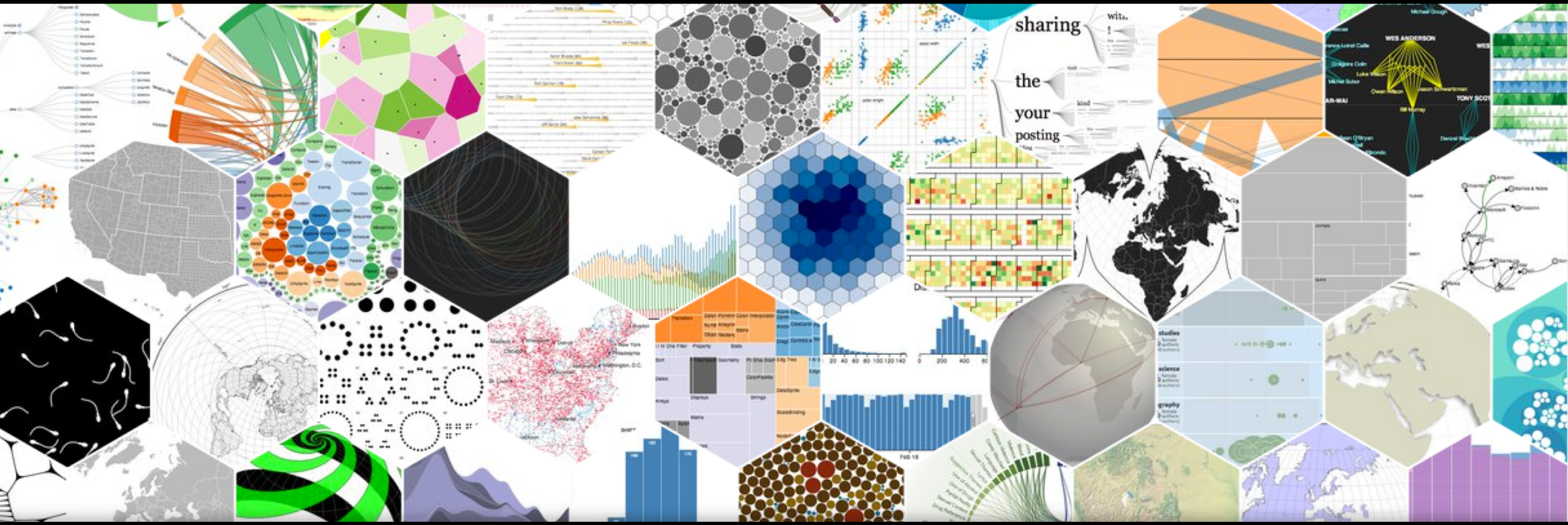
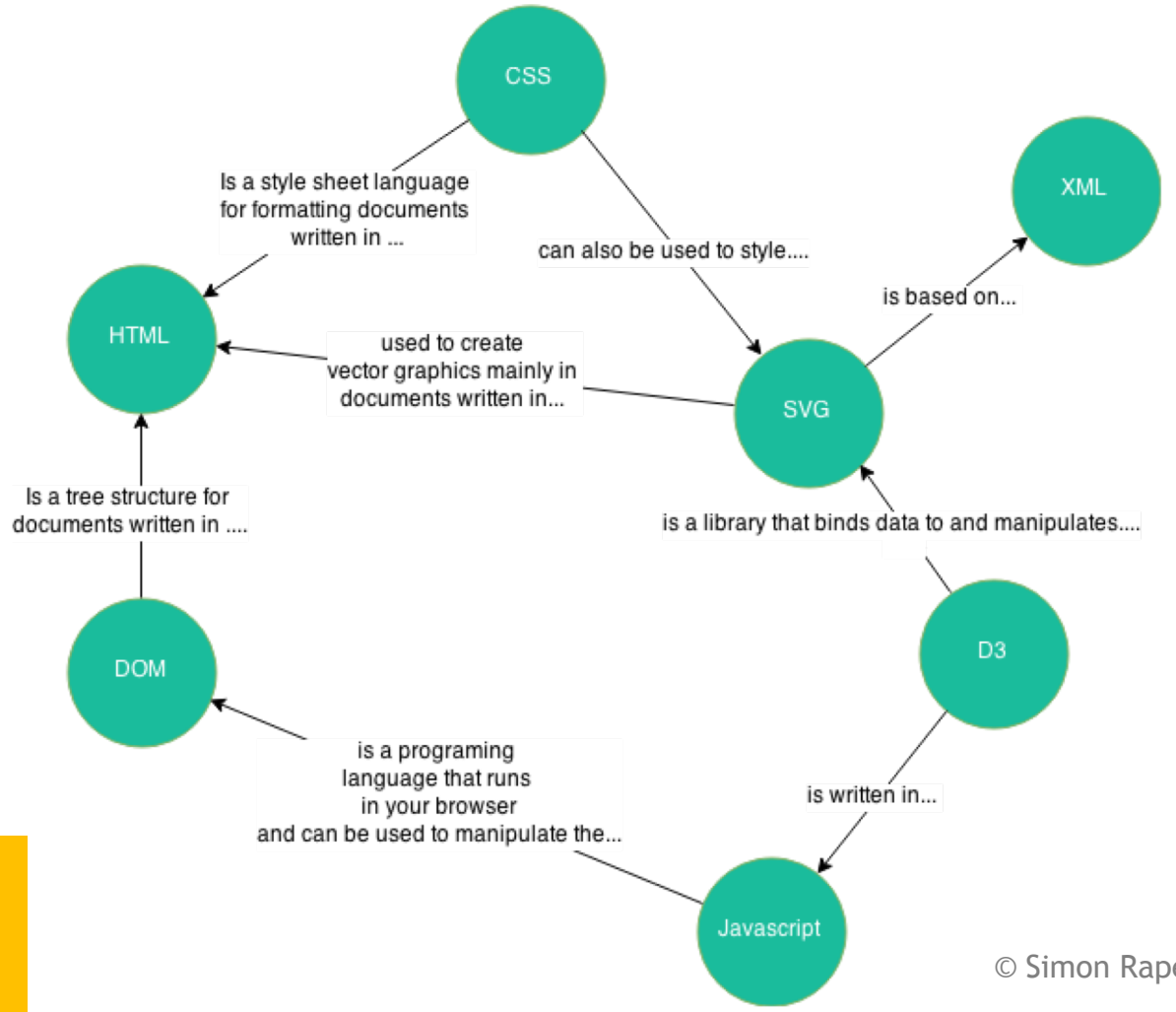
Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin



INTRODUCTION TO D3.JS

ELEMENTS AND WHY YOU NEED THEM

- **DOM** encodes the data structure and functionalities of a webpage accessed through HTML and JS
- Everything on a webpage is scripted in **HTML** at the top most level (i.e. wrapped inside HTML)
- **CSS** used to define look and feel of the page e.g. colours, styles
- **Javascript** is a basis for coding any dynamic elements and programmatic elements, loops, classes, functions, variables through JS. JS used to generate HTML/CSS code dynamically (there are also other alternatives such as php)
- **D3.JS** is a javascript library used to provide more generalized and high level functions to create visualizations
- **SVG** defines standard vector graphics, the basic building blocks in d3js for creating graphical objects



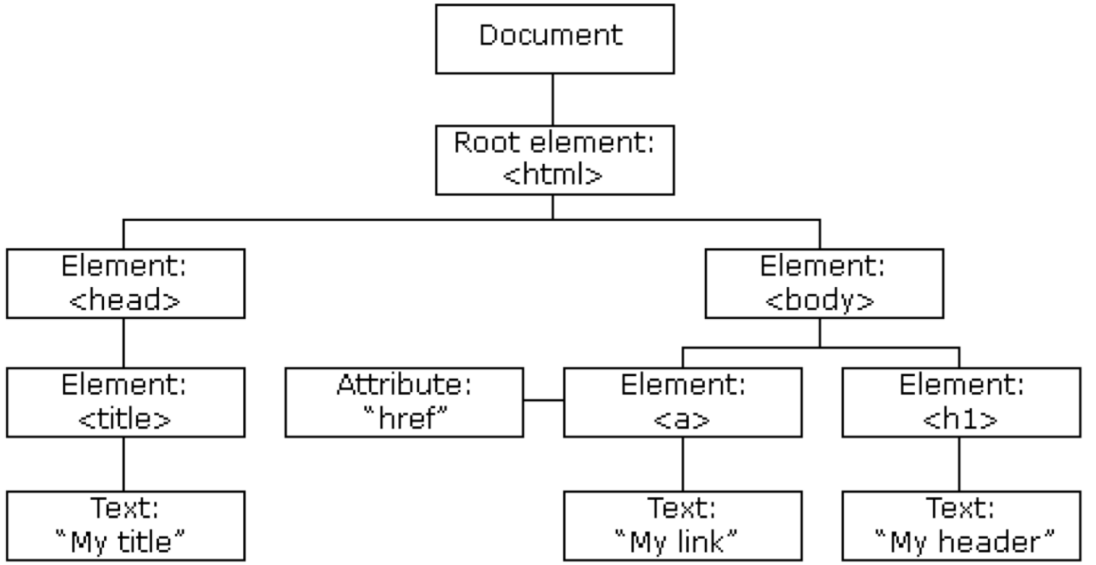
PLEASE NOTE YOU DON'T NEED TO BE AN EXPERT AT ALL OF THESE. For this module (and generally for visualization) you mainly need to learn javascript, d3.js and some SVG

© Simon Raper

DOCUMENT OBJECT MODEL (DOM)

Document Object Model

- a cross-platform and language-independent application programming interface for webpages that run in your browser
- essentially a data structure that holds all the elements of a webpage e.g. graphics, link, text blocks, form
- These are …
  - scripted with HTML
  - styled with CSS
  - and can be altered using javascript



Short intro

- https://www.w3schools.com/js/js_htmldom_elements.asp

Full reference (you don't really need this):

- https://www.w3schools.com/js/js_htmldom_elements.asp

HTML

Hypertext Markup Language (HTML)

- Building blocks of web pages
- Mainly for defining content i.e. specifying the text, images, videos, forms, links that exist on a web page

- NOT really intended for
  - Look and feel of elements (instead see CSS)
  - Low level specification of visual elements (instead see SVG and D3js)

```
<html>
  <head>
    <title>Hello world</title>
  </head>
  <body>
    <p>Some not basic paragraph text.</p>
    <a href="http://www.tcd.ie">A LINK</a>
    <img src="tcdlogo.jpg" >
  </body>
</html>
```

Save this code in a text file with **.html** extension and open it in a browser

CSS

Cascading Style Sheets

- Language for scripting appearance of HTML pages
- **Layout** of elements e.g. images, blocks of text, etc.
- **Styling** of content e.g. text colour, font, position, borders, table appearance, fill colour

```
<html>
  <head>
    <title>Hello world</title>
  </head>
  <body>
    <p style="color:red; font-family:arial; font-style:italic; font-size:28px; >Some not basic paragraph text.</p>
    <a href="http://www.tcd.ie">A LINK</a>
    <img style="width:300px;" src="tcdlogo.jpg" >
  </body>
</html>
```

JAVASCRIPT

- High-level interpreted programming language
- ECMAScript specification
- Mainly for Client-side scripting
- Relatively easy to learn.
- Supports event-driven, functional, and imperative paradigms
- HTML and CSS are static definitions, JS provides means of programmatically creating HTML & CSS elements

```
var x, y, z;

x = 5; y = 6;
z = x + y;

function myFunction(p1, p2) {
  return p1 * p2;   // The function returns the product of p1 and p2
}

var text = "The temperature is " + toCelsius(77) + " Celsius";

var person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};

var cars = ["Saab", "Volvo", "BMW"];

if (age < 18) text = "Too young";

var i;
for (i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";
}
```

Basic Tutorial https://www.w3schools.com/js/default.asp

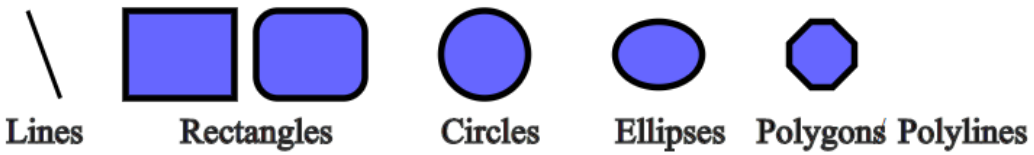
SCALABLE VECTOR GRAPHICS (SVG)

Vector-based graphics definition in XML format

Open standard developed by W3C since 1999

Includes definitions of

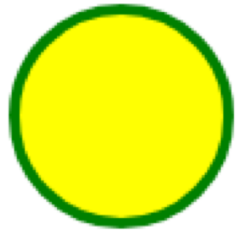
- Basic shapes, Text, Color
- Styling attributes
- Coordinate systems and transforms
- Gradients and patterns



Reference: https://www.w3.org/TR/SVG11/

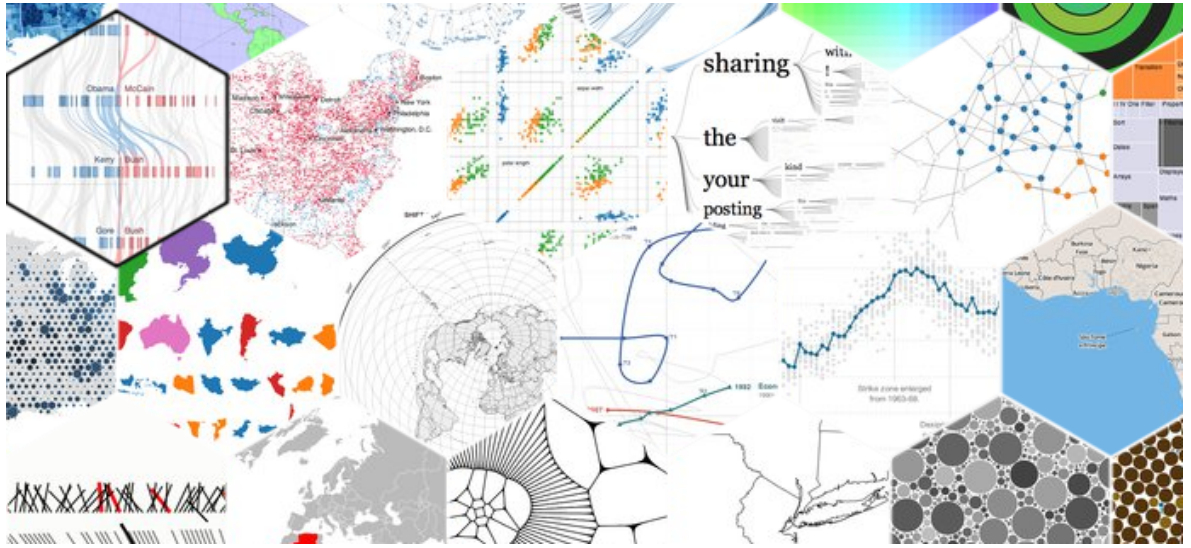
Simple Example:

```
<html>
<body>

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green"
 stroke-width="4" fill="yellow" />
</svg>

</body>
</html>
```



DATA-DRIVEN DOCUMENTS (D3.JS)

- a javascript library that allows you to bind data to the DOM and apply transformations to the document e.g. to create an interactive visualization with transitions and interactions.
- does not introduce a new visual representation; instead based directly from web standards: HTML, SVG, and CSS
- however it provides a more flexible, efficient, dynamic, fast alternative to doing things directly through these standards



More info: https://d3js.org/

Gallery: https://github.com/d3/d3/wiki/Gallery

D3.JS EXAMPLE

From: https://www.tutorialspoint.com/d3js/

Direct SVG Example

```
<html>
<head>
<script type = "text/javascript" src =
"https://d3js.org/d3.v4.min.js"></script>
<style> body { font-family: Arial; } </style>
</head>

<body>
  <div id = "svgcontainer">
            <svg width = "300" height = "300">
            <line x1 = "100" y1 = "100" x2 = "200" y2 = "200"
            style = "stroke:rgb(255,0,0); stroke-width:2"/>
            </svg>
  </div>
</body>
</html>
```

Equivalent D3.js Example

```
<html> <head>
<script type = "text/javascript" src =
"https://d3js.org/d3.v4.min.js"></script>
<style> body { font-family: Arial; } </style>
</head>
<body>
<div id = "svgcontainer"> </div>
<script language = "javascript">
            var width = 300;
            var height = 300;
            var svg = d3.select("#svgcontainer")
                        .append("svg")
                        .attr("width", width)
                        .attr("height", height);
                  svg.append("line")
                        .attr("x1", 100)
                        .attr("y1", 100)
                        .attr("x2", 200)
                        .attr("y2", 200)
                        .style("stroke", "rgb(255,0,0)")
                        .style("stroke-width", 2); </script>
</body>
</html>
```

RUNNING D3.JS PROGRAMS

d3js intended for online web visualizations in a browser.

- In theory it is intended client side scripting so you should be able to develop/test locally. However due to security restrictions these days on what browsers will allow you to run in your local machine, thus you will need to deploy it on a web server

Simplest option: Put it on a web folder and open it in a browser

Alternatively: Run it locally through a local web server.

- Mac has built in support. See: http://goo.gl/u1DcQk
- For Windows, one popular option is WAMP: http://www.wampserver.com/en/

You will need to either download the d3.js library and put it in your local folder or link to the library online

- e.g. <script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>

USEFUL TOOLS FOR EDITING

Javascript is interpreted by the browser. Any standard text editor can be used.

Some popular options with code-highlighting:

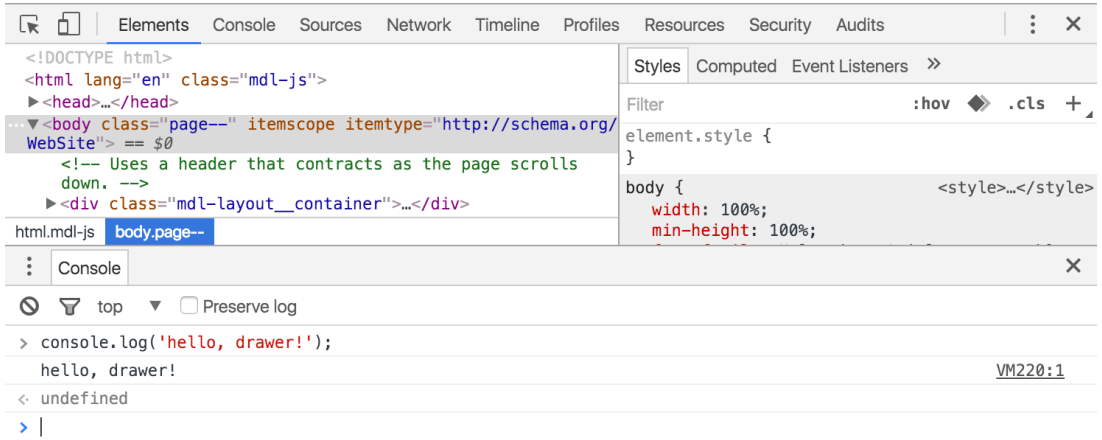
- Notepad++
- Atom (http://atom.io)

Online editors:

- Jsfiddle: http://jsfiddle.net/tommy351/P4Z75
- Observable: https://beta.observablehq.com/playground

Debugging usually through browser

- on Chrome/Firefox, use keyboard shortcut Ctrl Shift J (on Windows) or Ctrl Option J (on Mac).



- On Safari: use the Develop > Show Error **Console** menu item (Option-Cmd-C): If you don't see the Develop menu, you'll need to turn it on using **Safari** Preferences. **Open Safari** > Preferences, and click on the Advanced Tab.

GETTING STARTED

Choose a Text Editor

Write some basic code (see tutorial below)

Execute

- Upload to a webfolder OR Use local webserver (see previous slide)
- Run in a browser

Work through documentation/tutorials

- Introduction videos by Simon Raper provide a nice overview before starting to code: http://www.coppelia.io/2016/01/from-zero-to-d3/
- Then the following is a minimalist D3 Tutorial with lots of examples ready to try: https://www.tutorialspoint.com/d3js/d3js_data_join.htm

unnamed soundsculpture – embodiment of sound. onformative, 2012 [URL]
built with processing

INTRODUCTION TO PROCESSING

PROCESSING

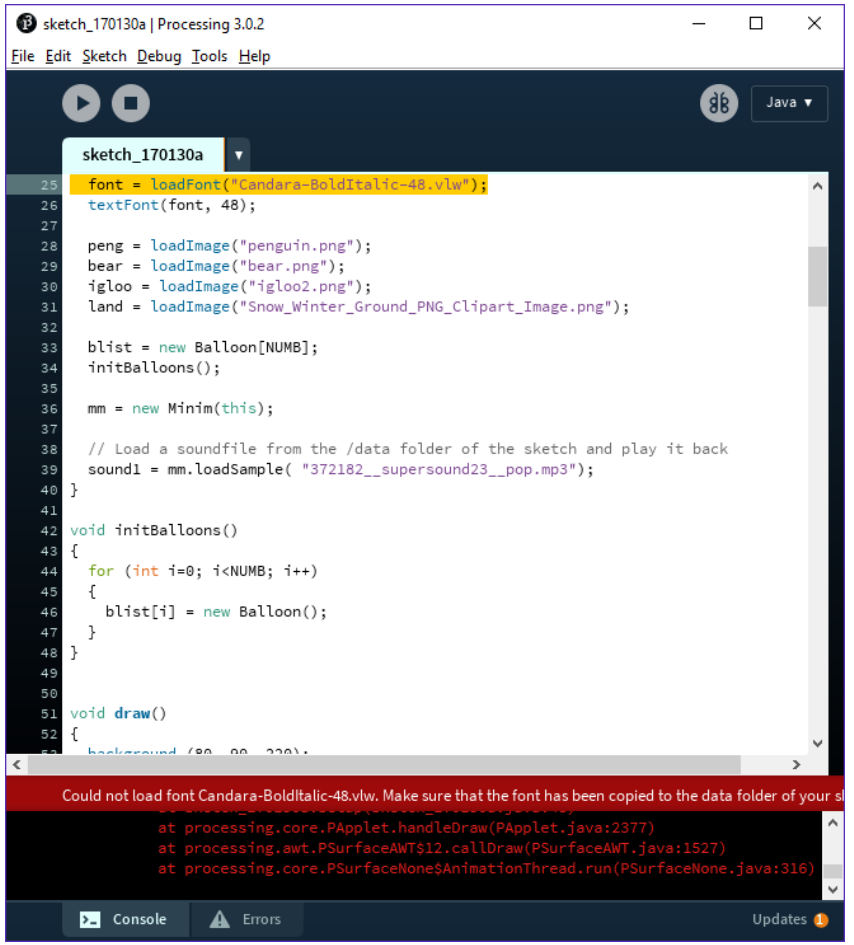
A free language and environment for writing visual programs with images, animation and interaction.

- Uses a relatively simple, easy to learn, syntax (similar to javascript/java)
- Can create programs that range from very quick and simple to highly complex.

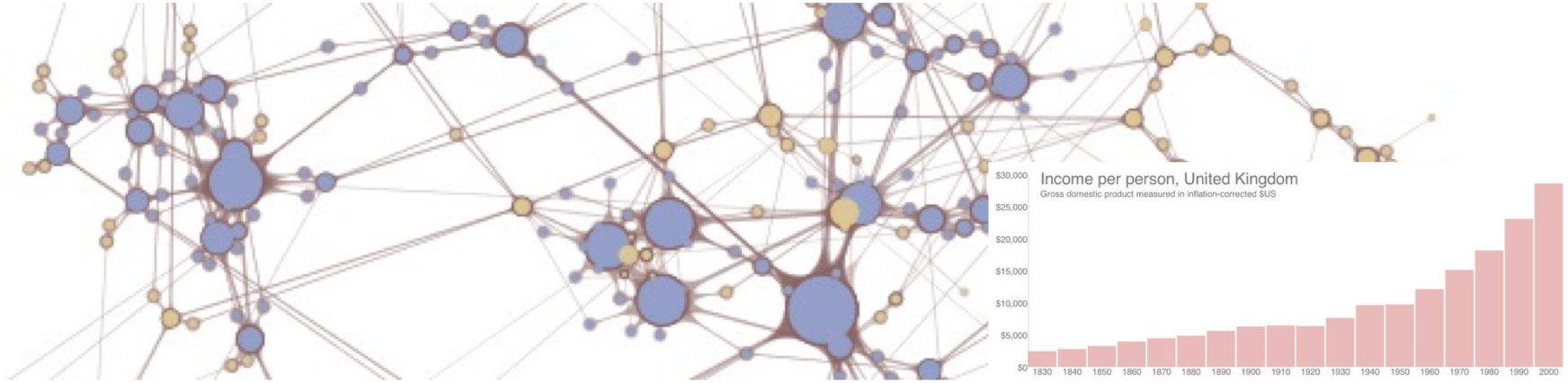
Available on multiple platforms with export options to windows/mac/~nix executable

- Website: http://www.processing.org
- Reference: http://www.processing.org/reference/index.html

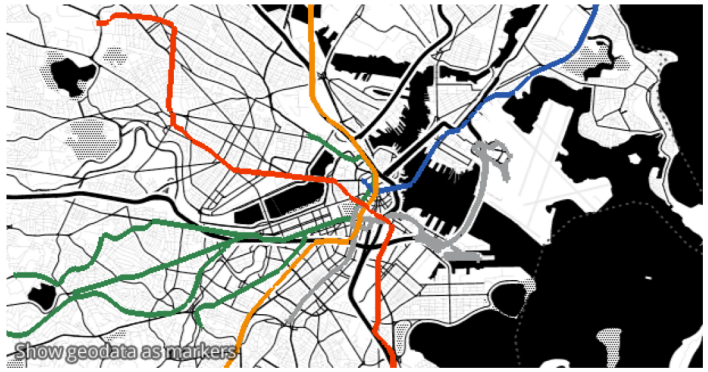
Come in different modes including: javascript, android, Python

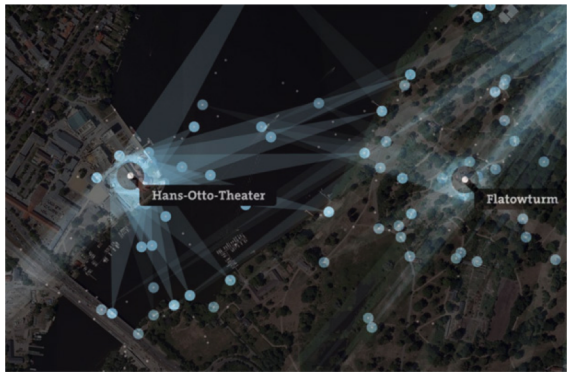


VISUALIZATION LIBRARIES IN PROCESSING



https://www.gicentre.net/utils





http://unfoldingmaps.org/

PROCESSING

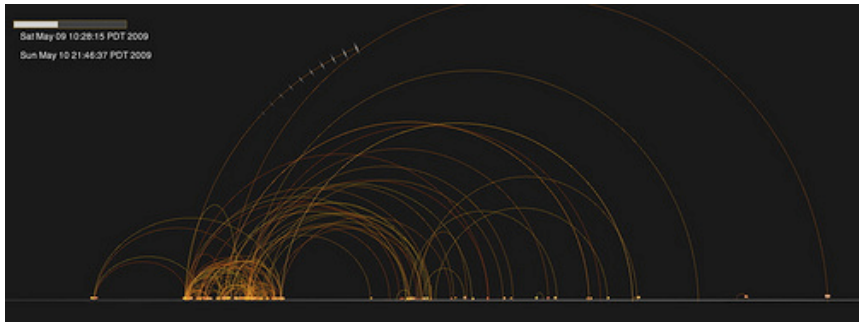


Image from Just Landed [URL] by Jer Thorp, 2009

Outline of Capabilities:

- Load, manipulate and display images
- Draw objects in 2D (and 3D) using points, lines and geometric objects
- Animation and User Input

Programs can be deployed as javascript web pages OR as standalone executable programs in multiple platforms

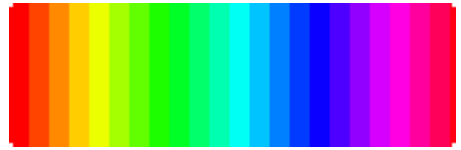
Rich collection of user contributed libraries for processing various interactive digital media including 3D models, sound, video, computer vision, computer graphics etc.

Extensions to Android, Raspberry Pi, Arduino

BASIC SYNTAX

```
char k;
int I;
float d;
color c = color ( 255, 0, 125);
```

- Built in functions **red()**, **green()**, and **blue()** can be used to extract individual values from a **color** object ; e.g. **float r = red (c);**
- Also supports HSB color mode



Arrays:

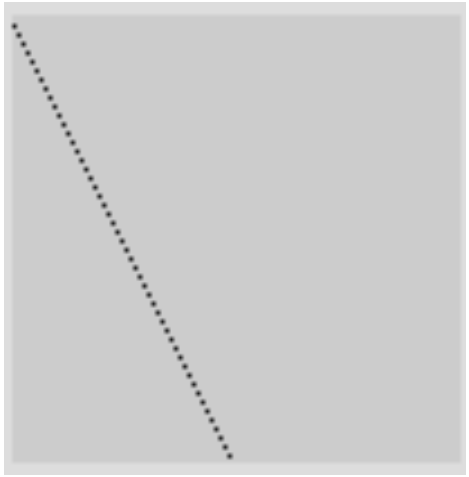
```
int[] numbersa = new int[3];
int[] numbersb = { 90, 150, 30 };
int b = numbersb[0] + numbersb[1];
```

SOME BASIC COMMANDS

```
size ( 100, 100 ) ;                 //Set window size to 100 x 100 pixels

background (255, 255, 255);         //Set background colour to White

fill (255, 0, 0);                   //Set fill colour to Red

stroke (0, 255, 0);                 //Set line colour to Green

point (205, 90)                     //Draw a point at {205, 90}

line (20, 20,    50, 50);           //Draw a line from {20, 20} to {50, 50}
```

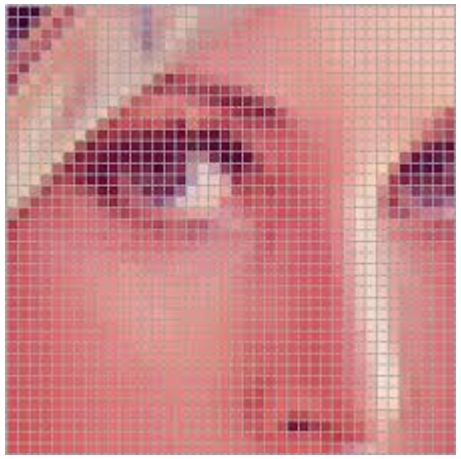
BASIC SYNTAX

```
for (int x=0; x<100; x=x+1)

{
 y = y +2;
 point (x, y);

}
```



built in class PImage to deal with images

```
PImage img;
img = LoadImage ( “picture.gif”);
image (img, 0, 0);
```



VECTOR DATA

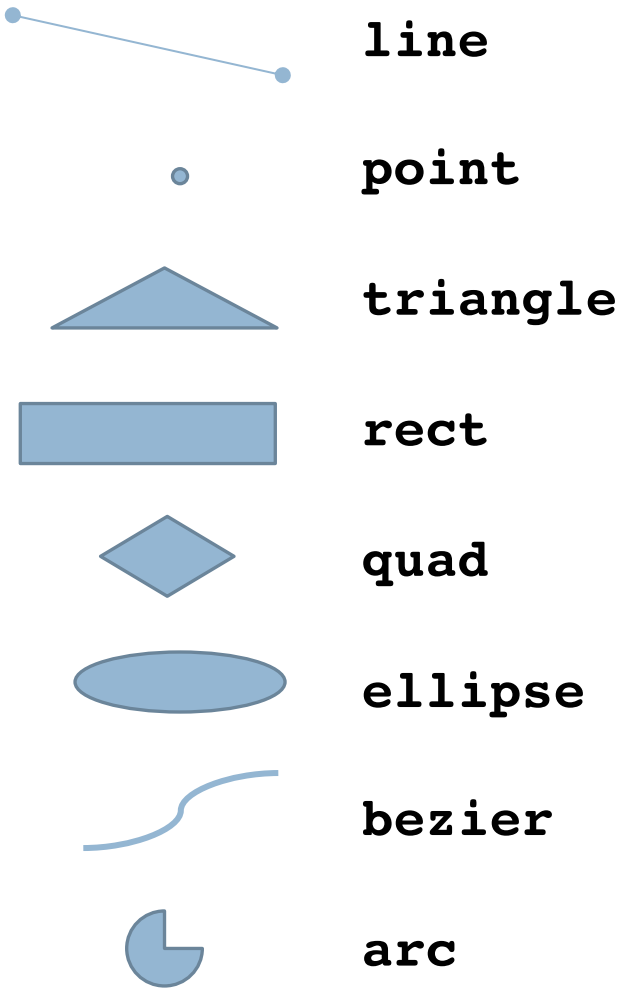
Creation of generalized vector objects in 2D and 3D using a number of commonly used *graphical primitives*

beginShape(...)

- POINTS, LINES, QUADS, TRIANGLES, TRIANGLE_STRIP, TRIANGLE_FAN, QUAD_STRIP

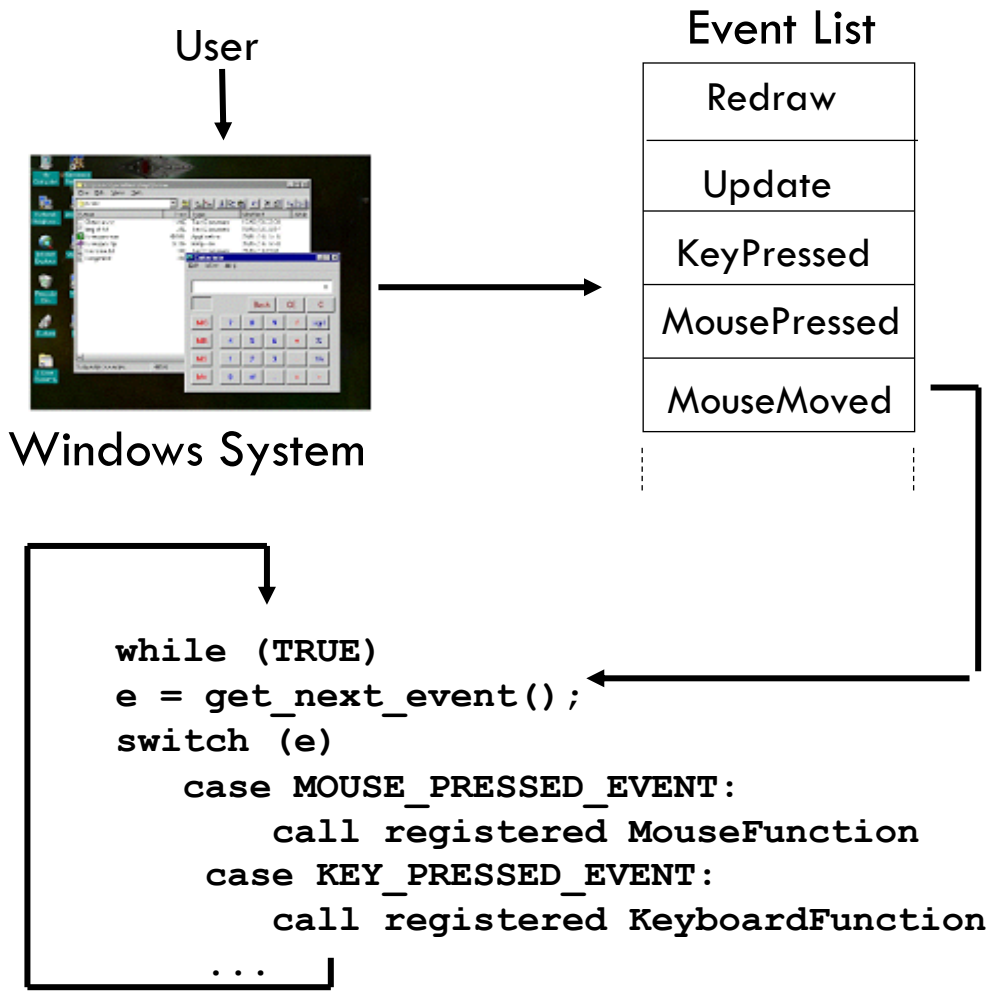
endShape()

vertex()



EVENT DRIVEN PARADIGM

```
void setup()
{
   size(800, 800);
}
void draw()
{
  rect(10, 10, 780, 780);
}
void keyPressed()
{
  if (key=='b')
    background(255, 0, 0);
    redraw();
}
void mouseMoved()
{
   ellipse(mouseX, mouseY, 10, 10);
}
```



PROCESSING LIBRARIES

HTTPS://PROCESSING.ORG/REFERENCE/LIBRARIES/

Official Libraries

- PDF Export
- Network
- Serial
- DXF Export
- Video
- Sound
- Hardware I/O

Third Party Libraries

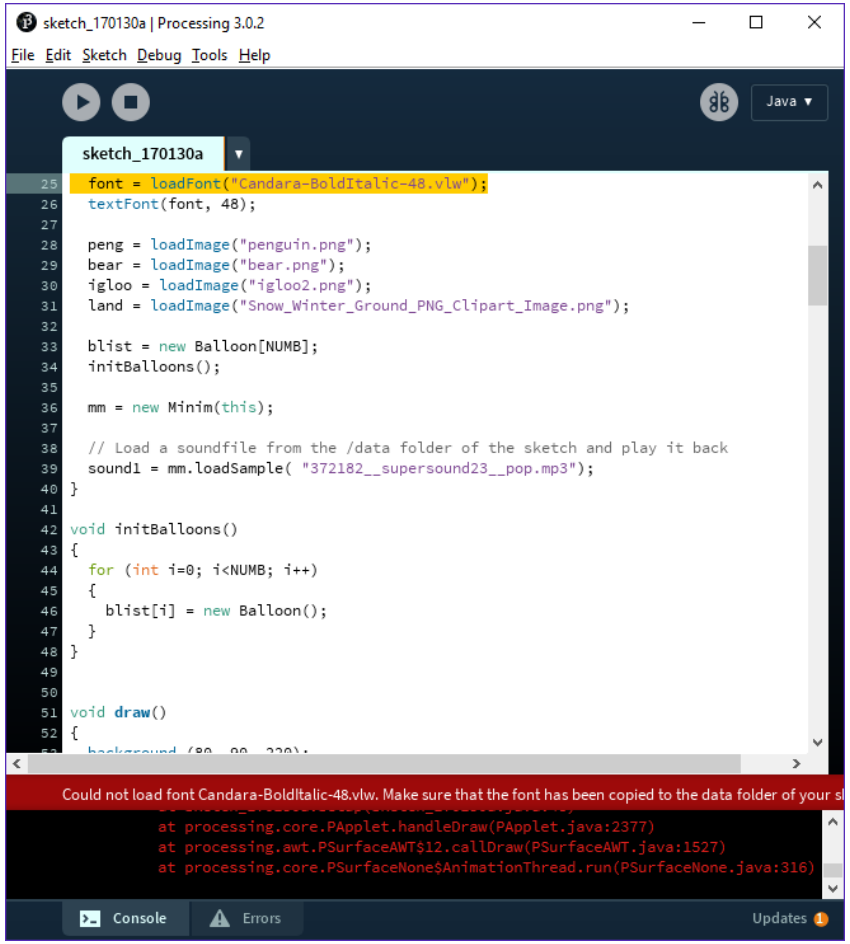
- 3D object loading, editing
- Animation tools
- GUI (Graphical User Interface)
- Hardware e.g. Arduino, Eyetracking, Motion sensor, Leap
- Math functions
- Simulation: advanced physics simulation
- Sound: loading, generation, processing
- Video processing and computer vision
- Typography
- Map visualization : unfolding
- Some Charts : giCentre

GETTING STARTED

Get the program from:

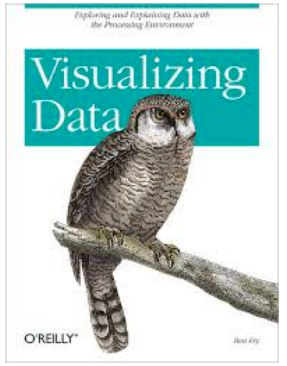
- http://www.processing.org/download/
- includes windows, mac and linux versions

Look at some examples/tutorials from the following slide



REFERENCES

Visualizing Data: Exploring and Explaining Data with the Processing Environment 1st Edition by Ben Fry



Make: Getting Started with Processing, Second Edition
Casey Reas and Ben Fry.

Tutorials: https://processing.org/tutorials/

Reference manual online:
http://www.processing.org/reference/