Developing Software for Human

Human-centered Design Codebase

Pittawat Taveekitworachai (Pete)

25 April 2022

Pete

Pittawat Taveekitworachai

- 👨🏻‍🔬 **Research Associate**, Innovative Cognitive Computing Lab, School of Information Technology, King Mongkut's University of Technology, Thailand
- 👨🏻‍🎓 **Visitor**, Intelligence Computer Entertainment Lab, Department of Human and Computer Intelligence, Ritsumeikan University, Japan
- ✍🏻 **Writer**, PETEPITTAWAT.DEV

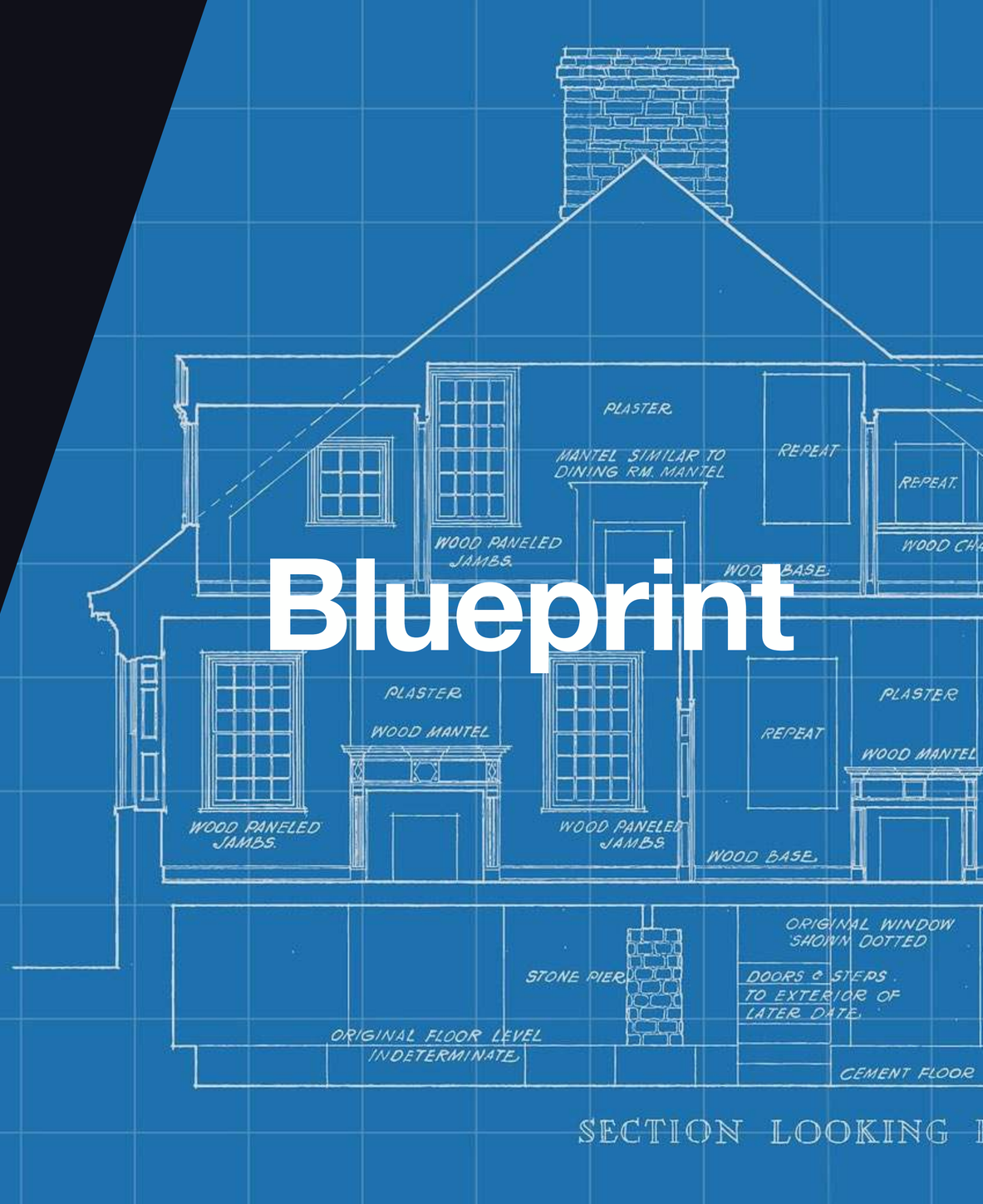
Developing **Software** 💾

=

Building a **House** 🏠

Software

Diagram



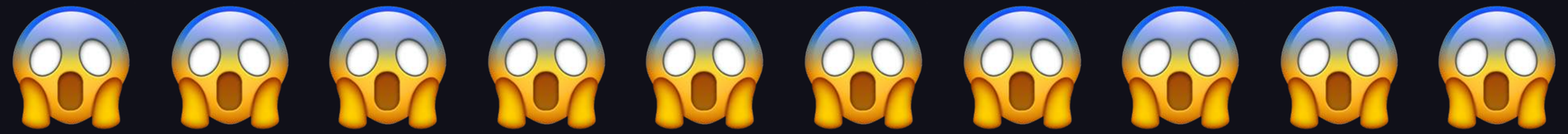
Blueprint

Feature

Function

What’re the **differences?** 🤔

A lot!



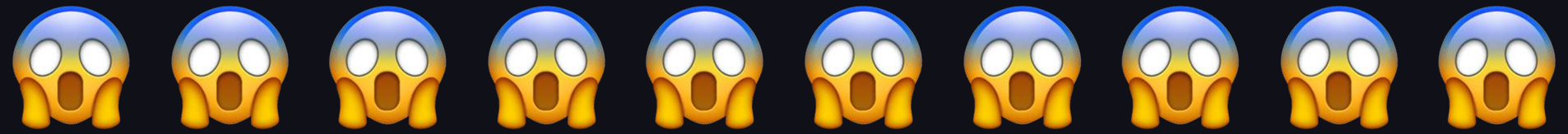
Intangible 

VS

Tangible 

What’re the **sameness?** 🤔

A lot!



It’s a **SOLUTION** 😎

😇 USER 😈


😇USER😈

WE are also a user

User of our codebase

As a Developer

INTRODUCING

Human-Centered Design

💡 Knowledge 💡

Knowledge in **Your Head** 

Knowledge in **The World** 

Fast

Muscle Memory

Knowledge in
Your Head 🧠

Invisible

Hard at first, but
PRACTICED

Interpreted

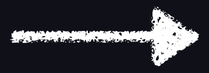
NO LEARNING
Required

Knowledge in
The World 🌏

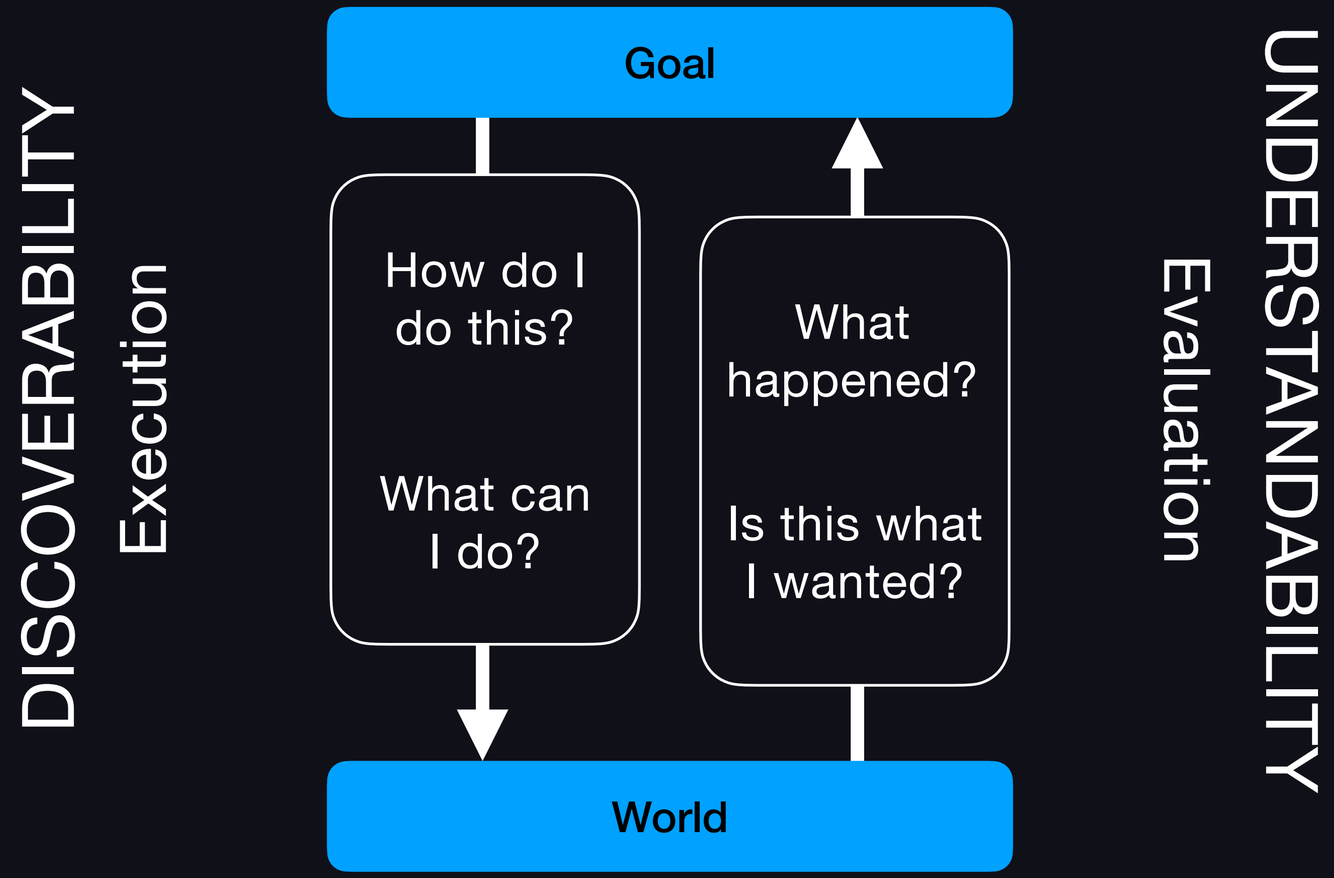
Visible

Easy to use, but need to
BE REMINDED

Knowledge in
The World 🌏



Knowledge in
Your Head 🧠


DISCOVERABILITY
Execution
Goal
How do I do this?
What can I do?
What happened?
Is this what I wanted?
Evaluation
UNDERSTANDABILITY
World

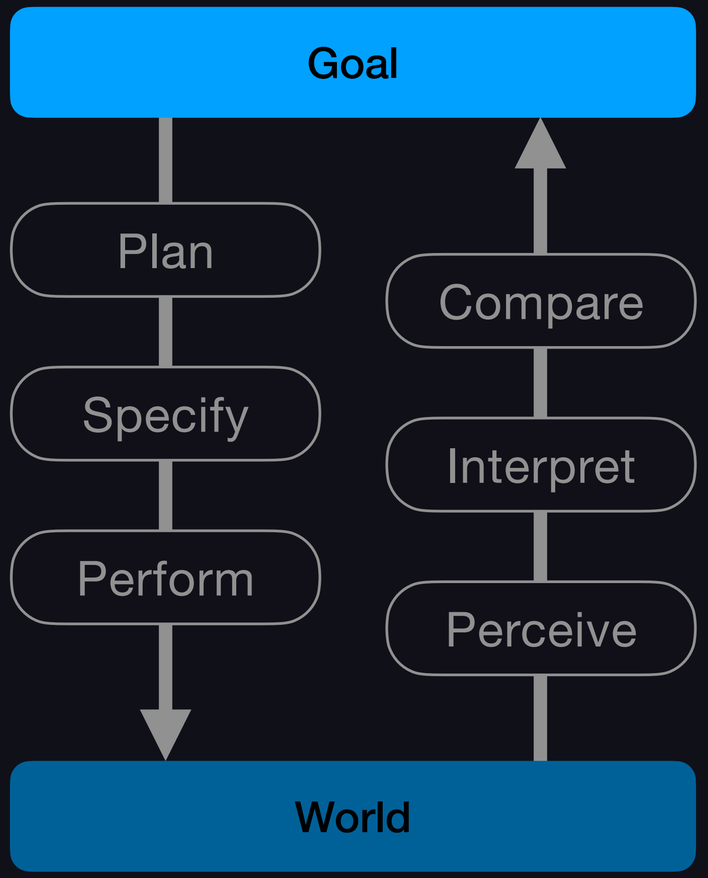
DISCOVERABILITY

Feedforward

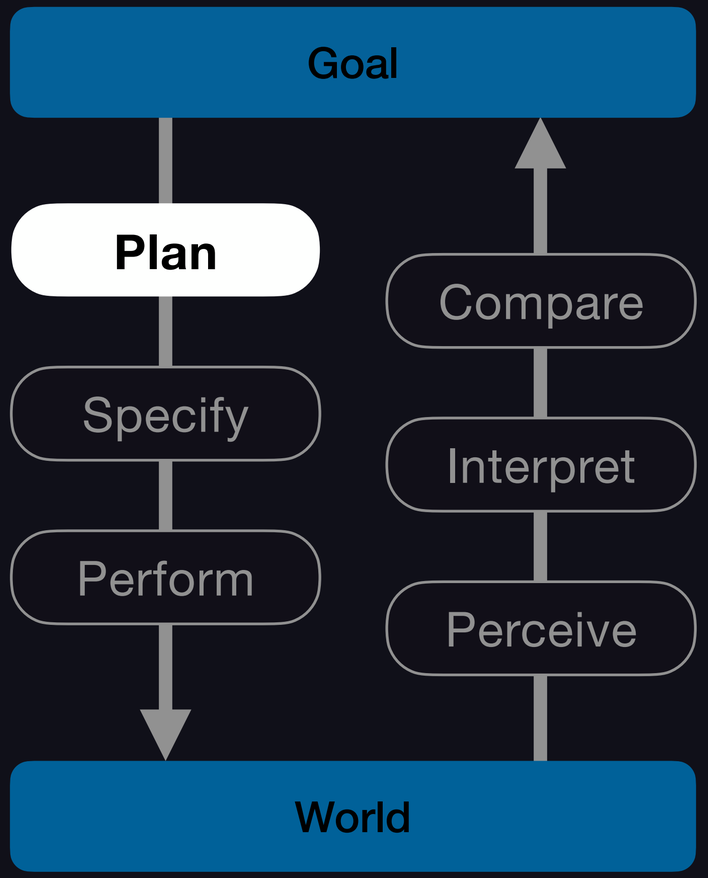
Feedback

UNDERSTANDABILITY

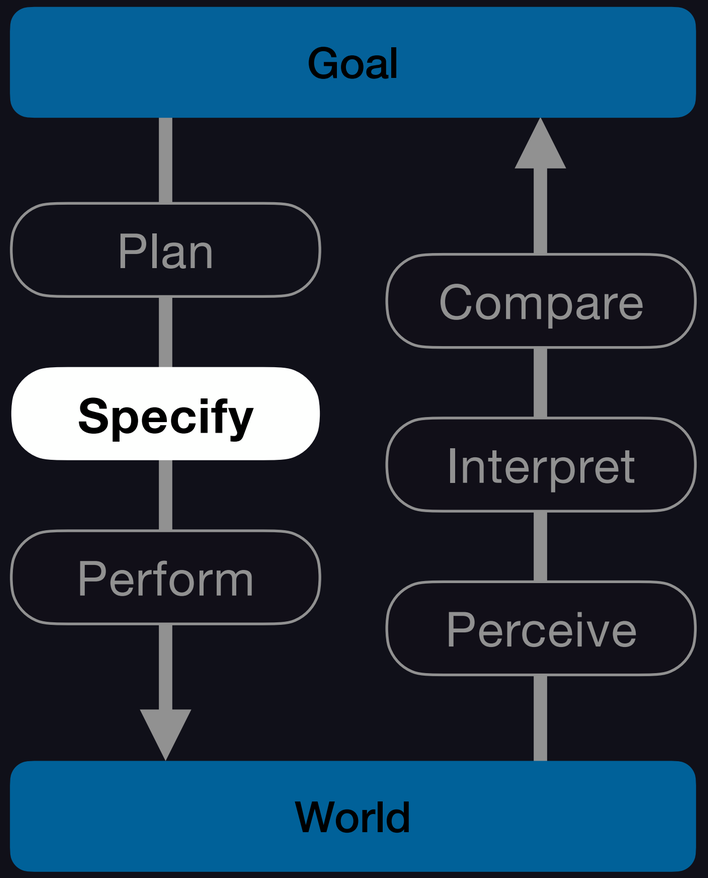
What do you want to accomplish?



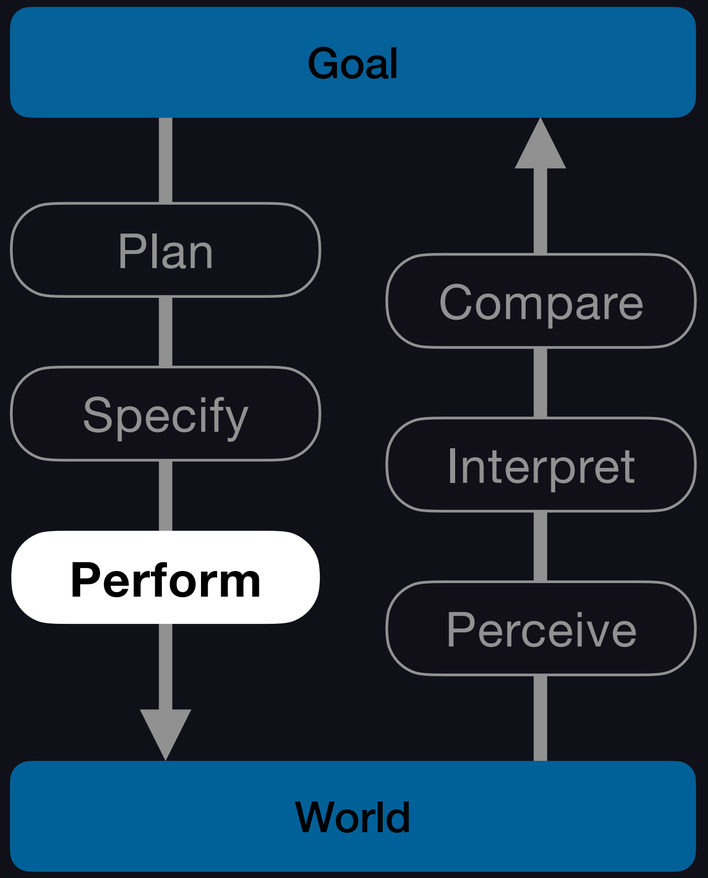
What are your options?

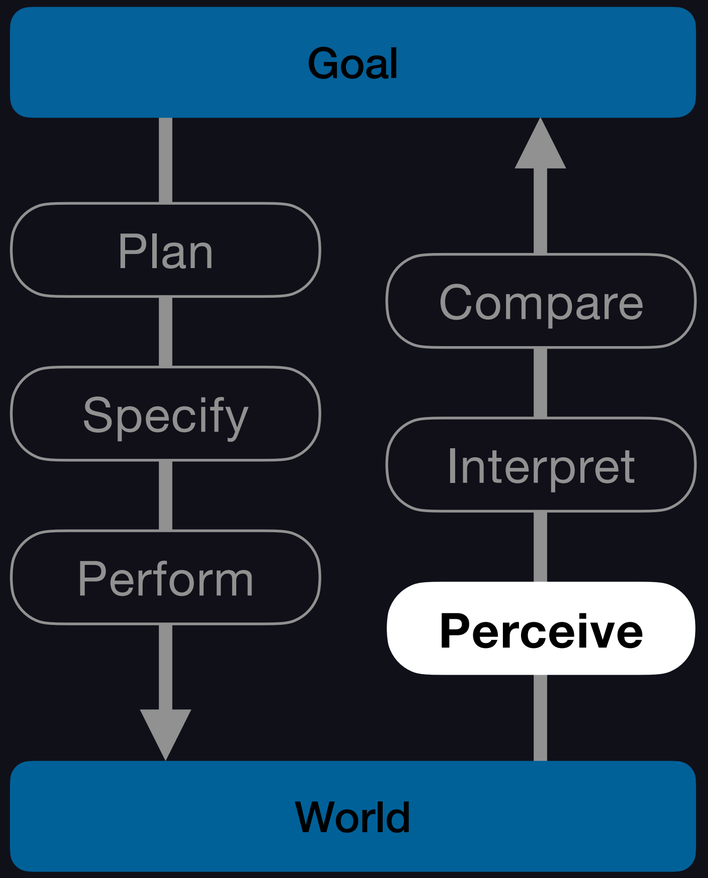


What action will you do?

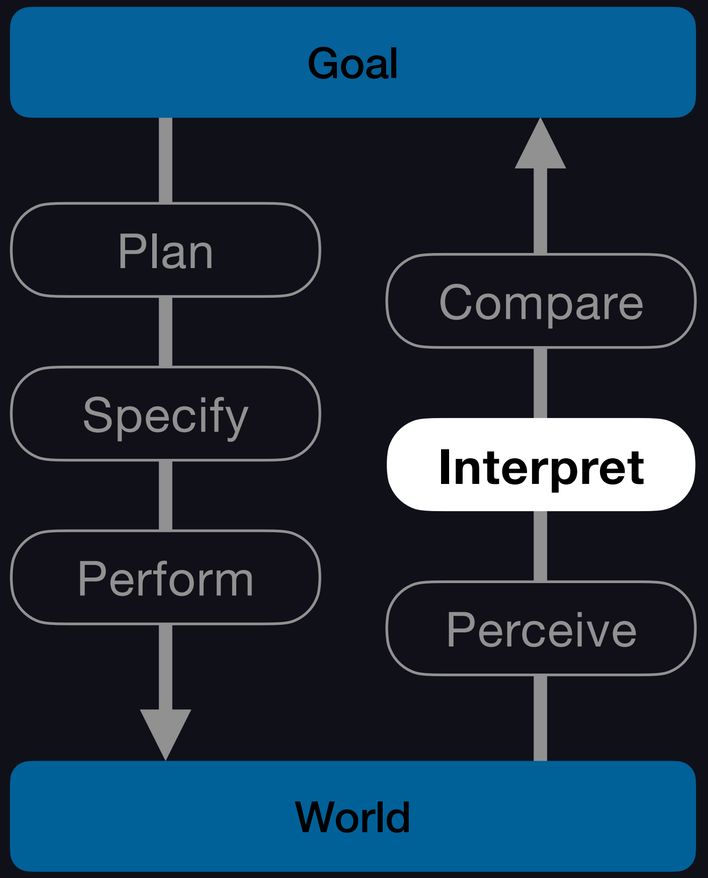


How will you do it?

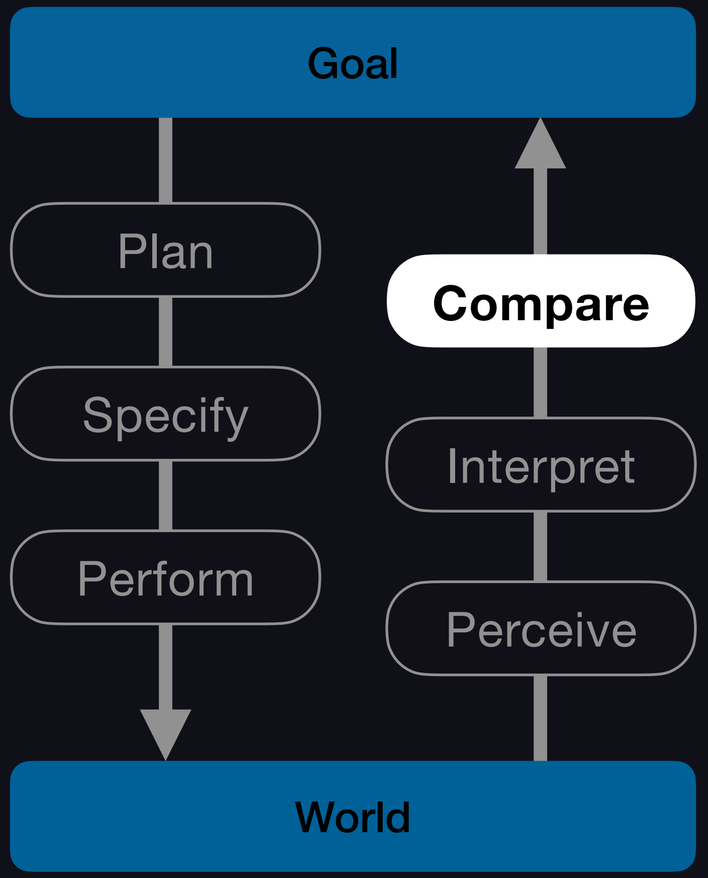




What happened?



What does it mean?


Goal
Plan
Specify
Perform
Compare
Interpret
Perceive
World

Is this OK?

Affordances

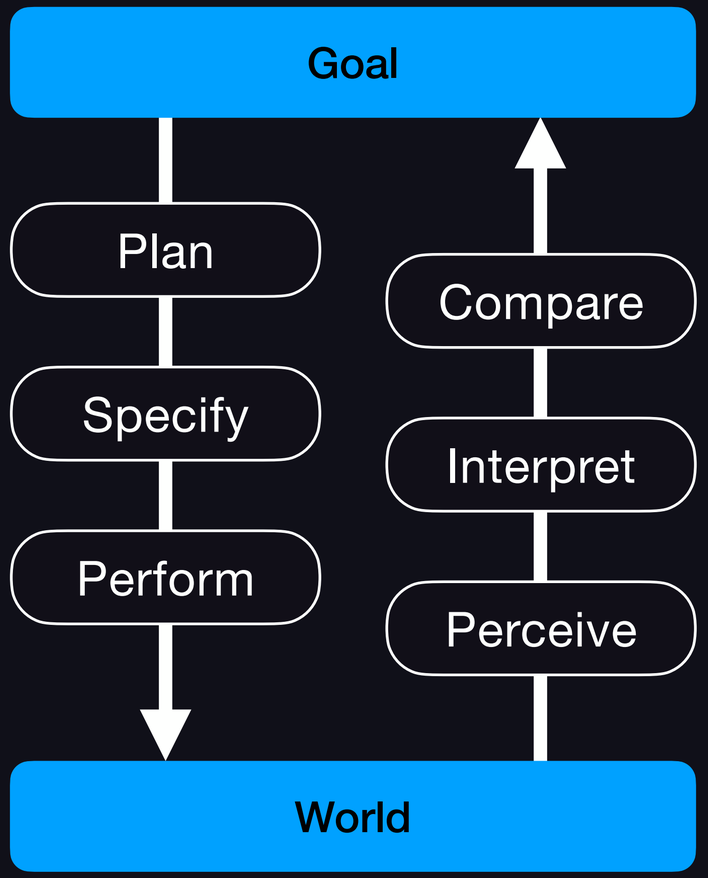
Signifiers

Constraints

Mappings

DISCOVERABILITY

Feedforward



UNDERSTANDABILITY

Feedback

Feedback

Conceptual Models


汉庭

Sticking out

Sticking out

Seems grab-able

Sticking out

Seems grab-able

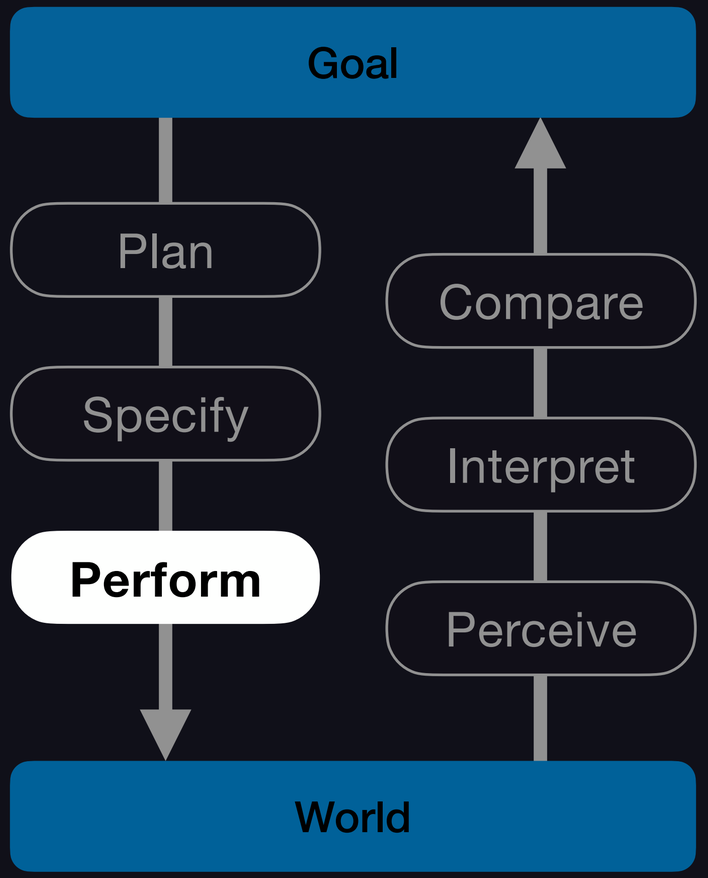
One side is attached to the wall

These are affordances!

- Sticking out
- Seems grab-able
- One side is attached to the wall

Affordances

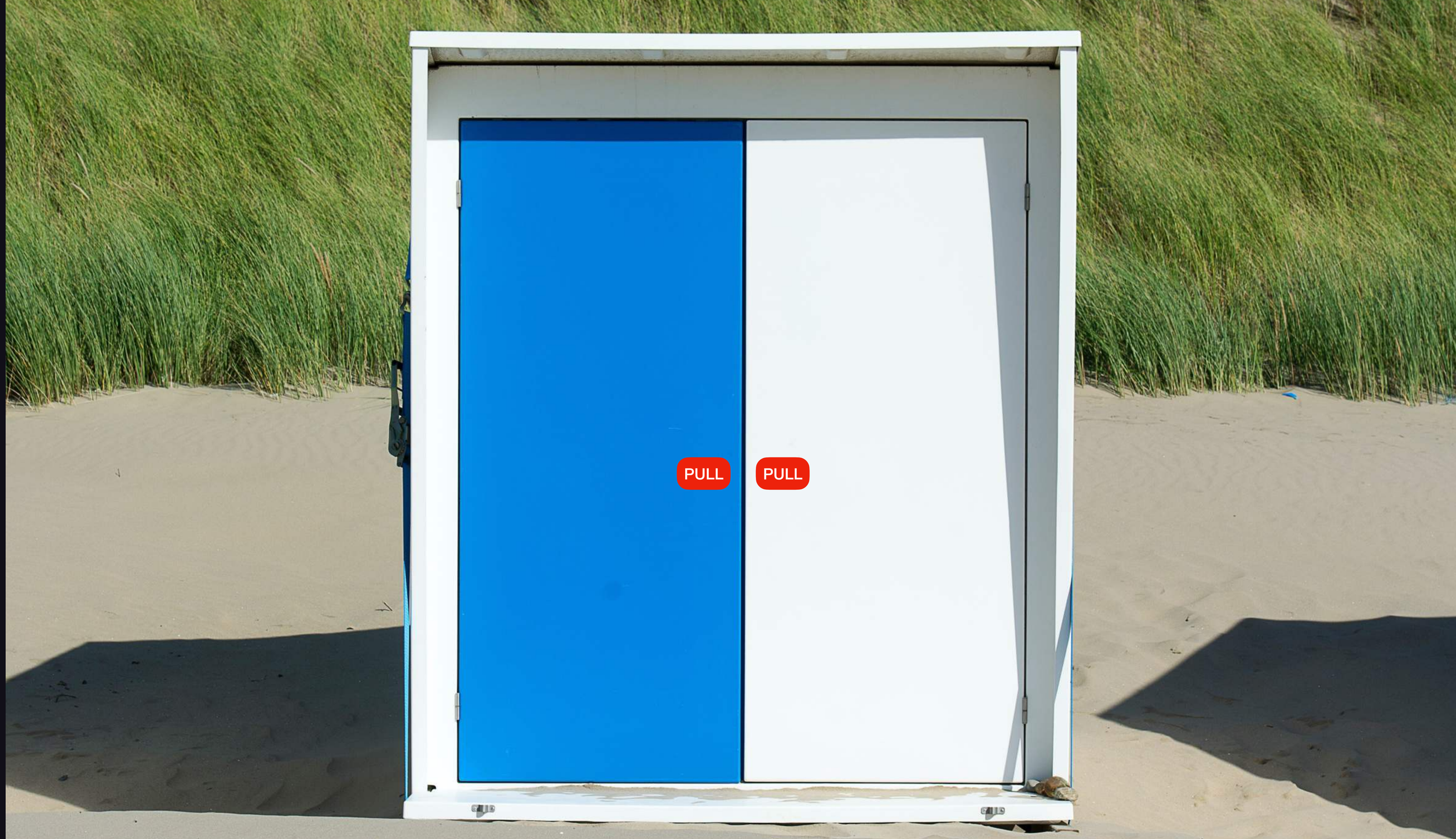
How will you do it?

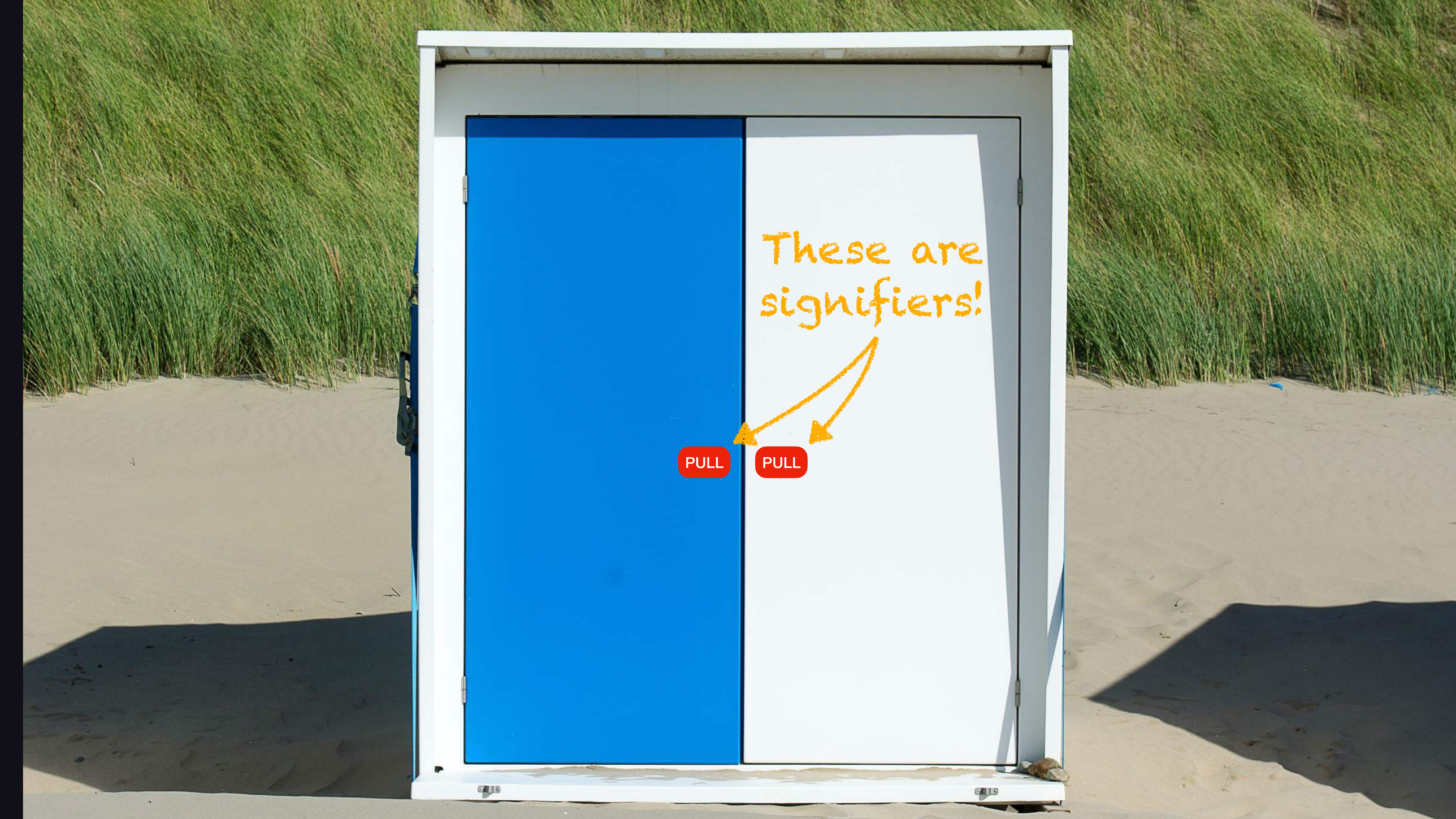


Double door!

~~Double door!~~

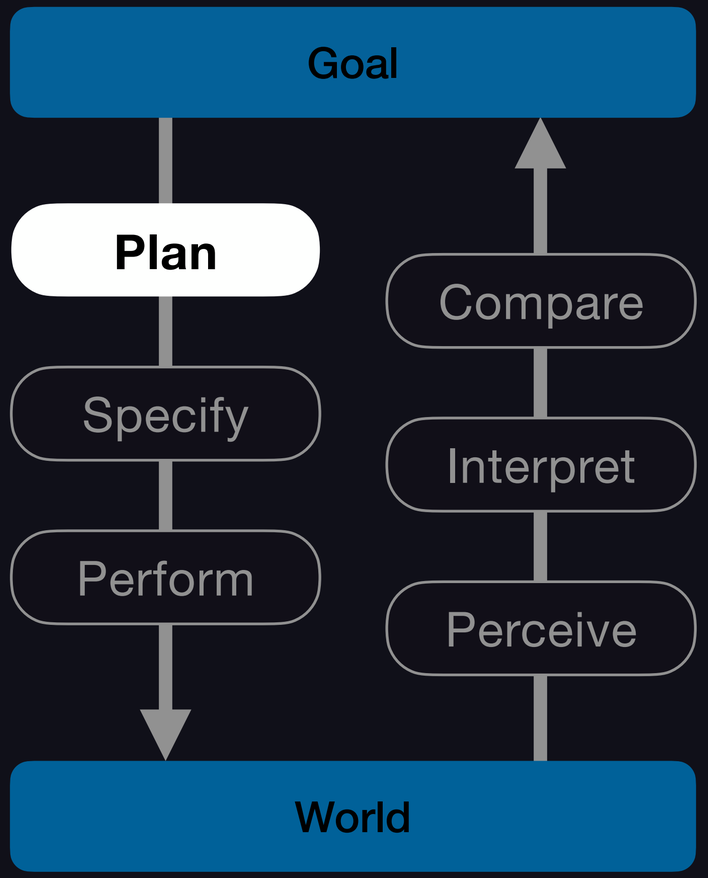
Dumbledore


PULL
PULL

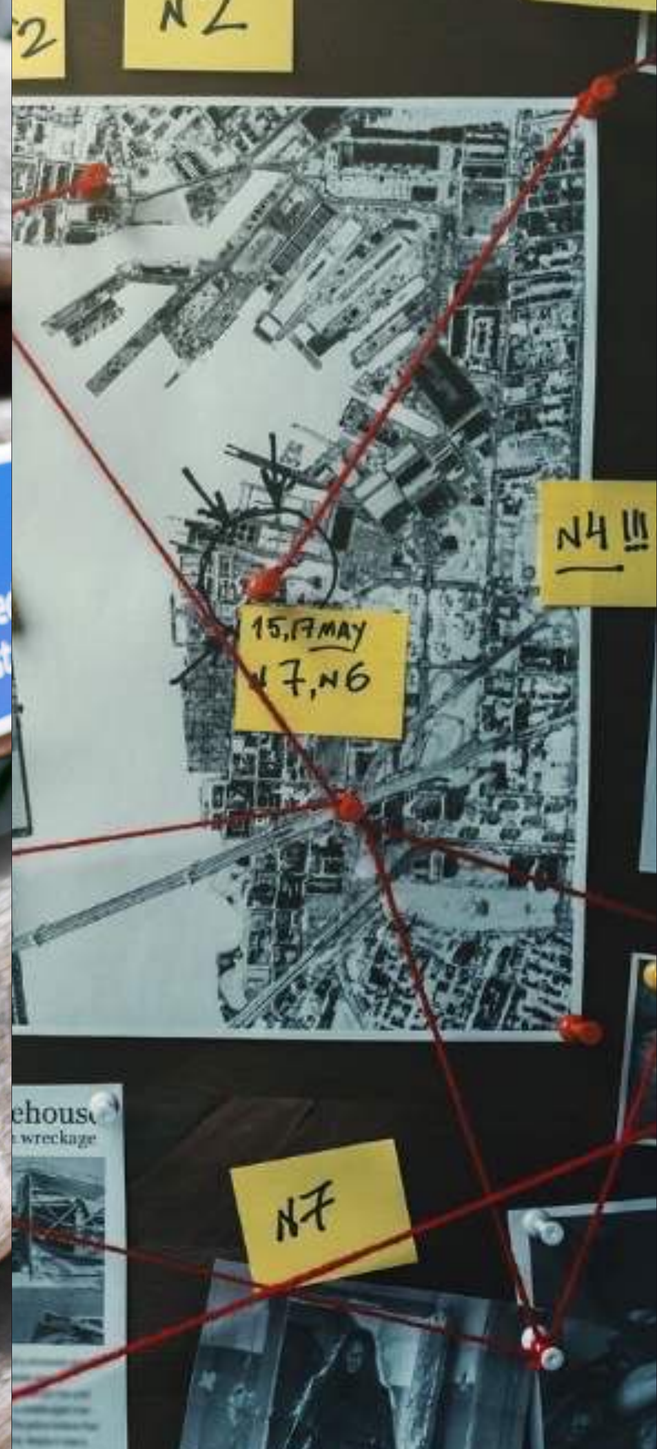

These are signifiers!
PULL
PULL

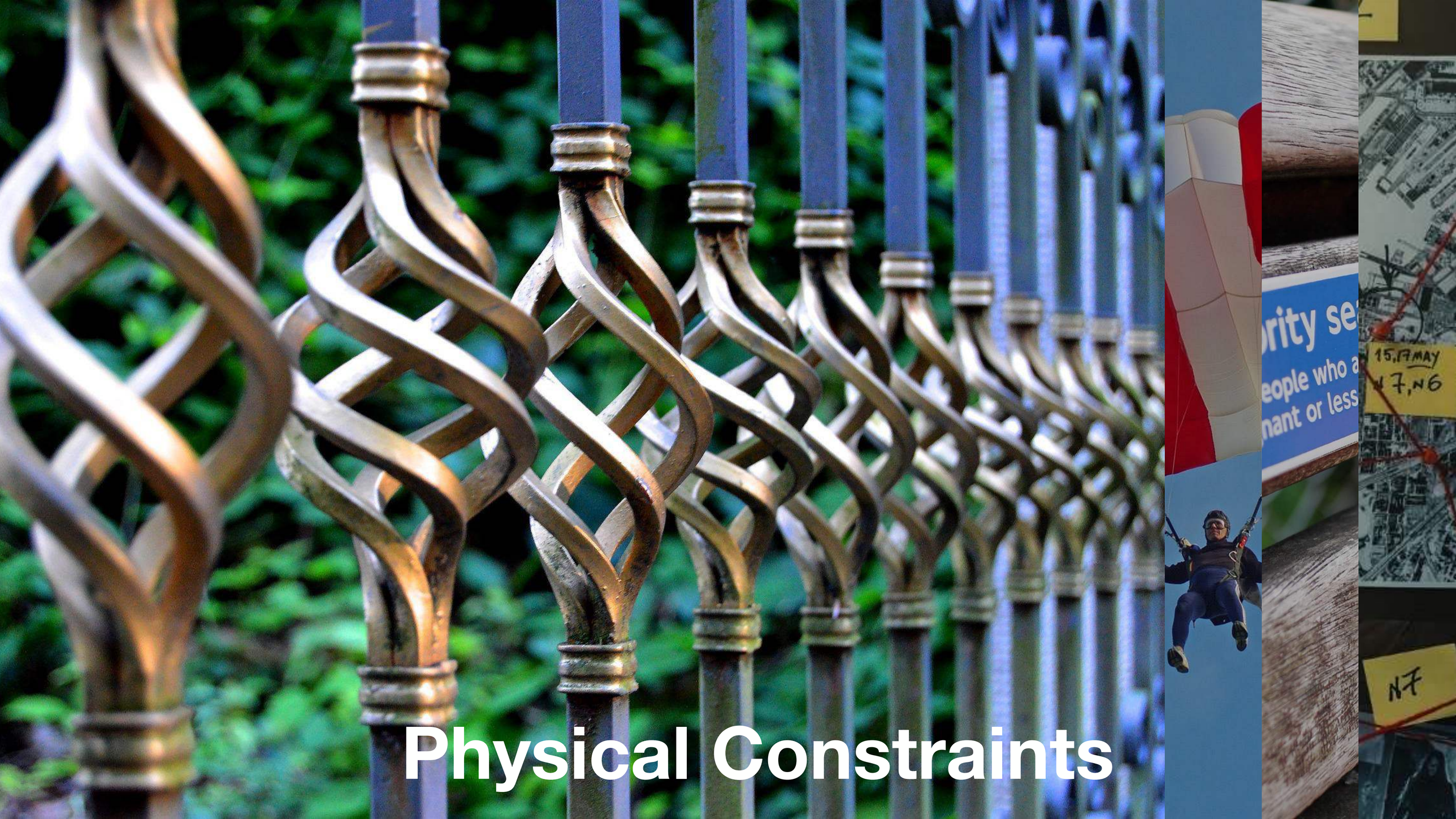
Signifiers

What are your options?




Priority seat
for people who are disable
pregnant or less able to st

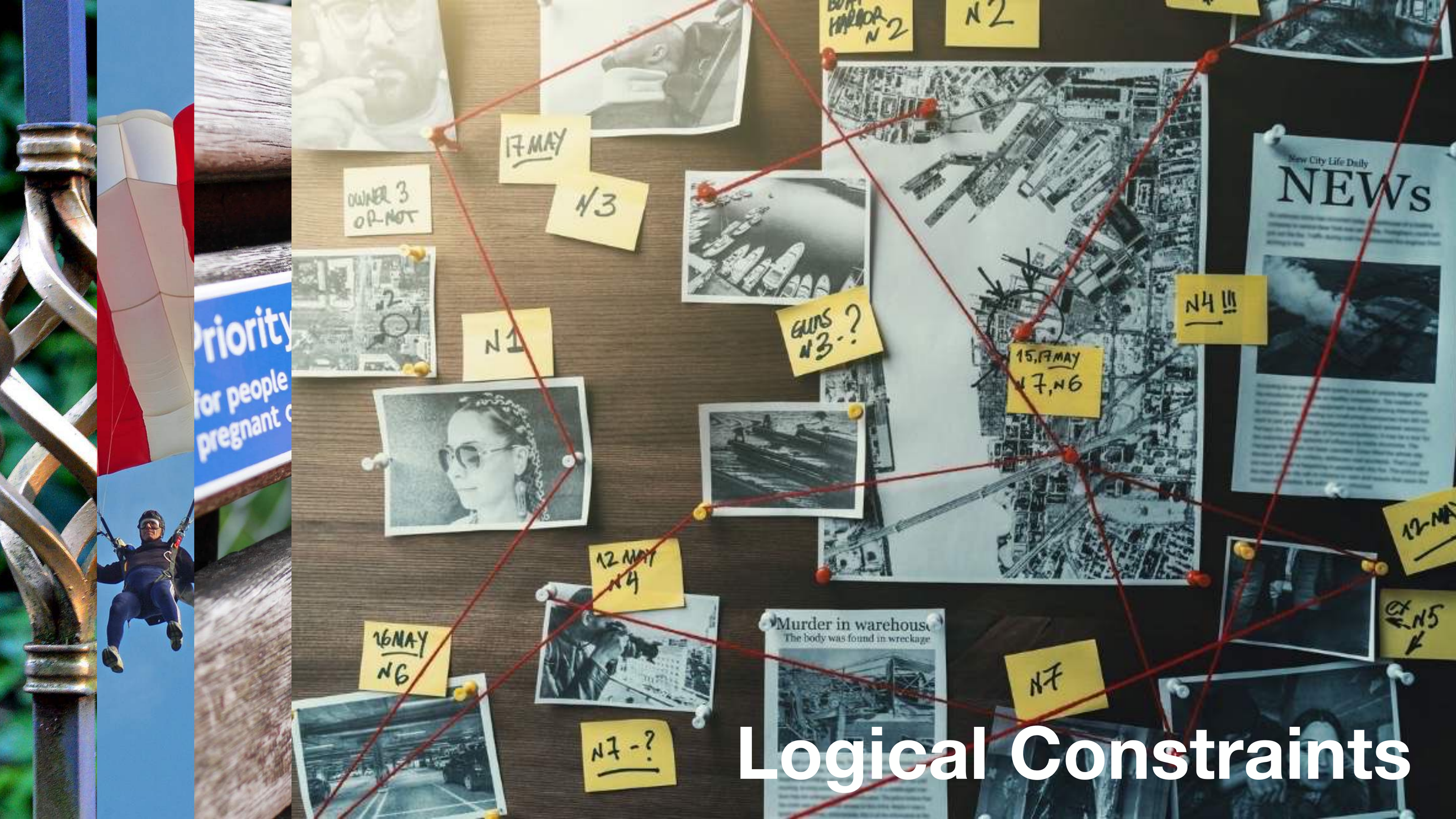

N2
N4
15,17MAY
N7,N6
N7



Physical Constraints


Semantic Constraints

Cultural Constraints



Logical Constraints

“When you have eliminated all which is impossible, then whatever remains, however improbable, must be the truth.”

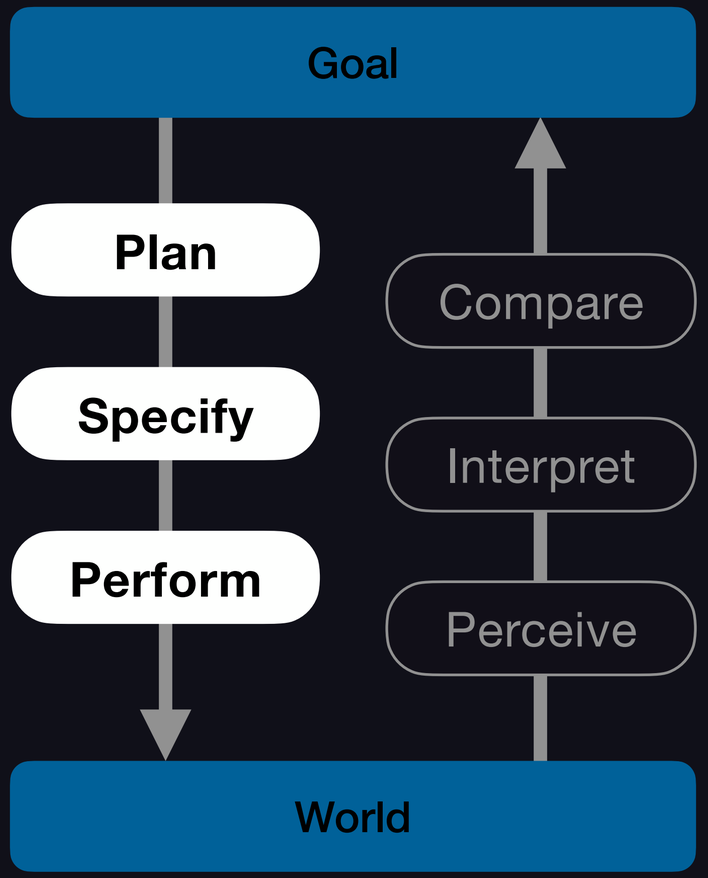
Sir Arthur Conan Doyle

Constraints

What are your options?

What action will you do?

How will you do it?




Bad Mapping

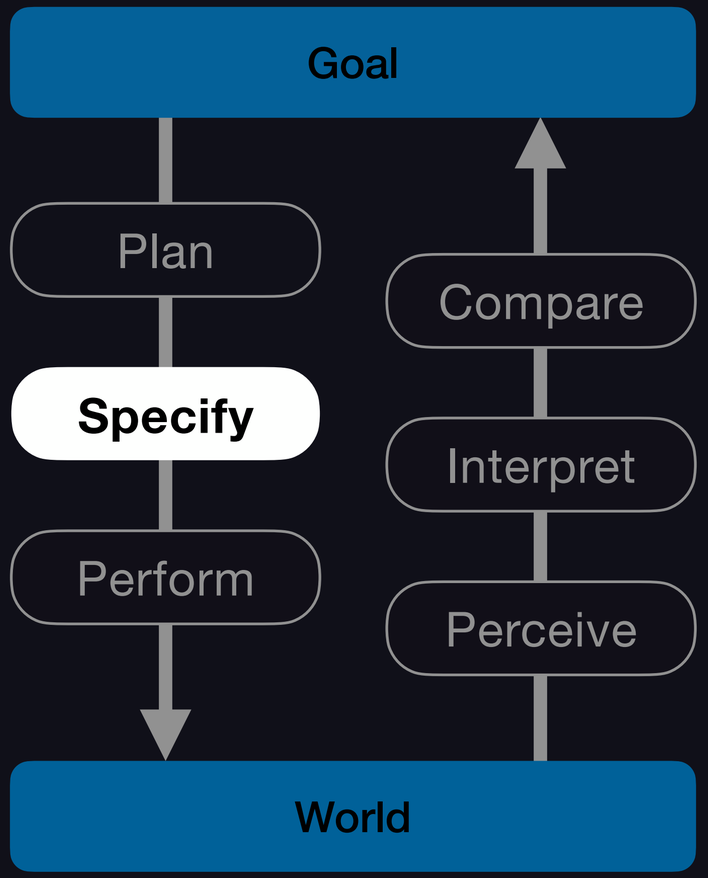

Good Mapping

Grouping in
the same way

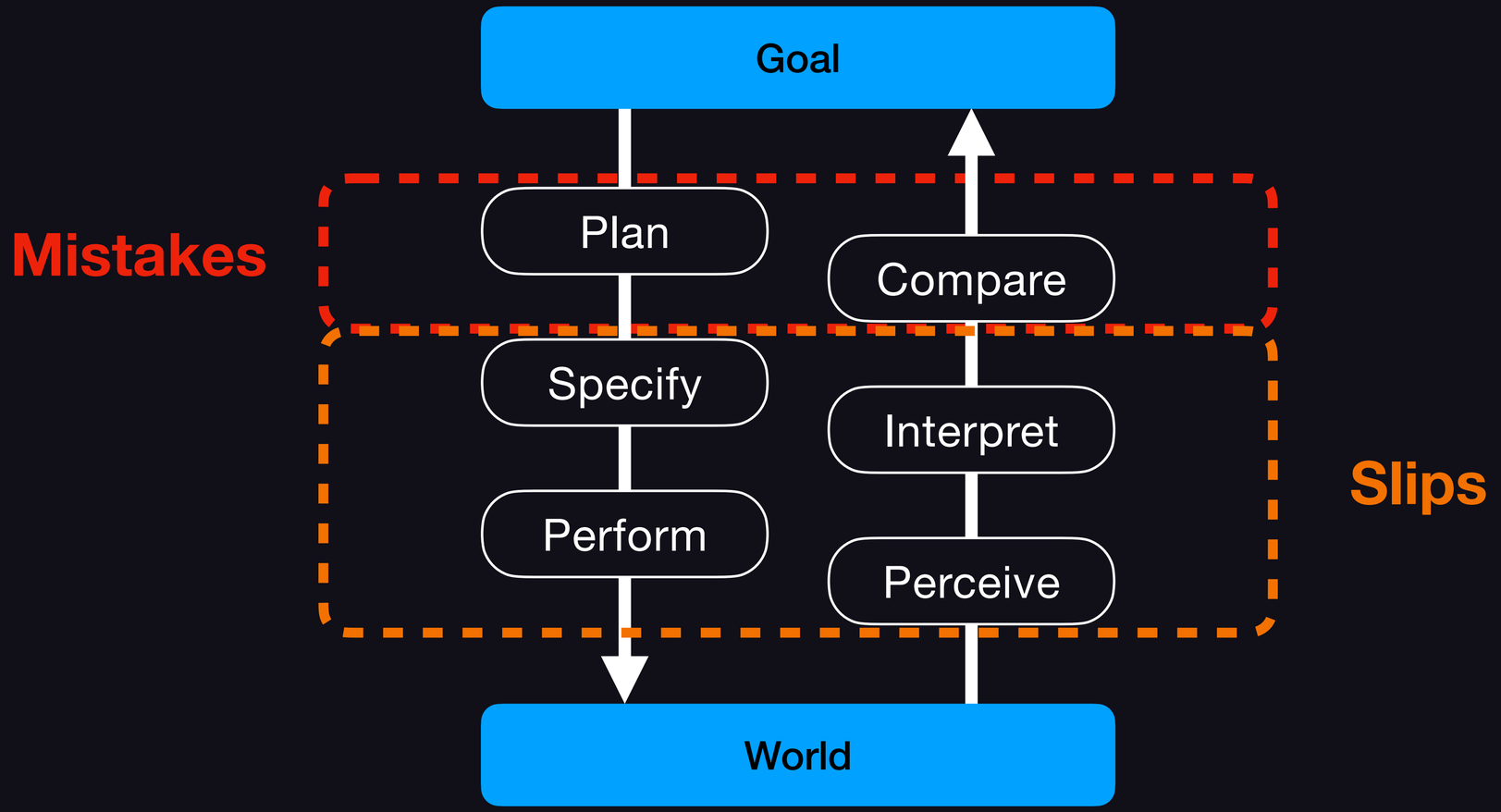

Grouping in
the same way
Good
proximity

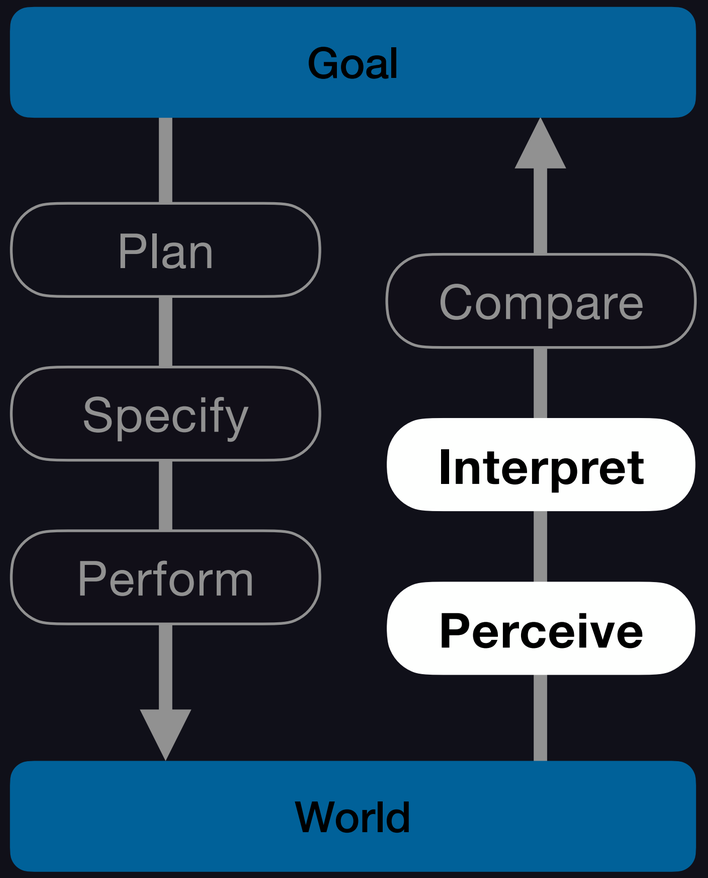
Mappings

What action will you do?




4
L


Goal
Plan
Compare
Mistakes
Specify
Interpret
Slips
Perform
Perceive
World

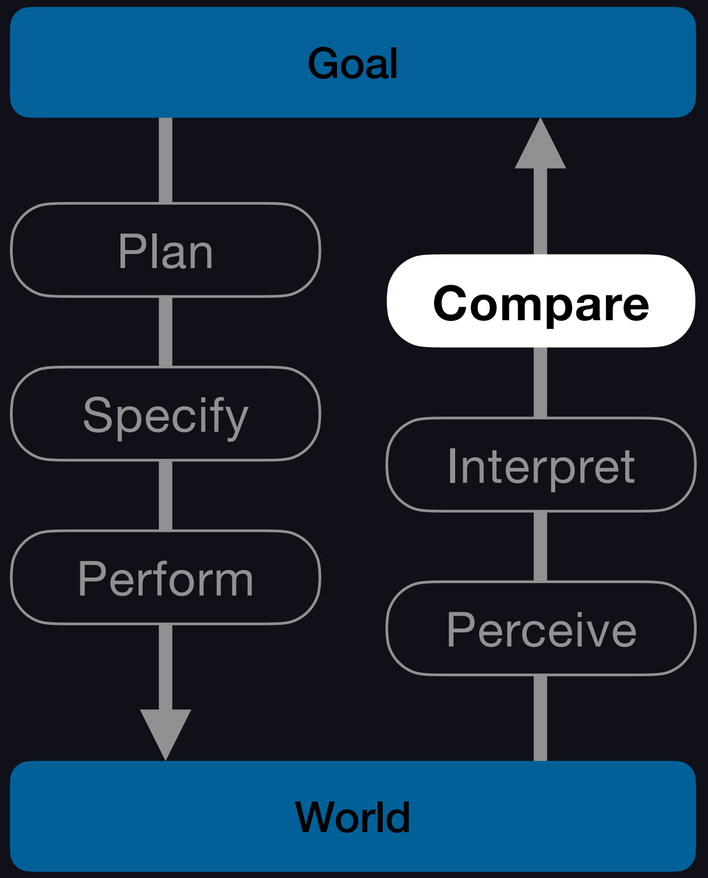


Feedback

What does it mean?

What happened?


SOURCE
VOL
MUTE
PRE-CH
CH LIST
CH
SETTINGS
GUIDE



Conceptual Models

Is this OK?

INTRODUCING

Human-Centered Design

FOR **DEVELOPERS**

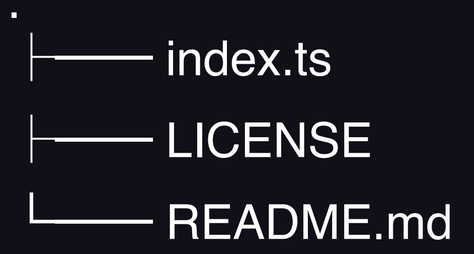
Organizing Things

An Introduction to A Good Project Structure

What did you usually **do**? 🪄✨

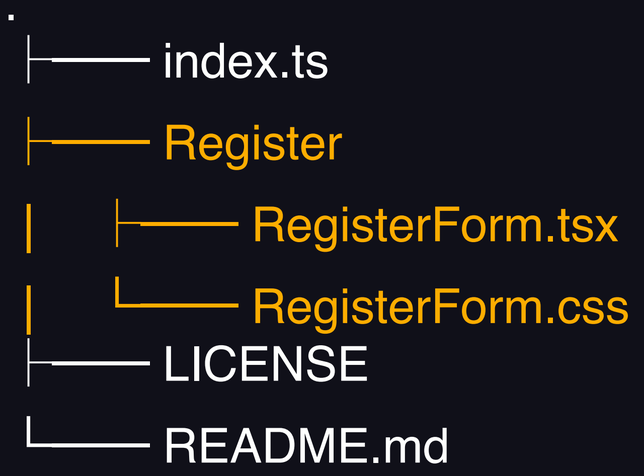
Day 0

Like this?



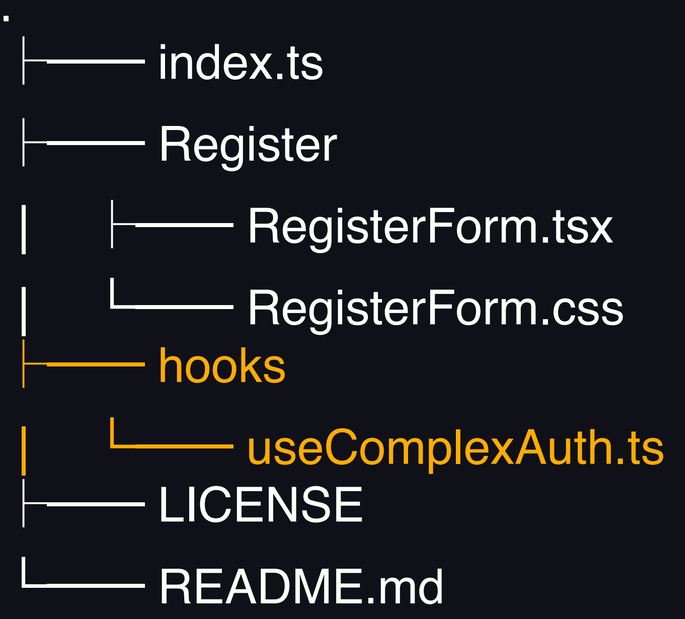
Day 1

Like this?



Day 2

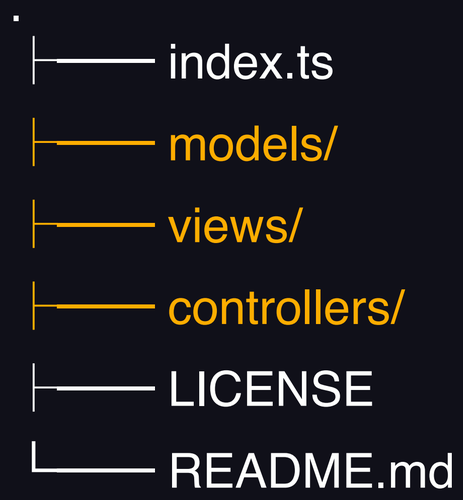
Like this?



Just Let It Evolve Over Time! ⏱️

WAIT! Because Now I know **MVC**

Day 0

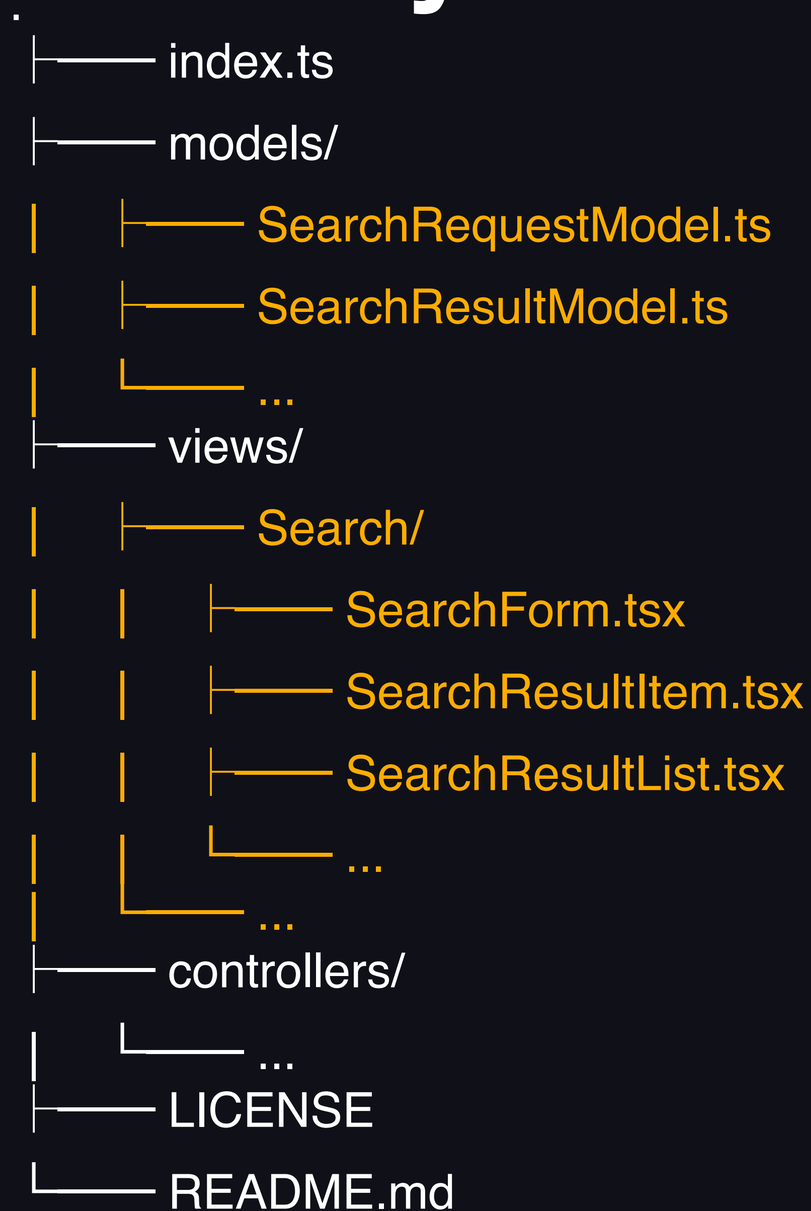


Day 30+

```
.
├── index.ts
├── models/
│   ├── UserModel.ts
│   ├── ShopModel.ts
│   ├── CartModel.ts
│   └── ...
├── views/
│   ├── RegisterForm/
│   │   ├── RegisterForm.tsx
│   │   └── RegisterForm.css
│   ├── Shop/
│   ├── Cart/
│   └── ...
├── controllers/
│   └── ...
├── LICENSE
```

What If You Need to Add **"Search 🔎"** Feature?

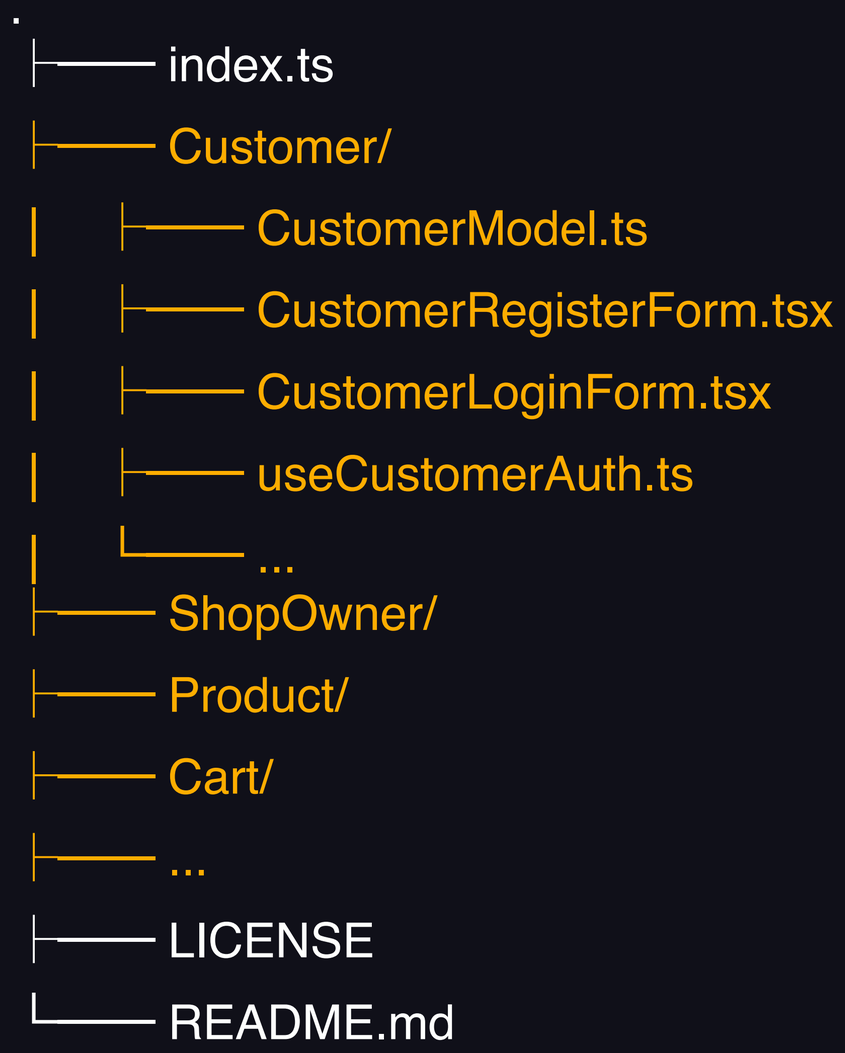
Day 31+


.
index.ts
models/
SearchRequestModel.ts
SearchResultModel.ts
...
views/
Search/
SearchForm.tsx
SearchResultItem.tsx
SearchResultList.tsx
...
...
controllers/
...
LICENSE
README.md

WAIT! I have Better Idea!

Let’s Divide It By **Domain**!

Day 0

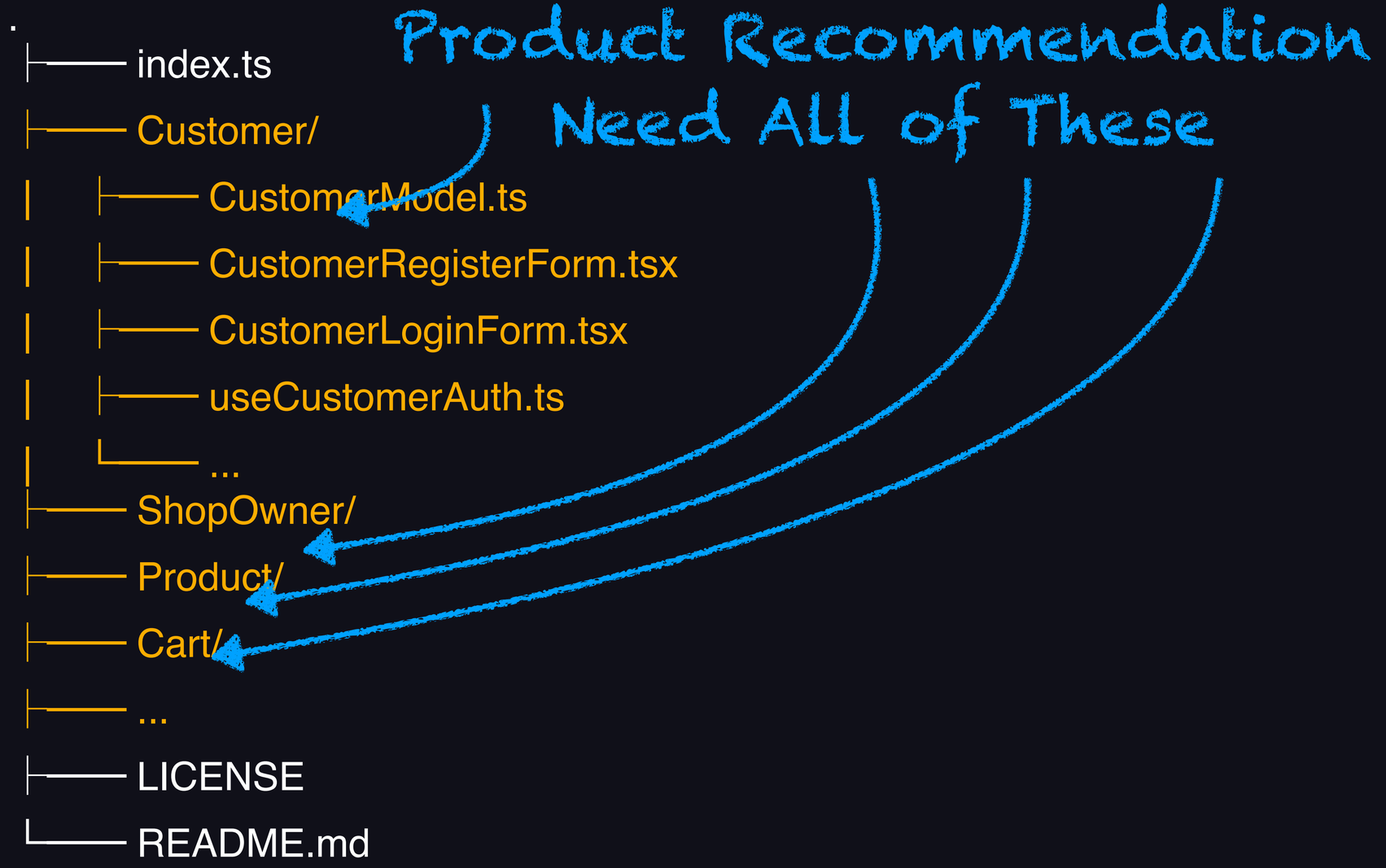

.
index.ts
Customer/
CustomerModel.ts
CustomerRegisterForm.tsx
CustomerLoginForm.tsx
useCustomerAuth.ts
...
ShopOwner/
Product/
Cart/
...
LICENSE
README.md

Great! No Problems Now 😎

Let’s Implement

"Product Recommendation! 📣"

Day 0


Product Recommendation
Need All of These
.
index.ts
Customer/
CustomerModel.ts
CustomerRegisterForm.tsx
CustomerLoginForm.tsx
useCustomerAuth.ts
...
ShopOwner/
Product/
Cart/
...
LICENSE
README.md

Where should We Put
"Product Recommendation"? 🤔

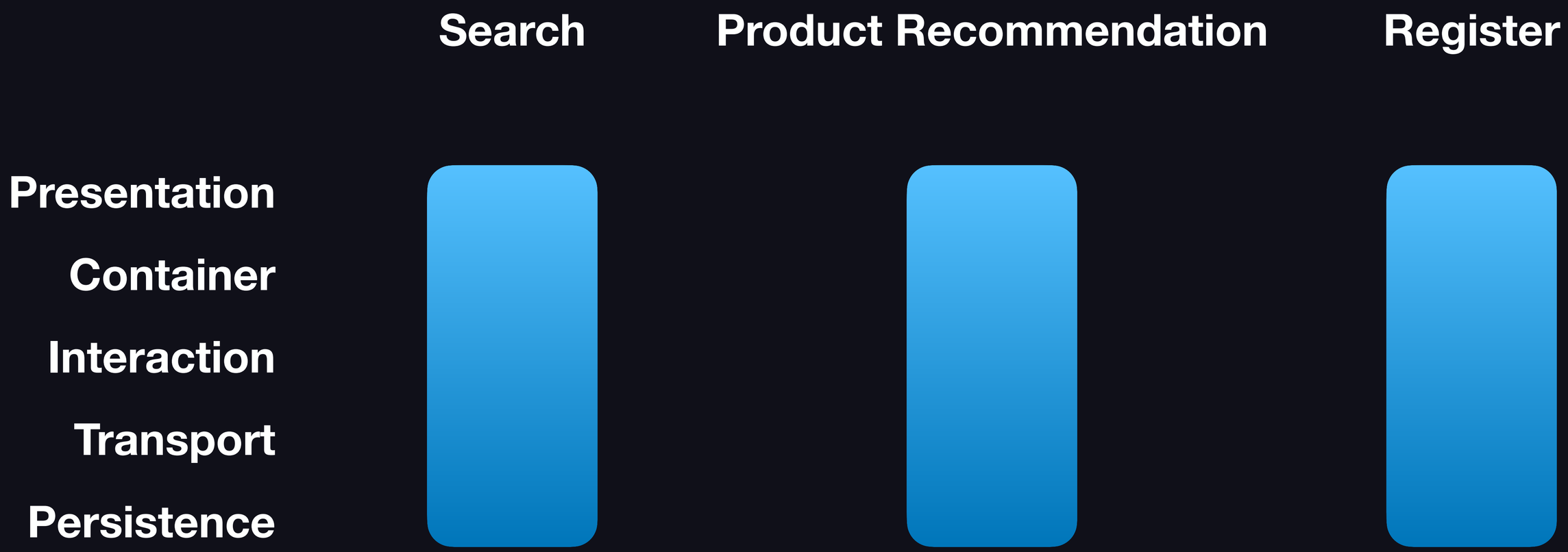
INTRODUCING

Feature-driven Project Structure

Features

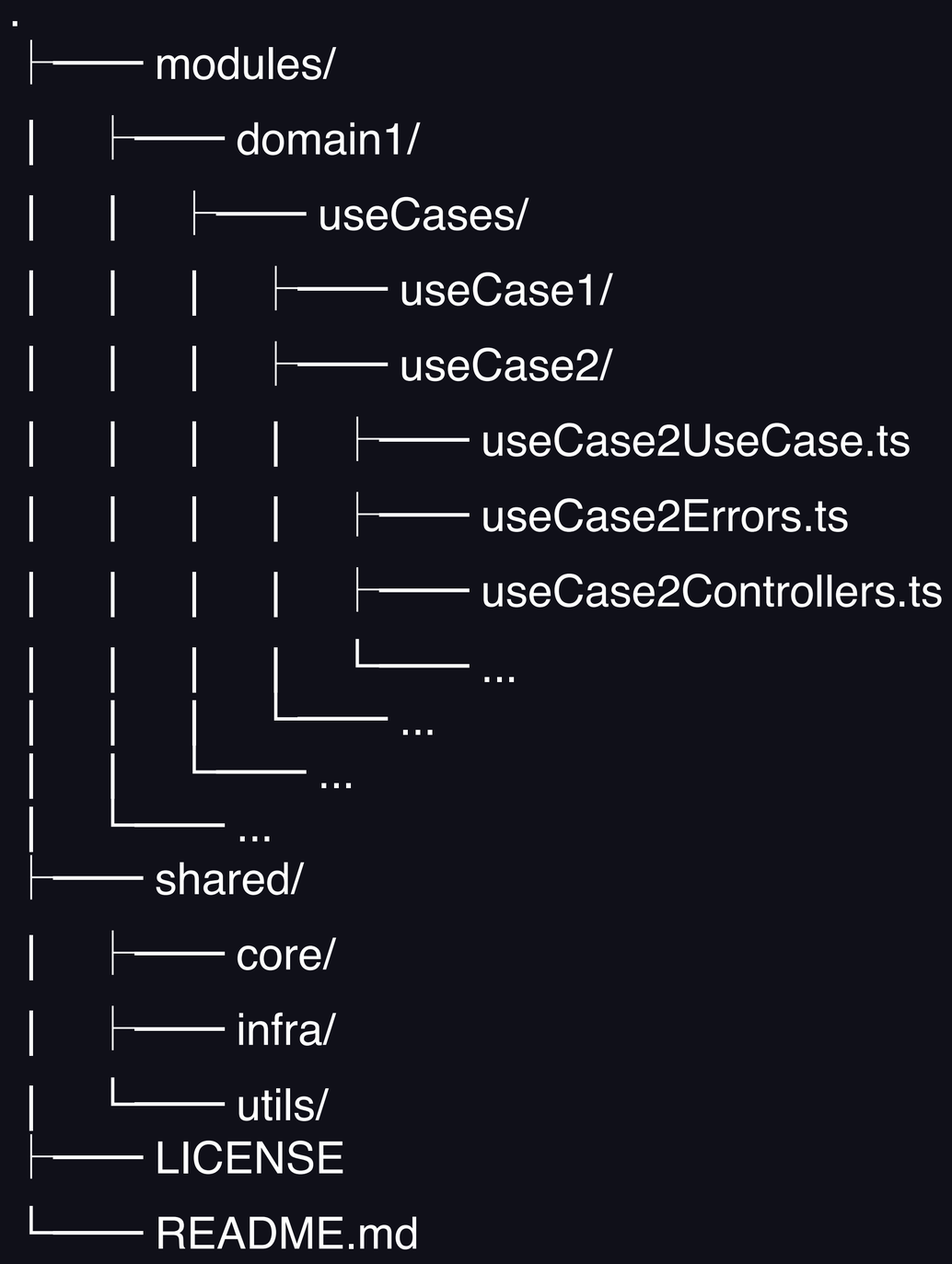
=

Use Cases


Search
Product Recommendation
Register
Presentation
Container
Interaction
Transport
Persistence

If something is used by **multiple features**? 🤔

It belongs to a "**shared** ♻" folder!


.
modules/
domain1/
useCases/
useCase1/
useCase2/
useCase2UseCase.ts
useCase2Errors.ts
useCase2Controllers.ts
...
...
...
...
shared/
core/
infra/
utils/
LICENSE
README.md

“Software architectures are structures that support the use cases of the system. Just as the plans for a house or a library scream about the use cases of those buildings, so should the **architecture of a software application scream about the use cases of the application**”

Robert C. Martin

Documentation & Repositories

One of Ways to Manage Complexity

Documentation Should Answer These Questions

What is this?

What is it for?

How Do I Get Started?

How Do I Run Test?

How Do I Debug It?

Where Are The Features?

Documentation Should Answer These Questions

What is this? e.g. It's a shopping front-end application

What is it for?

How Do I Get Started?

How Do I Run Test?

How Do I Debug It?

Where Are The Features?

Documentation Should Answer These Questions

What is this? e.g. It's a shopping front-end application

What is it for? e.g. Landing page front-end service

How Do I Get Started?

How Do I Run Test?

How Do I Debug It?

Where Are The Features?

Documentation Should Answer These Questions

What is this? e.g. It's a shopping front-end application

What is it for? e.g. Landing page front-end service

How Do I Get Started? e.g. Step 1 ...

How Do I Run Test?

How Do I Debug It?

Where Are The Features?

Documentation Should Answer These Questions

What is this? e.g. It's a shopping front-end application

What is it for? e.g. Landing page front-end service

How Do I Get Started? e.g. Step 1 ...

e.g. Use command ... How Do I Run Test?

How Do I Debug It?

Where Are The Features?

Documentation Should Answer These Questions

What is this? e.g. It's a shopping front-end application

What is it for? e.g. Landing page front-end service

How Do I Get Started? e.g. Step 1 ...

e.g. Use command ... How Do I Run Test?

e.g. Use command ... How Do I Debug It?

Where Are The Features?

Documentation Should Answer These Questions

What is this? e.g. It's a shopping front-end application

What is it for? e.g. Landing page front-end service

How Do I Get Started? e.g. Step 1 ...

e.g. Use command ... How Do I Run Test?

e.g. Use command ... How Do I Debug It?

Your Project Structure Should Scream It! Where Are The Features?

What If I Have A Question
"How something works"? 🤔

"Tests" Should Answer It!

"Tests" As **Documentation** 

Naming Things

One of the Hardest Problems in Computer Science

Principle #1 Consistency & Uniqueness

Each concept should be represented
by a single, unique name

Principle #2 Understandability

A name should describe the
concept it represents

Principle #3 Specificity

A name shouldn’t be
overly vague or overly specific

Principle #4 Brevity

A name should be neither
overly short nor overly long

Principle #5 Searchability

A name should be easily found
across code, documentation,
and other resources

Principle #6 Pronounceabiltiy

A name should be easy to
use in common speech

Principle #7 Austerity

A name should not be clever or rely on temporary concepts

More Info

https://www.namingthings.co

Comments

Is It A Grave Sin to Use Comments?

Code should explain
What 🕵🏻 and **How** 👷🏻‍♀️,

Comment should explain **Why** 🧐

BUT, this code really need to be explained in comment because it is **hard to understand** 😕!

There’s a chance that **refactoring** 👨‍🔧
your code could help!

Formattings & Styles

We Shouldn’t Do It Like A Fashionista

Coding

=

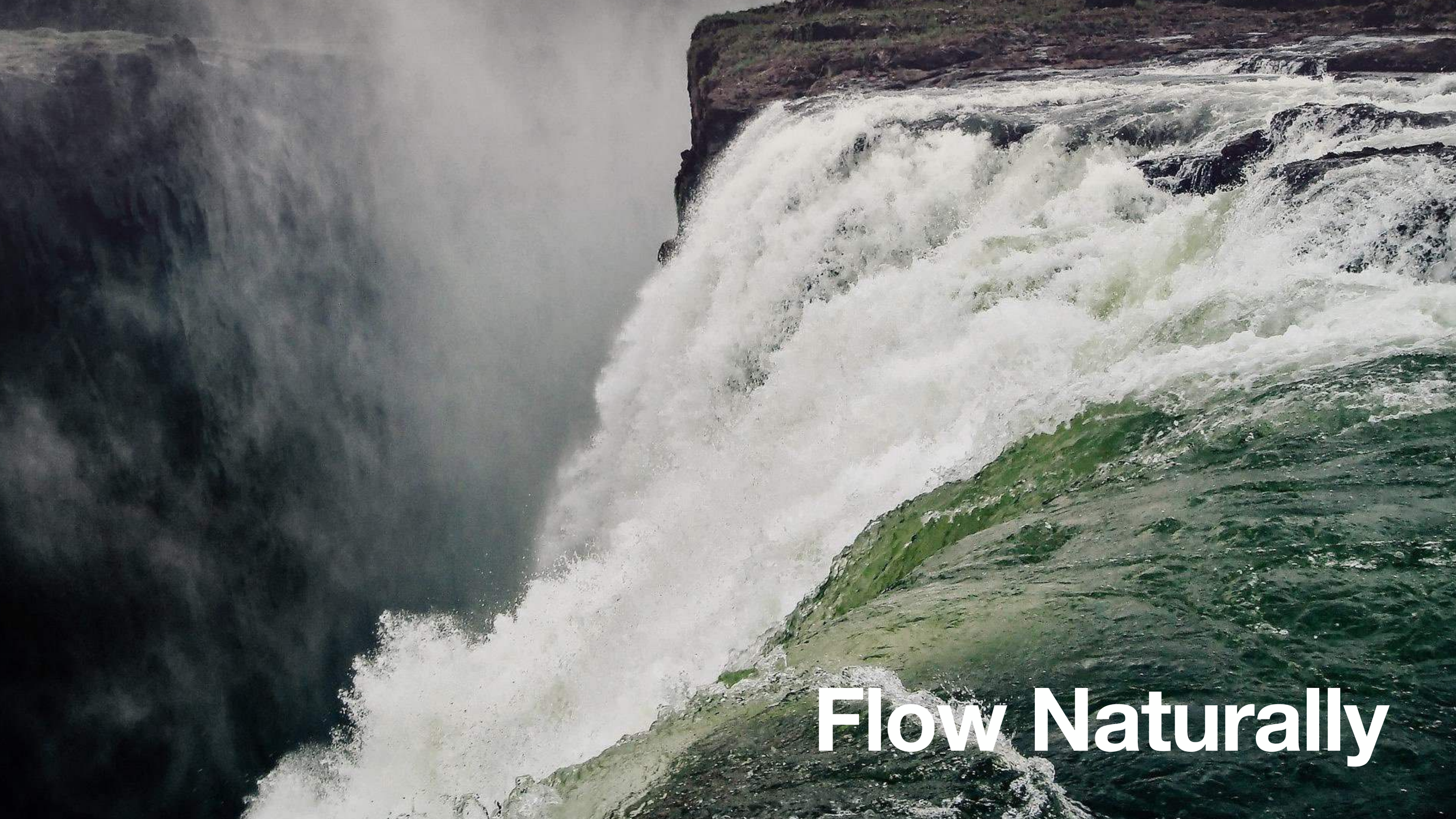
Storytelling

What are things
they have in **common**? 🤔


Good Visual
Appearance

Rule #1
Be Consistence

Rule #2
Use Proper Whitespace


Flow Naturally

Rule #3
Logically Order Code


Give Overview,
then Dive Deeper

Rule #4
Maintain Level of Abstraction

Bite-sized Information

Rule #5
Keep File Small

Adapt to a Culture

Rule #6
Follow Language Convention

Types

One of Your Best Friends

Use **Statically-typed** Languages

Prevent Silly
MISTAKES

ABSTRACTION
Techniques

Enforce
POLICY

Statically-typed
LANGUAGES

Make Implicit
EXPLICIT

Communicate
DESIGN INTENT

SCALE WELL

Errors & Exceptions

Another One of Your Best Friends

What did you usually **do**? 🪄✨

Like this?

```
const findUser = (userId: UserID) => {
  // DB query and conversion code
  // goes here

  if (!result.found) {
    return null
  }

  return user
}
```

Or this?

```
const findUser = (userId: UserID) => {
  // DB query and conversion code
  // goes here

  if (!result.found) {
    console.error("User not found")
    throw new Error(`User with user id: ${userId} not found`)
  }

  return user
}
```

Before Answer This Question

“What should We Do?”…

How the "**Exception ⚠️**" is Different From "**Error ❌**"? 🤔

Error ❌

Expected

Is a Part of a Domain

Essential Complexity

Exception ⚠️

Unexpected

Outside of Our Control

Accidental Complexity

Error ❌

UserAlreadyExisted

InvalidEmail

WrongPassword

Exception ⚠️

Database Server is Down

3rd Party API Server is Down

3rd Party API Sent Only 500 Response

Error ❌

Model Error As **"Types"**

Exception ⚠️

Surround It With try/catch

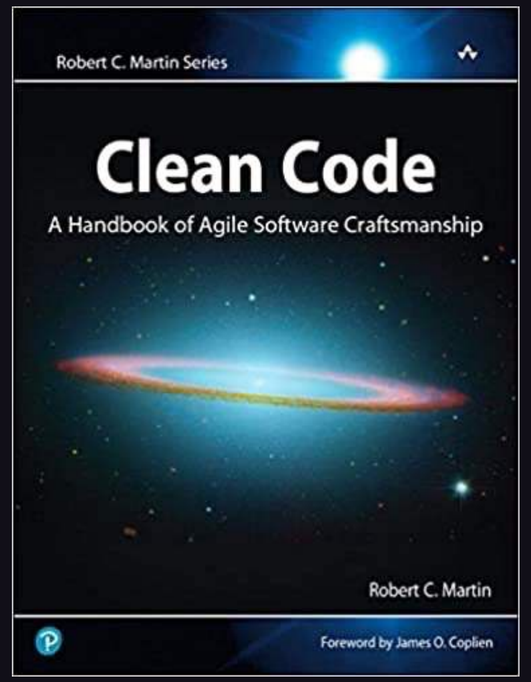
Key Takeaways

Develop for Users

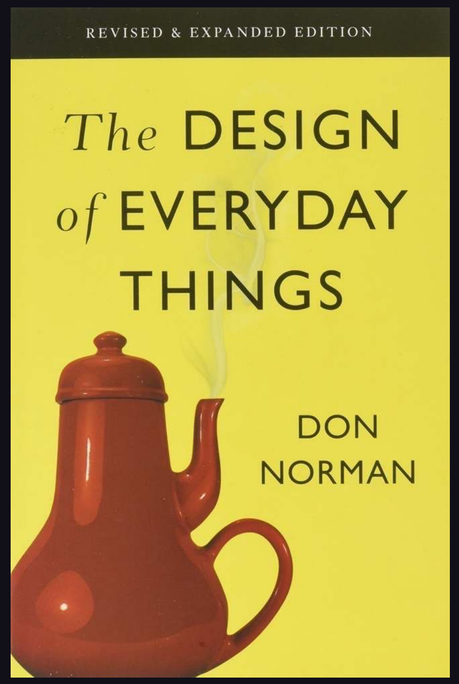
Develop for Human

Not tolerate
messy code

More Like This



Clean Code
by Robert C. Martin



The Design of Everyday Things
by Don Norman

Clean Architecture
by Robert C. Martin

"Programs must be written
for people to read,
and only incidentally for
machines to execute."

Harold Abelson