Creative Coding

Professor Danne Woo

creativecode.dannewoo.com

ARTS 249

Spring 2020

Thursday 2:00 PM – 5:50 PM

I-Building 213

Static Visual Design

Week 01: Intro to programming, Processing and creative coding

Week 02: Forms, Shapes and Variables

Week 03: Computational Color and Export

Week 04: Repetition, Decisions and Randomization

Week 05: Functions, Classes and Typography

Week 06: Data Visualization

Week 07: Midterm Presentation

Functions

Simple Custom Function

```
createCanvas(width, height);
background(color);
rect(x, y, width, height);
ellipse(x, y, width, height);
translate(x, y);
rotate(degrees);
```

Functions

Simple Custom Star Function

```
function star(x, y, d, points, innerPoints) {
  push();
  translate(x, y);
  beginShape();
  for (var i = 0; i < points; i++) {
    var posNegVal;
    if (i % 2 == 1) posNegVal = d*innerPoints;
    else posNegVal = d;
    var vertexX = sin(radians(i * (360/points))) * (d + posNegVal);
    var vertexY = cos(radians(i * (360/points))) * (d + posNegVal);
    vertex(vertexX, vertexY);
  }
  endShape();
  pop();
}
```

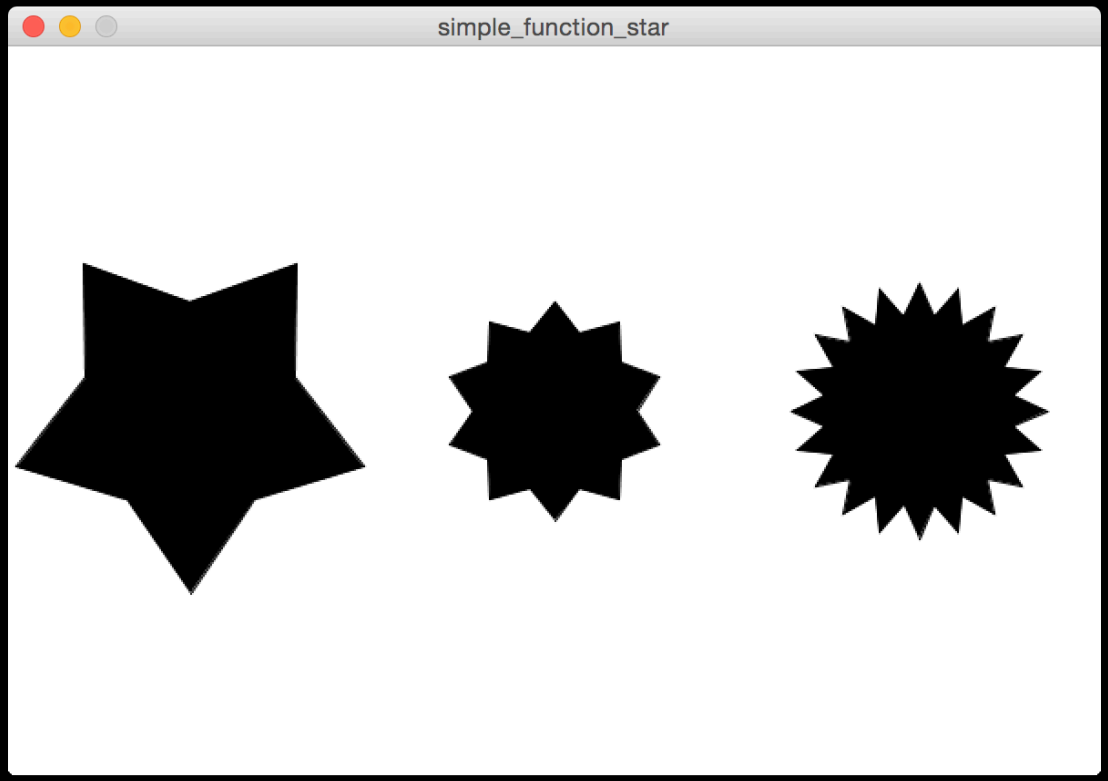
Functions

Simple Custom Star Function

```
function setup() {
  createCanvas(600, 400);
  background(255);
  fill(0);

  star(width/2 - 200, height/2, 50, 10, 0.2);
  star(width/2, height/2, 30, 20, 0.5);
  star(width/2 + 200, height/2, 35, 40, 0.5);
}
```

Functions

Simple Custom Star Function



Left: star(width/2 - 200, height/2, 50, 10, 0.2);

Middle: star(width/2, height/2, 30, 20, 0.5);

Right: star(width/2 + 200, height/2, 35, 40, 0.5);

Functions

Simple Custom Function

```
function face(x, y, w, h, c) {
  fill(c, 0, 0);
  ellipse(x, y, w, h);
  fill(0, c, 0);
  ellipse(x - w/6, y - h/8, w/4, h/4);
  ellipse(x + w/6, y - h/8, w/4, h/4);
  ellipse(x, y + h/8, w/8, h/8);
  ellipse(x, y + h/3, w/8, h/8);
}
```

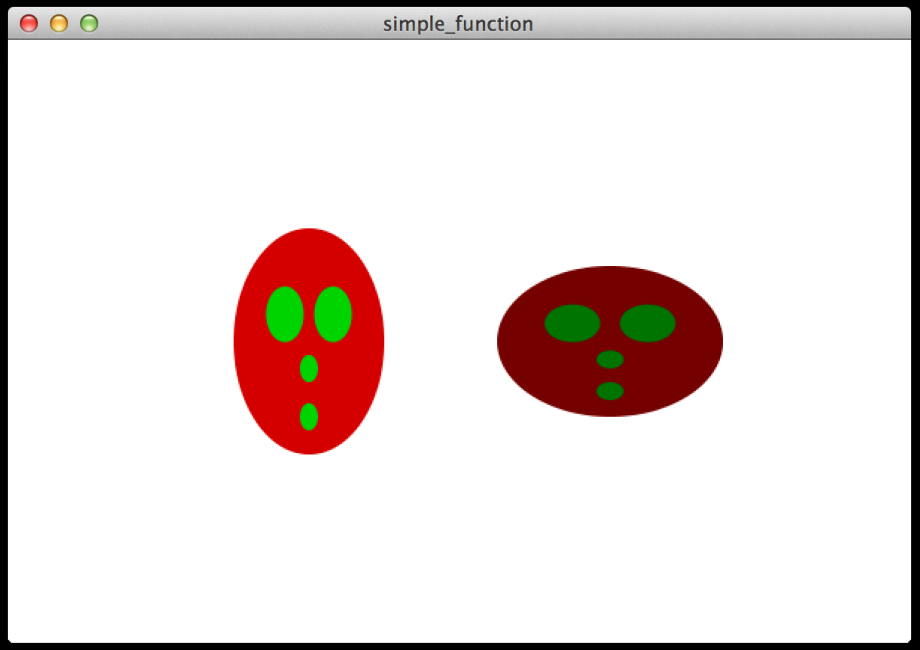
Functions

Simple Custom Function

```
function setup() {
  createCanvas(600, 400);
  background(255);
  noStroke();
  face(width/2 - 100, height/2, 100, 150, round(random(255)));
  face(width/2 + 100, height/2, 150, 100, round(random(255)));
}
function face(x, y, w, h, c) {
  fill(c, 0, 0);
  ellipse(x, y, w, h);
  fill(0, c, 0);
  ellipse(x - w/6, y - h/8, w/4, h/4);
  ellipse(x + w/6, y - h/8, w/4, h/4);
  ellipse(x, y + h/8, w/8, h/8);
  ellipse(x, y + h/3, w/8, h/8);
}
```

Functions

Simple Custom Function



LEFT:

face(width/2 - 100, height/2, 100, 150, round(random(255)));

RIGHT:

face(width/2 + 100, height/2, 150, 100, round(random(255)));

Classes

Simple Custom Classes

```
function ClassName(var1, var2) {
        this.function1() {
                fill(0, 0, 0);
                ellipse(var1, var2, 20, 20);
        }
        this.function2() {
                fill(0, 0, 0);
                rect(var1, var2, 10, 40);
        }
}
```

Classes

Simple Custom Classes

```
function Face(x, y, w, h, c) {
	// Create multiple functions to use.
	this.features = function() {
		fill(0, c, 0);
		ellipse(x - w/6, y - h/8, w/4, h/4);
		ellipse(x + w/6, y - h/8, w/4, h/4);
		ellipse(x, y + h/8, w/8, h/8);
		ellipse(x, y + h/3, w/8, h/8);
	}
	this.head = function() {
		rectMode(CENTER);
		fill(c, 0, 0);
		rect(x, y, w, h);
	}
}
```

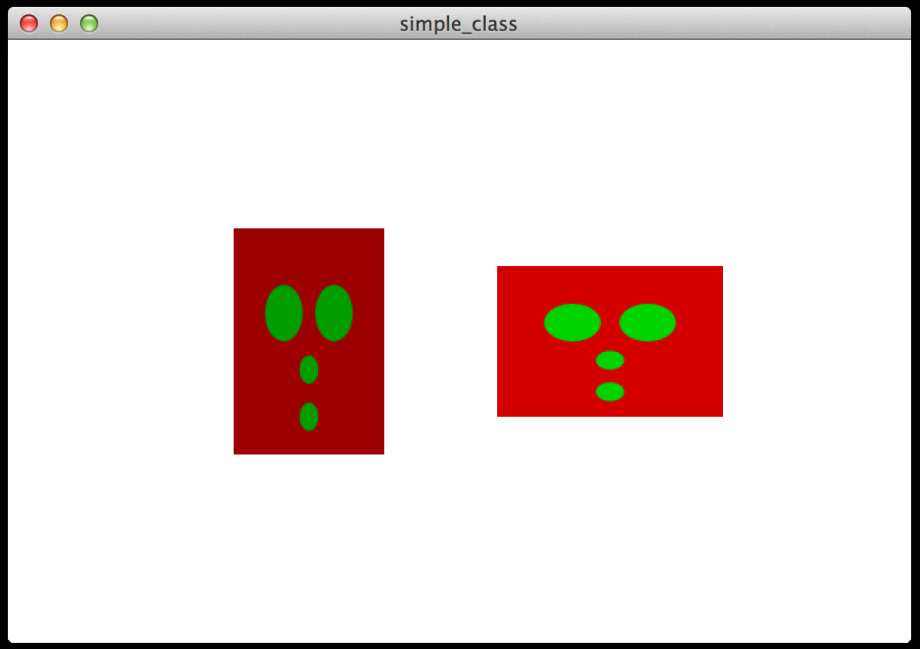
Classes

Simple Custom Classes

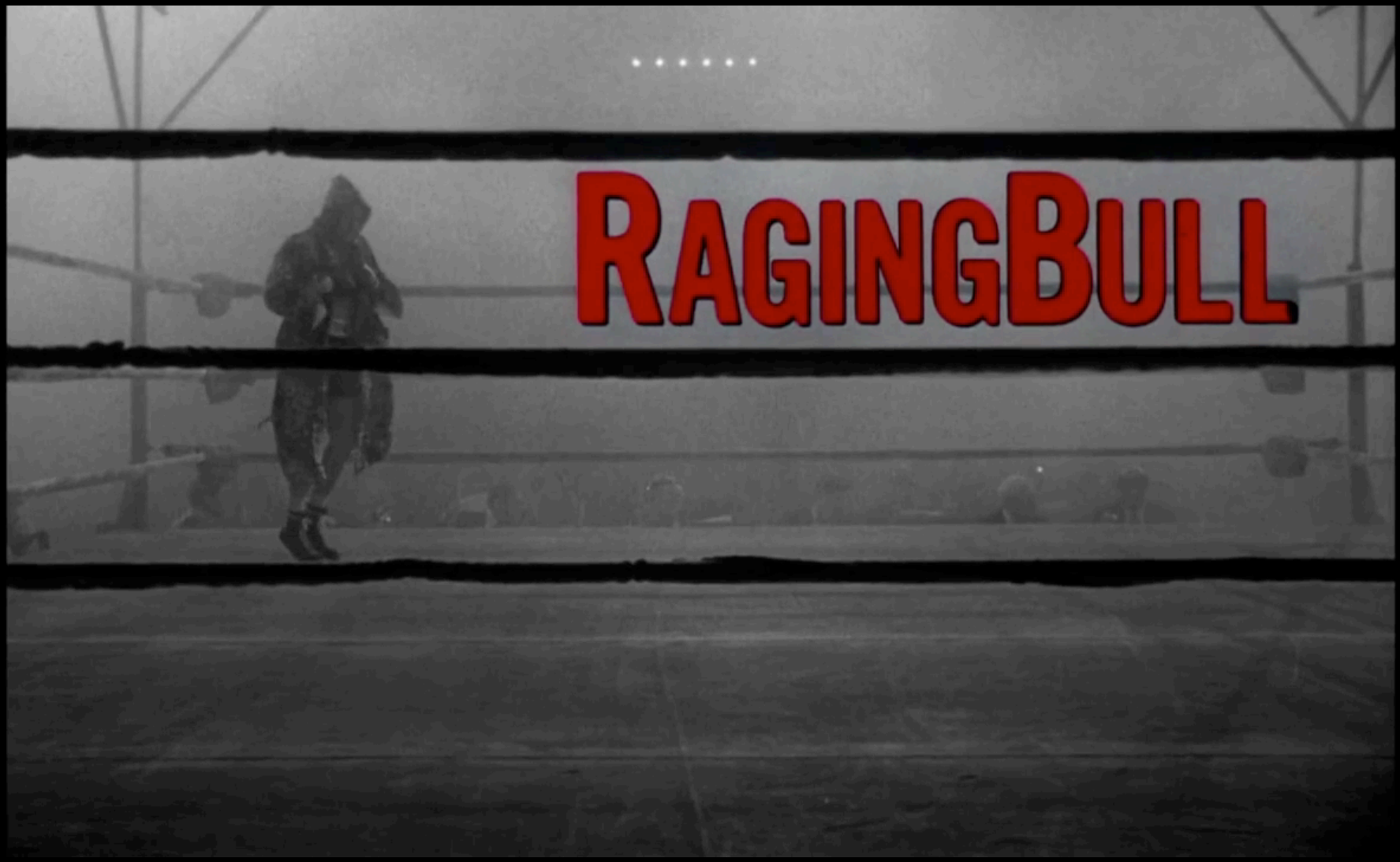
```
function setup() {
	createCanvas(600, 400);
	background(255);
	noStroke();

	//Two face objects with different values
	var f1 = new Face(width/2 - 100, height/2, 100, 150, random(255));
	var f2 = new Face(width/2 + 100, height/2, 150, 100, random(255));

	//Draw the head and/or face features function inside the class
	f1.head();
	f1.features();
	f2.head();
	f2.features();
}
```

Classes

Simple Custom Classes




```
var f1 = new Face(width/2 - 100, height/2, 100, 150, random(255));
var f2 = new Face(width/2 + 100, height/2, 150, 100, random(255));
f1.head();
f1.features();
f2.head();
f2.features();
```


RAGINGBULL

Brief History of Type

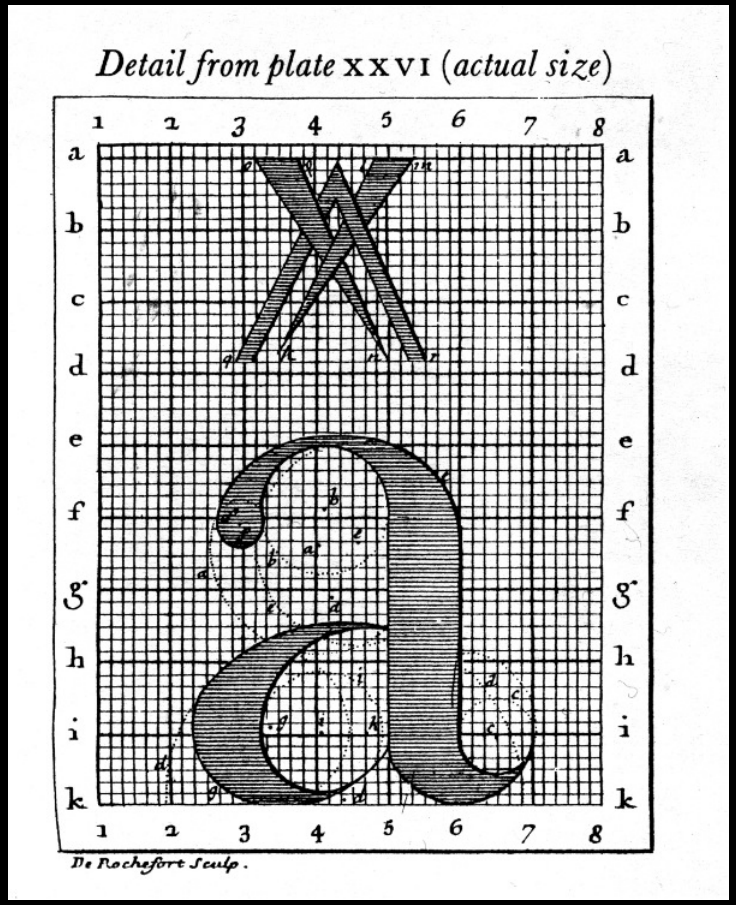
Illuminated Manuscripts

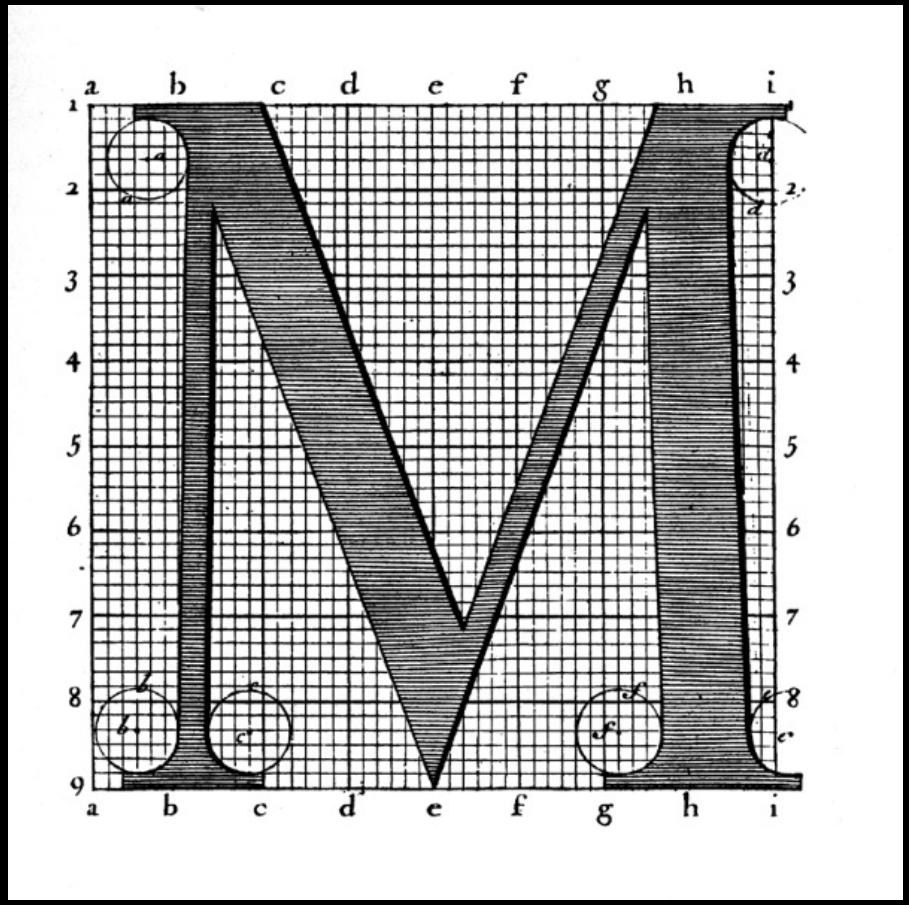
Brief History of Type

Gutenberg Printing Press

Brief History of Type

Designing on a Grid





Brief History of Type

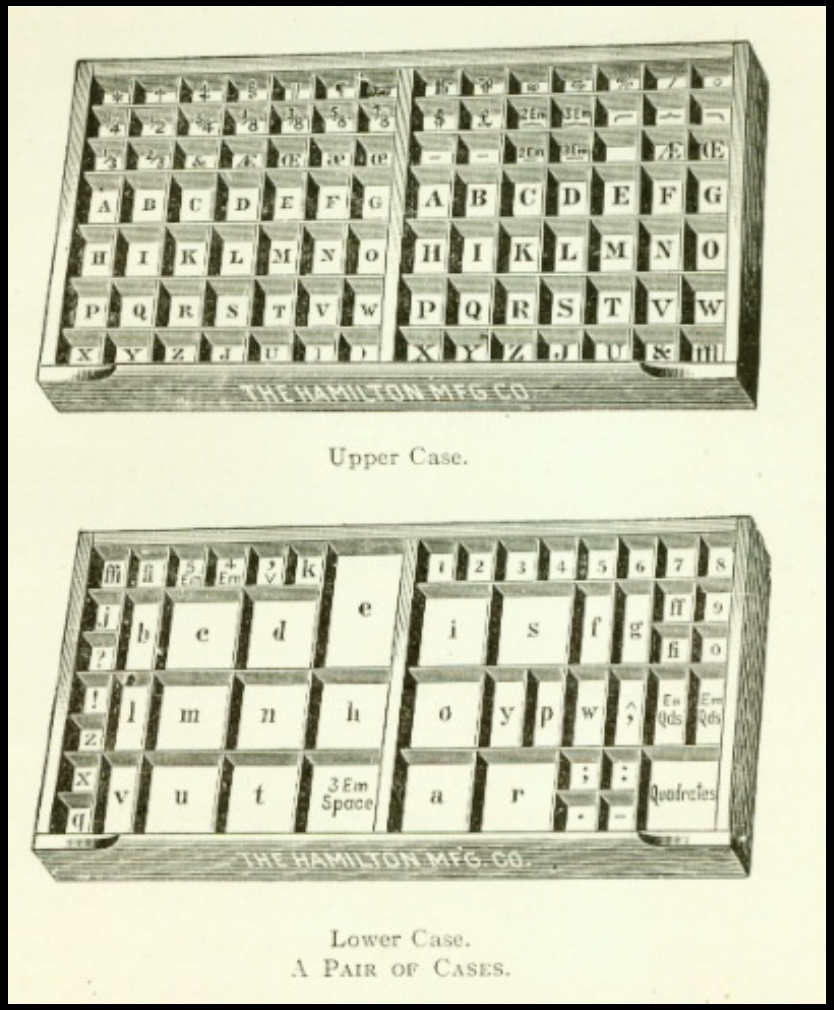
Linotype

Brief History of Type

Personal Computer

Brief History of Type

Lingo



Typeface


A few important typefaces
Avenir
(1988)
Bodoni
(1925)
Garamond
(1989)
Gill Sans
(1926)
Georgia
(1993)
Futura
(1936)
Gotham
(2000)
Helvetica
(1957)
Akkurat
(2004)

Typeface

Classifications

Serif

Sans Serif

Script

Display

Dingbat or Specialty

Typeface

Serif

Typeface

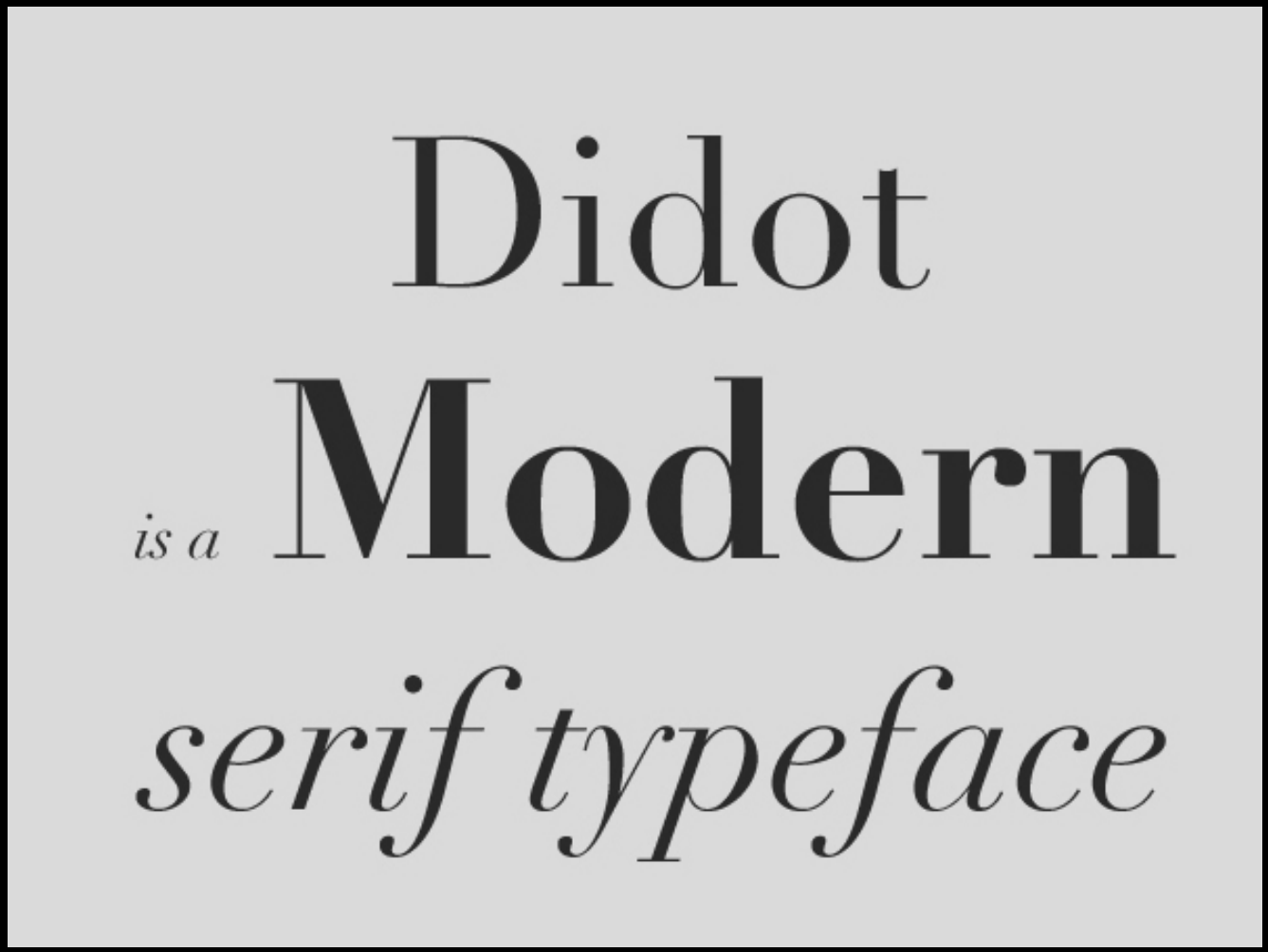
Old Style Serifs

Typeface

Transitional Serifs

Typeface

Modern Serifs



Typeface

Slab Serifs

Typeface

Sans Serif

Typeface

Grotesque

Typeface

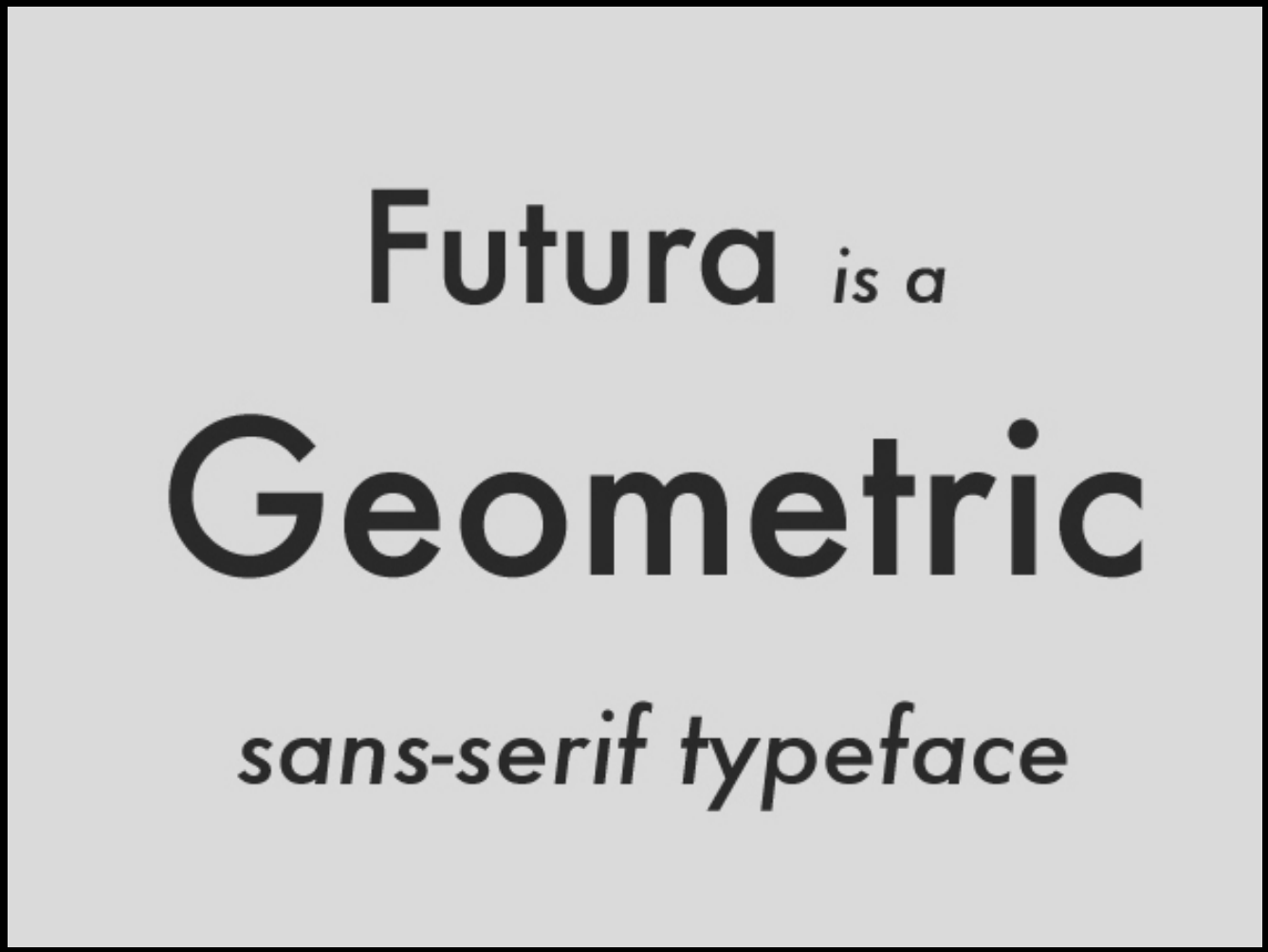
Neo-grotesque

Typeface

Humanist typefaces

Typeface

Geometric sans-serifs

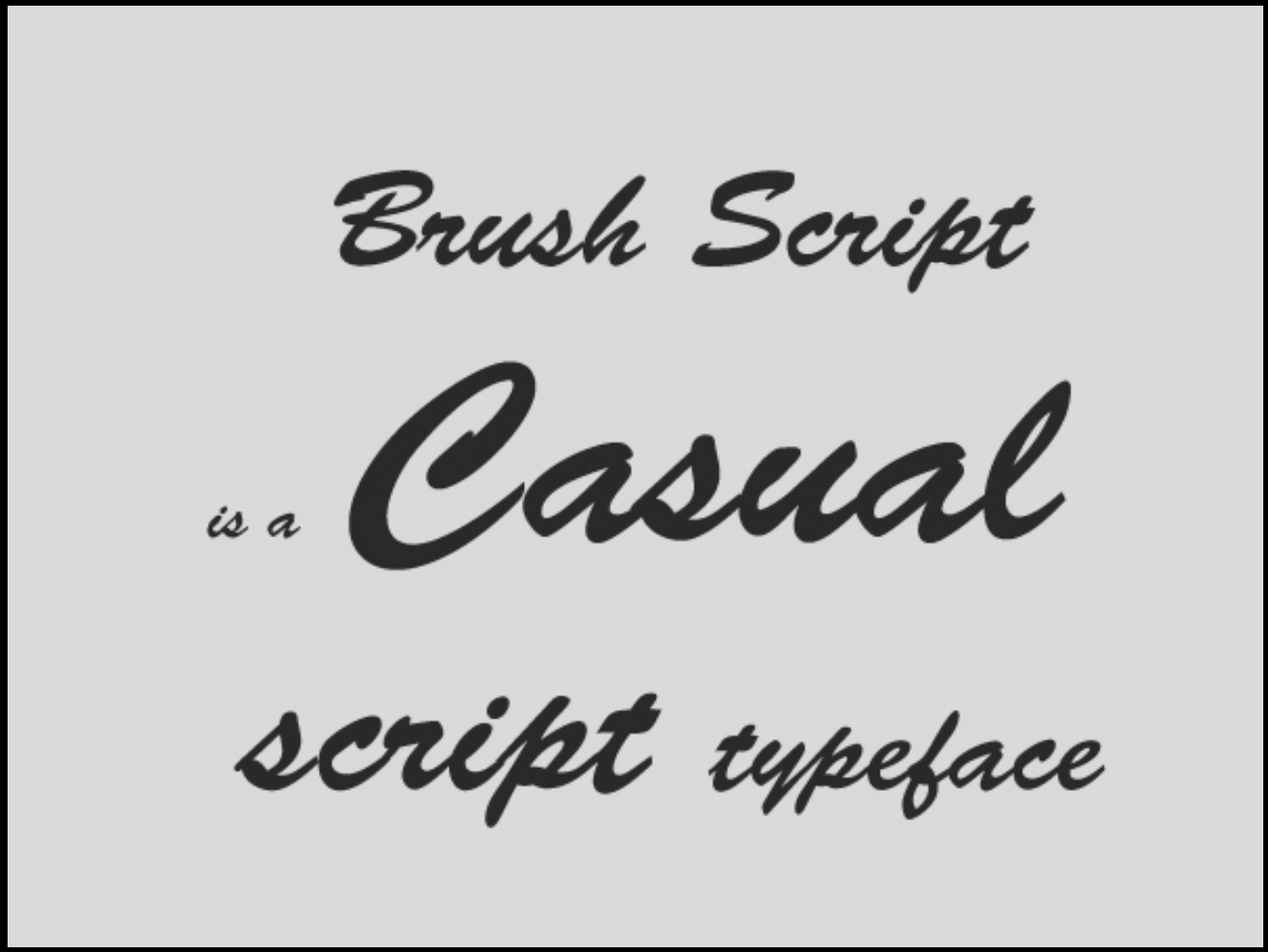


Typeface

Formal Scripts

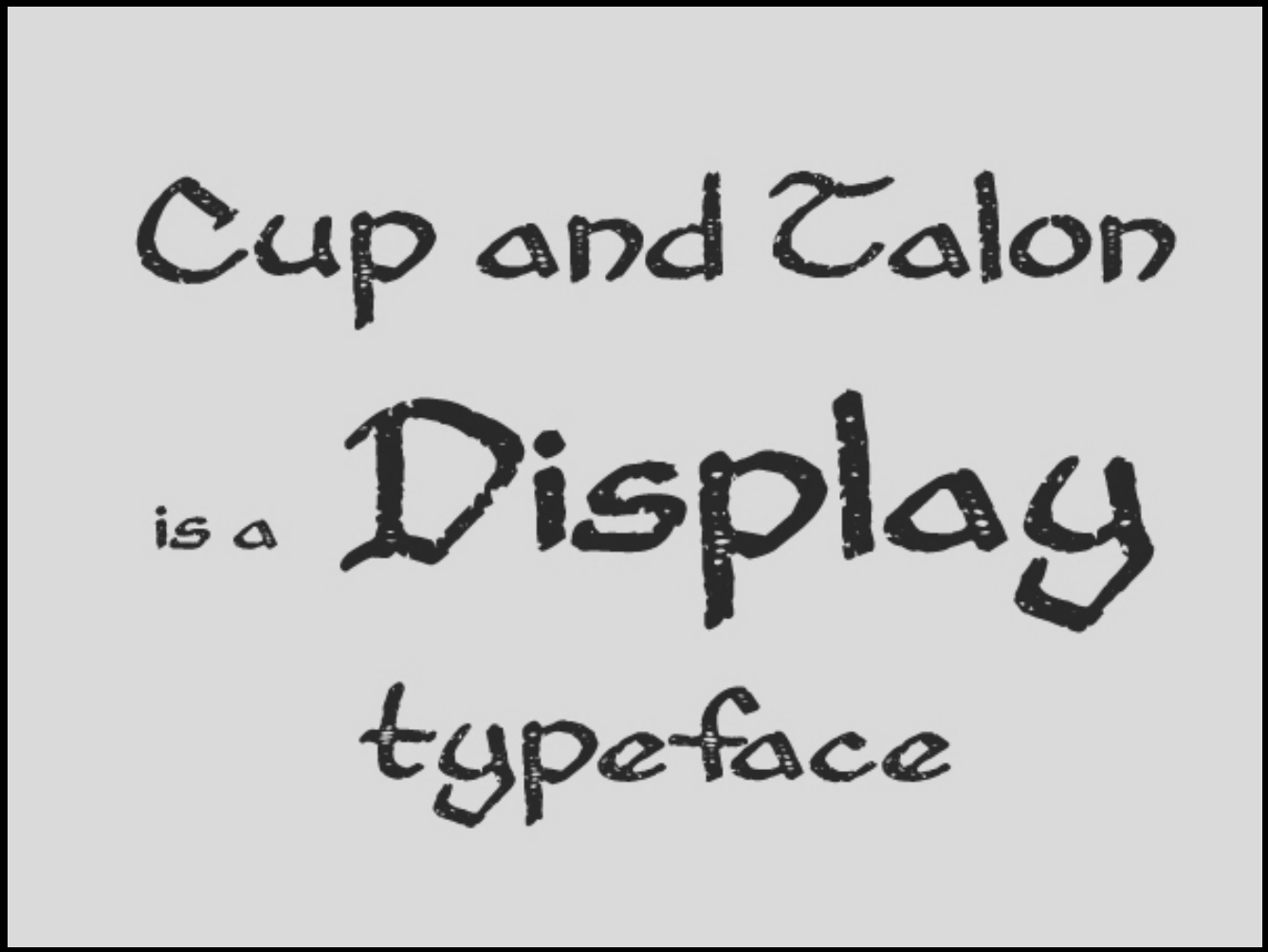
Typeface

Casual Scripts



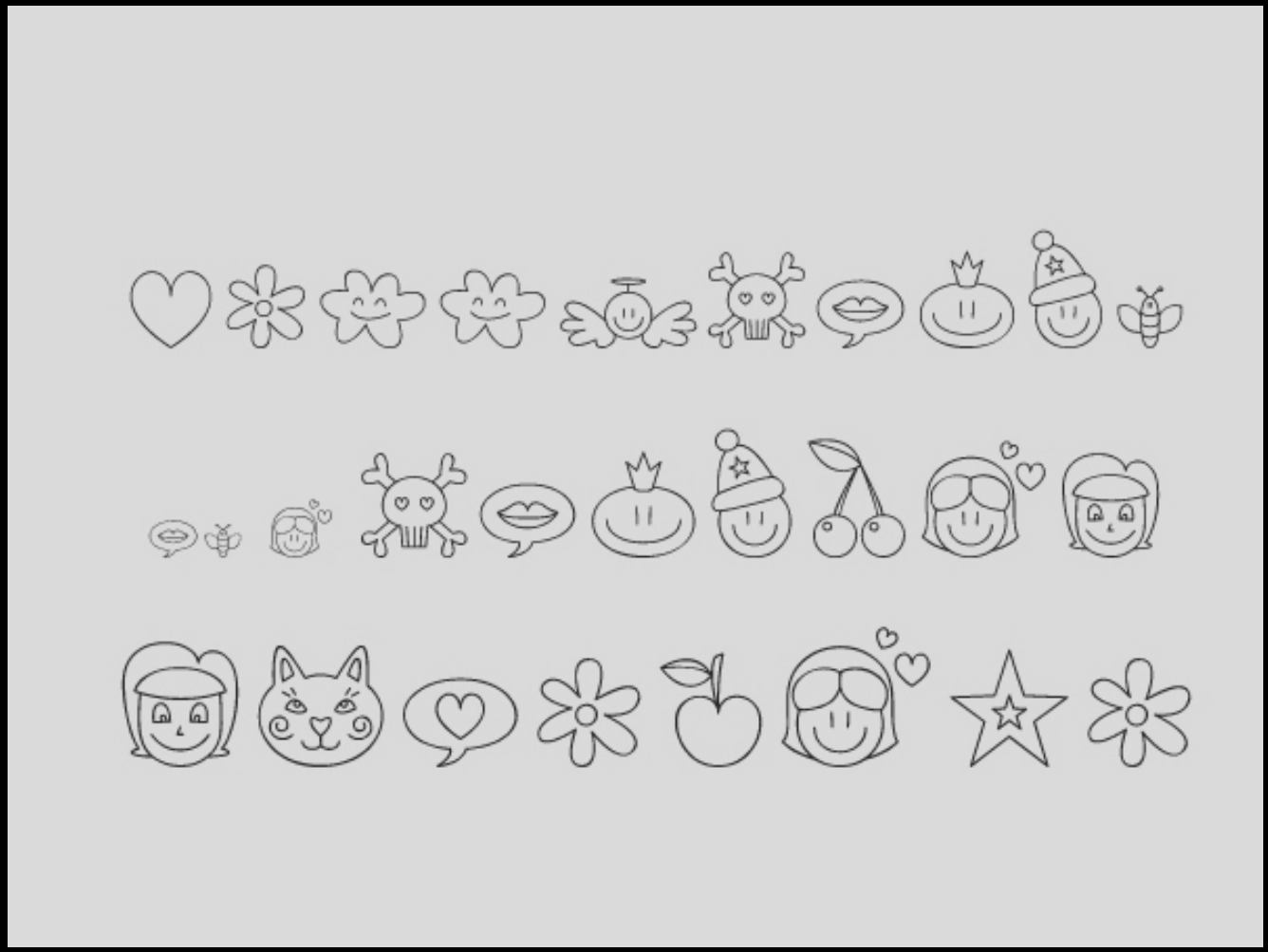
Typeface

Display



Typeface

Dingbats and Specialty Typefaces



Typeface

Proportional vs Monospace Typefaces

Typeface

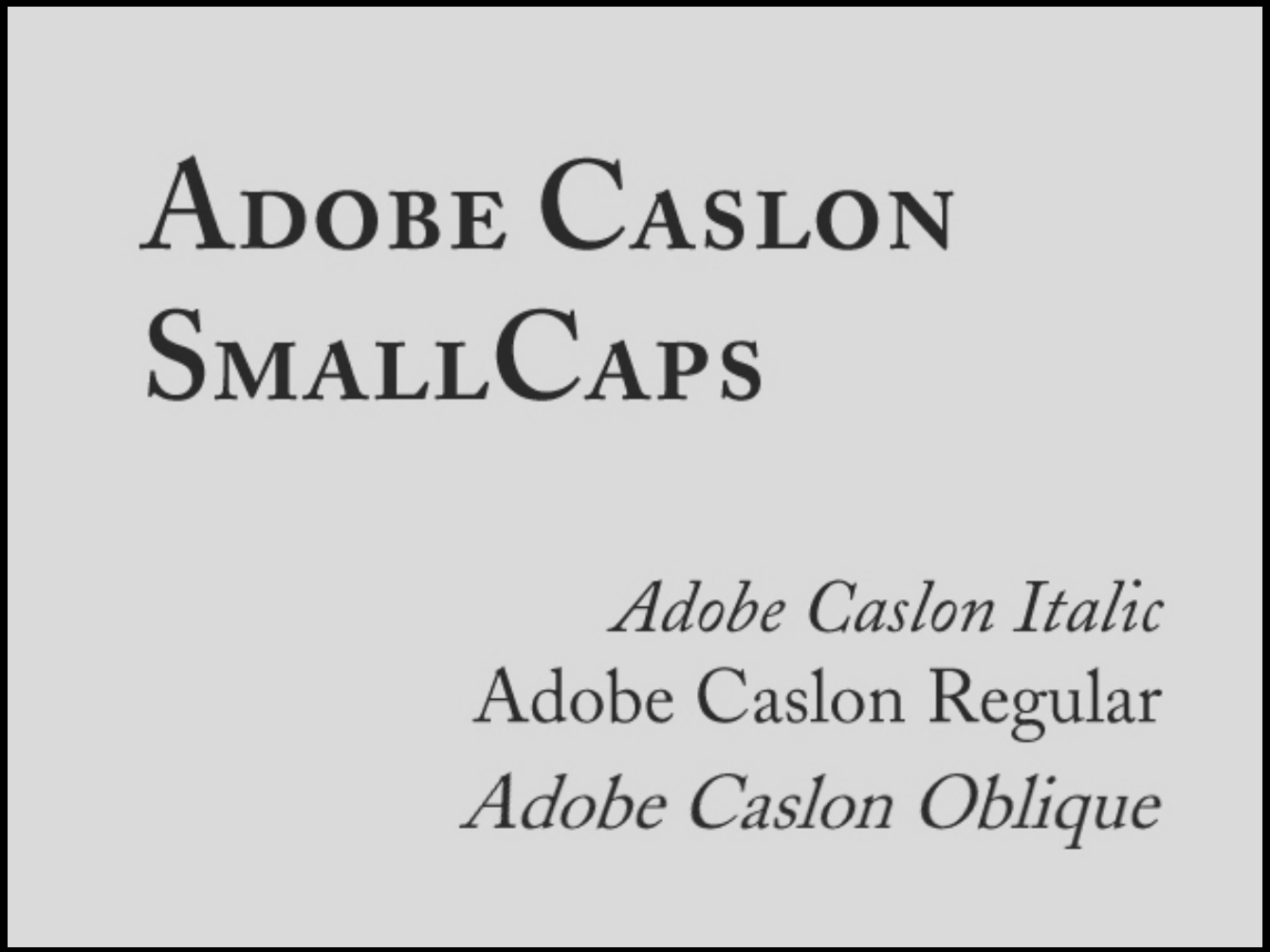
Weights

Benton Gothic Thin
Benton Gothic Light
Benton Gothic Medium
Benton Gothic Bold

Adobe Caslon Regular
Adobe Caslon SemiBold
Adobe Caslon Bold

Typeface

Styles

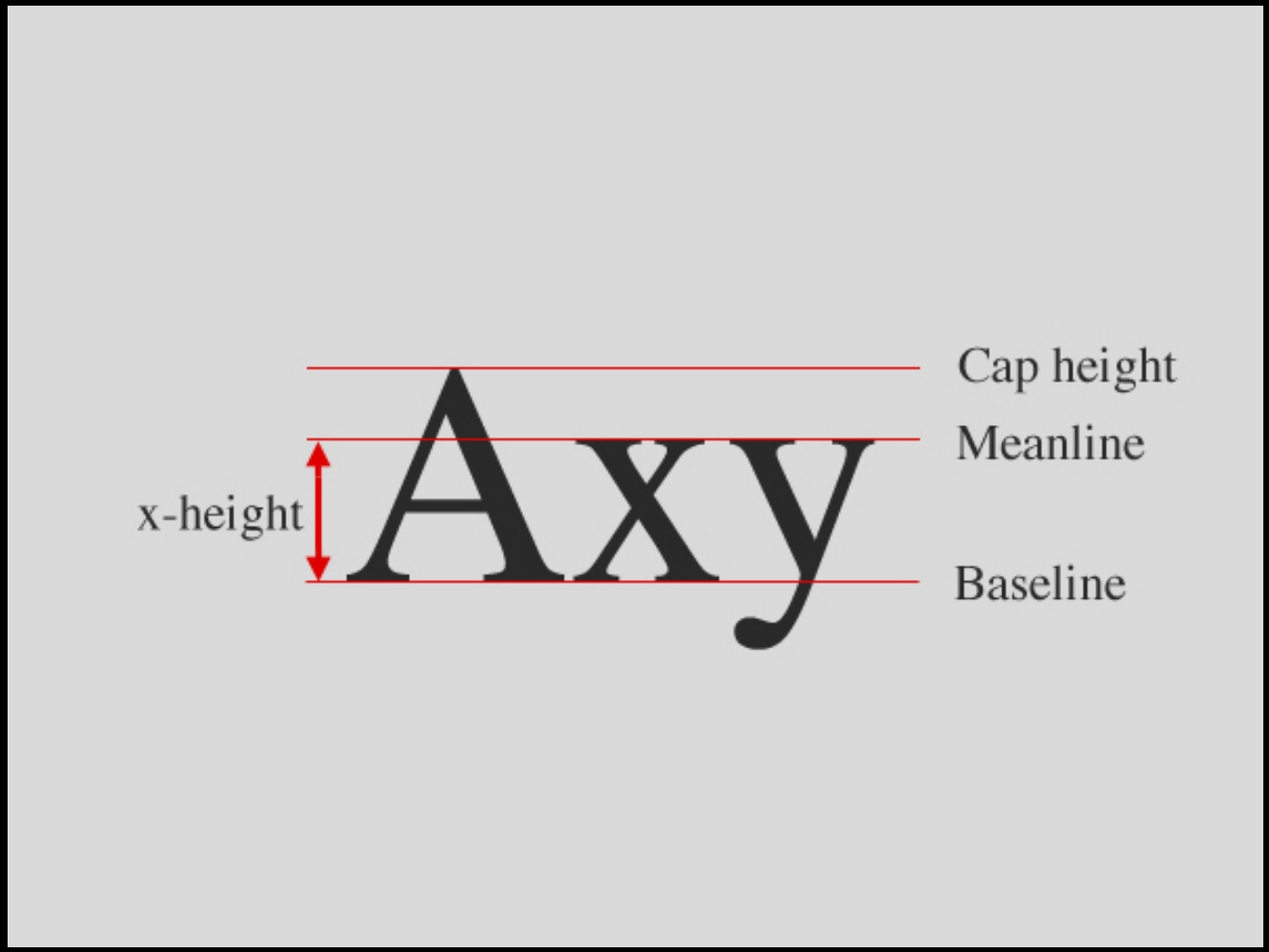


Typeface

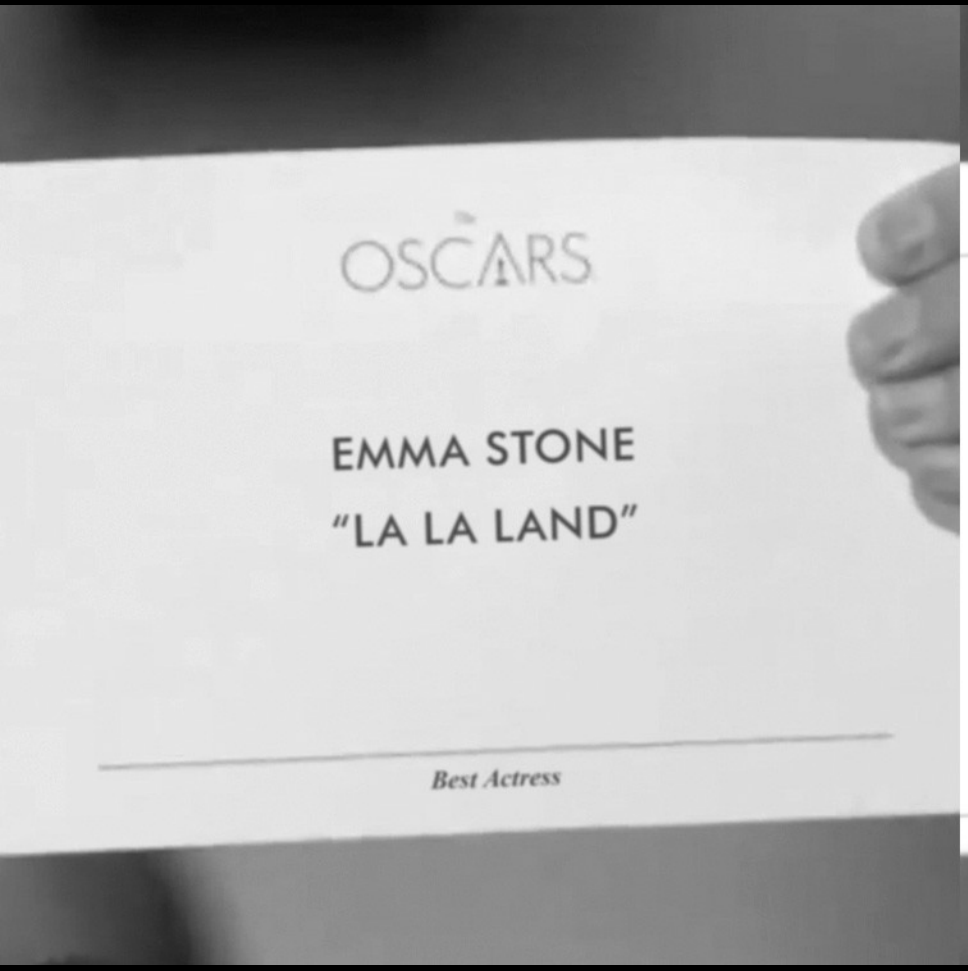
Moods

Typeface

Basic Anatomy




parmalat
milk
2% reduced
fat milk
38% less fat than whole milk
Vitamin A & D
Grade A UHT
One Quart (946mL)


OSCARS
EMMA STONE
"LA LA LAND"
Best Actress


Best Actress
EMMA STONE
"LA LA LAND"

Computational Type

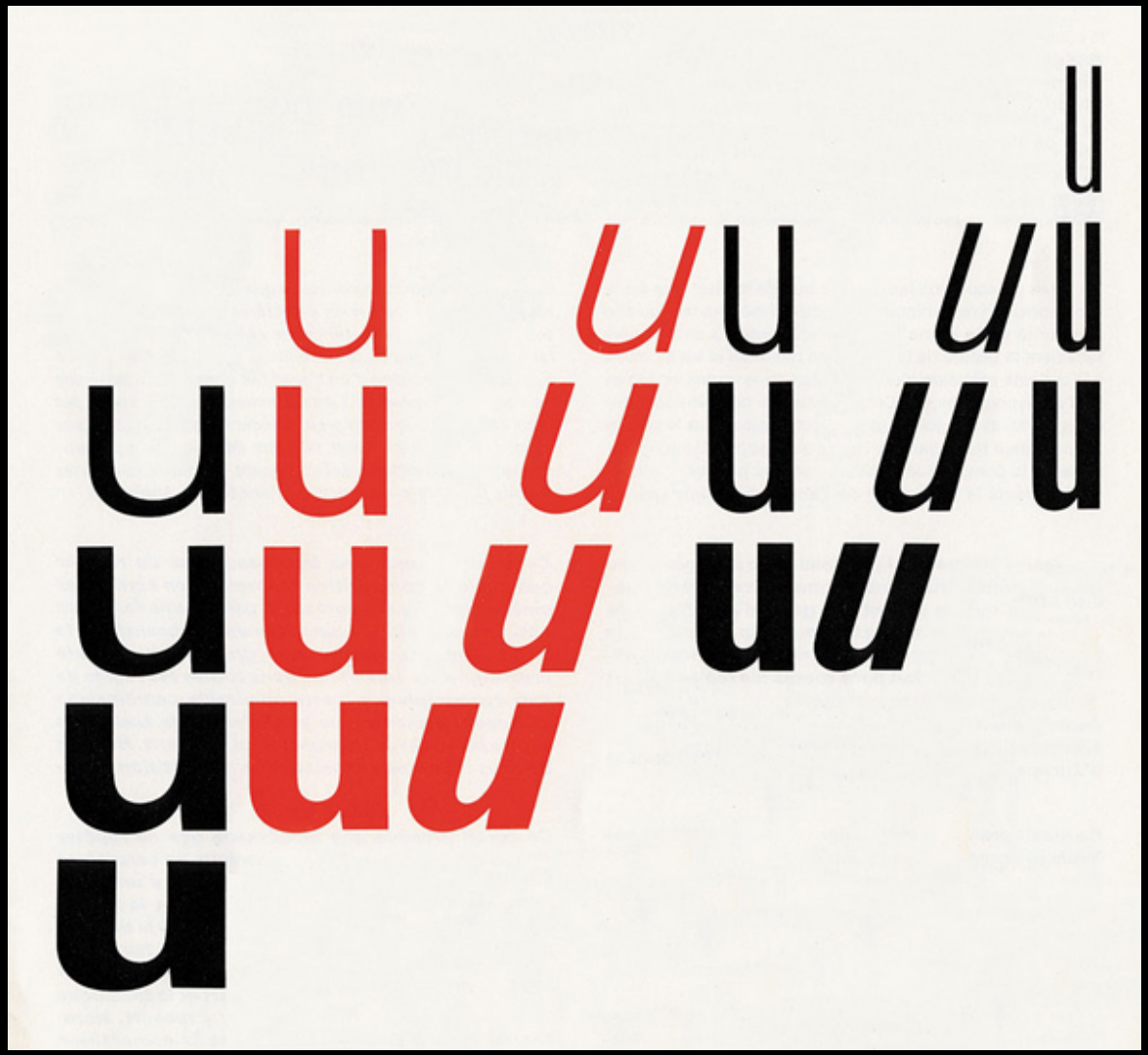
Rule-Based Typography

Creating a typeface from scratch using rules and parameters.

Outline Based

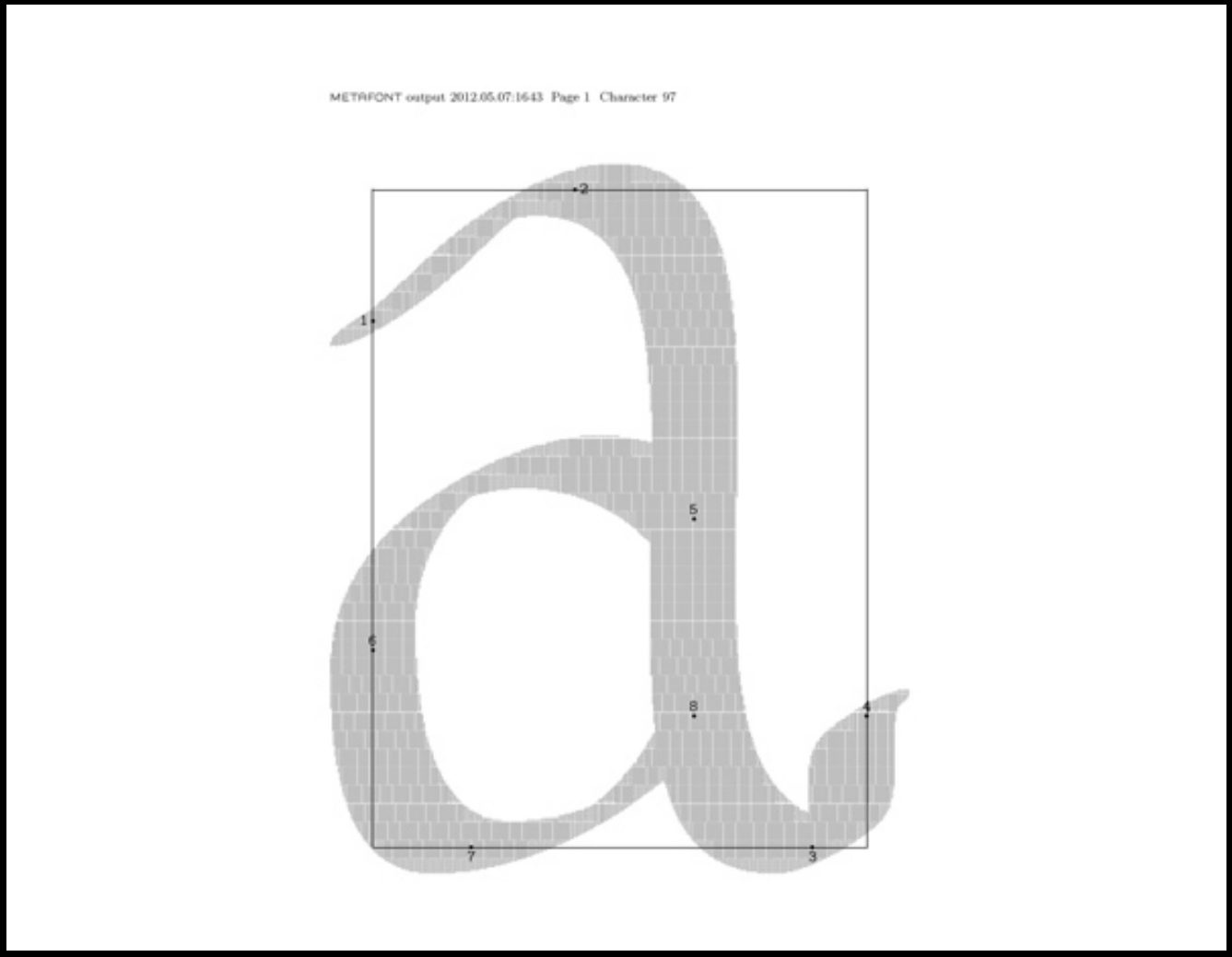
Starting with a pre-existing font and modifying the outline or interior based on parameters.

Rule-Base Type



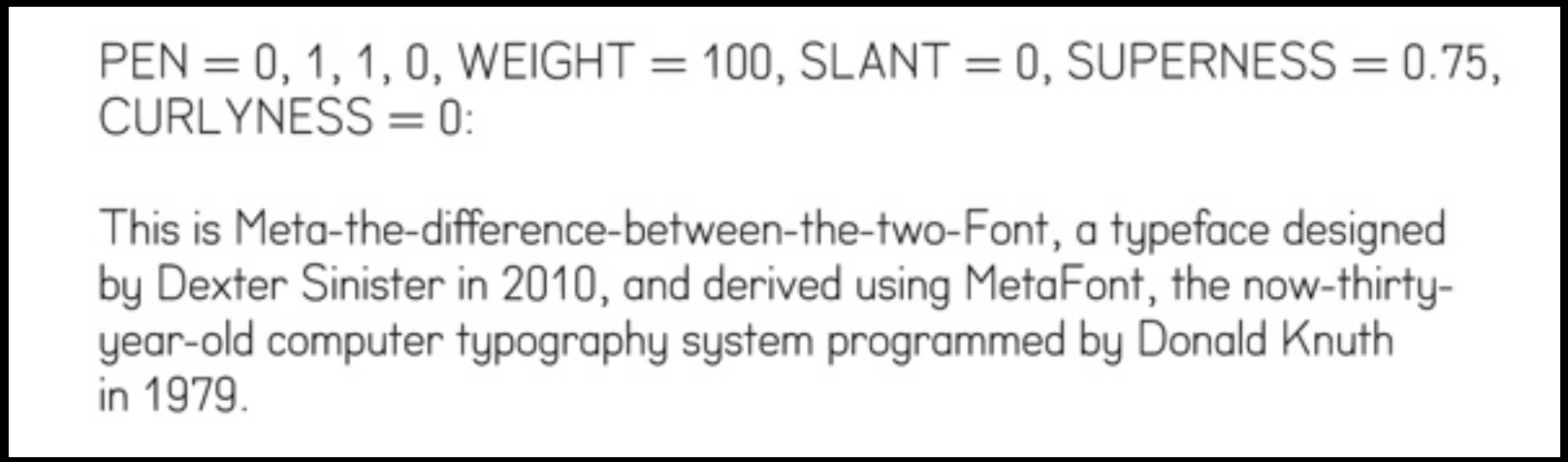
Rule-Base Type

MetaFont



Rule-Base Type

MetaFont



Rule-Base Type

MetaFont

OK, let's change the parameters of what you have been reading by setting the following excerpt from a lecture by Bruno Latour titled "What is the Style of Matters of Concern?" (2005) in Meta-the-difference-between-the-two-Font with PEN = 0, 1, 1, 30, WEIGHT = 25, SLANT = -0.1, SUPERNESS = 0.75, CURLYNESS = 30. Like so:

Imagine the following scene: you are trying to build a bridge over a rather tumultuous river. Let's say that one bank of this river is the "social" and the other, far away, inaccessible, separated by a violent current, by many eddies and dangerous rapids, is the "natural." Now suppose that, instead of trying to cross this river and build this bridge, you decide instead to GO WITH THE FLOW, that is, to get involved in a bit of canoeing, kayaking or rafting. Then the absence of a bridge is not such a problem. What counts is your ability to equip yourself with the right paraphernalia so that you can go down the river without drowning yourself. You might be scared to get into the turbulent river, you might regret the task of bridge building, but you will probably agree that the two riverbanks are bound to look rather different once you apprehend both of them from the point of view of such a kayaking movement forward. This flowing lateral direction, turned at 90° from the obsessive question of bridge building, is, if I am not mistaken, what William James has called "pure experience."

Rule-Base Type

MetaFont

Let's try another version, demonstrated by typesetting another fragment from another Latour lecture, "A Cautious Prometheus? A Few Steps Toward a Philosophy of Design (with Special Attention to Peter Sloterdijk)" (2008) with PEN = 0, 1, 1, 0, WEIGHT = 25, SLANT = 0, SUPERNESS = 0.71, CURLYNESS = 0, SERIFS = true:

When I was young, the word design (imported to French from English) meant no more than what we now call "relooking" in French (a good English word that, unfortunately, does not exist in English). To "relook" means to give a new and better "look" or shape to something—a chair, a knife, a car, a package, a lamp, an interior—which would otherwise remain too clumsy, too severe or too bared if it were left only to its naked function. "Design" in this old and limited meaning was a way to redress the efficient but somewhat boring emphasis of engineers and commercial staff. Design occurred by adding a veneer of form to their creations, some superficial feature that could make a difference in taste and fashion. Even if design could be greatly admired, it was always taken as one branch of an alternative: look not only at the function, but also at the design. This dichotomy was true even though the best design was one that, in good modernist fashion (as it did in "functionalism"), approximated function as closely as possible. "Design" was always taken in this "not only ... but also" balance. It was as if there were really two very different ways of grasping an object: one through its intrinsic materiality, the other through its more aesthetic or "symbolic" aspects.

Rule-Base Type

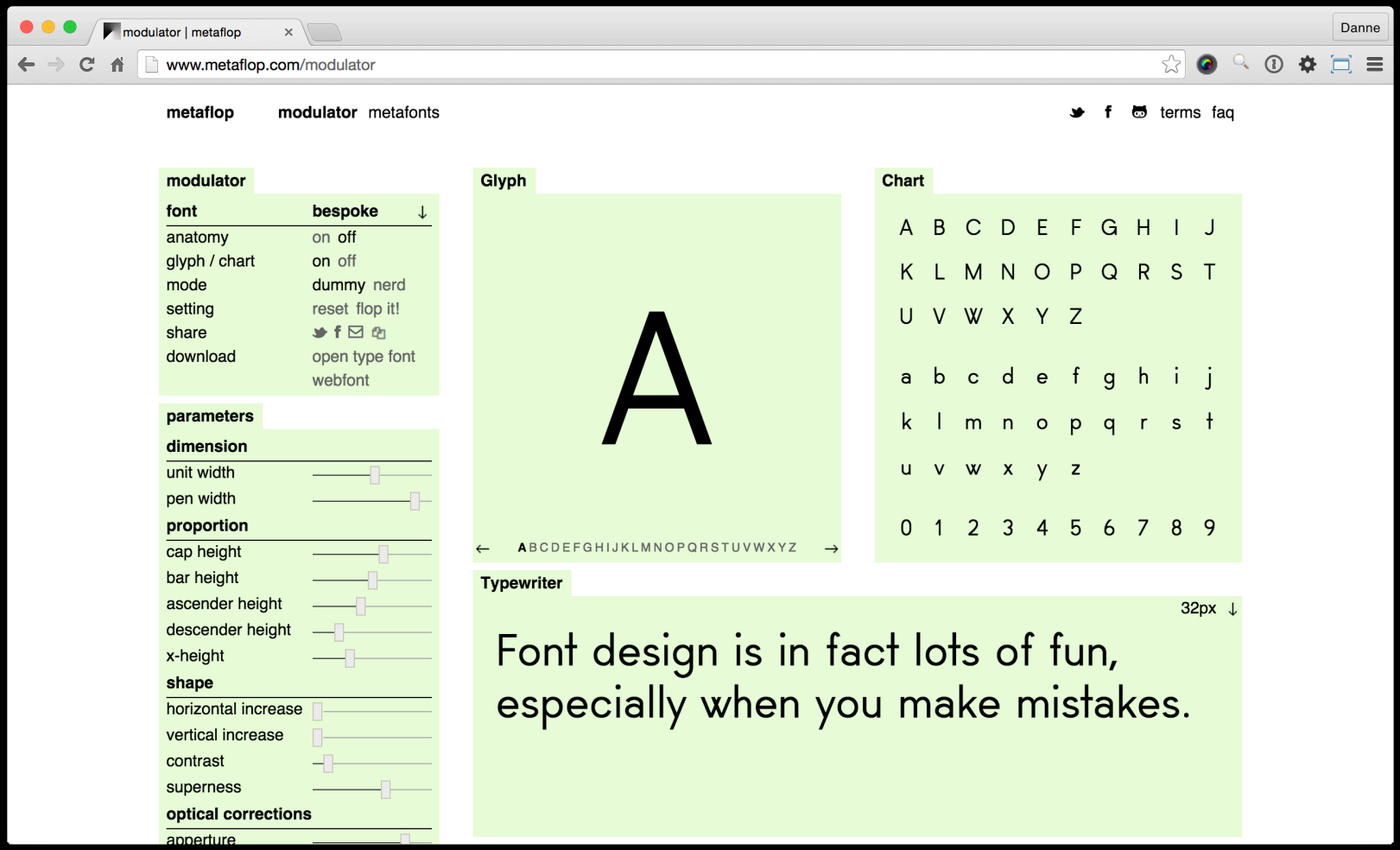
MetaFont

One final trial, this time used to set a piece of Latour's most recent essay at the time of writing, "An attempt at a 'Compositionist Manifesto'" (2010) with PEN = 0, 1, 1, 0, WEIGHT = 120, SLANT = 0, SUPERNESS = 0.5, CURLYNESS = 0:

I know full well that, just like the time of avantgardes or that of the Great Frontier, the time of manifestos has long passed. Actually, it is the time of TIME that has passed: this strange idea of a vast army moving forward, preceded by the most daring innovators and thinkers, followed by a mass of slower and heavier crowds, while the rearguard of the most archaic, the most primitive, the most reactionary people, trails behind. During this recently defunct time of time, manifestos were like so many war cries intended to speed up the movement, ridicule the Philistines, castigate the reactionaries. This huge warlike narrative was predicated on the idea that the flow of time had one—and only one—INEVITABLE and IRREVERSIBLE direction. The war waged by the avant-gardes would be won, no matter how many defeats they suffered. What this series of manifestos pointed to was the inevitable march of PROGRESS. So much so that these manifestos could be used like so many signposts to decide who was more "progressive" and who was more "reactionary."

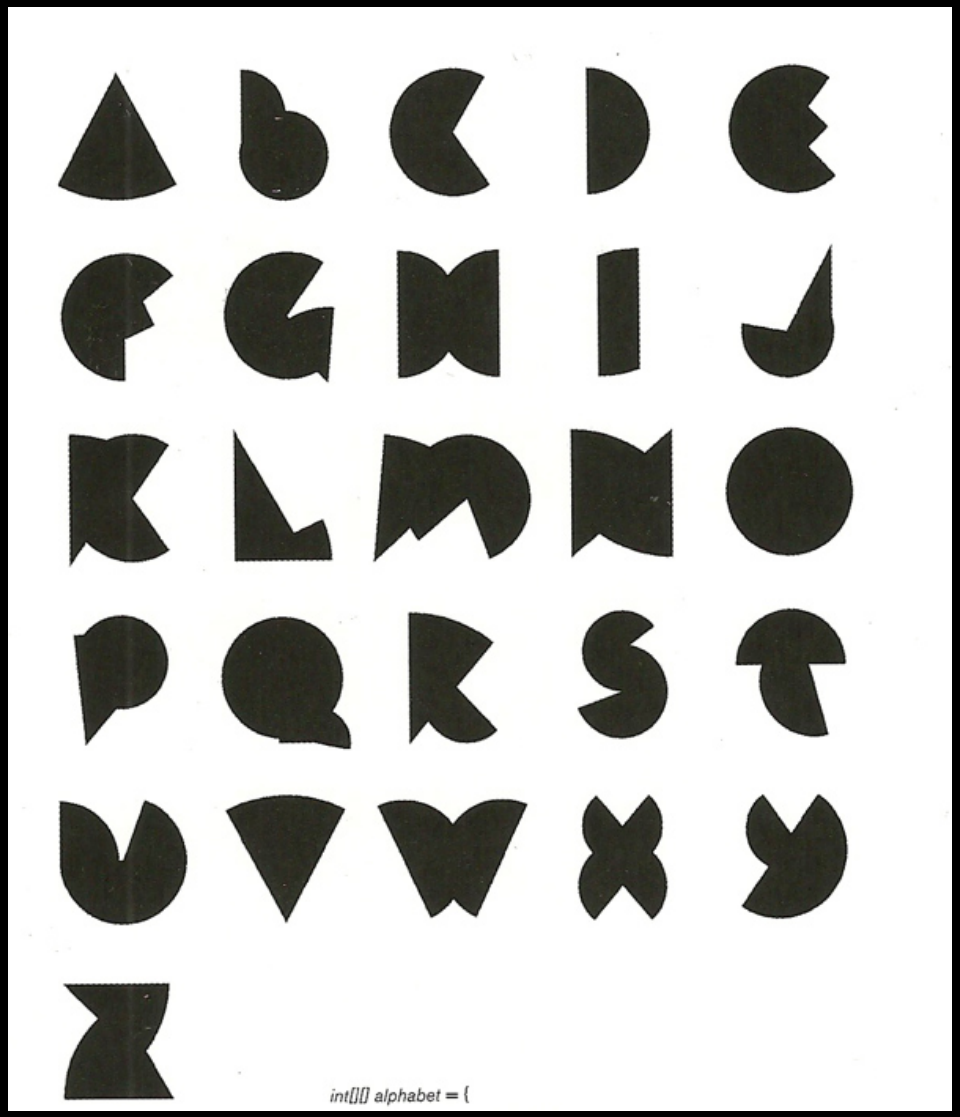
Rule-Base Type

MetaFlop



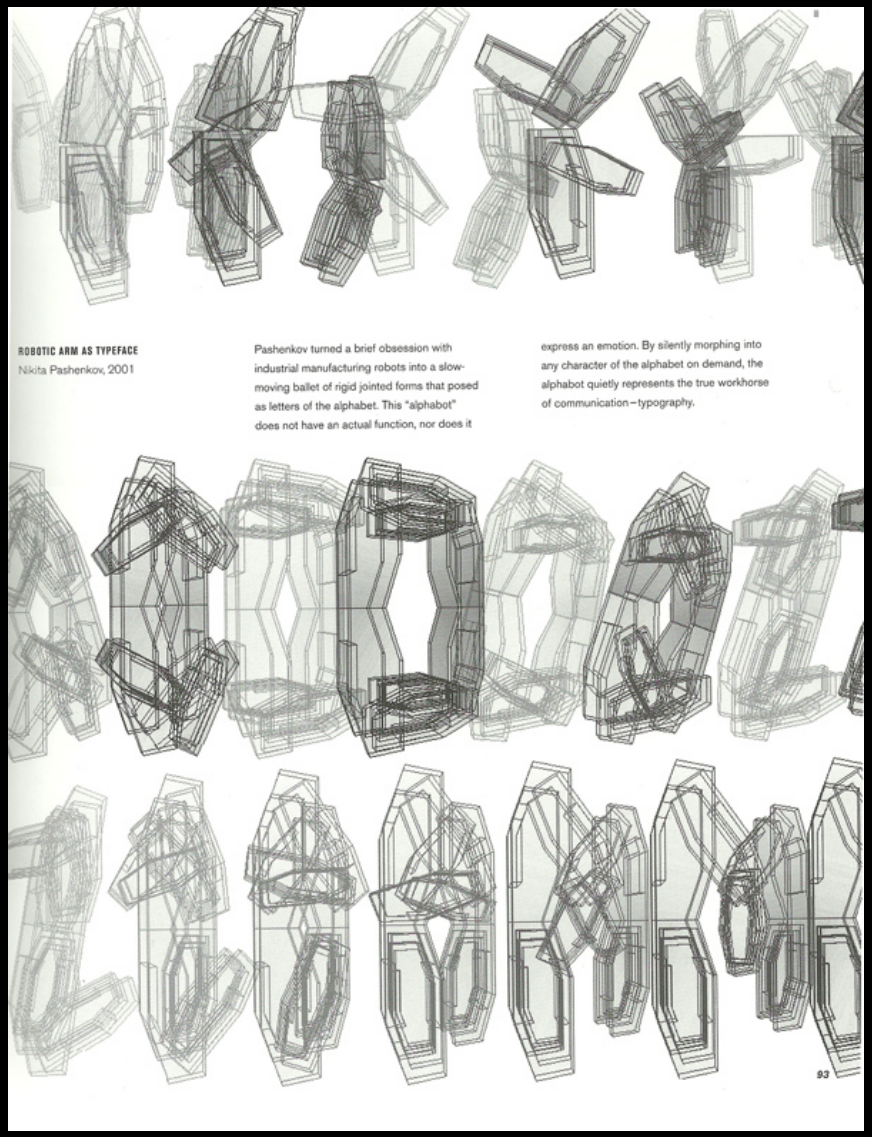
Rule-Base Type

int [] [] alphabet = { }



Rule-Base Type

Robotic Arm as Typeface



ROBOTIC ARM AS TYPEFACE
Nikita Pashenkov, 2001

Pashenkov turned a brief obsession with industrial manufacturing robots into a slow-moving ballet of rigid jointed forms that posed as letters of the alphabet. This "alphabot" does not have an actual function, nor does it express an emotion. By silently morphing into any character of the alphabet on demand, the alphabot quietly represents the true workhorse of communication—typography.

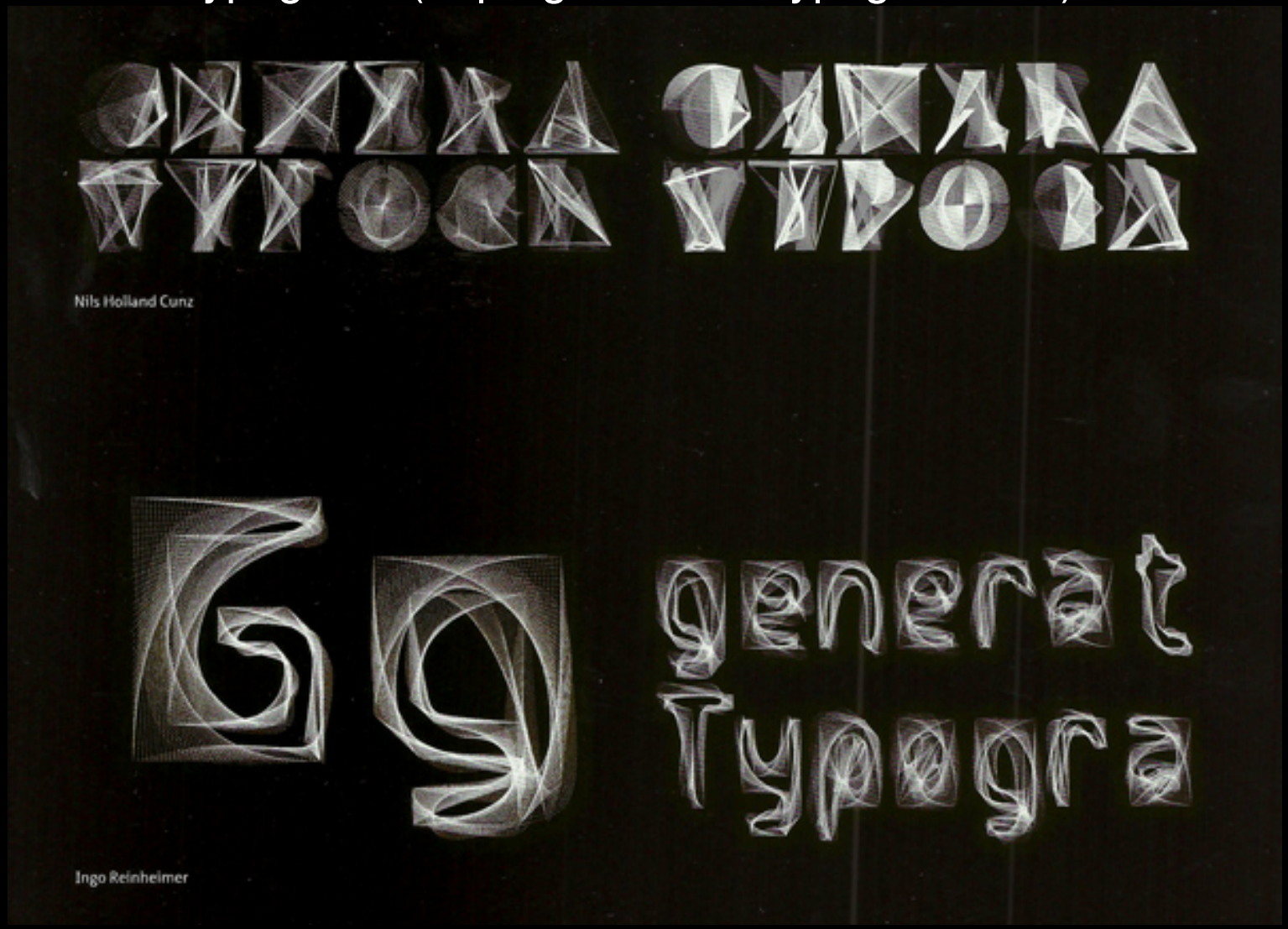
93

Rule-Base Type

Lava Sans Avec

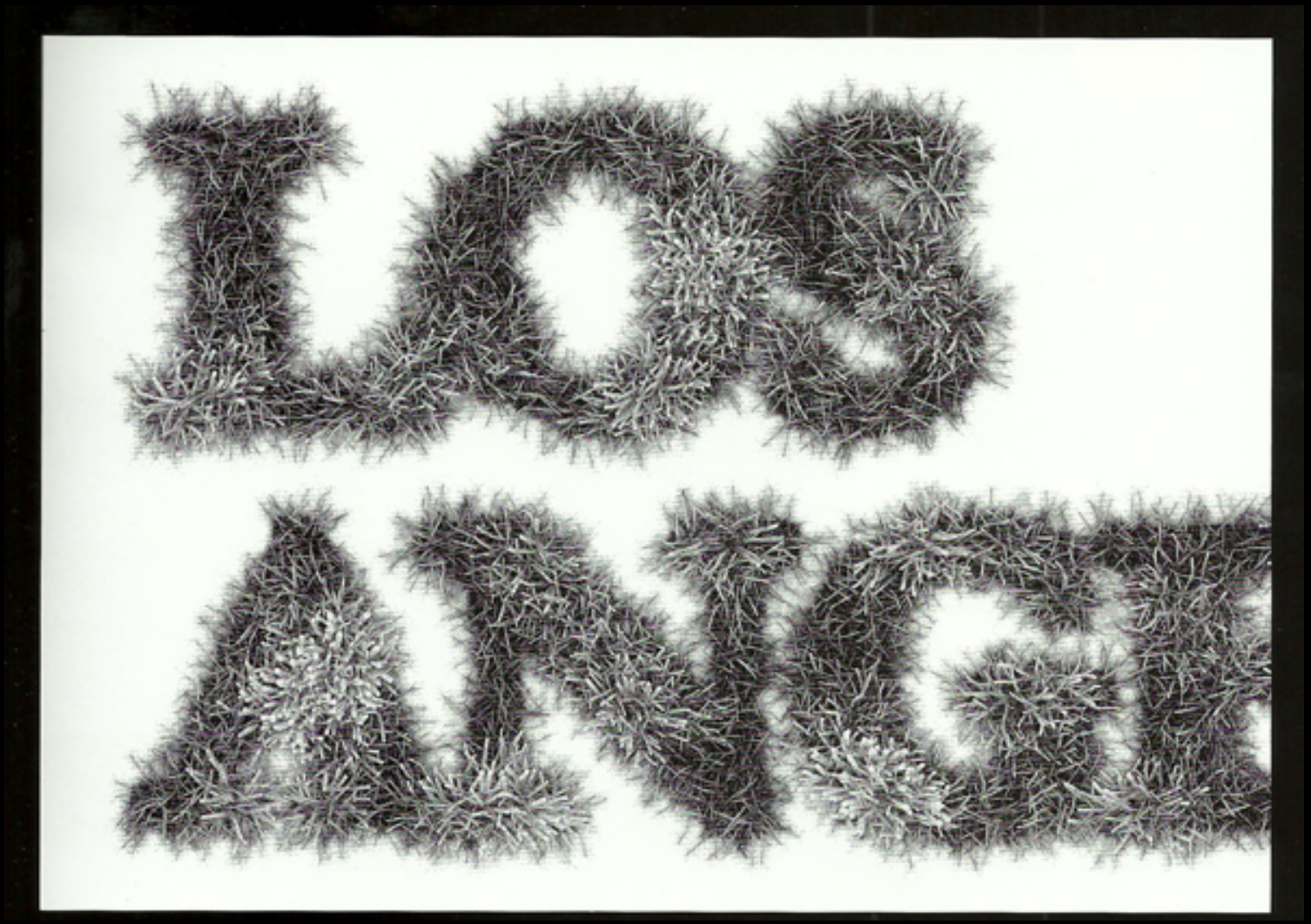
Outline Based

Generative Typografie (http://generative-typografie.de/)



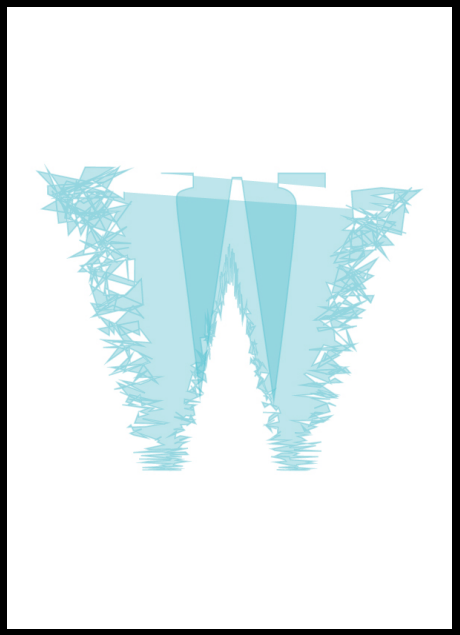
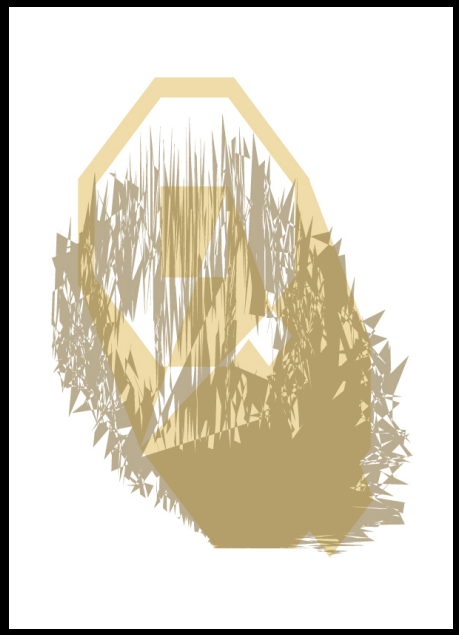
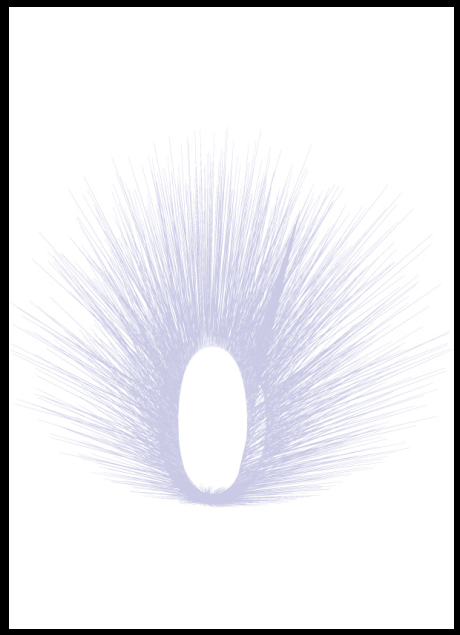
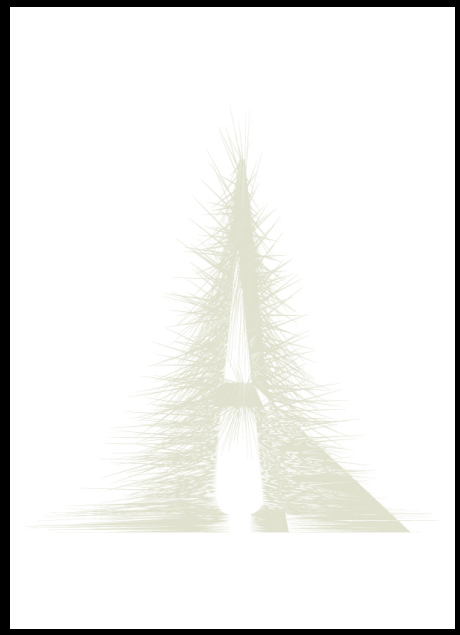
Outline Based

Growing Data



Outline Based

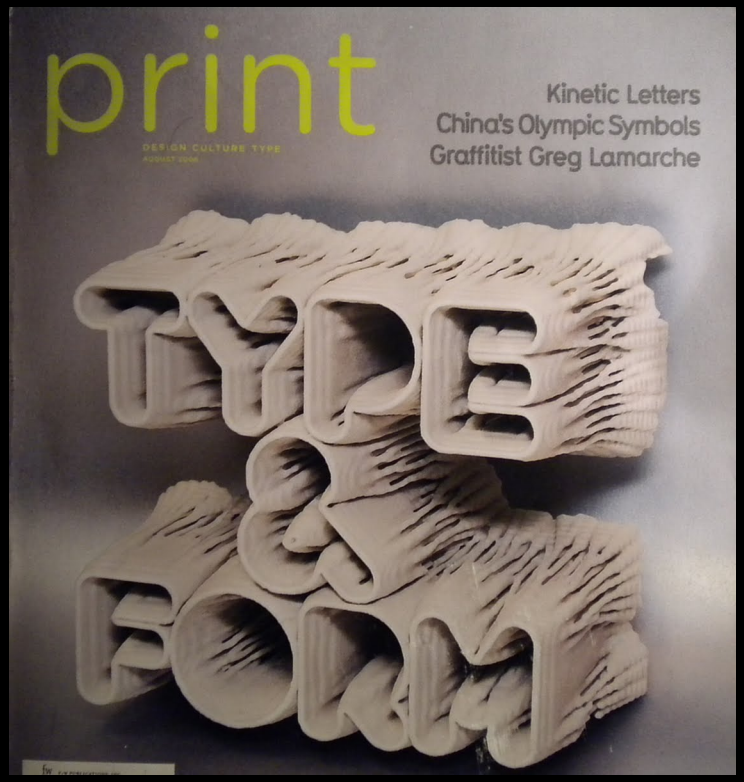
Type Galapagos

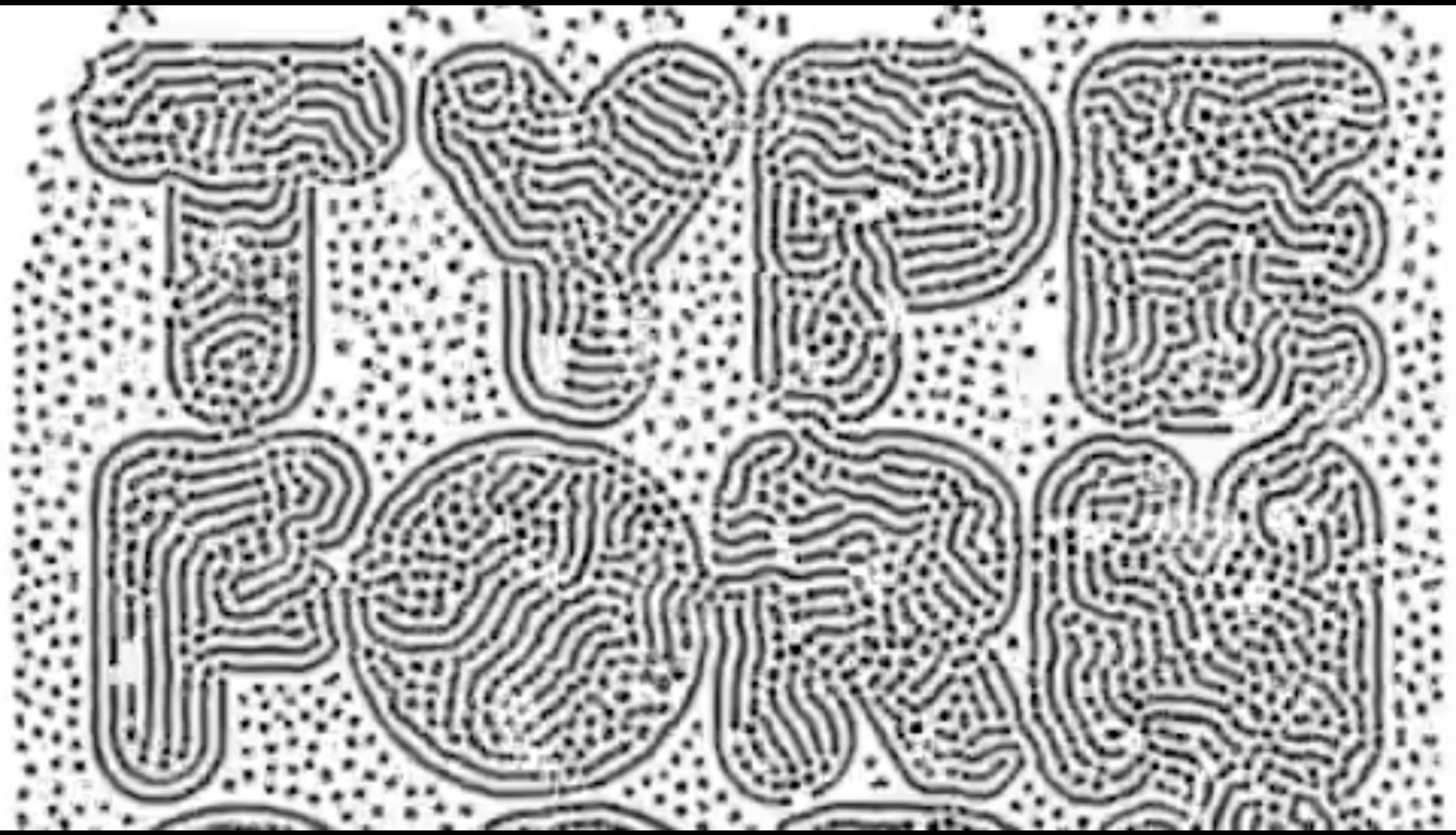





Outline Based

Outline Based

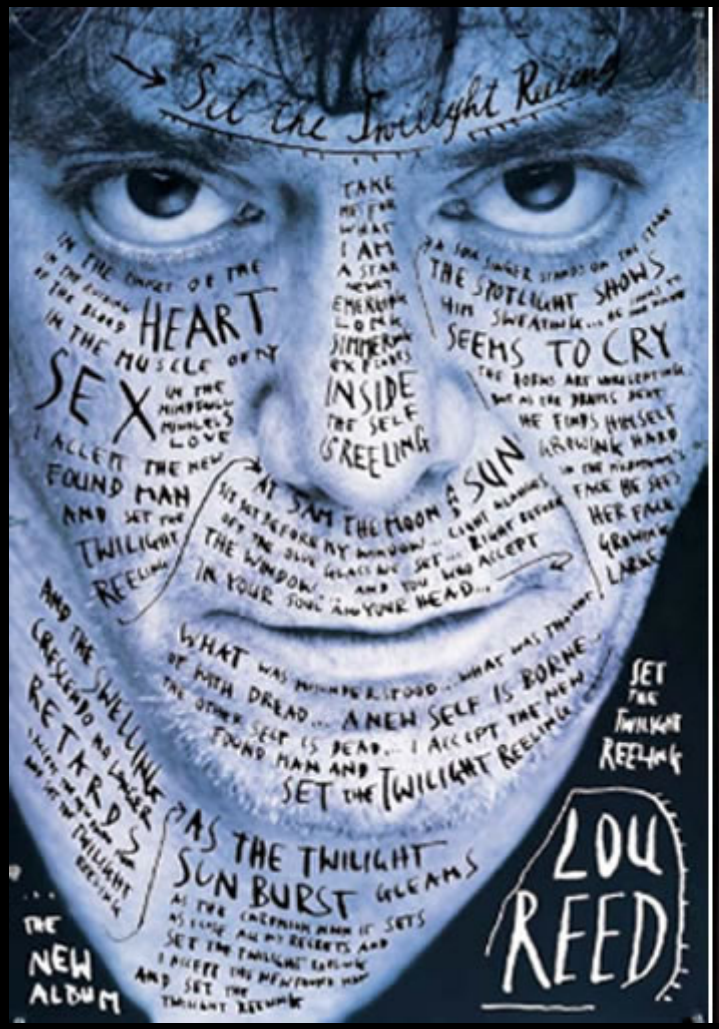
Type and Form, for Print Magazine

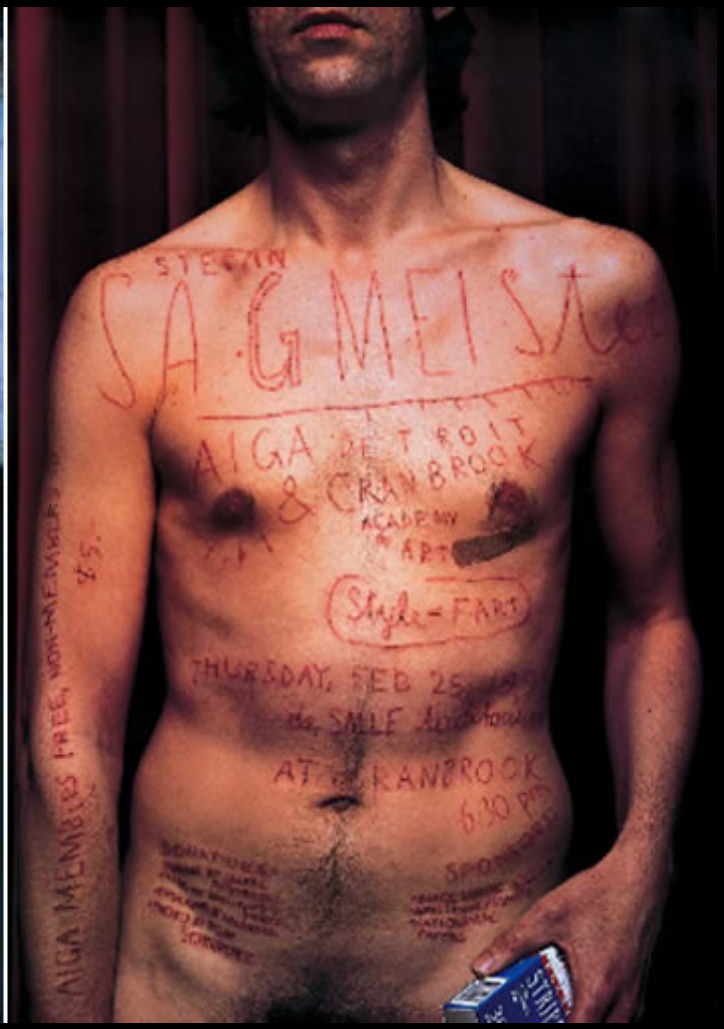




Artistic Type

Stephan Sagmeister





Artistic Type

Stephan Sagmeister

Artistic Type

HOTTEA

Artistic Type

5 Pointz

Artistic Type

David Carson

Typography

Simple Text - text(), textFont(), preLoad(), loadFont(), textSize(), textLeading(), textAlign(), textWidth(), textStyle(), p5.Font, textToPoints()

Arrays

```
var variableName1 = [19, 32, 87];
variableName1[0] = 19, variableName1[1] = 32, variableName1[2] = 87
var variableName2 = [ [19, 32, 87], [24, 82, 65], [72, 94, 87] ];
variableName2[1][2] = 65
variableName1.length;  //3
```

Typography

Simple Text

```
var word = "Hello world!";

function setup() {
  createCanvas(600, 400);
  background(255);
  fill(0);
  textSize(48);
  textAlign(CENTER);
  text(word, width/2, height/2 - 30);
  text(textWidth(word), width/2, height/2 + 30);
}
```

Typography

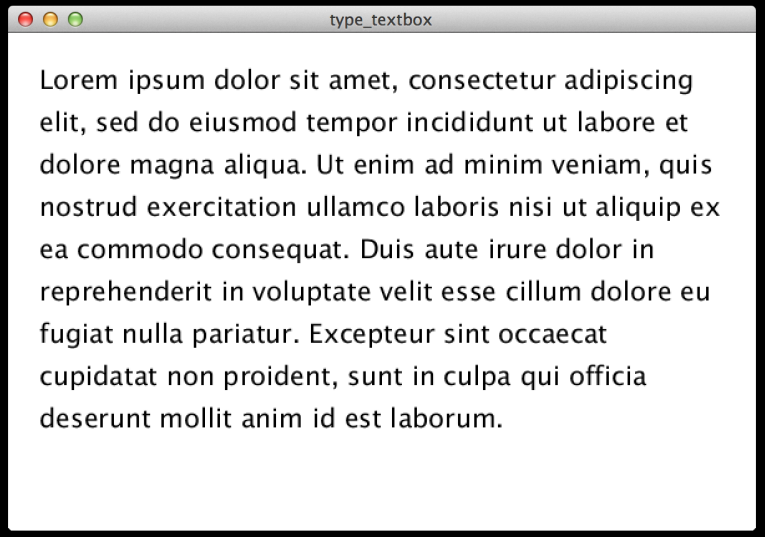
Simple Text

```
textSize(Number);
//size

textAlign(alignX, alignY);
// LEFT, CENTER, or RIGHT and TOP, BOTTOM, CENTER, or BASELINE

text(text, x, y);
// Pass in a string of text, x location and y location
```

Typography

Simple Textbox

```
var paragraph = "Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua…";

function setup() {
  createCanvas(600, 400);
  background(255);
  fill(0);
  textSize(21);
  textAlign(LEFT);
  textLeading(34);
  text(paragraph, 25, 25, width - 50, height - 50);
}
```



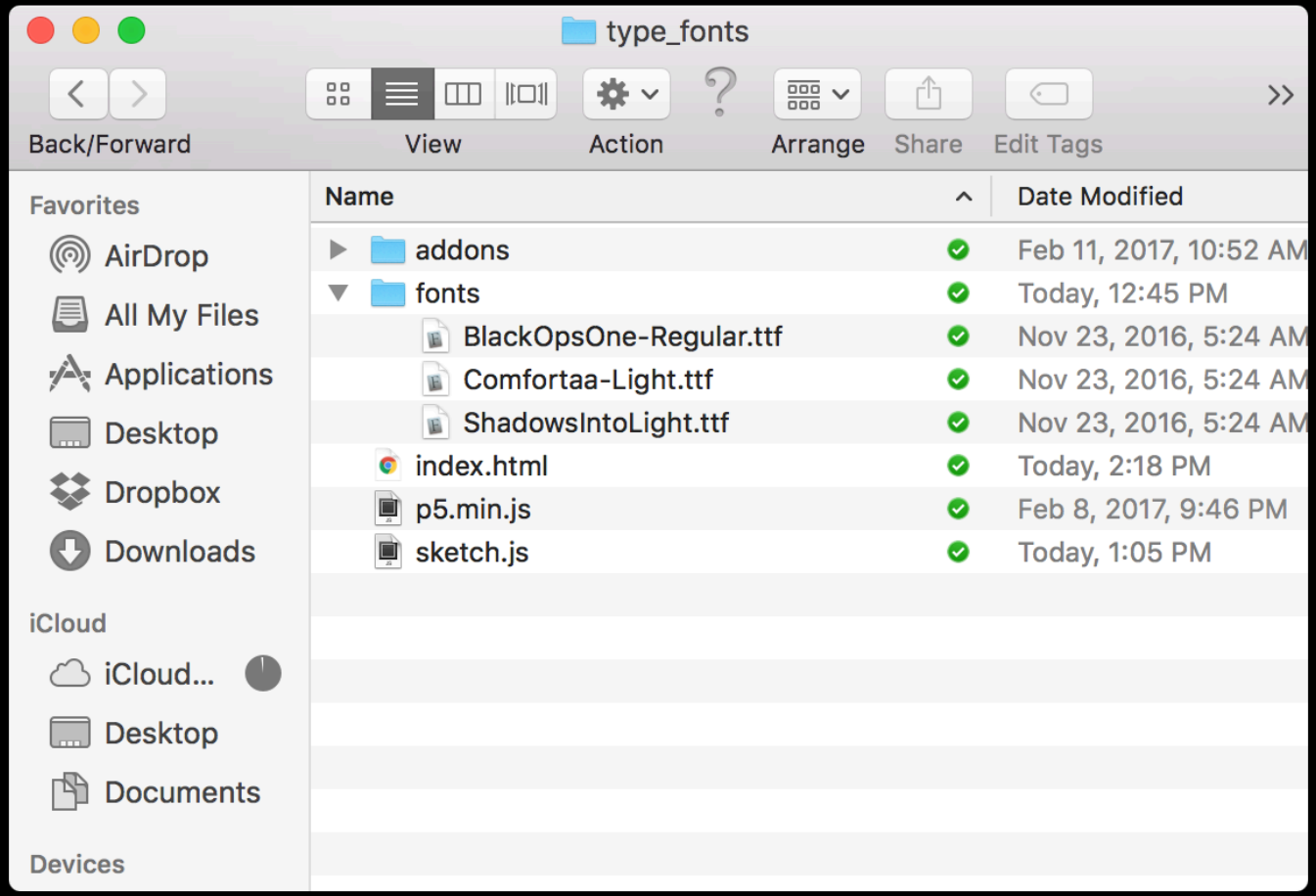
Typography

Simple Textbox

```
textLeading(Number);
// Leading size

text(text, x, y, w, h);
// Pass in a string, x location, y location, width and height
```

Typography

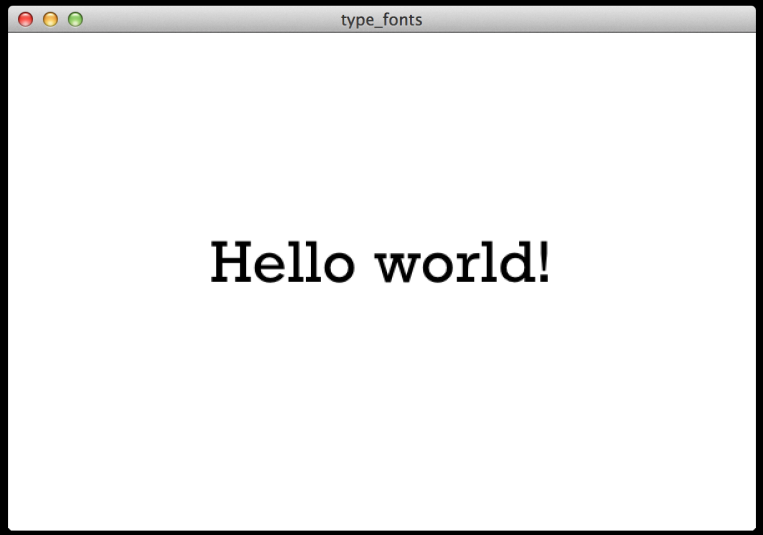
Fonts (TTF or OTF)



Typography

Fonts

```
var word = "Hello world!";
var font1, font2, font3;

function preload() {
  font1 = loadFont("fonts/BlackOpsOne-Regular.ttf");
  font2 = loadFont("fonts/Comfortaa-Light.ttf");
  font3 = loadFont("fonts/ShadowsIntoLight.ttf");
}

function setup() {
  createCanvas(600, 400);
  background(255);
  fill(0);
  textFont(font3, 100);
  textAlign(CENTER);
  text(word, width/2, height/2);
}
```



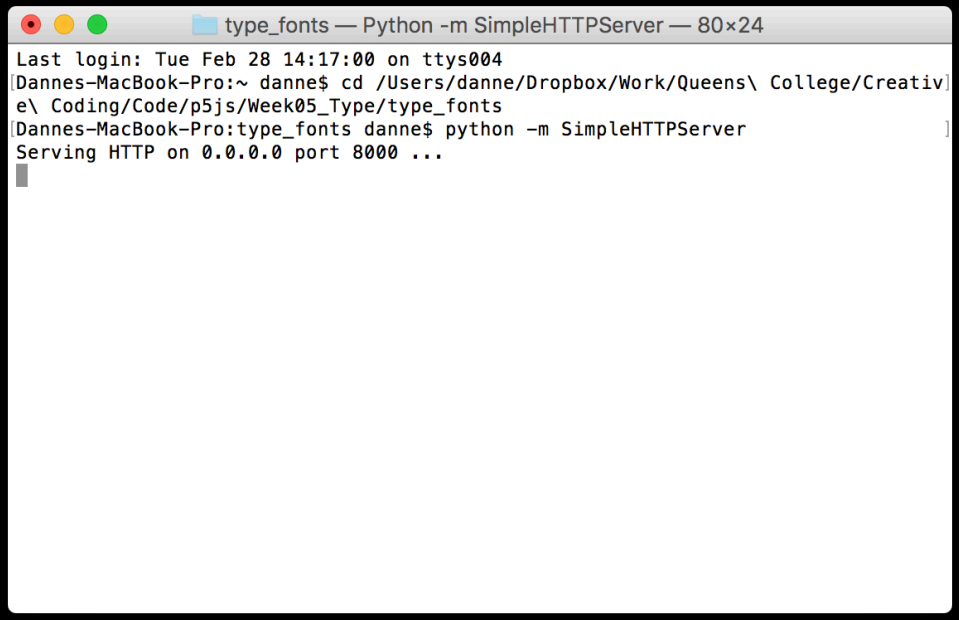
Typography

Fonts

If you are not using the web editor and programming on your local machine.
Error: Font loading error.

Fix: Need to Run a Local Server

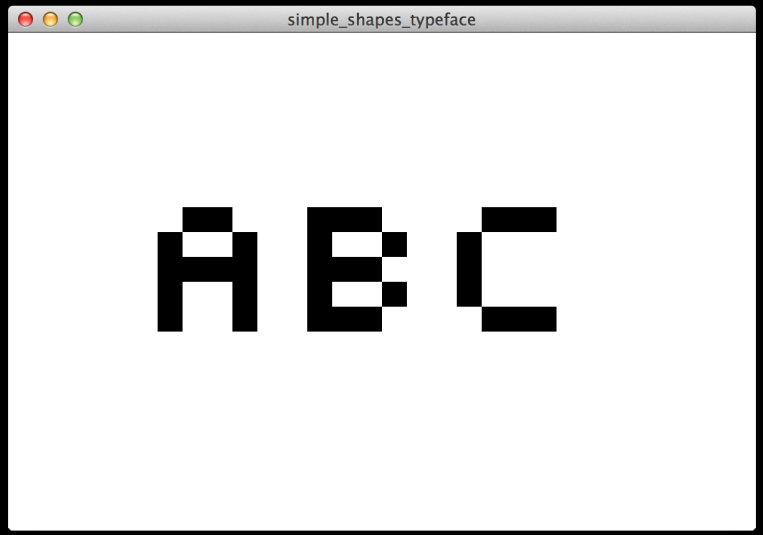
1. **Open Terminal**
2. Type **cd directory/folder/location** you can also click and drag the folder to Terminal.
3. Press **Enter**
4. Type **python -m SimpleHTTPServer**
5. Press **Enter**
6. Go to a web browser and type in **localhost:8000** for the URL. This will load the index.html file in that folder.



Typography

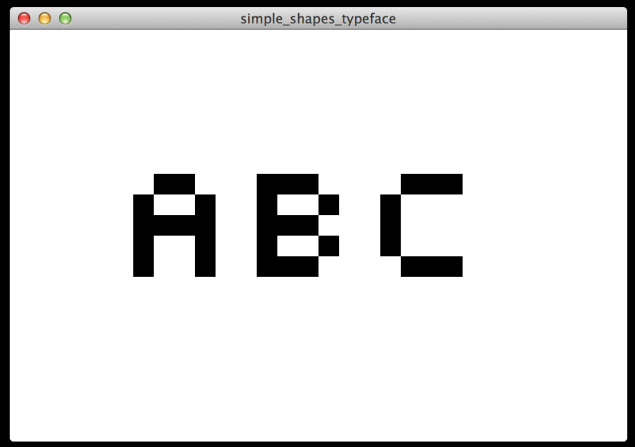
Type System

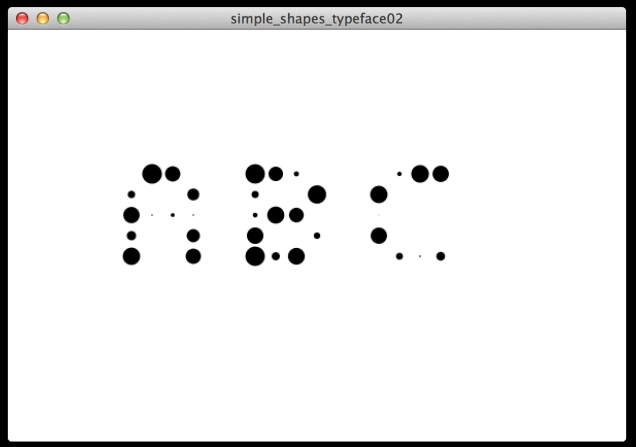
```
var letters = [ [ 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1 ], //A
[ 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0 ], //B
[ 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1 ] ]; //C
var blockSize = 20;
function setup() {
  createCanvas(600, 400);
  noStroke();
  background(255);
  fill(0);
  for(var i = 0; i < letters.length; i++) {
    var xPos = 0, yPos = 0;
    for(var j = 0; j < letters[i].length; j++) {
      if(letters[i][j] == 1) {
        rect(xPos, yPos, blockSize, blockSize);
      }
      xPos += blockSize;
      if(j % 4 == 3) {
        xPos = 0;
        yPos += blockSize;
      }
    }
    translate(blockSize * 6, 0);
  }
}
```

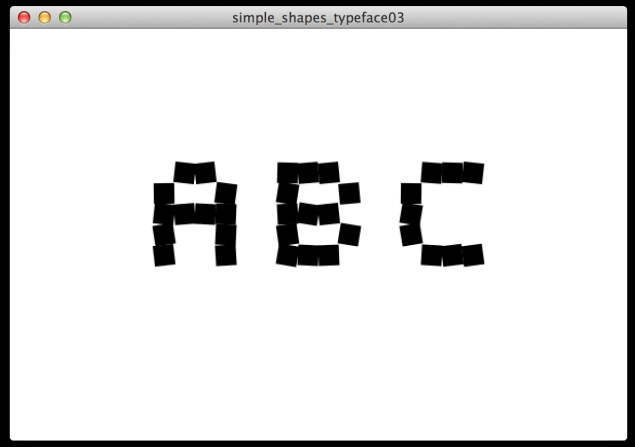


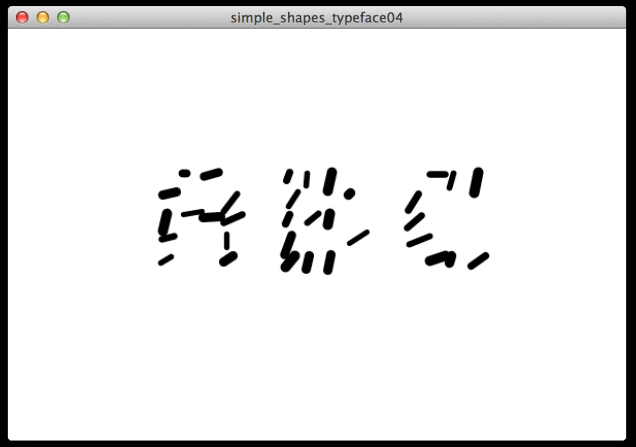
Typography

Type System







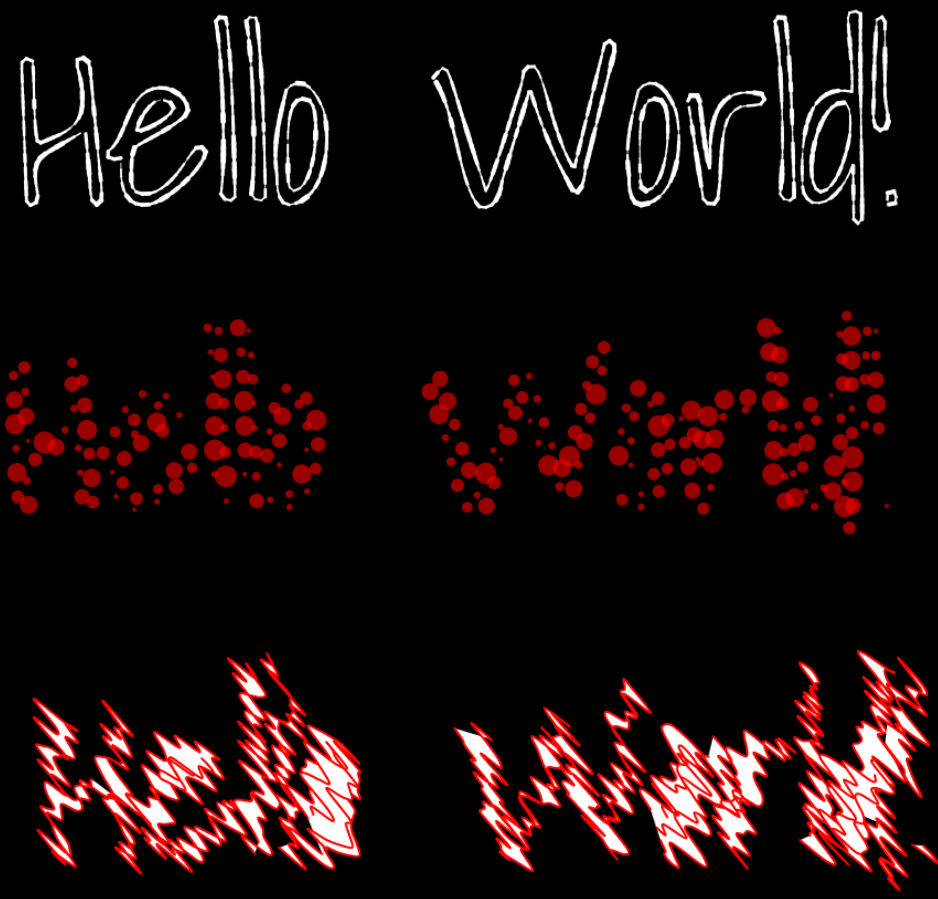


p5.Font

Get the Points

```
var font;
function preload() {
  font = loadFont('fonts/ShadowsIntoLight.ttf');
}

var points;
function setup() {
  createCanvas(300, 400);
  stroke(0);
  points = font.textToPoints('p5', 100, 100, 100, { sampleFactor: 0.15 });

  background(255);
  for (var i = 0; i < points.length; i++) {
    var p = points[i];
    ellipse(p.x, p.y, 3, 3);
  }
}
```

p5.Font



Rune.font

Get the Points

```
var fontSize = 48;
var word = "Hello World!";
var path, polys;
function setup(){
    createCanvas(600,400)
    background(0)
    f = new Rune.Font("fonts/ShadowsIntoLight.ttf")
    // load the font
    f.load(function(err){
        // this is a rune function
        path = f.toPath(word, 0, 0, 100);
        // this is another handy function to get polygons coordinates
        polys = path.toPolygons({ spacing:1 });
    })
}
```

In Class

1. In teams of two, sketch out a rule-based typeface.
2. Once you have an idea of what to create, try and make A, B and C in p5js based on the rules you put together.

Homework

1. **Option 1:** Pick a word and use p5js to design a typeface that represents that word. You can either create a rule-based typeface using simple shapes or use p5.Font to modify an existing typeface. You must use at least one of the following; for loop, if statement, random or noise. Meaning do not do something you can easily create in Illustrator.
   Option 2: Pick your favorite quote or poem and use different type treatments in p5js to create a layout that visually represents the selected quote or poem. You can only use text.
2. **Put your p5js sketch in the Dropbox folder before our next class and be prepared to talk about it.**

Creative Coding

Professor Danne Woo

dwoo@qc.cuny.edu

creativecode.dannewoo.com