Push this to the note page under d3js

Scale linear Episode 2

<https://www.youtube.com/watch?v=5RU15VWNwXo&t=6s>

A person with a mustache

AI-generated content may be incorrect.

The real world value **- input domain** the house IRL

A house with a brick path and flowers

AI-generated content may be incorrect.

The minuature version is the **output range of the scale**

A person holding a small piece of food

AI-generated content may be incorrect.

We scale up or down – like a stair case

scaleUtc() is a temporal scale – map a domain of time to a range of pixels

like dates to pixels or temperature to a color

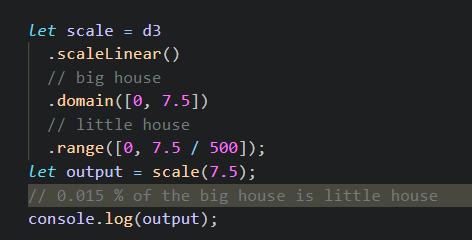
A screen shot of a computer

AI-generated content may be incorrect.

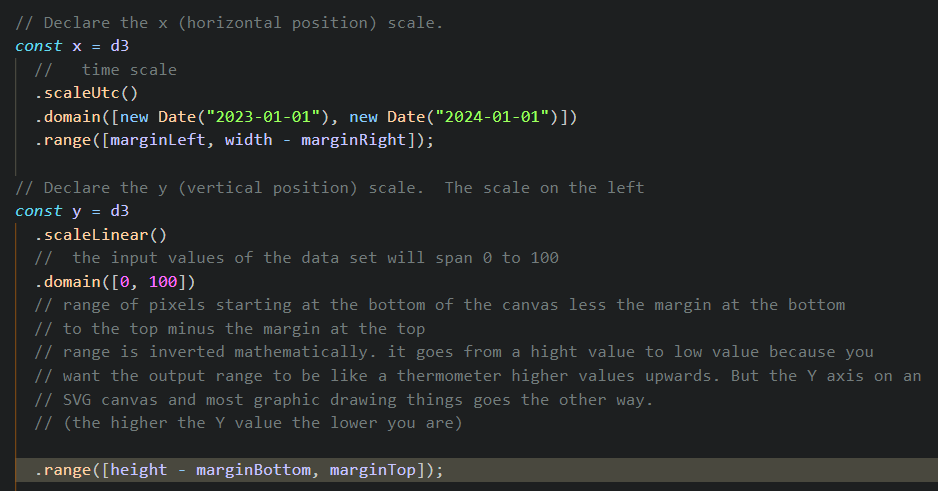
scaleLinear() that returns a scaling function that can be called . the default state is 1:1 ( it is and identity function <https://en.wikipedia.org/wiki/Identity_function> ) Here just calling and not passing in a domain or range

A computer screen with text and images

AI-generated content may be incorrect.



D3 is a library that takes data and puts it through these transformations to allow us to take a string of numbers and make it into s set of coordinates or values that make sense graphically speaking. That is percent to pixels , temperature to color. That kind of thing.



Part 4

<https://www.youtube.com/watch?v=LOaqG--VWas>

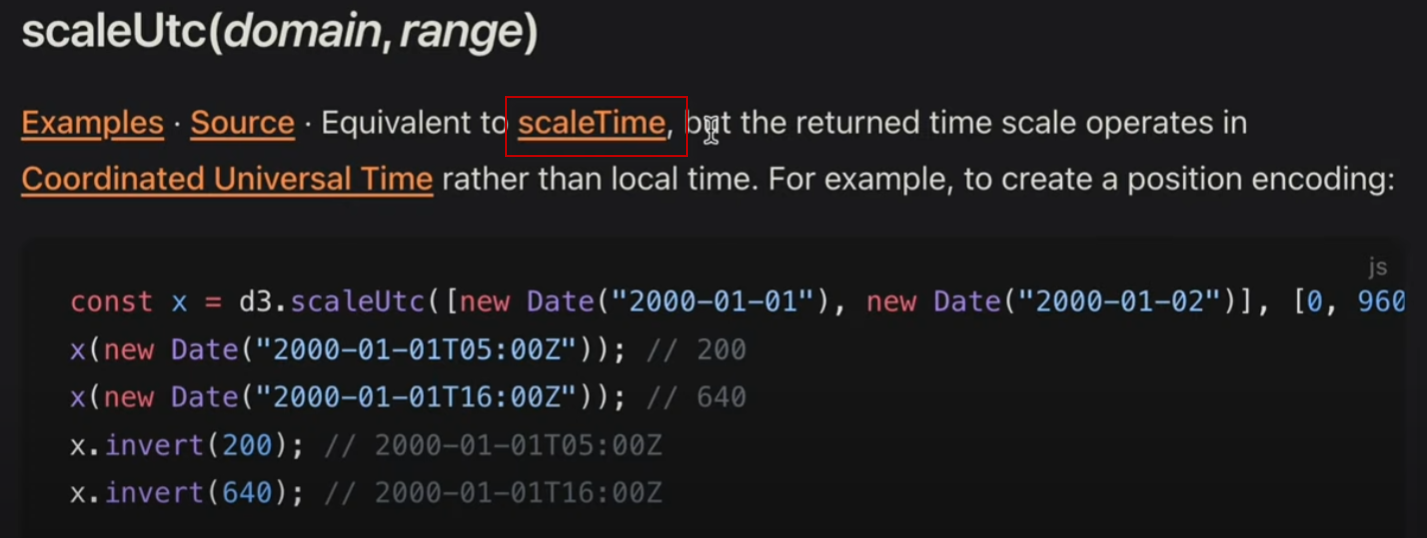
Interpolation – finding values between known points

<https://d3js.org/d3-scale/time>

**Time scales**[​](https://d3js.org/d3-scale/time#time-scales)

Time scales are a variant of [linear scales](https://d3js.org/d3-scale/linear) that have a temporal domain: domain values are coerced to [dates](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Date) rather than numbers, and invert likewise returns a date. Time scales implement [ticks](https://d3js.org/d3-scale/time#time_ticks) based on [calendar intervals](https://d3js.org/d3-time), taking the pain out of generating axes for temporal domains.



scaleUtc(domain,range)

**scaleTime(*domain*, *range*)**[​](https://d3js.org/d3-scale/time#scaleTime)

[Examples](https://observablehq.com/@d3/d3-scaletime) · [Source](https://github.com/d3/d3-scale/blob/main/src/time.js) · Constructs a new time scale with the specified [domain](https://d3js.org/d3-scale/linear#linear_domain) and [range](https://d3js.org/d3-scale/linear#linear_range), the [default](https://d3js.org/d3-interpolate/value#interpolate) [interpolator](https://d3js.org/d3-scale/linear#linear_interpolate) and [clamping](https://d3js.org/d3-scale/linear#linear_clamp) disabled. For example, to create a position encoding:

js

const x = d3.scaleTime([new Date(2000, 0, 1), new Date(2000, 0, 2)], [0, 960]);

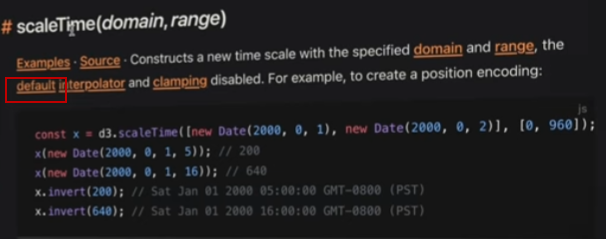
x(new Date(2000, 0, 1, 5)); // 200

x(new Date(2000, 0, 1, 16)); // 640

x.invert(200); // Sat Jan 01 2000 05:00:00 GMT-0800 (PST)

x.invert(640); // Sat Jan 01 2000 16:00:00 GMT-0800 (PST)

If *domain* is not specified, it defaults to [2000-01-01, 2000-01-02] in local time. If *range* is not specified, it defaults to [0, 1].



**interpolate(*a*, *b*)**[​](https://d3js.org/d3-interpolate/value#interpolate)

[Examples](https://observablehq.com/@d3/d3-interpolate) · [Source](https://github.com/d3/d3-interpolate/blob/main/src/value.js) · Returns an interpolator between the two arbitrary values *a* and *b*.

js

d3.interpolate("red", "blue")(0.5) // "rgb(128, 0, 128)"

The interpolator implementation is based on the type of the end value *b*, using the following algorithm:

1. If *b* is null, undefined or a boolean, use the constant *b*.
2. If *b* is a number, use [interpolateNumber](https://d3js.org/d3-interpolate/value#interpolateNumber).
3. If *b* is a [color](https://d3js.org/d3-color#color) or a string coercible to a color, use [interpolateRgb](https://d3js.org/d3-interpolate/color#interpolateRgb).
4. If *b* is a [date](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date), use [interpolateDate](https://d3js.org/d3-interpolate/value#interpolateDate).
5. If *b* is a string, use [interpolateString](https://d3js.org/d3-interpolate/value#interpolateString).
6. If *b* is a [typed array](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/TypedArray) of numbers, use [interpolateNumberArray](https://d3js.org/d3-interpolate/value#interpolateNumberArray).
7. If *b* is a generic [array](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/isArray), use [interpolateArray](https://d3js.org/d3-interpolate/value#interpolateArray).
8. If *b* is coercible to a number, use [interpolateNumber](https://d3js.org/d3-interpolate/value#interpolateNumber).
9. Use [interpolateObject](https://d3js.org/d3-interpolate/value#interpolateObject).

Based on the chosen interpolator, *a* is coerced to the suitable corresponding type.

**interpolateNumber(*a*, *b*)**[​](https://d3js.org/d3-interpolate/value#interpolateNumber)

[Examples](https://observablehq.com/@d3/d3-interpolatenumber) · [Source](https://github.com/d3/d3-interpolate/blob/main/src/number.js) · Returns an interpolator between the two numbers *a* and *b*.

js

d3.interpolateNumber(20, 620)(0.8) // 500

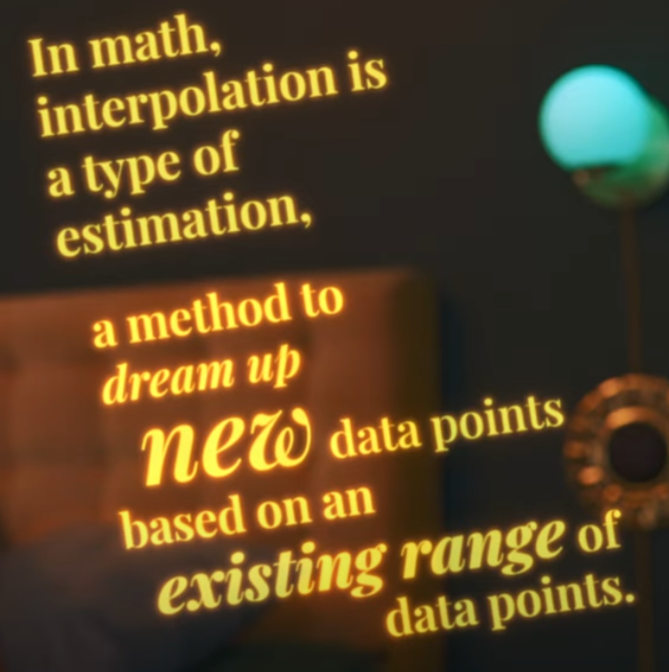
The returned interpolator is equivalent to:

js

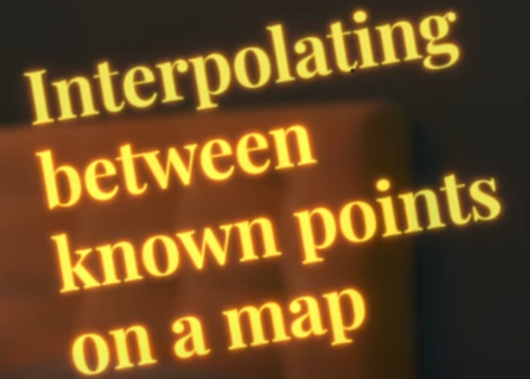
function interpolator(t) {

return a \* (1 - t) + b \* t;

}



We interpolate a thrown balls trajectory based on a few data points, then catch it – our brain is constructing a 3d arch to determine where the ball will be in the future and coordinating hands to intercept it at the right moment

.A yellow circle with a black background

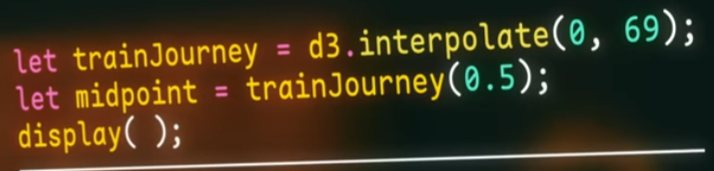
AI-generated content may be incorrect.

Start and end range of data points

Episode 4L

https://www.youtube.com/watch?v=6Vn1TWyOsYo

**d3.interpolate (Number) | Dirty D3 - Episode 4**



scaleTime, per default, uses the default interpolator, d3.interpolate behind the scenes.

 scaleLinear uses d3.interpolate under the hood (

scaling functions are like interpolaters with more bells and whistles

A black and white image of a bug and a person in a suit

AI-generated content may be incorrect.

A yellow text on a black background

AI-generated content may be incorrect.

Episode 5 - TBD

Color interpolation

1,601 views Premiered May 26, 2025 [#dataanalytics](https://www.youtube.com/hashtag/dataanalytics) [#JavaScript](https://www.youtube.com/hashtag/javascript) [#DataVisualization](https://www.youtube.com/hashtag/datavisualization)

In this episode of Dirty D3, MPJ dives deeper into the world of interpolation by exploring d3.interpolate and d3.interpolateNumber - the function that powers much of D3's ability to smoothly transition between numerical values. Source code & Full transcript: [https://src.fff.dev/f3d3d4/s01/e04](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbHViT1BadVdwRk5NcC12VWxLZ0hhLU1uN2xMZ3xBQ3Jtc0tuR0hsSlZNaWRNellkVW1sYTNiZEZWamVWNTFNM2d3dW56SEk5dWNtWDJyZ1pTSUtWcHVFcnlVcnVMdUh6aG9KLTZkTF8td2tKWWtEUFl1ZG1Bb0ktUFdkZVNrSW1vdUhoWjVPOWN3LTRtYkxiRXFjRQ&q=https%3A%2F%2Fsrc.fff.dev%2Ff3d3d4%2Fs01%2Fe04&v=6Vn1TWyOsYo) Building on Episode 3's introduction to interpolation concepts, we explore: • Creating and using number interpolators with practical examples • The relationship between interpolators and D3 scales • The mysterious "t-value" parameter (and why mathematicians are feral) • How d3.interpolate connects to d3.scaleLinear • A real-world application using train commute times between Stockholm, Uppsala, and Upplands Väsby • The mind-blowing revelation about time dilation in... Swedish suburbs? This episode bridges the gap between the abstract concept of interpolation and concrete D3 mechanics, showing how interpolation functions as the engine behind D3's powerful scaling capabilities. 📚 RESOURCES: Source code & Full transcript: [https://src.fff.dev/f3d3d4/s01/e04](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbEtEU3JZS1Bna3NhOTEwcF9ia0RQdTRUTFV6UXxBQ3Jtc0tsbUxBUFB5a2JnaENWZlNFMXdkZkZkb2xpWUxlcHZ4ZHdZYXk3Wl9pUjV2ZHQtSW0yTjdtb1h2SnFuZ1ZoRU9oY04zamxGSkM0b2l3QzFLT1lQVkN3aVhuTTBVRG5rcU9EcnVjZlBIRHRxWC1zQVE1TQ&q=https%3A%2F%2Fsrc.fff.dev%2Ff3d3d4%2Fs01%2Fe04&v=6Vn1TWyOsYo) D3.js Documentation: [https://d3js.org/getting-started](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbUFBUW4zdGkzdXJNNl9JbzB4cy1mVUNFbUh0UXxBQ3Jtc0trVUlPbTZBVGtvOFVGSHhQaHJDWTFRQVktVXVacUZNNWc4WHNwN1BMNW13Mmxsckg0TFBnUGh4TFh4UFhyOTFiam5CNnROUUEwU29hWmpXZFFZVkFGdmFFcU8tbDdHa293ZGtzcU15NVpGQlp0ckFHaw&q=https%3A%2F%2Fd3js.org%2Fgetting-started&v=6Vn1TWyOsYo) D3 Interpolate Docs: [https://d3js.org/d3-interpolate/value](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqazkybzRHaUVzMXZwTnNHNGVHcmcxeWk2VUg0d3xBQ3Jtc0tsMHlkUUNDNVlfSGh3WWh3WmdMbV9LSDdwSTRTcDY0TUtjUTZtVGtaUkZuRTRwQ0hWdlZPRDY5VU1PdnBVcFkzQS1aYWRCWWhERi1tY3hpTS1na1hSWGZPUkZreUowRzBlaVhOMm4tTWduY3V2dVZOSQ&q=https%3A%2F%2Fd3js.org%2Fd3-interpolate%2Fvalue&v=6Vn1TWyOsYo) D3 Scale Linear: [https://d3js.org/d3-scale/linear](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbXNyVnBCbmp3N2xYUGM4RWVFSDVzWllDRk9FZ3xBQ3Jtc0tsaFJFcTU3aF9Wb0Y0ZWlzZnVUMllSalNIaFBBU2xHa0xydW1CdnBHT0h6LVNuNlRnNVlURFlUWHl4UUtqMnlRT3AxOUJueHJneHozQmRENEcyWUpJODRENUZGV1BONEJPRE1sNUdkczFlZlNqcFdSWQ&q=https%3A%2F%2Fd3js.org%2Fd3-scale%2Flinear&v=6Vn1TWyOsYo) 👉 Follow along with our journey into the heart of D3.js as we continue to unravel the library's core concepts one rabbithole at a time! [#D3js](https://www.youtube.com/hashtag/d3js) [#DataVisualization](https://www.youtube.com/hashtag/datavisualization) [#JavaScript](https://www.youtube.com/hashtag/javascript) [#WebDevelopment](https://www.youtube.com/hashtag/webdevelopment) [#DataScience](https://www.youtube.com/hashtag/datascience) [#datavisualization](https://www.youtube.com/hashtag/datavisualization) [#dataviz](https://www.youtube.com/hashtag/dataviz) [#d3js](https://www.youtube.com/hashtag/d3js) [#dataanalytics](https://www.youtube.com/hashtag/dataanalytics)

GitHub code <https://github.com/funfunfunction/dirty-d3-notebooks>