Rx Marble Design System

There's more to picture
Then meets the eye.
Hey hey my my.

Neil Young

The idea behind the MarbleDesignSystem

Marble diagrams serve a **method for** us to **visualize processes over time**.
This helps programmers and engineers to understand and design reactive processes.

The overall goal of the *MarbleDesignSystem* is to **provide a unified way** of **reading and creating** stream based diagrams,
in particular one specific type of it, the **marble diagrams**.

This guide explains all building blocks of the design system step by step and in detail.

In general we have some main rules that system follows:

- **Consistent**
- **Intuitive**
- **Easy**
- **Detailed**
- **Customizable**

Consistency

There are several things to follow if you try to create a standard. One of them if more **critical for a positive outcome** than everything else, **consistency.**

By working with a **standardized, reproducible approach** we managed to create a **consistent way** of drawing marble diagrams.
A set of rules developed over many many iterations, adopted and simplified to serve as a guideline and blueprint for **creating and using** these diagrams.

Intuitive

As programming with Rx is hard we made sure to keep it intuitive.
By **including** a lot of **people** into the process of the creation of this guide we collected a lot of **personal feedback** to improve the system.
To make sure we consider a **common way** interpretation we created several **public polls** we were collected and evaluated the general understanding.
This helped us to make our system **intuitive to understand**.

Easy

As mindset behind the system are several principles. One of them is “**Easy to adopt and create**”, which means we want to provide **a way for everybody** to **read and create** marble diagrams.

To achieve this we create all diagrams in either googleSlides oder Powerpoint. We believe this two options **enable** a big group of **people** to **edit and draw** these **diagrams**.

Detailed

Marble diagrams exist since a long time now. As there was no well thought standard out there and not all edge cases considered, people started to create their own solutions to visualize processes. These let to a variety of different ways of drawing these diagrams. Some of the better approaches were able to visualize more complex prozesses, but there is one essential thing which nobody considered yet, but which is most critical to understand processes based on Rx. The internal behavior of operators.

This system is not only providing a consistent, standardized way of drawing marble diagrams, but also offers a way to visualize the internals of operators. Of course based on the systems rules it selfe.

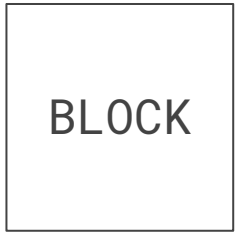
Customizable

TODO

Index

DESIGN TOKENS

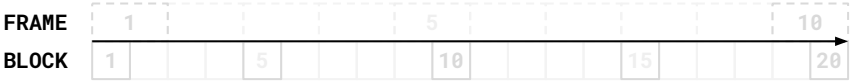
UNIT

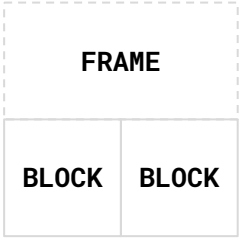


Base Unit:Block

Block Grid

1 Block = 1em





1 Frame = 2 Blocks

The unit for width and height in marble diagrams is called a block.
A block is a square with equal width and height.

Positioning of objects and text is also measured in blocks.

To have a convenient unit we use **em with the base of 1 blok**.

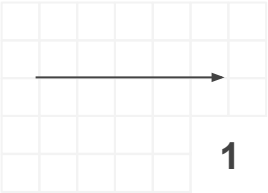
Sometimes time matters. The smallest unit of time is called frame.
A frame is a unit for time and expressed in width. **1 frame equals to the width of 2 blocks**.

FONT

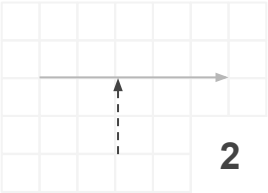
- FONT-FACE
- FONT-SIZE
- FONT-STYLE

Text is used to name things or give more information to a specific part of a diagram like observables, operators and events like subscribe or unsubscribe.
It is furthermore used in notifications to give more detailed information about their content.

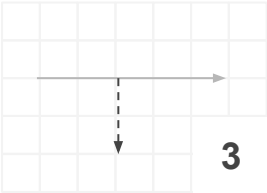
LINE



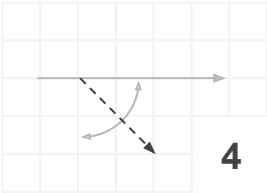
Style: Solid
Weight: 0.06em
Start: none
End: Filled Arrow



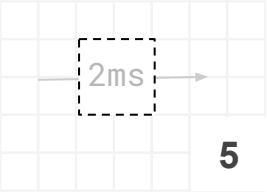
Style: Dashed
Weight: 0.06em
Start: Filled Arrow
End: none



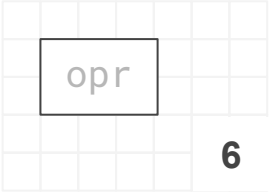
Style: Dashed
Weight: 0.06em
Start: none
End: Filled Arrow



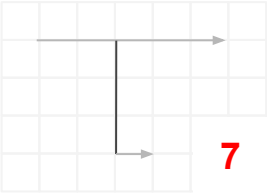
Style: Dashed
Weight: 0.06em
Start: none
End: Filled Arrow



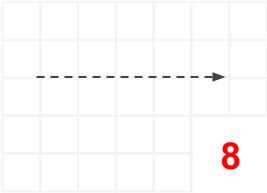
Style: Dashed
Weight: 0.06em
Start: none
End: none



Style: Solid
Weight: 0.06em
Start: none
End: none



Style: Solid
Weight: 0.06em
Start: none
End: none



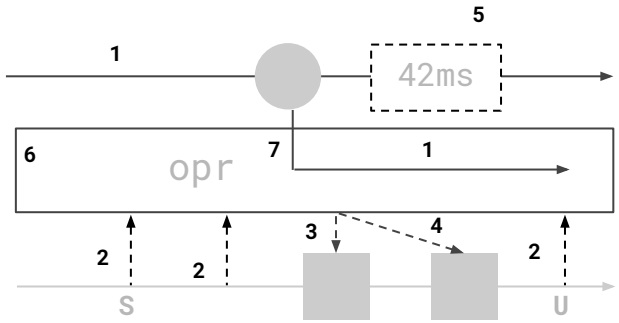
Style: Dashed
Weight: 0.06em
Start: none
End: Filled Arrow

Lines are here to symbolize time.
Time without a measurement as line or area as shown with **[1]** and **[5]**, a specific time period is shown under **[4]** or an interaction at a specific point in time ilke in **[2]** and **[3]**.

The interaction can be initialized from the consumer **[2]** to the producer **[3,6]**.

Lines from the consumer side **[2]** have in some cases description in form of the letters S, U and P.

Some lines from the producer side **[6]** can be angled. This is possible to the right side only. **[7]** **[8]**


5
1
42ms
6
opr
7
1
3
4
2
2
2
S
U

FONT-FACE

Sans-Serif
Font-Face

ABCDEFGHIJ
stuvwxyz01

Serif
Font-Face

ABCDEFGHIJ
stuvwxyz01

Monospace
Font-Face

ABCDEFGHIJ
stuvwxyz01

Proportional
Font-Face

ABCDEFGHIJ
stuvwxyz01

The design guide doesn’t limit the choice of font.

Only a few limitations are suggested.

Serif font-face and mono-space letter.

Serif Font-Face
Serif font-face has compared to the serif font-faces less visual noise.

Mono-space
Mono-space font has to be used in the description of observables, operators or in the text of notifications. This ensures that the with of text is directly proportional to the number of characters and therefore simplifies the positioning and alignment of text in diagrams.

FONT-SIZE

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01234567890()[]{}

1em (Big)

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01234567890!“§$%&/()=?

0.67em (Medium)

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01234567890!“§$%&/()=?

0.5em (Small)

The font-size is measured in em, a scalable unit for text. In this design system 1em is equal to 1 block, the unit for width and height.

There are 3 different sizes defined to describe marble diagrams. We will elaborate more on this later.

FONT-STYLE

ABCDEFGHIJKLMNOPQRSTUVWXYZ Normal
abcdefghijklmnopqrstuvwxyz
01234567890()[]{}

ABCDEFGHIJKLMNOPQRSTUVWXYZ Bold
abcdefghijklmnopqrstuvwxyz
01234567890!“§$%&/()=?

The font-styles are limited to 2 different styles, normal and bold.
We will elaborate more on this later.

COLOR

Line

Name: Line
Usage:
Line, border and text color

Background

Name: Background
Usage:
Background of operators or text

Inactive

Name: Inactive
Usage:
Inactive lines, shapes and text

Notification 1

Name:
notification 1
Usage:
notification color 1

Notification 2

Name:
Notification 2
Usage:
notification color 2

Notification 3

Name:
Notification 3
Usage:
notification color 3

Notification 4

Name:
notification 4
Usage:
notification color 4

All colors are are barrier free and used for specific things in the diagram.

There are **3 fixed colors** and **4 to n colors** that are **free to choose**.

For Text, lines, and arrows the **“line” color** is used.

The **“background” color** is not only used for the background but also for text to get an acceptable color ratio.

For **“notification” colors** you can use the defined colors 1 - 4 or any other color of your choice. All notification colors have the same meaning. There is no “danger” or “success” color.
You can check your pallet here.

The **“inactive” color** is used for anything that should be recognized as inactive.

COLOR

Notification 1	Notification 2	Notification 3	Notification 4
#FFCB21	#89C540	#F77C00	#6734BA
Notification 1	Notification 2	Notification 3	Notification 4
#D01C8B	#F1B6DA	#B8E186	#4DAC26
Notification 1	Notification 2	Notification 3	Notification 4
#36175E	#553285	#7B52AB	#9768D1

Colors serve the purpose to distinguish between notifications or observables.

Try to use as less colors as possible to keep the visual noise of a marble diagram low.

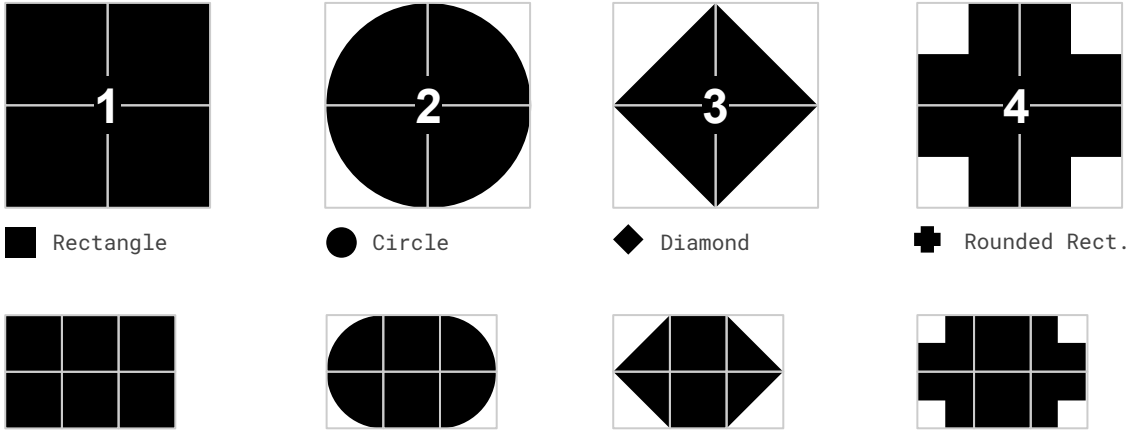
If you create your own color patterns keep in mind that you can use text inside shapes.

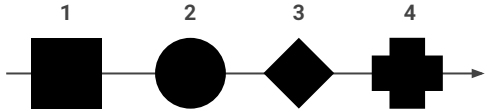
Also always check your palette including text with a color contrast tool like accessible-color-matrix

Example Palettes:
Palette 1
Palette 2
Palette 3

Links:
http://chromelens.xyz/
Contrast checker
Spectrum, ChromeLense
Color Palettes
ColorByCulture

SHAPE





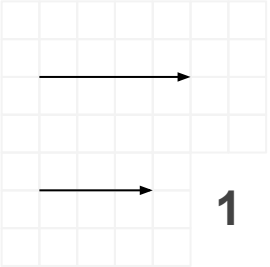
Shapes, same as colors help to distinguish between notifications. You can use the suggested set or create your own set of shapes.

Try to use as less different shapes as possible to keep the visual noise of a marble diagram low.

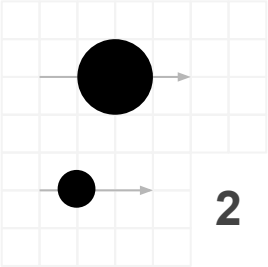
If you create your own shapes keep in mind that evers shape should be same height and width to keep the sizing and spacing as easy as possible.

Also consider that every shape should be able to have dynamic with. (*dynamic in number of blocks*)

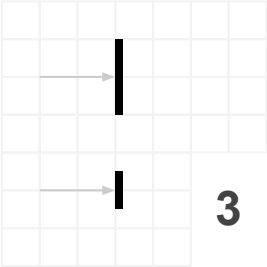
SIZE



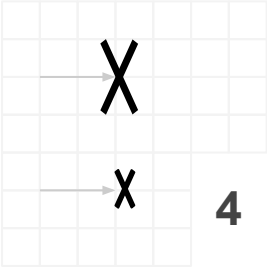
Time



Notifications



Completion



Error

There are 2 different sizes, **normal (2em)** and small **(1em)**.

Small size is only possible for this subset:
[1] Time
[2] Notification
[3] Completion
[4] Error

Small size is rarely used but can serve in some cases as a useful detail information. For example to sketch higher order observables or operators.

COMPONENTS

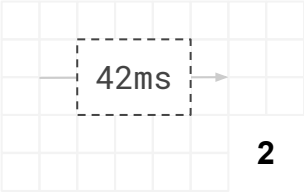
COMPONENTS

Components are the smallest entities in marble diagrams.
The different design tokens are applied on components. Diagrams are a composition of components.

COMPONENTS

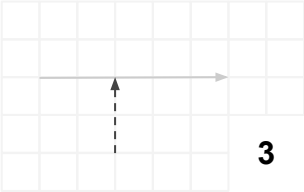
1

Time



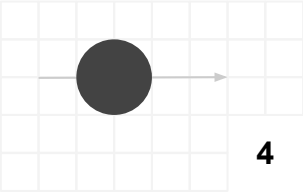
2

Time Span



3

Consumer Event
(GOOD NAME?)

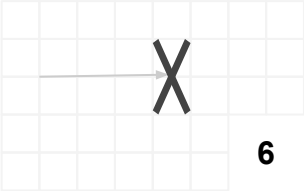


4

Notification

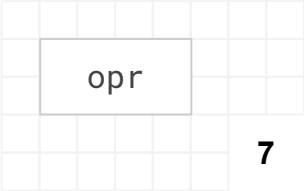
5

Completion



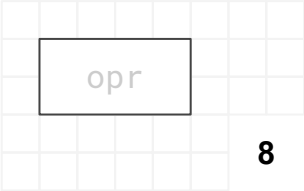
6

Error



7

Operator

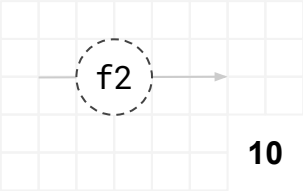


8

Operator Contex

9

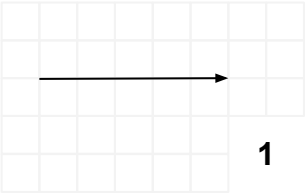
Operation



10

Time Point
REODRER

TIME



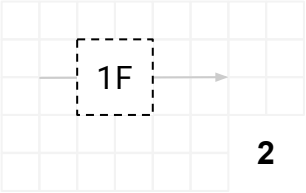
Time

Time in marble diagrams **is** represented as **an arrow going from left to right**.

The arrow at the same time is also a representation of an observable it selfe.

Time is measured in frames, a superset of blocks.
1 frame is 2 blocks

TIME SPAN



Object:
Shape: Rectangle

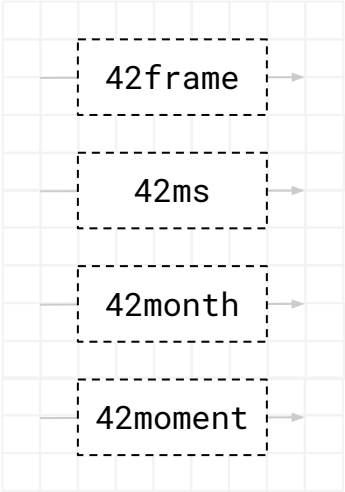
Time Span is here to **specify** a certain **duration of time**.

The description of the time span is placed inside the box and consists out of an **amount directly followed by a unit**. No space inbetween.

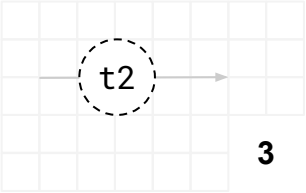
The amount is a number.
The unit, as singular, can be any known unit of time as well as the unit frame, which is equal to one block.

For time **units** you can **use shortcuts** as ms for milliseconds or s for seconds as well as full name i.e. month or year.

Of course **you can** also **use any** other **imaginary unit of time** like a moment ;-)



TIME POINT



Object:
Shape: Circle

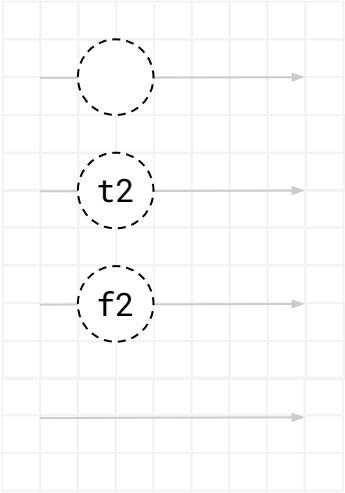
Time Span is here to **specify** a certain **duration of time**.

The description of the time span is placed inside the box and consists out of an **amount directly followed by a unit**. No space inbetween.

The amount is a number.
The unit, as singular, can be any known unit of time as well as the unit frame, which is equal to one block.

For time **units** you can **use shortcuts** as ms for milliseconds or s for seconds as well as full name i.e. month or year.

Of course **you can** also **use any** other **imaginary unit of time** like a moment ;-)



CONSUMER EVENT

3

Consumer Events

Consumer events are directed bottom up. It starts from the result observable and ends at the operator context, or the observable it selfe.

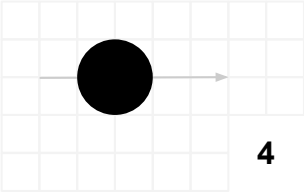
Consumer events can be **described over the legend** and an index number.

If you subscribe **[S]** to an observable you basically “start” the stream of possible notifications.

If you connect **[C]** to an observable…

If you unsubscribe **[U]** from an observable you basically “stop” the stream of possible notifications.

NOTIFICATIONS



4

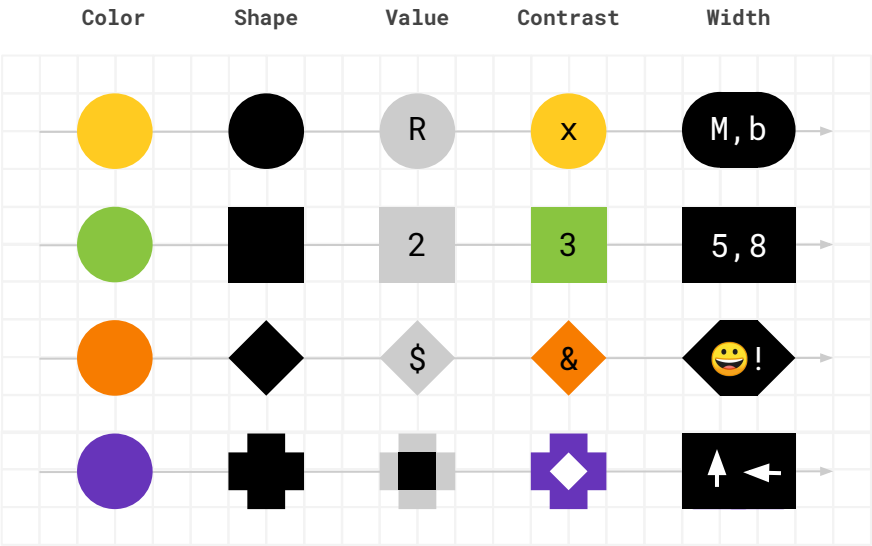
Notification

Notifications are the possible **content of observables**.
They ocurre over time and are a representation of any particular thing.

Notifications can have:
- **Color**
- **Shape**
- **Value** (proper contrast)
- **With** (full blocks)

Color and **value** combination should always have a proper contrast. Find a onlinetool here: [Contrast checker]
The **value color can be only black or white**.

With is here to provide enough space for more content than just 1 chars. Every shape can have a dynamic width.



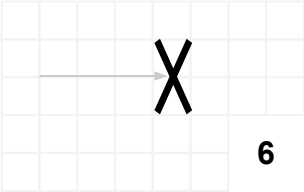
COMPLETION

5

Completion

Completion symbolizes one possible end of an observable.

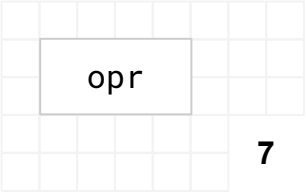
ERROR



Error symbolizes one possible end of an observable.

Error

OPERATOR



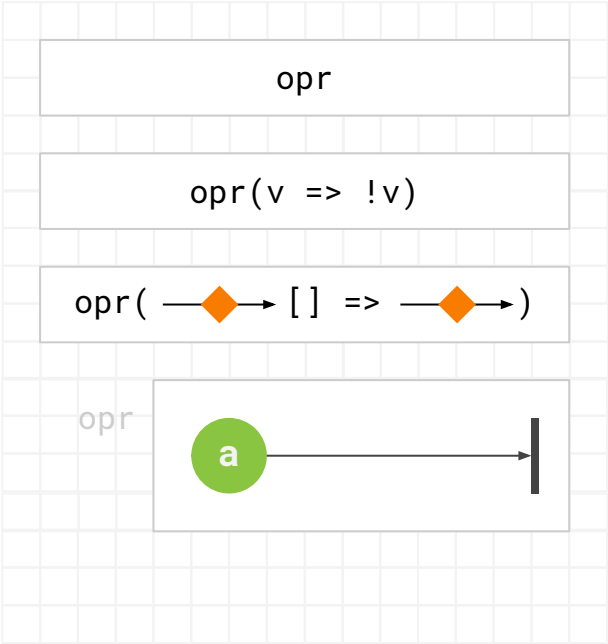
7

Operator

The **operator** symbolizes the logic which **interacts with** observables and their **notifications**.

Operators can contain any other component in small as well as normal. Also text.

In case the content contains components they can have a description on their left side.



Remove text in operator. All description is on the left side.

Cosider get rid of the mini icons inside the opr.

OPERATOR CONTEXT

8

Operator Contex

Operator Context

How to draw operator context?

Try to solve it with the legend

OPERATION

9

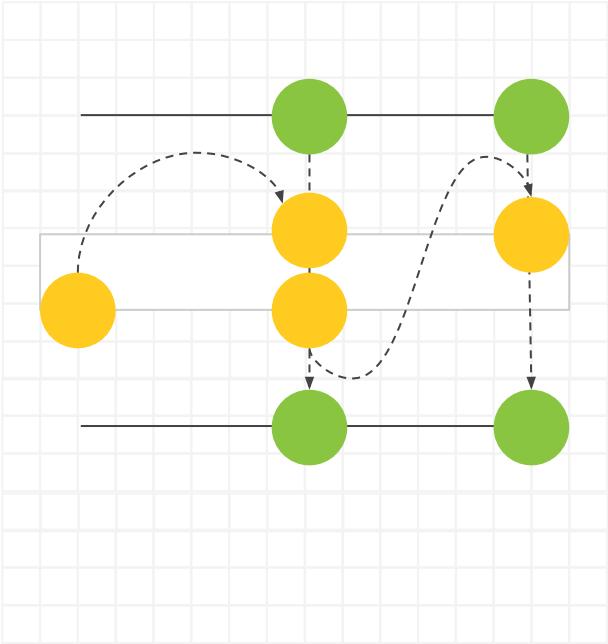
Operation

Operation is an event triggered from the consumer of an observable.

Operation symbolizes the interaction with the notifications and operators.

The operator start from notifications

Examples for inactive operations (stop at operator stop at output)

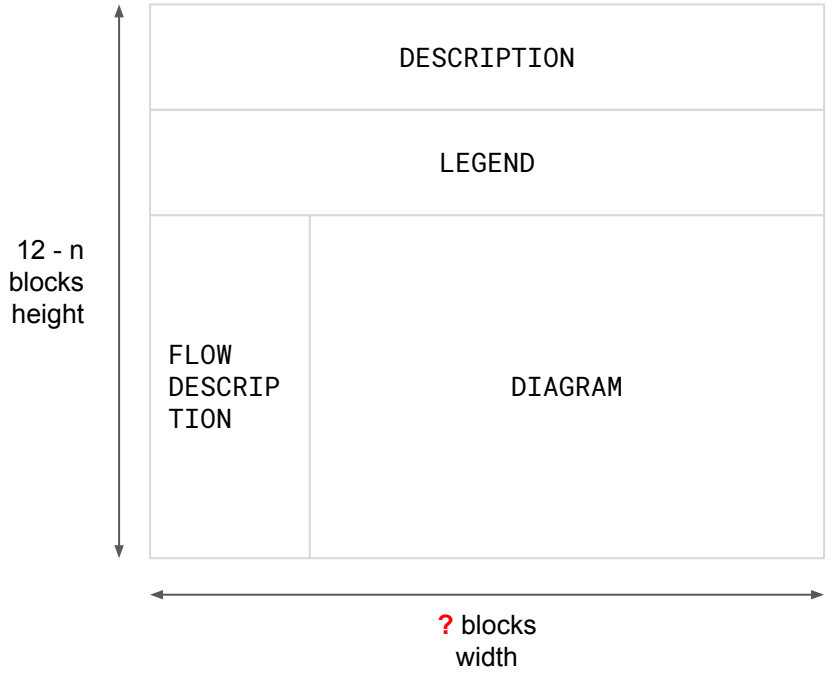


DIAGRAMS

SECTIONS

A diagram consists at least of the diagram section.

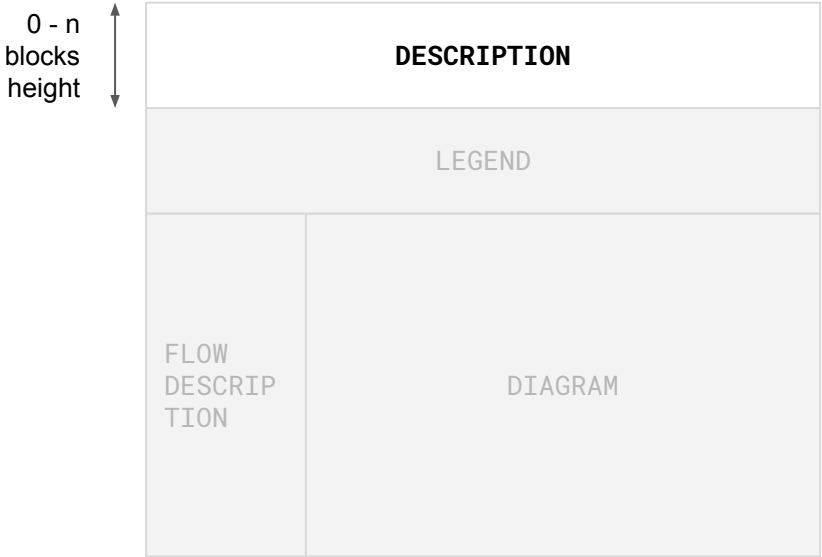
Description, legend as well as flow description are optional.



DESCRIPTION

The description section is the first section possible in a diagram. It can have **variable height**.

This section is optional as most of the time we want to have our textual description as text, not as image but it’s completely fine to include the description in the diagram.



LEGEND

Description can be anything text as well as any multimedia is allowed.
This is the only section without any restrictions.
Therefor it allows a variety of oportunities to describe the contnet
of the following diagram.

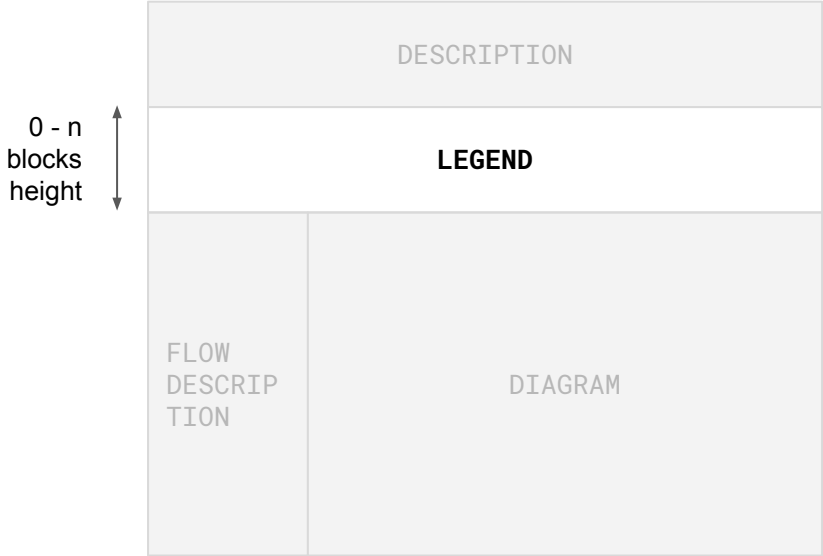
LEGEND

The legend section is the second possible section in a diagram. It can have **variable height**.

This section is optional too, but compared the the description section it is always part of the diagrams as we want to have the legend as part of the diagram it selfe.

Use **small font size for the legend**. For **font weight use normal for description** and **bold for** indexes, operator params, or content **references**.

The legend can give information about:
- Time per frame
- Name and color
- Data type and shape
- Content
- Indexe
- Operator params:



LEGEND

Time per frame:
If important you can specify the time per frame.

Name and color:
Information related variable name of input or output observables.
Also color is included and used to distinguish observables.

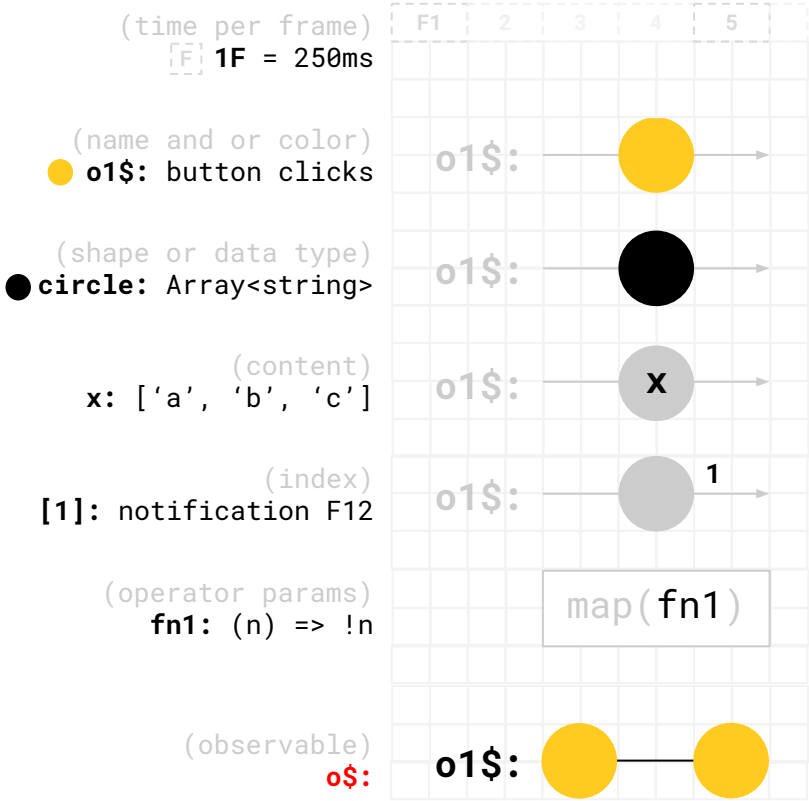
Data type and shape:
If needed you can also distinguish data types from each other.
This is done by shapes.

Content:
The content can be used to directly place the notification inside i.e. “1” or “a”.
If you want to be more specific use the content as reference to describe complex objects. I.e. “x: {name: ‘Eric’}“

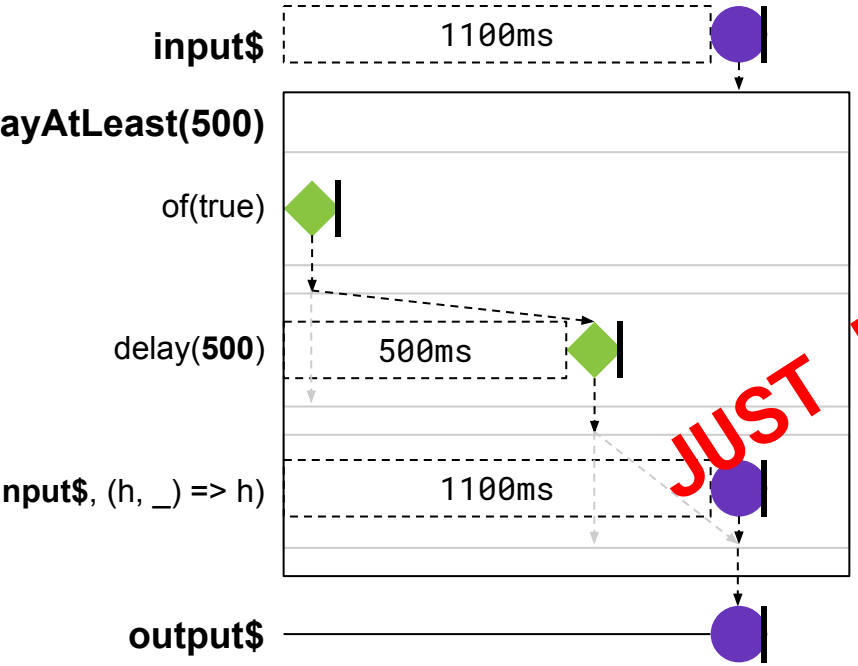
Indexe:
Index chars can be letters used to reference consumer events or numbers for everything else.

Operator params:
Details for operator params like functions or notifications.

ADD OBSERVABLES TOO HERE

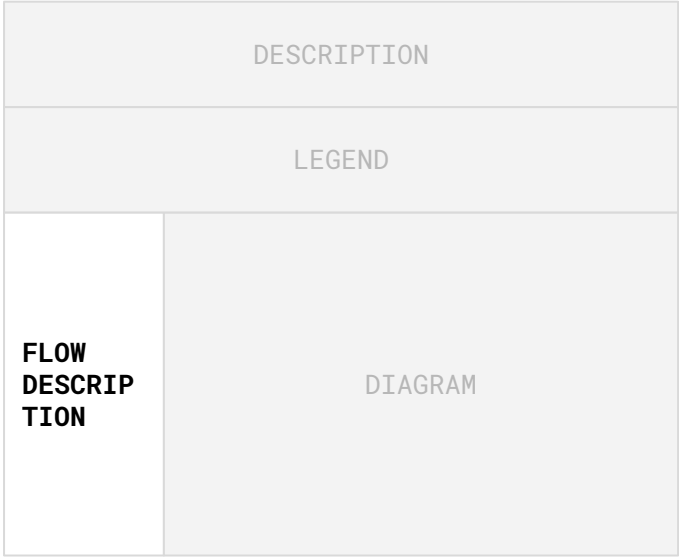


FLOW DESCRIPTION



JUST DO IT!

equal to diagram height



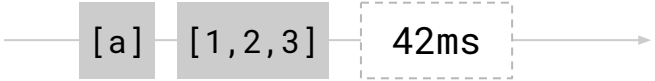
0 or 10 blocks width

FLOW DESCRIPTION

Bold+Big (Observable) **click$**

Normal+Big (Operator) scan(fn,s)

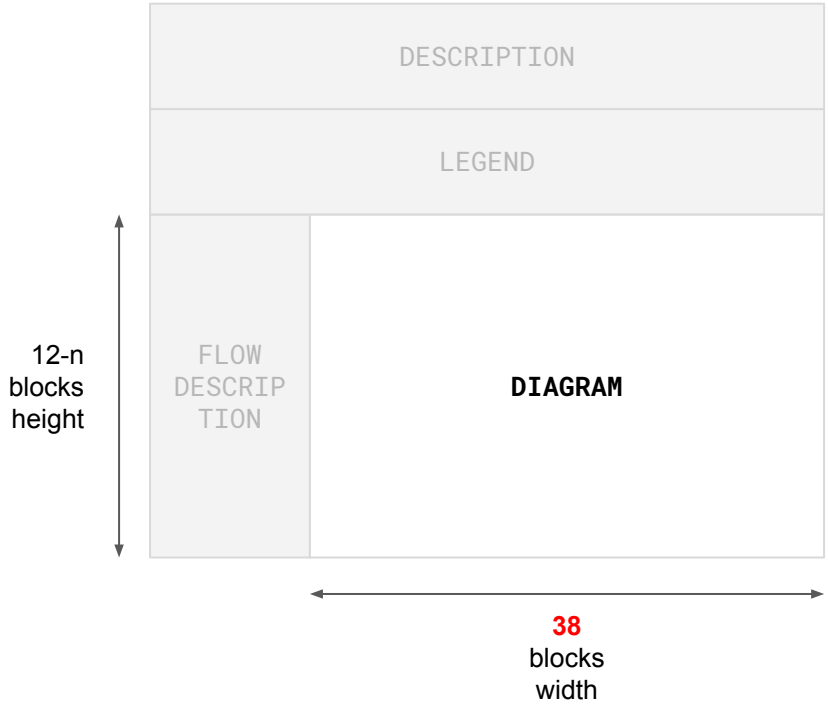
Normal+Medium (Notification, Timespan)



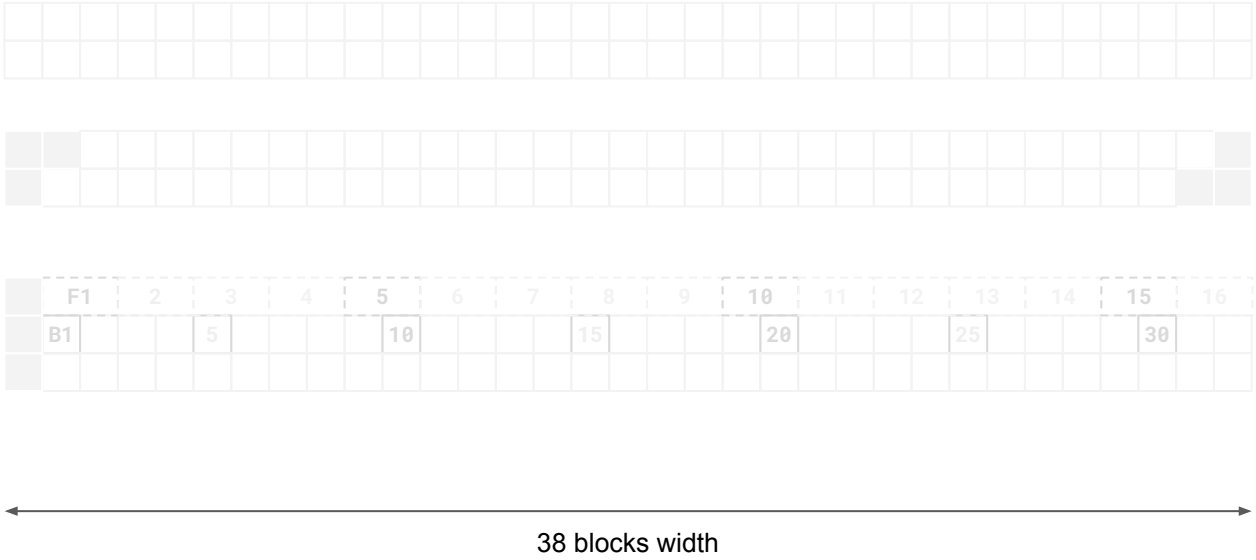
Bold+Small(Index + Legend)

DIAGRAM

```
Diagram:
- Grid
  - Padding
  - Frame scale
  - Block scale
  - Min-With
- Positioning
  - Lanes
  - Horizontal
  - Vertical
- Notifications
  - Content
  - Shape
  - Color
  - Multiple notifications
  - Dynamic with
  - Inner components
  - Inner state => scan, buffer
- Operator
  - Styles
  - Operations and lines
  - Operator grouping
- Multiple components
- Inactive components
- Order of involved observables
```



GRID



The diagram is drawn in a **grid** which is made out **of blocks.** It's **optional** to display.

The grid has a **padding of 1 block.**

Also **optional** are the **blocks and frames scales.**

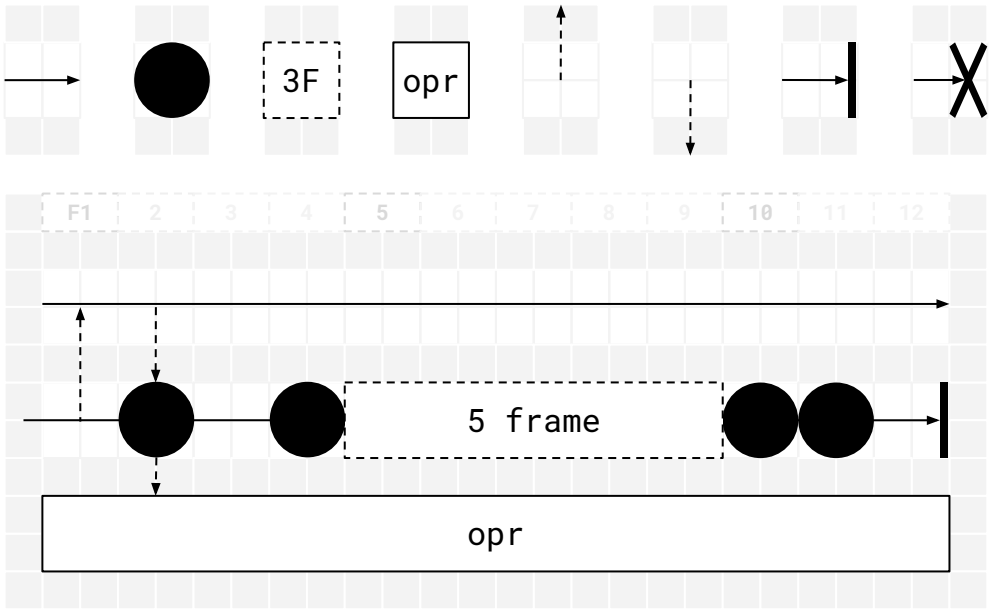
The **block scale** is mainly here to position components. It is **very rarely used**.

The **frame scale** is here **to measure** the exact **time** in diagrams.

For a consistent appearance we suggest a **minimal with of 38 blocks**.

Fix min width!

POSITIONING



For every **component** you have to provide a **space of 2 by 2 blocks**. This is important in combination with the frame scale.

The components are **arranged in lanes**.

The minimum **vertical space is 1 block.**

Lines are placed either horizontally or vertically **on the edges of a block**.

Horizontal lines can only be placed in the center of a frame.

Between lanes there has to be a **horizontal space of 1 block**.

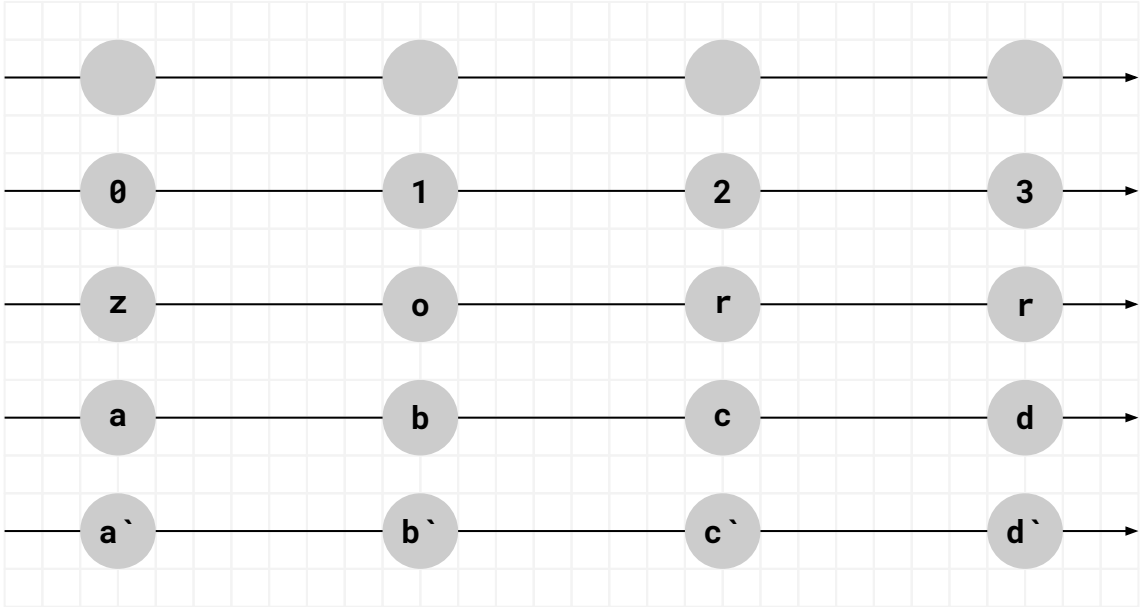
Mention vertical separation of observables if needed
KUDOS @ncjamieson

Include @CedricSoulas solution for complete and start in the same frame

NOTIFICATION AND CONTENT

a: ['a', 'b', 'c', 'd', 'e']
b: ['e', 'f', 'g', 'h', 'i']
c: ['i', 'k']

a`: 'abcde'
b`: 'efghi'
c`: 'ik'



In marble diagrams the **emissions of observables are** called **notifications.**

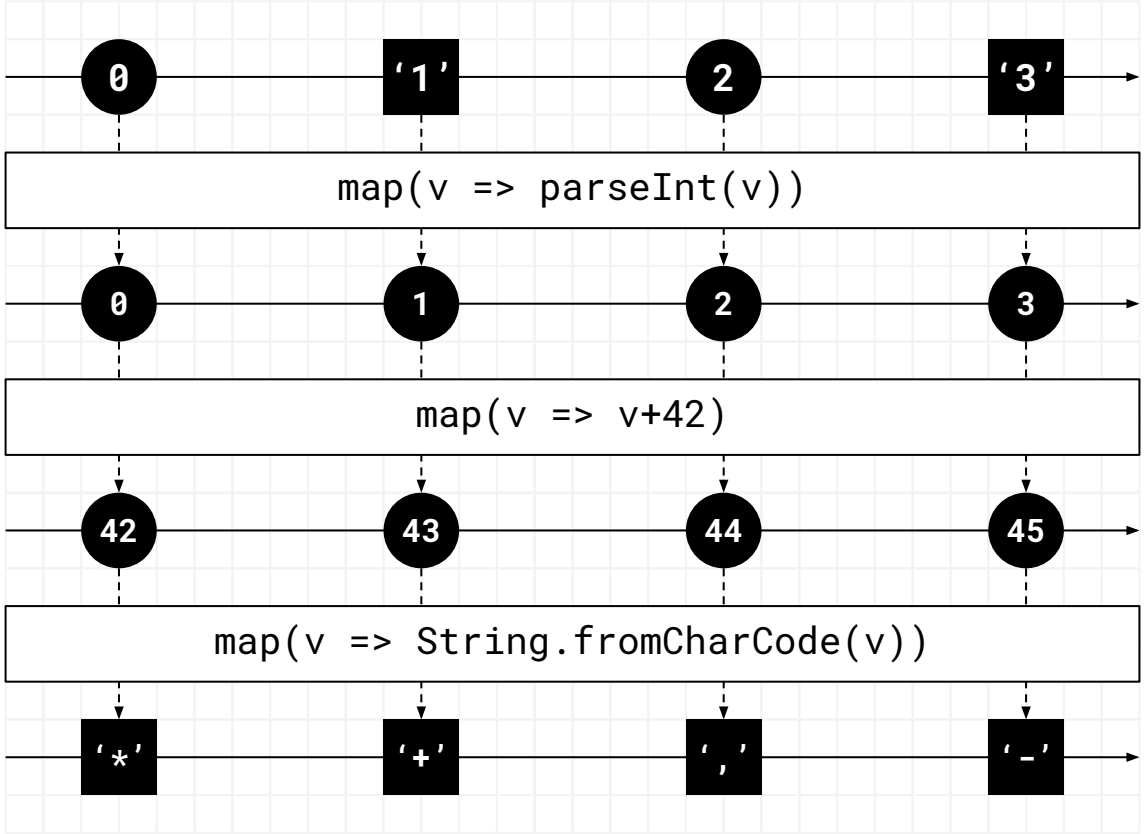
They are displayed as shapes with color.
To give more **information about** the **data of a notification** you can **use content**.

Content is **any set of characters** placed centered **in the shapes area**.

Content can be **directly representing the notification or** serving as a **reference for the legend** with details.

You can use the **prime symbol [`]** to show **relations in derived notifications**. A => A` => A``

NOTIFICATION AND SHAPE

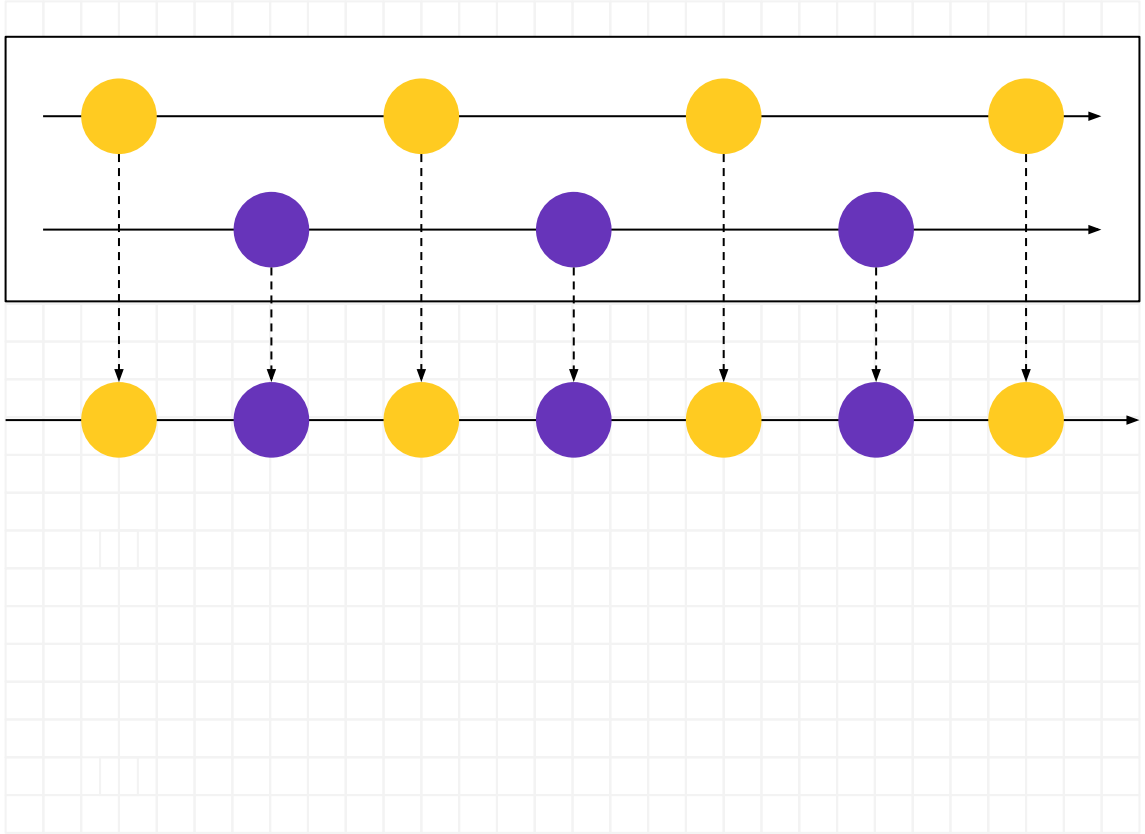


Notifications have shape. **Shape is** closely connected to content and serves an **abstraction for the data type**.
I.e. number or complex objects

In some cases it helps the readability of a diagram.

Especially when you have **data type transformations** or **observables that emit different data types**.

NOTIFICATION AND COLOR



Color in diagrams **is used to distinguish** notifications from different **observables**.

Colors are **applied to input observables**.

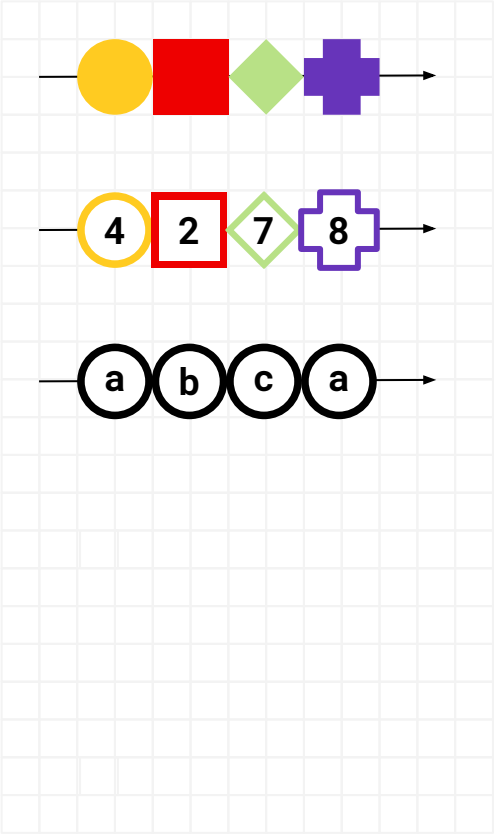
The **output observable can be drawn in 2 versions.**

a) **It has a different color**.
b) **It has no own color**, but is a result of all colors from the input.

The **output observable has no own color**, but is a result of all colors from the input.

In more complex diagrams you have to **decide when to switch back to a single color** or not.

MULTIPLE NOTIFICATIONS IN ONE FRAME



As **notifications** can only be placed in frames, they **can not overlap**.

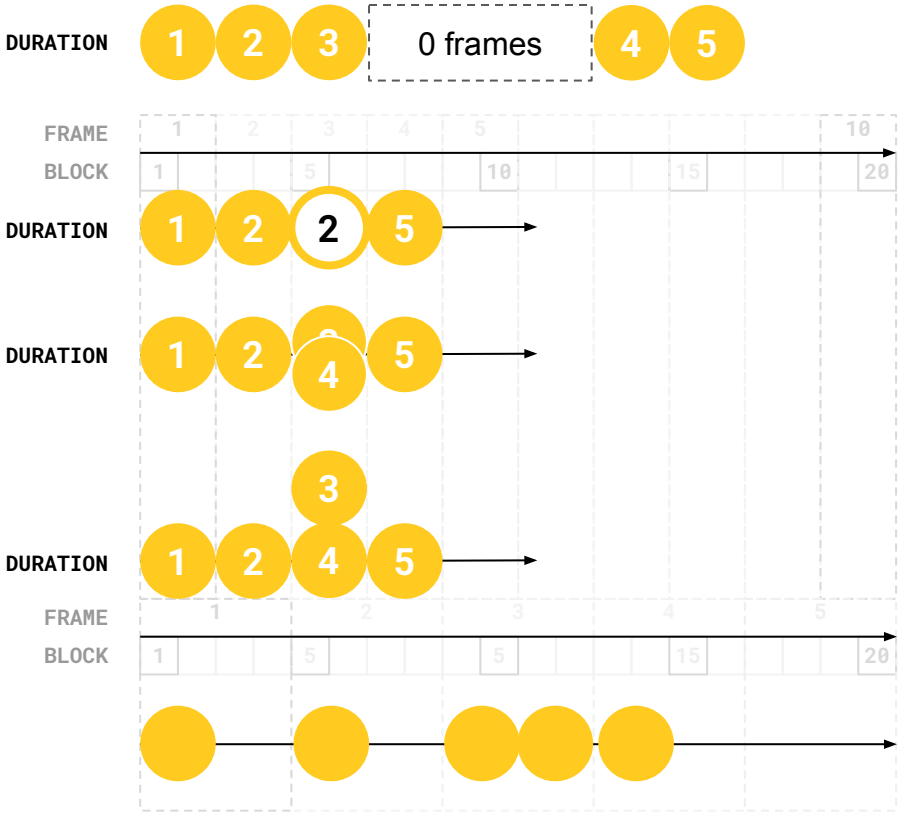
However technically it's possible to have **multiple notifications in one frame**.

In very rare cases there can be **multiple different colors and shapes** in the same frame. In this case we **use the legend** to provide enough information.

Is this solution good?
Examples for colors and shapes?
Multiple different colored Notifications => HOW TO?

INCLUDE:
Show a dynamic width frame with multiple notifications! => maybe beyond the standard section?

MULTIPLE NOTIFICATIONS IN ONE FRAME



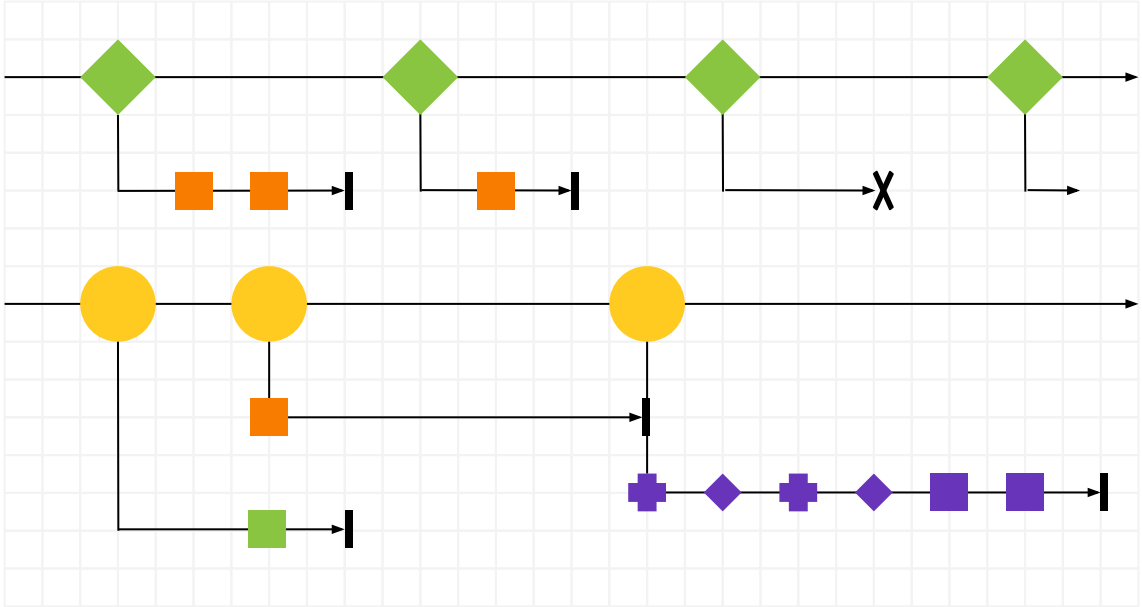
In several diagrams there is the situation where we have to display multiple components like notifications and or error or complete.

In the following we describe the actual set of all explored options

Duration:

NOTIFICATION AND DYNAMIC WIDTH

NOTIFICATION AND INNER COMPONENTS



If you want to display higher order observables you can use small components size for it.

Small size is rarely used and only serve as a symbol.

As normal as well as small components take a 2 by 2 blocks space **all positioning and spacing rules apply also for small** components.

It does not reflect the real behaviour of the inner observable.

The **real behavior is displayed in the operator** which takes these notifications.

OPERATORS STYLES

Name

```
barFo
```

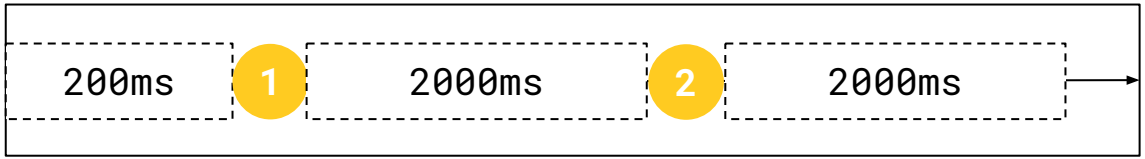
Name and param(psudo)

```
barFo(v => v * 2)
```

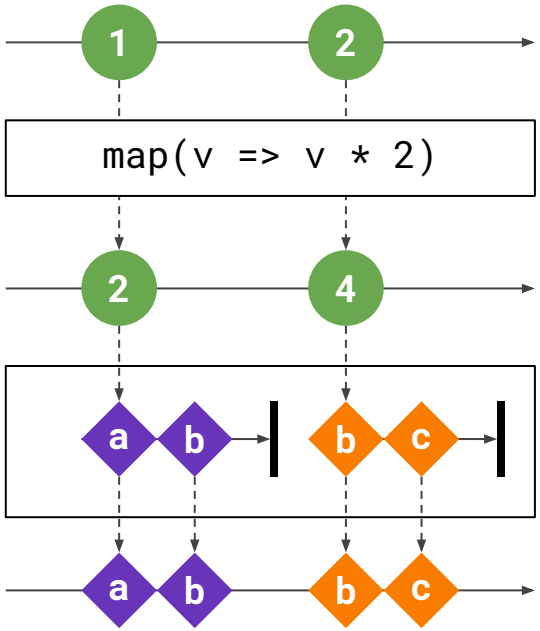
Name and param(marble)

```
barFo( ● => ■→| )
```

Internals



OPERATOR AND LINE



Operators perform **operations** which **happen at a certain point in time** and are displayed as **vertical dashed lines with and arrow at the lower end**.

Form **simple operations** the **lines** are **behind** the **operator** box. This helps to have no overlappings with the operator content.

For operators where the **internal visible** the **lines** are **on top** of the operator and **end in another component.**

Add curved lines example (scan)

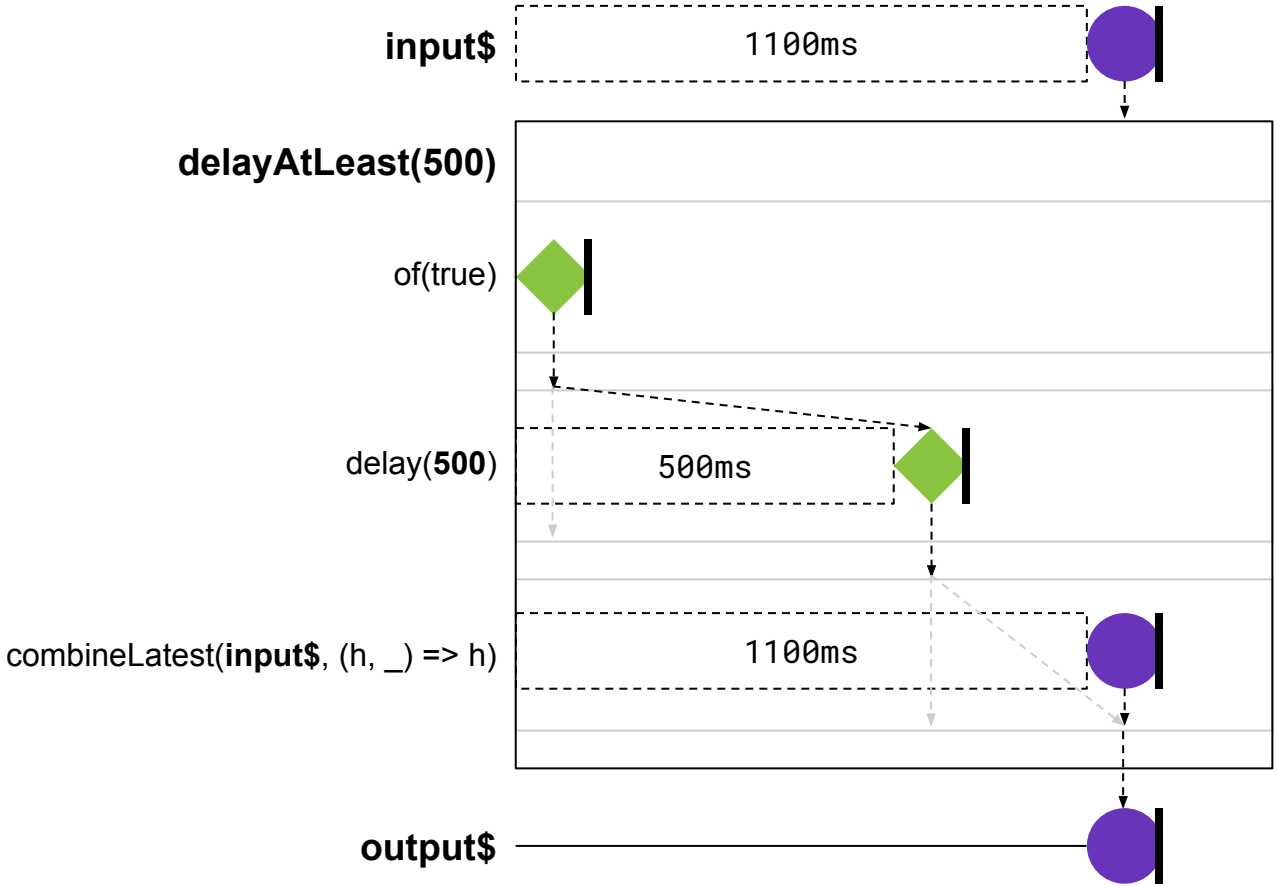
Show angled line from cedric

OPERATOR GROUPING

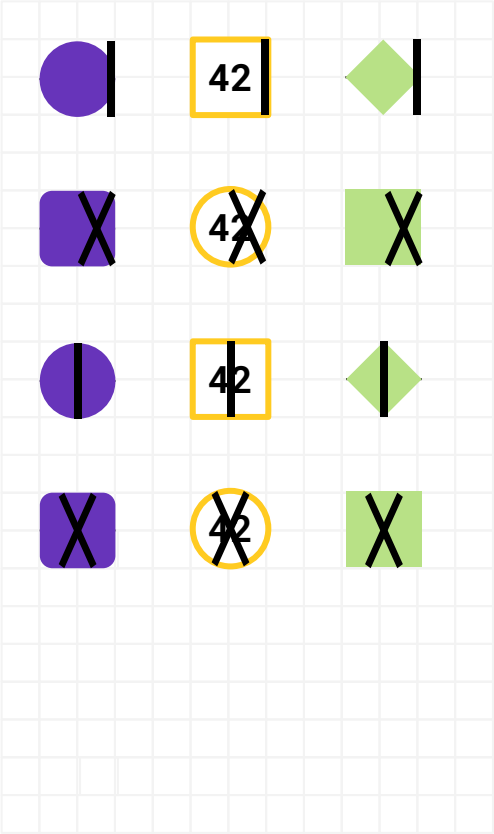
Find better solution

Can we avoid the gray nested box?

SHOW Vertical Separation!!!!!!!
Ref window operator

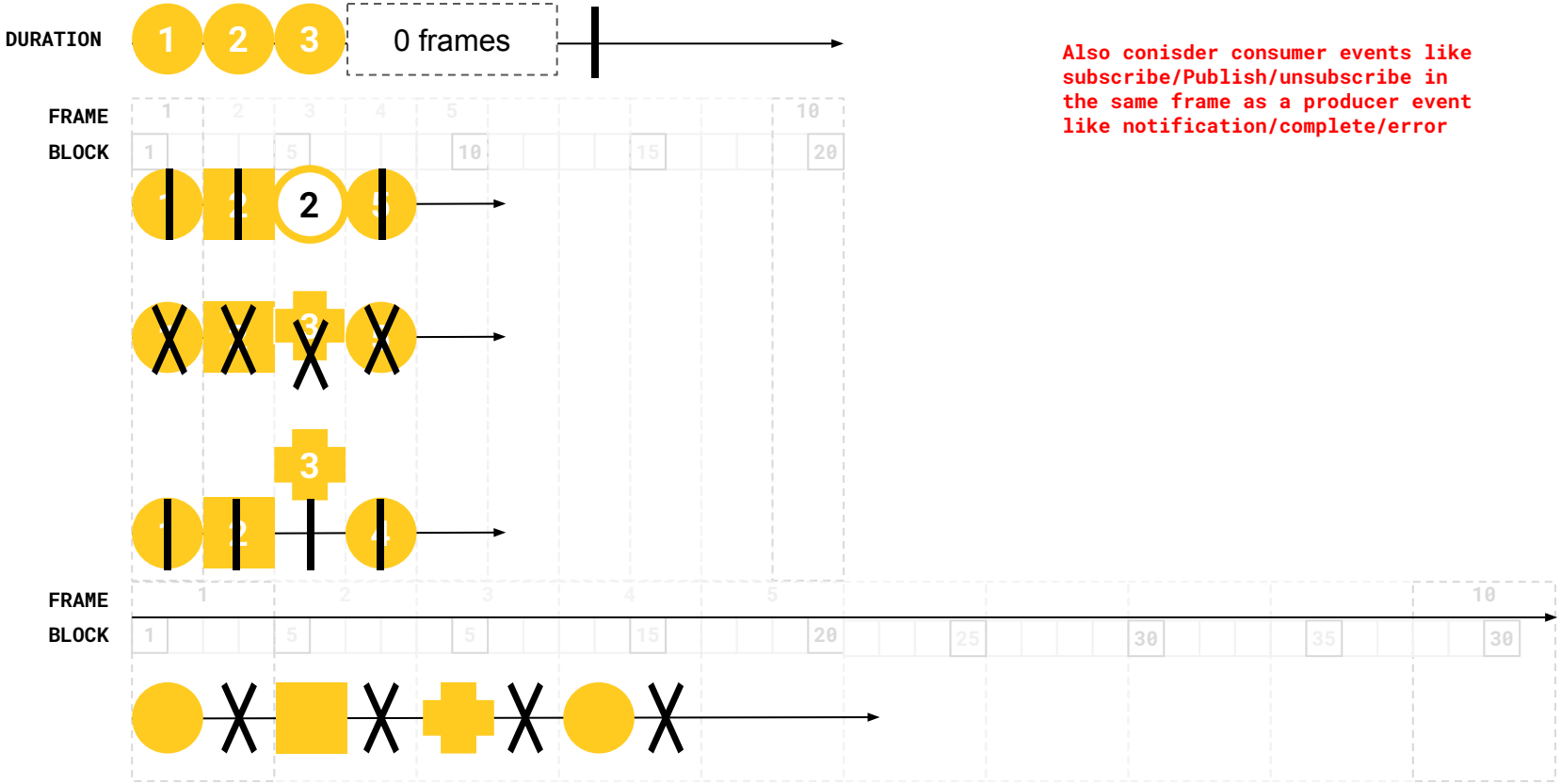


MULTIPLE COMPONENTS IN ONE FRAME



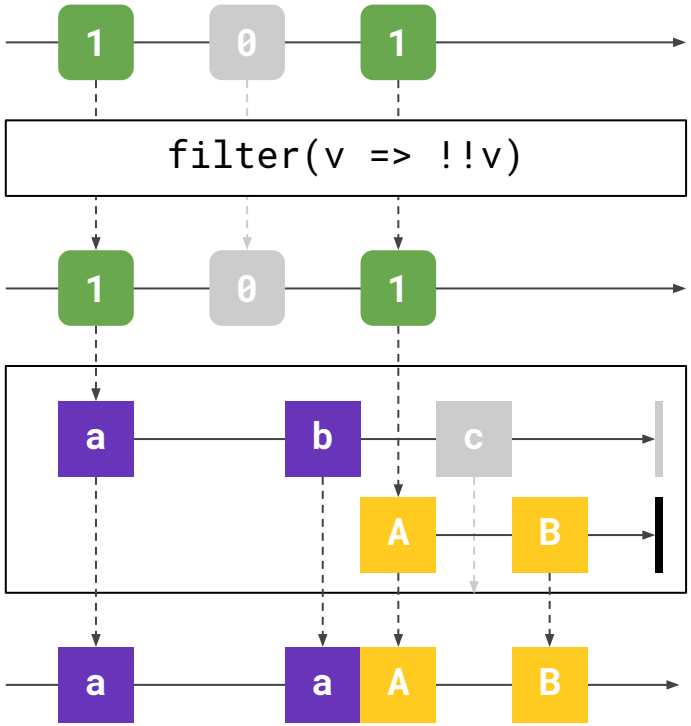
Contrast problem!!!
UGLY!!!
Combo with multiple notifications
not working!

MULTIPLE COMPONENTS IN ONE FRAME



Also conisder consumer events like subscribe/Publish/unsubscribe in the same frame as a producer event like notification/complete/error

INACTIVE COMPONENTS



Sometimes it serves helpful information to **show notifications** and **operations** as well as **complete** and **error**, **in inactive color if they don't occur**.

Inactive operations can also end on the operator context. This could be on the **in and outside**.

Include rules for i.e. filter operator. Where does the black line stop where does the gray line start

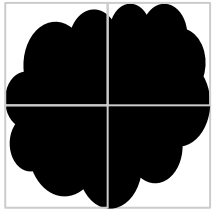
ORDER OF INVOLVED OBSERVABLES

NUMBER OF EXAMPLES PER OPERATOR

Less complexity and more different diagrams/examples of notification distance, complete and error!

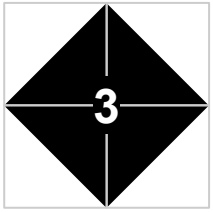
BEYOND THE STANDARD

ALTERNATIVE SHAPE

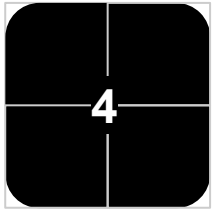


Cloud

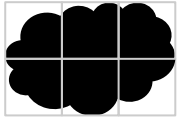
Bottle

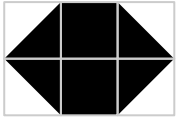


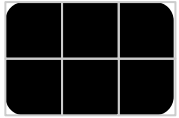
Diamond

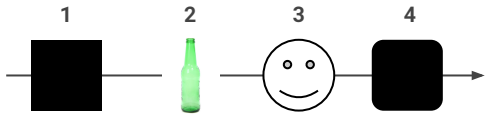


Rounded Rect.



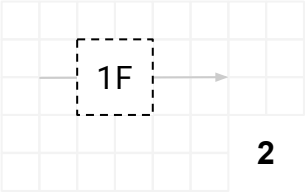






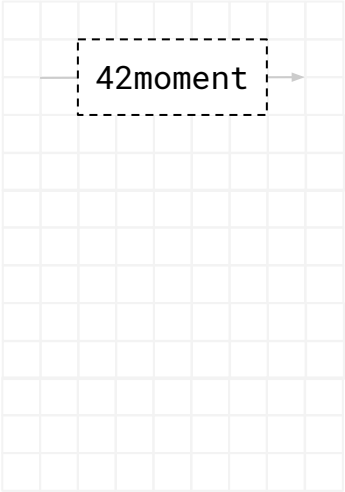
Also possible to use Images, emoticons and so on

ALTERNATIVE UNITS

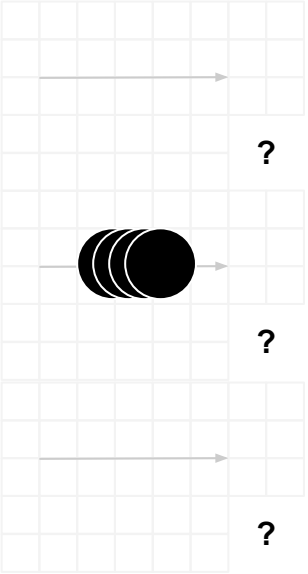


Time Span is here to **specify** a certain **duration of time**.

Of course **you can** also **use any** other **imaginary unit of time** like a moment ;-)

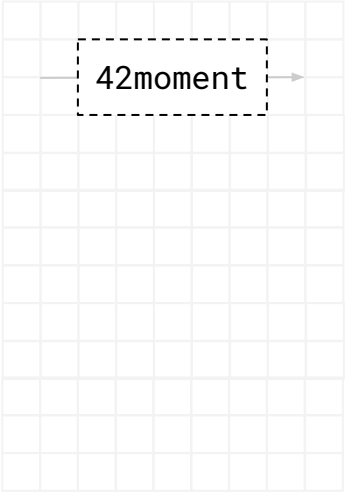


TODOS

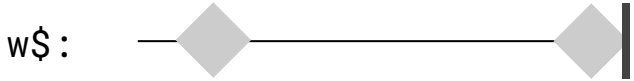


Diagonal Arrows

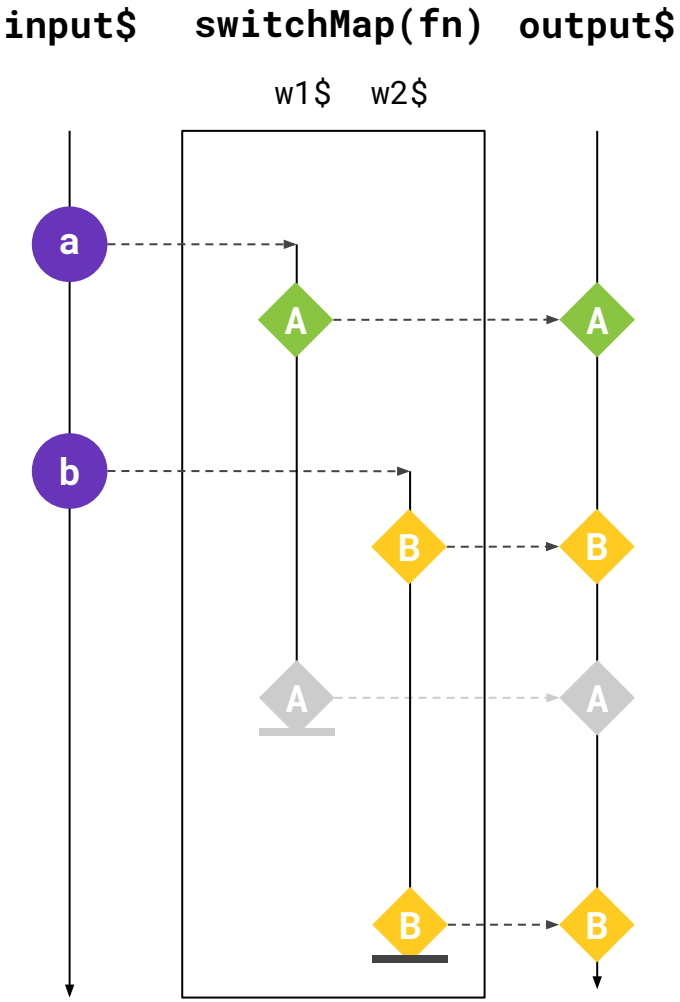
Overlapping shapes



Vertical Layout



fn: (n) => ws$



Contributers

- **Nicholas Jamieson** @ncjamieson
- **Cédric Soulas** @CedricSoulas
- **#RxJS Community** over twitter polls