Better Software Design with *Domain Modeling*

Eric Normand - Func Prog Sweden 2023

Software design is subtle

Good information → good decisions → good design

- Domain
- Data
- Operations
- Composition
- Volatility
- Scope
- Runnable specifications
- Time
- Architecture
- Stratified design

- Domain
- Data
- Operations
- Composition
- Volatility
- Scope
- Runnable specifications
- Time
- Architecture
- Stratified design

Data

Super Mega Galactic

Super Mega Galactic

Raw Burnt Charcoal

Super Mega Galactic

Raw Burnt Charcoal

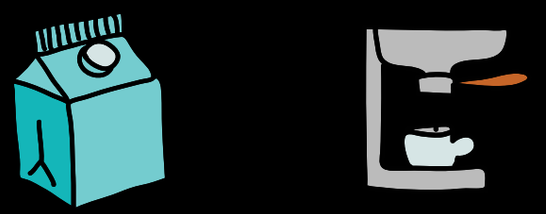
Soy milk

Espresso

Hazelnut Chocolate Almond

Super Mega Galactic

Raw Burnt Charcoal



Soy milk Espresso

Hazelnut Chocolate Almond

```
{
  “size”: “super”,
  “roast”: “burnt”,
  “add-ins”: [“espresso”, “soy”]
}
```

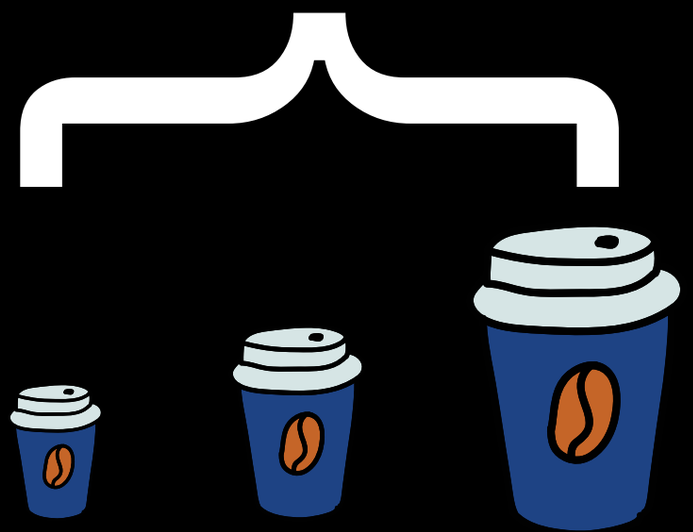
Super Mega Galactic

one of

Super Mega Galactic

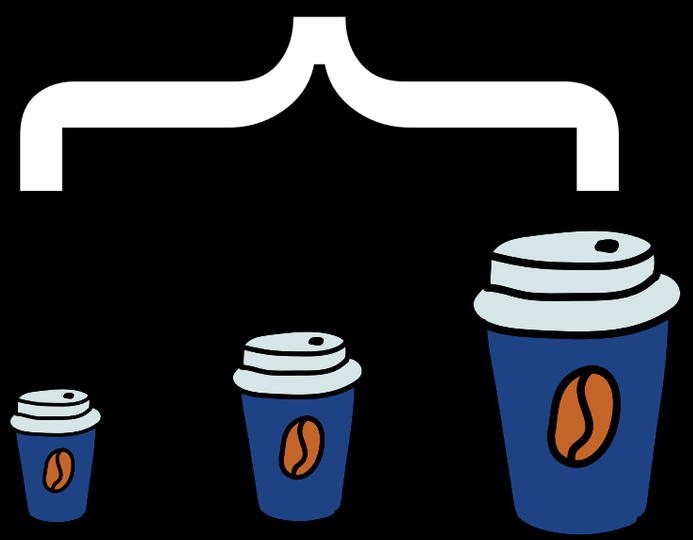
one of

alternative



Super Mega Galactic

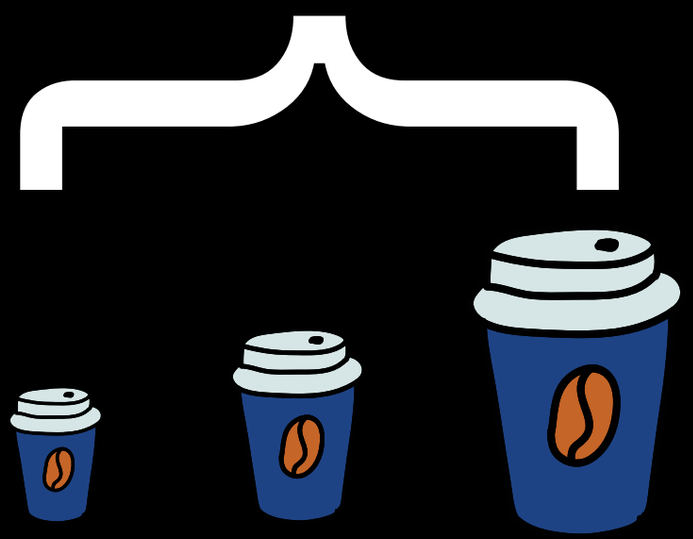
one of → alternative



Super Mega Galactic

“super” “mega” “galactic”

one of → alternative

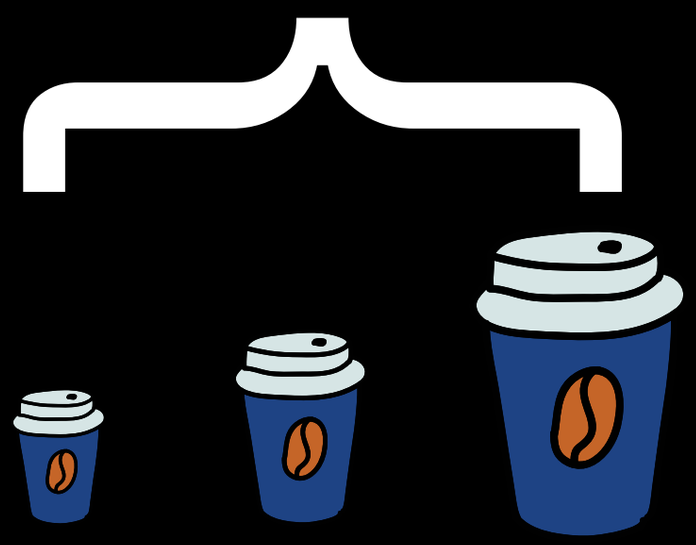


Super Mega Galactic

"super" "mega" "galactic"

one of

one of → alternative



Super Mega Galactic

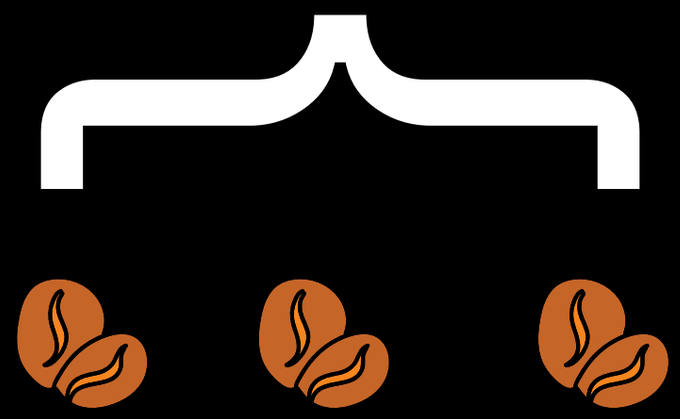
```
type Size = "super"    |
            "mega"     |
            "galactic";
```

“super” “mega” “galactic”

one of

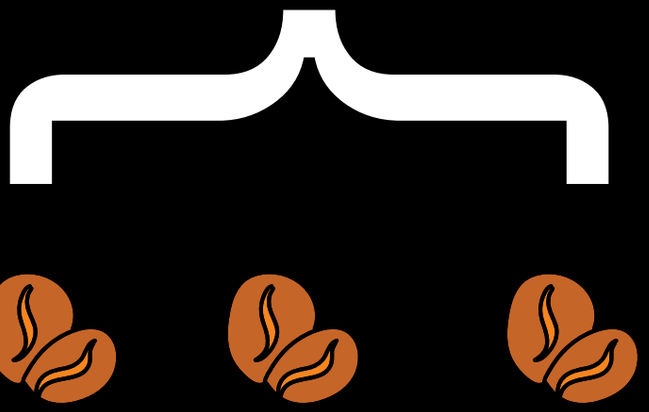
Raw Burnt Charcoal

one of



Raw Burnt Charcoal

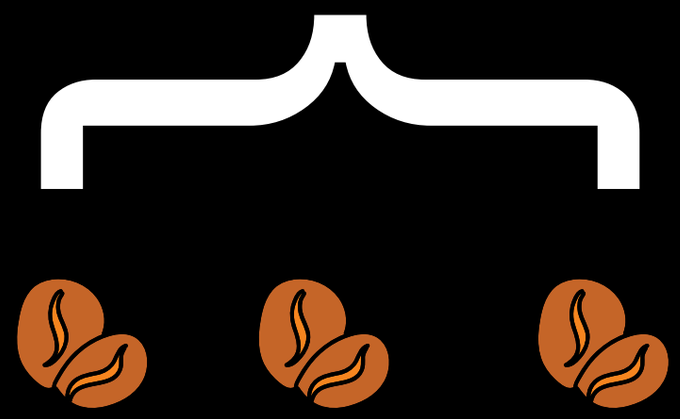
one of



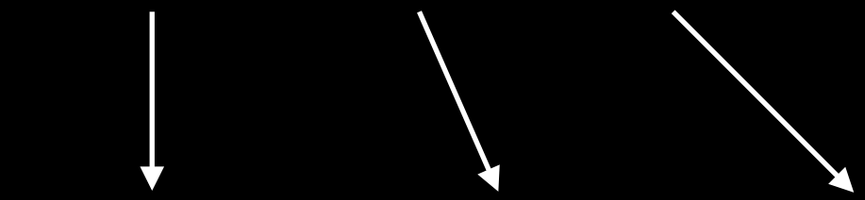
alternative

Raw Burnt Charcoal

one of → alternative



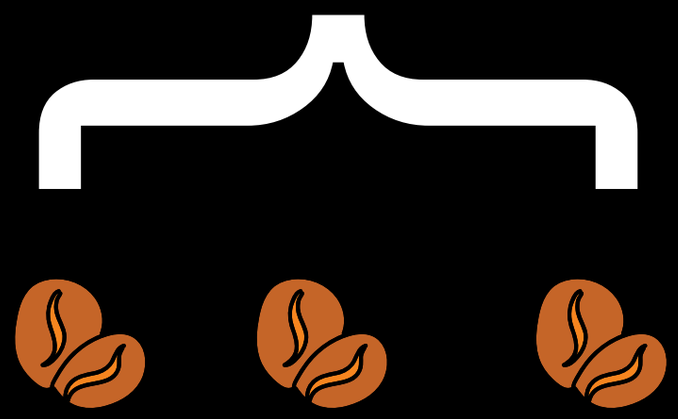
Raw Burnt Charcoal



```
“raw”  “burnt”  “charcoal”
```

one of

alternative

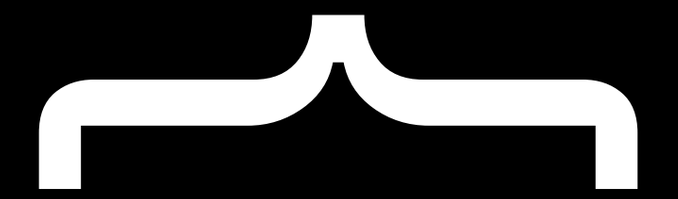


Raw Burnt Charcoal

“raw” “burnt” “charcoal”

one of

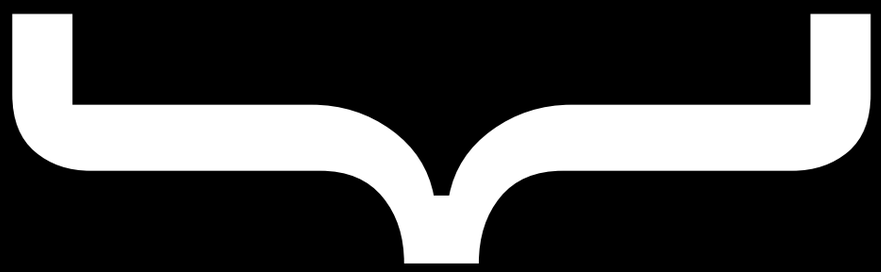
one of → alternative



Raw Burnt Charcoal

```
type Roast = "raw"      |
             "burnt"    |
             "charcoal";
```

“raw” “burnt” “charcoal”



one of

Soy milk Espresso

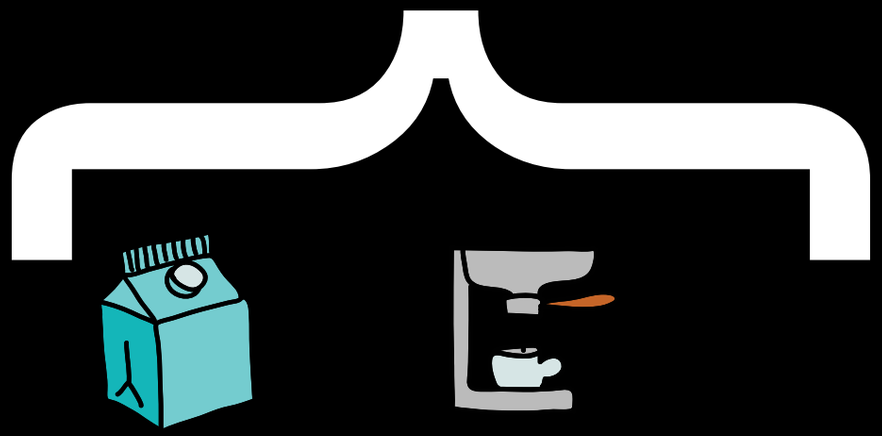
Hazelnut Chocolate Almond

one of

Soy milk Espresso

Hazelnut Chocolate Almond

one of

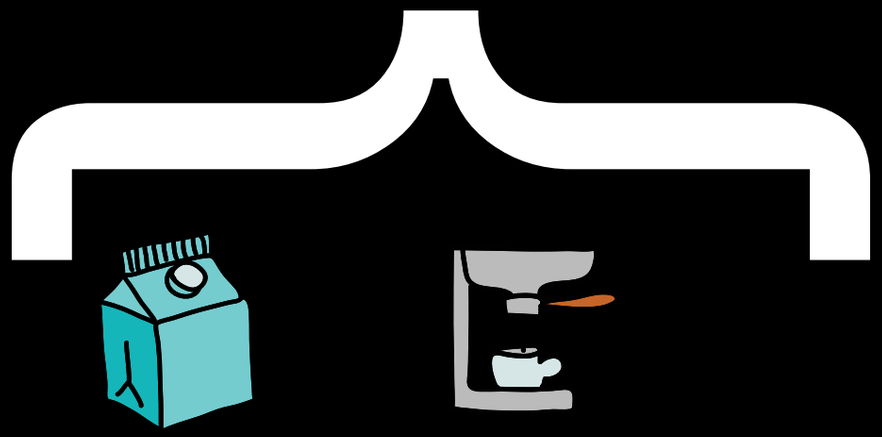


alternative

Soy milk Espresso

Hazelnut Chocolate Almond

one of → alternative

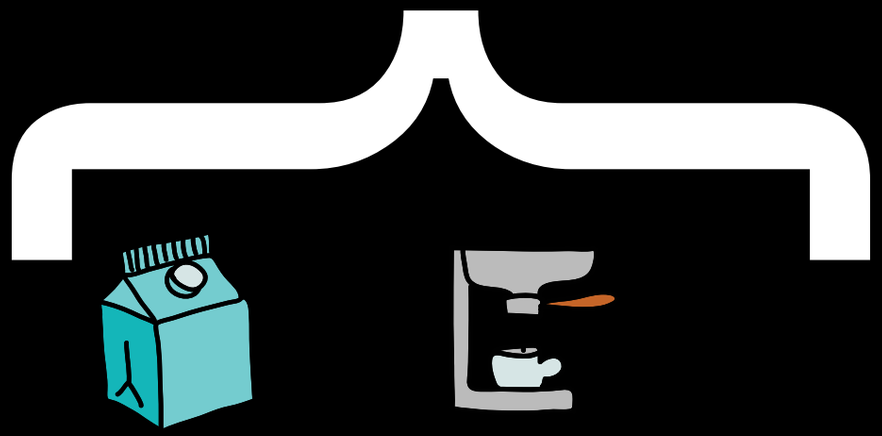


Soy milk Espresso

```
type AddIn = "soy"       |
             "espresso"  |
             "hazelnut"  |
             "chocolate" |
             "almond"    ;
```

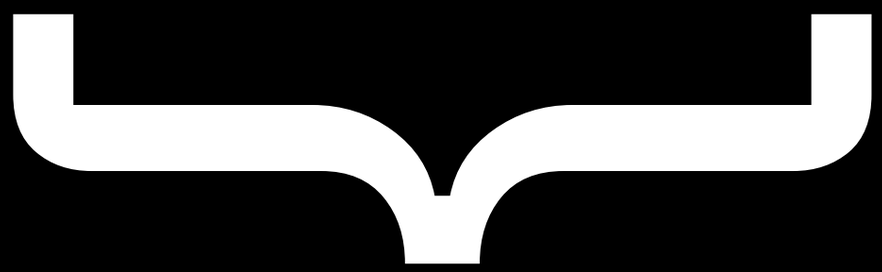
Hazelnut Chocolate Almond

one of



Soy milk Espresso

Hazelnut Chocolate Almond

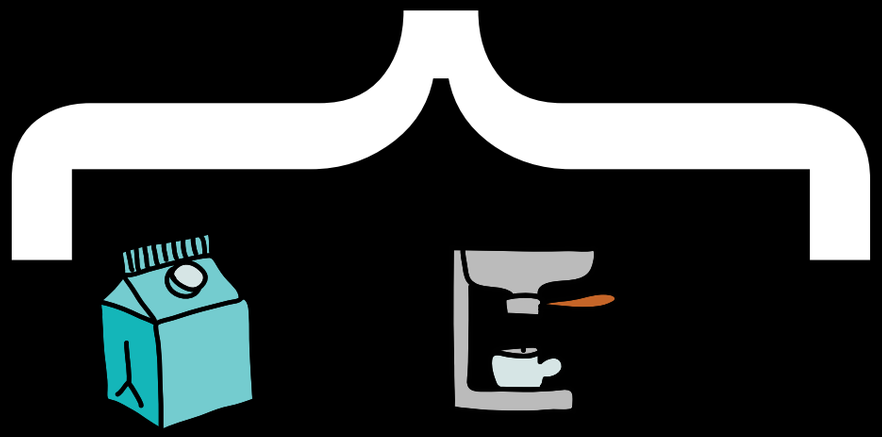


zero or more

alternative

```
type AddIn = "soy"       |
             "espresso"  |
             "hazelnut"  |
             "chocolate" |
             "almond"    ;
```

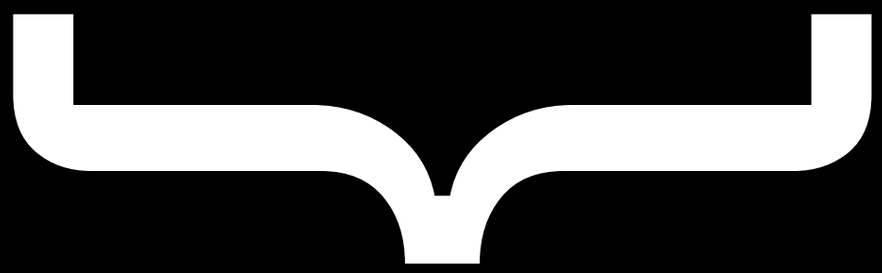
one of → alternative



Soy milk Espresso

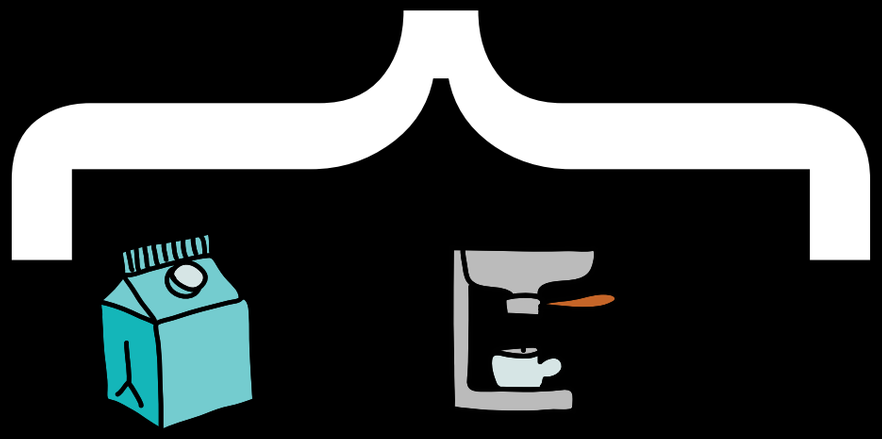
```
type AddIn = "soy"       |
             "espresso"  |
             "hazelnut"  |
             "chocolate" |
             "almond"    ;
```

Hazelnut Chocolate Almond



zero or more → collection

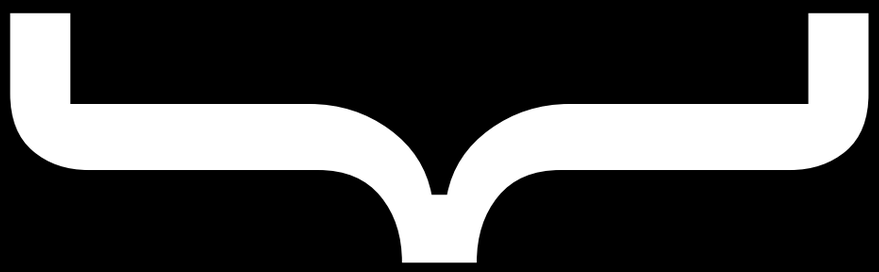
one of → alternative



Soy milk Espresso

```
type AddIn = "soy"       |
             "espresso"  |
             "hazelnut"  |
             "chocolate" |
             "almond"    ;
```

Hazelnut Chocolate Almond



zero or more → collection

```
type AddIns = AddIn[];
```

Super

Burnt

Soy milk

Espresso

all of

Super

Burnt

Soy milk

Espresso

combination

all of

Super

Burnt

Soy milk

Espresso

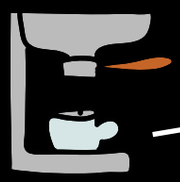
combination

all of

Super → `Size`

Burnt → `Roast`

Soy milk



Espresso → `AddIn[]`

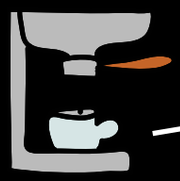
combination

all of

Super → Size

Burnt → Roast

Soy milk



Espresso → AddIn[]

```
type Coffee = {
  "size": Size;

  "roast": Roast;

  "add-ins": AddIn[];
};
```

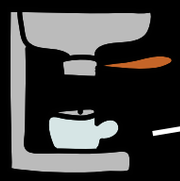
combination

all of

Super → Size

Burnt → Roast

Soy milk



Espresso → AddIn[]

```
type Coffee = {
  "size": Size;
  "roast": Roast;
  "add-ins": AddIn[];
};
```

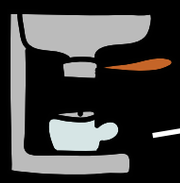
combination

all of

Super → Size

Burnt → Roast

Soy milk



Espresso → AddIn[]

```
type Coffee = {
  "size": Size;
  "roast": Roast;
  "add-ins": AddIn[];
};
```

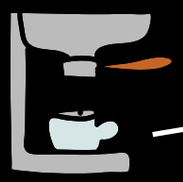
combination

all of

Super → Size → `"size": Size;`

Burnt → Roast → `"roast": Roast;`

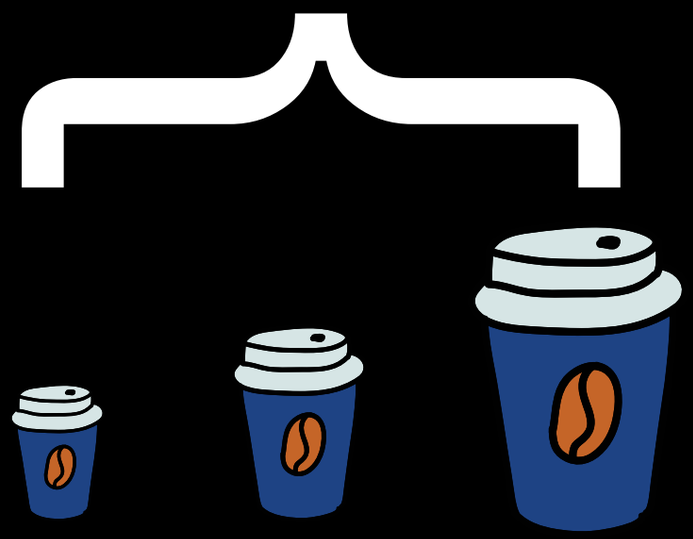
Soy milk



Espresso → `AddIn[]` → `"add-ins": AddIn[];`

```
type Coffee = {
  "size": Size;
  "roast": Roast;
  "add-ins": AddIn[];
};
```

one of → alternative



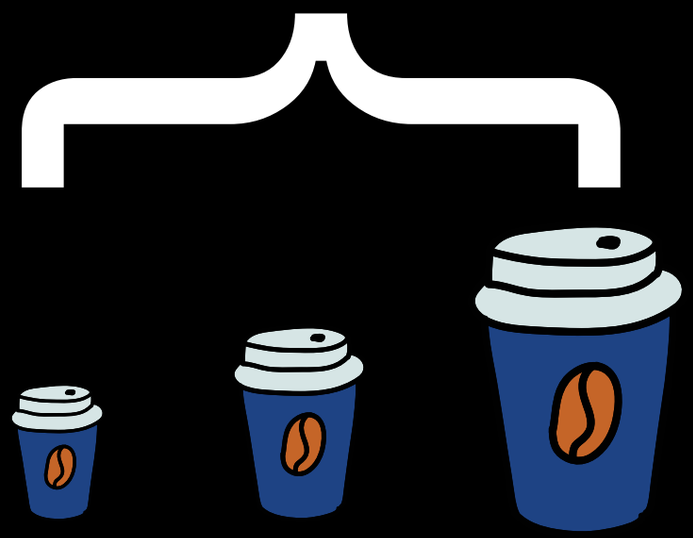
Super Mega Galactic

“super” “mega” “galactic”

one of

```
type Size = "super"    |
            "mega"     |
            "galactic";
```

one of → alternative



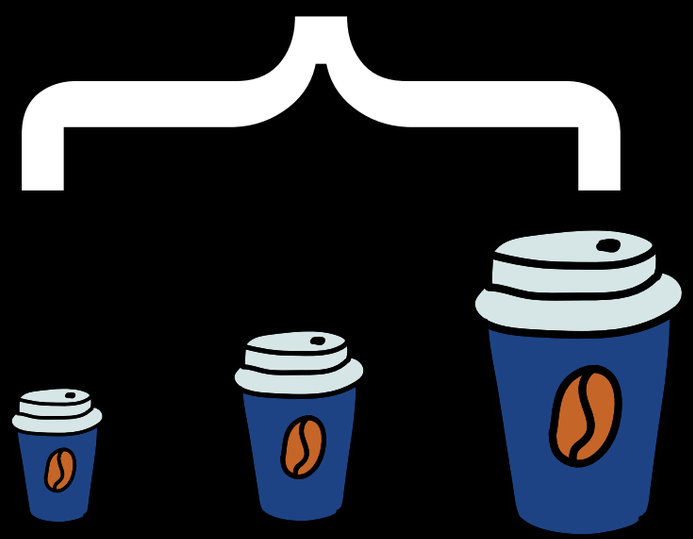
Super Mega Galactic

“super” “mega” “galactic”

one of

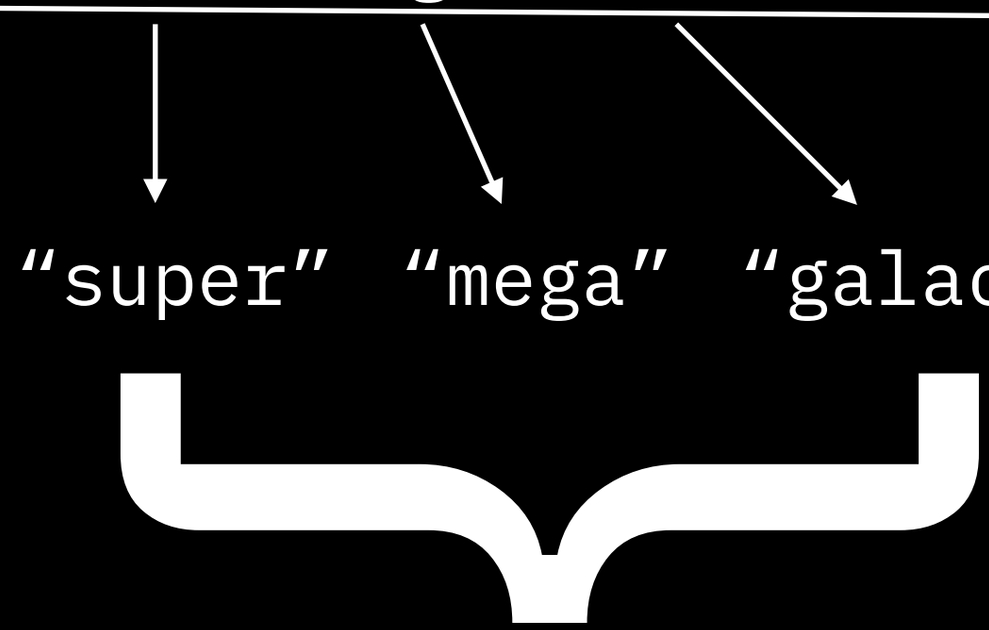
```
type Size = "super"    |
            "mega"     |
            "galactic";
```

one of → alternative



Super Mega Galactic

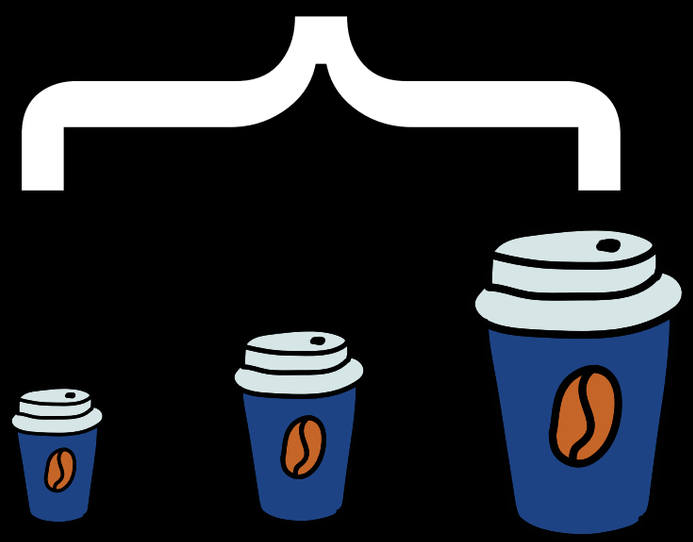
Model




```
type Size = "super"    |
            "mega"     |
            "galactic";
```

Model

one of → alternative



Super Mega Galactic

Code

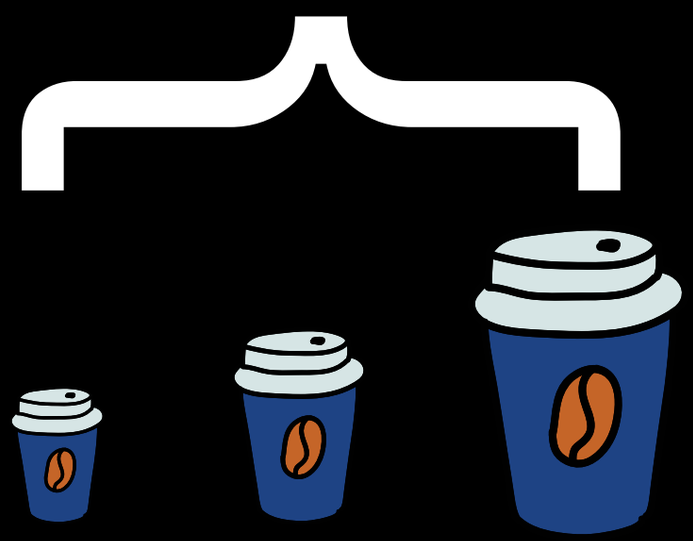
“super” “mega” “galactic”

one of

```
type Size = "super"    |
            "mega"     |
            "galactic";
```

Model

one of → alternative



Super Mega Galactic

Relationship

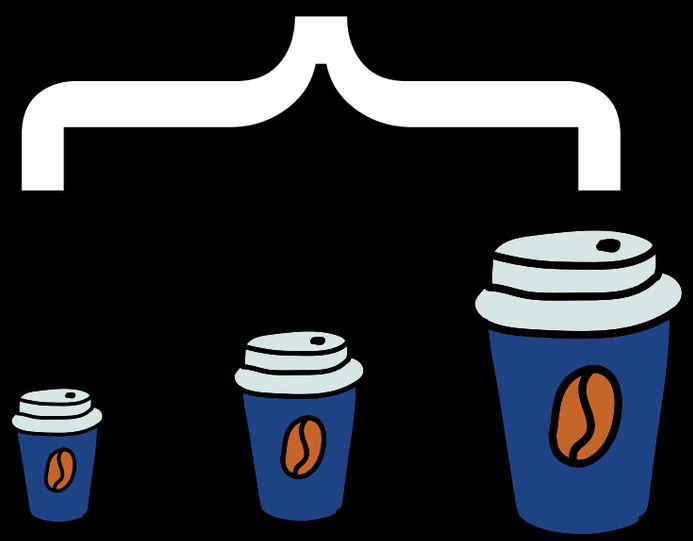
Code

“super” “mega” “galactic”

one of

```
type Size = "super"    |
            "mega"     |
            "galactic";
```

one of → alternative



Super Mega Galactic

Model

Relationship

Encoding

“super” “mega” “galactic”

one of

```
type Size = "super"    |
            "mega"     |
            "galactic";
```

Code

alternative	Relationship	Model
	Encoding	Code

alternative	Relationship	Model
	Encoding	Code
Strings + union type		

alternative	Relationship	Model
	Encoding	Code

Strings
+
union type
	`type Size = "super"  |`
	`"mega"  |`
	`"galactic";`	

alternative	Relationship	Model
	Encoding	Code
Strings + union type	`type Size = "super" | "mega" | "galactic";`	
Strings + enum		

alternative	Relationship	Model
	Encoding	Code
Strings + union type	`type Size = "super" | "mega" | "galactic";`	
Strings + enum	`enum Size { super = "super", mega = "mega", galactic = "galactic", }`	

alternative	Relationship	Model
	Encoding	Code
Strings + union type	`type Size = "super" | "mega" | "galactic";`	
Strings + enum	`enum Size { super = "super", mega = "mega", galactic = "galactic", }`	
Classes + Interface		

alternative	Relationship	Model
	Encoding	Code
Strings + union type	`type Size = "super" | "mega" | "galactic";`	
Strings + enum	`enum Size { super = "super", mega = "mega", galactic = "galactic", }`	
Classes + Interface	`interface Size { name: string; }`	

alternative	Relationship	Model
	Encoding	Code
Strings + union type	`type Size = "super" | "mega" | "galactic";`	
Strings + enum	`enum Size { super = "super", mega = "mega", galactic = "galactic", }`	
Classes + Interface	`interface Size { name: string; }`	`class Super implements Size { name = “super” }`

alternative	Relationship	Model
	Encoding	Code

Strings + union type

```
type Size = "super"   |
            "mega"    |
            "galactic";
```

Strings + enum

```
enum Size {
    super    = "super",
    mega     = "mega",
    galactic = "galactic",
}
```

Classes + Interface

```
interface Size {
  name: string;
}
```

```
class Super implements Size {
  name = “super”
}
```

```
class Mega implements Size {
  name = “mega”
}
```

alternative	Relationship	Model
	Encoding	Code

Strings + union type

```
type Size = "super"   |
            "mega"    |
            "galactic";
```

Strings + enum

```
enum Size {
    super    = "super",
    mega     = "mega",
    galactic = "galactic",
}
```

Classes + Interface

```
interface Size {
  name: string;
}
```

```
class Super implements Size {
  name = “super”
}
```

```
class Mega implements Size {
  name = “mega”
}
```

```
class Galactic implements Size {
  name = “galactic”
}
```

Fit

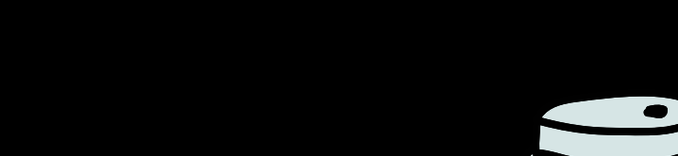
Super Mega Galactic

x

Raw Burnt Charcoal


Super
Mega
Galactic

x


Raw
Burnt
Charcoal


Raw
Burnt
Charcoal
Raw
Burnt
Charcoal
Raw
Burnt
Charcoal

```
type Coffee = {
  "size": Size;
  "roast": Roast;
};
```

```
{
 “size”  : “super”,
 “roast” : “raw”
}
```

```
type Coffee = {
  "size": Size;
  "roast": Roast;
};
```

```
{
 “size”  : “super”,
 “roast” : “raw”
}
```

```
{
 “size”  : “super”,
 “roast” : “burnt”
}
```

```
type Coffee = {
  "size": Size;
  "roast": Roast;
};
```

```
{
 “size”  : “super”,
 “roast” : “raw”
}
```

```
{
 “size”  : “super”,
 “roast” : “burnt”
}
```

```
{
 “size”  : “super”,
 “roast” : “charcoal”
}
```

```
type Coffee = {
  "size": Size;
  "roast": Roast;
};
```

```
{
 “size”  : “super”,
 “roast” : “raw”
}
```

```
{
 “size”  : “super”,
 “roast” : “burnt”
}
```

```
{
 “size”  : “super”,
 “roast” : “charcoal”
}
```

```
type Coffee = {
  "size": Size;
  "roast": Roast;
};
```

```
{
 “size”  : “mega”,
 “roast” : “raw”
}
```

```
{
 “size”  : “super”,
 “roast” : “raw”
}
```

```
{
 “size”  : “super”,
 “roast” : “burnt”
}
```

```
{
 “size”  : “super”,
 “roast” : “charcoal”
}
```

```
type Coffee = {
  "size": Size;
  "roast": Roast;
};
```

```
{
 “size”  : “mega”,
 “roast” : “raw”
}
```

```
{
 “size”  : “mega”,
 “roast” : “burnt”
}
```

```
type Coffee = {
  "size": Size;
  "roast": Roast;
};
```

```
{
 “size”  : “super”,
 “roast” : “raw”
}
```

```
{
 “size”  : “super”,
 “roast” : “burnt”
}
```

```
{
 “size”  : “super”,
 “roast” : “charcoal”
}
```

```
{
 “size”  : “mega”,
 “roast” : “raw”
}
```

```
{
 “size”  : “mega”,
 “roast” : “burnt”
}
```

```
{
 “size”  : “mega”,
 “roast” : “charcoal”
}
```

```
type Coffee = {
  "size": Size;
  "roast": Roast;
};
```

```
{
 “size”  : “super”,
 “roast” : “raw”
}
```

```
{
 “size”  : “super”,
 “roast” : “burnt”
}
```

```
{
 “size”  : “super”,
 “roast” : “charcoal”
}
```

```
{
 “size”  : “mega”,
 “roast” : “raw”
}
```

```
{
 “size”  : “mega”,
 “roast” : “burnt”
}
```

```
{
 “size”  : “mega”,
 “roast” : “charcoal”
}
```

```
{
 “size”  : “galactic”,
 “roast” : “raw”
}
```

```
type Coffee = {
  "size": Size;
  "roast": Roast;
};
```

```
{
 “size”  : “super”,
 “roast” : “raw”
}
```

```
{
 “size”  : “super”,
 “roast” : “burnt”
}
```

```
{
 “size”  : “super”,
 “roast” : “charcoal”
}
```

```
{
 “size”  : “mega”,
 “roast” : “raw”
}
```

```
{
 “size”  : “mega”,
 “roast” : “burnt”
}
```

```
{
 “size”  : “mega”,
 “roast” : “charcoal”
}
```

```
{
 “size”  : “galactic”,
 “roast” : “raw”
}
```

```
{
 “size”  : “galactic”,
 “roast” : “burnt”
}
```

```
type Coffee = {
  "size": Size;
  "roast": Roast;
};
```

```
{
 “size”  : “super”,
 “roast” : “raw”
}
```

```
{
 “size”  : “super”,
 “roast” : “burnt”
}
```

```
{
 “size”  : “super”,
 “roast” : “charcoal”
}
```

```
{
 “size”  : “mega”,
 “roast” : “raw”
}
```

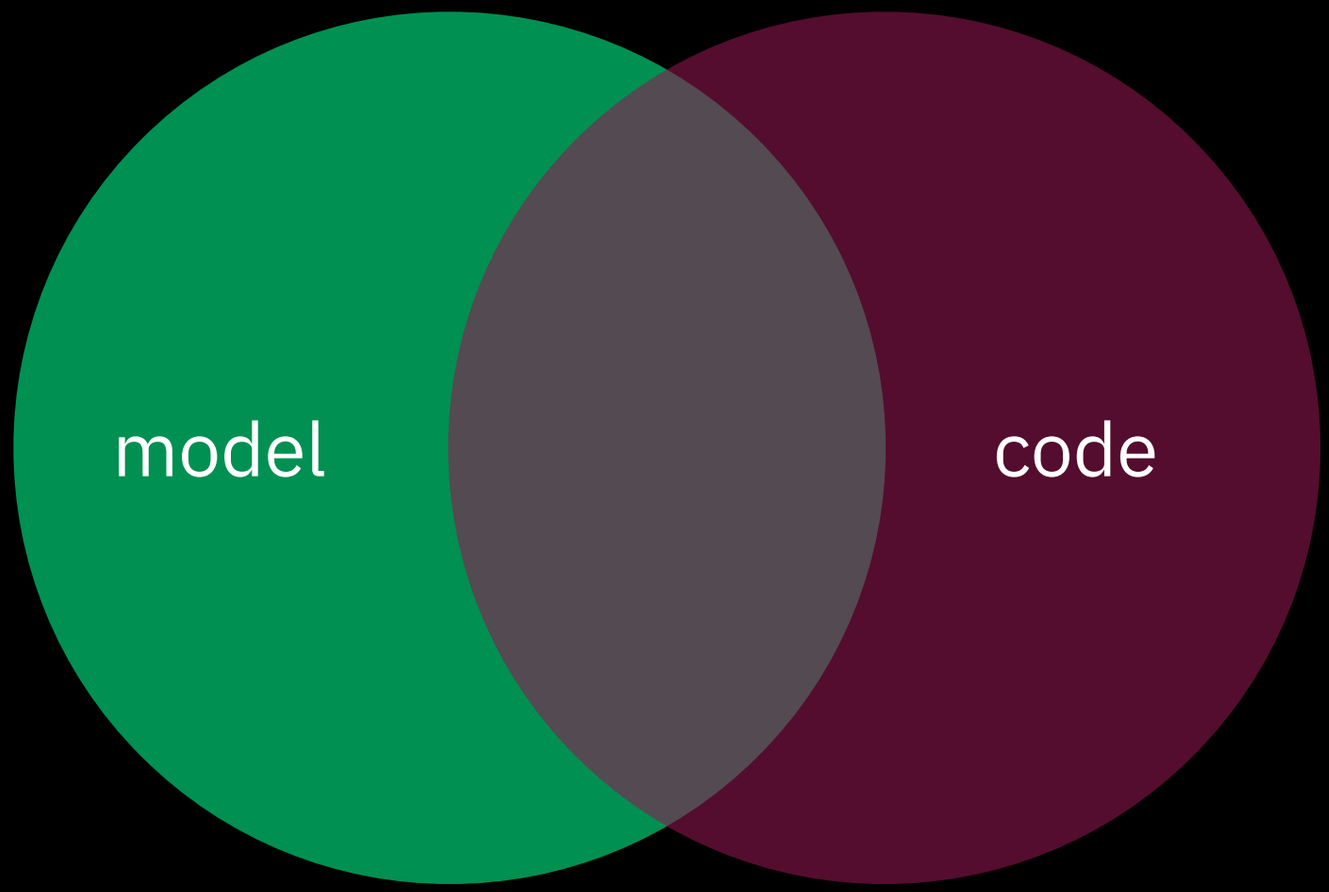
```
{
 “size”  : “mega”,
 “roast” : “burnt”
}
```

```
{
 “size”  : “mega”,
 “roast” : “charcoal”
}
```

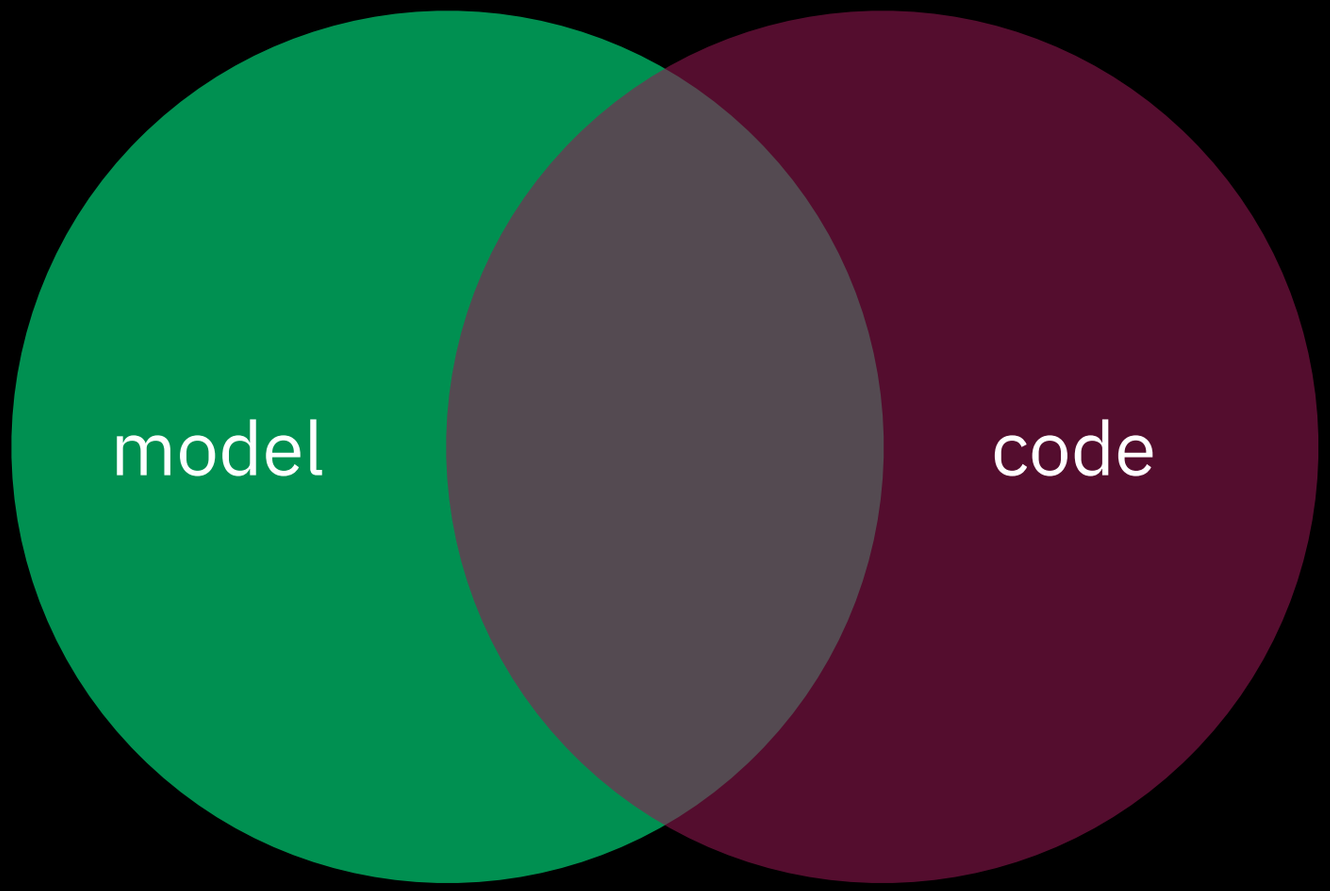
```
{
 “size”  : “galactic”,
 “roast” : “raw”
}
```

```
{
 “size”  : “galactic”,
 “roast” : “burnt”
}
```

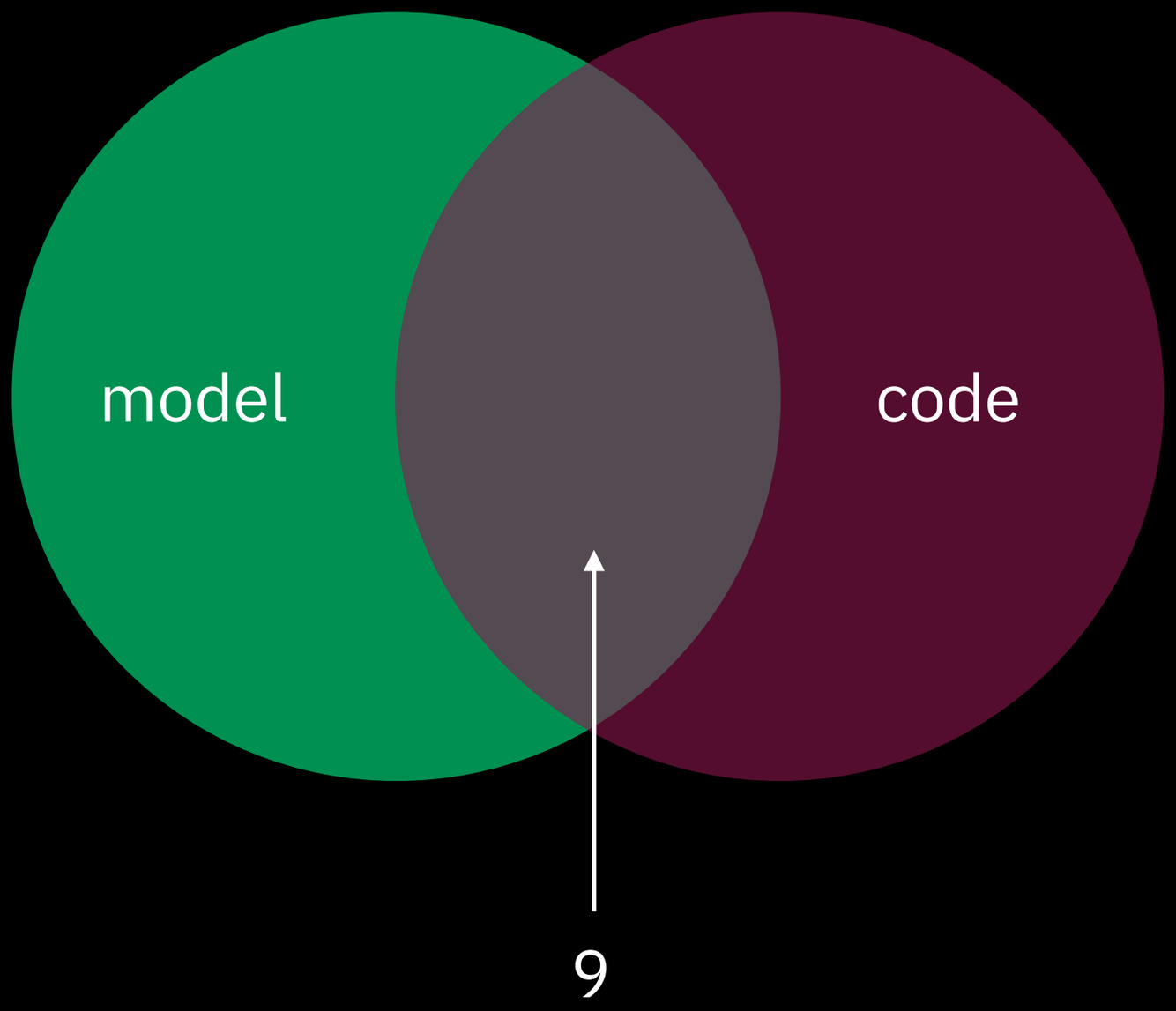
```
{
 “size”  : “galactic”,
 “roast” : “charcoal”
}
```


model
code

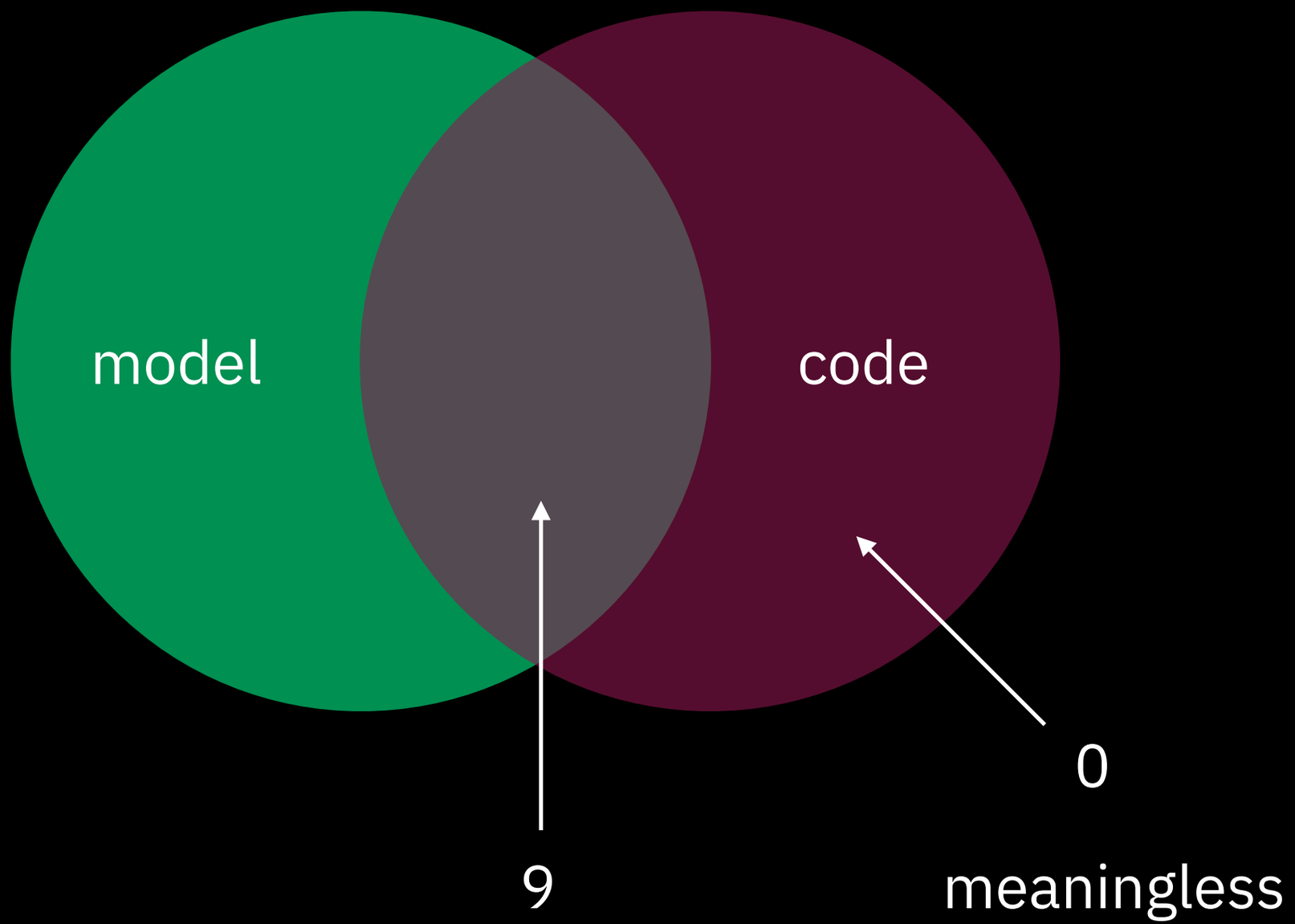
size x roast



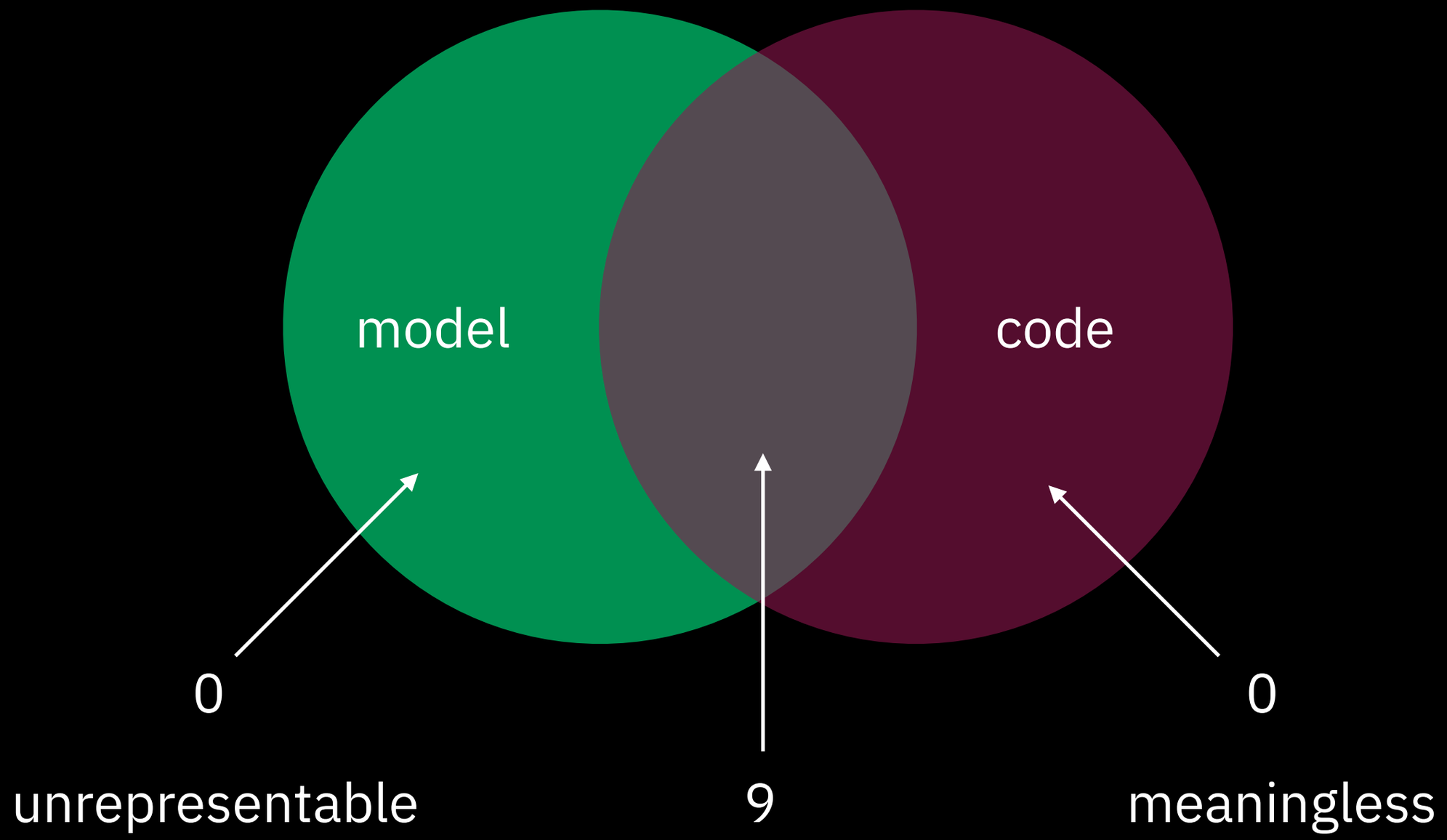
size x roast



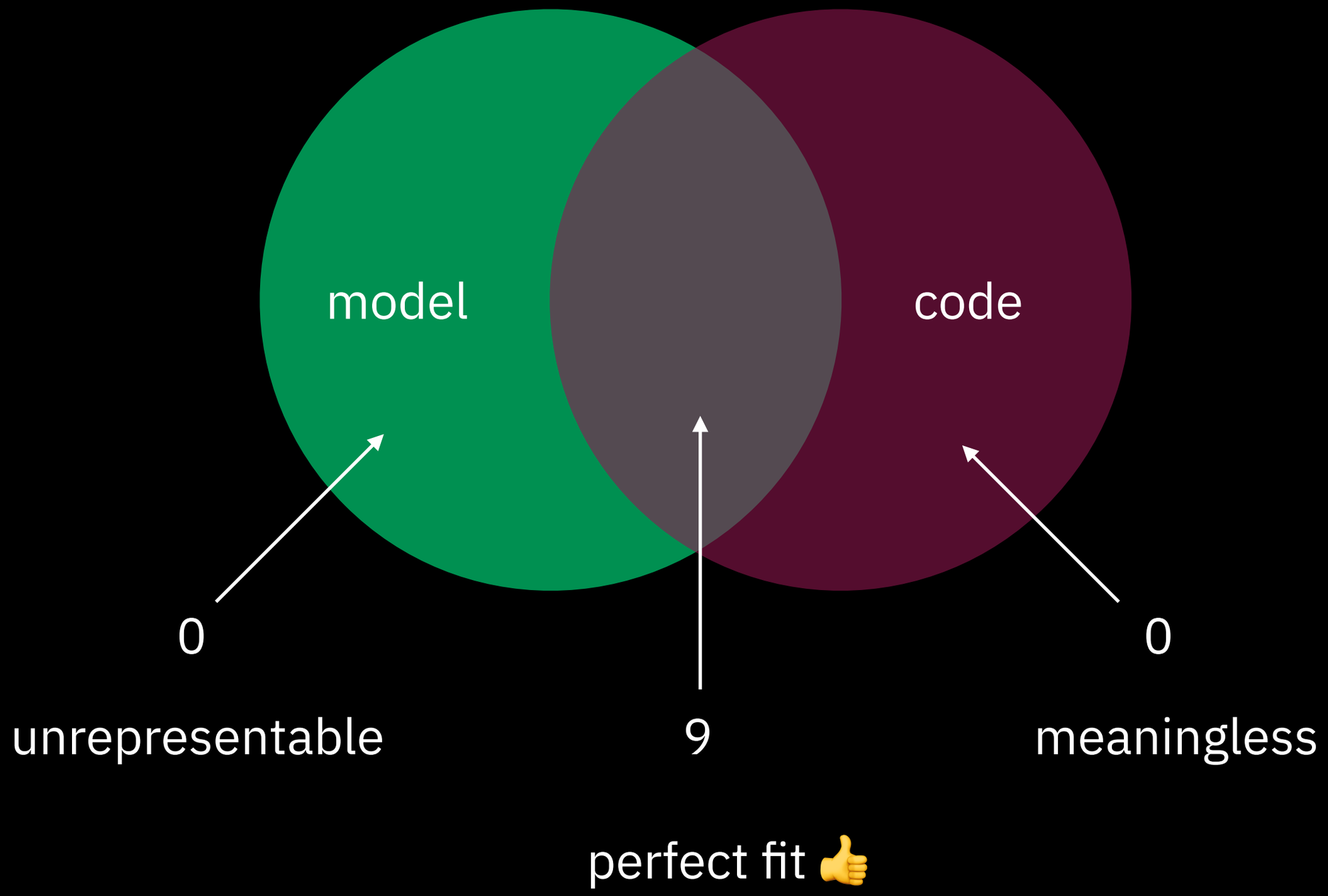
size x roast



size x roast



size x roast



perfect fit 👍


Super
Mega
Galactic

x


Raw
Burnt
Charcoal


1
Raw
2
Burnt
3
Charcoal
4
Raw
5
Burnt
6
Charcoal
7
Raw
8
Burnt
9
Charcoal





Soy milk Espresso Hazelnut Chocolate Almond

How many combinations are there? (up to 3 add-ins)





Soy milk Espresso Hazelnut Chocolate Almond

How many combinations are there? (up to 3 add-ins)

for 0 add-ins

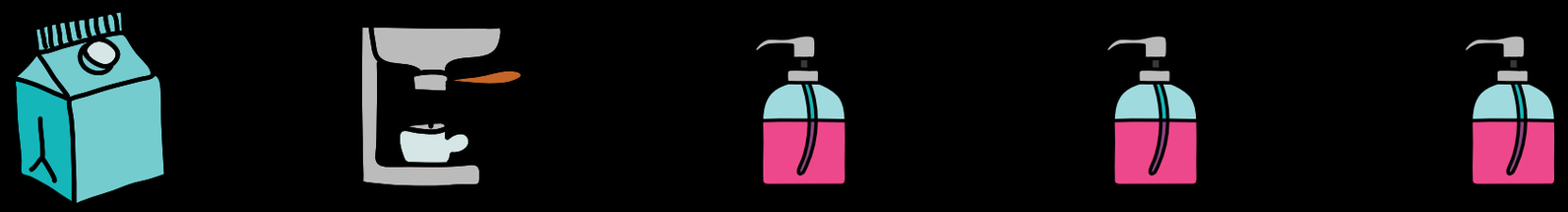




Soy milk Espresso Hazelnut Chocolate Almond

How many combinations are there? (up to 3 add-ins)

for 0 add-ins []

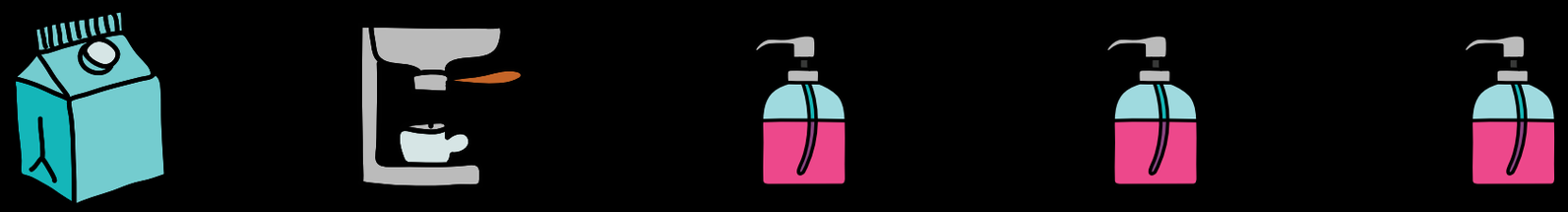


Soy milk Espresso Hazelnut Chocolate Almond

How many combinations are there? (up to 3 add-ins)

for 0 add-ins [ ]

for 1 add-in



Soy milk Espresso Hazelnut Chocolate Almond

How many combinations are there? (up to 3 add-ins)

for 0 add-ins `[]`

for 1 add-in `[“soy”] [“almond”] …`





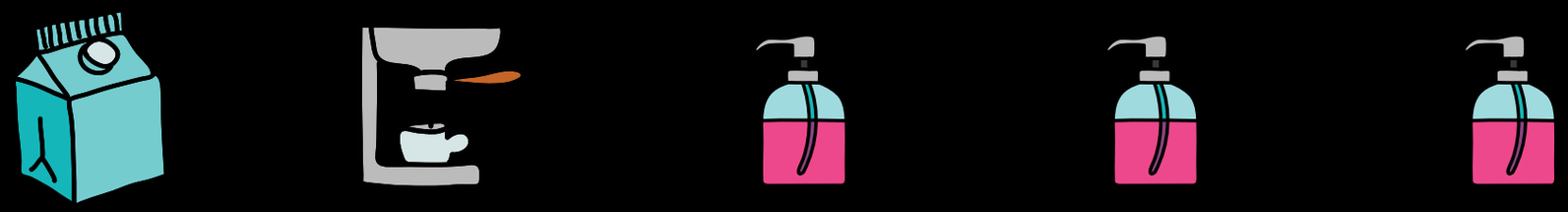
Soy milk Espresso Hazelnut Chocolate Almond

How many combinations are there? (up to 3 add-ins)

for 0 add-ins [ ]

for 1 add-in `[“soy”] [“almond”]` …

for 2 add-ins



Soy milk Espresso Hazelnut Chocolate Almond

How many combinations are there? (up to 3 add-ins)

for 0 add-ins `[]`

for 1 add-in `[“soy”] [“almond”] …`

for 2 add-ins `[“soy”, “espresso”]`
`[“espresso”, “soy”] …`





Soy milk Espresso Hazelnut Chocolate Almond

How many combinations are there? (up to 3 add-ins)

for 0 add-ins []

for 1 add-in `[“soy”] [“almond”] …`

for 2 add-ins `[“soy”, “espresso”]`
`[“espresso”, “soy”] …`

for 3 add-ins





Soy milk Espresso Hazelnut Chocolate Almond

How many combinations are there? (up to 3 add-ins)

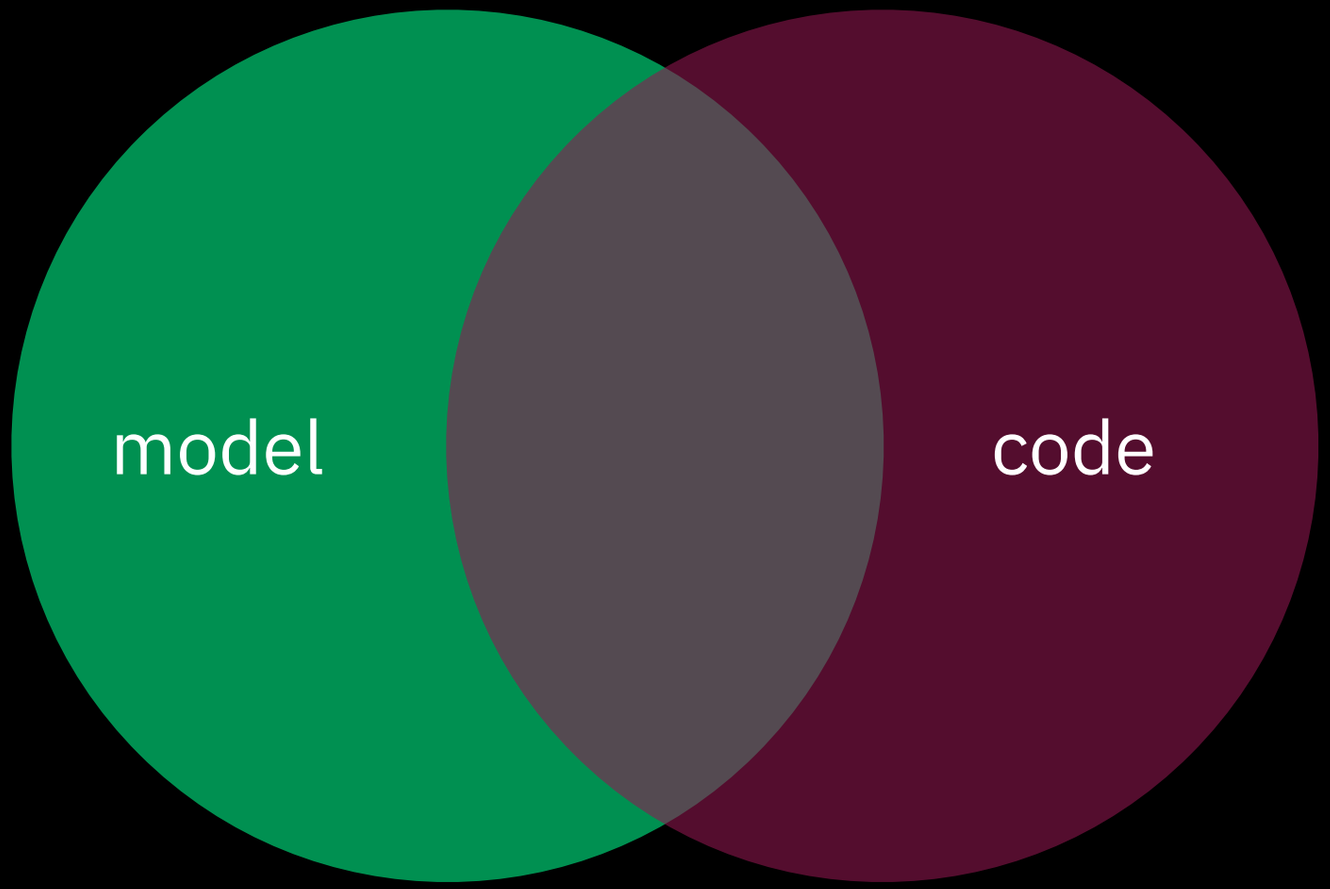
for 0 add-ins []

for 1 add-in ["soy"] ["almond"] …

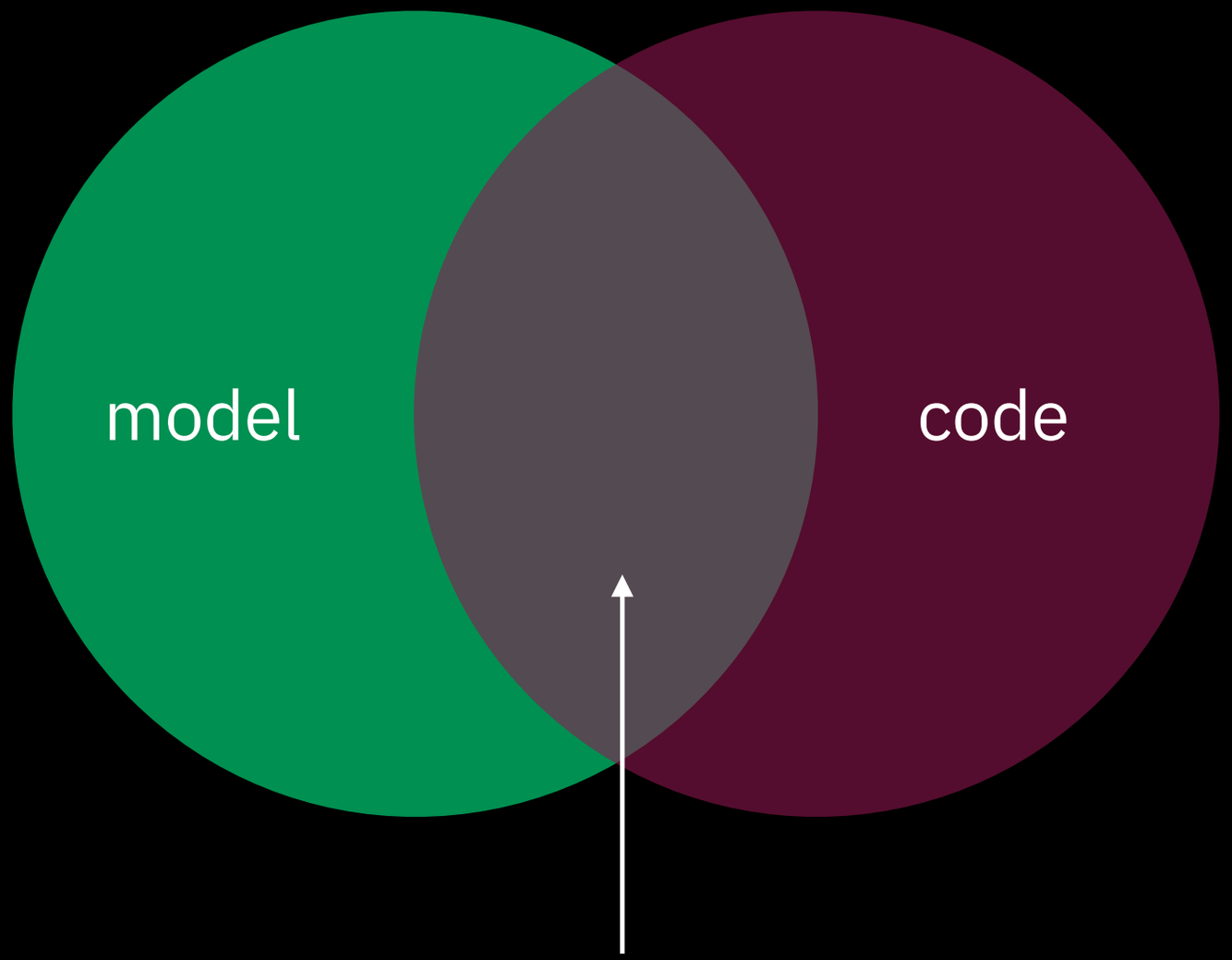
for 2 add-ins ["soy", "espresso"]
["espresso", "soy"] …

for 3 add-ins ["soy", "soy", "almond"]
["almond", "soy", "soy"]
["soy", "almond", "soy"] …

add-ins

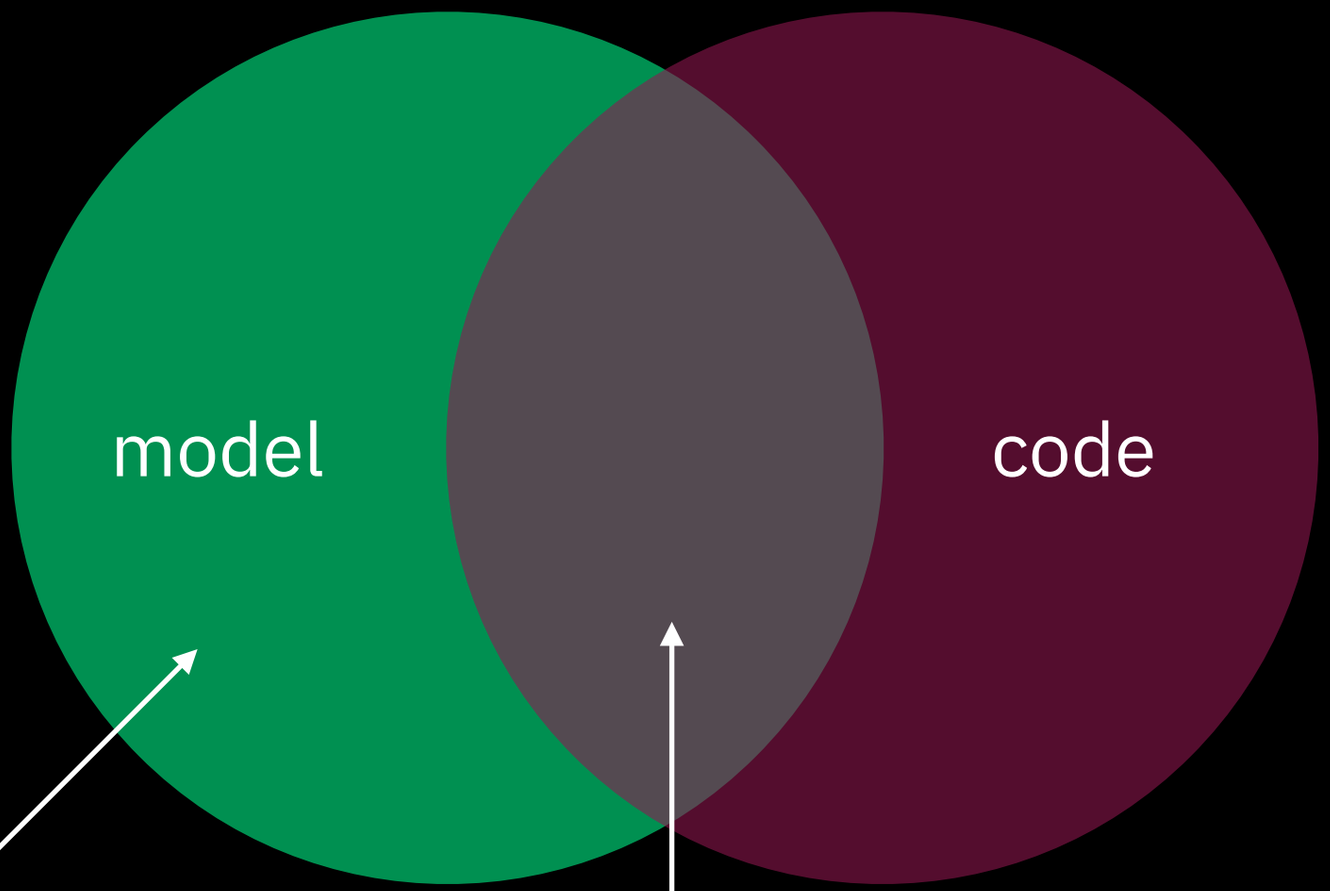


add-ins



can represent
everything

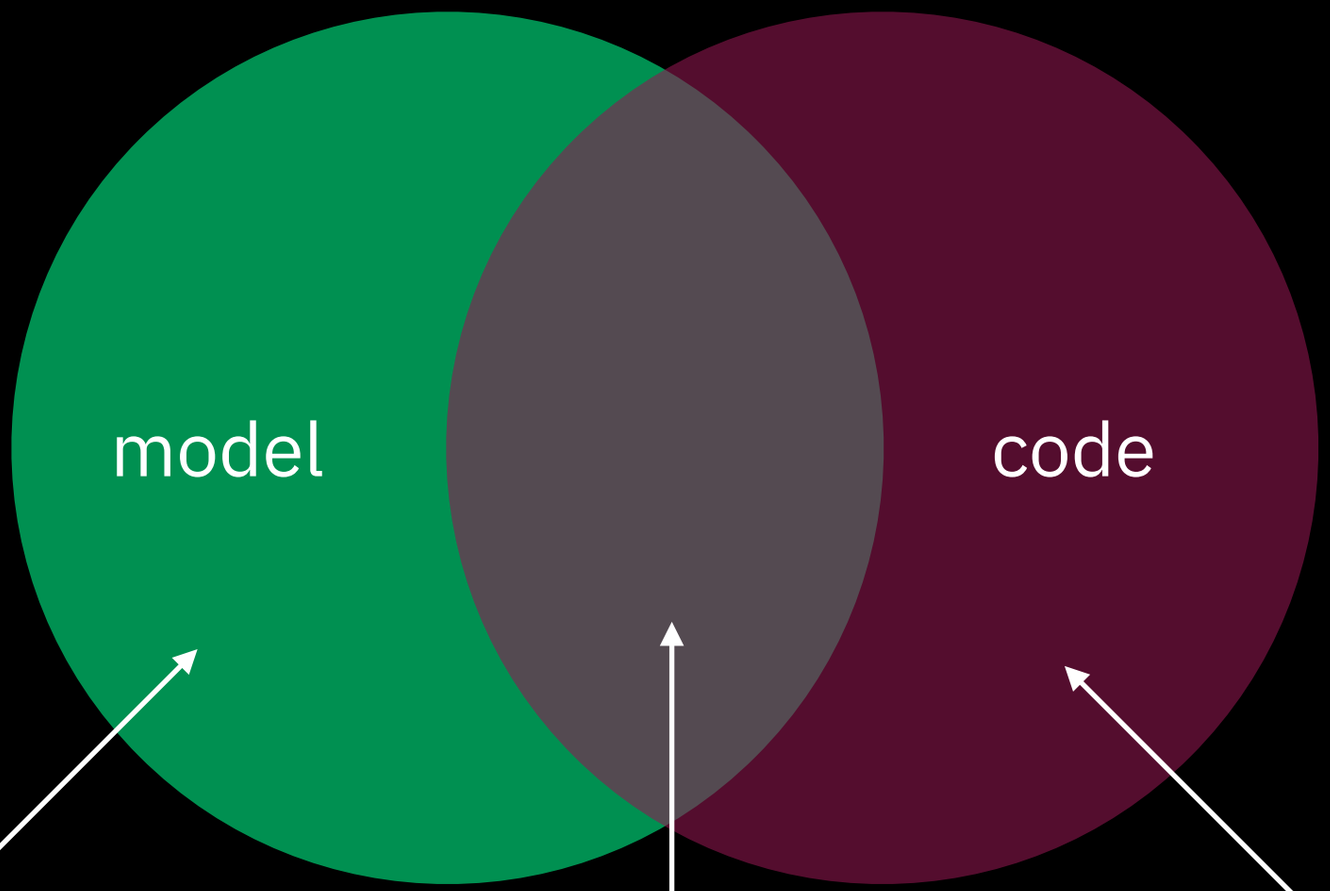
add-ins



0
unrepresentable

can represent
everything

add-ins

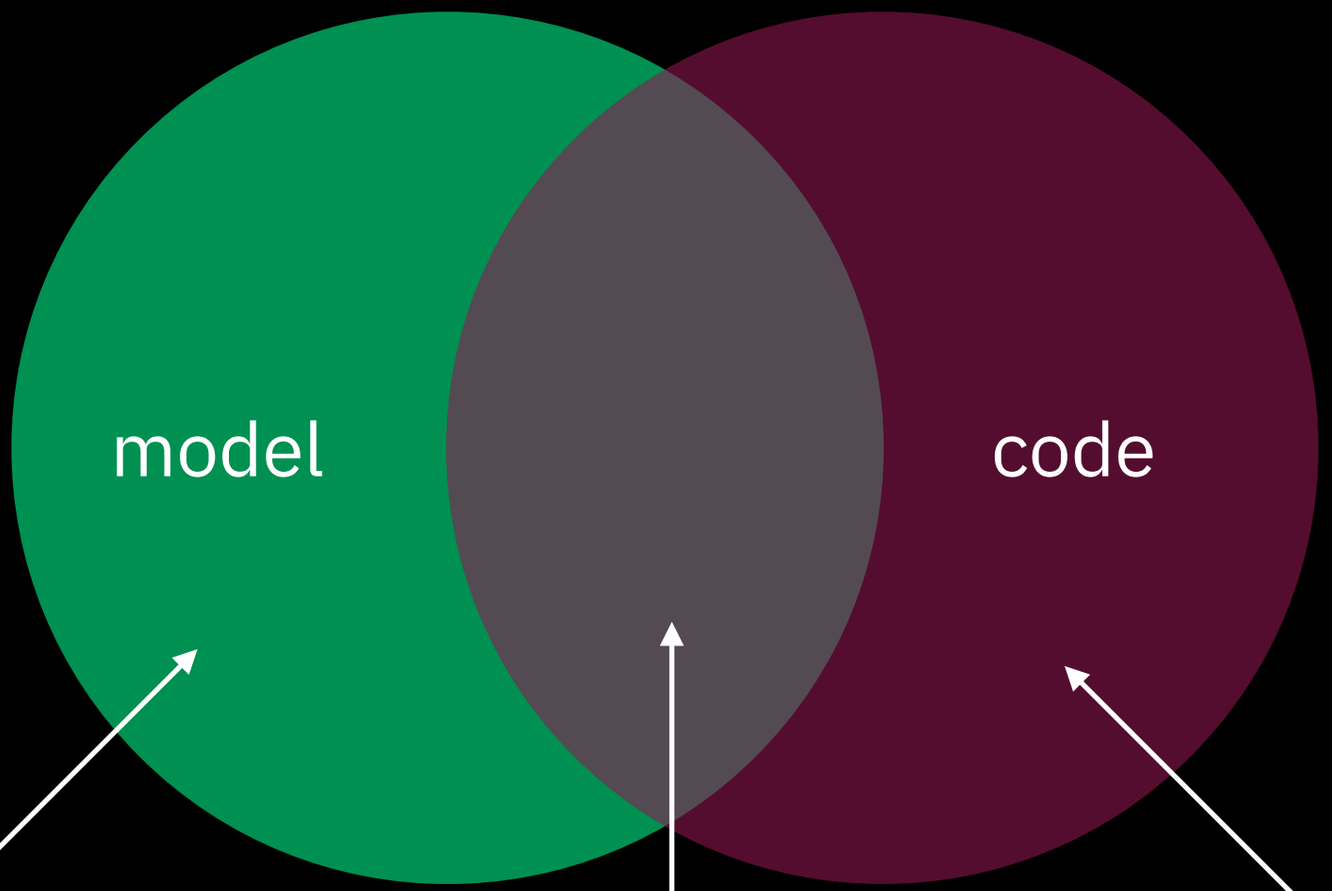


0
unrepresentable

can represent
everything

meaningless
distinctions

add-ins



0
unrepresentable

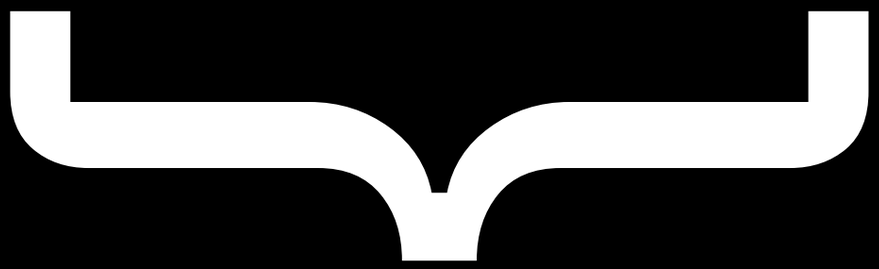
can represent
everything

meaningless
distinctions

less than perfect fit

Soy milk Espresso

Hazelnut Chocolate Almond



zero or more → collection

```
type AddIns = AddIn[];
```

collection	Relationship	Model
	Encoding	Code

collection	Relationship	Model
	Encoding	Code
Array		

collection	Relationship	Model
	Encoding	Code
Array	`type AddIns = AddIn[];`	

collection	Relationship	Model
	Encoding	Code
Array	`type AddIns = AddIn[];`	
Set		

collection	Relationship Encoding	Model Code
Array	`type AddIns = AddIn[];`	
Set	Won’t work! No duplicates.	

collection	Relationship Encoding	Model Code
Array	`type AddIns = AddIn[];`	
Set	Won’t work! No duplicates.	
Object		

collection	Relationship Encoding	Model Code
Array	`type AddIns = AddIn[];`	
Set	Won’t work! No duplicates.	

Object	`type AddIns = {`
	`[addIn: string] : number;`
	`};`	

collection	Relationship Encoding	Model Code
Array	`type AddIns = AddIn[];`	
Set	Won’t work! No duplicates.	

Object	`type AddIns = {`
	`[addIn: string] : number;`
	`};`	

```
{
  “almond”: 1,
  “soy”: 2
}
```

collection	Relationship Encoding	Model Code
Array	`type AddIns = AddIn[];`	
Set	Won’t work! No duplicates.	

Object	`type AddIns = {`
	`  [addIn: string] : number;`
	`};`	

make sure key is one of the accepted add-in strings

```
{
  “almond”: 1,
  “soy”: 2
}
```

Array

Object

Array

Object

[]

Array

Object

[]

{}

Array

```
[]
```

Object

```
{}

{“espresso”: 1}
```

Array

```
[]

[“espresso”]
```

Object

```
{}

{“espresso”: 1}
```

Array	Object
`[]`	`{}`
`[“espresso”]`	`{“espresso”: 1}`
`[“espresso”, “espresso”]`	

Array	Object
`[]`	`{}`
`[“espresso”]`	`{“espresso”: 1}`
`[“espresso”, “espresso”]`	`{“espresso”: 2}`

Array	Object
`[]`	`{}`
`[“espresso”]`	`{“espresso”: 1}`
`[“espresso”, “espresso”]`	`{“espresso”: 2,` `“soy”: 1}`

Array	Object
`[]`	`{}`
`[“espresso”]`	`{“espresso”: 1}`
`[“espresso”, “espresso”]`	`{“espresso”: 2}`

`[“espresso”,`
` “soy,`
` “espresso”]`
	`{“espresso”: 2,`
	` “soy”: 1}`

Array	Object
`[]`	`{}`
`[“espresso”]`	`{“espresso”: 1}`
`[“espresso”, “espresso”]`	`{“espresso”: 2}`

`[“espresso”,`
` “soy,`
` “espresso”]`
	`{“espresso”: 2,`
	` “soy”: 1}`

Normalize by sorting

Array	Object
`[]`	`{}`
`[“espresso”]`	`{“espresso”: 1}`
`[“espresso”, “espresso”]`	`{“espresso”: 2}`

`[“espresso”,`
`“soy,`
`“espresso”]`
	`{“espresso”: 2,`
	`“soy”: 1}`

Normalize by sorting

```
[“espresso”,
 “espresso”]
```

Array	Object
`[]`	`{}`
`[“espresso”]`	`{“espresso”: 1}`
`[“espresso”, “espresso”]`	`{“espresso”: 2}`

`[“espresso”,`
`“soy,`
`“espresso”]`
	`{“espresso”: 2,`
	`“soy”: 1}`

Normalize by sorting

`[“espresso”,` `“espresso”]`	`{“espresso”: 2,` `“soy”: 0}`

Array	Object
`[]`	`{}`
`[“espresso”]`	`{“espresso”: 1}`
`[“espresso”, “espresso”]`	`{“espresso”: 2}`
`[“espresso”, “soy, “espresso”]`	`{“espresso”: 2, “soy”: 1}`
`[“espresso”, “espresso”]`	`{“espresso”: 2, “soy”: 0}`

Normalize by sorting

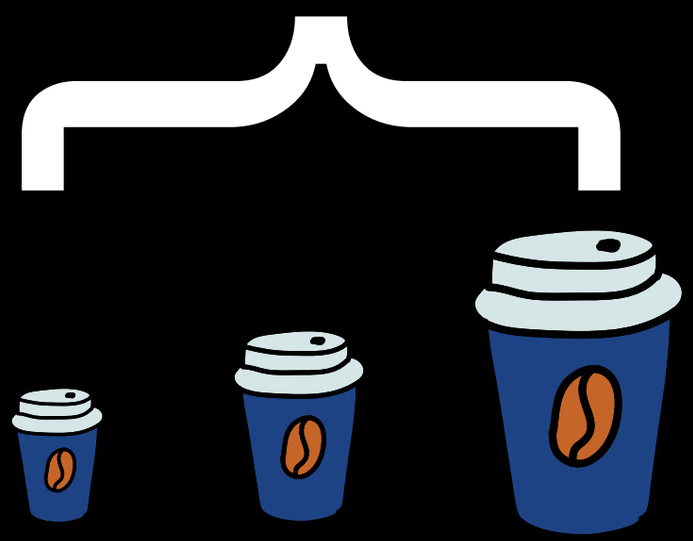
Normalize by removing zeroes

Array vs Object

?

Constrain with other lenses

one of → alternative



Super Mega Galactic

Model

Relationship

Encoding

“super” “mega” “galactic”

one of

```
type Size = "super"    |
            "mega"     |
            "galactic";
```

Code

Relationship Model

Encoding Code

Super Mega Galactic

alternative

Relationship Model

Encoding Code

Super Mega Galactic

alternative

# of ml

count

Relationship	Model
Encoding	Code

Super Mega Galactic

alternative

# of ml

count

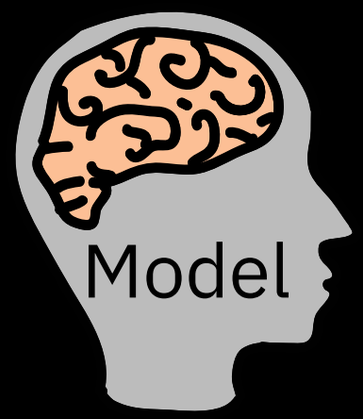
Americano Latte Cappuccino Macchiato

size related to style

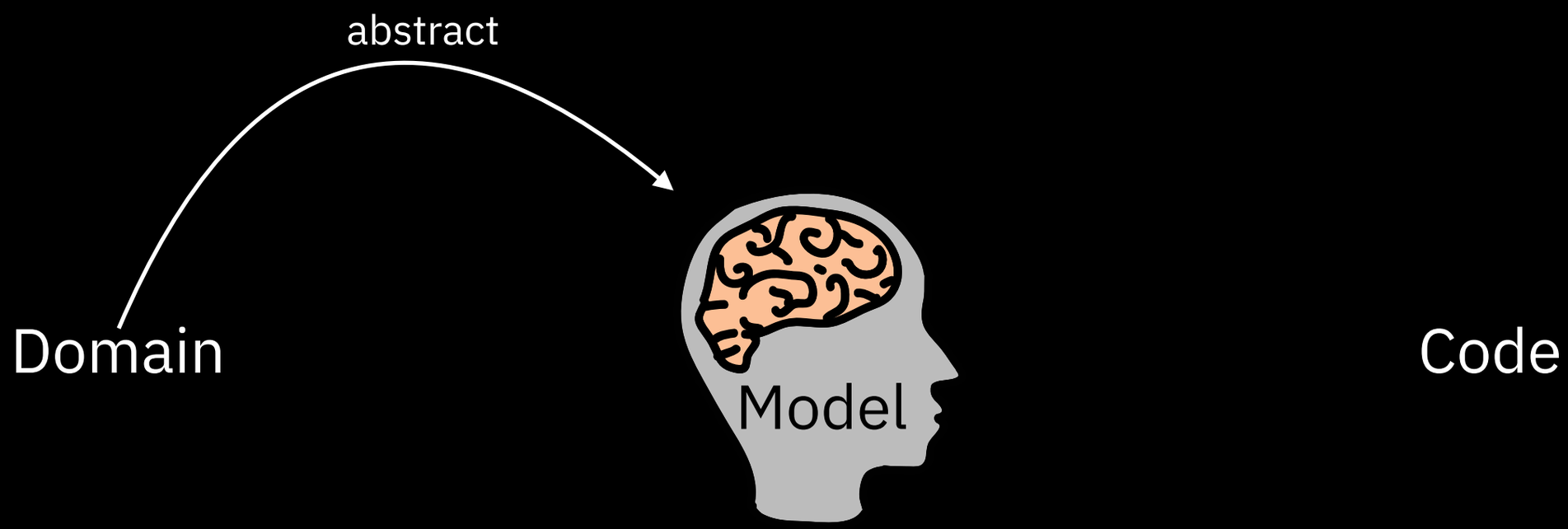
Relationship Model

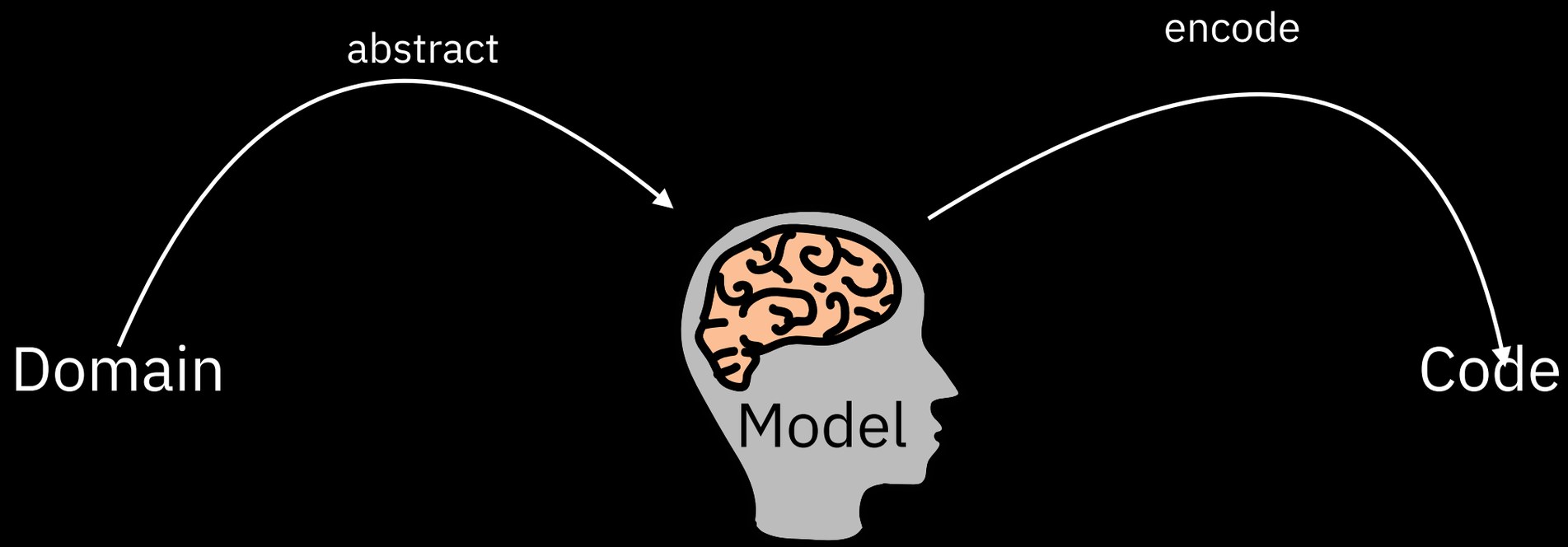
Encoding Code

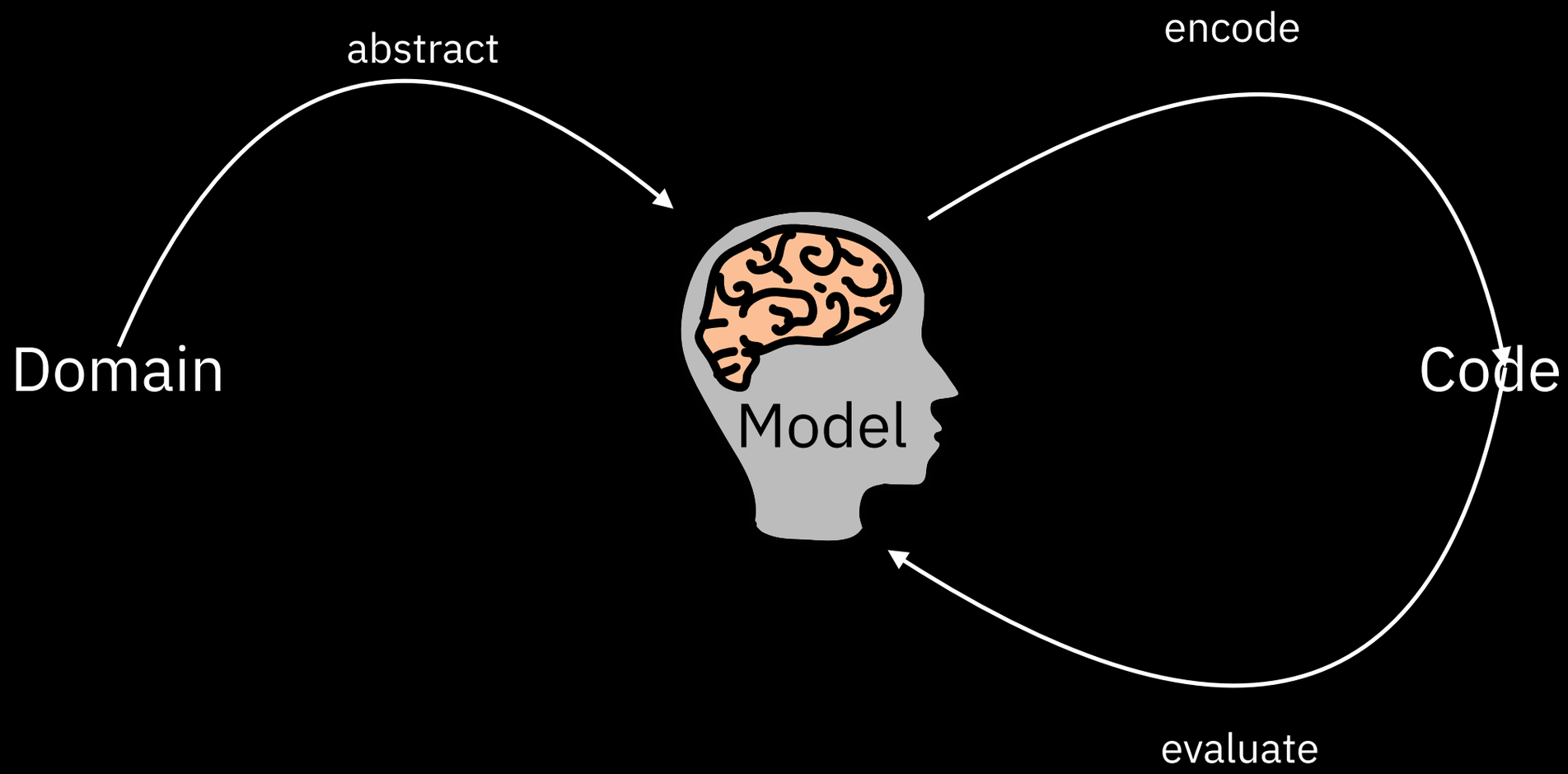
Domain

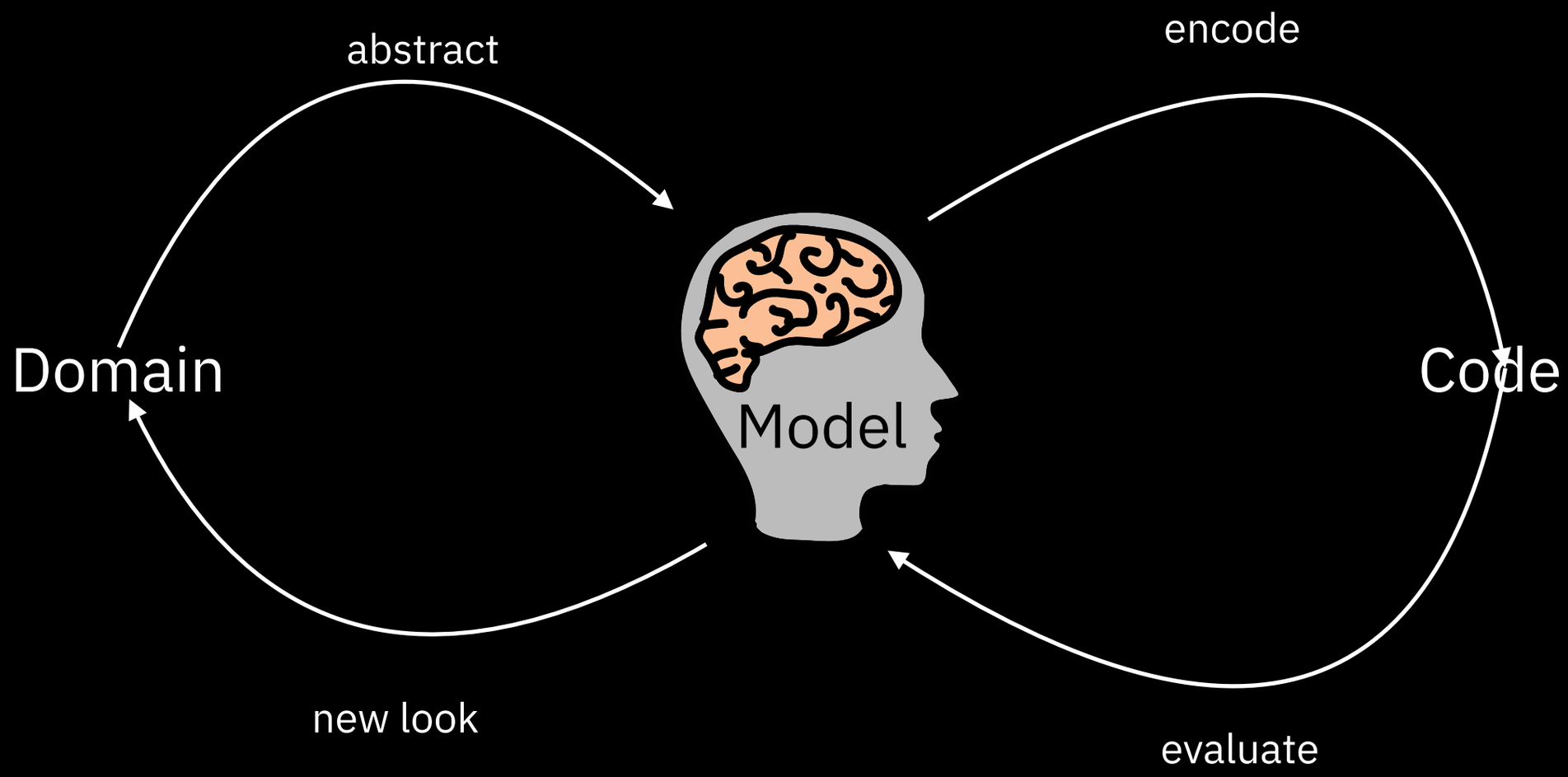


Code

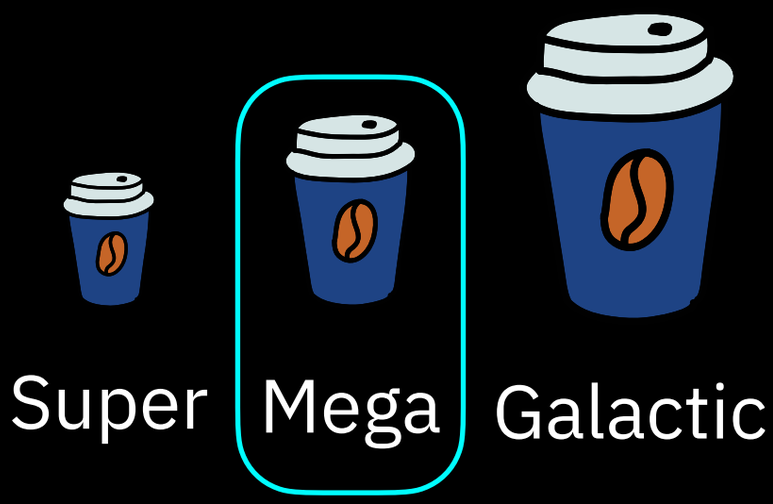

abstract
Domain
Model
Code

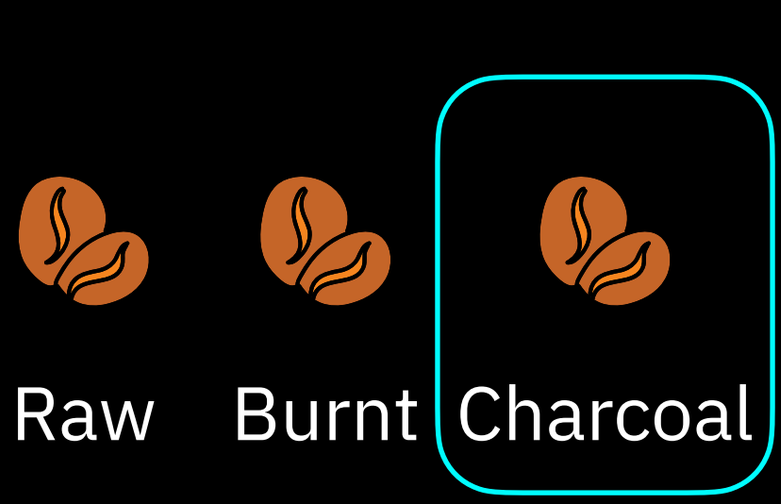

abstract
encode
Domain
Model
Code

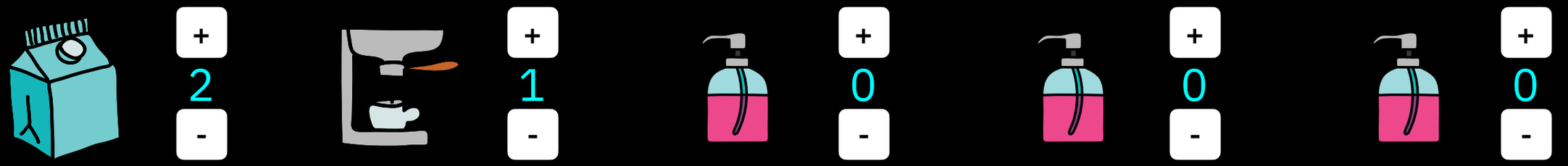

abstract
encode
Domain
Model
Code
evaluate


abstract
encode
Domain
Model
Code
new look
evaluate

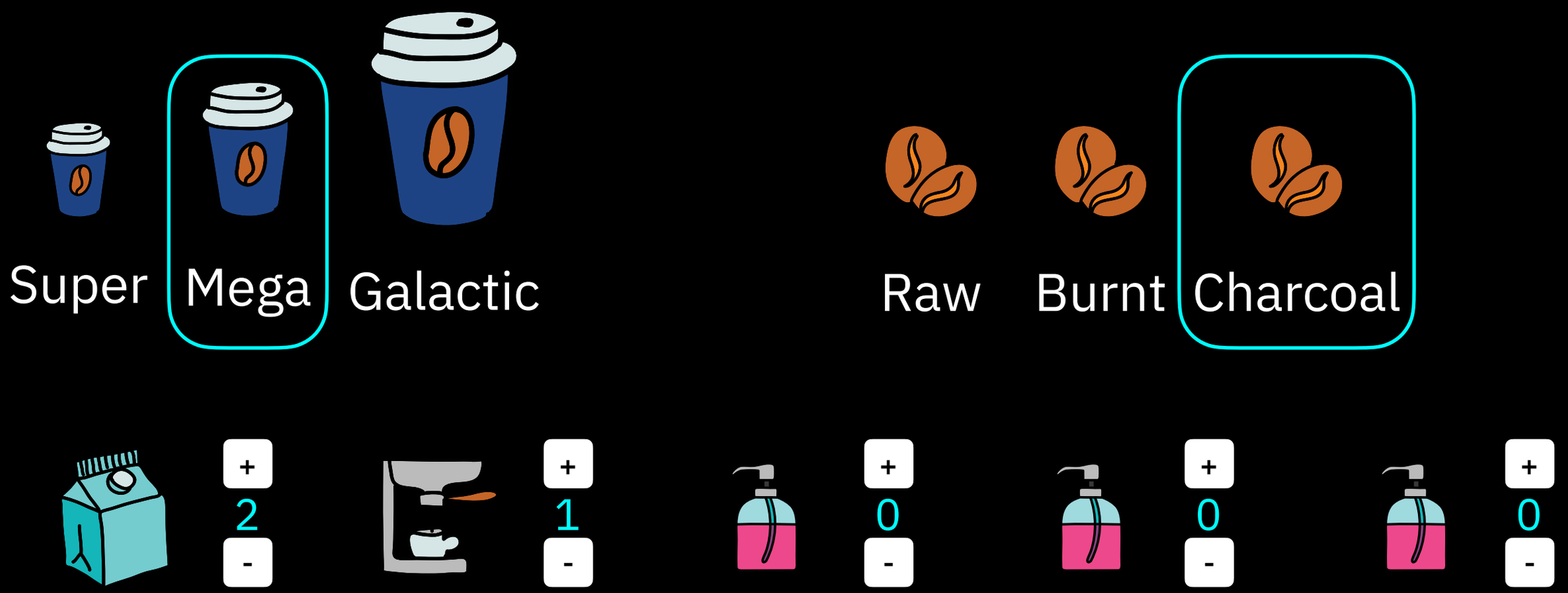
Operations



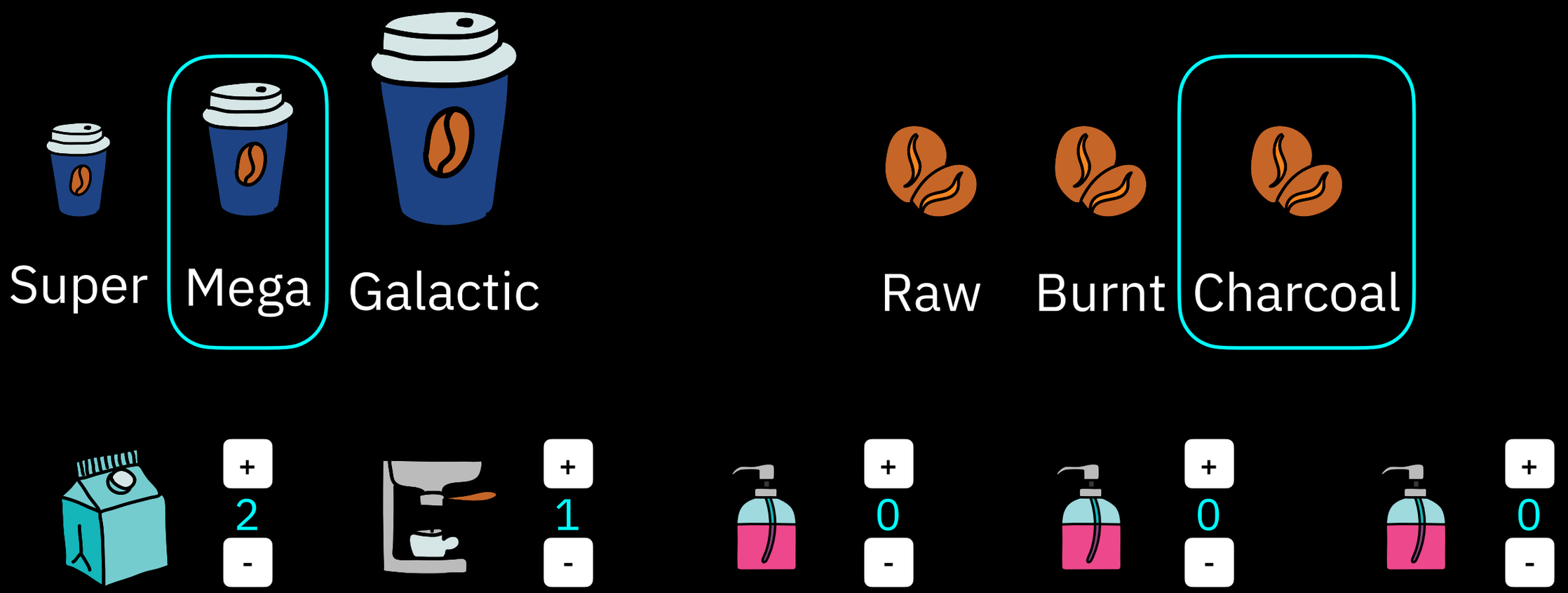




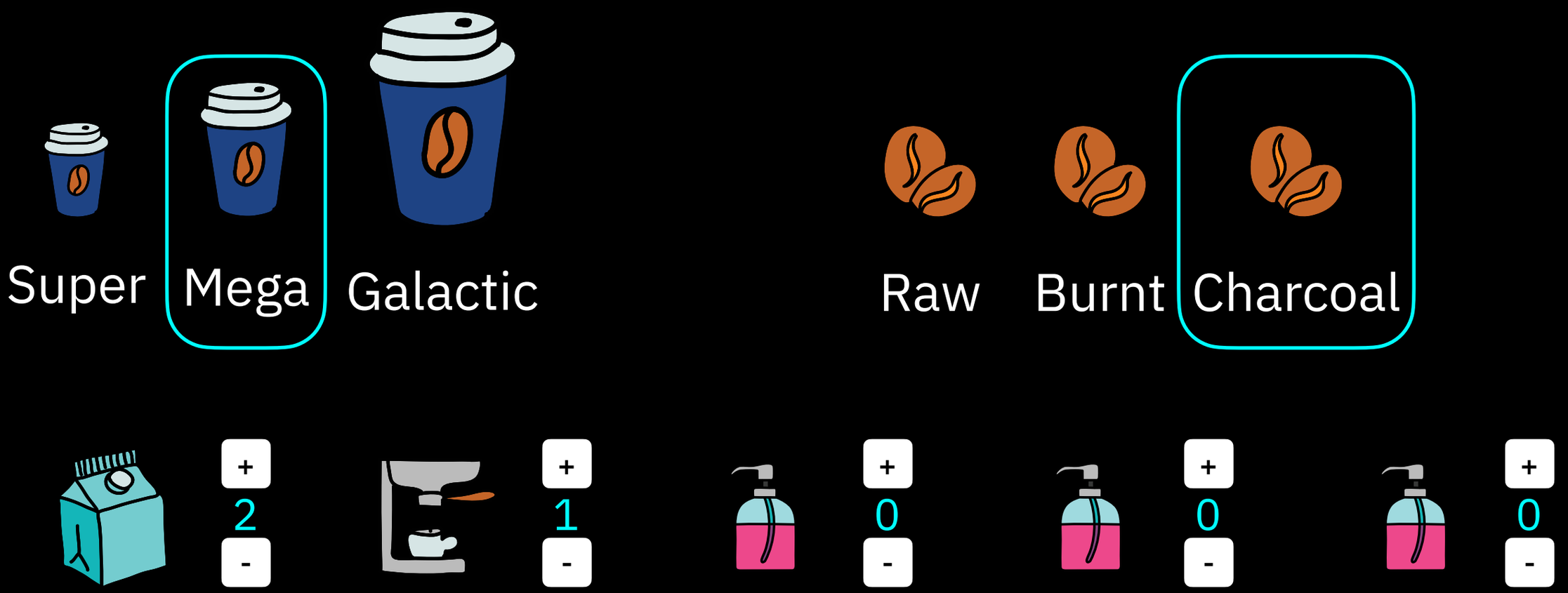
Soy milk Espresso Hazelnut Chocolate Almond



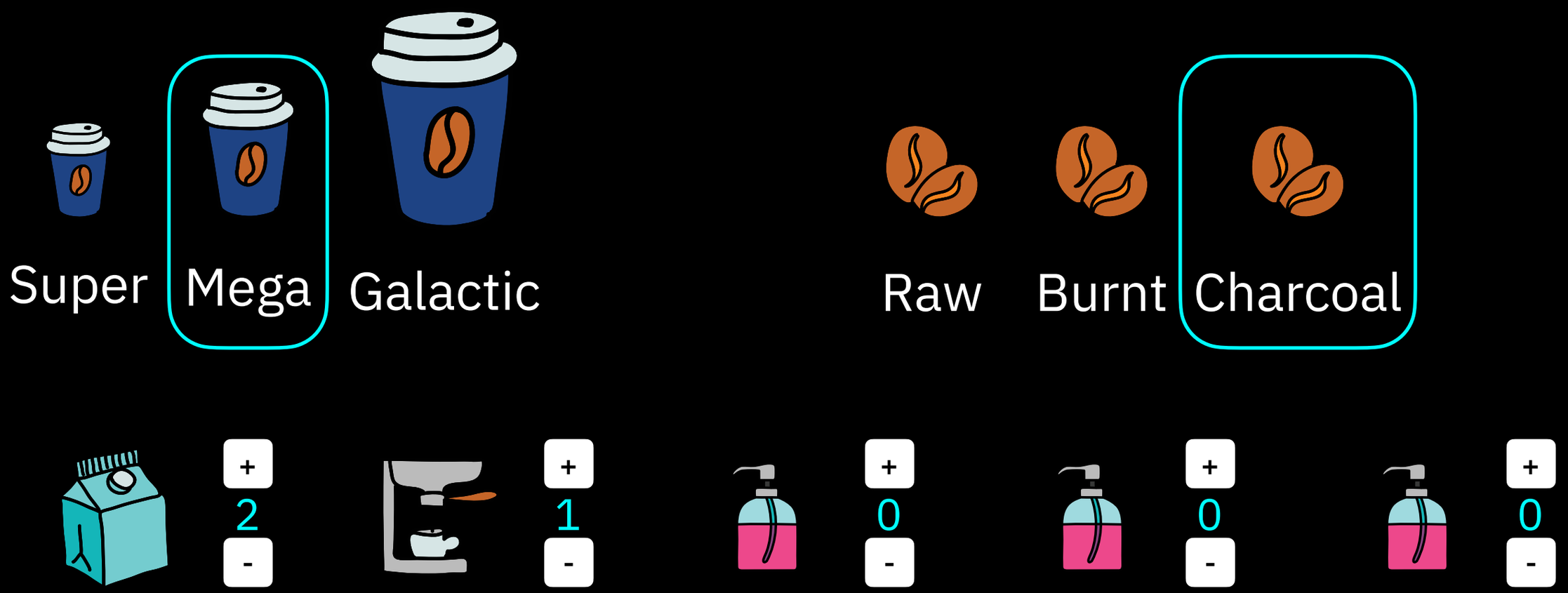

```
function setSize(coffee, size)      // => coffee
```




```
function setSize(coffee, size)      // => coffee
function setRoast(coffee, roast)    // => coffee
```




```
function setSize(coffee, size)      // => coffee
function setRoast(coffee, roast)    // => coffee
function addAddIn(coffee, addIn)    // => coffee
```




```
function setSize(coffee, size)       // => coffee
function setRoast(coffee, roast)     // => coffee
function addAddIn(coffee, addIn)     // => coffee
function removeAddIn(coffee, addIn) // => coffee
```

Function signatures are great for representing use cases

Function Signature

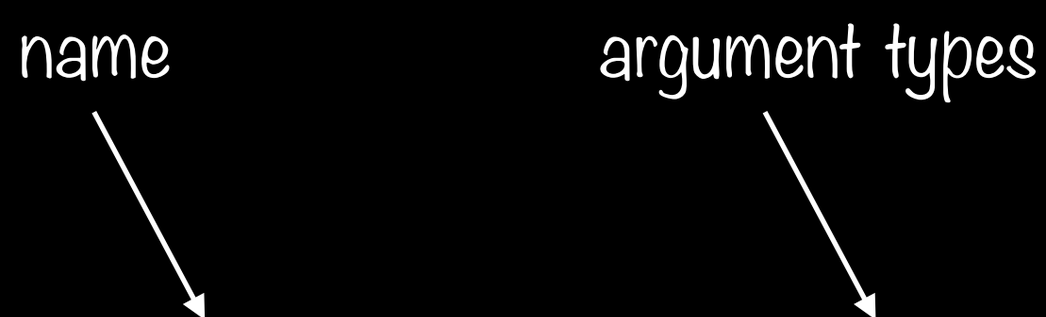
```
function setSize(coffee, size) // => coffee
```

Function Signature

name

```
function setSize(coffee, size) // => coffee
```

Function Signature




```
function setSize(coffee, size) // => coffee
```

Function Signature

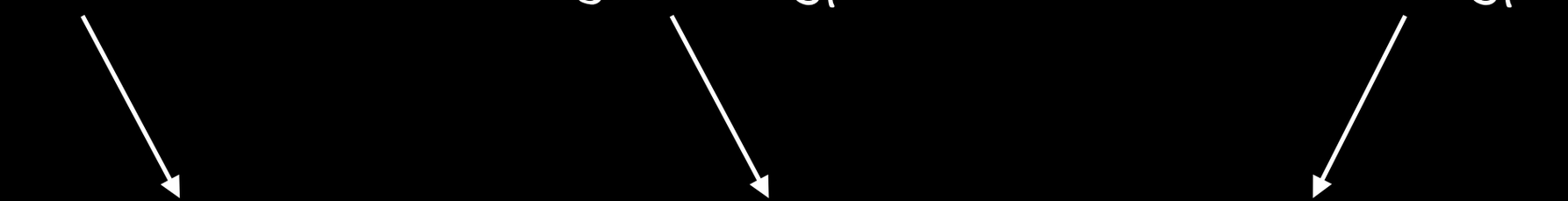
name

argument types

return type

```
function setSize(coffee, size) // => coffee
```

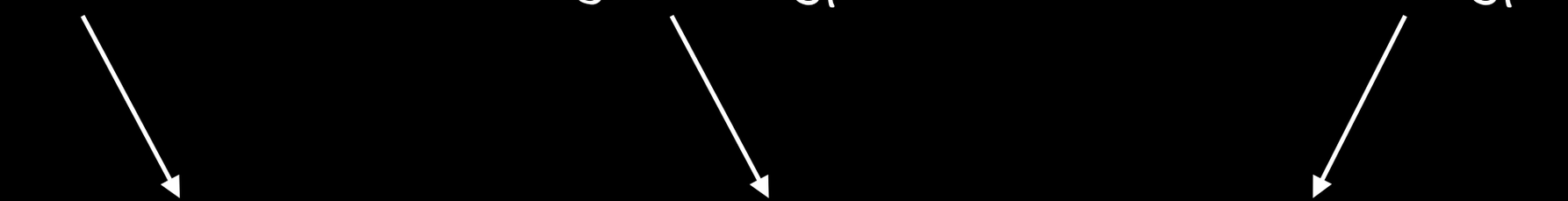
Function Signature




```
function setSize(coffee, size) // => coffee
```

- concise and precise requirement

Function Signature




```
function setSize(coffee, size) // => coffee
```

- concise and precise requirement
- right level of abstraction for reasoning

Function Signature

name

argument types

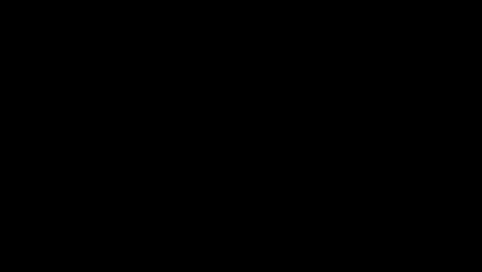
return type

```
function setSize(coffee, size) // => coffee
```

- concise and precise requirement
- right level of abstraction for reasoning
- easy to implement later

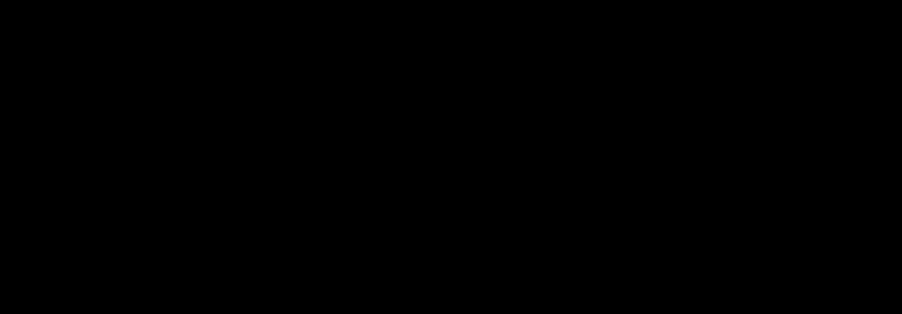
```
function setSize(coffee, size) // => coffee
```

```
function setSize(coffee, size) // => coffee
```



```
{
  “size”: “mega”,
  “roast”: “raw”,
  “add-ins”: []
}
```

```
function setSize(coffee, size) // => coffee
```



```
{
  “size”: “mega”,
  “roast”: “raw”,
  “add-ins”: []
}
```

“mega”

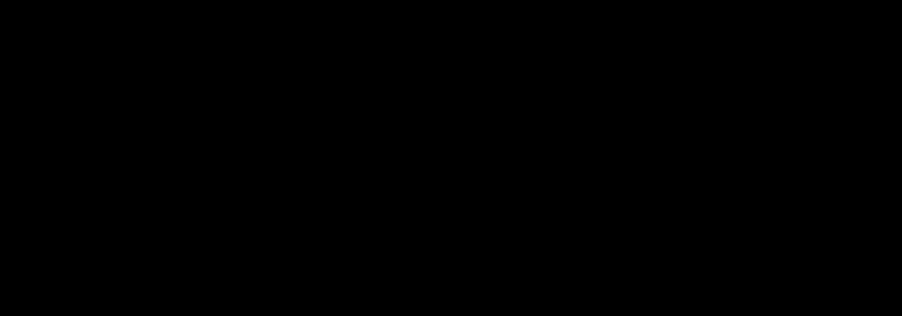
```
function setSize(coffee, size) // => coffee
```

```
{
  “size”: “mega”,
  “roast”: “raw”,
  “add-ins”: []
}
```

“mega”

?

```
function setSize(coffee, size) // => coffee
```



```
{
  “size”: “mega”,
  “roast”: “raw”,
  “add-ins”: []
}
```

“mega”

throw

Total Functions

valid return for all valid inputs

```
function setSize(coffee, size) // => coffee | null
```

```
{
  “size”: “mega”,
  “roast”: “raw”,
  “add-ins”: []
}
```

“mega”

null

Total Functions

valid return for all valid inputs

augment the return type

```
function setSize(coffee, size) // => coffee | null
```

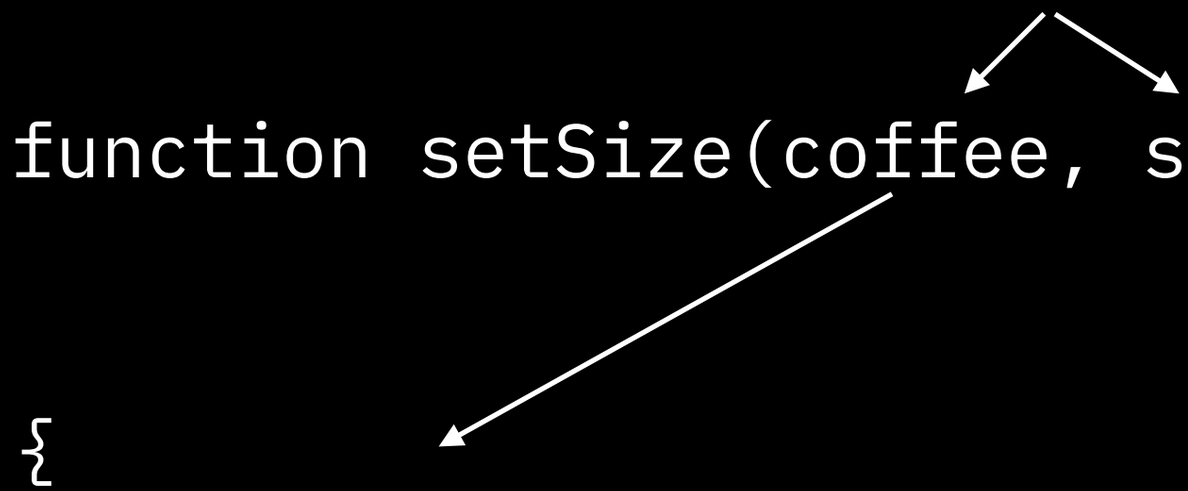
```
{
  “size”: “mega”,
  “roast”: “raw”,
  “add-ins”: []
}
```

“mega”

null

Total Functions

valid return for all valid inputs



Total Functions

valid return for all valid inputs

```
coffee.size !== size
```

```
function setSize(coffee, size) // => coffee
```

```
{
  “size”: “mega”,
  “roast”: “raw”,
  “add-ins”: []
}
```

```
“mega”
```

Total Functions

valid return for all valid inputs

restrict the argument types

```
coffee.size !== size
```

```
function setSize(coffee, size) // => coffee
```

```
{
  “size”: “mega”,
  “roast”: “raw”,
  “add-ins”: []
}
```

```
“mega”
```

Total Functions

valid return for all valid inputs

```
function setSize(coffee, size) // => coffee
```

```
{
  “size”: “mega”,
  “roast”: “raw”,
  “add-ins”: []
}
```

“mega”

Total Functions

valid return for all valid inputs

```
function setSize(coffee, size) // => coffee
```

```
{
  “size”: “mega”,
  “roast”: “raw”,
  “add-ins”: []
}
```

“mega”

return
unchanged

Total functions are good building blocks

- Augmenting the return - weird return type
- Restricting the arguments - weird argument type
- Change the meaning - not always possible
- Avoid throwing errors

Barrista

Barrista

```
function howManyAddIns(coffee, addIn) // => number
```

Barrista

```
function howManyAddIns(coffee, addIn) // => number
```

Marketing

Barrista

```
function howManyAddIns(coffee, addIn) // => number
```

Marketing

```
function hasAddIn(coffee, addIn) // => boolean
```

Barrista

```
function howManyAddIns(coffee, addIn) // => number
```

Marketing

```
function hasAddIn(coffee, addIn) // => boolean
```

Cashier

Barrista

```
function howManyAddIns(coffee, addIn) // => number
```

Marketing

```
function hasAddIn(coffee, addIn) // => boolean
```

Cashier

```
function price(coffee) // => number
```

Array

Array

```
function howManyAddIns(addIns, addIn) { // => number
  return addIns.filter(a => a === addIn).length;
}
```

Array

```
function howManyAddIns(addIns, addIn) { // => number
  return addIns.filter(a => a === addIn).length;
}
```

Array

```
function howManyAddIns(addIns, addIn) { // => number
  return addIns.filter(a => a === addIn).length;
}
```

Object

Array

```
function howManyAddIns(addIns, addIn) { // => number
  return addIns.filter(a => a === addIn).length;
}
```

Object

```
function howManyAddIns(addIns, addIn) { // => number
  return addIns[addIn] || 0;
}
```

Array

```
function howManyAddIns(addIns, addIn) { // => number
  return addIns.filter(a => a === addIn).length;
}
```

linear search

Object

```
function howManyAddIns(addIns, addIn) { // => number
  return addIns[addIn] || 0;
}
```

constant time

Array

Array

```
function hasAddIn(addIns, addIn) { // => boolean
  return addIns.indexOf(addIn) !== -1;
}
```

Array

```
function hasAddIn(addIns, addIn) { // => boolean
  return addIns.indexOf(addIn) !== -1;
}
```

Array

```
function hasAddIn(addIns, addIn) { // => boolean
  return addIns.indexOf(addIn) !== -1;
}
```

Object

Array

```
function hasAddIn(addIns, addIn) { // => boolean
  return addIns.indexOf(addIn) !== -1;
}
```

Object

```
function hasAddIn(addIns, addIn) { // => boolean
  return !!addIns[addIn];
}
```

Array

```
function hasAddIn(addIns, addIn) { // => boolean
  return addIns.indexOf(addIn) !== -1;
}
```

linear search

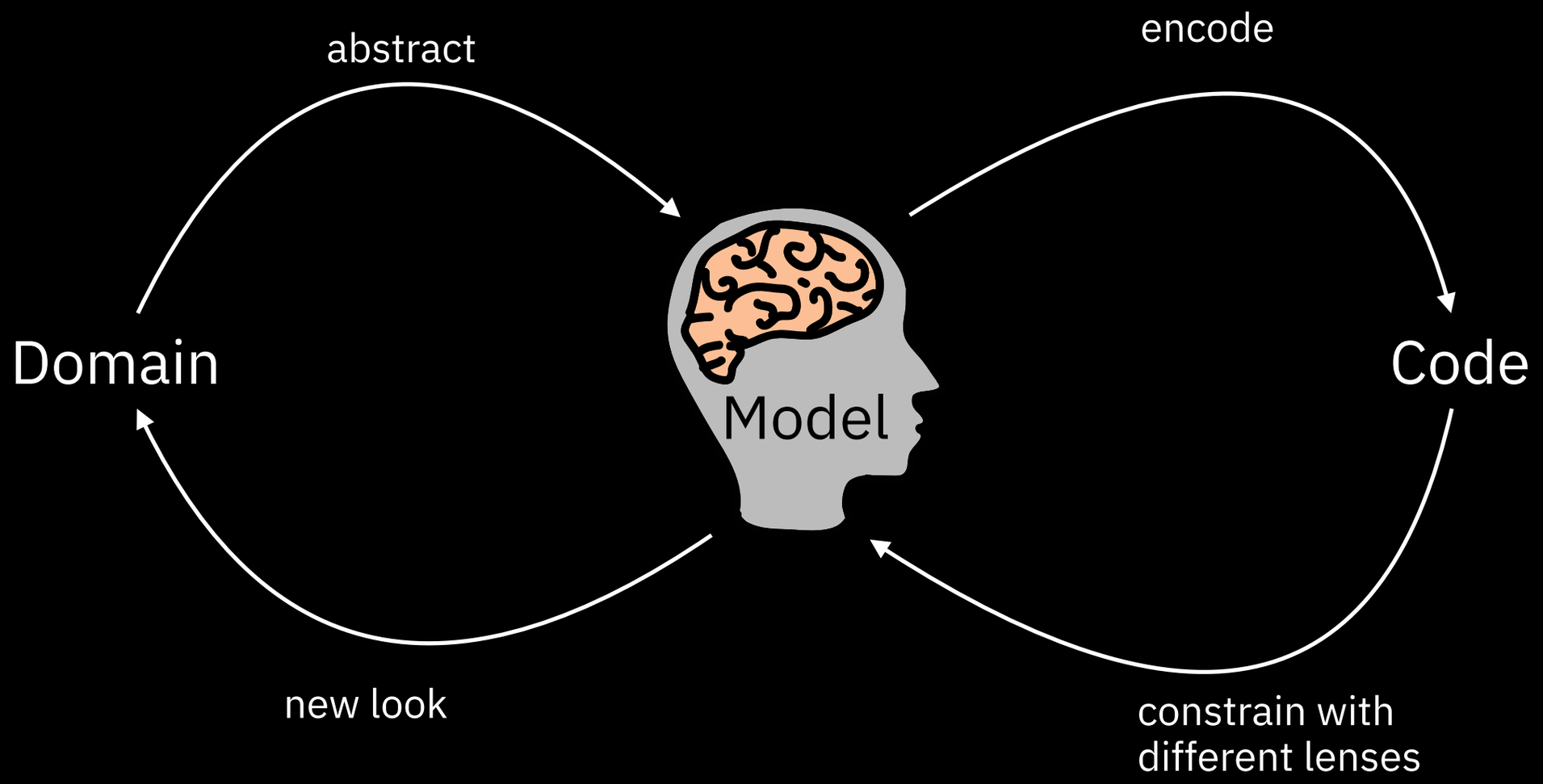
Object

```
function hasAddIn(addIns, addIn) { // => boolean
  return !!addIns[addIn];
}
```

constant time

Array vs Object

?


abstract
encode
Domain
Model
Code
new look
constrain with
different lenses

Volatility

How often does the business change the sizes?

Super Mega Galactic

```
type Size = "super"    |
            "mega"     |
            "galactic";
```

How often does the business change the sizes?

Super Mega Galactic

```
type Size = "super"    |
            "mega"     |
            "galactic";
```

Never? Every ten years?

How often does the business change the roasts?

Raw Burnt Charcoal

```
type Roast = "raw"      |
             "burnt"    |
             "charcoal";
```

Never? Every ten years?

How often does the business change sales and promotions?

How often does the business change sales and promotions?

Seasonally/monthly/weekly.

How often does the business change the add-ins?

Soy milk

Espresso

Hazelnut Chocolate Almond

```
type AddIn = "soy"       |
             "espresso"  |
             "hazelnut"  |
             "chocolate" |
             "almond"    ;
```

How often does the business change the add-ins?

Soy milk

Espresso

Hazelnut Chocolate Almond

```
type AddIn = "soy"       |
             "espresso"  |
             "hazelnut"  |
             "chocolate" |
             "almond"    ;
```

Seasonally/Hourly.

Decades

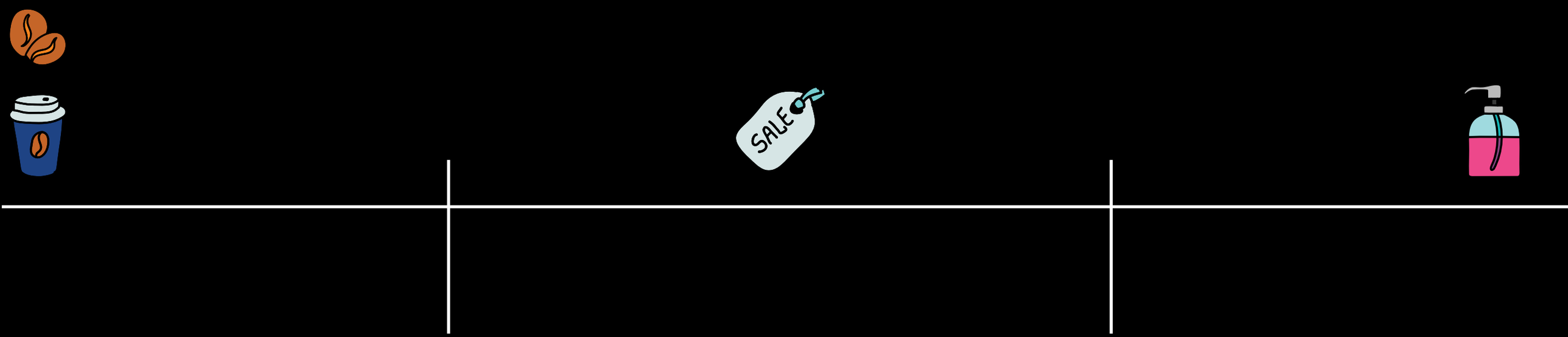
Yearly

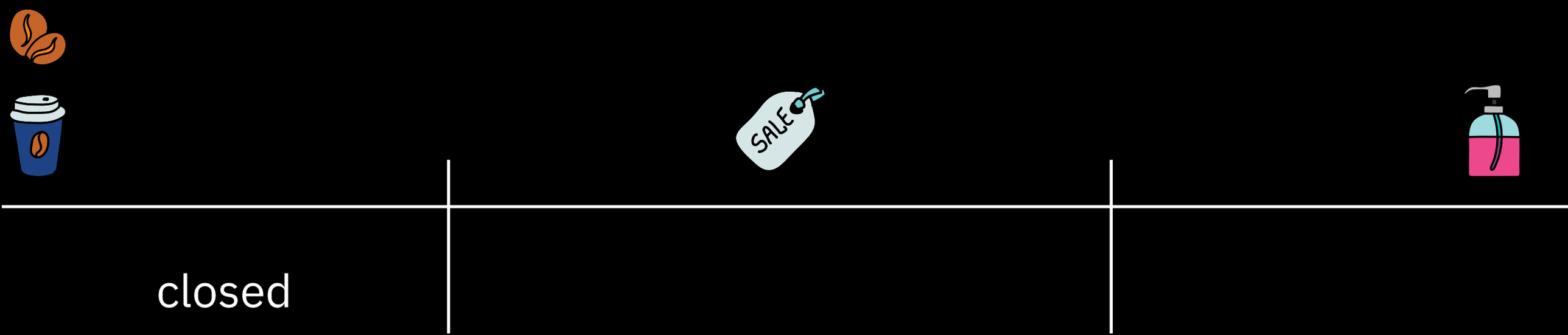
Daily

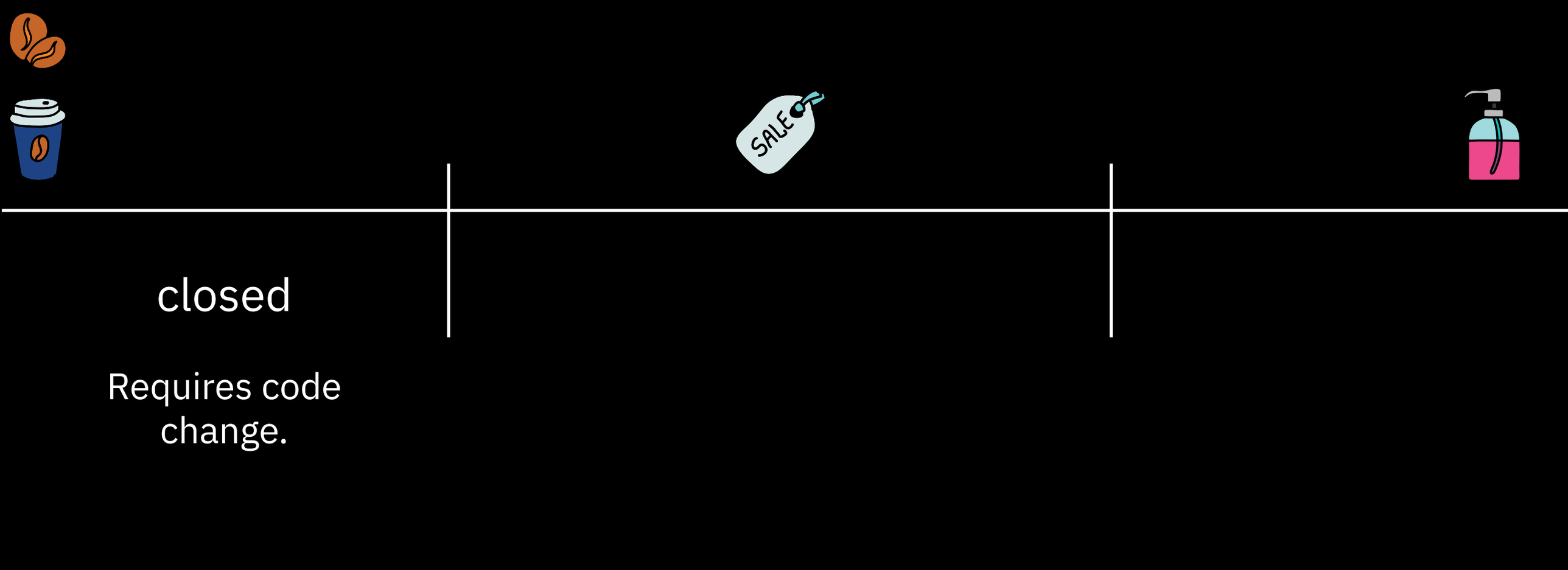
Hourly

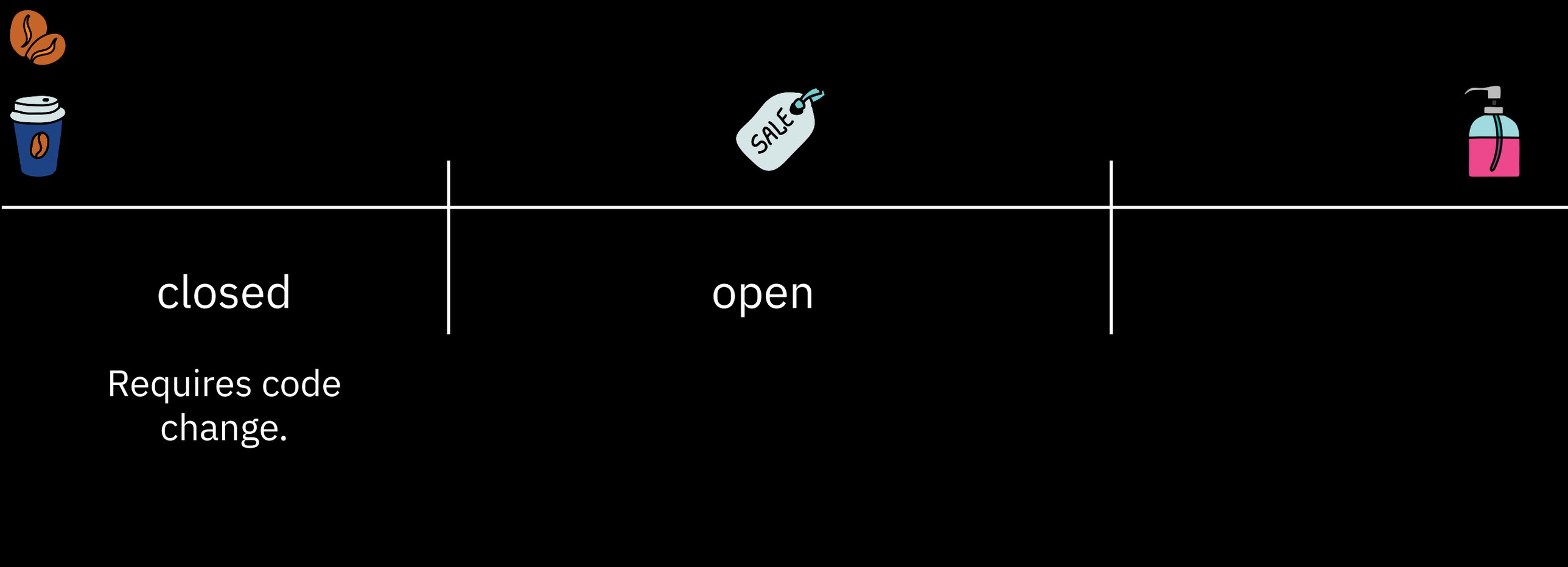
How does this align with development/deployment cycle?

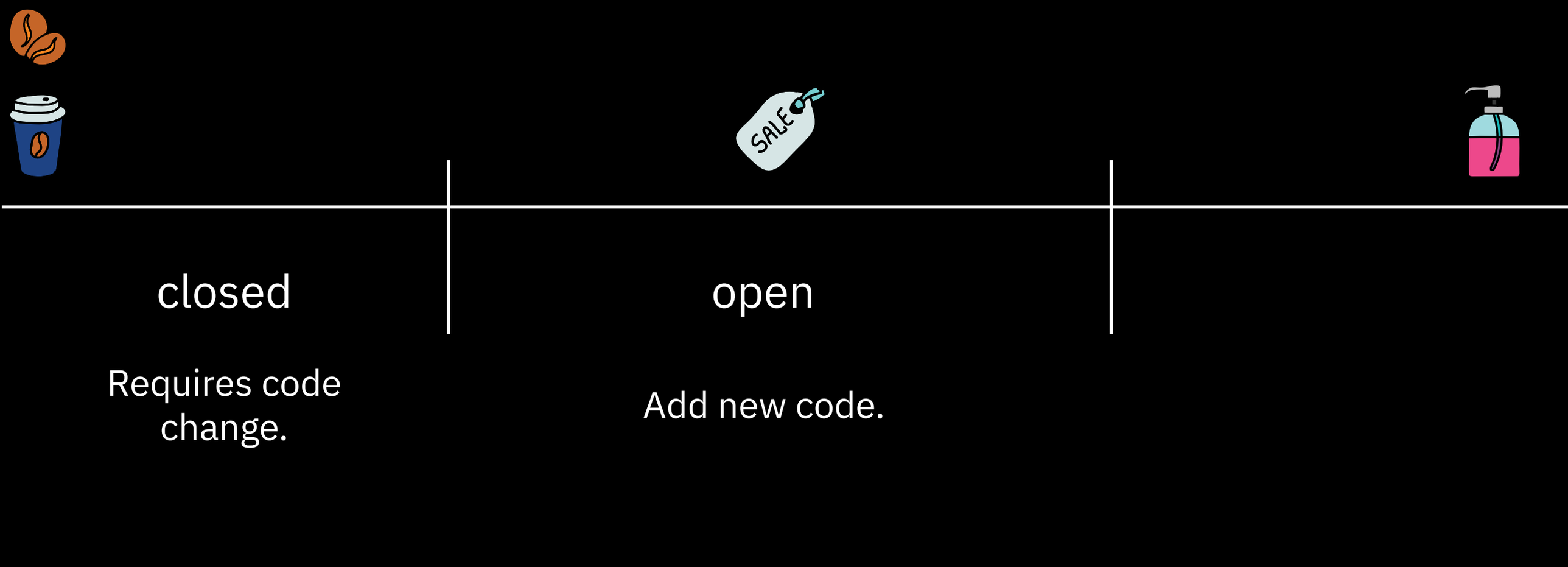
Decades | Yearly | Daily | Hourly

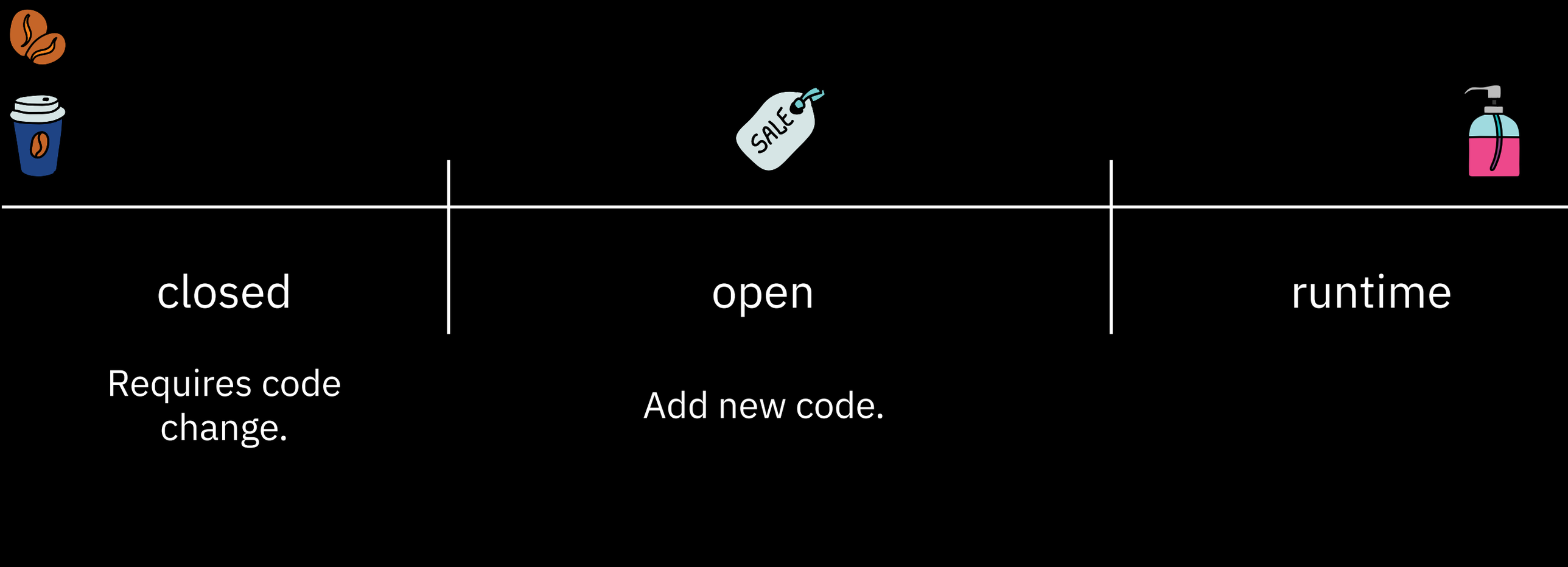

SALE


SALE
closed


SALE
closed
Requires code change.


SALE
closed
open
Requires code change.


SALE
closed
Requires code change.
open
Add new code.


SALE
closed
open
runtime
Requires code change.
Add new code.

closed	open	runtime
Requires code change.	Add new code.	No code changes.

closed - requires code change

closed - requires code change

```
type Size = "super"    |
            "mega"     |
            "galactic";
```

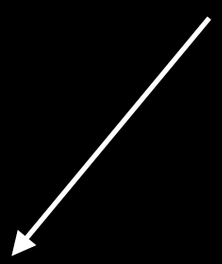
open - add new code

open - add new code

```
interface Size {
  name: string;
}
```

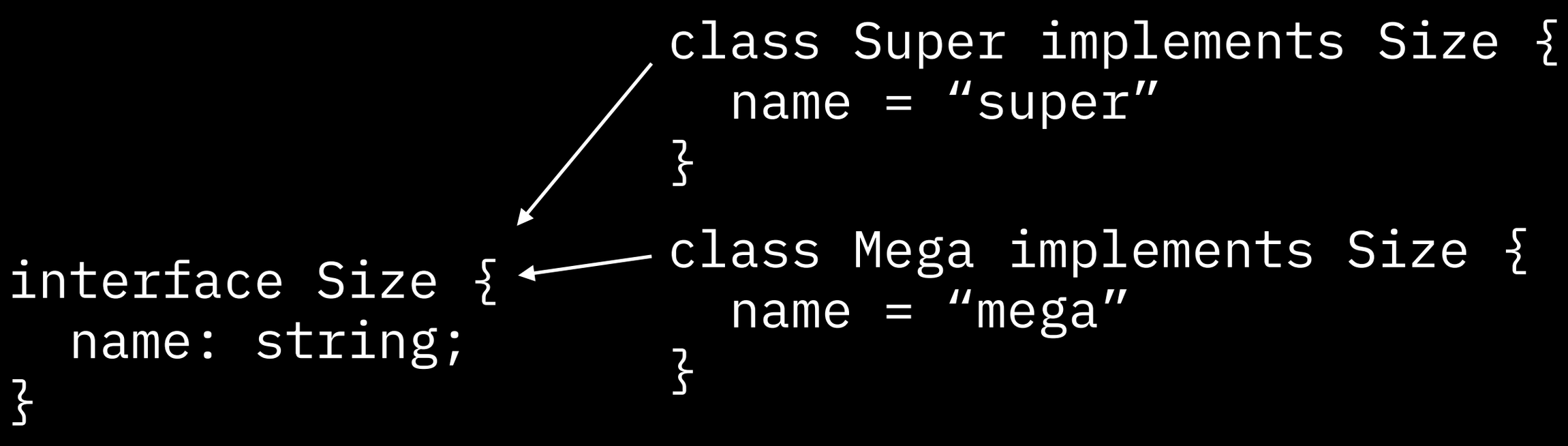
open - add new code

```
class Super implements Size {
  name = “super”
}
```

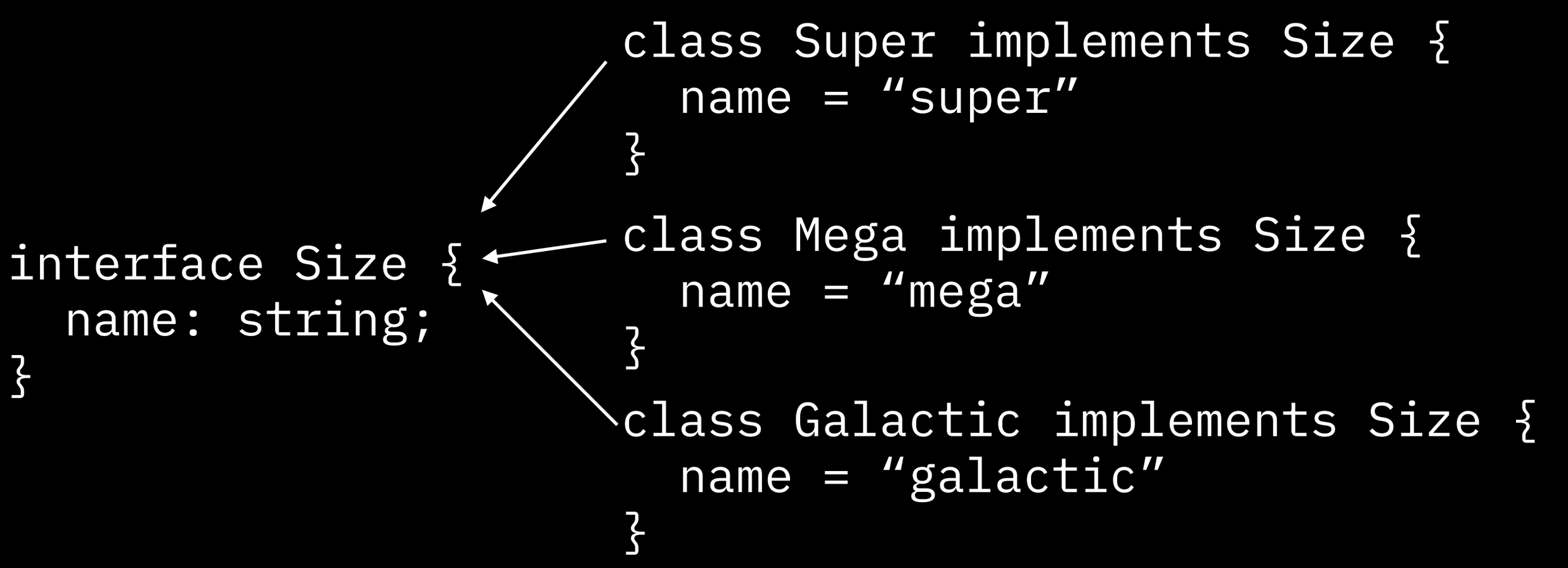


```
interface Size {
  name: string;
}
```

open - add new code



open - add new code



runtime - no code changes

runtime - no code changes

```
type Size = string;
```

runtime - no code changes

```
type Size = string;

type Size = { name = string; price = number };
```

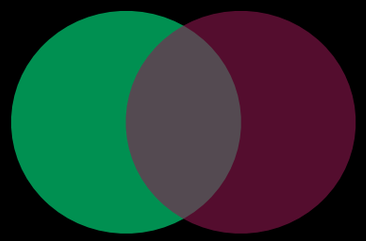
Data modeling

Data modeling

Encoding relationships

Data modeling

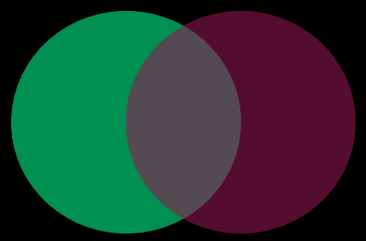
Encoding relationships



Evaluating fit

Data modeling

Encoding relationships

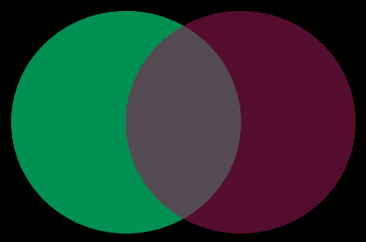


Evaluating fit

Operation modeling

Data modeling

Encoding relationships



Evaluating fit

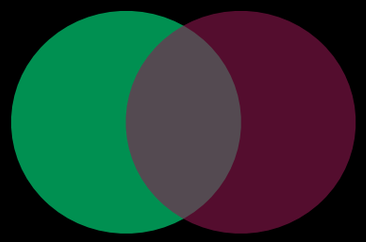
Operation modeling

```
setSize(coffee, size) // => coffee
```

Function signatures

Data modeling

Encoding relationships



Evaluating fit

Operation modeling

```
setSize(coffee, size) // => coffee
```

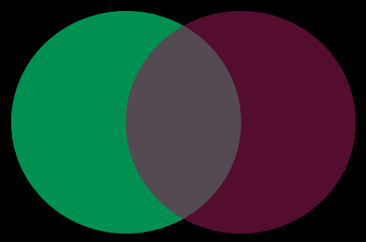
Function signatures

`throw`

Total functions

Data modeling

Encoding relationships



Evaluating fit

Operation modeling

```
setSize(coffee, size) // => coffee
```

Function signatures

~~`throw`~~

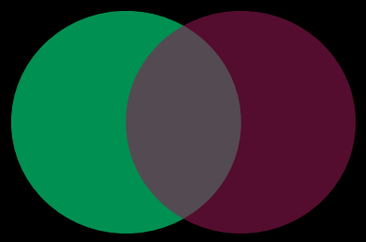
Total functions

`O(n)`

Evaluating implementation complexity

Data modeling

Encoding relationships



Evaluating fit

Operation modeling

```
setSize(coffee, size) // => coffee
```

Function signatures

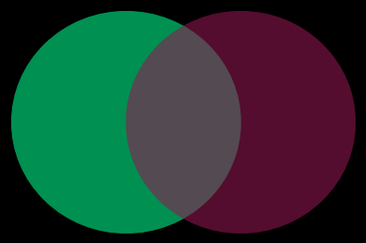
`throw` Total functions

`O(n)` Evaluating implementation complexity

Volatility modeling

Data modeling

Encoding relationships



Evaluating fit

Operation modeling

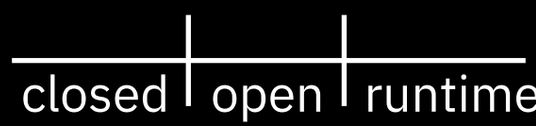
```
setSize(coffee, size) // => coffee
```

Function signatures

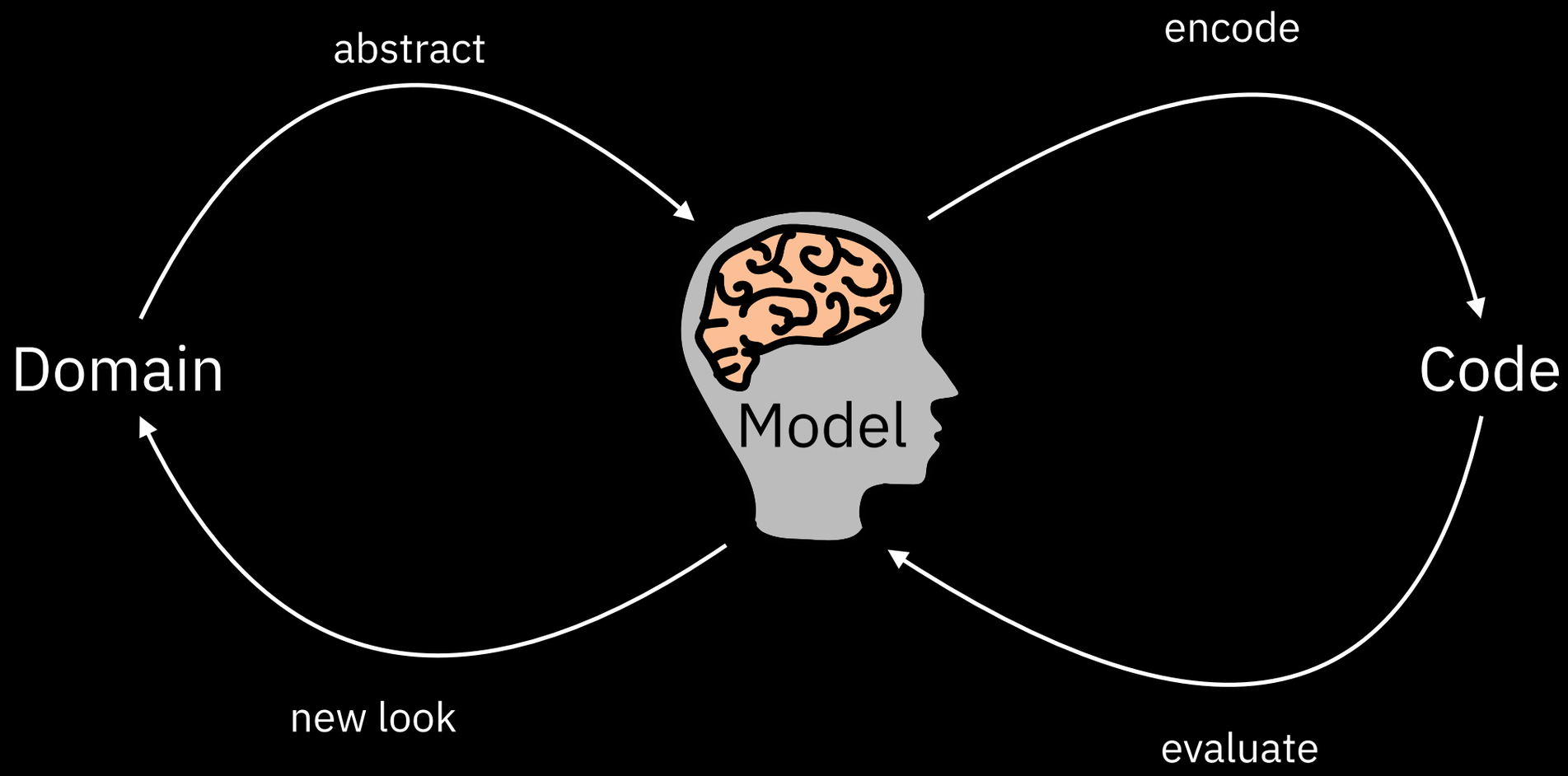
`throw` Total functions

`O(n)` Evaluating implementation complexity

Volatility modeling



Analyzing frequency of change


abstract
encode
Domain
Model
Code
new look
evaluate

ericnormand.me

Coffee cup Image by Freepik

https://www.freepik.com/free-vector/list-different-types-coffee_951047.htm