

# Data Visualization

Learn how to develop data-oriented web applications

---

Jean-Loïc De Jaeger

Lead Data Engineer at Galeries Lafayette

## Data visualization intro

---

## What is Data Visualization?

---

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

# Importance of Data Visualization

## Why is it important?

- Enhances the understanding of complex data.
- Facilitates quicker decision-making through visual representation.
- Enables the identification of patterns, trends, and outliers.
- Aids in communicating information effectively across different audiences.

## Common Uses

Data visualization is widely used in business analytics, health informatics, finance, marketing, and many other fields to help stakeholders make informed decisions based on data insights.

# Understanding Data Types

## **Quantitative Data:**

Numeric information that measures or quantifies something.

Examples include height, weight, and sales numbers. Visualizations often include histograms, line charts, or scatter plots.

## **Qualitative Data**

Descriptive information that does not include numbers. Examples are colors, labels, and names. Common visualizations include pie charts and bar graphs.

## **Ordinal Data**

A mix of qualitative and quantitative, where the data can be ordered but the intervals between data points are not uniform. Examples are rankings, scales of satisfaction. Suitable for bar charts, line graphs.

## **Nominal Data**

Categorically discrete data such as yes/no, or male/female. These are often visualized using pie charts or bar charts.

# Understanding Data Structures

The structure of data—whether it is univariate, bivariate, or multivariate—determines the complexity and the type of visualization techniques that can be effectively employed.

- **Univariate:** Analyzing one variable. For example, distribution of ages in a survey.
- **Bivariate:** Exploring the relationship between two variables. Examples include comparing height and weight.
- **Multivariate:** Involves three or more variables, complex to visualize. Techniques include parallel coordinates or radar charts.

# Design Principles in Data Visualization

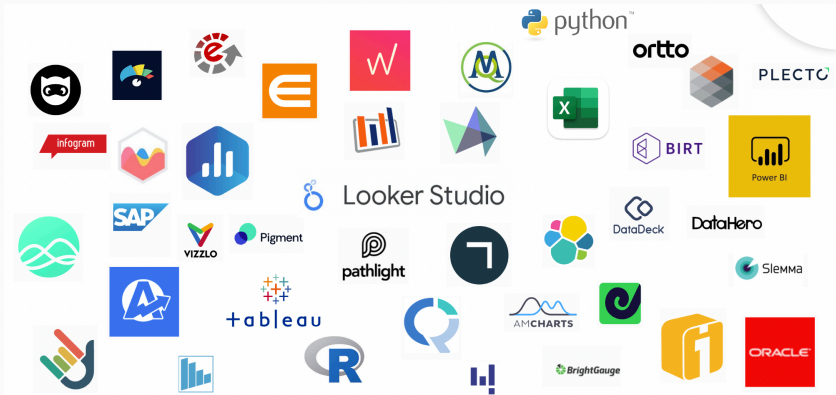
## Core Principles

- **Clarity:** Simplify visuals to enhance understanding and avoid clutter.
- **Accuracy:** Always depict data truthfully without distortion.
- **Efficiency:** Use simple, direct visual forms for quick comprehension.
- **Aesthetics:** Ensure the visual appeal supports data communication.
- **Balance:** Maintain a balance between comprehensive detail and simplicity.

## Design Pitfalls to Avoid

- Avoid overcomplicating visuals with excessive details or colors.
- Choose chart types that correctly represent the underlying data.
- Consider accessibility to make visuals clear for all audiences.

# Popular Tools and Software





## Overview of Popular Tools

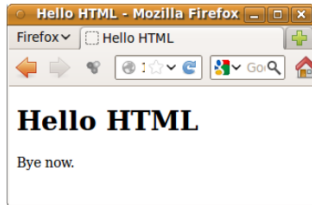
- **Tableau:** A leading tool for creating interactive and shareable dashboards.
- **Microsoft Power BI:** A comprehensive business intelligence platform that integrates well with Microsoft's ecosystem.
- **Metabase:** An open-source business intelligence tool that allows easy querying and visualization of data.
- **D3.js:** A JavaScript library for producing dynamic, interactive data visualizations in web browsers.
- **Plotly:** A library for creating interactive graphs and dashboards in Python, R, and JavaScript. It is built on the top of D3.js.

## D3.js Introduction

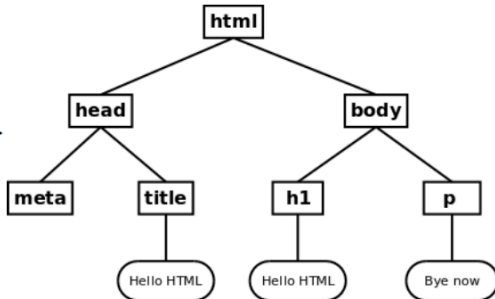
---

Let's review what the DOM is

# DOM review



```
1 <!doctype html>
2 <html lang="fr">
3 <head>
4   <meta charset="utf-8">
5   <title>Hello HTML</title>
6 </head>
7 <body>
8   <h1>Hello HTML</h1>
9   <p>Bye now.</p>
10 </body>
11 </html>
```



# DOM review

- The Document Object Model (DOM) is a programming interface for web documents.
- It represents the structure of a document as a hierarchical tree of nodes.
- The DOM allows you to manipulate the contents and structure of a web page using JavaScript.
- Each node in the DOM tree represents an element, attribute, or text content of the document.
- Nodes can be selected using methods like `getElementById`, `getElementsByClassName`, or `querySelector`.



## Foundation of dynamic user interfaces

The DOM is a powerful tool for building dynamic web pages and creating interactive user interfaces.



# Data-Driven Documents

# D3.js Introduction

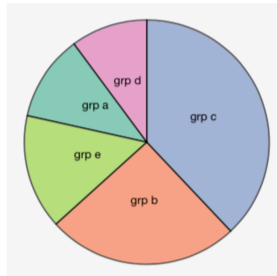
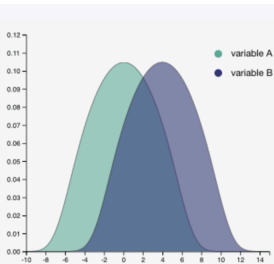
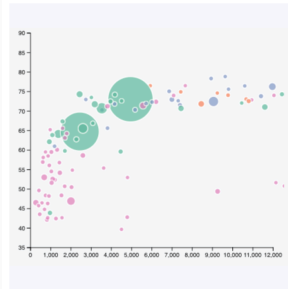
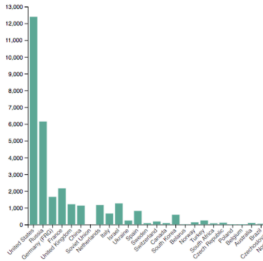
- D3.js (Data-Driven Documents) is a JavaScript library for creating data visualizations on the web.
- It uses web standards like HTML, SVG, and CSS to create interactive and dynamic visualizations.
- It provides powerful data manipulation and transformation capabilities, allowing you to work with data from a variety of sources and in a variety of formats.
- D3.js provides a wide range of built-in visualization tools like bar charts, line charts, scatterplots, and more.
- It also allows you to create custom visualizations using its flexible API and powerful graphics capabilities.



## Main takeaway

Low level JavaScript library to create any kind of visualization

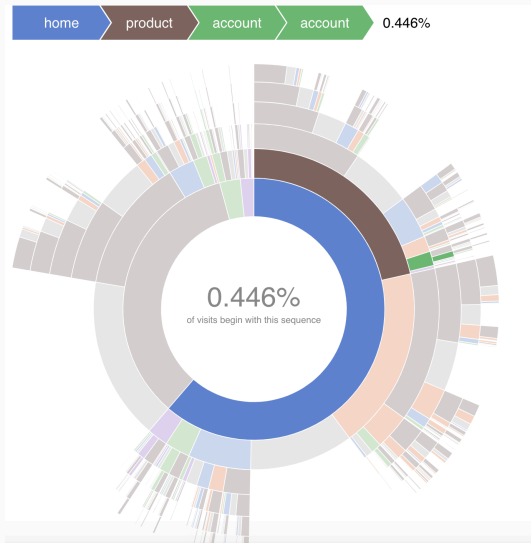
# D3.js visualization examples



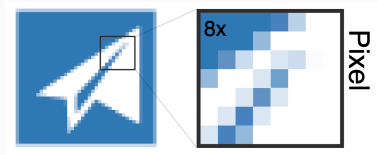


# D3.js visualization examples

More examples <https://d3-graph-gallery.com/>

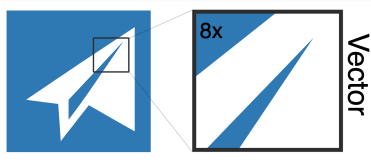


## Bitmap



- Bitmap (or raster) graphics are made up of a grid of individual pixels.
- Each pixel in a bitmap image contains information about its color and brightness.
- Bitmap images are resolution-dependent, meaning that they can appear blurry or pixelated when scaled up or down.
- Bitmap images are commonly used for photographs and complex images with many colors and gradients.

# SVG



- SVG (Scalable Vector Graphics) is a vector-based image format that uses mathematical descriptions of shapes to create images.
- SVG images are resolution-independent, meaning that they can be scaled up or down without losing quality.
- SVG images are commonly used for logos, icons, and other images with simple shapes and colors.
- SVG images can be edited and manipulated using software like Adobe Illustrator or Inkscape.

# Build shapes with SVG

This is my first sentence



## HTML

```
<p>This is my first sentence</p>
<svg>
  <circle class="target"
    style="fill: #69b3a2"
    stroke="black" cx=50 cy=50
    r=40></circle>
</svg>
<!-- Load d3.js -->
<script
src="https://d3js.org/d3.v4.js"></script>
<script src="app.js"></script>
```

## JavaScript

```
d3
  .select(".target")
  .style("stroke-width", 8)
```

# Build three circles



## JavaScript

```
var svg = d3.select("#dataviz_area")
svg.append("circle")
  .attr("cx", 2)
  .attr("cy", 2)
  .attr("r", 40)
  .style("fill", "blue");

svg.append("circle")
  .attr("cx", 140)
  .attr("cy", 70)
  .attr("r", 40)
  .style("fill", "red");

svg.append("circle")
  .attr("cx", 300)
  .attr("cy", 100)
  .attr("r", 40)
  .style("fill", "green");
```

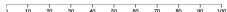
# Build three circles



```
var svg = d3.select("#viz_area")
// Create a scale: transform value in pixel
var x = d3.scaleLinear()
    // This is the min and the max
    // of the data: 0 to 100 if percentages
    .domain([0, 100])
    // This is the corresponding value
    // I want in Pixel
    .range([0, 400]);

svg.append("circle")
    .attr("cx", x(10)).attr("cy", 100)
    .attr("r", 40).style("fill", "blue");
svg.append("circle")
    .attr("cx", x(50)).attr("cy", 100)
    .attr("r", 40).style("fill", "red");
svg.append("circle")
    .attr("cx", x(100)).attr("cy", 100)
    .attr("r", 40).style("fill", "green");
```

# Build three circles

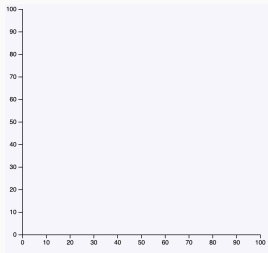


```
var svg = d3.select("#Viz_area")
var x = d3.scaleLinear()
    .domain([0, 100])
    .range([0, 400]);

// Show the axis that corresponds
// to this scale
svg.call(d3.axisBottom(x));

// Add 3 dots for 0, 50 and 100%
svg.append("circle")
    .attr("cx", x(10)).attr("cy", 100)
    .attr("r", 40).style("fill", "blue");
svg.append("circle")
    .attr("cx", x(50)).attr("cy", 100)
    .attr("r", 40).style("fill", "red");
svg.append("circle")
    .attr("cx", x(100)).attr("cy", 100)
    .attr("r", 40).style("fill", "green");
```

# Build three circles



```
// set the dimensions and margins of the graph
var margin = {top: 10, right: 40, bottom: 30, left: 30},
    width = 450 - margin.left - margin.right,
    height = 400 - margin.top - margin.bottom;

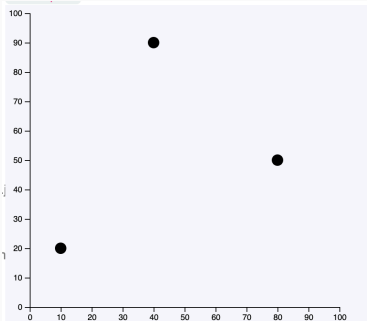
// append the svg object to the body of the page
var svg = d3.select("#Area")
    .append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    // translate this svg element to leave some margin.
    .append("g")
    .attr("transform", "translate(" + margin.left + "," +
        + margin.top + ")");

// X scale and Axis
var x = d3.scaleLinear()
    .domain([0, 100])
    .range([0, width]);
svg
    .append('g')
    .attr("transform", "translate(0," + height + ")")
    .call(d3.axisBottom(x));

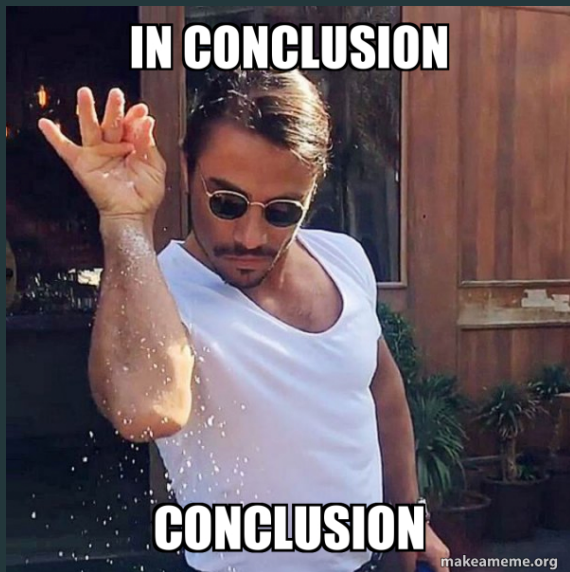
// X scale and Axis
var y = d3.scaleLinear()
    .domain([0, 100])
    .range([height, 0]);
svg
    .append('g')
    .call(d3.axisLeft(y));
```



# Build three circles



```
// Same as previous code  
  
// Add 3 dots for 0, 50 and 100%  
svg  
  .selectAll("whatever")  
  .data(data)  
  .enter()  
  .append("circle")  
    .attr("cx", function(d){ return x(d.x) })  
    .attr("cy", function(d){ return y(d.y) })  
    .attr("r", 7)
```



### Conclusion of d3.js

- Very low-level data visualization library
- Very powerful
- Not that easy to learn/use
- Good fit for very customized visualization
- Not ideal for standard visualization and dashboards
- A lot of data visualization libraries are based on d3.js



# Plotly Introduction

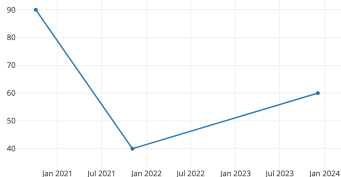
- Plotly is a JavaScript graphing library that allows users to create interactive visualizations.
- It is based on **D3.js**
- Plotly supports a wide range of chart types, including scatter plots, line charts, bar charts, pie charts, and more.
- Plotly's interactive features include zooming, panning, and hovering over data points to display additional information.
- The library is compatible with a wide range of programming languages, including JavaScript, Python, R, and MATLAB.



## Main takeaway

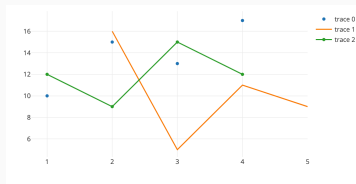
Much much easier to use than D3.js. It's often a better fit for most of data visualization purposes

# Plotly configuration object



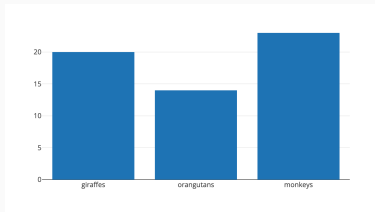
```
var trace1 = {  
  x: ['2020-10-04', '2021-11-04',  
      '2023-12-04'],  
  y: [90, 40, 60],  
  type: 'scatter'  
};  
  
var data = [trace1];  
  
var layout = {  
  title: 'Scroll and Zoom',  
  showlegend: false  
};  
  
Plotly.newPlot('myDiv', data, layout,  
  {scrollZoom: true});
```

# Plotly configuration object



```
var trace1 = {  
  x: [1, 2, 3, 4],  
  y: [10, 15, 13, 17],  
  mode: 'markers',  
  type: 'scatter'  
};  
var trace2 = {  
  x: [2, 3, 4, 5],  
  y: [16, 5, 11, 9],  
  mode: 'lines',  
  type: 'scatter'  
};  
var trace3 = {  
  x: [1, 2, 3, 4],  
  y: [12, 9, 15, 12],  
  mode: 'lines+markers',  
  type: 'scatter'  
};  
var data = [trace1, trace2, trace3];  
Plotly.newPlot('myDiv', data);
```

# Plotly configuration object



```
var data = [  
  {  
    x: ['giraffes', 'orangutans',  
        'monkeys'],  
    y: [20, 14, 23],  
    type: 'bar'  
  }  
];  
Plotly.newPlot('myDiv', data);
```