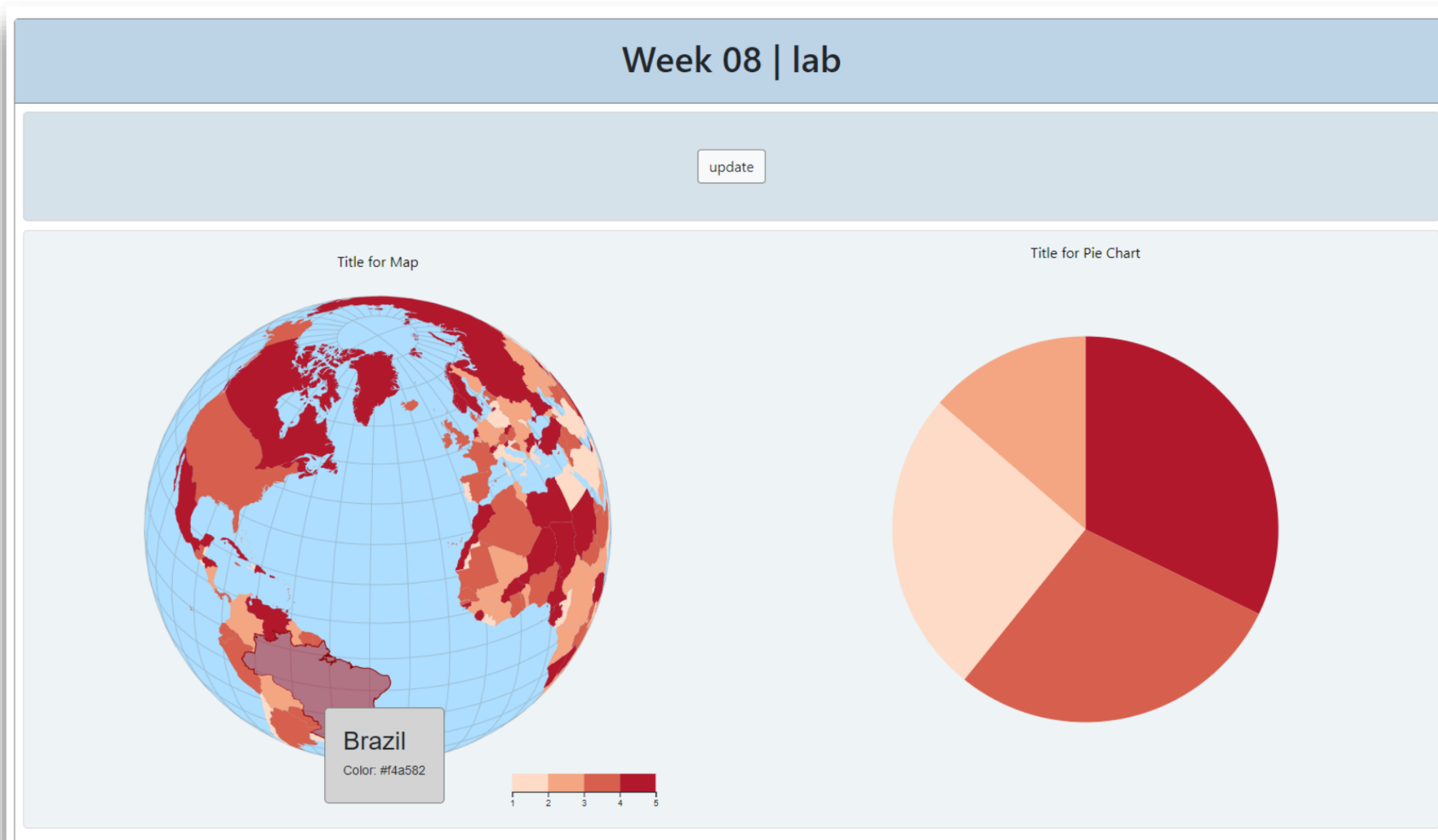


# Lab - Week 6

Sketching, Storyboarding, D3 layouts, Maps



Sketching

Interaction Storyboarding

# Sketching

# Sketching

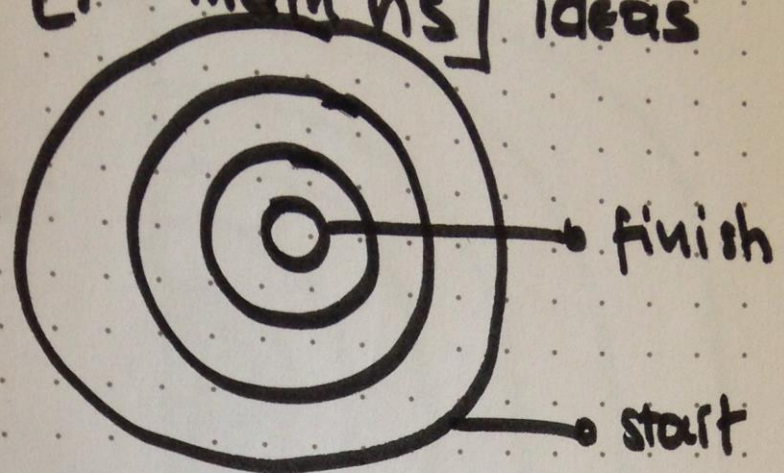
# Design and sketching are *constructive* perception



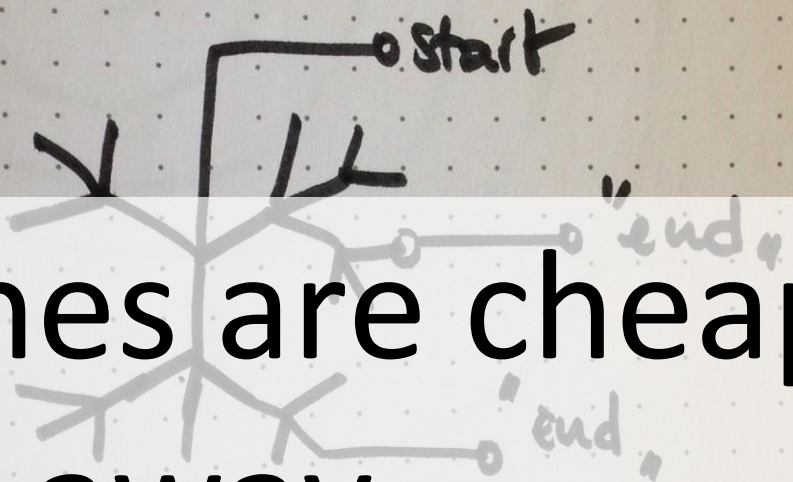


New sketch [for main vis] ideas

①



②



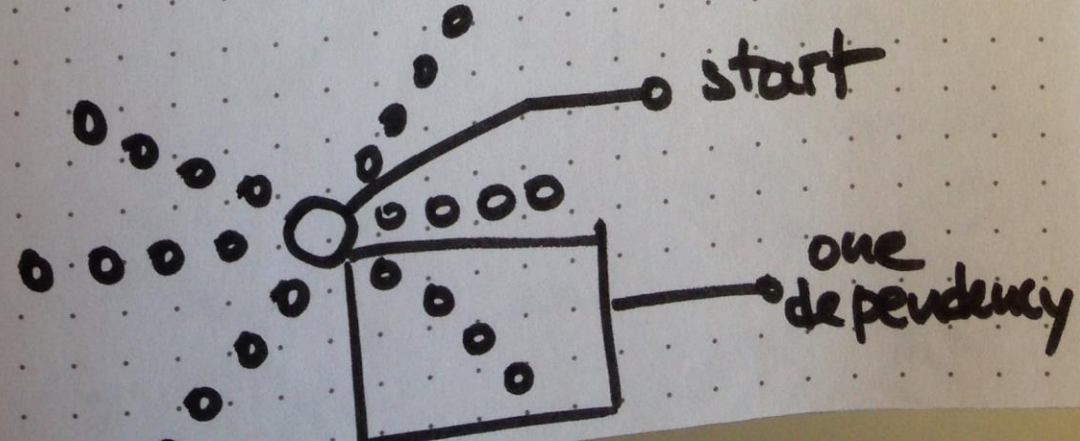
Sketches are cheap, fast, and easy to throw away

③



"course"  
footprint  
N Gerhardt's  
flower

④

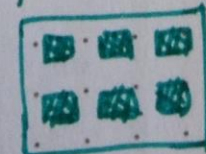


⑧



dependencies

⑨



detail of module  
or concept

conventional  
way to  
go through  
modules



“

My strategy has always been:  
Be wrong as fast as we can.

— Andrew Stanton, Pixar





# What do you need?





# Activity

Create as many different sketches to visualize these two quantities as you can. (2 min)



42

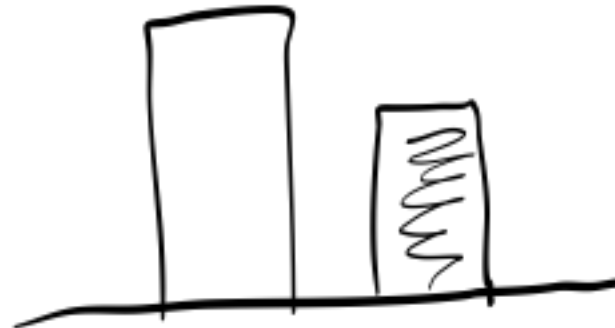
23

# Most likely results

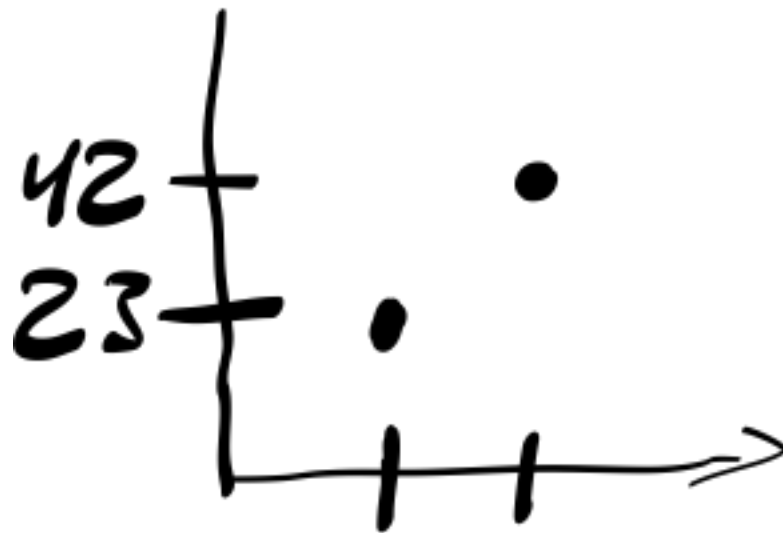
Pie Chart



Bar Chart



Scatterplot



23

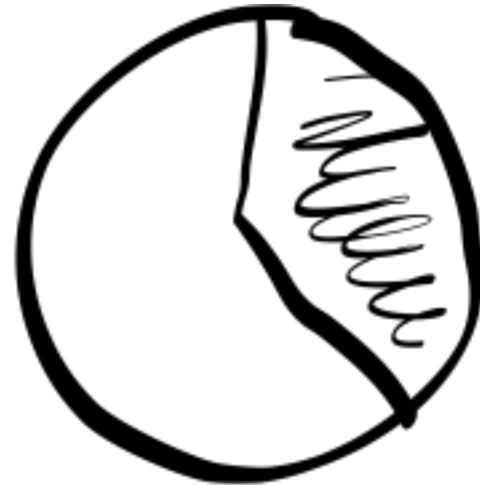
42

Numbers

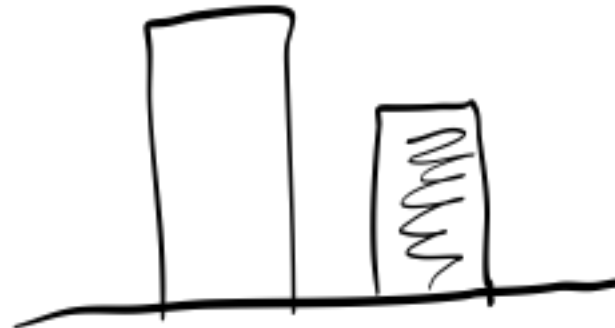


# Design Fixation: Blind adherence to a set of ideas or concepts

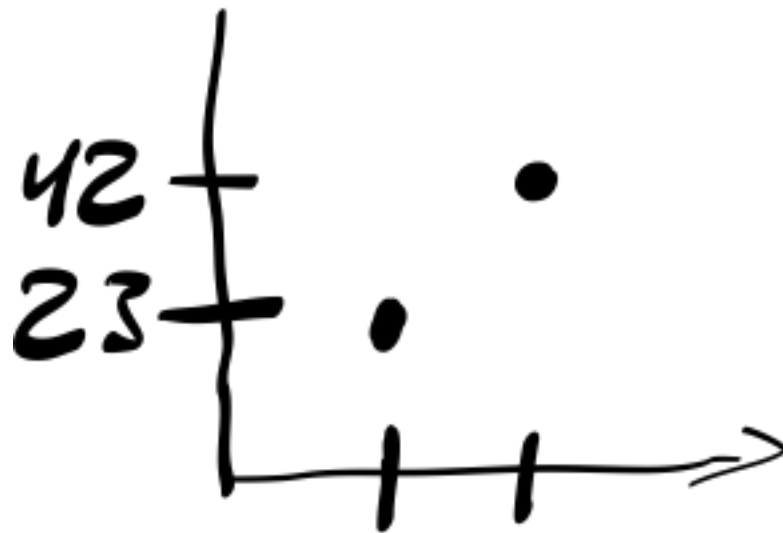
Pie Chart



Bar Chart



Scatterplot



23

42

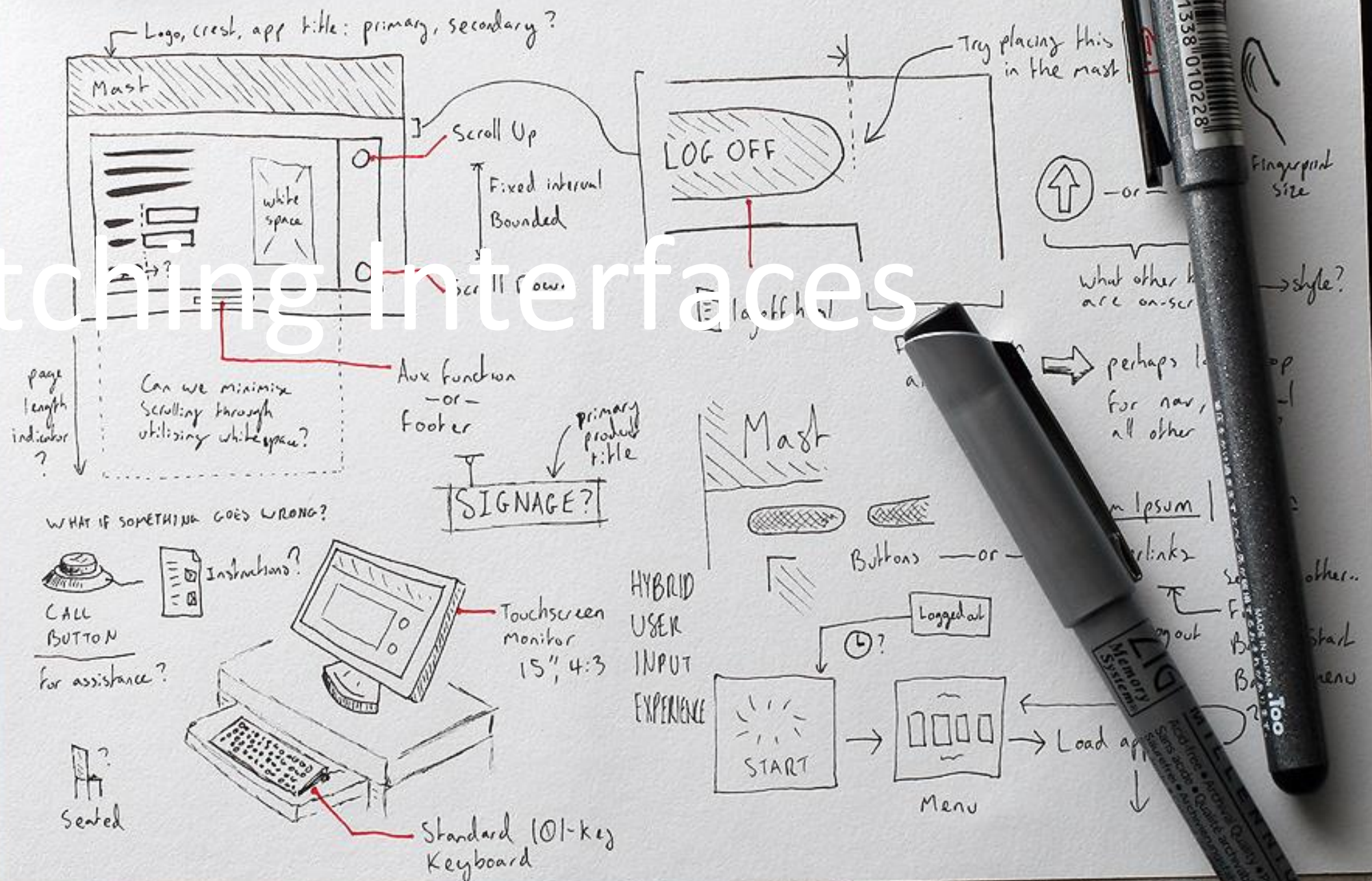
Numbers

# Interaction Storyboard



# Create a Storyboard

# Sketching Interfaces





St

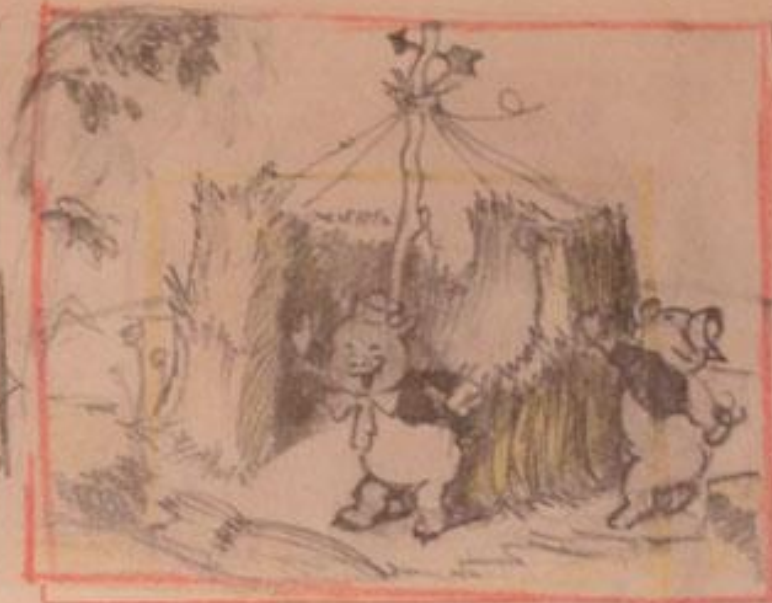


Storyboard: A  
linear  
sequence of  
illustrations  
that visualize a  
story.

Scene I

First pig  
building  
straw hovel.

MOVE  
DOWN  
TO  
ABOUT  
3/2  
use  
larger  
FIELD



Scene II

Second pig  
building  
house of sticks

SHOWN  
STICKS?

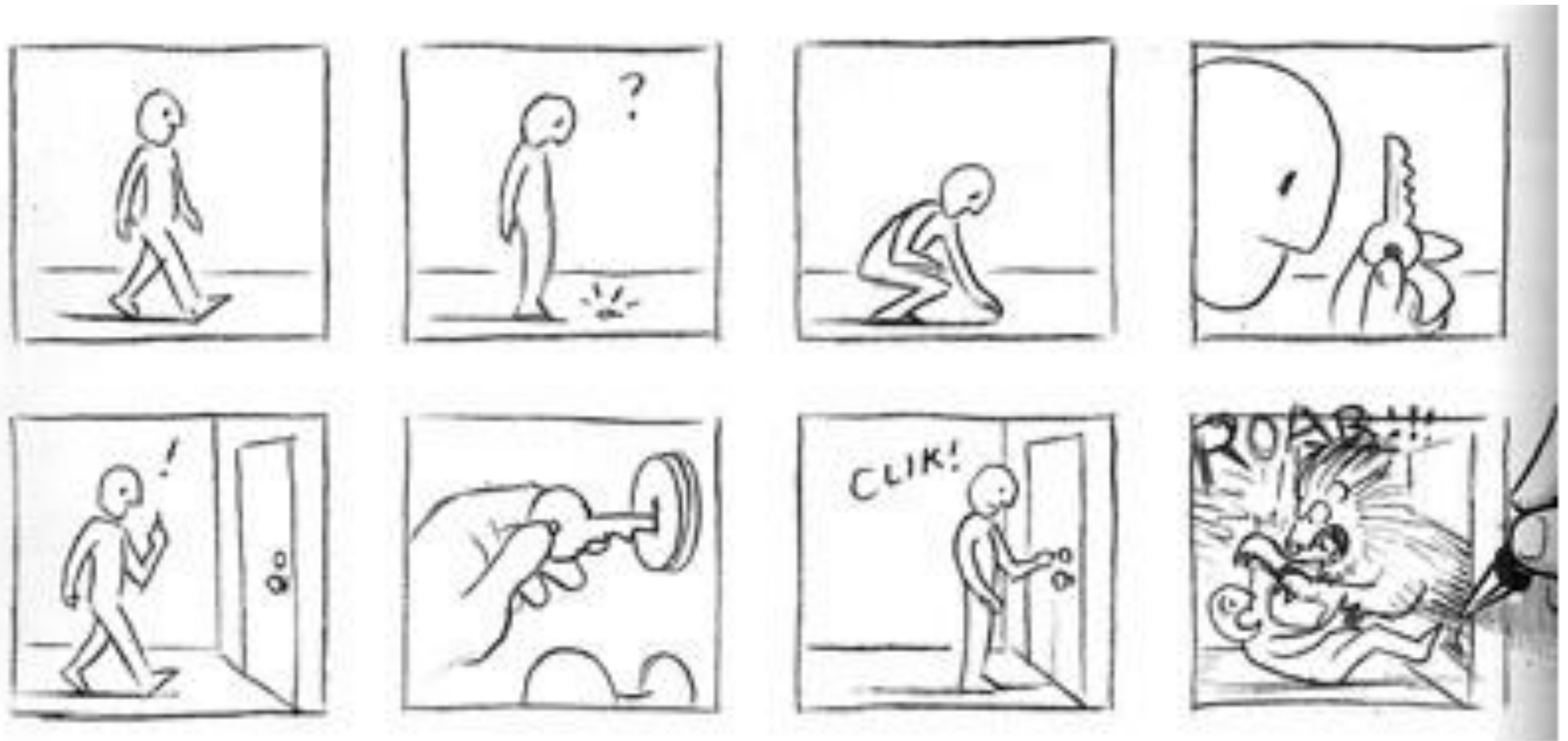


Scene III

Third pig  
building  
house of brick

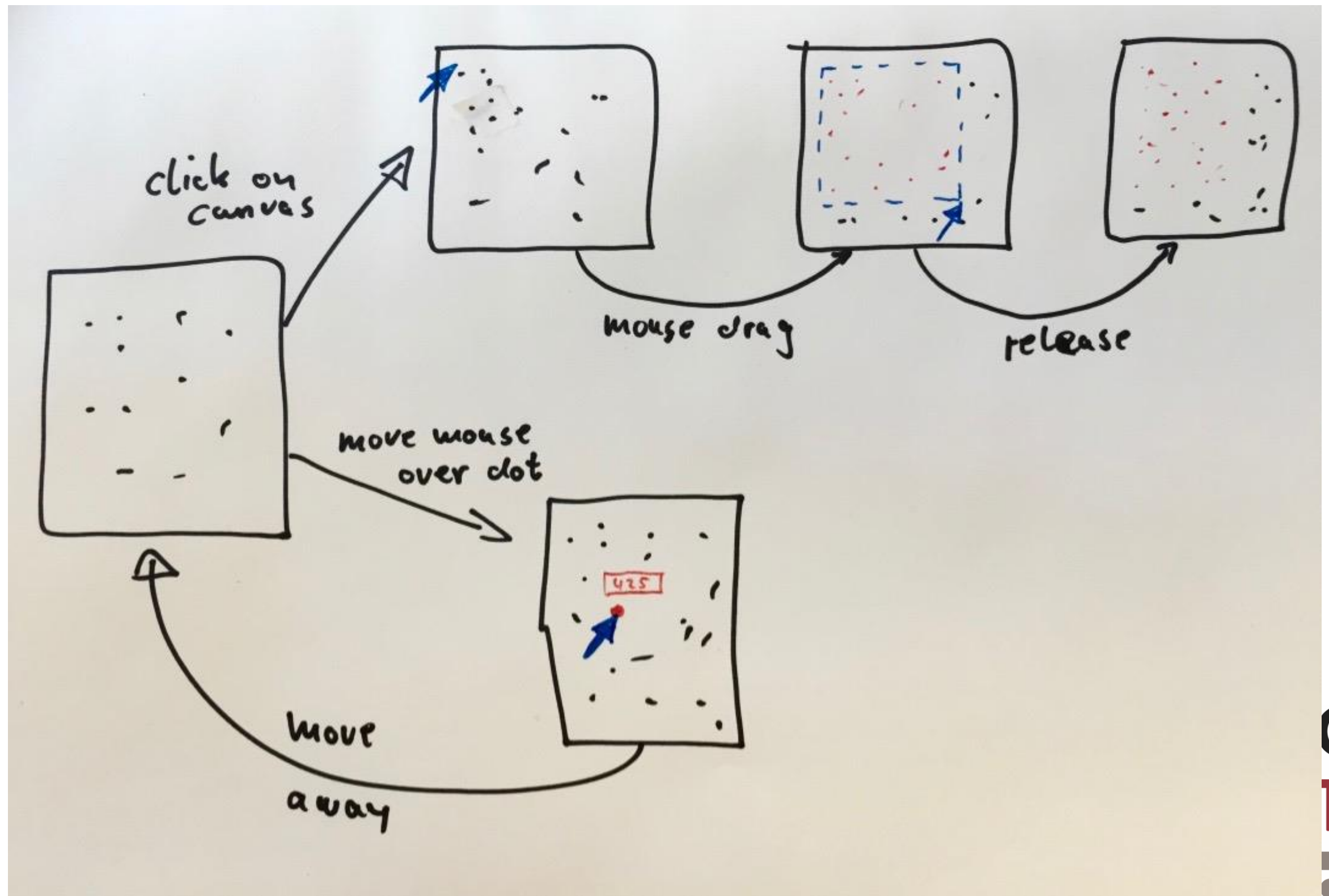


Each frame is substantial. Leaving it out alters the story.

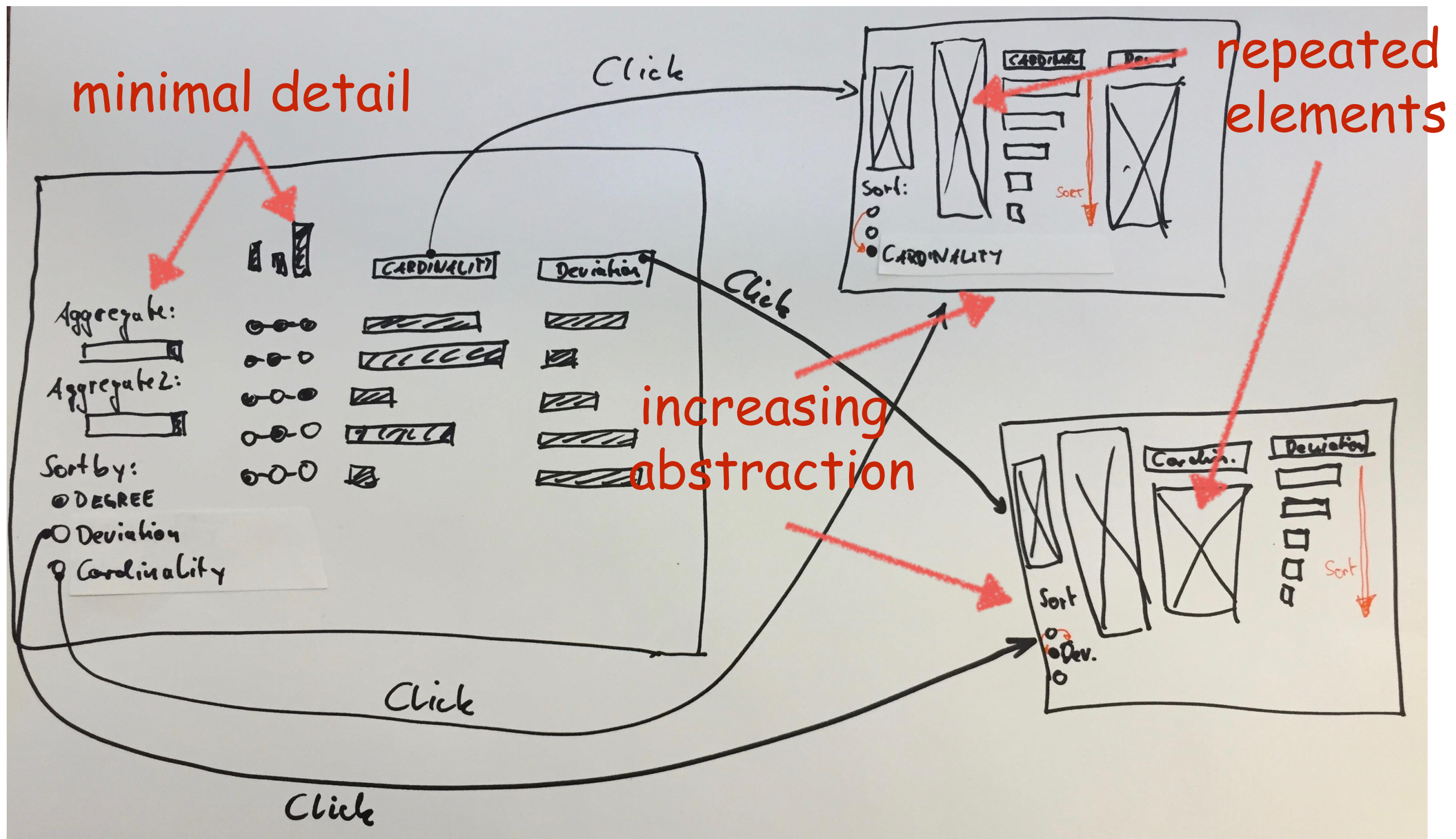


Scott McCloud, 'Understanding Comics'



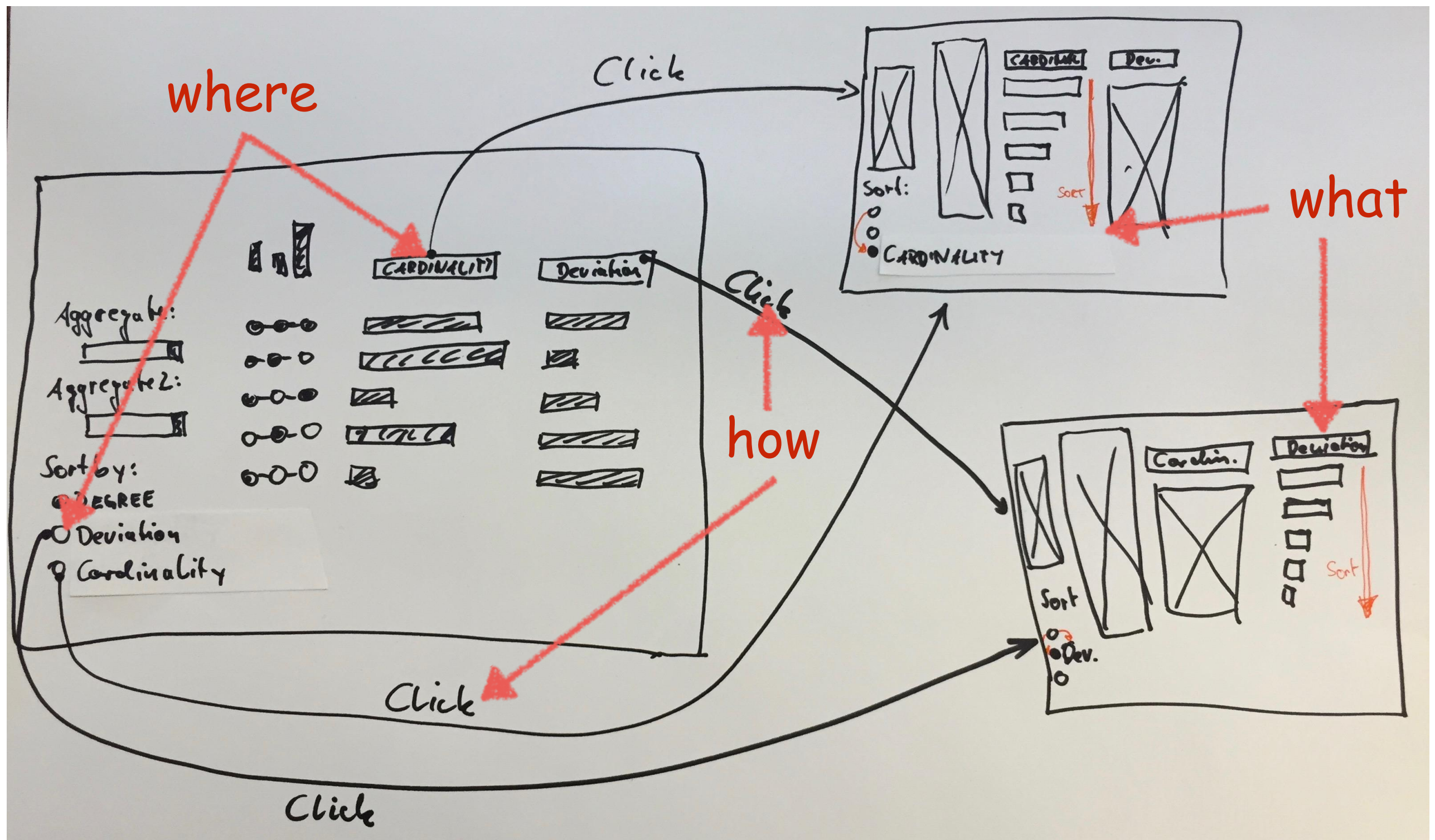


# Abstract the visual elements for **simplicity** and **clarity**



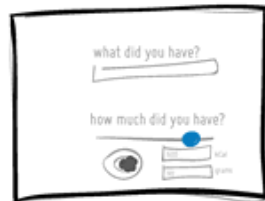
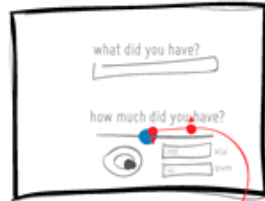


# Where and how an action is triggered and what does change





### Add and adjust quantity



! Make quantity understandable

### Auto text completion



KEYPRESS



KEYDOWN: ENTER

! Combine functions & simplify the interface (your own + favourites + list, all in one)

### Divide Preview from Input

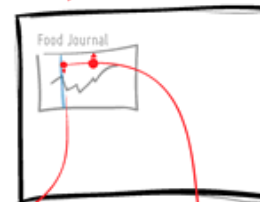


! Visualize where data exists at all times

### Calorie Preview + Select Days vs. Ranges



MOUSEOVER

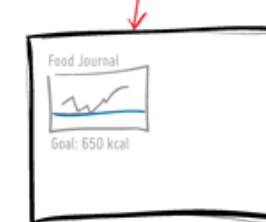
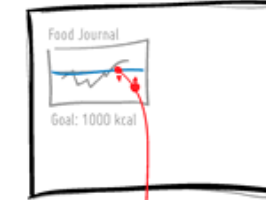


MOUSEOVER

### MouseOver Delete



### Resizable Goals



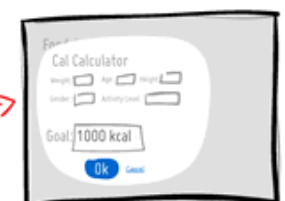
### Stretchable Timeline



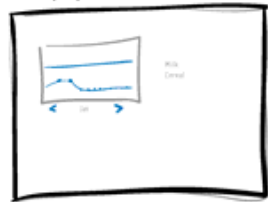
! Contextualize Interactions

! Ease data entry by editing time ranges.

### Inline Calory Target Calculator



### Prepopulate with defaults



! Ease the learning curve

### Add Repeating Items



# Data structures

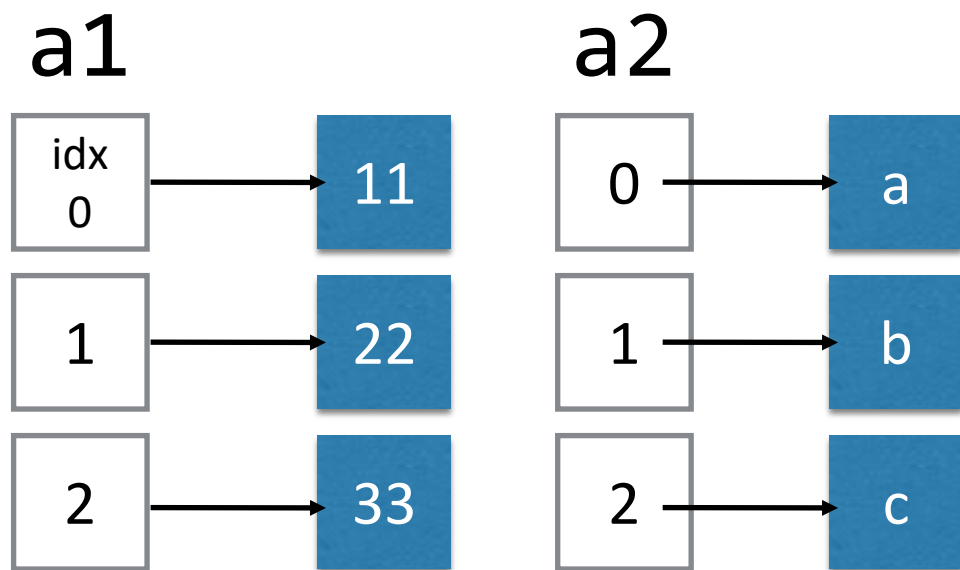
Array to Map (Object)

# Data Structures in JavaScript

arrays:

```
a1 = [11, 22, 33]
```

```
a2 = ['a', 'b', 'c']
```



Access element by idx:

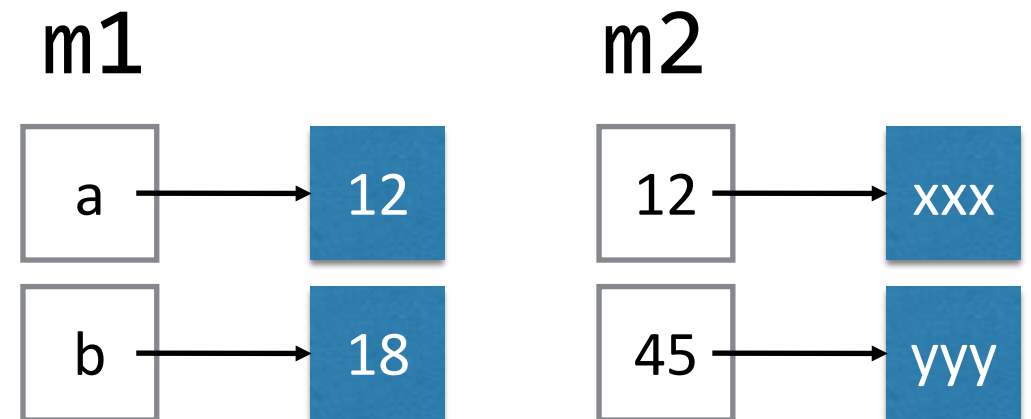
```
a1[2] -> 33
```

```
a2[0] -> 'a'
```

objects (maps):

```
m1 = { 'a': 12, 'b': 18 }
```

```
m2 = { 12: 'xxx', 45: 'yyy' }
```



Access element by key:

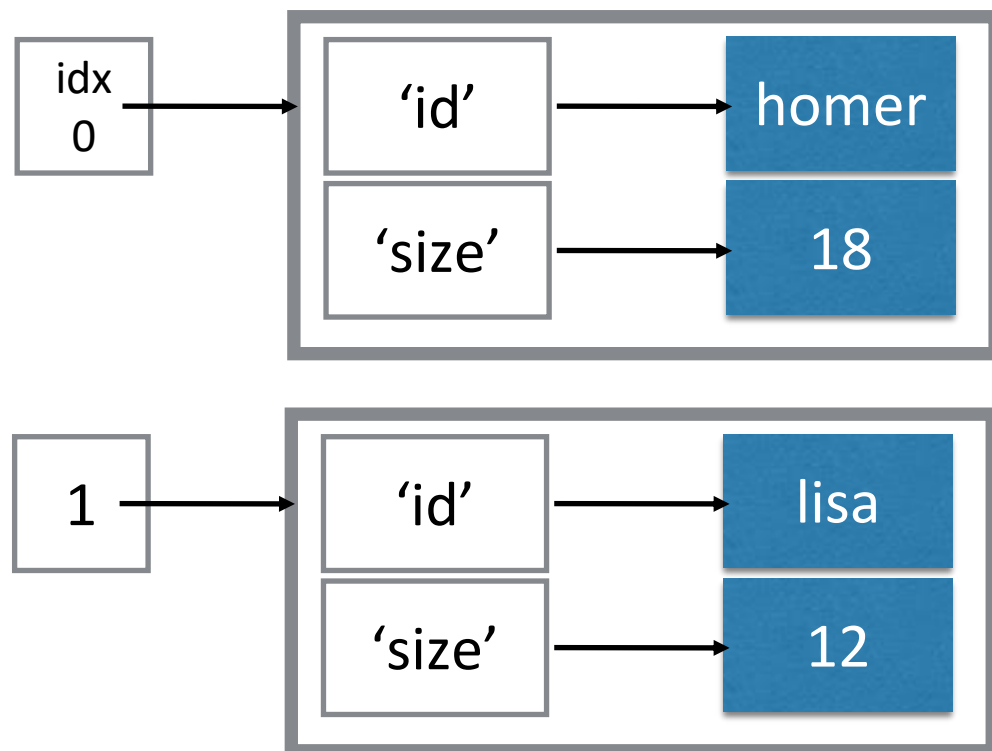
```
m1['b'] -> 18
```

```
m2[12] -> 'xxx'
```



# Data Structures in JavaScript

```
data = [  
  { 'id': 'homer', 'size': 18 },  
  { 'id': 'lisa', 'size': 12 }  
]
```



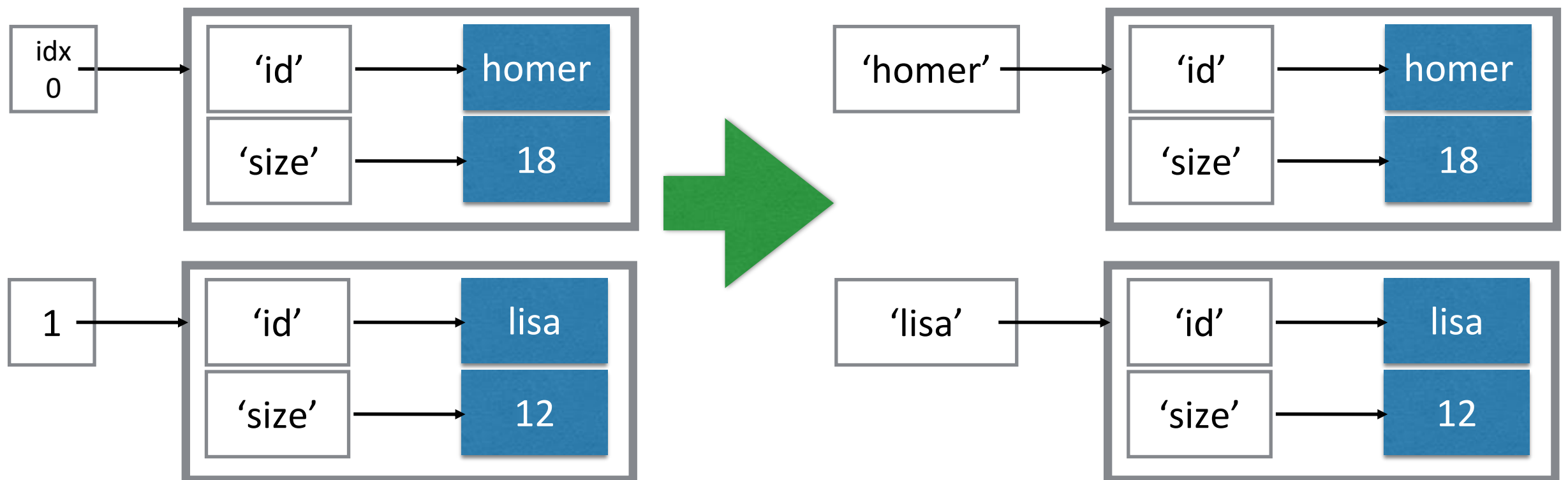
What's the size of homer?

**Strategy 1:** brute force search

**Strategy 2:** convert data

# Optimize data structures for frequent operations

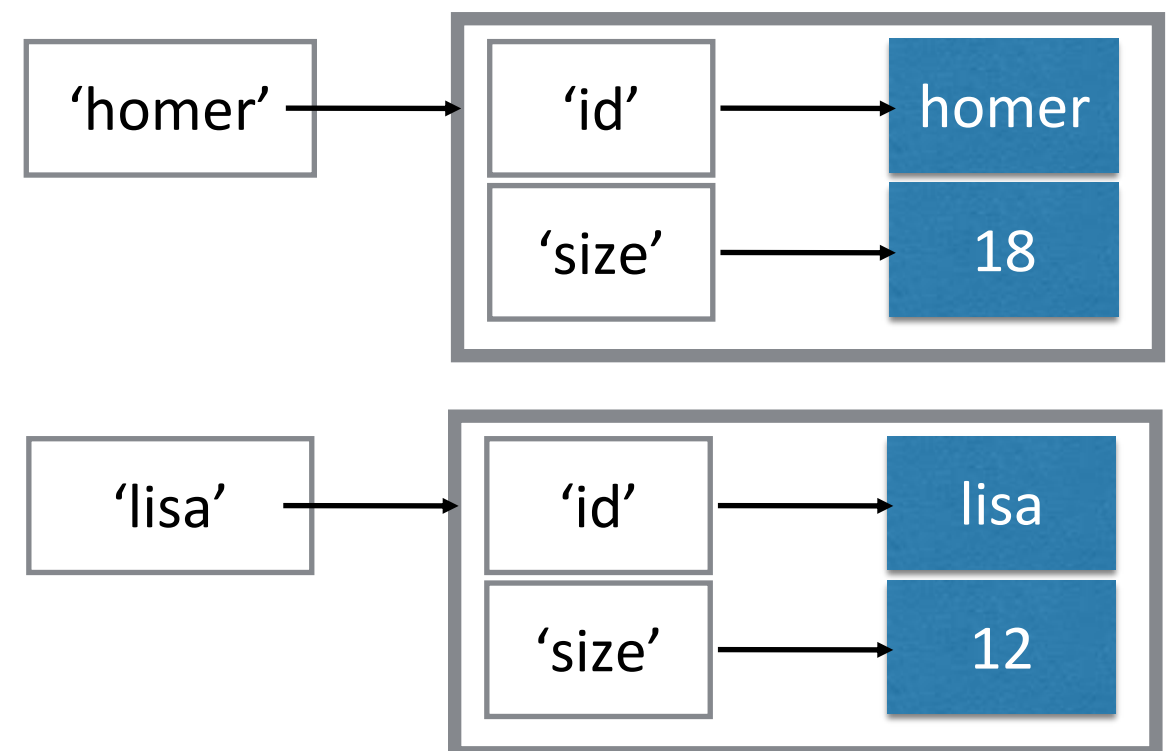
```
data = [  
    {'id': 'homer', 'size': 18},  
    {'id': 'lisa', 'size': 12}  
]
```



# Optimize data structures for frequent operations

```
data = [  
  {'id': 'homer', 'size': 18},  
  {'id': 'lisa', 'size': 12}  
]
```

```
dataMap = {  
  'homer':  
    {'id': 'homer', 'size': 18},  
  'lisa':  
    {'id': 'lisa', 'size': 12}  
}
```





# Optimize data structures for frequent operations

```
data = [  
  {'id': 'homer', 'size': 18},  
  {'id': 'lisa', 'size': 12}  
]
```

```
dataMap = {  
  'homer':  
    {'id': 'homer', 'size': 18},  
  'lisa':  
    {'id': 'lisa', 'size': 12}  
}
```

```
let dataMap = {}  
data.forEach(function(d){dataMap[d['id']]=d;} )  
                                     Key      Value
```

# Optimize data structures for frequent operations

```
data = [  
  {'id': 'homer', 'size': 18},  
  {'id': 'lisa', 'size': 12}  
]
```

```
dataMap = {  
  'homer':  
    {'id': 'homer', 'size': 18},  
  'lisa':  
    {'id': 'lisa', 'size': 12}  
}
```

What's the size of homer?  
`dataMap['homer']['size']`

```
let dataMap = {}  
data.forEach(function(d){dataMap[d['id']] = d;})
```


Null values

# Data can be messy!

## Take care of undefined values!

```
myData =[
  {'a':null, 'b': 18},
  {'a':12, 'b': undefined},
  {'a':33, 'b':54}
]
```

What should  
we do?






# Data can be messy!

## Take care of undefined values!

```
myData =[
  {'a':null, 'b': 18},
  {'a':12, 'b': undefined},
  {'a':33, 'b':54}
]
```

What should  
we do?



### Option 1 —> Filter:


```
myCleanData = myData.filter(
  function(d){
    return d.a != null && d.b != null
  })
```

# Data can be messy!

## Take care of undefined values!

```
myData =[
  {'a':null, 'b': 18},
  {'a':12, 'b': undefined},
  {'a':33, 'b':54}
]
```

What should  
we do?



### Option 1 —> Filter:

```
myCleanData = myData.filter(
  function(d){
    return d.a != null && d.b != null
  })
```


a = null, b=1	d.a != null && d.b != null	FALSE
a = undefined, b=null	d.a != null && d.b != null	FALSE
a= 2, b = null	d.a != null && d.b != null	FALSE

# Data can be messy!

## Take care of undefined values!

```
myData =[
  {'a':null, 'b': 18},
  {'a':12, 'b': undefined},
  {'a':33, 'b':54}
]
```

What should  
we do?



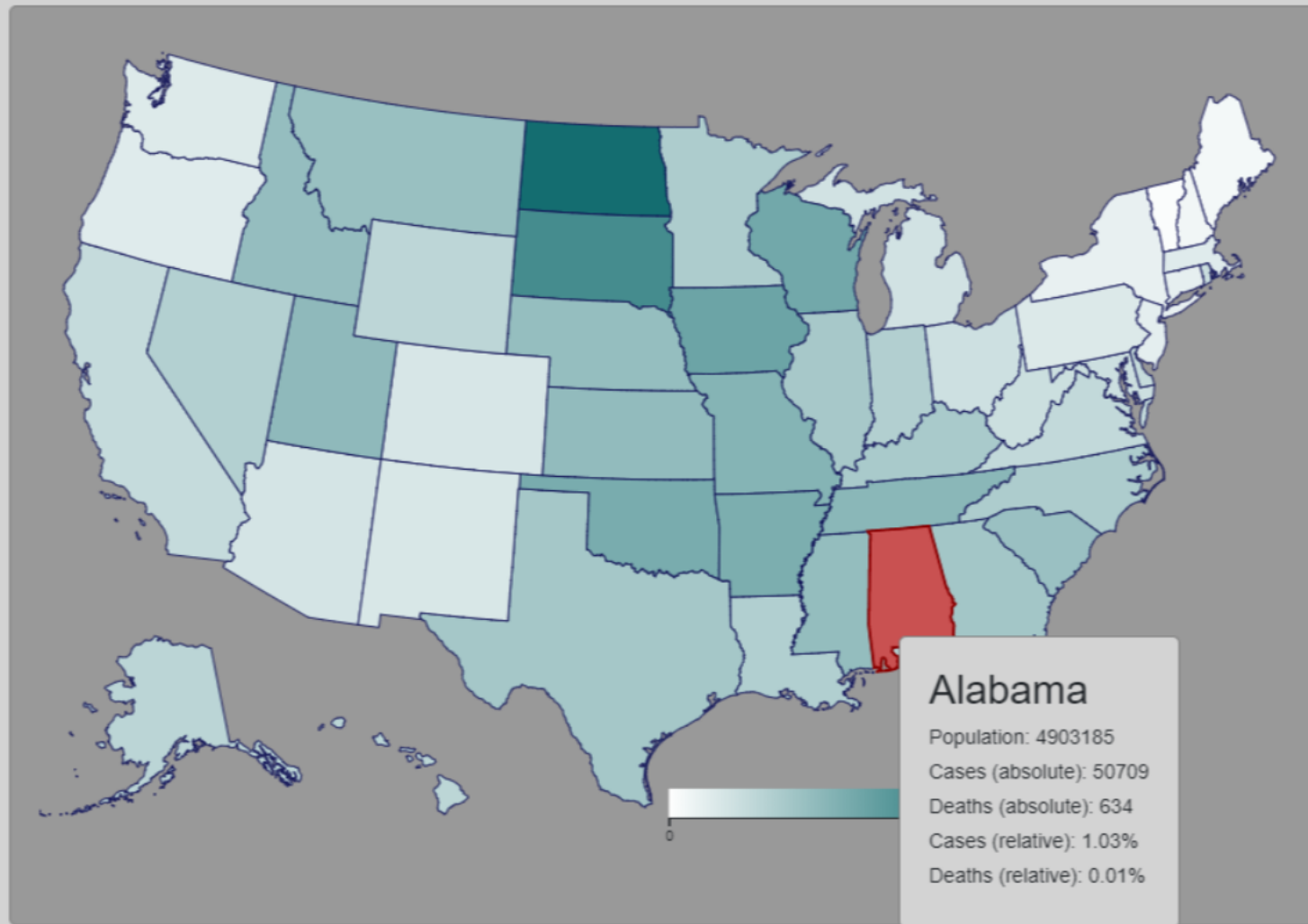
Option 2 —> Replace by 0:

```
myData.forEach(
  function(d){
    if (d.a == null) {d.a=0};
    if (d.b == null) {d.b=0};
    return d;
  })
```

**DESTRUCTIVE !!!**

# Homework

## Corona Virus Dashboard



map view

Cases (relative to populat ↕)

