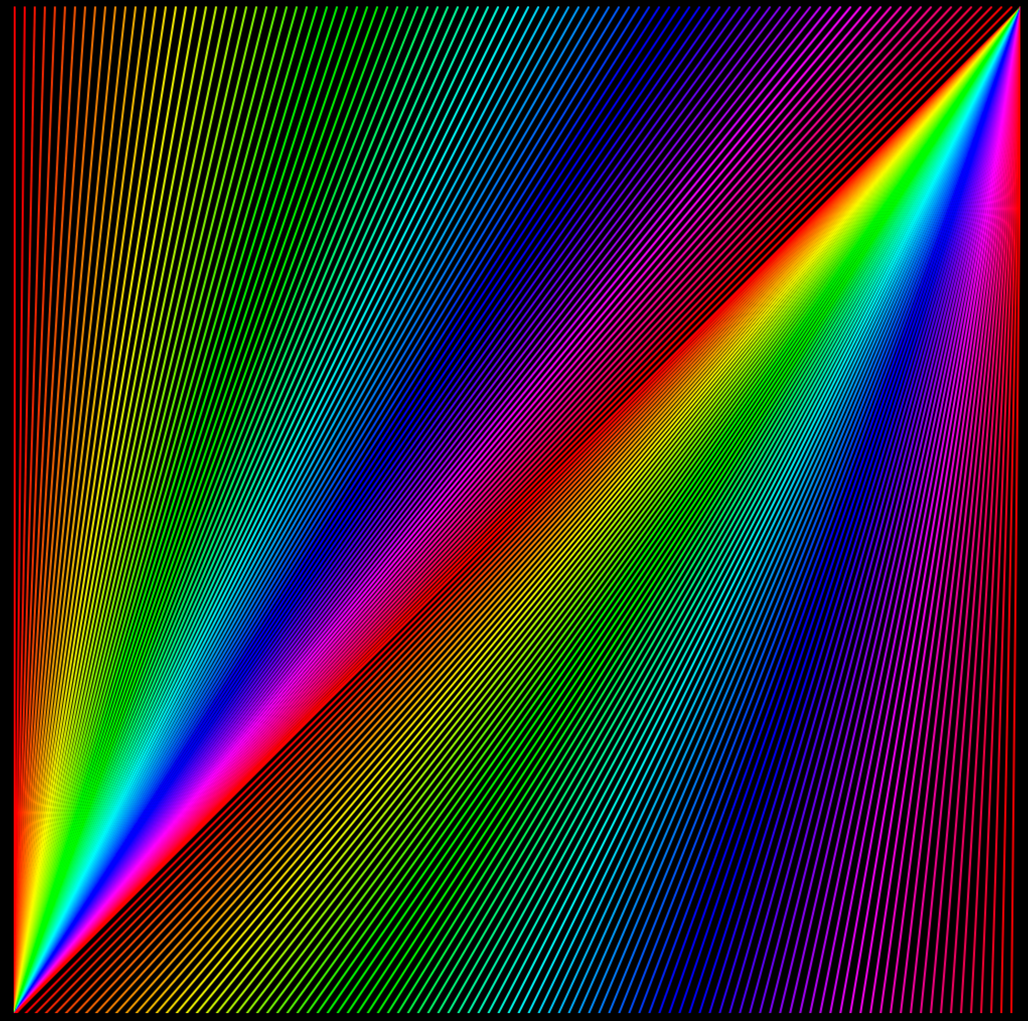
And even a bit more advanced:



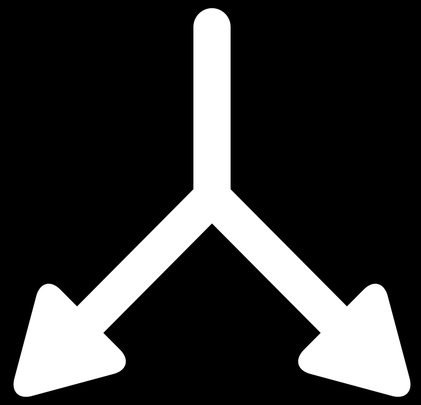
Loops

Have a look at:

`colorMode(HSB)`



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

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

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

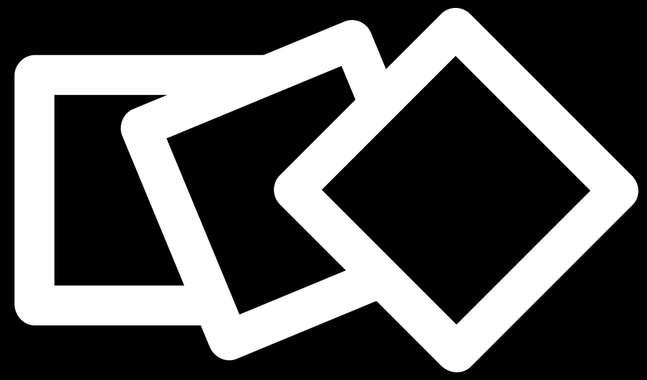
Branches

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

Branches

```
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(angle)
```

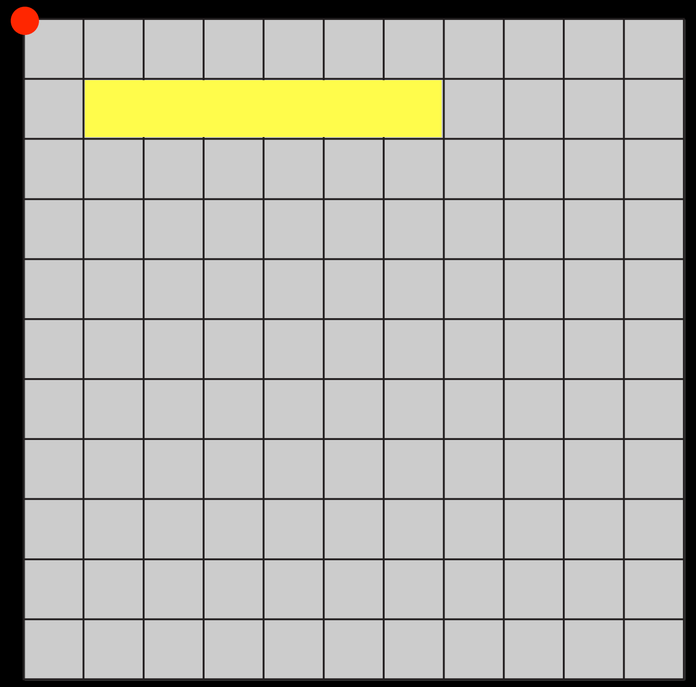
rotates the coordinate system around the actual origin point

Transformations

`rotate(angle)`

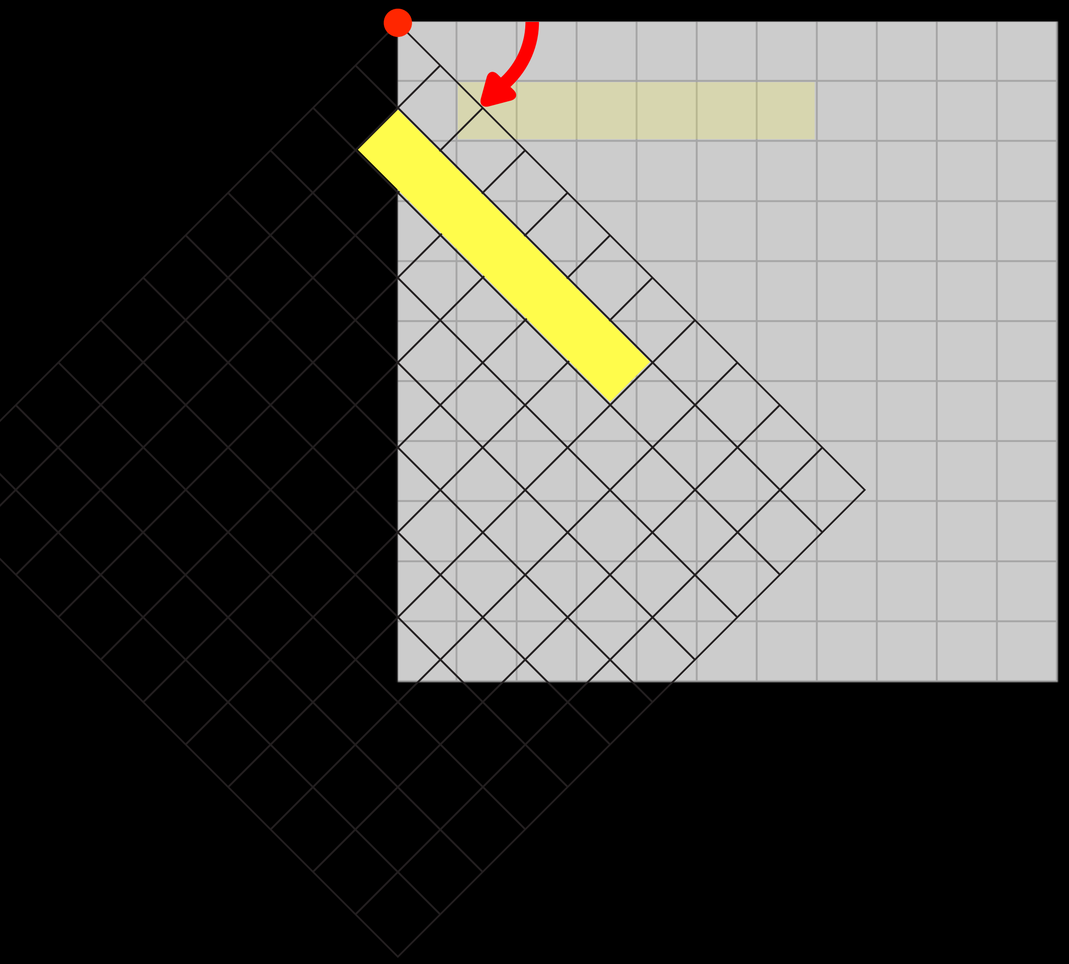
`angle` is a value in radians
(from `0` to `TWO_PI`)

Transformations



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

Transformations



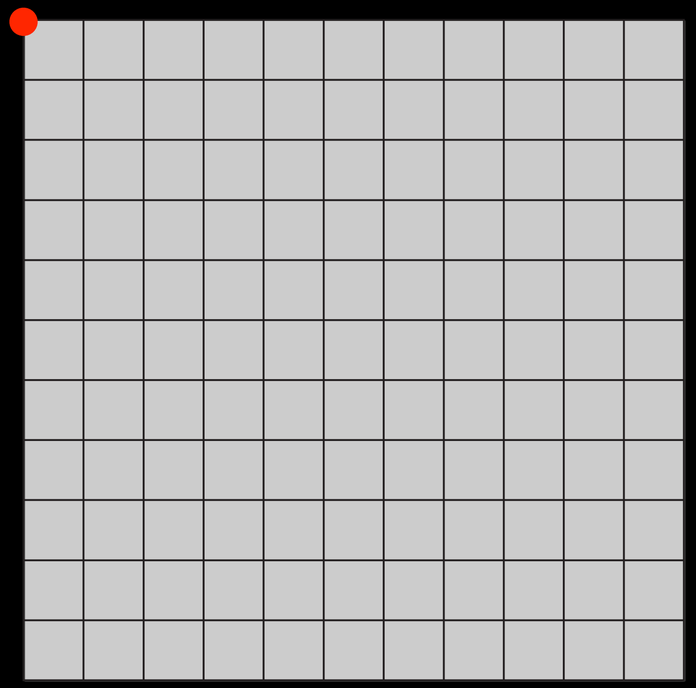
```
rotate(radians(45));
rect(25,25,150,25);
```

Transformations

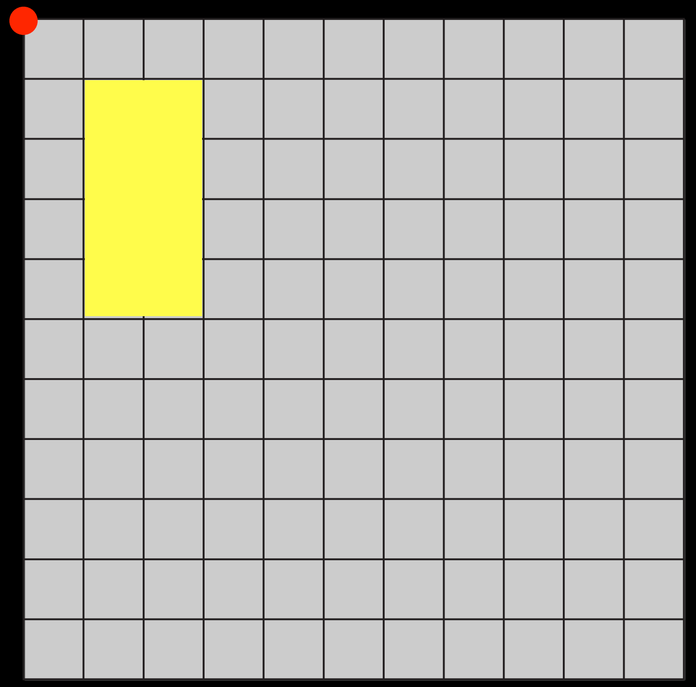
```
translate(x, y)
```

moves the coordinate system x pixels to the right
and y pixels downwards

Transformations

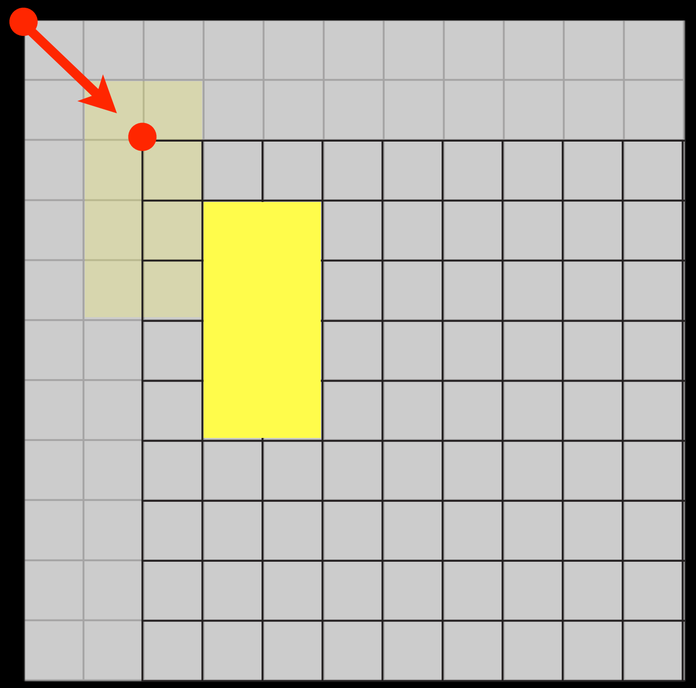


Transformations



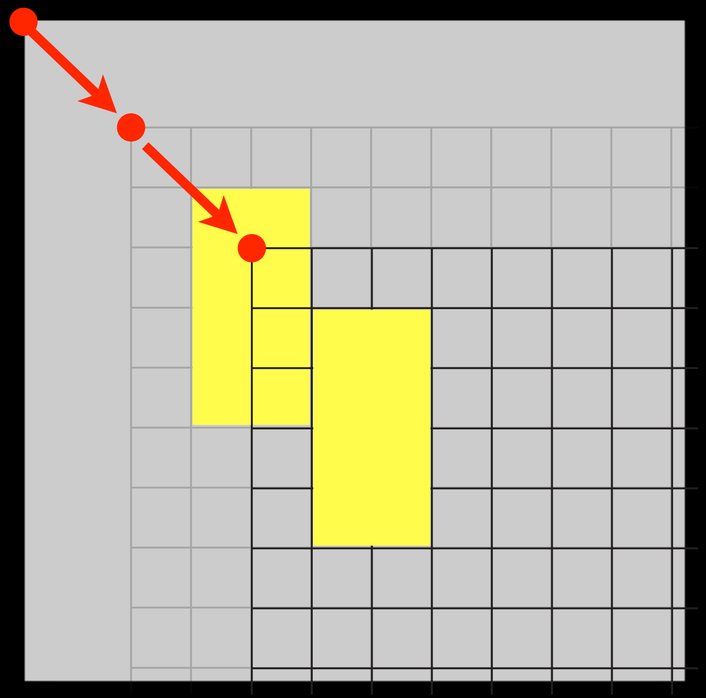
```
rect(25,25,50,100);
```

Transformations



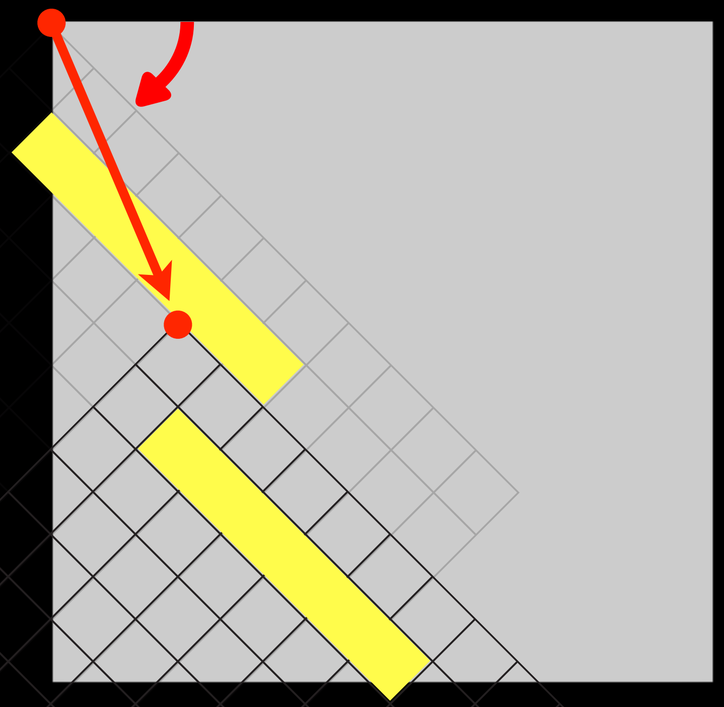
```
translate(50, 50);
rect(25,25,50,100);
```

Transformations



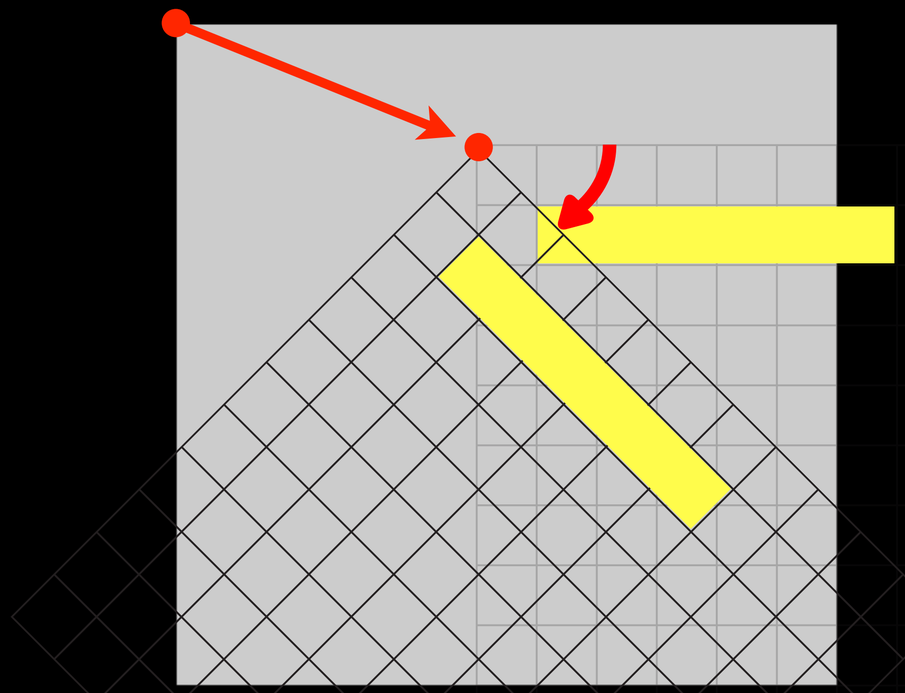
```
translate(50, 50);
rect(25,25,50,100);
translate(50, 50);
rect(25,25,50,100);
```

Transformations



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

Transformations



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

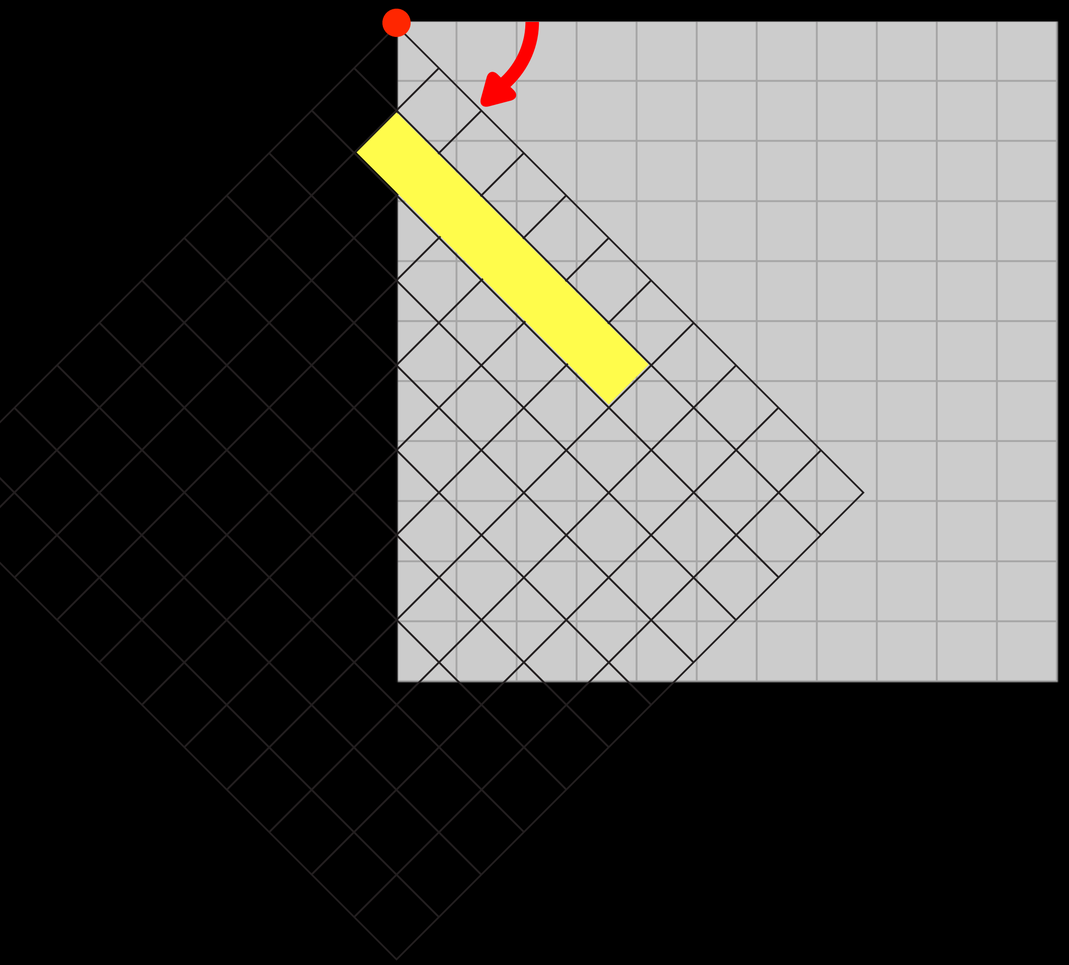
Note the difference when changing the order

Transformations

```
push()
pop()
```

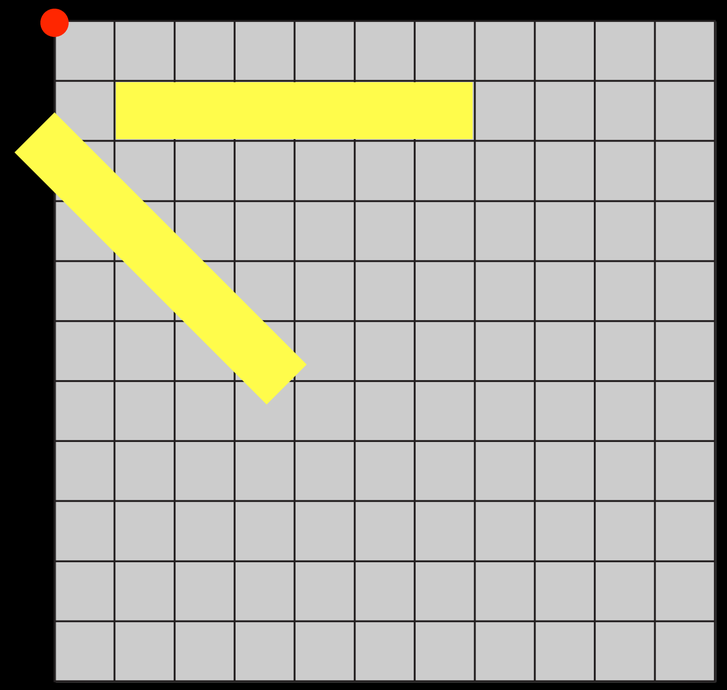
save and load the
coordinate systems

Transformations



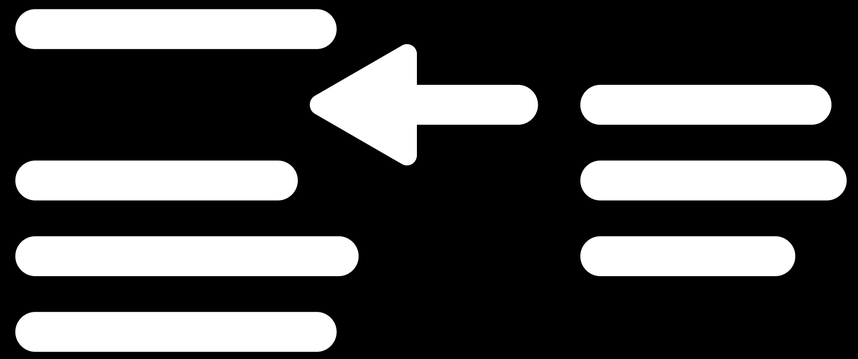
```
push();
rotate(radians(45));
rect(25,25,150,25);
pop();
```

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

```
function circle(x, y, r) {
  ellipse(x, y, r, r);
}
circle(20, 50, 100);
circle(40, 40, 100);
```

Defining parameters

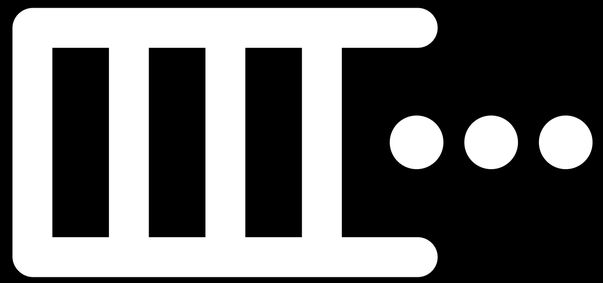
Parameter values

Functions

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

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

Arrays



Arrays

Arrays are lists of values

Arrays

```
var a = [4, 7, 3];
```

Name of the variable
Array indicators
Values

Arrays

Getting values:

```
var a = [4, 7, 3];

a[0] → 4
a[1] → 7
a[2] → 3
a[3] → undefined
```

Arrays

Appending values:

```
var a = [4, 7, 3];
a.push(15)

// a contains [4, 7, 3, 15] afterwards
```

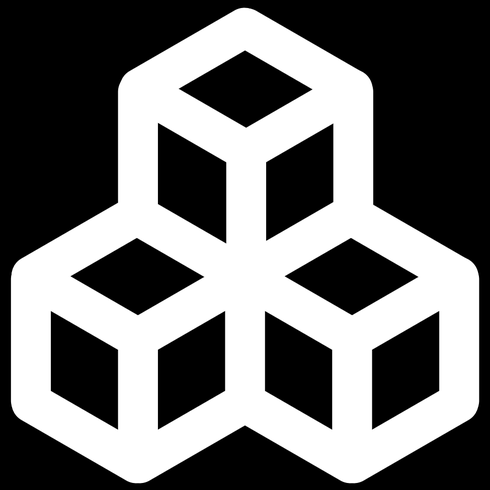
Arrays

Removing values:

```
var a = [4, 7, 3];
a.pop()

// a contains [4, 7] afterwards
```

Objects



Objects

Objects can store information in Key-Value-Pairs

Objects

```
var obj = {name:'John', age:31};
```

Name of the variable
Object indicators
Keys
Key-Value-Pair separators
Values

Objects

Getting values:

```
var obj = {name:'John', age:31};

obj.name   → 'John'
obj.age    → 31
obj.weight → undefined
```

Alternatively:

```
obj['age'] → 31
```

Objects

Appending values:

```
var obj = {name:'John', age:31};
obj.weight = 80;

// obj contains {name:'John', age:31,
   weight:80} afterwards
```

Objects

Removing values:

```
var obj = {name:'John', age:31};
delete obj.age;

// obj contains {name:'John'} afterwards
```