# Summary :

To create our pie chart visualization, we will be using the following D3 methods.

|  |  |
| --- | --- |
| D3 Method | Description |
| SVG Path | Creates a path in SVG from the defined commands. |
| d3.scaleOrdinal() | Creates an ordinal scale |
| d3.pie() | Creates an pie generator |
| d3.arc() | Creates an arc generator |

# SVG Path

A path element is used to create a path on the SVG. The SVG path takes commands to draw paths in SVG.

# Example: SVG Path

<body>

<svg height="210" width="400">

<path d="M150 0 L75 200 L225 200 Z" />

</svg>

</body>

The above code defines the following path starting from the point (150,0), creates a line from (150,0) to (75,200), another line from (75,200) to (225,200) and finally a closing path to (150,0).

# d3.scaleOrdinal()

We already learned about [Scales in D3](https://www.tutorialsteacher.com/d3js/scales-in-d3). The d3.scaleOrdinal() constructs a new ordinal scale with an empty domain and the specified range. In an ordinal scale, the order is of utmost importance and not the difference.

# Example:

<body>

<script>

var color = d3.scaleOrdinal(['#4daf4a','#377eb8','#ff7f00','#984ea3','#e41a1c']);

console.log(color(0))

console.log(color(1))

console.log(color(2))

console.log(color(3))

console.log(color(4))

console.log(color(5))

</script>

</body>

In the above example, the color variable will act as a scaling function which will return specified color for the specified index. For example, color(0) will return #4daf4a, color(1) will return #377eb8 color(2) will return #ff7f00 and so on. If a value at the specified index does not exists in the specified range, then it will restart from 0. i.e., range does not contain element at 5th index (index starts with 0), so it will start again from 0 and will return #4daf4a.

# d3.pie()

The d3.pie() function takes in a dataset and creates handy data for us to generate a pie chart in the SVG. It calculates the start angle and end angle for each wedge of the pie chart. These start and end angles can then be used to create actual paths for the wedges in the SVG.

# Consider the following example.

Example: d3.Pie()

<script>

var data = [2, 4, 8, 10];

var pie = d3.pie()

console.log(pie(data))

</script>

In the above example, we defined a d3.pie() object and provided our data to the pie function. This function calculated certain fields for the pie chart like startAngle and endAngle along with the data values.

# d3.arc()

The d3.arc() generates an arc. These are the paths that will create our pie's wedges. Arcs need an inner radius and outer radius. If the inner radius is 0, the result will be a piechart, otherwise the result will be a donut chart. We need to supply these generated arcs to our SVG path elements.

The following example demonstrates a simple pie chart.

# Example: Pie Chart

<body>

<svg width="300" height="200"> </svg>

<script>

var data = [2, 4, 8, 10];

var svg = d3.select("svg"),

width = svg.attr("width"),

height = svg.attr("height"),

radius = Math.min(width, height) / 2,

g = svg.append("g").attr("transform", "translate(" + width / 2 + "," + height / 2 + ")");

var color = d3.scaleOrdinal(['#4daf4a','#377eb8','#ff7f00','#984ea3','#e41a1c']);

// Generate the pie

var pie = d3.pie();

// Generate the arcs

var arc = d3.arc()

.innerRadius(0)

.outerRadius(radius);

//Generate groups

var arcs = g.selectAll("arc")

.data(pie(data))

.enter()

.append("g")

.attr("class", "arc")

//Draw arc paths

arcs.append("path")

.attr("fill", function(d, i) {

return color(i);

})

.attr("d", arc);

</script>

</body>

We first define all our variables like width of the svg and, height of the svg. We calculate the radius as Math.min(width, height) / 2 to ensure that our generated pie will fit into the bounds of the SVG. For this, we choose whichever of the width and height is the minimum value.

And finally, we add a path element for each of our wedges. We provide the arc generated earlier and fill it with a color from our color scale.

For this example, we will take the following dataset that we have stored in a 'browseruse.csv' file. This dataset shows the browser statistics for a dummy website.

browseruse.csv

browser,percent

Chrome,73.70

IE/Edge,4.90

Firefox,15.40

Safari,3.60

Opera,1.00

Let's create our pie chart visualization for the above dataset.

<!DOCTYPE html>

<html>

<head>

<style>

.arc text {

font: 10px sans-serif;

text-anchor: middle;

}

.arc path {

stroke: #fff;

}

.title {

fill: teal;

font-weight: bold;

}

</style>

<script src="https://d3js.org/d3.v4.min.js"></script>

</head>

<body>

<svg width="500" height="400"></svg>

<script>

var svg = d3.select("svg"),

width = svg.attr("width"),

height = svg.attr("height"),

radius = Math.min(width, height) / 2;

var g = svg.append("g")

.attr("transform", "translate(" + width / 2 + "," + height / 2 + ")");

var color = d3.scaleOrdinal(['#4daf4a','#377eb8','#ff7f00','#984ea3','#e41a1c']);

var pie = d3.pie().value(function(d) {

return d.percent;

});

var path = d3.arc()

.outerRadius(radius - 10)

.innerRadius(0);

var label = d3.arc()

.outerRadius(radius)

.innerRadius(radius - 80);

d3.csv("browseruse.csv", function(error, data) {

if (error) {

throw error;

}

var arc = g.selectAll(".arc")

.data(pie(data))

.enter().append("g")

.attr("class", "arc");

arc.append("path")

.attr("d", path)

.attr("fill", function(d) { return color(d.data.browser); });

console.log(arc)

arc.append("text")

.attr("transform", function(d) {

return "translate(" + label.centroid(d) + ")";

})

.text(function(d) { return d.data.browser; });

});

svg.append("g")

.attr("transform", "translate(" + (width / 2 - 120) + "," + 20 + ")")

.append("text")

.text("Browser use statistics - Jan 2017")

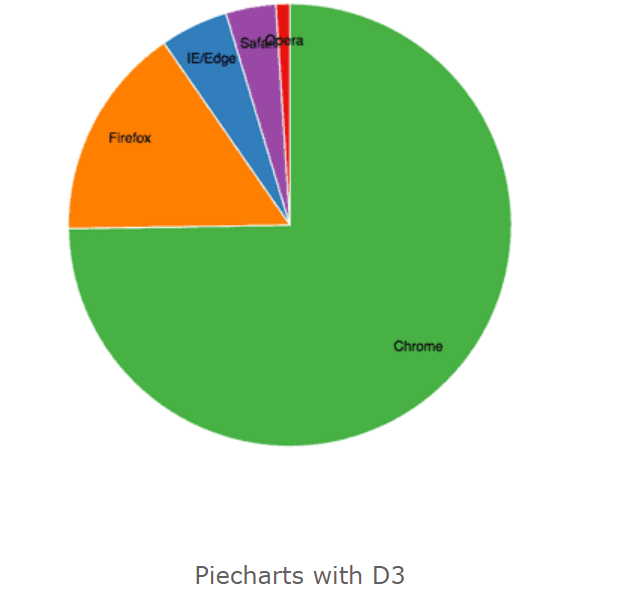
.attr("class", "title")

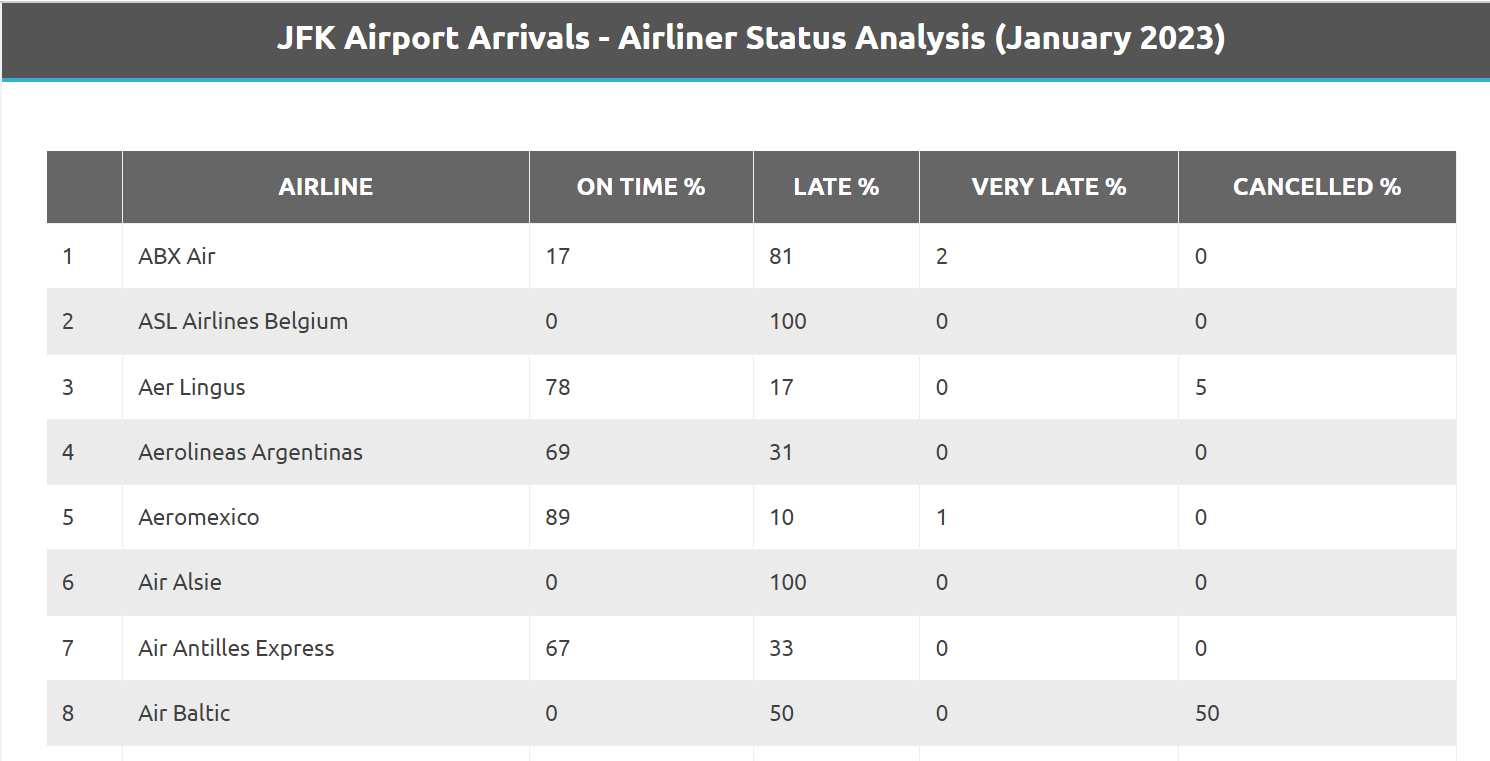
</script>

</body>

</html>

The above code will display the following pie chart.





The above information has been taken from a variety of web sources and we strongly urge all users of this data to check with each Airline in question as to it's own flight performance at JFK Airport.

# Our own code:

function main() {

var svg = d3.select('svg'),

width = svg.attr('width'),

height = svg.attr('height'),

radius = Math.min(width, height) / 2;

var g = svg.append('g')

.attr('transform', 'translate(' + width / 2 + ',' + height / 2 + ')');

var color = d3.scaleOrdinal(['#C7CEEA','#B5EAD7','#FFDAC1', '#FF9AA2', ])

var pie = d3.pie().value(function(d){

return d.ONTIME && d.LATE && d.CANCELLED

})

var path = d3.arc()

.outerRadius(radius - 20)

.innerRadius(100);

var label = d3.arc()

.outerRadius(radius)

.innerRadius(radius - 150);

d3.csv('https://raw.githubusercontent.com//rizwanzahid710/data/main/airline/JFK\_data.csv').then(

function(data){

console.log(data)

var arc = g.selectAll('.arc')

.data(pie(data))

.enter().append('g')

.attr('class', 'arc')

arc.append('path')

.attr('d', path)

.attr('fill', function(d){return color(d.data.AIRLINE);})

arc.append('text')

.attr('transform', function(d){return 'translate(' + label.centroid(d) + ')';})

.text(function(d){return d.data.AIRLINE});

}

);

}

# data scraping from website using d3 js

In all fairness, dilemmas surrounding airlines, airports and air traffic control have been studied in great detail and curiosity for economists and scholars under the field of Science, Technology & Society." It turns out that these issues are not simple to resolve and the system is difficult to optimize. This is investigated in great detail by *Allenby and Sarewitz* in the Techno-Human Condition in which they address the complexity of problem solving and predicting even the near future (focusing on social and technological aspects of air traffic control systems and human interaction).

However, this data could be analyzed against multiple years and could result in improvements. If the airline or airport knows that there will be these sorts of dilemmas, they could notify users in advance or offer some form of compensation. This is under the premise that these conditions are unavoidable (weather is relative to location and season), and the best course of action would be a social resolution.

Recently, airlines have experienced disruptions due to entrepreneurial innovation and business models. An American example is "Spirit Airlines," and while they offer very low quality flights (no carry on, no free drinks) they offer cheap alternatives to fly which targets an audience of college students, travelers, backpackers and short "get-aways." This is addressed in this report, since aspects of social media and user experience are disrupting the stocks and value chain for large companies that cannot adapt. Once a companies image is tarnished, it is very difficult to recover. Therefore, dashboards such as these can give some direction to potentially mitigating or creating business plans to improve the companies profits.

The d3. pie() is **used to construct a pie generator that has its default settings**. This pie generator takes an array of data and then returns an array of objects that contains details about each arc angle.

# Scrape a site

# Example

|  |
| --- |
| { |
|  | "name": "web-scraper-d3-project", |
|  | "version": "0.0.1", |
|  | "description": "An example web scraper using D3", |
|  | "main": "server.js", |
|  | "author": "Robert Higdon", |
|  | "scripts": { |
|  | "start": "node server.js", |
|  | "scrape": "node scraper.js" |
|  | }, |
|  | "dependencies": { |
|  | "d3": "\*", |
|  | "express": "\*", |
|  | "jade": "\*", |
|  | "path": "^0.11.14", |
|  | "request": "\*", |
|  | "x-ray": "\*" |
|  | } |
|  | } |

# How do I scrape data from a website using JavaScript?

First, install Cheerio and Axios by running the following command: npm install cheerio axios . Then create a new file called crawler. js and copy/paste the following code: const axios = require('axios'); const cheerio = require('cheerio'); const getPostTitles = async () => { try { const { data } = await axios.

How do I scrape specific data from a website?

# How do we do web scraping?

1. Inspect the website HTML that you want to crawl.
2. Access URL of the website using code and download all the HTML contents on the page.
3. Format the downloaded content into a readable format.
4. Extract out useful information and save it into a structured format.

**Web scraping is completely legal if you scrape data publicly available on the internet**. But some kinds of data are protected by international regulations, so be careful scraping personal data, intellectual property, or confidential data.