js Libraries

Интернет програмирање

Why js frameworks / libraries?

- Reusable templates, elements, and components feature
- Ability to automate tasks
- Permit customization of coding and debugging
- Improve developers' productivity
- Streamline development procedures

Libraries vs. Frameworks

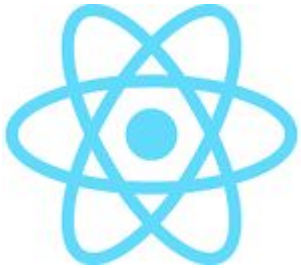
- The main distinction between a framework and a library is that a framework inverts program control. It informs the developer of what they require. A library, however, does not. Instead, a programmer calls the library when and where he needs it.
- Difference translates to JavaScript specifically.
  - Frameworks in JavaScript will usually provide you with a backbone for building and developing (mostly) web applications and pages, be it frontend or backend.
  - Libraries will provide you with utilities to enhance your web application, be it visuals, such as tools for animations and reactive content, or programming utilities that reduce and streamline certain tasks by reusing flexible code.
- The basic difference between a Javascript library vs framework (or in any other language) is:
  - Your code will call a Library.
  - A Framework will call your code.
  - A Framework will usually define the backbone of your application, while the library will help you develop its visuals, internal functions, and algorithms.

Trade Offs

Framework	**Library**
Concise	Explicit
Template-centric	JavaScript-centric
Separate template	Single-file component
Standard	Non-standard

What Are JavaScript Libraries?

- JavaScript libraries contain various functions, methods, or objects to perform practical tasks on a webpage or JS-based application.
- Similarly, a JavaScript library has codes or functions that developers can reuse and repurpose.
- JavaScript libraries are collections of prewritten code snippets that can be used (and reused) to perform common JavaScript functions
- A Library is a package of code that developers can use in their own projects to help solve a specific problem. The objective of a library's code is to be flexible and highly reusable so it can be deployed in many different kinds of projects. Developers are in control of exactly when to use the library in their own code and can use it more than once if needed.



Popular Libraries in Javascript

- **React.js**
  - The most popular and widely used JavaScript library out there these days. It is an open-source library. This is sometimes considered as a framework because there are so many packages available for reacting like handling routes which you can integrate into react.js and can develop any type of application.
- **jQuery**
  - Old javascript library that is used for DOM (Document Object Model) manipulation and handling events like mouse clicks, scrolling, etc. Also, it contains AJAX which handles the Asynchronous part of any application.
- **D3.js**
  - Stands for Data-Driven Documents. It is basically used for data visualization. It takes a certain amount of data and applies it to manipulate the Document object model.
- **Underscore.js**
  - Lightweight JavaScript library and not a complete framework that was written by Jeremy Ashkenas that provides utility functions for a variety of use cases in our day-to-day common programming tasks.
- **Lodash**
  - JavaScript library that works on the top of underscore.js. Lodash helps in working with arrays, strings, objects, numbers, etc. It provides us with various inbuilt functions and uses a functional programming approach.
- **Anime.js**
  - Small, lightweight JavaScript library with a simple and small powerful API. It works with the DOM element, CSS, and JavaScript object.
- **Chart.js**
  - Open-source JavaScript library on Github that allows you to draw different types of charts by using the HTML5 canvas element. Since it uses canvas, you have to include a polyfill to support older browsers

What Are JavaScript Frameworks?

- JavaScript frameworks are a full toolset that help shape and organize your website or web application.
- JavaScript framework is a set of JavaScript code libraries that provide pre-written code for everyday programming tasks to a web developer
- A Framework, on the other hand, provides a more tightly structured code in which developers can introduce their own code in specific points of control. These points allow the extension of the framework's functionality to fit the developer's specific needs. The framework is the one in control of when to call the code developers have provided when it deems necessary.

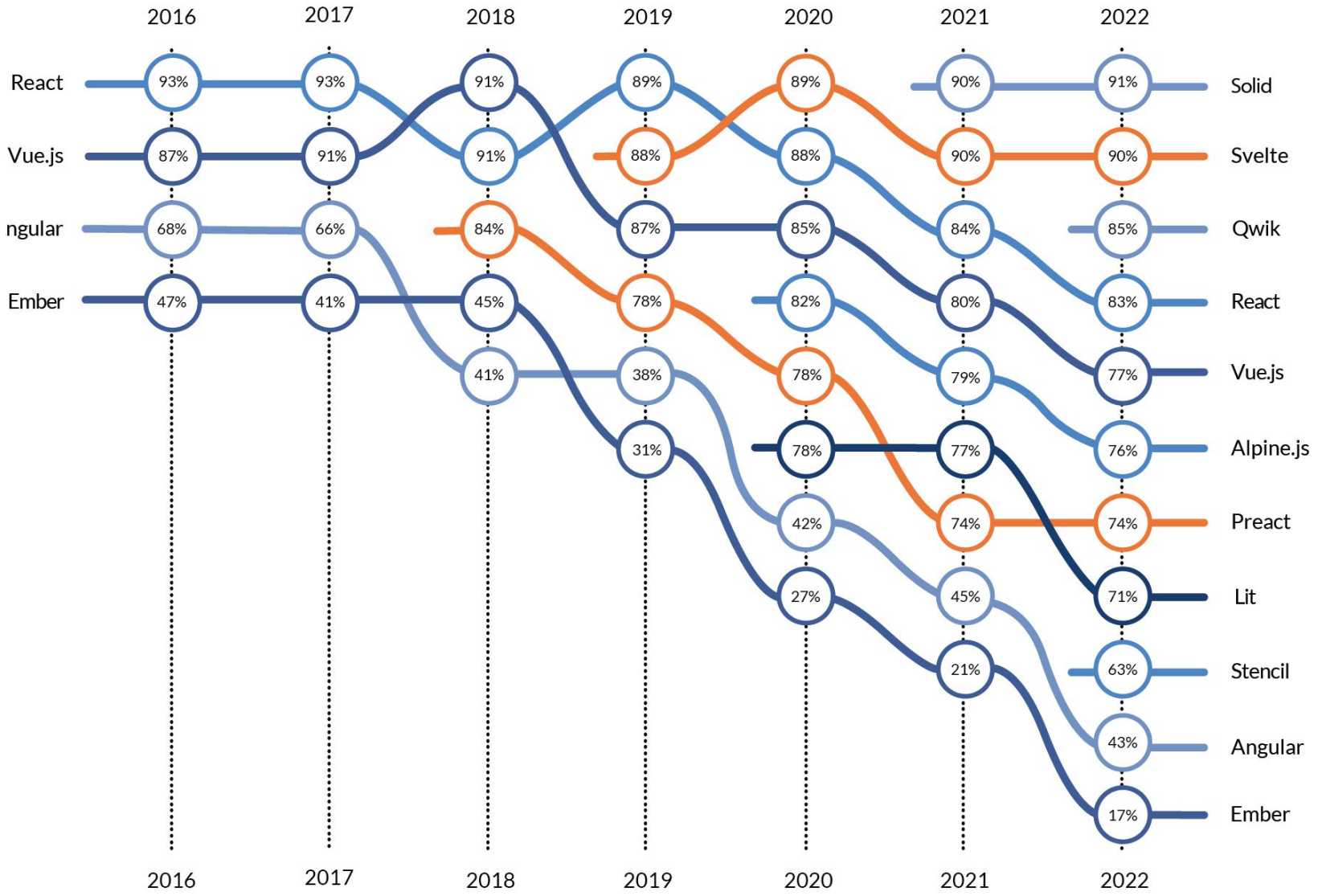
Frameworks available in Javascript

- Angular
  - Open-source framework developed by Google. This framework is based on typescript(a superset of javascript) you can say that some strict type checking applies to javascript. It is free and used for cross-platform development.
- Vue.js
  - Open-source framework for javascript. It is used for building great user Interfaces and single-page applications(SPA's).
- Ember.js
  - Open-source JavaScript framework used for developing large client-side web applications which is based on Model-View-Controller (MVC) architecture. Ember is designed for reducing development time and increasing productivity, it is one of the fastest-growing front-end application frameworks being adopted worldwide.
- Next.js
  - React-based framework. It has the power to Develop beautiful Web applications for different platforms like Windows, Linux, and mac. If you have little experience in react and looking forward to knowing more about react ecosystem then you should have knowledge about the Next.js framework.
- Mocha
  - Testing framework for Javascript running on Node.js. The frameworks make it easier to test asynchronous Javascript concepts in the browser. Mocha is widely used for testing Javascript codes before deploying them onto the server.

js frameworks / libraries by developer usage

Source: State of JavaScript 2022

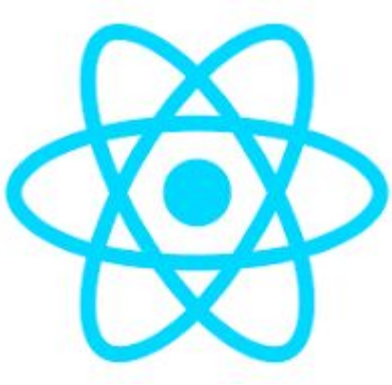
js frameworks / libraries by level of satisfaction



Source: State of JavaScript 2022

Usage

- Data Visualization in Maps and Charts
  - Chart.js, Apexcharts, Algolia Places
- DOM Manipulation
  - jQuery, Umbrella JS
- Data Handling
  - D3.js
- Database
  - TaffyDB, ActiveRecord.js
- Forms
  - wForms, LiveValidation, Validanguage, qForms
- Animations
  - Anime.js, JSTweener
- Image Effects
  - ImageFX, Reflection.js
- Fonts
  - typeface.js
- Math and String Functions
  - Date.js, Sylvester, JavaScript URL Library
- User Interface and Its Components
  - ReactJS, Glimmer.js



React

React

- React is a JavaScript library for building user interfaces.
- React is used to build single-page applications.
- React allows us to create reusable UI components.
- React, sometimes referred to as a frontend JavaScript framework
- JavaScript library created by Facebook.
- React is a tool for building UI components.
- React creates a VIRTUAL DOM in memory.

React history

- 2010 - The first signs of React
- 2011 - Created by Facebook
- 2012 - Used at Instagram
- 2013 - Open sourced
- 2014 - The year of Expansion
- 2015 - React Native
- 2016 - React 15
- 2017 - The year of further improvements

Why React?

- Flexibility
- Developer experience
- Corporate investment
- Community support
- Performance
- Testability

Setting up a React Environment

- Install Node.js and npm
- Test

  `npm -v`

  `node -v`
- If you have npx and Node.js installed, you can create a React application by using create-react-app.
- Run this command to create a React application named my-react-app:

  ```
  npx create-react-app my-react-app
  ```
- The create-react-app will set up everything you need to run a React application.

React JSX

- JSX stands for JavaScript XML.
- JSX allows us to write HTML in React.
- JSX makes it easier to write and add HTML in React.

JSX

```
React.createElement(
  "div",
  { className: "split" },

React.createElement("input"),
        React.createElement(
          "p",
          null,
          "Sunday 4:01 pm")
);
```

```
<div className="split">
  <input/>
  <p>
    Sunday 4:01 pm
  </p>
</div>
```

React Components

- Components are like functions that return HTML elements.
- Components are independent and reusable bits of code.
- They serve the same purpose as JavaScript functions, but work in isolation and return HTML.
- Components come in two types
  - Class components
  - Function components
- Why Function Components?
  - Complexities around the use of classes
  - Functions in combination with hooks enable straightforward components creating structured apps

Пример - HTML

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
      <title>First React Demo</title>
      <link rel="stylesheet" type="text/css" href="styles.css" />
   </head>
   <body>
      <div id="dateJs"></div>
      <div id="dateReact"></div>
      <script crossorigin
src="https://unpkg.com/react@17/umd/react.production.min.js"></script>
      <script crossorigin
src="https://unpkg.com/react-dom@17/umd/react-dom.production.min.js"></script>
      <script src="script.js"></script>
   </body>
</html>
```

Пример - js

```
const jsDiv = document.getElementById("dateJs");
const reactDiv = document.getElementById("dateReact");
const render = () => {
 jsDiv.innerHTML = `
   <div class="split">
     JS template
     <input />
     <p>${new Date()}</p>
   </div>
 `;
 var divToRender = React.createElement(
   "div", {
     className: "split"
   },
   "React template ",
   React.createElement("input"),
   React.createElement("p", null, new Date().toString())
 );
 ReactDOM.render(divToRender, reactDiv);
};
setInterval(render, 1000);
```

Svelte

History

- The predecessor is Ractive.js
- Version 1 of Svelte was written in JavaScript and was released on 29 November 2016
- Version 2 was released on 19 April 2018
- Version 3 was written in TypeScript and was released on 21 April 2019
- Version 4 was released on 22 June 2023

Size and performances

React

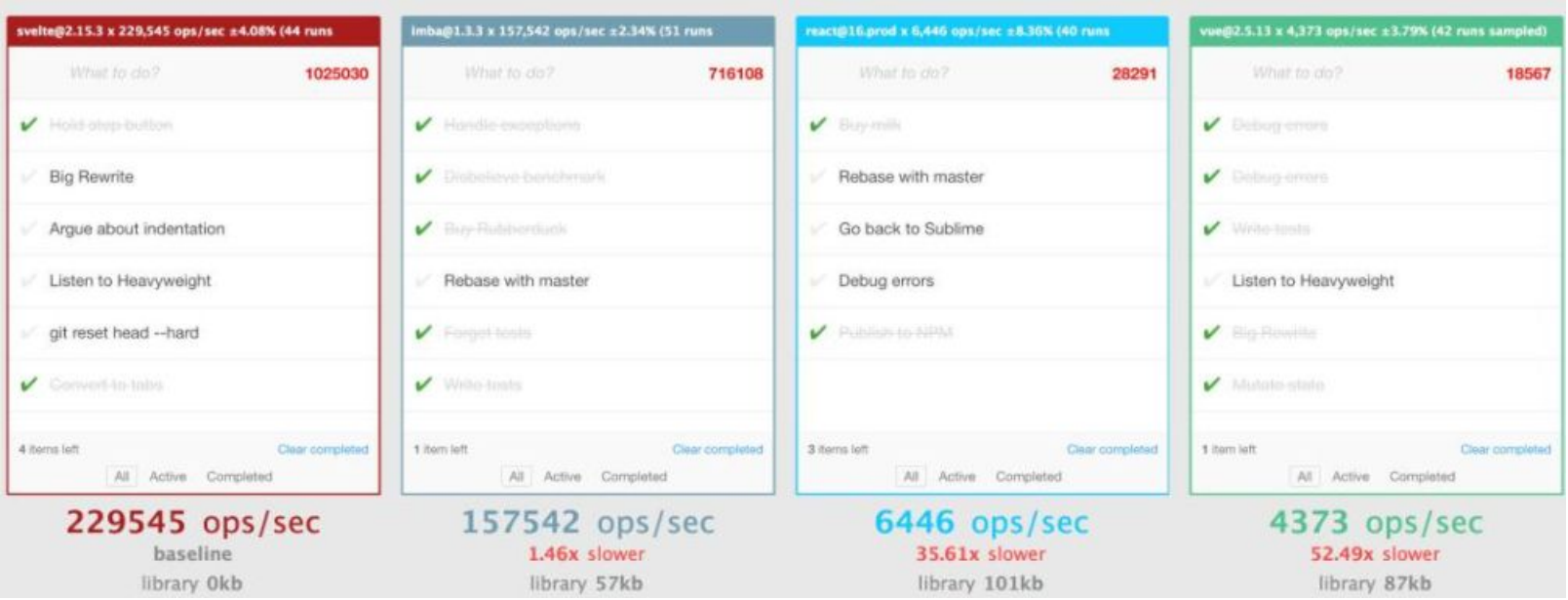
>100 KB

Vue

>80 KB

Svelte

<3 KB



Performance

Memory allocation in MBs ± 95% confidence interval

Name	vue-v3.2.37	svelte-v3.50.1	angular-v13.0.0	react-v17.0.2
ready memory Memory usage after page load.	1.1 (1.30)	0.9 (1.02)	1.9 (2.12)	1.3 (1.50)
run memory Memory usage after adding 1,000 rows.	4.3 (1.81)	3.3 (1.38)	5.3 (2.24)	5.5 (2.34)
update every 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	4.3 (1.76)	3.3 (1.33)	5.3 (2.17)	6.0 (2.45)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	1.5 (1.54)	1.1 (1.20)	2.6 (2.73)	2.1 (2.20)
run memory 10k Memory usage after adding 10,000 rows.	30.2 (2.05)	22.9 (1.55)	32.3 (2.19)	39.2 (2.66)
geometric mean of all factors in the table	1.67	1.28	2.28	2.19

Duration in milliseconds ± standard deviation (Slowdown = Duration/Fastest)

Name	svelte -v2. 9.7-non-non-keyed	react-v16.4.1-non-keyed	angular-v.1.0-keyed
create rows Duration for creating 1000 rows after the page loaded.	220.9 ± 5.2 (1.2)	184.7±(1.0)	185.2 ± 10.2(1.0)
replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations)	56.1 ± 3.9(1.0)	61.4±2.6 (1.1)	161.2 ± 2.7(2.9)
partial update Time to update the text of every 10th row (with 5 warmup iterations) for a table with 10k rows.	65.2 ± 2.5(1.0)	81.3±(1.2)	68.8 ± 3.7(1.1)
select row Duration to highlight a row in response to a click on the row (with 5 warmup iterations).	8.8 ± 3.4(1.0)	10.4 ± 2.3(1.0)	7.9 ± 4.3(1.0)
not aligned Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	15.1 ± 4.2(1.0)	14.8 ± 4.5(1.0)	105.8 ± 1.8(6.6)
remove row Duration to remove a row. (with 5 warmup iterations).	29.8 ± 0.7(1.0)	37.8 ± 1.5 (1.3)	47.1 ± 3.0 (1.6)
creat many rows Duration to remove a row. (with 5 warmup iteration)	2,313.9 ± 49.0(1.4)	1.945.2 ± 24.4(1.1)	1,693.9 ± 70.1 (1.0)
append rows to large table Duration for adding 1000 rows on a table of 10.000 rows	296.5 ± 4.6(1.2)	267.8 ± 5.5(1.1)	243.3 ± 6.3
clear rows Duration to cleare the table filled with 10.000 rows.	174.6 ± 2.5(1.0)	175.3 ± 1.6(1.0)	263.9 ± 3.0 (1.5)
slowdown geomertic mean	1.08	1.0	1.54

Code

```jsx
import React, { useState } from 'react'

export default () => {
  const [a, setA] = useState(1)
  const [b, setB] = useState(2)

  function handleA(event) {
    setA(+event.target.value)
  }

  function handleB(event) {
    setB(+event.target.value)
  }

  return (
    <div>
      <input type="number" value={a} onChange={handleA}/>
      <input type="number" value={b} onChange={handleB}/>

      <p>{a} + {b} = {a + b}</p>
    </div>
  )
}
```

```vue
<template>
 <div>
   <input type="number" v-model.number="a">
   <input type="number" v-model.number="b">

   <p>{{a}} + {{b}} = {{a + b}}</p>
 </div>
</template>

<script>
 export default {
   data () {
     return {
       a: 1,
       b: 2
     }
   }
 }
</script>
```

```svelte
<script>
  let a = 1
  let b = 2
</script>

<input type="number" bind:value={a}>
<input type="number" bind:value={b}>

<p>{a} + {b} = {a + b}</p>
```

Key differences

- Compiler centered
- No VirtualDOM overhead
- Simpler, native reactivity
- Small bundles, faster app code
- Write less code

Examples

```
document.querySelector('input').addEventListener('input',
function(event) {
  span.textContent = `Hello ${this.value}`;
});
```

```
<span>Hello, {{ name }}</span>
<input @input="name = $event.target.value">
```

More examples

```
<script>
    export let foo;

    // Values that are passed in as props
    // are immediately available
    console.log({ foo });
</script>

<script>
    let count = 0;

    function handleClick() {
        // calling this function will trigger an
        // update if the markup references `count`
        count = count + 1;
    }
</script>
```

And more examples

```
<script>
  export let name;

  function toggleName() {
    if (name === "world") {
      name = "Svelte";
    } else {
      name = "world";
    }
  }
</script>

<main>
  <h1>Hello {name}!</h1>
  <button on:click="{toggleName}">Toggle name</button>
  <p>
    Visit the <a href="https://learn.svelte.dev/">Svelte tutorial</a> to learn
    how to build Svelte apps.
  </p>
</main>
```

Logic and Loops

```
<div>
  {#if counter > 0}
    More than zero
  {:elseif counter < 0}
    Less than zero
  {:else}
    Zero
  {/if}
</div>

<ul>
  <!-- Destructure loop variables -->
  {#each developers as {name, location}, i}
    <li>{i} - {name}, {location}</li>
  {:else}
    <li>No developers!</li>
  {/each}
</ul>
```

Async

```
<script>
  let promise = fetch('https://reqres.in/api/users')
    .then(res => res.json())
    .then(res => res.data);
</script>

<div>
  {#await promise}
    Fetching...
  {:then users}
    {#each users as {name}} {name} {/each}
  {:catch error}
    <p>Couldn't fetch users: {error}</p>
  {/await}
</div>
```

Drawback

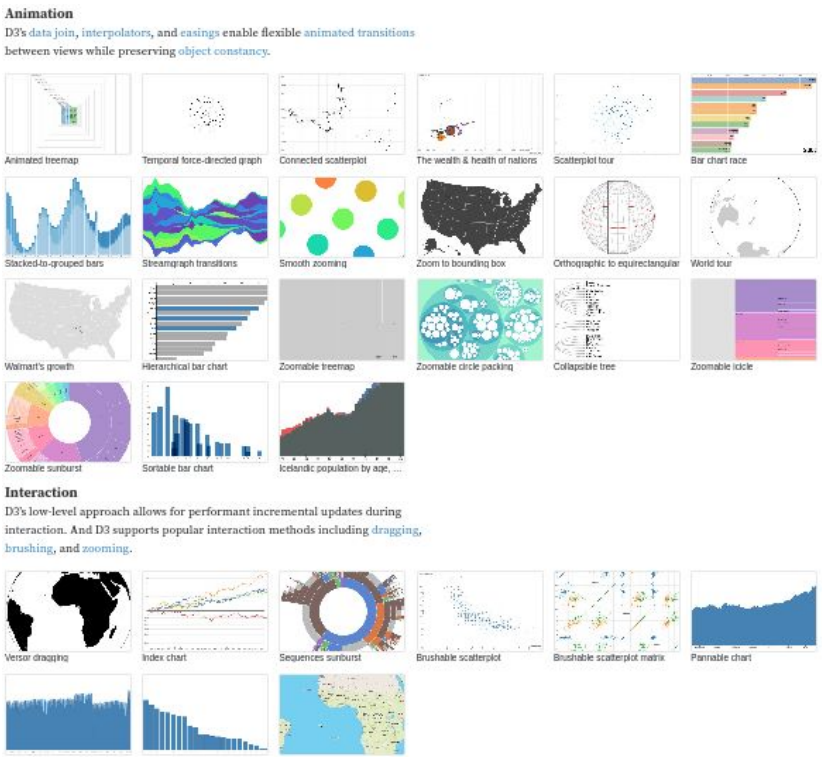
- Lack of community
- Only one Component is allowed per (.svelte) file
- Only one script tag is allowed per component (and file)
- Svelte components cannot (easily) be created via pure JS

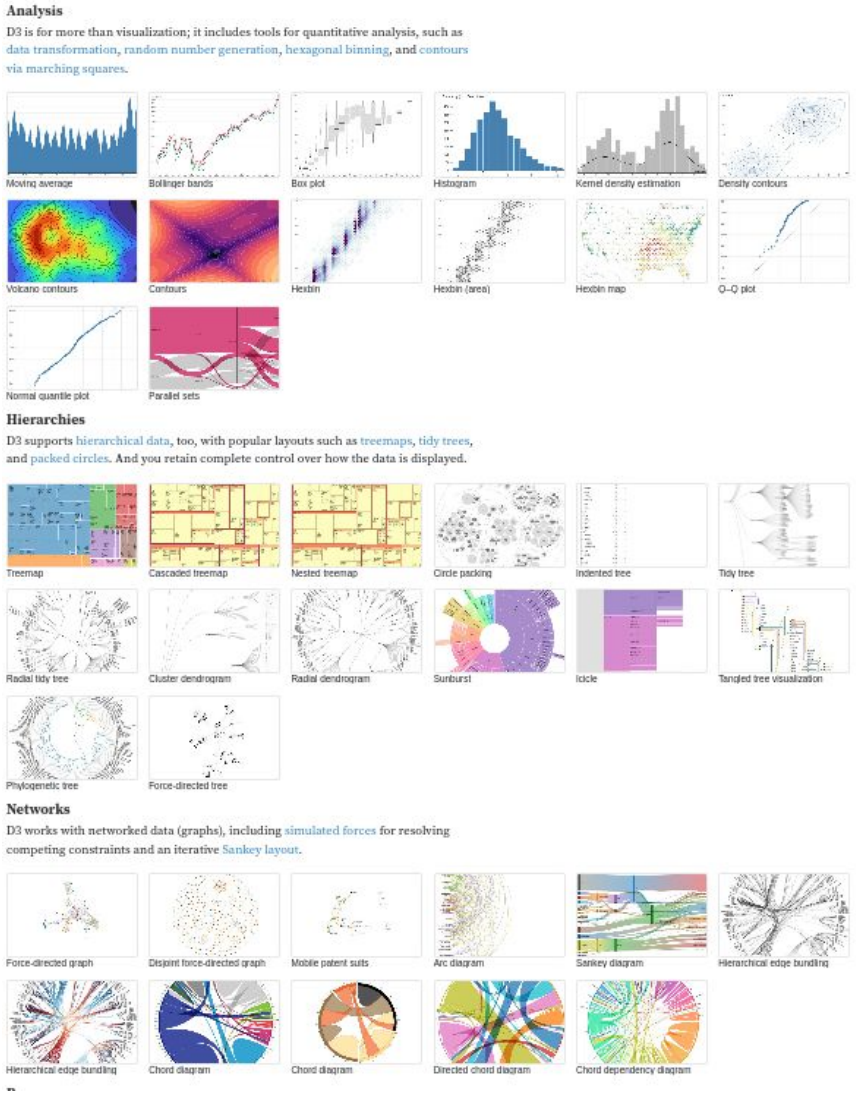
D3

About D3.js

- D3 stants for Data-Driven Document
- A JavaScript library for manipulating documents based on data
  - by Mike Bostock
- Dynamic, interactive data visualizations
- Nicely integrated with HTML5
- Supported by major browsers

D3 examples


Animation
D3's data join, interpolators, and easings enable flexible animated transitions between views while preserving object constancy.
Animated treemap
Temporal force-directed graph
Connected scatterplot
The wealth & health of nations
Scatterplot tour
Bar chart race
Stacked-to-grouped bars
Streamgraph transitions
Smooth zooming
Zoom to bounding box
Orthographic to equirectangular
World tour
Walmart's growth
Hierarchical bar chart
Zoomable treemap
Zoomable circle packing
Collapsible tree
Zoomable icicle
Zoomable sunburst
Sortable bar chart
Icelandic population by age, ...
Interaction
D3's low-level approach allows for performant incremental updates during interaction. And D3 supports popular interaction methods including dragging, brushing, and zooming.
Versor dragging
Index chart
Sequences sunburst
Brushable scatterplot
Brushable scatterplot matrix
Pannable chart


Analysis
D3 is for more than visualization; it includes tools for quantitative analysis, such as data transformation, random number generation, hexagonal binning, and contours via marching squares.
Moving average
Bollinger bands
Box plot
Histogram
Kernel density estimation
Density contours
Volcano contours
Contours
Hexbin
Hexbin (area)
Hexbin map
Q–Q plot
Normal quantile plot
Parallel sets
Hierarchies
D3 supports hierarchical data, too, with popular layouts such as treemaps, tidy trees, and packed circles. And you retain complete control over how the data is displayed.
Treemap
Cascaded treemap
Nested treemap
Circle packing
Indented tree
Tidy tree
Radial tidy tree
Cluster dendrogram
Radial dendrogram
Sunburst
Icicle
Tangled tree visualization
Phylogenetic tree
Force-directed tree
Networks
D3 works with networked data (graphs), including simulated forces for resolving competing constraints and an iterative Sankey layout.
Force-directed graph
Disjoint force-directed graph
Mobile patent suits
Arc diagram
Sankey diagram
Hierarchical edge bundling
Hierarchical edge bundling
Chord diagram
Chord diagram
Directed chord diagram
Chord dependency diagram

Running D3

- What does D3 require to run?
  - A web browser
  - D3 source code
  - Valid HTML document
  - Server
- D3 source code can be added to any HTML file by including:

  <script src="https://d3js.org/d3.v7.min.js"></script>

D3 advantages

- Provide a way to map data to documents
- General purpose visualization library
- Handle data transformation
- Provide basic math and layout algorithms
- Extremely fast
- Encourages code reusability
- Supports large datasets and provides an easy way of loading and transforming data

D3 disadvantages

- Backward compatible
- Rendering things
- Abstracting basic graphic primitives

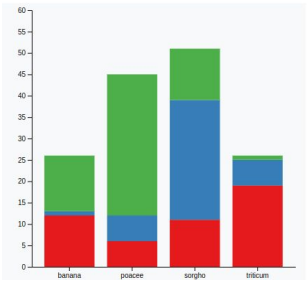
Key Features of D3.js

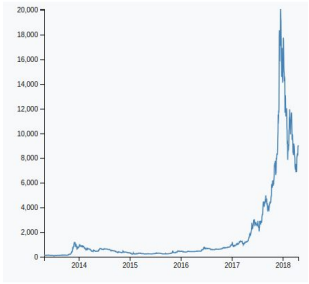
- Data Binding
  - Seamlessly connects data points with visual elements, enabling dynamic updates as the data changes.
- Scalability
  - Scales efficiently to handle large datasets, allowing you to visualize complex information without performance bottlenecks.
- Customization
  - Offers fine-grained control over every aspect of your visualization, from colors and shapes to animations and interactions.
- Interactivity
  - Lets you make your visualizations interactive, allowing users to explore and filter data through elements like hover effects and drill-down capabilities.
- Open Source
  - Open-source project with a vibrant community, providing continuous updates, extensive documentation, and readily available support.

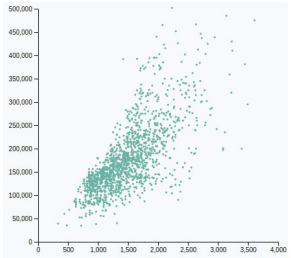
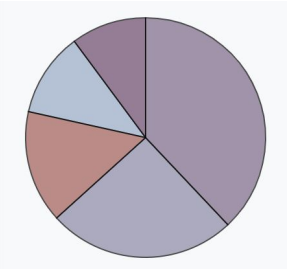
Basic Building Blocks of D3.js

- Selections
  - Identify and manipulate specific elements within the document, allowing you to target and modify visual components based on data.
- Data Joins
  - Link data points with visual elements, enabling the creation of dynamic visualizations that update as the data changes.
- DOM Manipulation
  - Create and modify SVG elements, giving you precise control over the visual representation of your data.

Chart Types

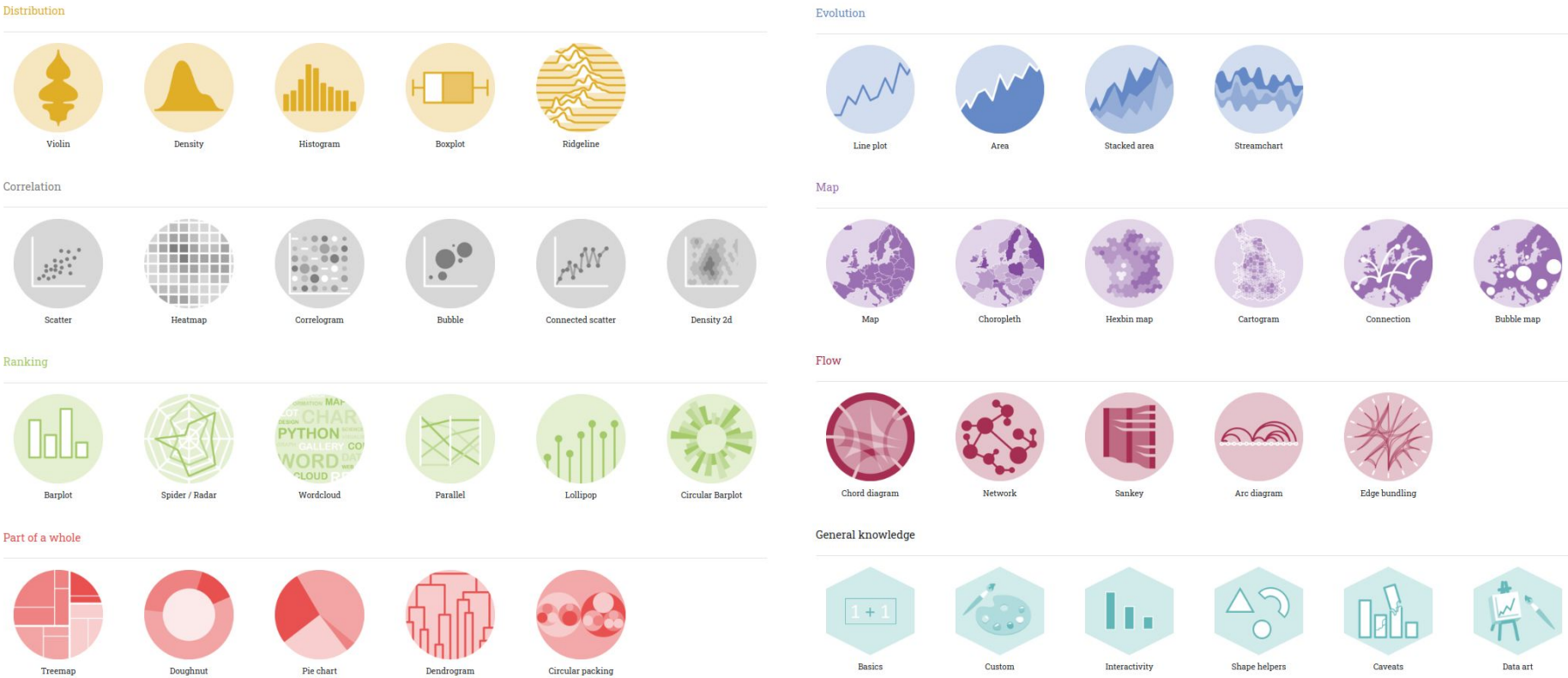




- Bar Charts
  - A versatile option for comparing data points across categories, often used for representing nominal or ordinal data.
- Line Charts
  - Ideal for showcasing trends and patterns over time, effectively visualizing continuous data changes.
- Scatter Plots
  - Useful for revealing relationships between two or more variables, enabling the identification of correlations and outliers.
- Pie Charts
  - Effective for displaying proportional data, highlighting the contribution of each category to the overall value.
- Network Graphs
  - Perfect for visualizing relationships between entities, such as social networks or physical connections.

Chart Types


Distribution
Violin
Density
Histogram
Boxplot
Ridgeline
Correlation
Scatter
Heatmap
Correlogram
Bubble
Connected scatter
Density 2d
Ranking
Barplot
Spider / Radar
Wordcloud
Parallel
Lollipop
Circular Barplot
Part of a whole
Treemap
Doughnut
Pie chart
Dendrogram
Circular packing
Evolution
Line plot
Area
Stacked area
Streamchart
Map
Map
Choropleth
Hexbin map
Cartogram
Connection
Bubble map
Flow
Chord diagram
Network
Sankey
Arc diagram
Edge bundling
General knowledge
1 + 1
Basics
Custom
Interactivity
Shape helpers
Caveats
Data art

Interactive Visualizations

- Hover Effects
  - Highlight specific data points or regions upon hovering with the mouse, providing additional context and details.
- Tooltips
  - Display informative pop-up messages when hovering over elements, enriching the user experience with additional data insights.
- Filtering
  - Allow users to dynamically filter the data displayed based on specific criteria, enabling focused exploration.
- Zooming and Panning
  - Facilitate navigation through large datasets or detailed visualizations by enabling zooming and panning functionalities.

Use D3 to create a new DOM element

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>D3 Test</title>
    <script type="text/javascript" src="d3/d3.v3.js"></script>
</head>
<body>
    <script type="text/javascript">
        d3.select("body").append("p").text("New paragraph!");
    </script>
    <p>Original paragraph</p>
</body>
</html>
```

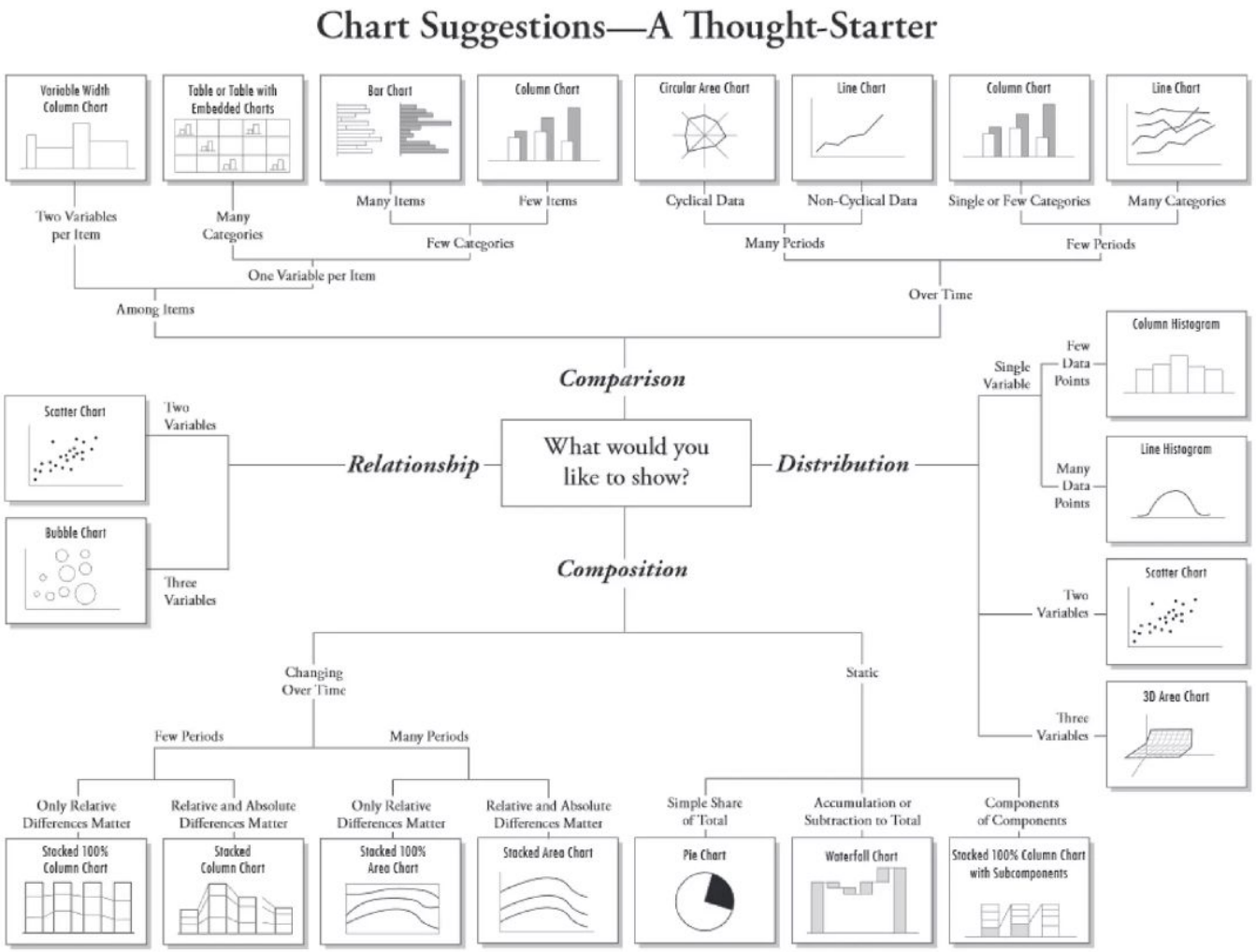
Some features

- Select element
  - `d3.select(), d3.selectAll()`
- Binding Data
  - With D3, you can bind data values to elements in the DOM
    - ```
      var dataset = [5, 10, 15, 20, 25];
      d3.select(“body”).selectAll(“p”) .data(dataset)
      .enter() .append(“p”) .text(function(d) { return(d)
      });
      ```
- D3 Scales
  - ```
    var scale = d3.scale.linear().domain([100,
    500]).range([10, 350]);
    ```
- Data loading
  - Supported formats: csv, html, json, text, tsv, xhr, xml

Modify elements

METHOD	USAGE	EXAMPLE
`.attr()`	Update selected element attribute	`d3.select("p").attr("name", "fred")`
`..classed()`	Assigns or unassigns the specified CSS class names on the selected elements	`d3.select("p").classed("radio", true);`
`.style()`	Updates the style property	`d3.select("p").style("color", "blue");`
`.property()`	Used to set an element property	`d3.select('input').property('value', 'hello world')`
`.text()`	Updates selected element text content	`d3.select('h1').text('Learning d3.js')`
`.html()`	Sets the inner HTML to the specified value on all selected elements	`d3.select('div').html('h1>learning d3.js</h1>')`
`.append()`	Appends a new element as the last child of the selected element	`d3.select("div").append("p")`
`.insert()`	Works the same as the .append() method, except you can specify another element to insert before	`d3.select("div").insert("p", "h1")`
`.remove()`	Removes selected element from the DOM	`d3.select("div").remove("p")`

Charts



Source: storytellingwithdata.com

Anime.js

Anime.js

- A JavaScript Animation Library
  - Empowers web developers and designers to create captivating animations with ease and efficiency.
- Lightweight and Performant
  - Boasting a small footprint and optimized code, smooth animations without compromising page performance.
- Intuitive and Flexible API
  - User-friendly API that's approachable for beginners and powerful enough for seasoned developers.
- Cross-browser Compatibility
  - Enjoy seamless animation across various browsers, reaching a wider audience without compatibility concerns.
- 2016 by Julian Garnier

Key Features

- Keyframe Animations
  - Define animation milestones for smooth transitions and complex movements, creating dynamic and eye-catching visuals.
- Timing Functions
  - Control the pacing and rhythm of your animations with a variety of timing functions, ranging from linear to elastic and custom options.
- Easing Options
  - Fine-tune the animation's progression with easing functions, adding natural-looking acceleration and deceleration for a polished feel.
- Transformations
  - Apply various transformations to animated elements, including scaling, rotation, translation, and skewing, for limitless creative possibilities.
- SVG and DOM Animations
  - Seamlessly animate both Scalable Vector Graphics (SVG) and Document Object Model (DOM) elements, offering flexibility for diverse animation projects.

Creating Basic Animations

- Targeting Elements
  - Select the element you want to animate using CSS selectors or DOM methods.
- Defining Properties
  - Specify the animation properties you want to change, such as position, opacity, or color.
- Setting Duration and Timing
  - Define the animation duration and choose a suitable timing function for the desired effect.
- Starting the Animation
  - Trigger the animation using various methods like callbacks, promises, or event listeners.

Main features

- **Targets** – this includes a reference to the element(s) we want to animate. It could be a CSS selector (div, #square, .rectangle), DOM node or node list, or plain JavaScript object. There is also an option to use a mix of the above in an array.
- **Properties** – this includes all properties and attributes that can be animated when dealing with CSS, JavaScript objects, DOM, and SVG.
- **Property Parameters** – this includes property-related parameters like duration, delay, easing, etc.
- **Animation Parameters** – this includes animation-related parameters like direction, loop, etc.

Import and example

- npm install animejs
- <script src="path/to/anime.min.js"></script>

```
let animation = anime({
 targets: ‘div’,
 // Properties
 translateX: 100,
 borderRadius: 50,
 // Property Parameters
 duration: 2000,
 easing: ‘linear’,
 // Animation Parameters
 direction: ‘alternate’
});
```

Specifying Target Elements

- CSS Selectors

```
var blue = anime({
    targets: '.blue',
    translateY: 200
});
```

- DOM Node

```
var special = anime({
  targets: document.getElementById('special'),
  translateY: 200
});

var multipleAnimations = anime({
  targets: [document.querySelectorAll('.blue'), '.red, #special'],
  translateY: 250
});
```

Effects

- Blurs and Shadows
  - Add depth and dimension to your animations with blur and shadow effects, creating a more realistic and visually appealing experience.
- Color Manipulation
  - Animate color changes seamlessly, transitioning between hues or incorporating gradients for dynamic and eye-catching visuals.
- Animation Trails
  - Leave trails behind moving elements, emphasizing their direction and speed, adding a touch of flair and dynamism.
- Combining Effects: Layer multiple

Example

- HTML

```html
<div class="circle"></div>
```

- CSS

```css
.circle {
  height: 50px;
  width: 50px;
  border-radius: 50%;
  background-color: red;
}
```

- JS

```js
anime({
  targets: '.circle',
  translateX: 250
})
```

Example

```
<!DOCTYPE html>
<html>
    <head>
        <title>First animation of animejs</title>
        <script src=
"https://cdnjs.cloudflare.com/ajax/libs/animejs/3.2.0/anime.min.js">
        </script>
    </head>
    <body>
        <div style="background: blue;
                    height: 40px;
                    width: 40px;">
        </div>
        <script>
            let animation = anime({
                targets: "div",
                translateX: 150,
                height: "80px",
                width: "80px",
                duration: 2000,
                easing: "linear",
                direction: "alternate",
                loop: true,
            });
        </script>
    </body>
</html>
```

Resources

- https://react.dev/
- https://svelte.dev/
- https://d3js.org/
- https://animejs.com/
- https://www.w3schools.com/
- https://www.freecodecamp.org/