DSCI 554 LECTURE 3

DESIGN SPACE OF VISUALIZATIONS, GRAPHING IN THE BROWSER, INTRODUCTION TO D3 AND VEGA

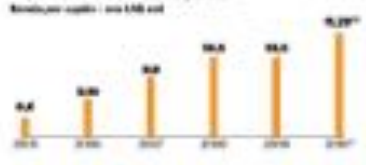
Dr. Luciano Nocera

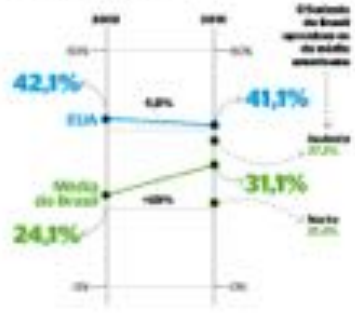
OUTLINE

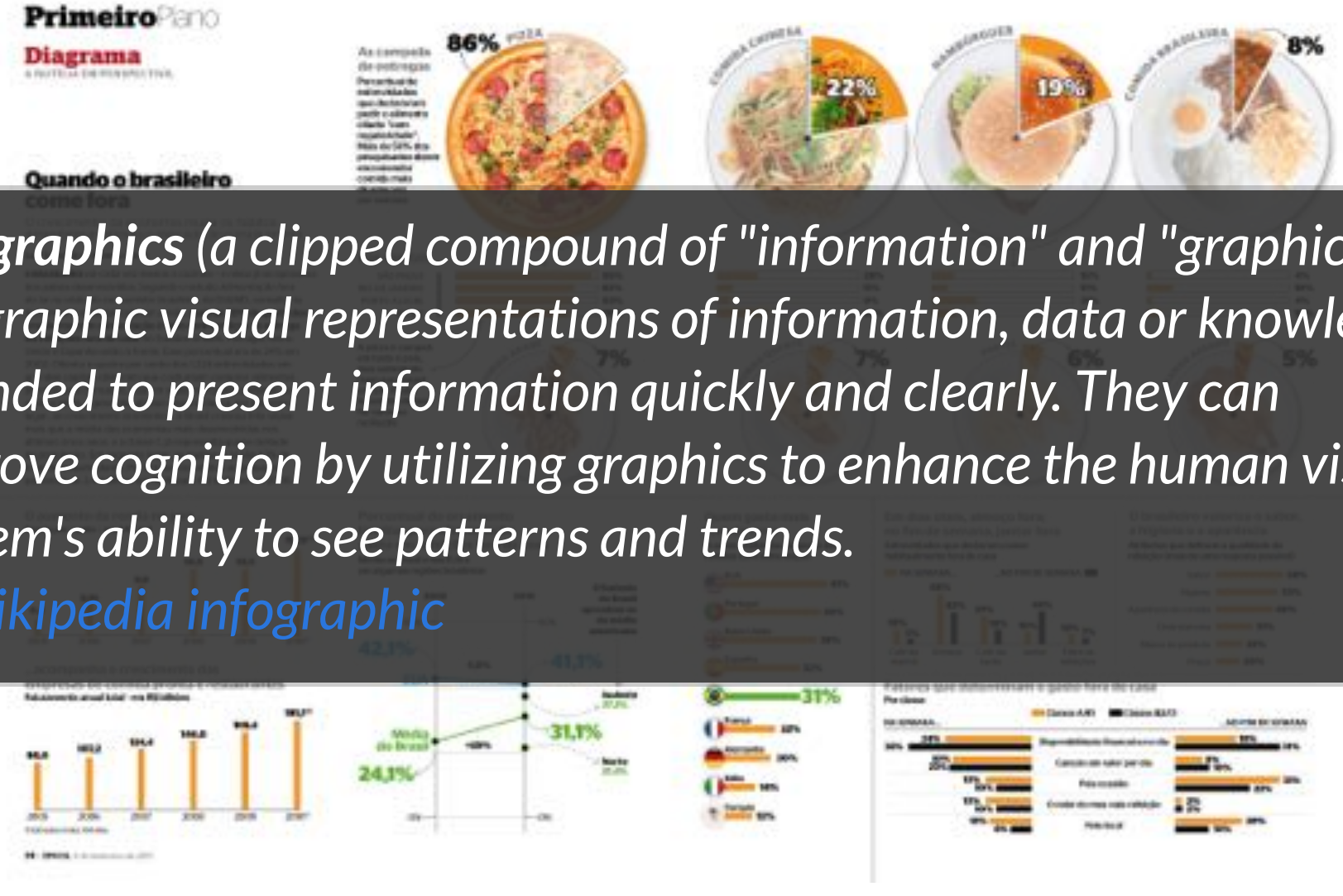
- Design space and design trade-offs
- Graphing in the browser
- Introduction to D3
- Introduction to Vega and Vega-lite

Diagrama

Quando o brasileiro come fora



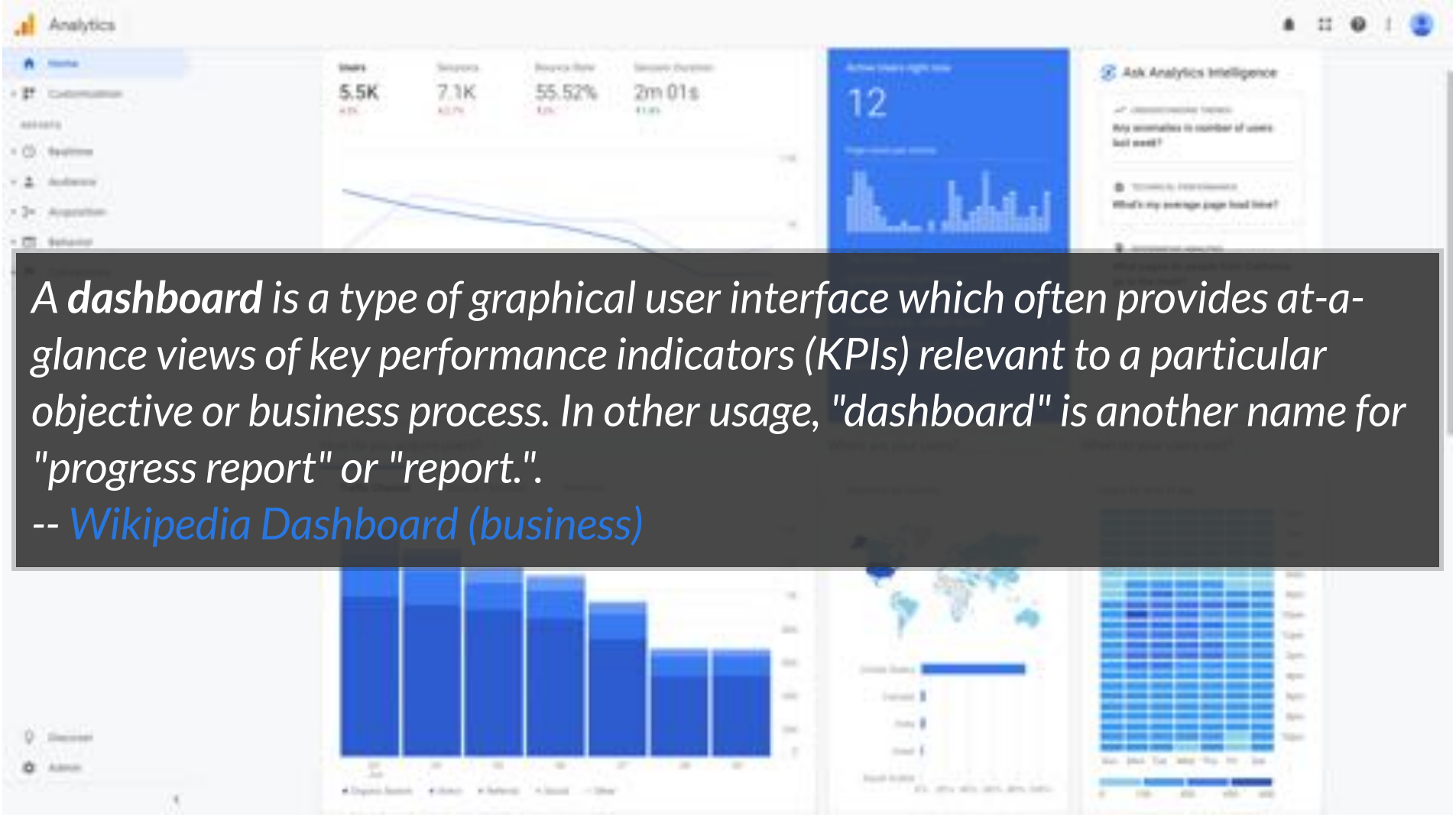




Infographics *(a clipped compound of "information" and "graphics") are graphic visual representations of information, data or knowledge intended to present information quickly and clearly. They can improve cognition by utilizing graphics to enhance the human visual system's ability to see patterns and trends.*

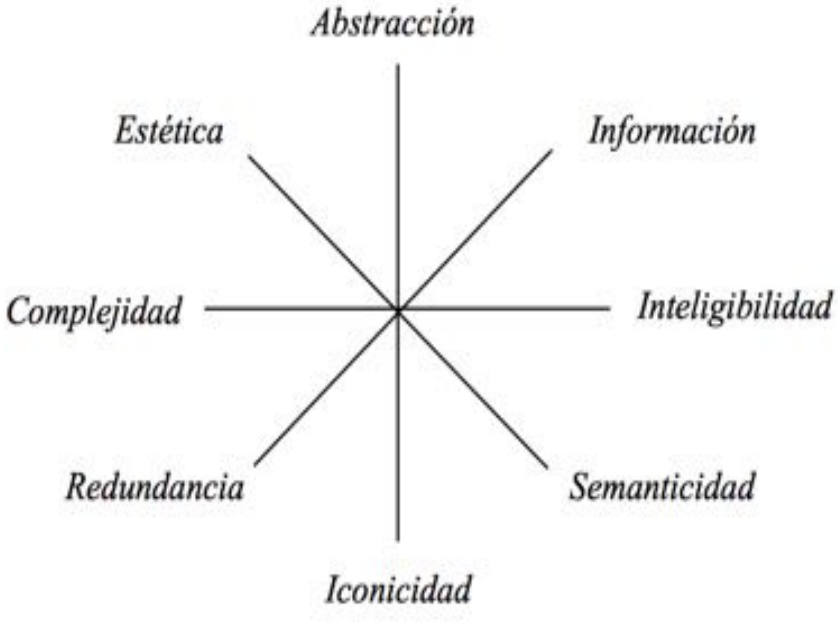
-- Wikipedia infographic


Analytics
Users
5.5K
Sessions
7.1K
Bounce Rate
55.52%
Session Duration
2m 01s
12
Ask Analytics Intelligence
How do you acquire users?
Where are your users?
When do your users visit?



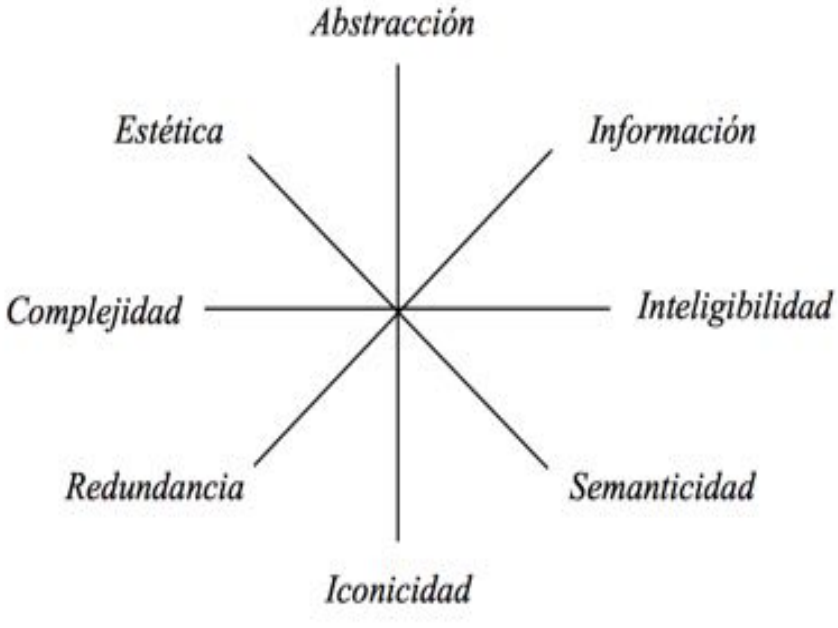
*A **dashboard** is a type of graphical user interface which often provides at-a-glance views of key performance indicators (KPIs) relevant to a particular objective or business process. In other usage, "dashboard" is another name for "progress report" or "report.".*
-- Wikipedia Dashboard (business)

Design Space [Costa 1998]



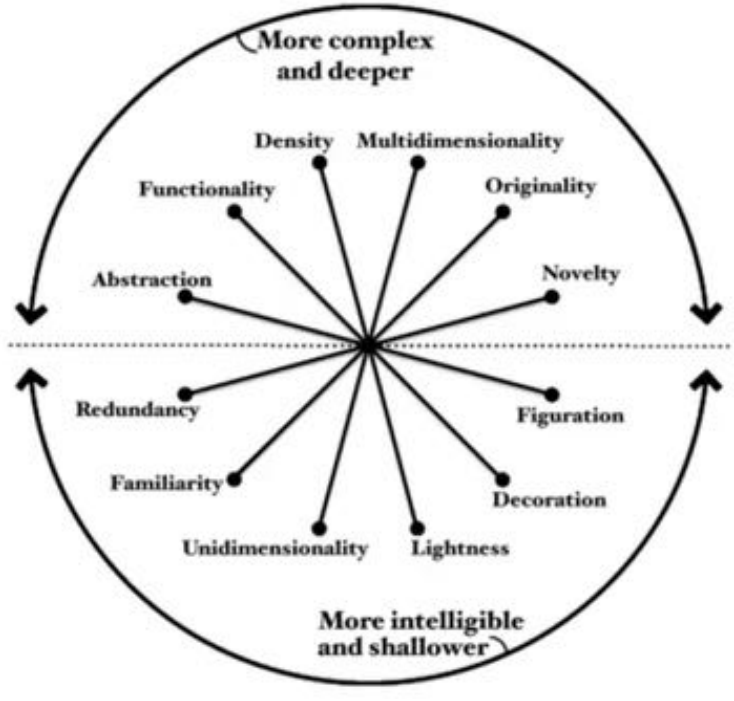
La esquemática: visualizar la información, Joan Costa Solà-Segalés, 1998.

Design Space [Costa 1998]



La esquemática: visualizar la información, Joan Costa Solà-Segalés, 1998.

Visualization Wheel [Cairo 2012]



Cairo, Alberto. The Functional Art: An introduction to information graphics and visualization. 2012.

Less complex → More complex

FIGURATION-ABSTRACTION

Measures the distance from referent to the representation

DECORATION-FUNCTIONALITY

Measures the amount of informative content

LIGHTNESS-DENSITY

Measures the amount of content displayed in relation to space

UNIDIMENSIONALITY-MULTIDIMENSIONALITY

Measures the number of layers and forms used to encode the data

FAMILIARITY-ORIGINALITY

Measures how challenging the forms are for the user to understand

REDUNDANCY-NOVELTY

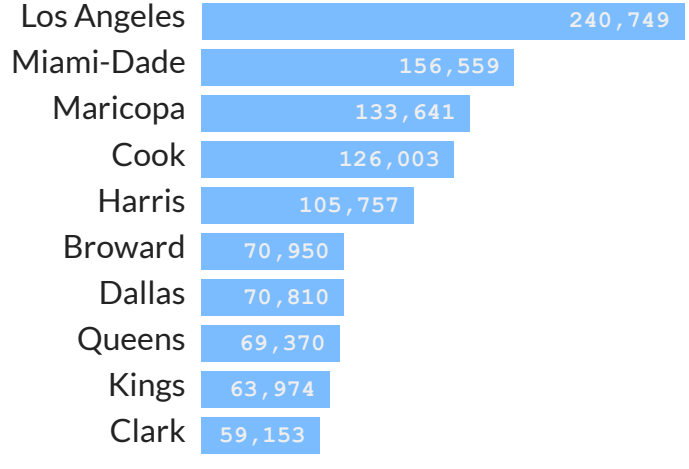
Measures the number of times things are explained

OUTLINE

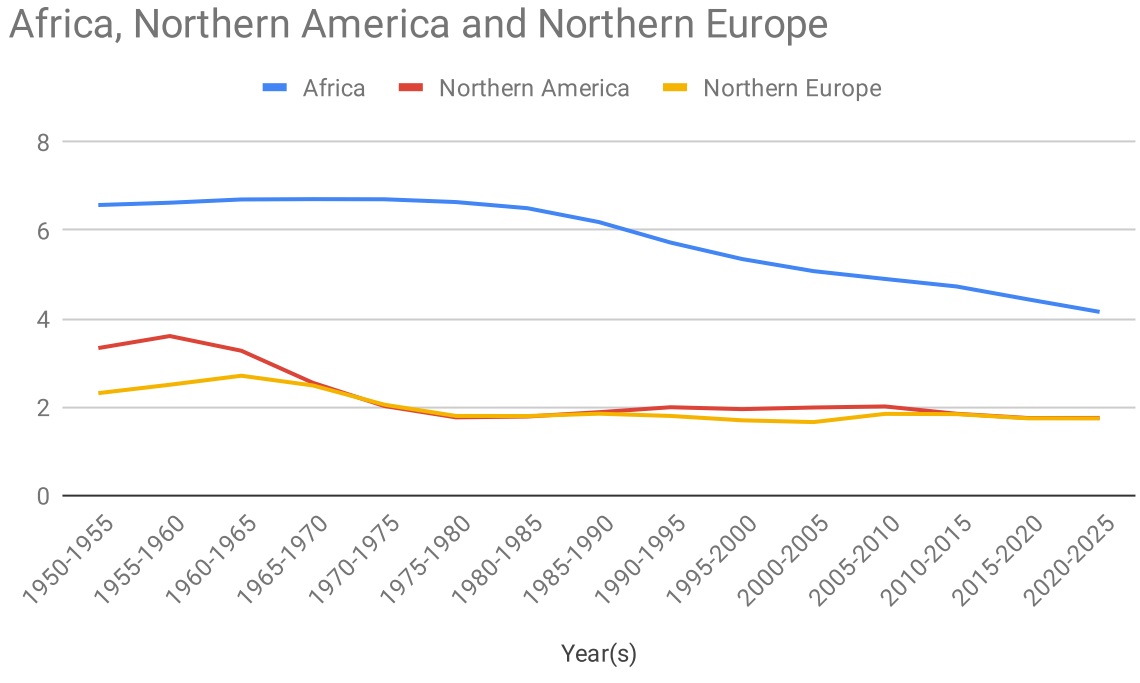
- Design space and design trade-offs
- Graphing in the browser
- Introduction to D3
- Introduction to Vega and Vega-lite

BAR CHART IN PLAIN HTML

Top 10 COVID-19 confirmed in US, Aug 31 2020 (source Johns Hopkins University)

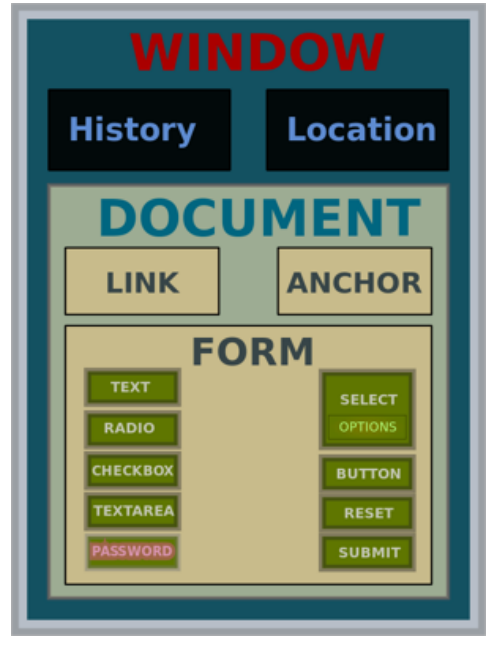


LINE CHART IN SVG



DOCUMENT OBJECT MODEL (DOM)

- Hierarchical box model (conceptual data model)
- Used in browsers for pages (HTML documents)
- Implemented as *"Javascript object"*
- In the DOM everything is implemented as a node:
  - The document is the *document node*
  - HTML elements are *element nodes*
  - HTML attributes are *attribute nodes*
  - Text inside HTML elements are *text nodes*
  - Comments are *comment nodes*

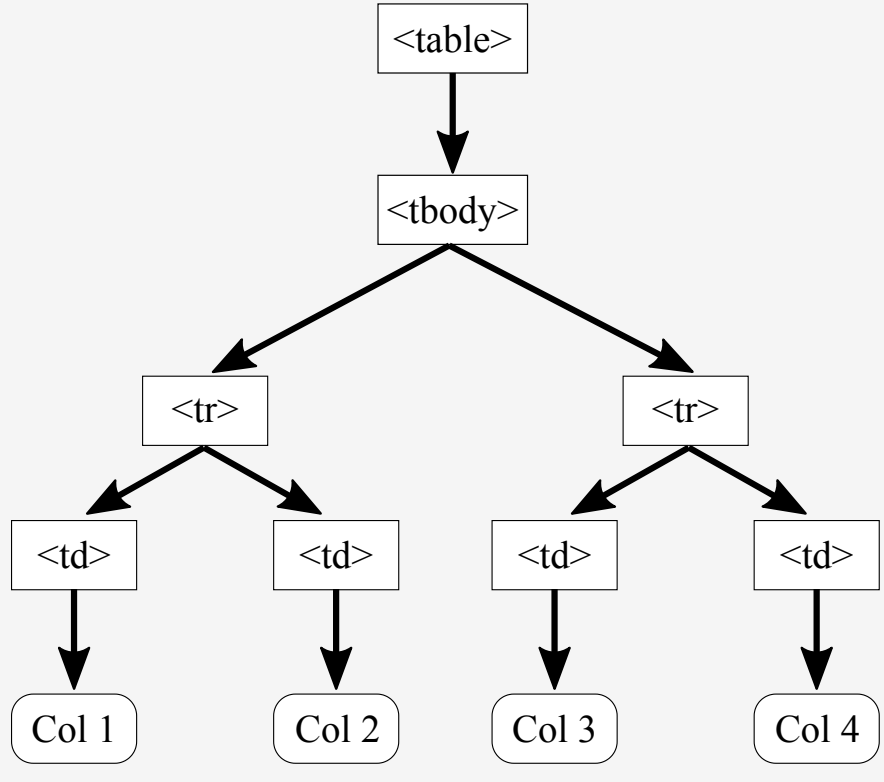


DOM EXAMPLE

HTML TABLE

```
<TABLE>
  <TBODY>
  <TR>
    <TD>Col 1</TD>
    <TD>Col 2</TD>
  </TR>
  <TR>
    <TD>Col 3</TD>
    <TD>Col 4</TD>
  </TR>
  </TBODY>
</TABLE>
```

DOM TREE OF HTML TABLE



CSS BOX MODEL

- Everything in CSS has a box around it
- There are 2 main types of boxes:
  - Inline: occupy the space bounded by the tag
  - Block: start on a new line and take up the full width
- The type of box is controlled by the CSS `display: [inline|block]` property
- You can override the defaults type as `block` or `inline`

Examples

```
span is <span>inline</span>
```

span is inline

```
div is <div>block</div>
```

div is
block

```
div can become <div style="display: inline">inline</div>
```

div can become inline

GRAPHING WITH HTML, SVG, CSS

GLOBAL VS. ELEMENT SPECIFIC ATTRIBUTES

GLOBAL ATTRIBUTES (`id`, `class`, `style`)

```
<!-- Use id to reference containers for dynamic charts -->
<div id="chart1"></div>
<svg id="chart2"></svg>

<style>
  div.bar { background-color: red; }
  circle.dot { fill: red; }
</style>

<!-- Use class to apply common styles -->
<div class="bar" style="width: 600px">bar 1</div>

<svg style="background-color: lightpink">
  <circle class="dot" cx="5" cy="5" r="2"/>
</svg>
```

ELEMENT SPECIFIC ATTRIBUTES

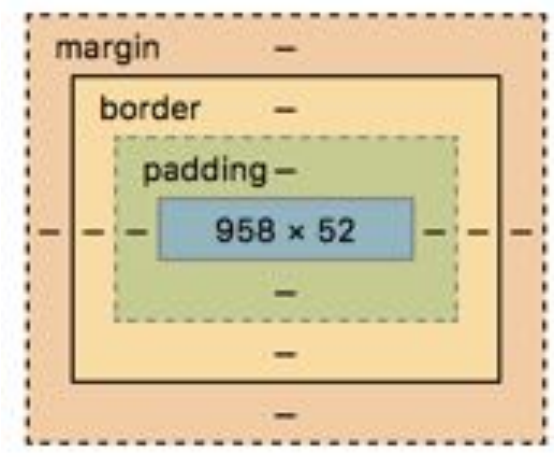
```
<!-- Use element specific attributes to place and size -->
<img src="pict.png" width="20px"\>

<svg style="background-color: lightpink">
  <circle cx="5" cy="5" r="2"/>
</svg>
```

`style` FOR COLORING & BOX SIZING

Control color in HTML tags

```
background-color: brown; /* background color of an element */
color: white;  /* color of text within element */
```

Control box model parameters

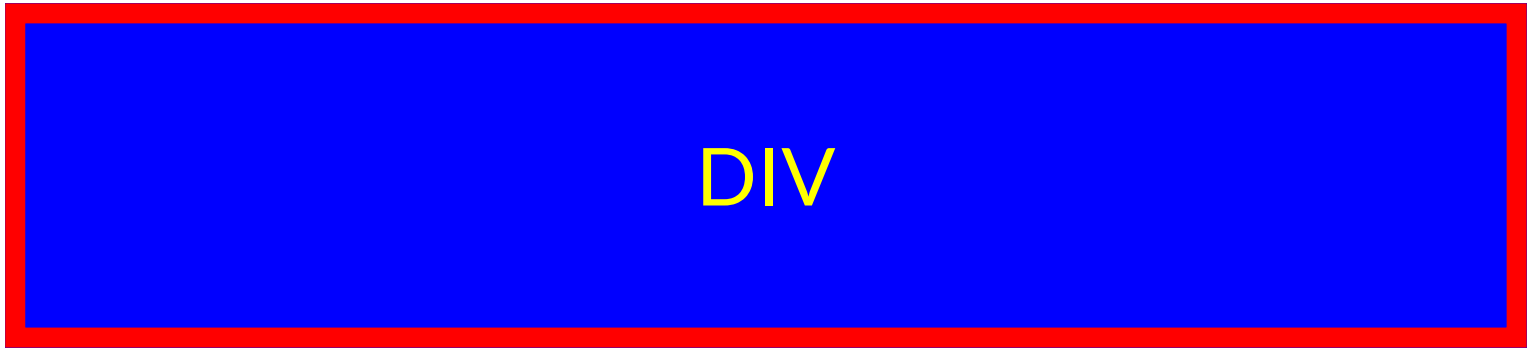


Margins: extra space around an element.
Border: border around an element.
Padding: extra space within an element.

```
padding: 1em;  /* Apply to all four sides */
padding: 5% 10%;  /* vertical | horizontal */
padding: 1em 2em 2em;  /* top | horizontal | bottom */
padding: 5px 1em 0 2em;  /* top | right | bottom | left */

border: 1px solid red;  /* width | style | color */

margin: 1em;  /* Apply to all four sides */
margin: 5% auto;  /* vertical | horizontal */
margin: 1em auto 2em;  /* top | horizontal | bottom */
margin: 2px 1em 0 auto;  /* top | right | bottom | left */
```

`style` FOR COLORING & BOX SIZING (EXAMPLE)

before



after

```
before
<div style="color: yellow; background-color:blue; margin: 100px 100px; padding: 50px 50px; border: 10px solid red; ">DIV</div>
after
```

`style` FOR SIZING

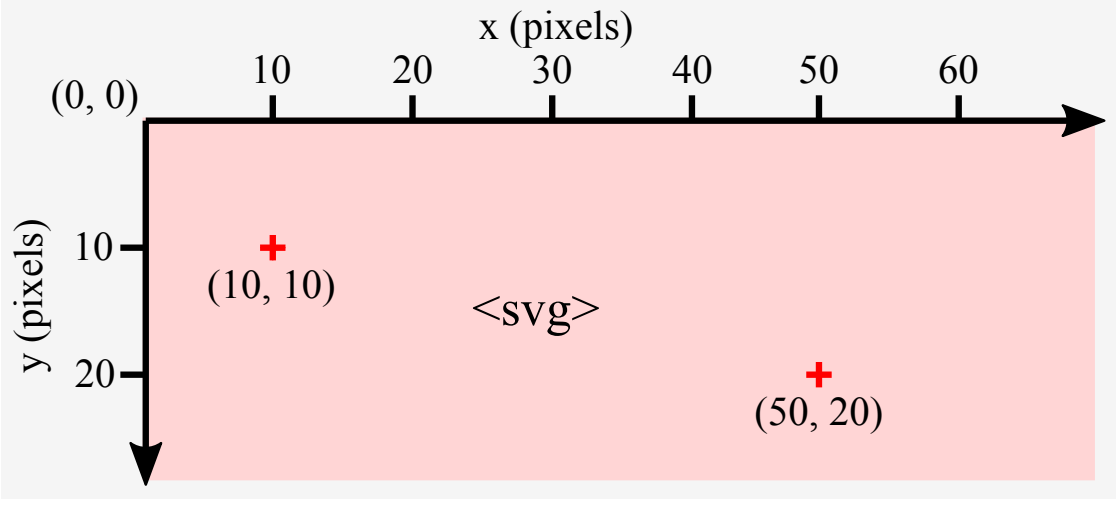
```
<div style="background-color: red; margin-bottom: 10px;">bar1</div>
<div style="background-color: red; margin-bottom: 10px; width: 200px; height: 20px;">bar2</div>
<div style="background-color: red; width: 100px; height: 20px;">bar3</div>
```

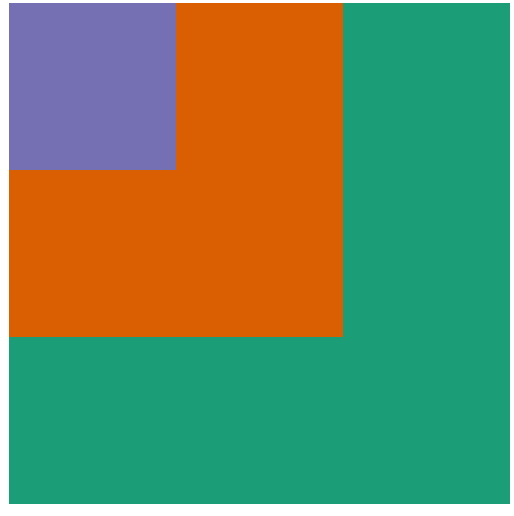
CSS INHERITANCE AND CSS CLASSES

In CSS properties are inherited by descendants.
Style the parent not to repeat properties in descendants
Define a class as a short hand for common descendants properties

```
<style>
  .redbar {
    background-color: red;
    height: 50px;
    margin-bottom: 10px;
  }
</style>

<div style="height: 20px; font-size: 0.5em; font-weight: bolder;">
  <div class="redbar">bar1</div>
  <div class="redbar" style="width: 200px;">bar2</div>
  <div class="redbar" style="width: 100px;">bar3</div>
</div>
```

SVG





- SVG elements are drawn in the <svg>
- Coordinate system in pixels starting up left corner of <svg>
- Painter's algorithm defines drawing order:
  coding order is drawing order (last drawn on top)

HTML VS. SVG FOR CHARTING

	HTML	SVG
Placement & sizing	CSS box model relative/absolute placement	Cartesian coordinate system and viewport
Primitives	CSS box model boxes	Named shapes, curves
Text	inner HTML	<text>
Drawing order	CSS box model / DOM order	Painter's algorithm coding order → depth order
Hierarchy	CSS box model	Nesting using <g>

simpler for graphing

SVG BASIC SHAPES

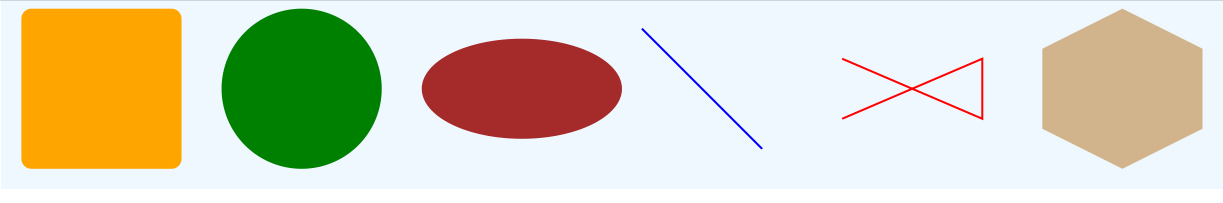
- <rect>: rectangle (w rounded corners)
- <circle>: circle
- <ellipse>: ellipse
- <line>: line
- <polyline>: polyline
- <polygon>: polygon

W3C Scalable Vector Graphics (SVG) 1.1 (Second Edition)

SVG BASIC SHAPES ATTRIBUTES

- Positions & size
- Fill (interior color)
- Stroke (border color)

```
<rect x="10" y="10" width="80" height="80" rx="5" ry="5" fill="orange"/>

<circle cx="150" cy="50" r="40" fill="green"/>

<ellipse cx="260" cy="50" rx="50" ry="25" fill="brown"/>

<line x1="320" y1="20" x2="380" y2="80" stroke="blue"/>

<polyline points="420,35 490,65 490,35 420,65" stroke="red" fill="none"/> <!-- open -->

<polygon points="560,10 600,30 600,70 560,90 520,70 520,30" fill="tan"/> <!-- closed -->
```

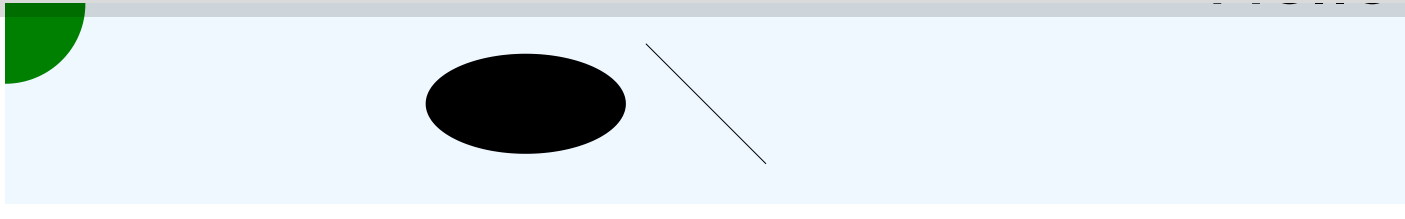


W3C Scalable Vector Graphics (SVG) 1.1 (Second Edition)

SVG VISIBILITY & ATTRIBUTES DEFAULTS

- Positions & size (e.g., x, y, x1, y1, x2, y2, cx, cy, width, height, r, rx, ry) defaults to *"0"*
- Fill defaults to *"black"* for shapes and *"none"* for text
- Stroke defaults to *"none"*

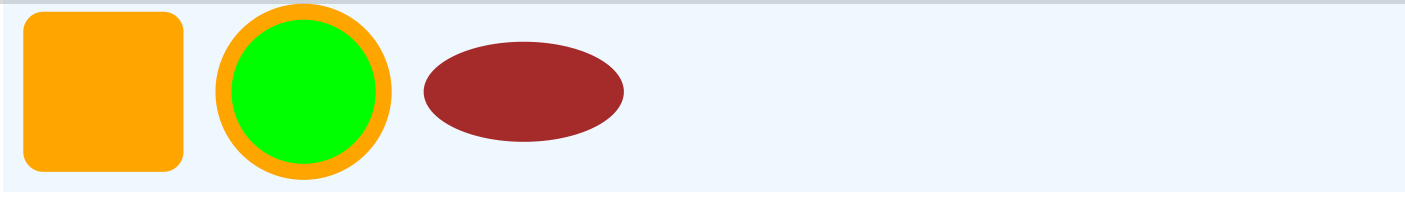
```
<rect x="10" y="10" idth="80" height="80" rx="5" ry="5" fill="orange"/> <!-- width="0" -->

<circle x="150" y="50" r="40" fill="green"/> <!-- cx="0" cy="0"-->

<ellipse cx="260" cy="50" rx="50" ry="25" background-color="brown"/> <!-- fill="black" -->

<line x1="320" y1="20" x2="380" y2="80" color="red"/> <!-- stroke="none" -->

<polyline points="420,35 490,65 490,35 420,65" stroke="myred" fill="none"/> <!-- stroke="none" -->

<polygon oints="560,10 600,30 600,70 560,90 520,70 520,30" fill="tan"/> <!-- points="" -->

<text x="600">Hello</text> <!-- text element requires a y, here y="none" -->
```



SVG STYLING PROPERTIES

MOST BUT NOT ALL SVG attributes have CSS styling properties

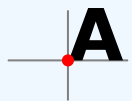
```
<svg>
  <!-- these work -->

  <rect style="x: 10; y: 10; width: 80; height: 80; rx: 5; ry: 5; fill: orange;"/> <!-- rx, ry now works! -->

  <circle style="cx: 150; cy: 50; r: 40; fill: lime; stroke:orange; stroke-width: 8px;"/>

  <ellipse style="cx: 260; cy: 50; rx: 50; ry: 25; fill: brown;"/> <!-- rx, ry now works! -->

  <!-- these don't work! -->

  <line style="x1: 320; y1: 20; x2: 380; y2: 80; stroke: blue;"/> <!-- x1, x2, y1, y2 = 0 -->

  <polyline style="points: 420,35 490,65 490,35 420,65; stroke: red; fill: none;"/> <!-- points = '' -->

  <polygon style="points: 560,10 600,30 600,70 560,90 520,70 520,30; fill: tan;"/> <!-- points = '' -->
</svg>
```



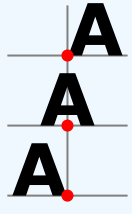
POSITIONING SVG TEXT

```
<svg>
  <text x="0" y="70">Text0<text/>
</svg>
```

`x, y` define where the text is placed in the SVG (by default the text bottom left is located at `x, y`)



text-anchor controls the horizontal placement of the text relative to `x, y`



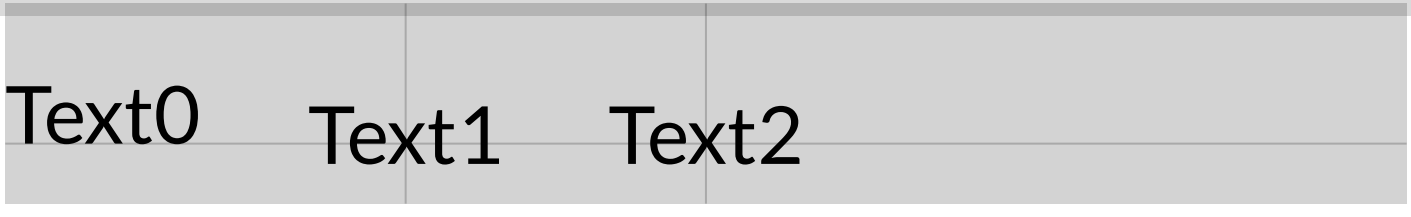

```
text-anchor: start ← default
text-anchor: middle
text-anchor: end
```

alignment-baseline controls the vertical placement of the text relative to `x, y`


```
alignment-baseline: hanging
alignment-baseline: middle
alignment-baseline: unset ← default
```

CENTERING SVG TEXT

```
<svg>
  <text x="0" y="70">Text0<text/>
  <text x="200" y="70" text-anchor="middle" alignment-baseline="middle">Text1<text/>
  <text x="350" y="70" style="text-anchor: middle; alignment-baseline: middle;">Text2<text/>
</svg>
```



What will appear on the page?

```
<svg>
<polygon points="60,10 100,30 100,70 60,90 20,70 20,30" fill="tan">text</polygon>
</svg>
```

A. Tan polygon
B. Tan polygon and text
C. Only text
D. Nothing

What will appear on the page?

```
<svg>
<polygon points="60,10 100,30 100,70 60,90 20,70 20,30" fill="tan">text</polygon>
</svg>
```

A. Tan polygon
B. Tan polygon and text
C. Only text
D. Nothing

What will appear on the page?

```
<svg>
  <circle cx="100" cy="100" fill="red"/>
  <text>Hello</text>
</svg>
```

A. Red circle
B. Red circle and text
C. Text
D. Nothing

What will appear on the page?

```
<svg>
  <circle cx="100" cy="100" fill="red"/>
  <text>Hello</text>
</svg>
```

A. Red circle

B. Red circle and text

C. Text

D. Nothing

What choice corresponds to the figure shown?

```
<svg style="background-color: lightgrey">
  <rect A="100" B="0" C="50" D="200" fill="red"/>
</svg>
```

A. A = x1, B = y1, C = x2, D = y2

B. A = y1, B = x1, C = y2, D = x2

C. A = x, B = y, C = width, D = height

D. A = y, B = x, C = height, D = width

OUTLINE

- Design space and design trade-offs
- Graphing in the browser
- Introduction to D3
- Introduction to Vega and Vega-lite

D3

TL;DR. *JavaScript library to bind data to the DOM, i.e., vocabulary of graphical marks come directly from web standards: HTML, SVG, and CSS.*

D3

TL;DR. *JavaScript library to bind data to the DOM, i.e., vocabulary of graphical marks come directly from web standards: HTML, SVG, and CSS.*

To generate a graphic:

1. *Get the data (properly formatted as an array!)*
2. *Use D3 to map the data to HTML elements*
3. *Leave-it to the browser to do the rest!*

D3

TL;DR. *JavaScript library to bind data to the DOM, i.e., vocabulary of graphical marks come directly from web standards: HTML, SVG, and CSS.*

To generate a graphic:

1. *Get the data (properly formatted as an array!)*
2. *Use D3 to map the data to HTML elements*
3. *Leave-it to the browser to do the rest!*

Basic mechanisms to map data to HTML elements:

1. *Select & append*
2. *Data join*

PROGRAMMING PARADIGMS & VISUALIZATION TOOLS

Paradigm	**Imperative**	**Declarative**
Properties	○ Specify How to do ○ Lower-level ○ Less accessible	○ Specify What to do ○ Higher-level ○ More accessible
Examples	Google Charts Matplotlib WebGL	D3 ggplot2

```
// https://d3js.org Version 5.5.0. Copyright 2018 Mike Bostock.
(function(t,n){"object"==typeof exports&&"undefined"!=typeof module?n(exports):"function"==typeof
define&&define.amd?define(["exports"],n):n(t.d3=t.d3||{})})(this,function(t){"use strict";function
n(t,n){return t<n?-1:t>n?1:t>=n?0:NaN}function e(t){return 1===t.length&&(t=function(t){return
function(e,r){return n(t(e),r)}}(t)),{left:function(n,e,r,i){for(null==r&&(r=0),null==i&&
(i=n.length);r<i;){var o=r+i>>>1;t(n[o],e)<0?r=o+1:i=o}return r},right:function(n,e,r,i)
{for(null==r&&(r=0),null==i&&(i=n.length);r<i;){var o=r+i>>>1;t(n[o],e)>0?i=o:r=o+1}return
r}}}function r(t,n){return[t,n]}function i(t){return null===t?NaN:+t}function o(t,n){var
e,r,o=t.length,a=0,u=-1,f=0,c=0;if(null==n)for(;++u<o;)isNaN(e=i(t[u]))||(c+=(r=e-f)*(e-
(f+=r/++a)));else for(;++u<o;)isNaN(e=i(n(t[u],u,t)))||(c+=(r=e-f)*(e-(f+=r/++a)));if(a>1)return
c/(a-1)}function a(t,n){var e=o(t,n);return e?Math.sqrt(e):e}function u(t,n){var
e,r,i,o=t.length,a=-1;if(null==n){for(;++a<o;)if(null!=(e=t[a])&&e>=e)for(r=i=e;++a<o;)null!=
(e=t[a])&&(r>e&&(r=e),i<e&&(i=e))}else for(;++a<o;)if(null!=
(e=n(t[a],a,t))&&e>=e)for(r=i=e;++a<o;)null!=(e=n(t[a],a,t))&&(r>e&&(r=e),i<e&&
(i=e));return[r,i]}function f(t){return function(){return t}}function c(t){return t}function
s(t,n,e){t=+t,n=+n,e=(i=arguments.length)<2?(n=t,t=0,1):i<3?1:+e;for(var
r=-1,i=0|Math.max(0,Math.ceil((n-t)/e)),o=new Array(i);++r<i;)o[r]=t+r*e;return o}function
l(t,n,e){var r,i,o,a,u=-1;if(n=+n,t=+t,e=+e,t===n&&e>0)return[t];if((r=n<t)&&(i=t,t=n,n=i),0===
(a=h(t,n,e))|...
```

Immediately-Invoked Function Expression (IIFE) signature: `(function() {})();`

ABOUT D3

- Library based on modern Web standards
- Created and maintained by Mike Bostock
- Website: https://d3js.org
- Documentation: https://github.com/d3/d3/wiki
- Gallery: https://github.com/d3/d3/wiki/Gallery

- D3 stands for Data-Driven Documents
- Concentrates on the data as opposed to the representation
- High expressiveness: good for custom/novel forms

WHAT D3 DOES

1. Loads data in the browser (DOES NOT HIDE THE DATA!)
2. Binds data to document elements
3. Transforms elements by interpreting each element's bound datum and setting its visual properties
4. Transitions elements between states in response to user input

BASIC D3 OPERATIONS

1. Select elements
2. Add new elements to selected elements
3. Delete selected elements
4. Modify selected elements to position and style

BASIC D3 OPERATIONS ARE IMPLEMENTED WITH FUNCTION CHAINING

BASIC D3 OPERATIONS ARE IMPLEMENTED WITH FUNCTION CHAINING

- Function Chaining is a design pattern

BASIC D3 OPERATIONS ARE IMPLEMENTED WITH FUNCTION CHAINING

- Function Chaining is a design pattern
- Code writtent using Function Chaining is simpler to understand:

```
//without chaining
obj.method3(obj.method2(obj.method1()));

//same
var s = obj.method1();
s = s.method2();
s = s.method3();

//with chaining
obj.method1()
  .method2()
  .method3();
```

BASIC D3 OPERATIONS ARE IMPLEMENTED WITH FUNCTION CHAINING

- Function Chaining is a design pattern
- Code writtent using Function Chaining is simpler to understand:

```
//without chaining
obj.method3(obj.method2(obj.method1()));

//same
var s = obj.method1();
s = s.method2();
s = s.method3();

//with chaining
obj.method1()
   .method2()
   .method3();
```

- Function chaining is implemented using *"this"*:

```
var obj = {
  method1: function() {
    console.log('method1');
    return this;  //"this" refers to the current object instance
  },
  method2: function(a) {
    console.log('method2');
    return this;
  }
};

obj.method1().method2();
  method1
  method2
```

1. SELECT ELEMENTS

- A D3 selection is a list (or array) of nodes with a parent
- `select()` and `selectAll()` return a `selection`
- Take a CSS selector or a function as argument

```
var selection = d3.select(selector);  //select first matching element in the document
var selection = d3.selectAll(selector);  //select all matching elements in the document

//use chaining to select on the selection object
var selection = selection.select(selector);  //select first matching element in the selection
var selection = selection.selectAll(selector);  //select all matching elements in the selection
```

D3 selection documentation

1. SELECT ELEMENTS

- A D3 selection is a list (or array) of nodes with a parent
- `select()` and `selectAll()` return a `selection`
- Take a CSS selector or a function as argument

```
var selection = d3.select(selector);  //select first matching element in the document
var selection = d3.selectAll(selector);  //select all matching elements in the document

//use chaining to select on the selection object
var selection = selection.select(selector);  //select first matching element in the selection
var selection = selection.selectAll(selector);  //select all matching elements in the selection
```

```
var s = d3.select('body');  //selects <body> in <html>
s.select('p');  //selects first <p> in <body>

d3.select('body')  //same as above using chaining
  .select('p');

d3.selectAll('p');  //selects all <p> in parent
d3.select('#chart');  //selects first element in parent with id="chart"
d3.selectAll('.red');  //selects all elements in parent with class="red"
```

D3 selection documentation

2. ADD NEW ELEMENTS TO SELECTED ELEMENTS

- `selection.append(type)` appends a new element to the last child of each selected element
- `selection.insert(type[, before])` inserts a new element `before` the first element matching the specified before selector for each selected element
- `selection.append` and `selection.insert` return added elements

Example 1

```
<body></body>
```

```
d3.select('body')
  .append('p')  //append p as the last child of body
  .text('Text0');  //set text to "Text0"
```

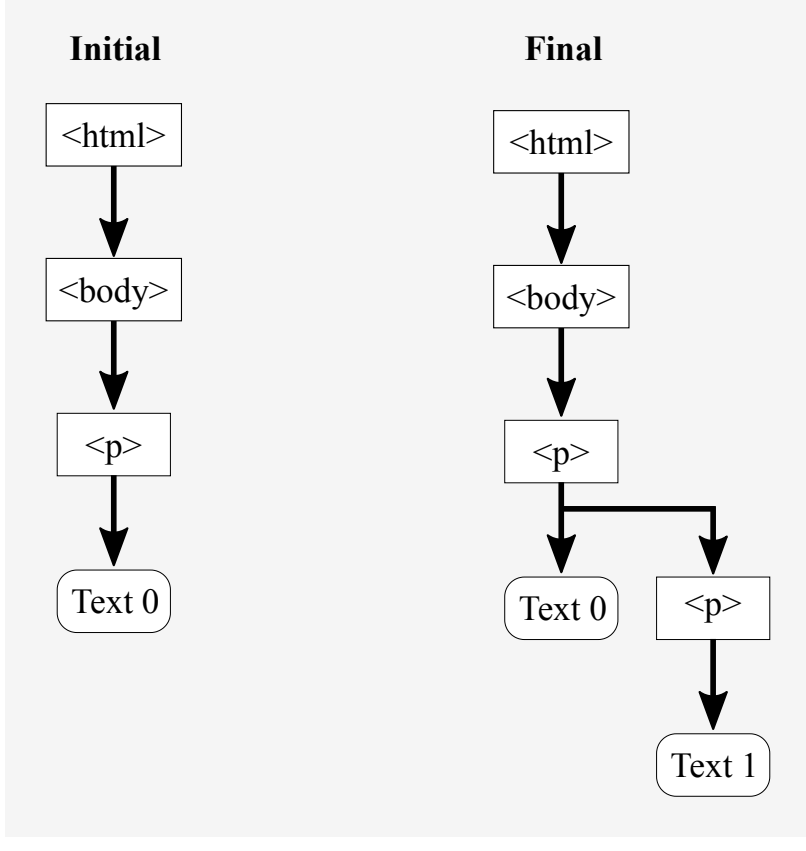
Example 2

```
<body><p>Text0</p></body>
```

```
d3.selectAll('p')  //selects all elements of type p
  .append('p')  //append p as the last child of the selection
  .text('Text1');  //set text to "Text1"
```

```
<body><p>Text0<p>Text1</p></p></body>
```

DOM TREE OF EXAMPLE 2



3. DELETE SELECTED ELEMENTS

- Deleting is done with a selection
- `selection.remove()` Removes the selected elements from the document
- Returns the removed elements

```
<body><p>Text0</p></body>
```

```
d3.selectAll('p')  //selects all elements of type p
  .remove();  //removes selected elements
```

```
<body></body>
```

4. MODIFY SELECTED ELEMENTS TO POSITION AND STYLE

- Modifications are done on the selection
- `selection.text(text)` changes the text of HTML elements elements
- `selection.style(style)` sets the style of HTML elements elements

```
<p>Paragraph1</p>
<p>Paragraph2</p>
```

```
d3.selectAll('p')  //select all p in document
  .text('Paragraph');
d3.select('p')  //select first p in document
  .text('paragraph2');
  .style('color', 'red');
```

GRAPHING WITH SELECT AND APPEND

```
<div id="chart"></div>

<script>
d3.select('#chart')
  .append('div')
  .attr('width', '300px')
  .attr('background-color', '#eee')
  .text('300,000');
</script>
```

What will appear on the page?

```
<body>
  <p>Paragraph1</p>
  <p>Paragraph2</p>

  <script type="text/javascript">
  d3.selectAll('p')
    .text('Paragraph1');
  d3.select('p')
    .text('Paragraph2')
    .style('color', 'red');
  </script>
</body>
```

A. Paragraph1

Paragraph2

B. Paragraph2

Paragraph1

C. Paragraph1

Paragraph2

D. Paragraph2

Paragraph1

What will appear on the page?

```
<body>
  <p>Paragraph1</p>
  <p>Paragraph2</p>

  <script type="text/javascript">
  d3.selectAll('p')
    .text('Paragraph1');
  d3.select('p')
    .text('Paragraph2')
    .style('color', 'red');
  </script>
</body>
```

A. Paragraph1

Paragraph2

B. Paragraph2 ←

Paragraph1

C. Paragraph1

Paragraph2

D. Paragraph2

Paragraph1

What will appear on the page?

```
<body>
  <script type="text/javascript">
    d3.select("div").text("div").style("color", "orange");
  </script>
</body>
```

A. Error because text does not have the color property
B. Blank page
C. Error because div does not support text
D. div

What will appear on the page?

```
<body>
  <script type="text/javascript">
    d3.select("div").text("div").style("color", "orange");
  </script>
</body>
```

A. Error because text does not have the color property
B. Blank page ←
C. Error because div does not support text
D. div

What will appear on the page?

```
<body>
  <p>text</p>
  <script type="text/javascript">
    d3.selectAll("p").text("p").style("color", "orange");
  </script>
</body>
```

A. Error because text does not have a style property
B. Blank page
C. Error because p does not support text
D. p

What will appear on the page?

```
<body>
  <p>text</p>
  <script type="text/javascript">
    d3.selectAll("p").text("p").style("color", "orange");
  </script>
</body>
```

A. Error because text does not have a style property
B. Blank page
C. Error because p does not support text
D. p ←

What will appear on the page?

```
<body>
  <span>Potato </span>
  <script type="text/javascript">
    d3.select("body").select("span").text("Tomato ");
    d3.selectAll("span").append("span").text("Salad ");
    d3.selectAll("span").append("span").text("Carrot ");
  </script>
</body>
```

A. Potato Tomato Salad Carrot
B. Tomato Salad Carrot
C. Tomato Salad Carrot Carrot
D. Potato

What will appear on the page?

```
<body>
  <span>Potato </span>
  <script type="text/javascript">
    d3.select("body").select("span").text("Tomato ");
    d3.selectAll("span").append("span").text("Salad ");
    d3.selectAll("span").append("span").text("Carrot ");
  </script>
</body>
```

A. Potato Tomato Salad Carrot
B. Tomato Salad Carrot
C. Tomato Salad Carrot Carrot ←
D. Potato

GRAPHING WITH DATA JOIN

- Mechanism to bind data to elements in the document
- Central to D3 operations
- Works on the selection!

Data join creating 3 paragraphs and setting their text to the data

```
var dataset = [0, 1, 2];

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function(d) { return d; });
```

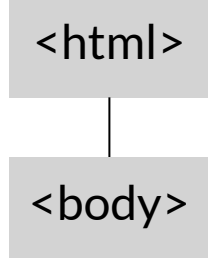
0

1

2

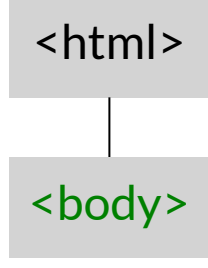
DATA JOIN EXAMPLE (1)

```
var dataset = [0, 1, 2];

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function(d) { return d; });
```



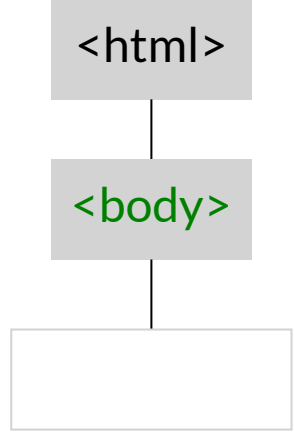
DATA JOIN EXAMPLE (2)

```
var dataset = [0, 1, 2];

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function(d) { return d; });
```



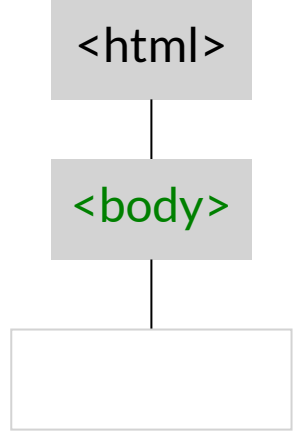
DATA JOIN EXAMPLE (3)

```
var dataset = [0, 1, 2];

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function(d) { return d; });
```



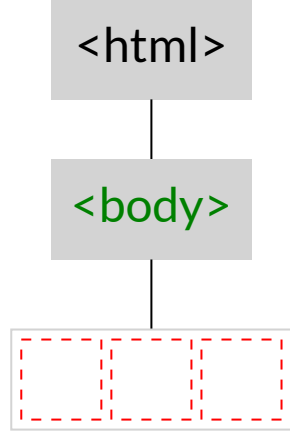
DATA JOIN EXAMPLE (4)

```
var dataset = [0, 1, 2];

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function(d) { return d; });
```



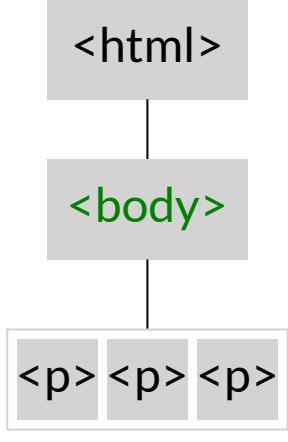
DATA JOIN EXAMPLE (5)

```
var dataset = [0, 1, 2];

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function(d) { return d; });
```



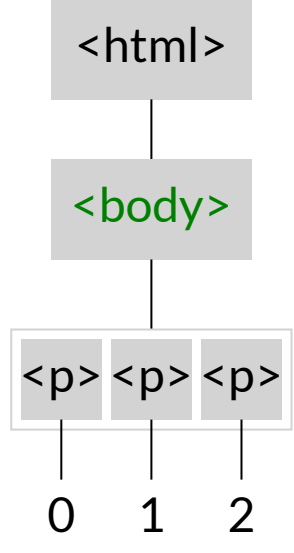
DATA JOIN EXAMPLE (6)

```
var dataset = [0, 1, 2];

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function(d) { return d; })
```



DATA JOIN EXAMPLE (7)

```
var dataset = [0, 1, 2];

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function(d) { return d; });
```



CONFIGURATION FUNCTIONS

Configurations made with functions taking datum and optional index as arguments

```
var dataset = [1, 2, 3];

d3.select('body')
  .selectAll('p')
  .data(dataset)
  .enter()
  .append('p')
  .text(function(d, i) { return 'd=' + d + ' i=' + i; })
  .style('color', function(d, i) { return (i % 2) ? 'red' : 'blue'; });
```

d=1 i=0

d=2 i=1

d=3 i=2

DATA ARRAY

```
var dataset = [1, 2, 3];

d3.select('body')
  .selectAll('p')
  .data(dataset) //dataset must be an array!
  .enter()
  ...
```

EXAMPLES ARRAYS THAT CAN BE PASSED TO DATA()

```
var data = [  //array of objects
  {name: 'A', frequency: .08167},
  {name: 'B', frequency: .01492},
  {name: 'C', frequency: .02780},
  {name: 'D', frequency: .04253},
  {name: 'E', frequency: .12702}
];

var data = [  //array of arrays
  [ 0,  1,  2,  3],
  [ 4,  5,  6,  7],
  [ 8,  9, 10, 11],
  [12, 13, 14, 15]
];
```

What will appear on the page?

```
var dataset = ["red ", "blue ", "red ", "blue ", "red "];

d3.select('body').selectAll('span')
  .data(dataset)
  .enter()
  .append('span')
  .text(function (d) { return d });

d3.selectAll("span")
  .style("text-decoration", function (d, i) { return (i % 2) ? "none" : "underline"; })
  .append("span").text("blue ");
```

A. red blue red blue red

B. red blue red blue red

C. red blue blue blue red blue blue blue red blue

D. red blue blue blue red blue blue blue red blue

What will appear on the page?

```
var dataset = ["red ", "blue ", "red ", "blue ", "red "];

d3.select('body').selectAll('span')
  .data(dataset)
  .enter()
  .append('span')
  .text(function (d) { return d });

d3.selectAll("span")
  .style("text-decoration", function (d, i) { return (i % 2) ? "none" : "underline"; })
  .append("span").text("blue ");
```

A. red blue red blue red

B. red blue red blue red

C. red blue blue blue red blue blue blue red blue ←

D. red blue blue blue red blue blue blue red blue

What will appear on the page?

```
<body>
<script type="text/javascript">
  var dataset = ['red ', 'blue ', 'red ', 'blue ', 'red '];

  d3.select('body')
    .selectAll('span')
    .data(dataset)
    .enter()
    .append('span')
    .text(function (d) { return d });

  d3.selectAll("span")
    .style("text-decoration", "underline");

  dataset = shuffle(dataset);
  d3.select('body')
    .selectAll('span')
    .data(dataset)
    .enter()
    .append('span')
    .text(function (d) { return d });

  d3.selectAll("span")
    .style("text-decoration", "none");

  function shuffle(array) {  // Shuffles array.
    var m = array.length, t, i;
    while (m) {
      i = Math.floor(Math.random() * m--);
      t = array[m], array[m] = array[i], array[i] = t;
    }
    return array;
  }
</script>
</body>
```

A. red blue red blue red
B. red blue red blue red
C. red blue red blue red red blue red blue red
D. red blue red blue red red blue red blue red

What will appear on the page?

```
<body>
<script type="text/javascript">
  var dataset = ['red ', 'blue ', 'red ', 'blue ', 'red '];

  d3.select('body')
    .selectAll('span')
    .data(dataset)
    .enter()
    .append('span')
    .text(function (d) { return d });

  d3.selectAll("span")
    .style("text-decoration", "underline");

  dataset = shuffle(dataset);
  d3.select('body')
    .selectAll('span')
    .data(dataset)
    .enter()
    .append('span')
    .text(function (d) { return d });

  d3.selectAll("span")
    .style("text-decoration", "none");

  function shuffle(array) {  // Shuffles array.
    var m = array.length, t, i;
    while (m) {
      i = Math.floor(Math.random() * m--);
      t = array[m], array[m] = array[i], array[i] = t;
    }
    return array;
  }
</script>
</body>
```

A. red blue red blue red
B. red blue red blue red ←
C. red blue red blue red red blue red blue red
D. red blue red blue red red blue red blue red

OUTLINE

- Design space and design trade-offs
- Graphing in the browser
- Introduction to D3
- Introduction to Vega and Vega-lite

VEGA

TL;DR. *Graphic & data described as JSON (spec)*

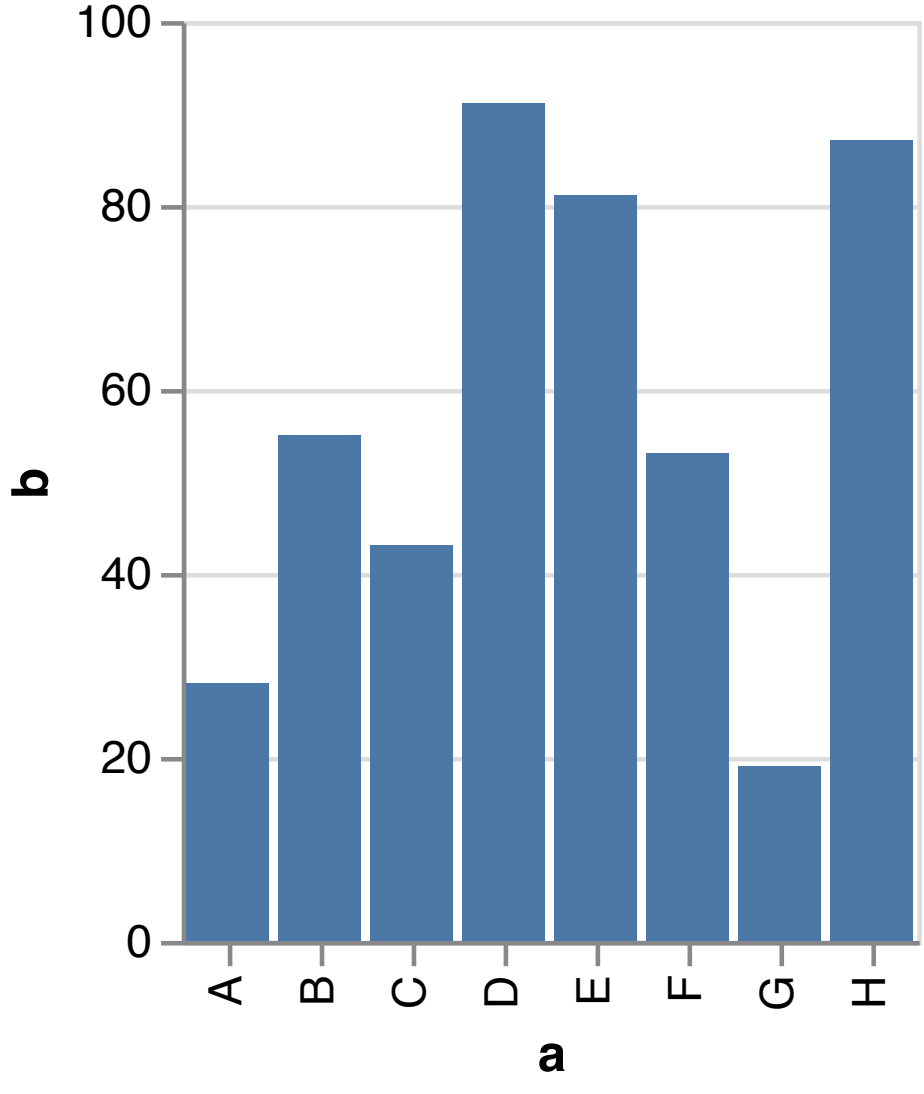
VEGA

TL;DR. *Graphic & data described as JSON (spec)*

To generate a graphic:

1. *Write a Vega or Vega-lite spec*
2. *Use a Javascript library (vegaEmbed) to render the spec*

EXAMPLE VEGA BAR CHART



VEGA-LITE

- High-level visualization grammar in a concise JSON syntax
- Support interactive multi-view graphics
- Compiles to Vega

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v3.json",
  "description": "A simple bar chart with embedded data.",
  "data": {
    "values": [
          {"a": "A","b": 28},
          {"a": "B","b": 55},
          {"a": "C","b": 43},
          {"a": "D","b": 91},
          {"a": "E","b": 81},
          {"a": "F","b": 53},
          {"a": "G","b": 19},
          {"a": "H","b": 87}
    ]
  },
  "mark": "bar",
  "encoding": {
    "x": {"field": "a", "type": "ordinal"},
    "y": {"field": "b", "type": "quantitative"}
  }
}
```

VEGA-LITE

- High-level visualization grammar in a concise JSON syntax
- Support interactive multi-view graphics
- Compiles to Vega

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v3.json",
  "description": "A simple bar chart with embedded data.",
  "data": {
    "values": [
          {"a": "A","b": 28},
          {"a": "B","b": 55},
          {"a": "C","b": 43},
          {"a": "D","b": 91},
          {"a": "E","b": 81},
          {"a": "F","b": 53},
          {"a": "G","b": 19},
          {"a": "H","b": 87}
    ]
  },
  "mark": "bar",
  "encoding": {
    "x": {"field": "a", "type": "ordinal"},
    "y": {"field": "b", "type": "quantitative"}
  }
}
```

VEGA

- Visualization grammar
- Describes the appearance and interactive behavior of a visualization in JSON