Click

Click

Click

Click

Click

Click

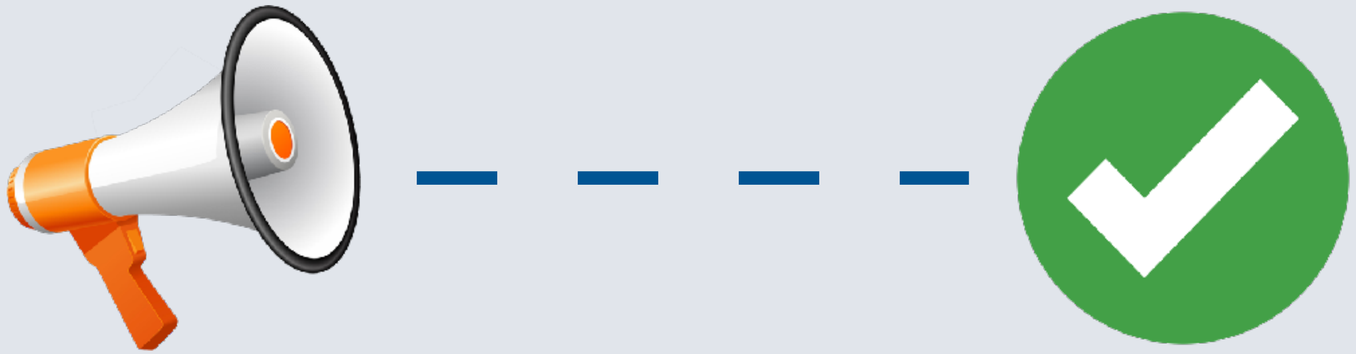
Click

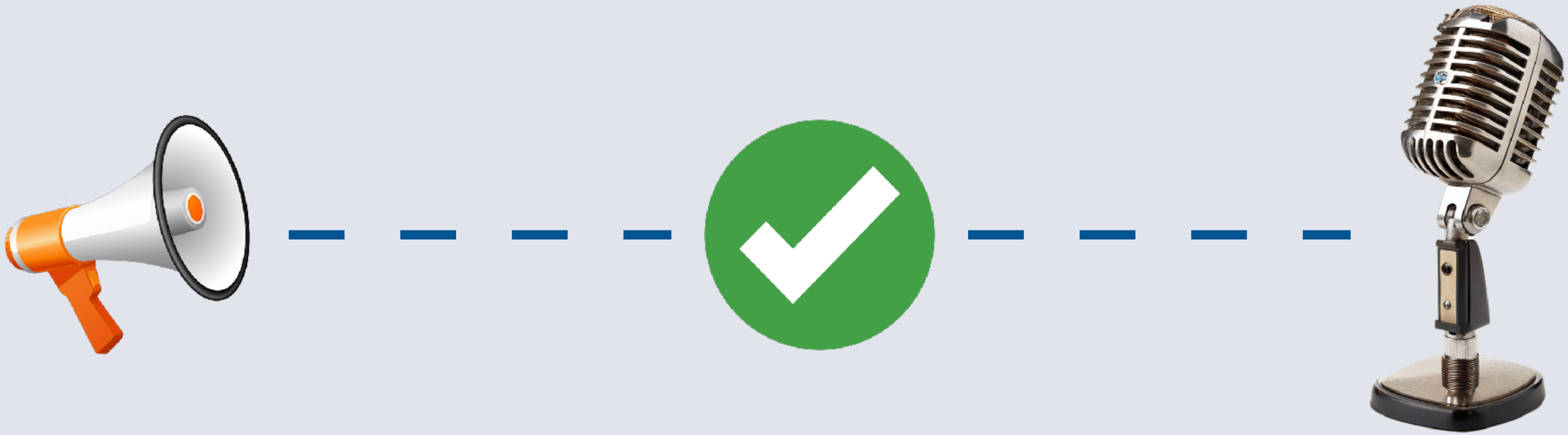
Click

Click

Click













Types Are Worth The Typing

Types Are Worth The Typing

Types Are Still Worth The Typing

Types Are Still Worth The Typing

(Sometimes)

LLM output is not "new machine code"

LLM output is not "new machine code"

Keep caring about the language

LLM output is not "new machine code"

Keep caring about the language

Pick one with a great type system

LLM output is not "new machine code"

Keep caring about the language

Pick one with a great type system

And learn it well

LLM output is not "new machine code"

Keep caring about the language

Pick one with a great type system

And learn it well

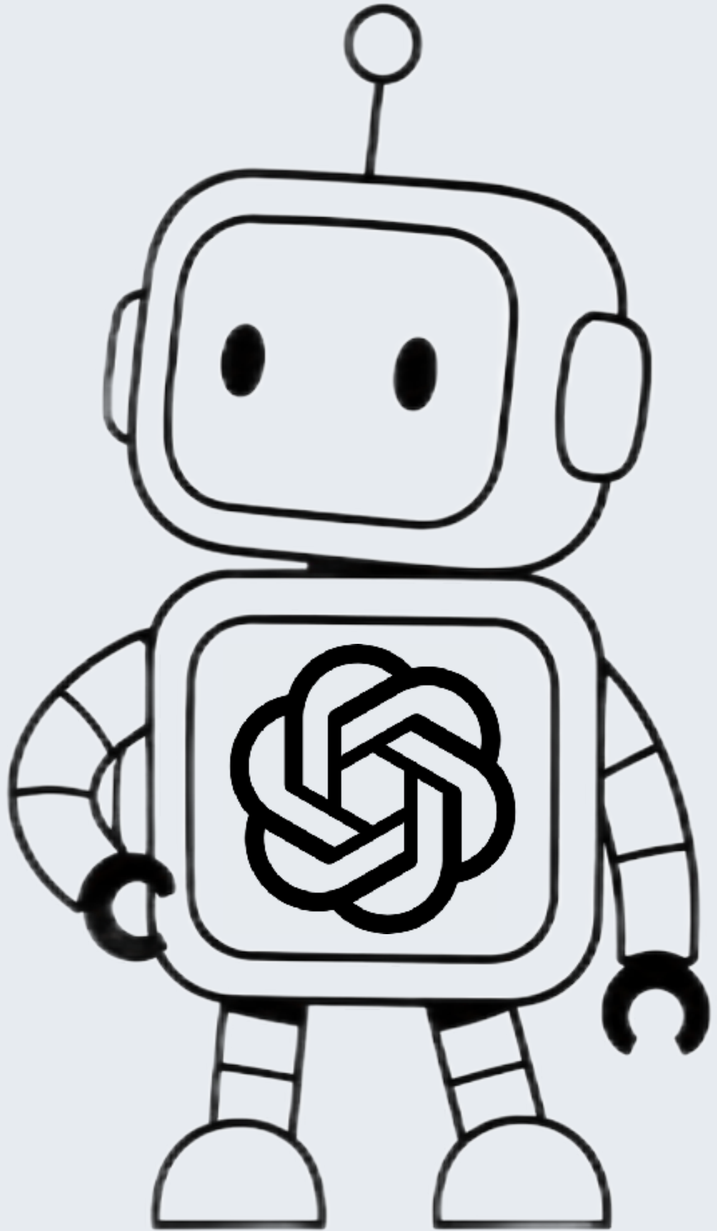
LLM output is not "new machine code"

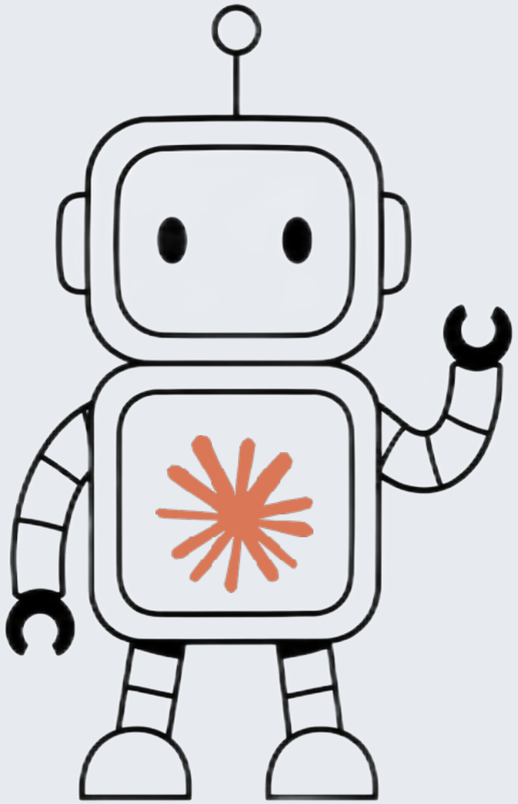
Keep caring about the language

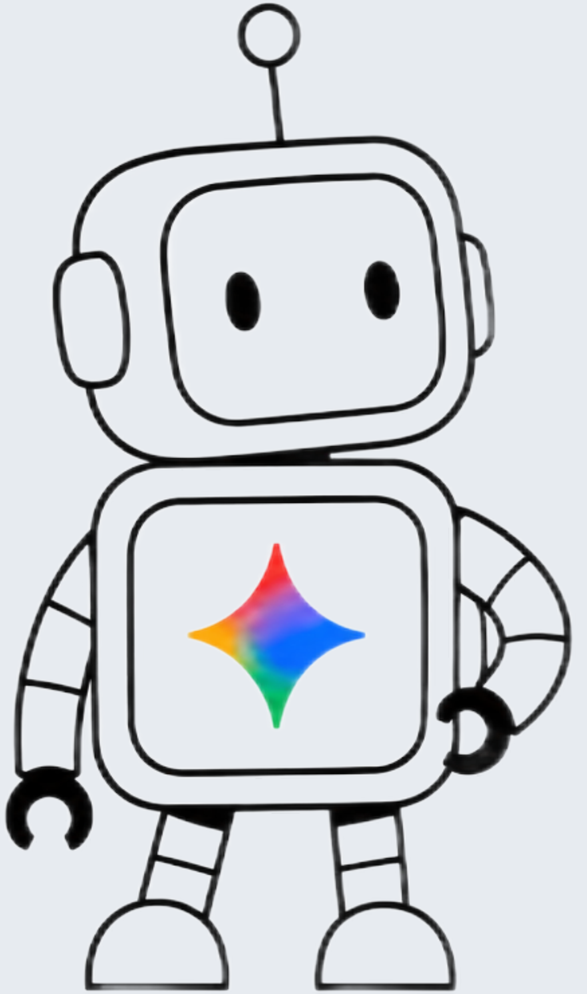
Pick one with a great type system

And learn it well

Why care about the language?

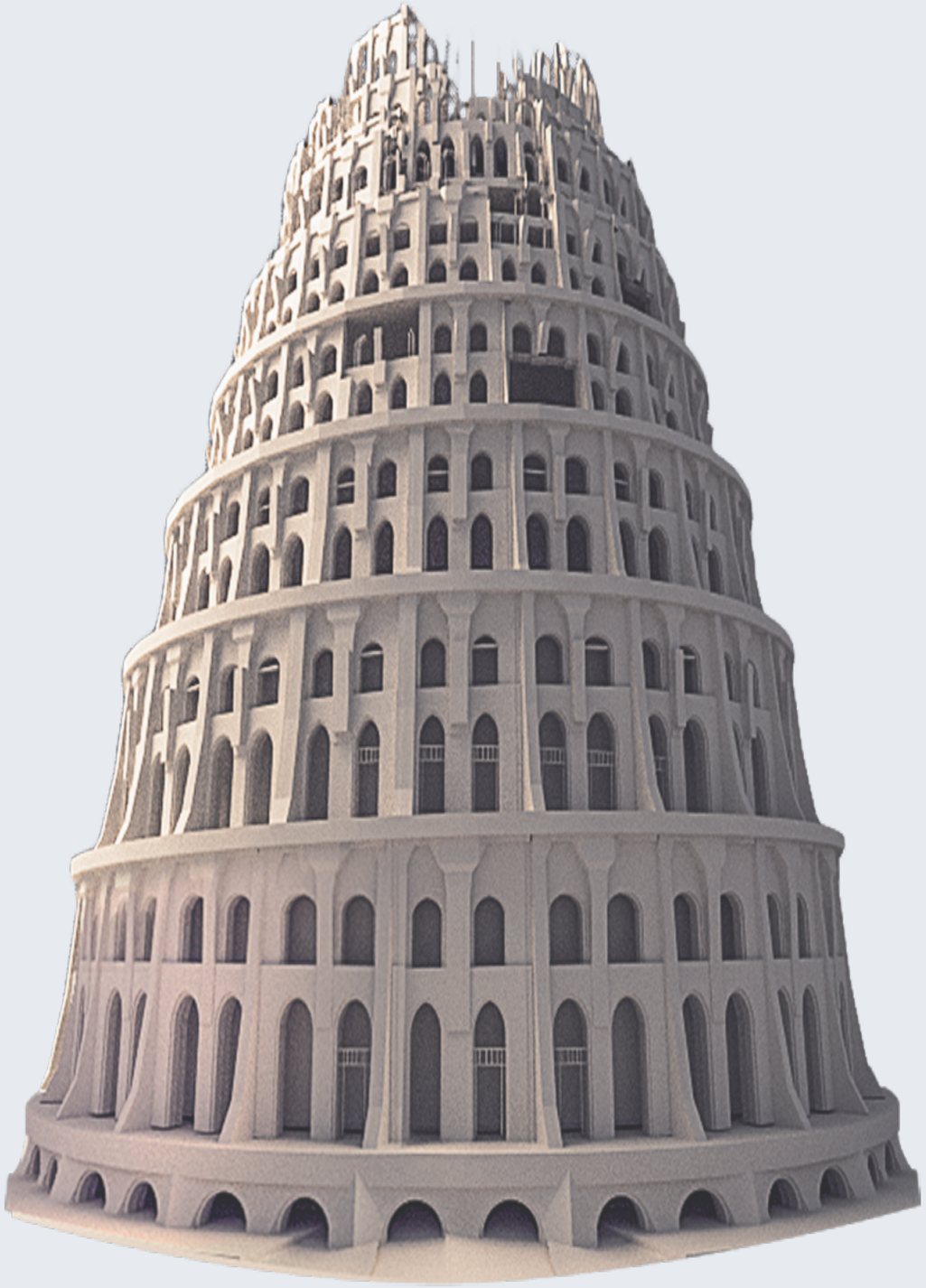


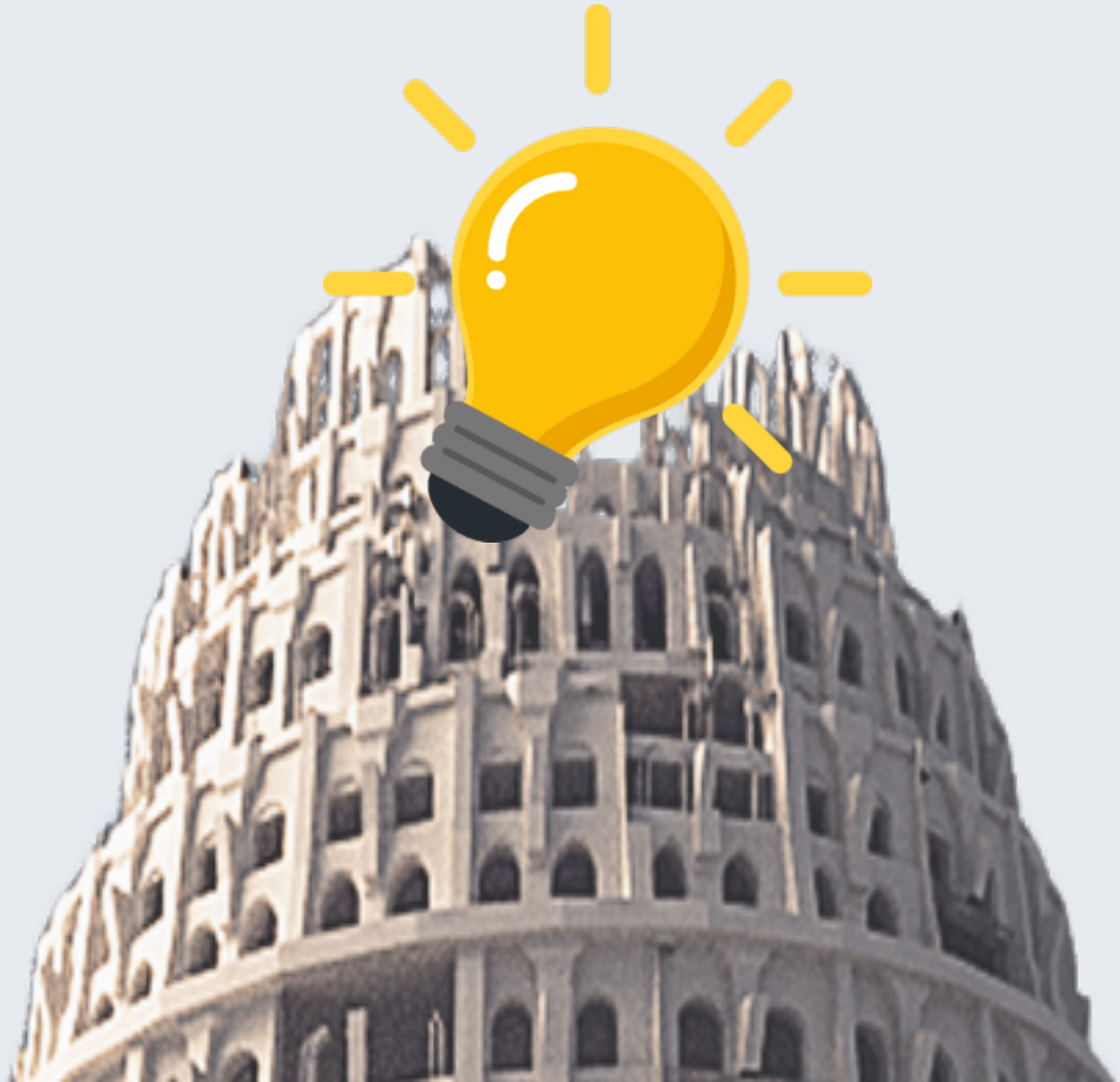


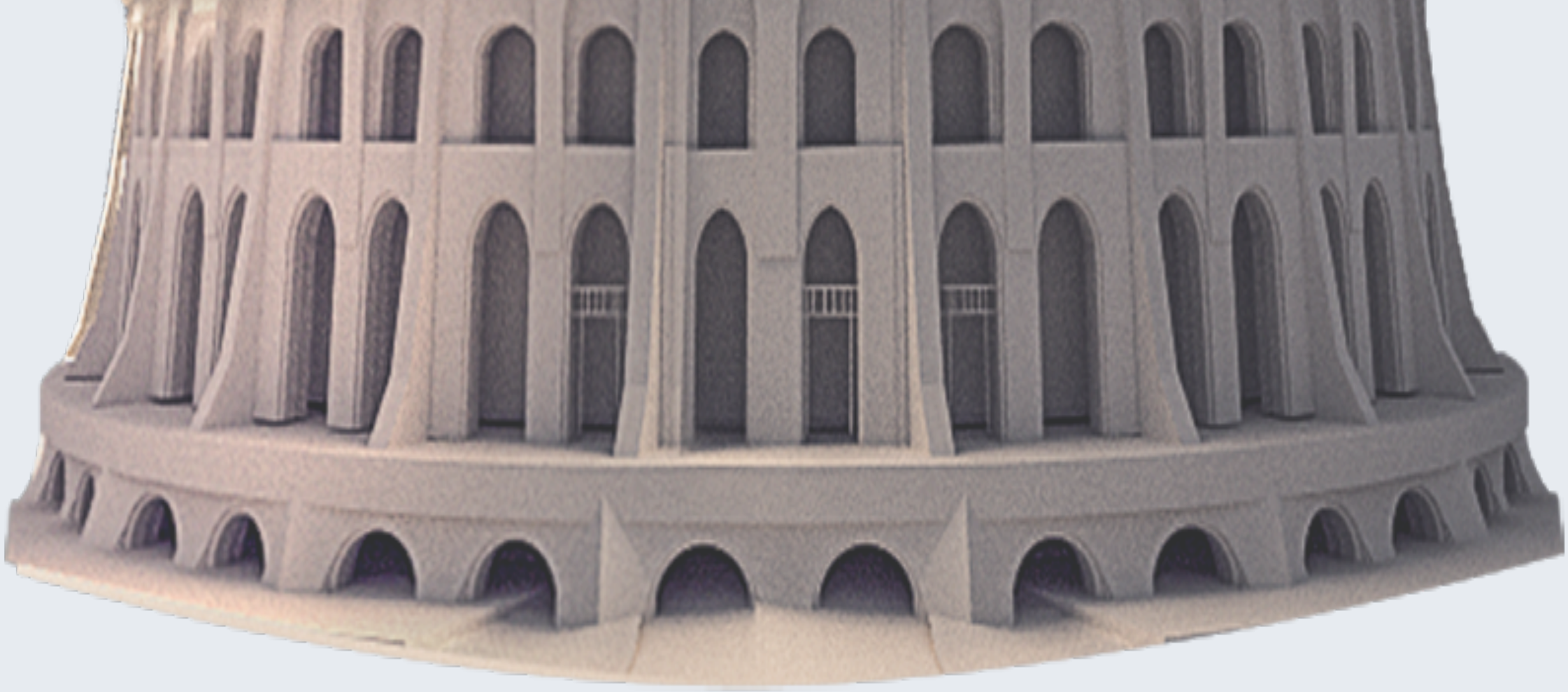


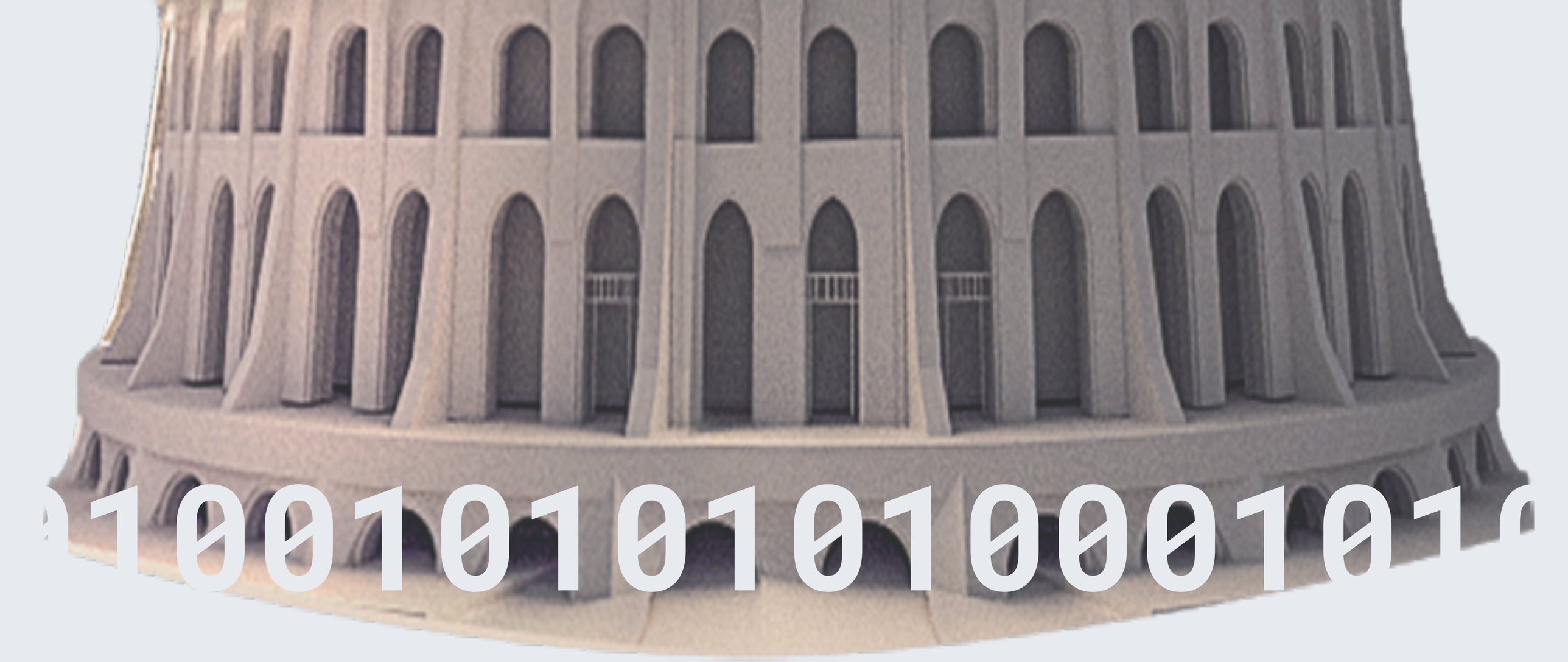


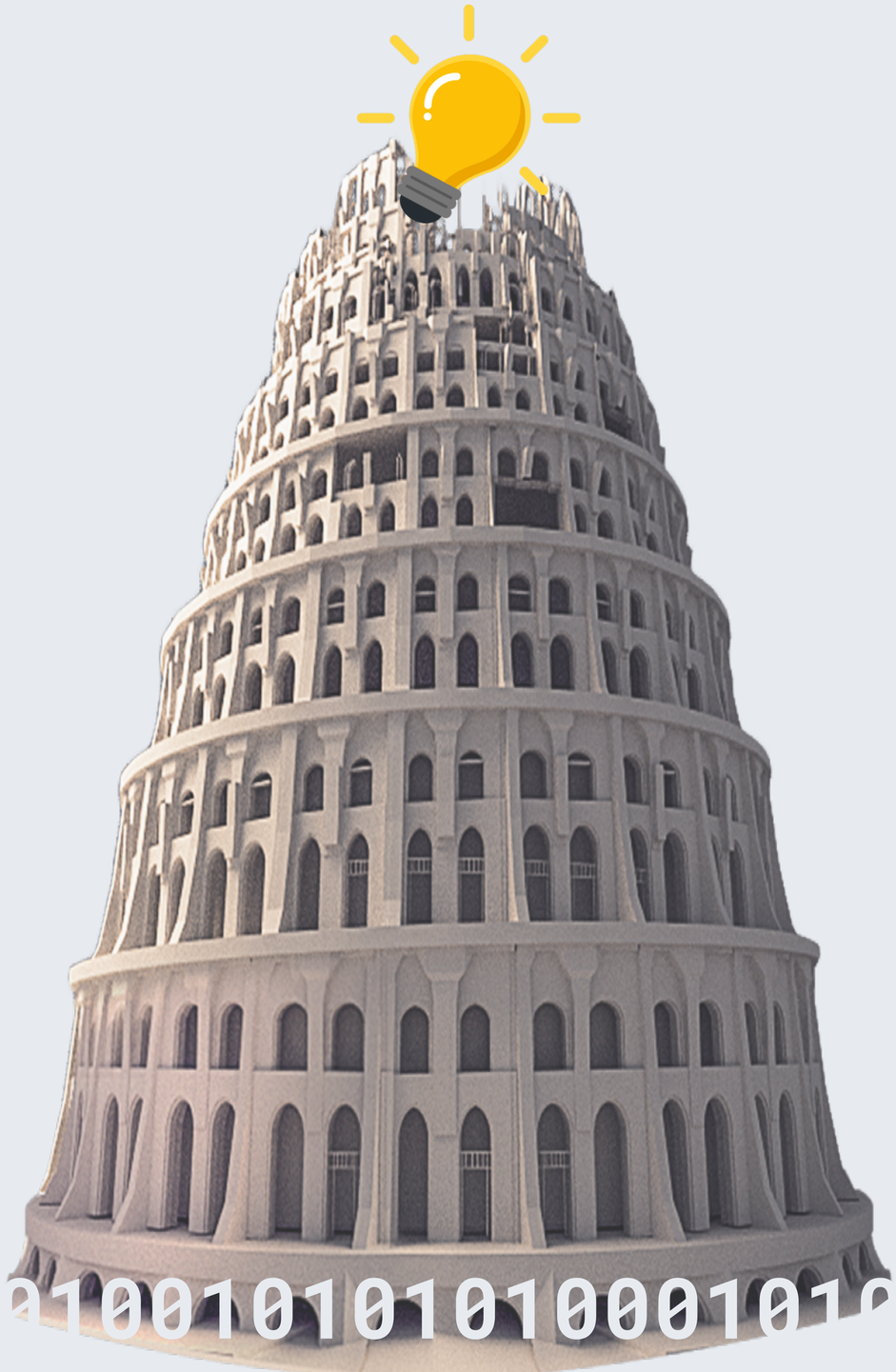


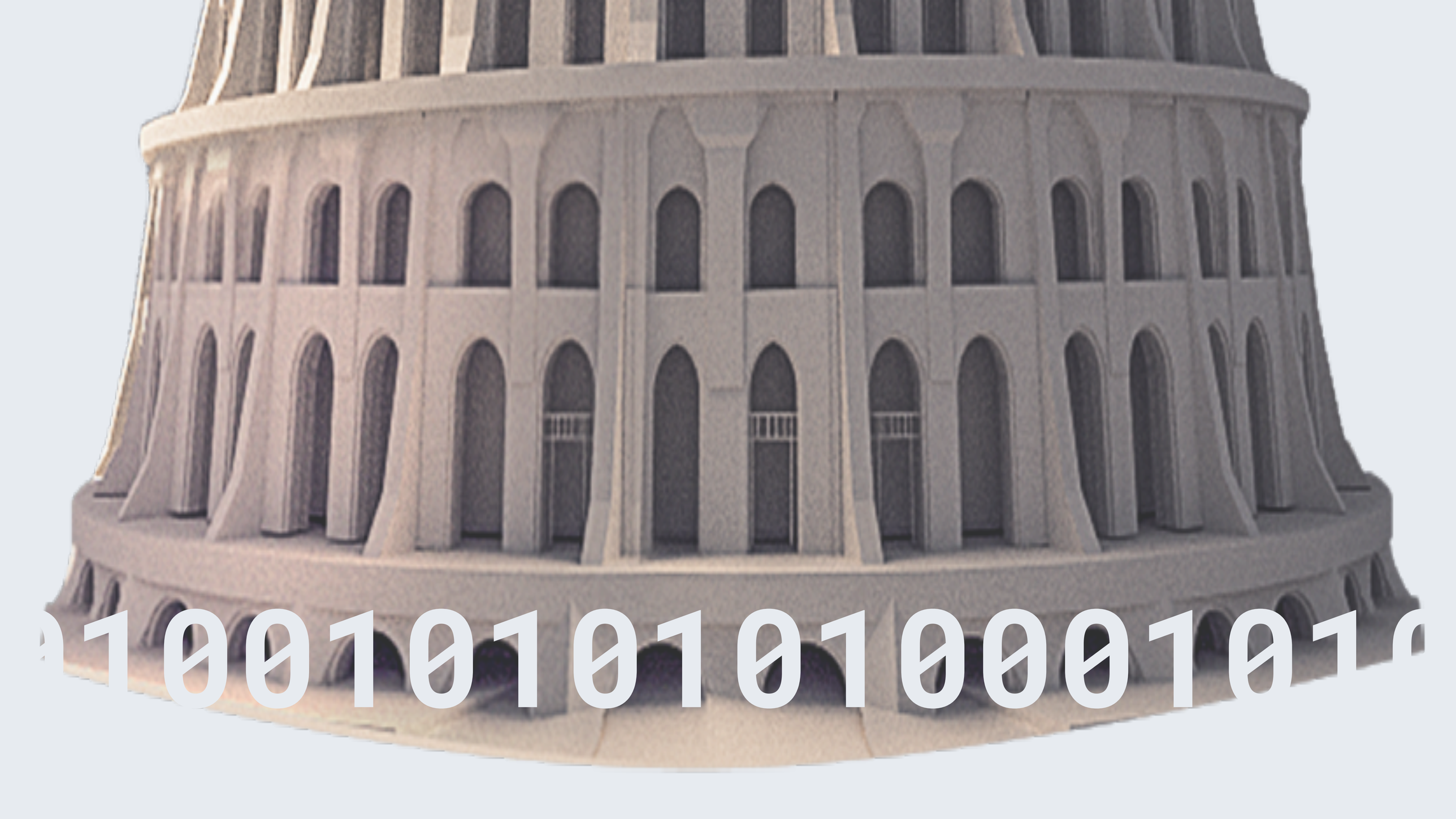







1001010101000101


01001010101000101

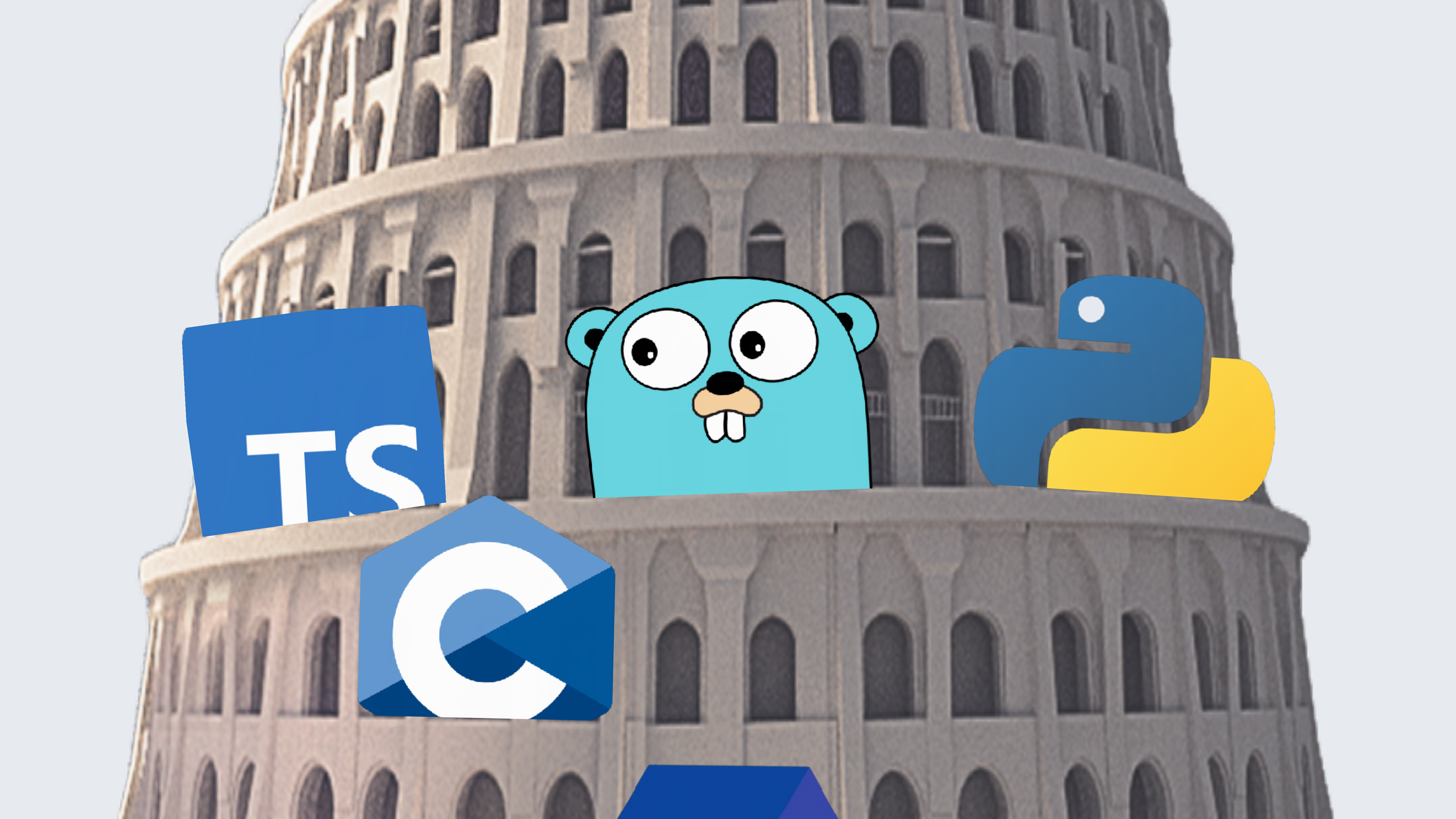

1001010101000101

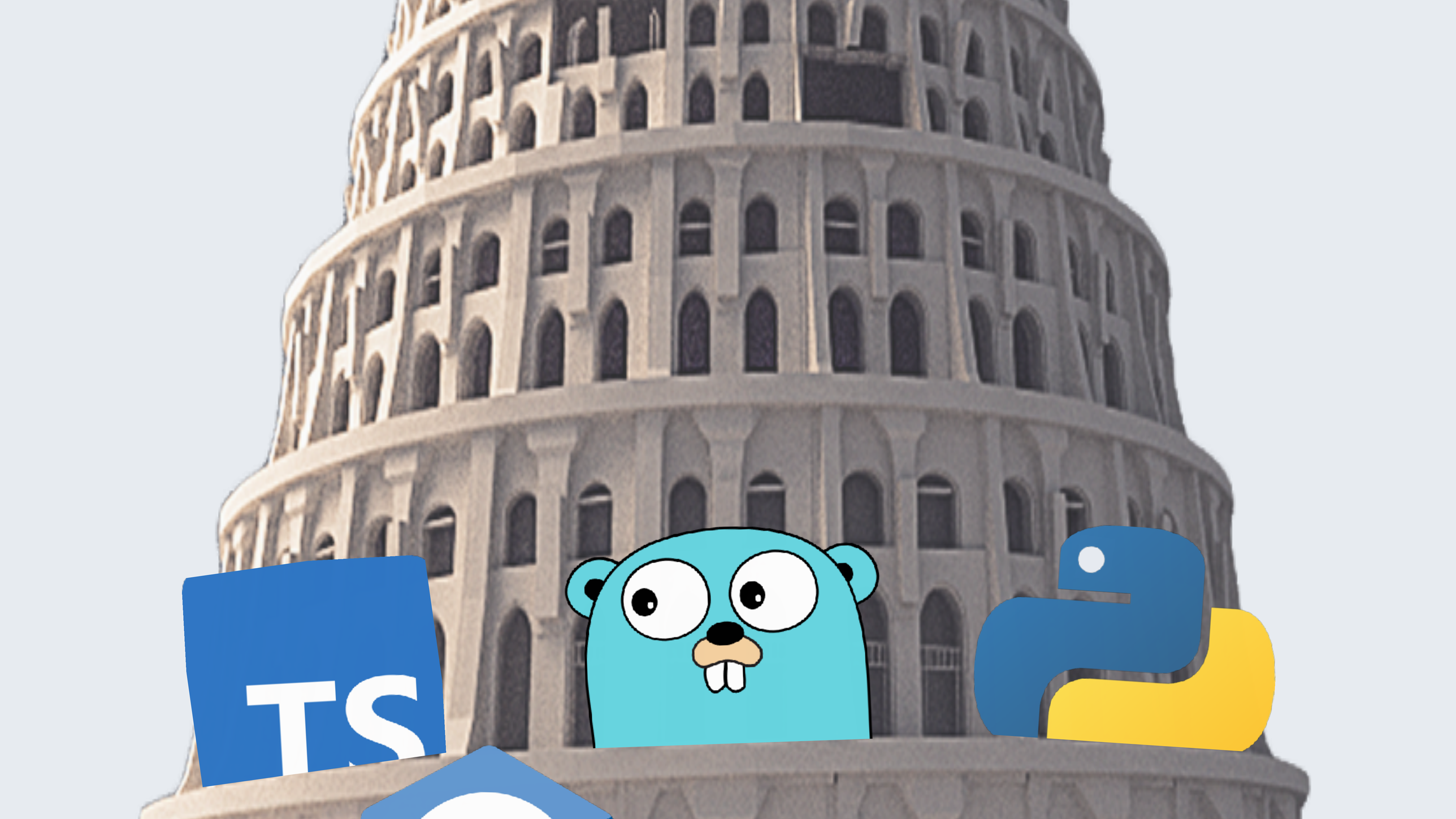

ASM
1001010101000101


ASM


ASM




TS


TS


TS



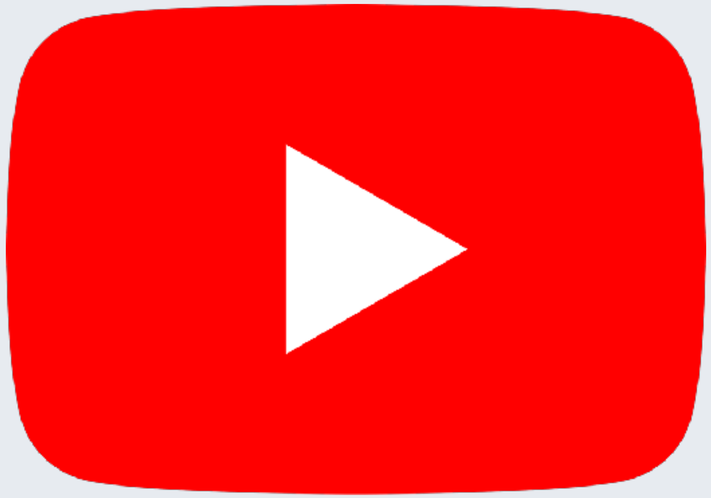



TS
ASM
01001010101000101010

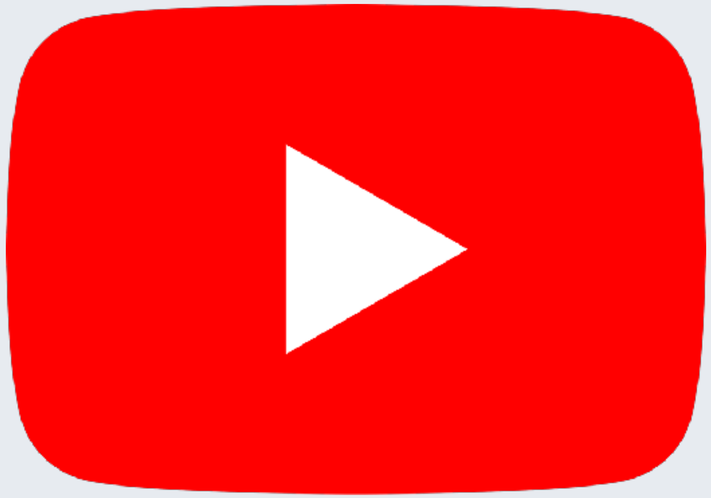

TS

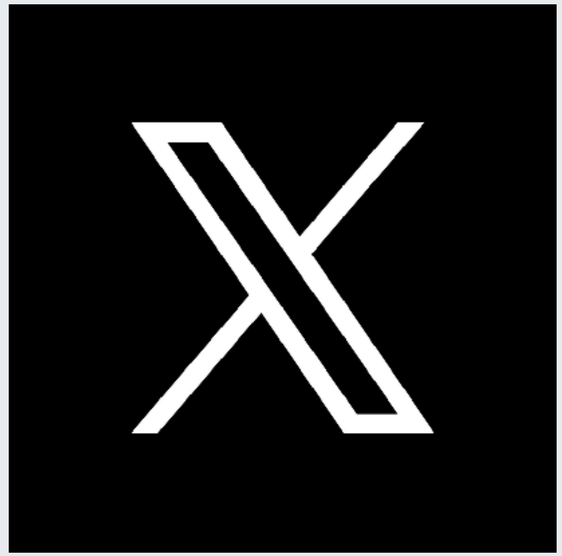


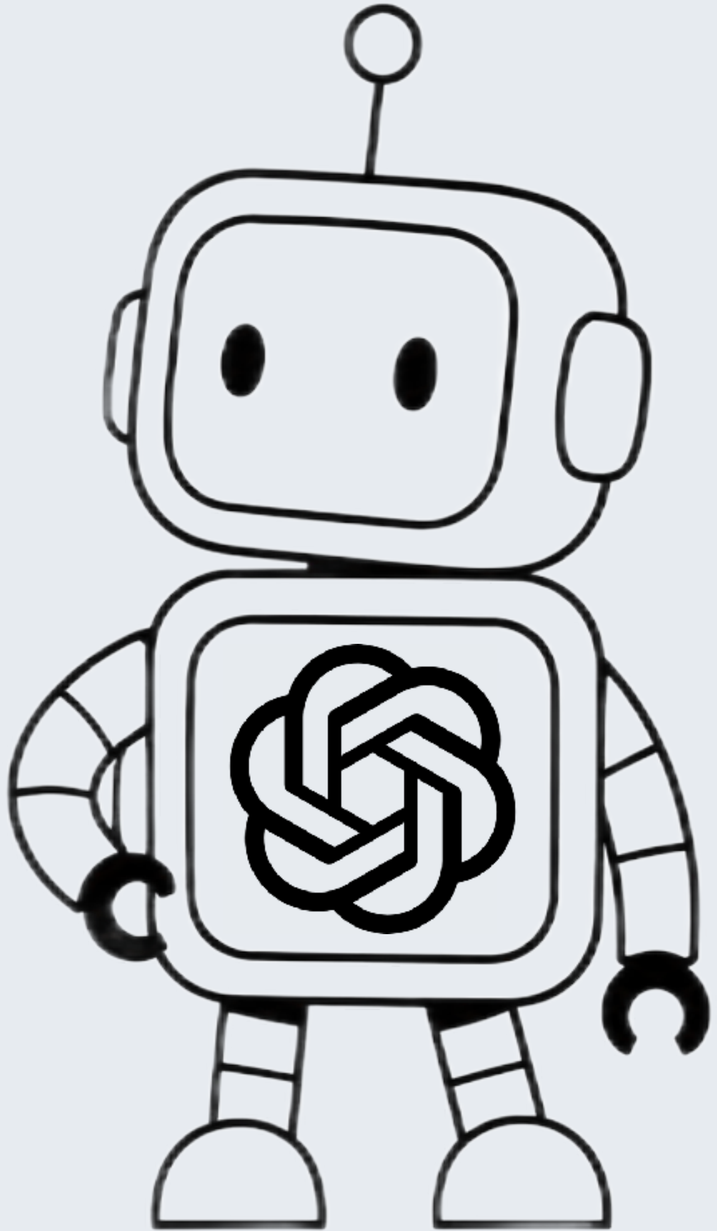


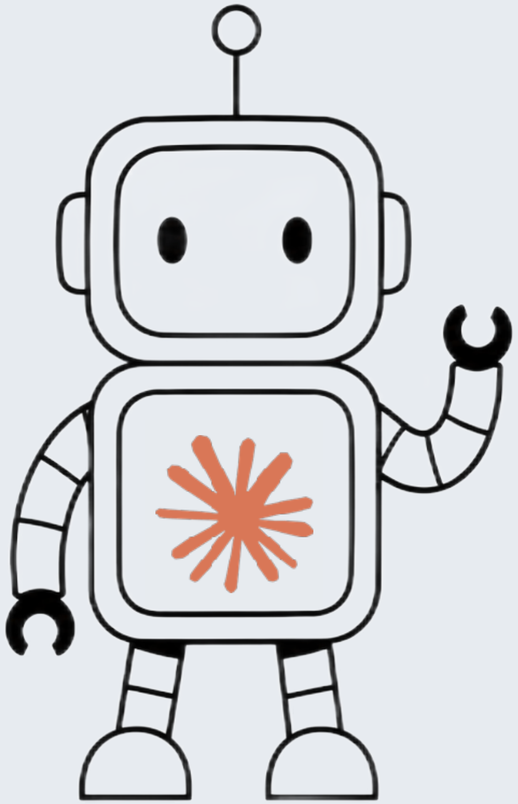


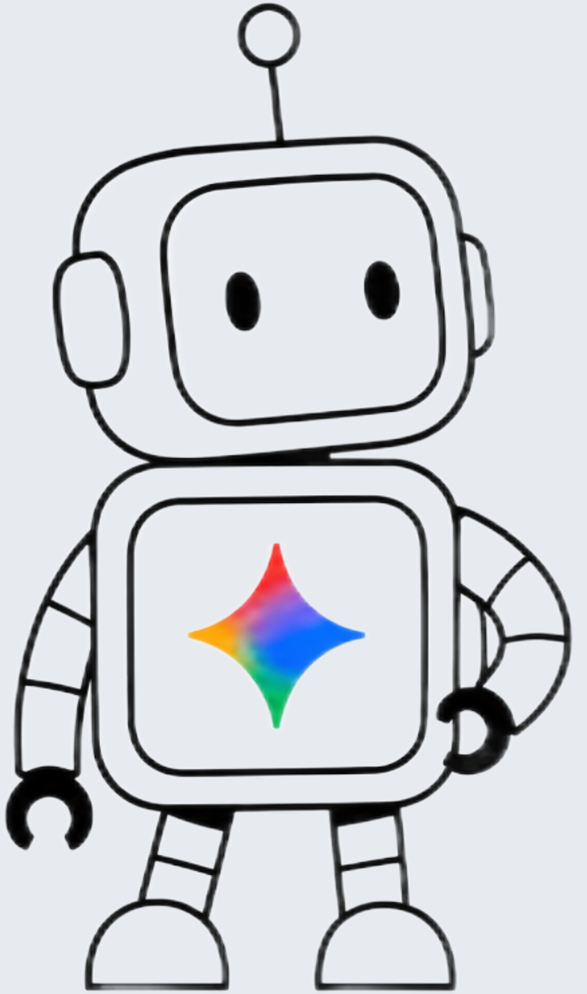






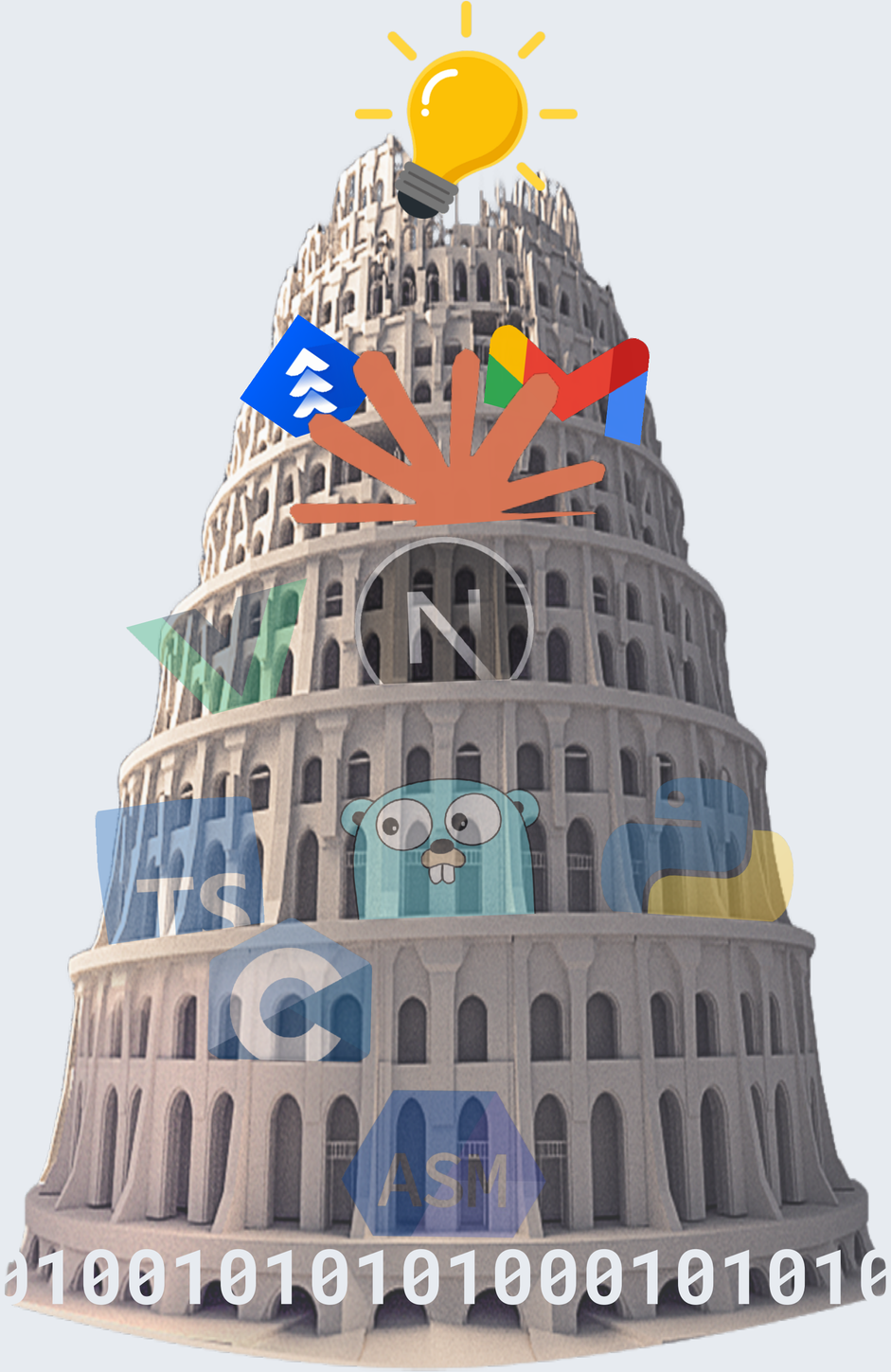


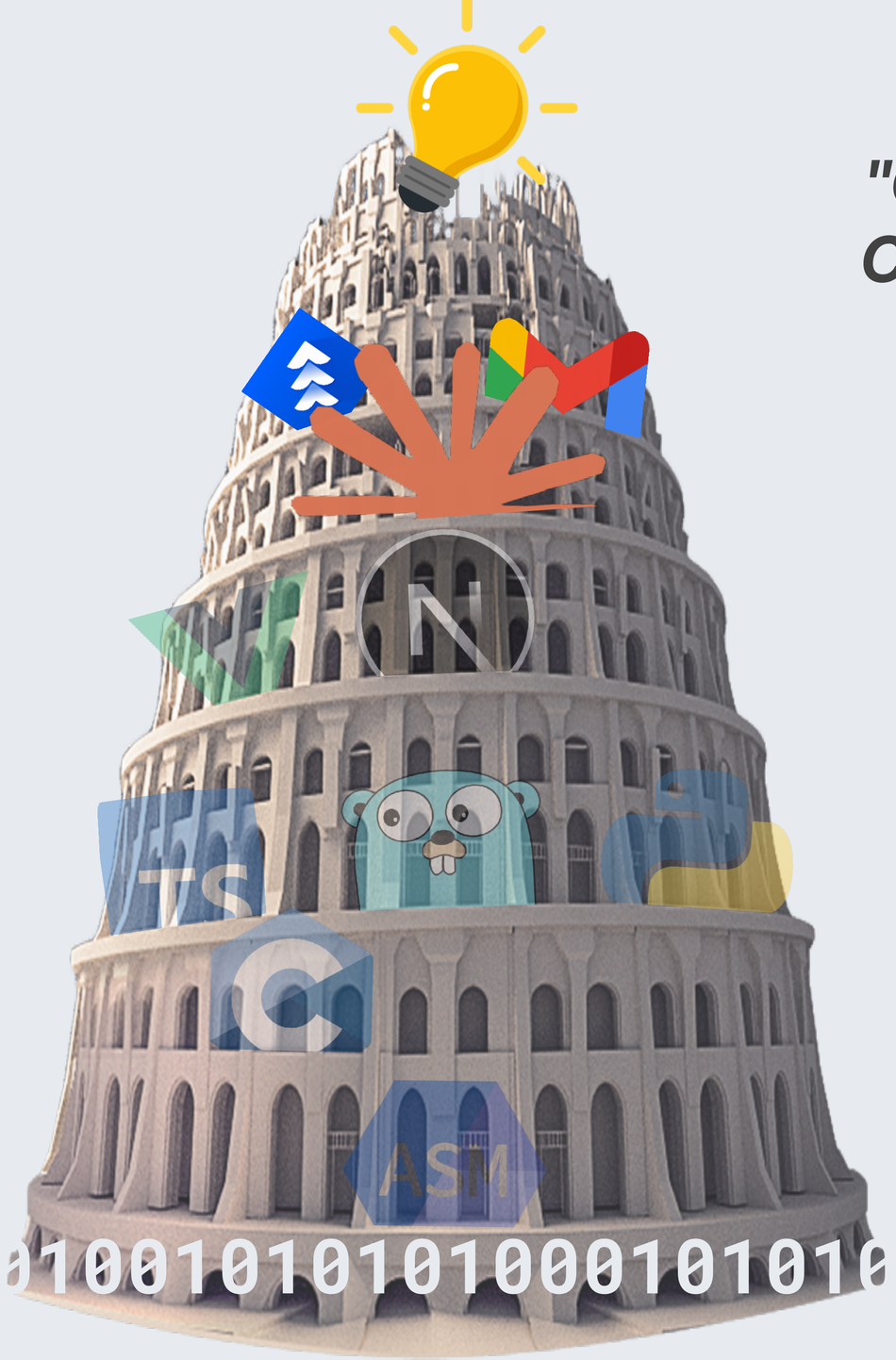





TS
ASM
01001010101000101010


TS
ASM
01001010101000101010

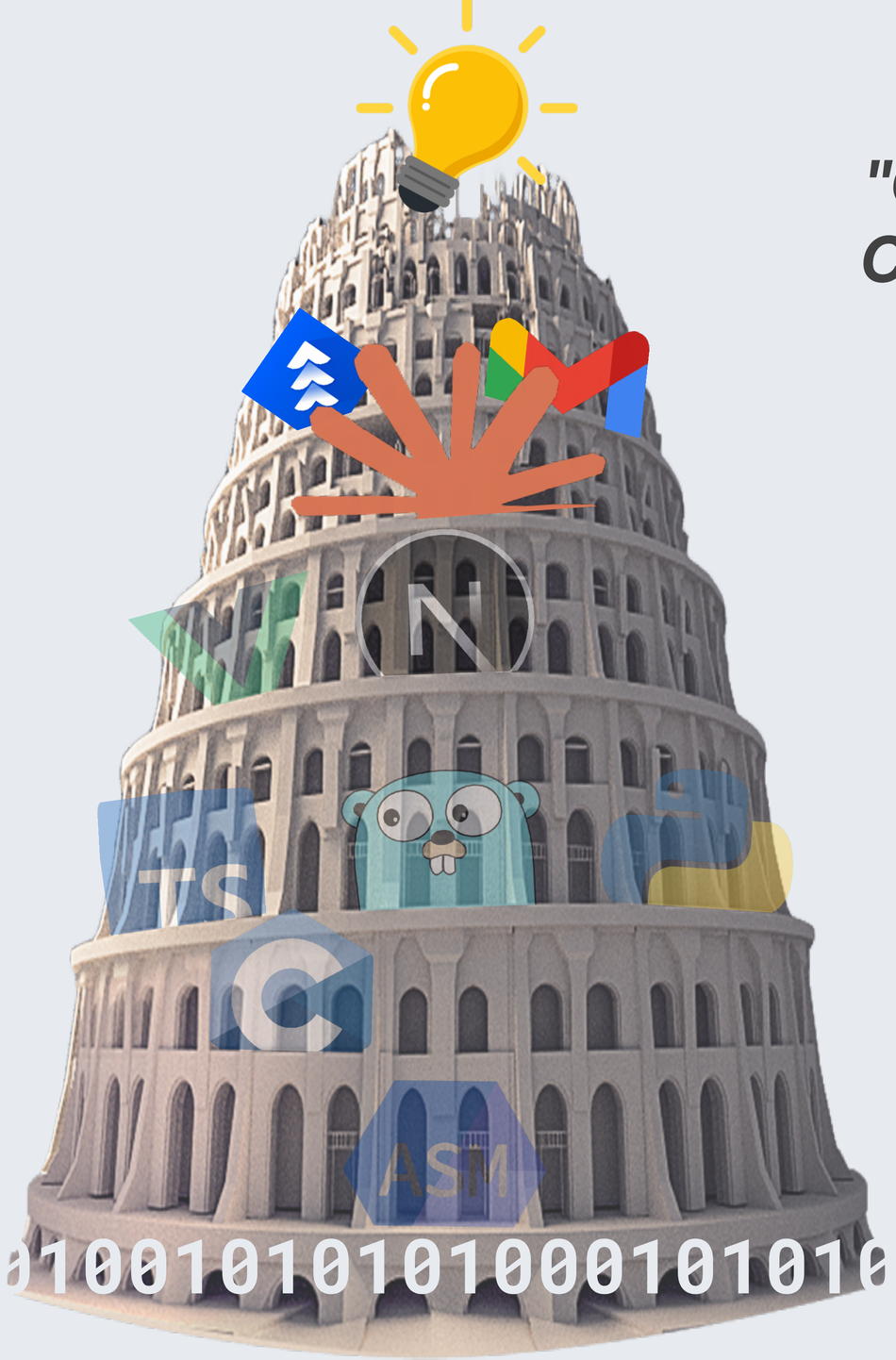

N
TS
C
ASM
01001010101000101010



"Coding is largely solved [...] because Claude can do it."

Boris Cherny,
Head of Claude Code, Anthropic





"Coding is largely solved [...] because Claude can do it."

Boris Cherny,
Head of Claude Code, Anthropic



"There's a new programming language. It's called English."

Jensen Huang,
CEO, Nvidia







"It's just fancy autocomplete."

Some guy on Twitter


TS
ASM
01001010101000101010



The spec has to live *somewhere*



The spec has to live *somewhere*

Why not in English?



The spec has to live *somewhere*

Why not in English?

Ambiguous



The spec has to live
somewhere

Why not in English?

Ambiguous

Complicated



The spec has to live
somewhere

Why not in English?

Ambiguous

Complicated

Can't automate it



The spec has to live
somewhere

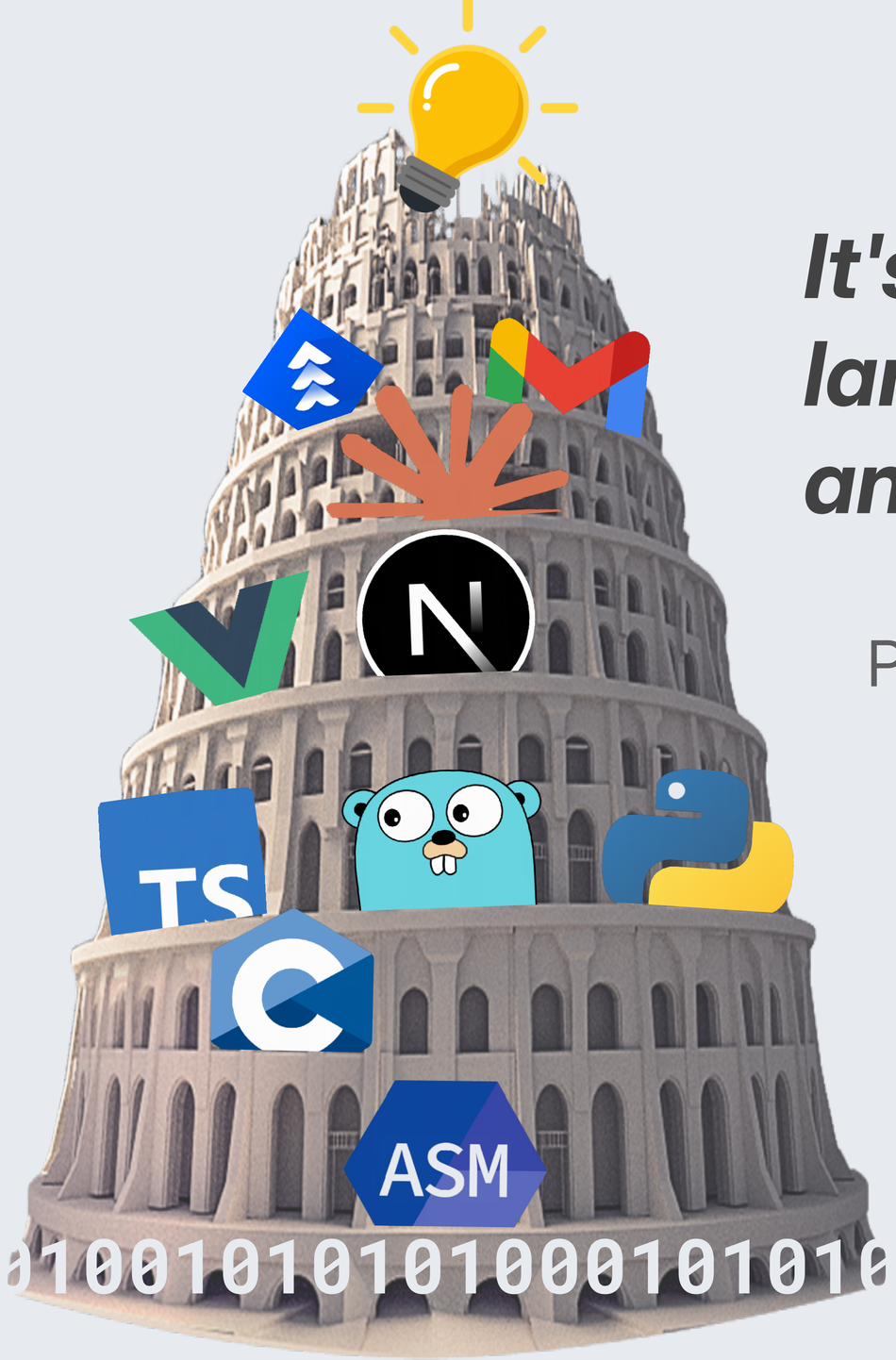
Why not in English?

Ambiguous

Complicated

Can't automate it

Not a Harness



It's like writing in a more powerful language that doesn't always work, and then losing the source code.

Paul Graham



We wouldn't be committing code.

We wouldn't be committing **code**.

We'd be committing **prompts**.



Filip Sodic

Working with compilers at [Wasp] and for fun
Teaching Haskell at UniZG

`sodic.dev`



Filip Sodic

Working with compilers at [Wasp] and for fun
Teaching Haskell at UniZG

`sodic.dev`

Haskell



Filip Sodic

Working with compilers at [Wasp] and for fun
Teaching Haskell at UniZG

sodic.dev

TypeScript

Haskell



Filip Sodic

Working with compilers at [Wasp](#) and for fun
Teaching Haskell at UniZG

sodic.dev

Wasp

TypeScript

Haskell


TS
ASM
0100201010100010101

Why care about the language's properties?



"Fill up the tank!"

Diesel or petrol?

How much? Full?

"Fill up the tank!"

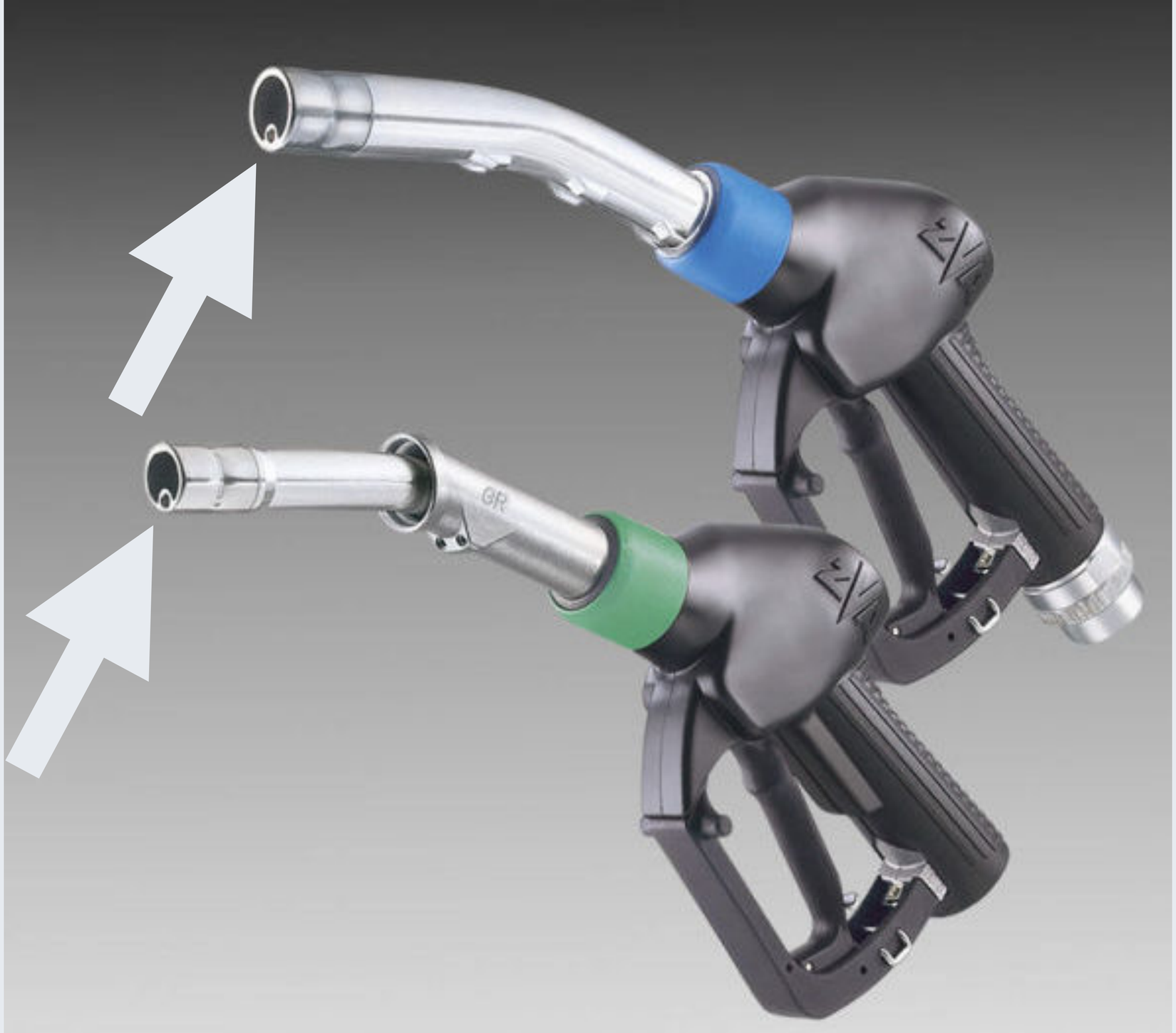
Premium or normal?

"Fill up the tank with diesel!"

Can we do better?

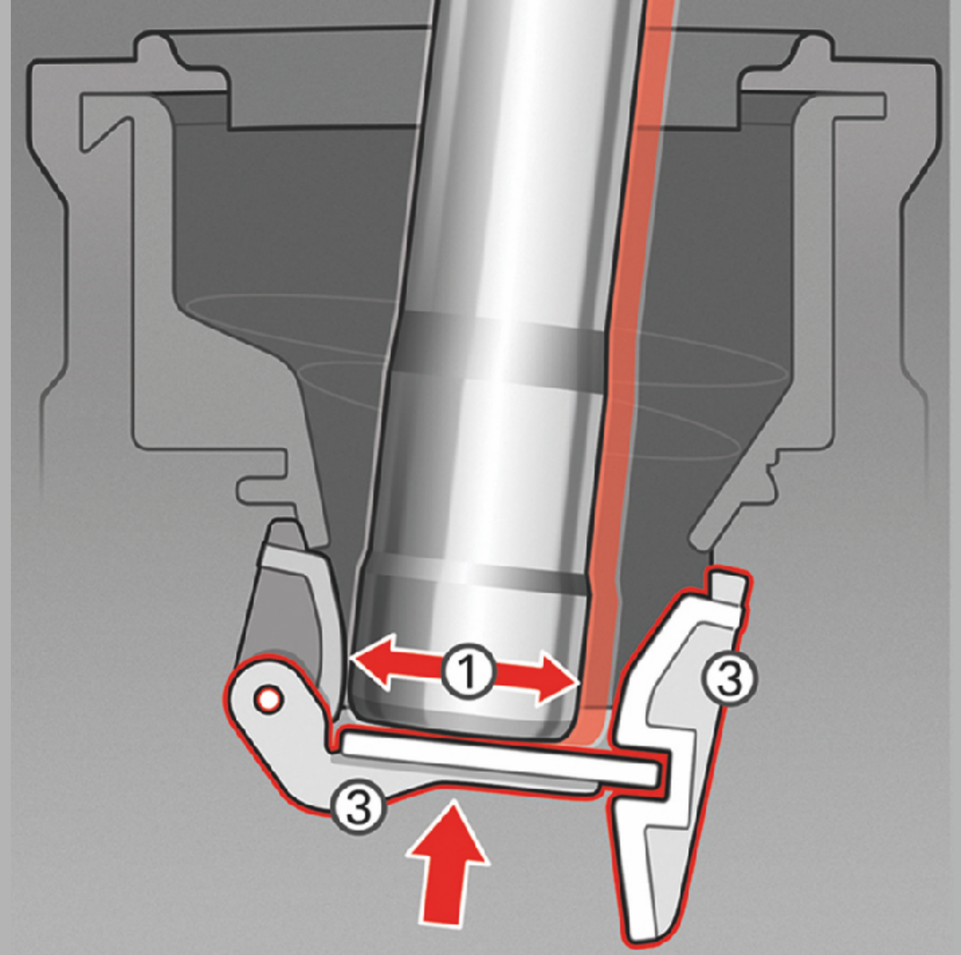


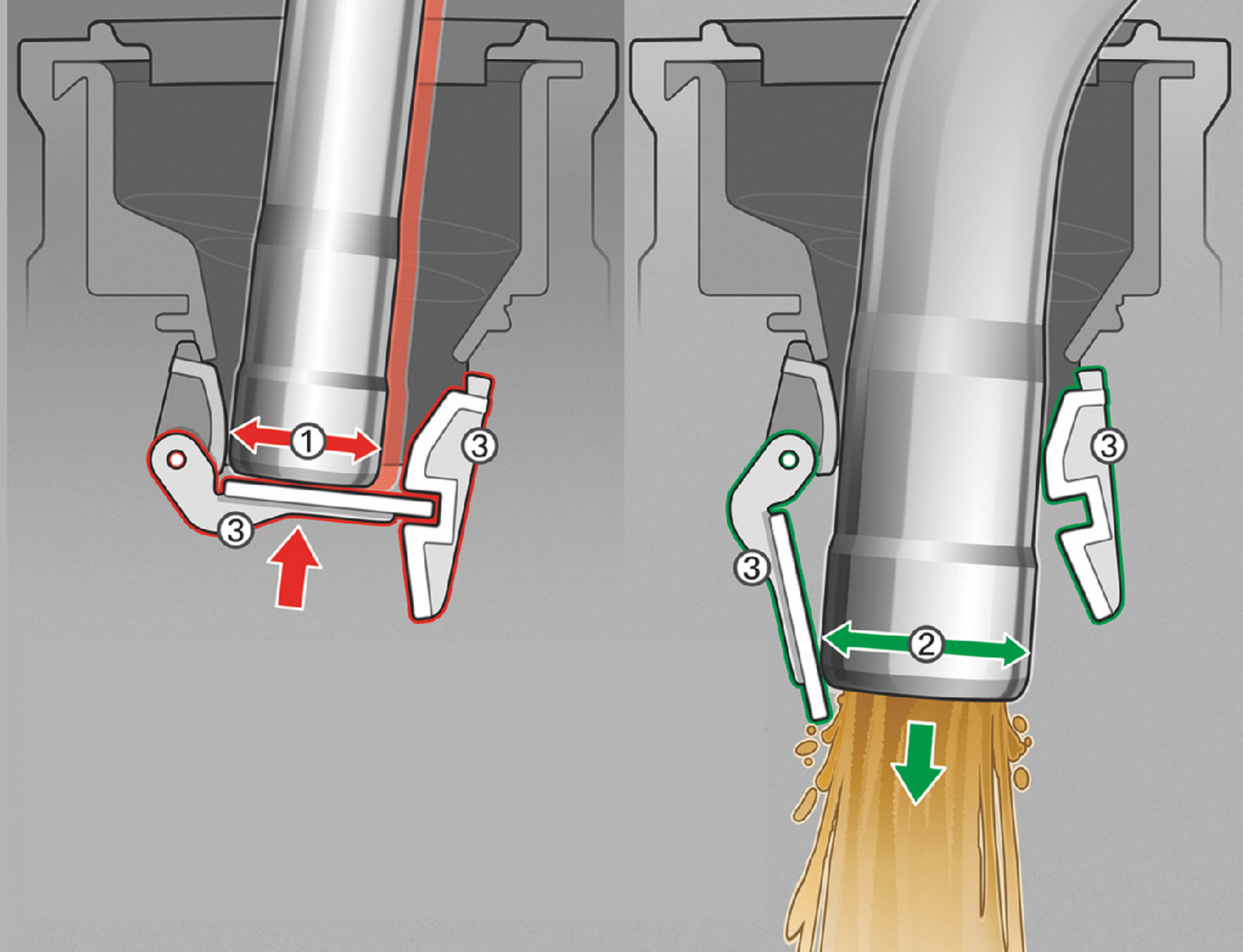

GR

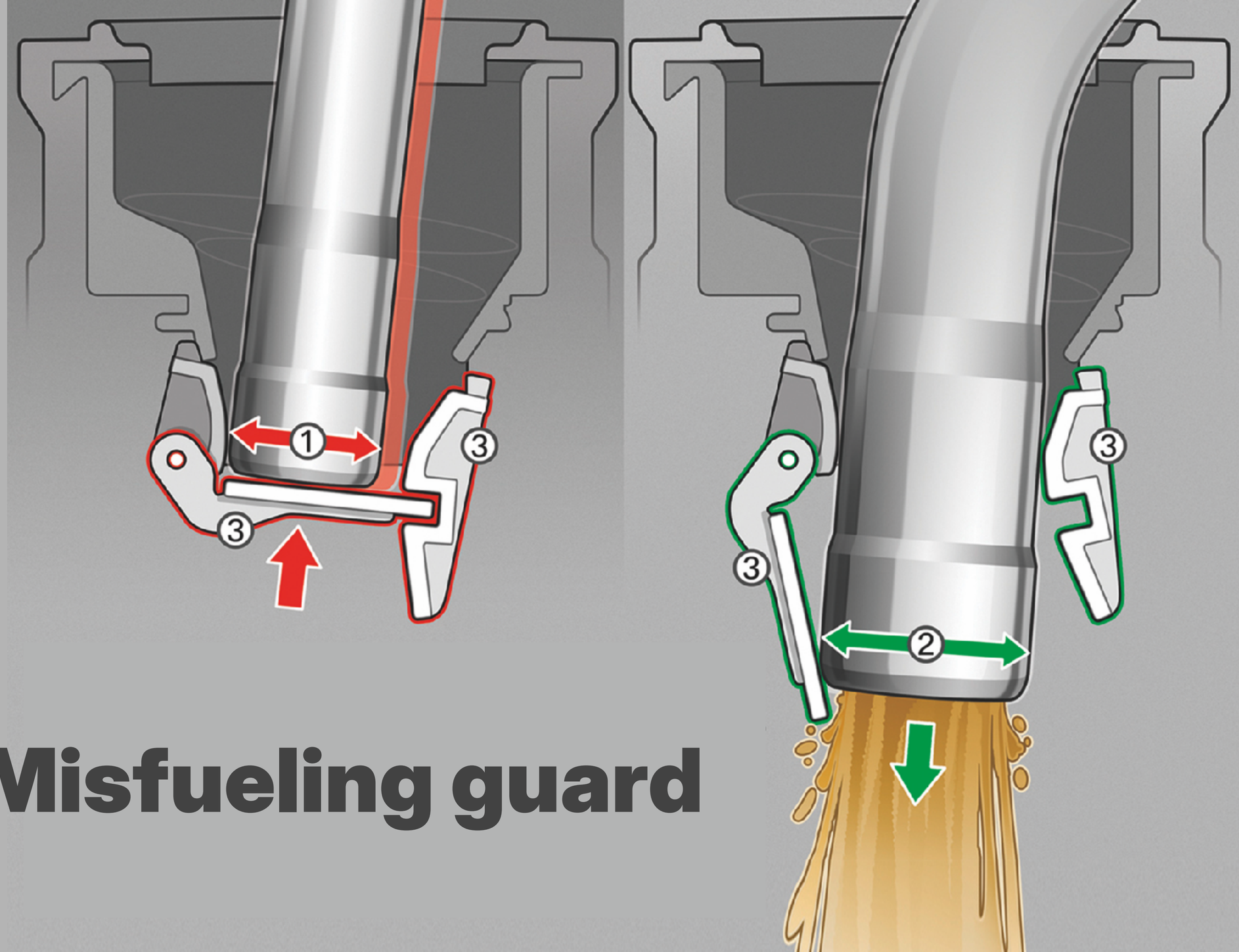

GR






1
3
3

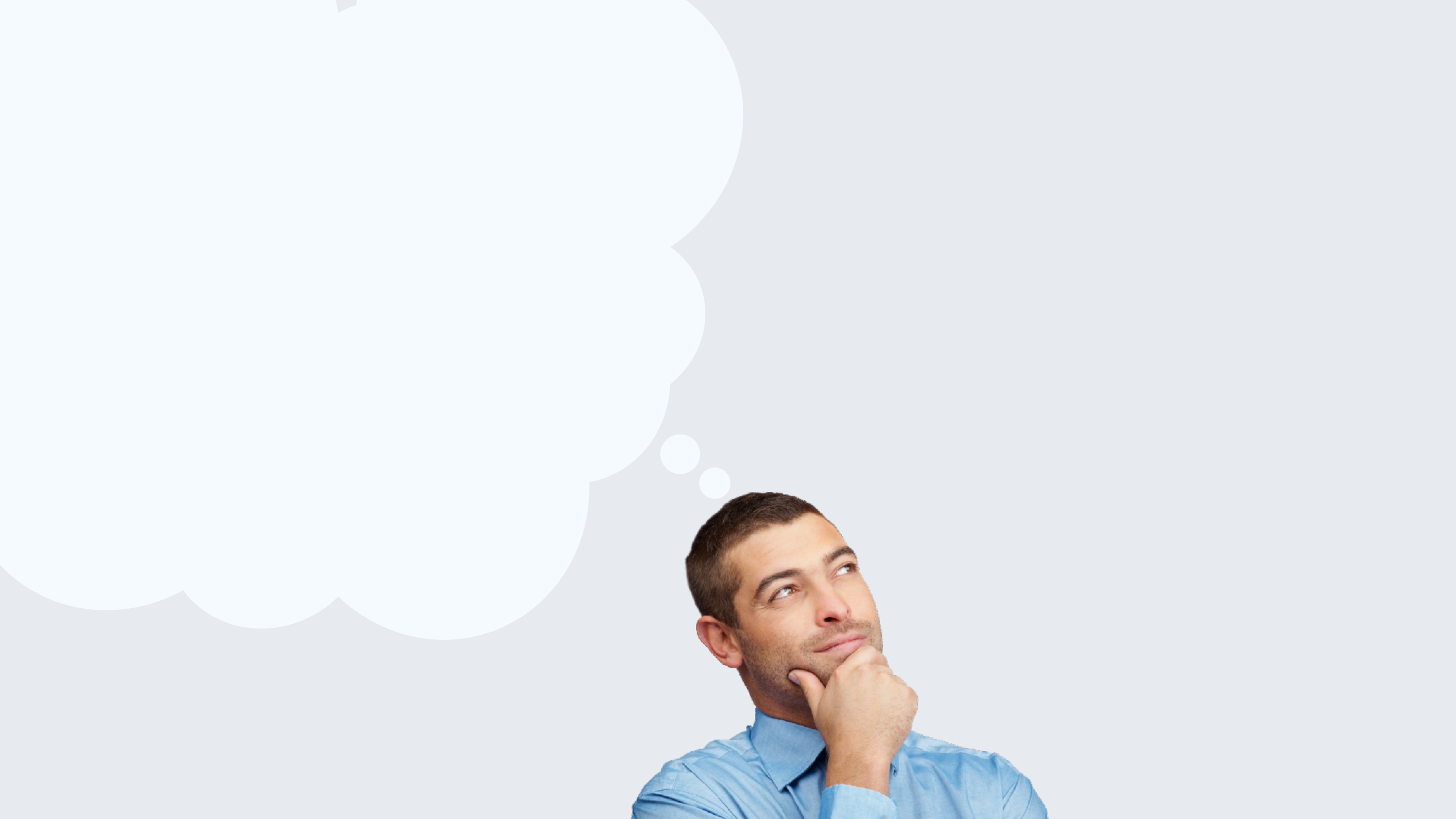

1
3
3
3
2



Misfueling guard

The context is embedded in the system





`run-wasp-app dev` should then allow the `--db-type` to be either `SQLite`, `MySql`, or `PostgreSQL`.

t allow a `--db-image` flag.

Before:

```
$ tree -L 1 .wasp/out
.wasp/out
├── Dockerfile
├── db
├── installedFullStackNpmDependencies.json
├── server
└── web-app

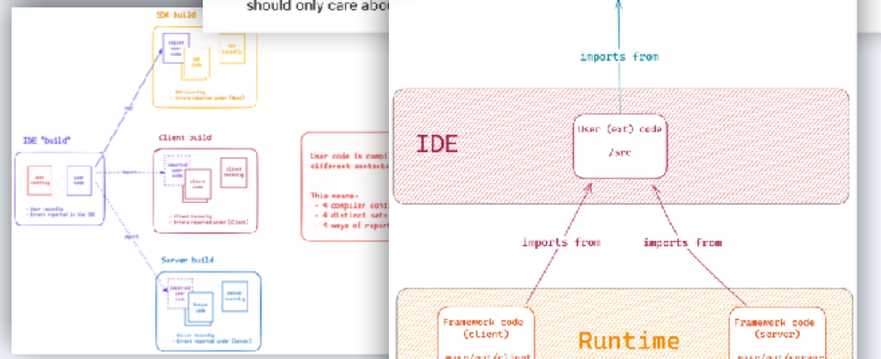
3 directories, 2 files
```

After:

```
$ tree -L 1 .wasp/out
.wasp/out
├── Dockerfile
├── db
├── installedFullStackNpmDependencies.json
├── node_modules
├── package-lock.json
├── package.json
├── server
└── web-app

4 directories, 4 files
```

queries.ts	queries.jsx	queries.tsx	queries
Start: ✅ Lsp : ✅	Start: ❌ Lsp : ❌	Start: ❌ Lsp : ❌	Start: ❌ Lsp : ❌
Start: ✅ Lsp : ❌	Start: ❌ Lsp : ❌	Start: ❌ Lsp : ❌	Start: ❌ Lsp : ❌
Start: ✅ Lsp : ❌	Start: ❌ Lsp : ✅	Start: ❌ Lsp : ✅	Start: ❌ Lsp : ❌
Start: ✅ Lsp : ❌	Start: ❌ Lsp : ❌	Start: ❌ Lsp : ✅	Start: ❌ Lsp : ❌
Start: ✅ Lsp : ✅	Start: ❌ Lsp : ✅	Start: ❌ Lsp : ✅	Start: ❌ Lsp : ❌



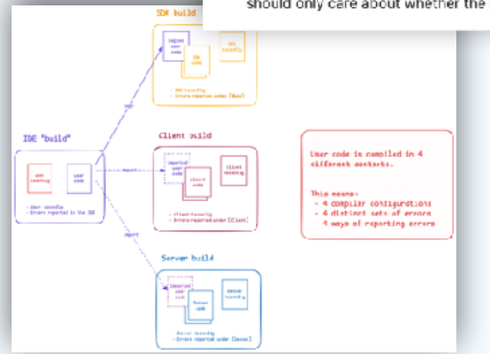


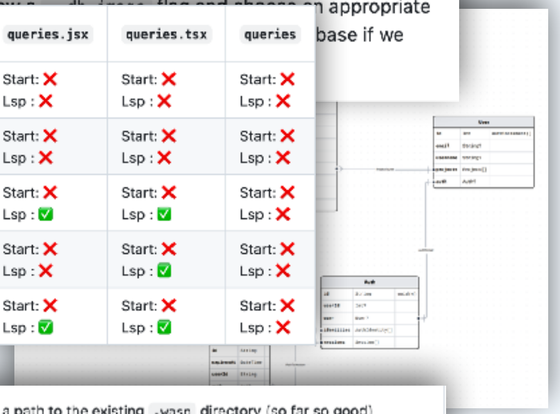
`run-wasp-app dev` should then allow the `--db-type` to be either `SQLite`, `MySql`, or `PostgreSQL`.

- If `--db-type` is `SQLite`, then we don't allow a `--db-image` flag.
- If `--db-type` is `MySQL` or `PostgreSQL`, then we allow a `--db-image` flag and choose an appropriate defau… base if we decid…

Real file name (right) Imported as (down)	`queries.js`	`queries.ts`	`queries.jsx`	`queries.tsx`	`queries`
`queries.js`	Start: ✅ Lsp : ✅	Start: ✅ Lsp : ✅	Start: ❌ Lsp : ❌	Start: ❌ Lsp : ❌	Start: ❌ Lsp : ❌
`queries.ts`	Start: ✅ Lsp : ❌	Start: ✅ Lsp : ❌	Start: ❌ Lsp : ❌	Start: ❌ Lsp : ❌	Start: ❌ Lsp : ❌
`queries.jsx`	Start: ✅ Lsp : ❌	Start: ✅ Lsp : ❌	Start: ❌ Lsp : ✅	Start: ❌ Lsp : ✅	Start: ❌ Lsp : ❌
`queries.tsx`	Start: ✅ Lsp : ❌	Start: ✅ Lsp : ❌	Start: ❌ Lsp : ❌	Start: ❌ Lsp : ✅	Start: ❌ Lsp : ❌
`queries`	Start: ✅ Lsp : ✅	Start: ✅ Lsp : ✅	Start: ❌ Lsp : ✅	Start: ❌ Lsp : ✅	Start: ❌ Lsp : ❌

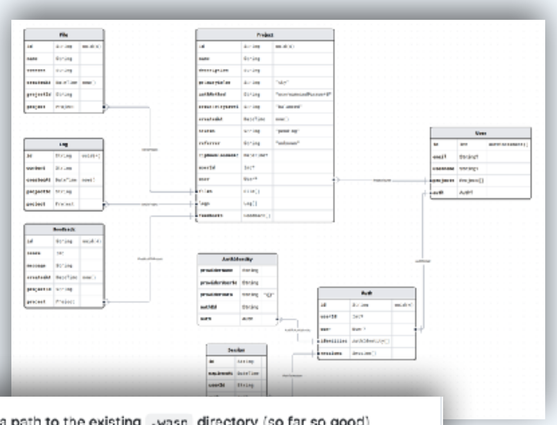
- If a `.wasp` directory does exist, it returns a path to the existing `.wasp` directory (so far so good)
- If the `.wasp` directory doesn't exist, it returns a path to the non-existing `.wasp` directory (this is the first weird part)
- If a `.wasp` file exists, it returns an error saying that `.wasp` is a file (but the function's name implies it should only care about whether the `.wasp` directory exists and report on that).



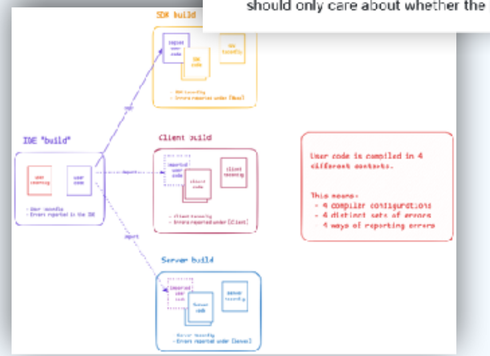


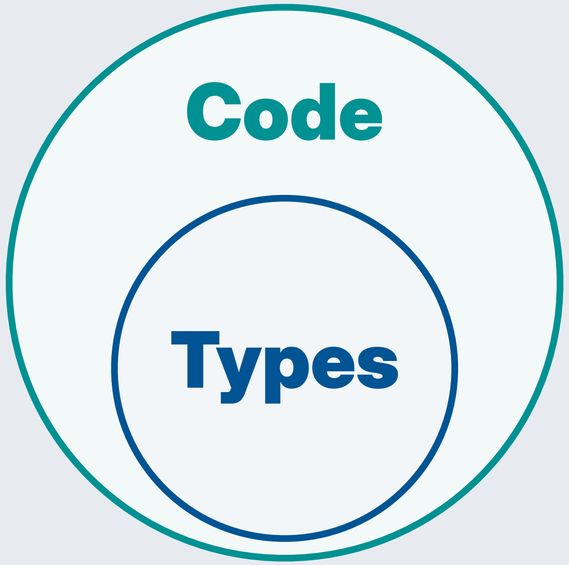


Types

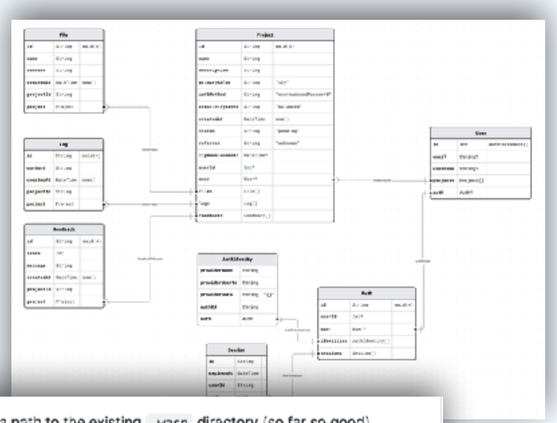


- If a `.wasp` directory does exist, it returns a path to the existing `.wasp` directory (so far so good)
- If the `.wasp` directory doesn't exist, it returns a path to the non-existing `.wasp` directory (this is the first weird part)
- If a `.wasp` file exists, it returns an error saying that `.wasp` is a file (but the function's name implies it should only care about whether the `.wasp` directory exists and report on that).

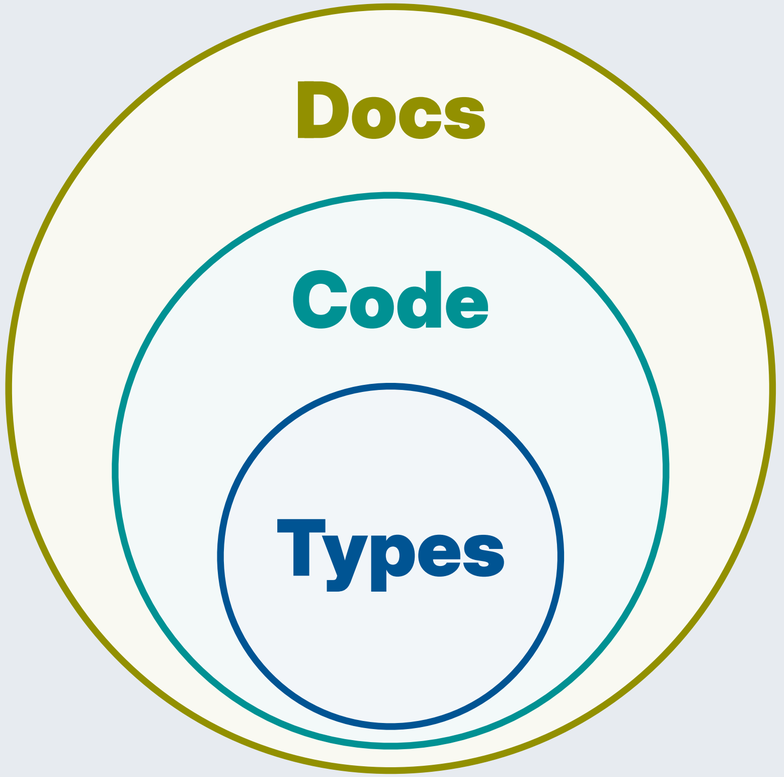




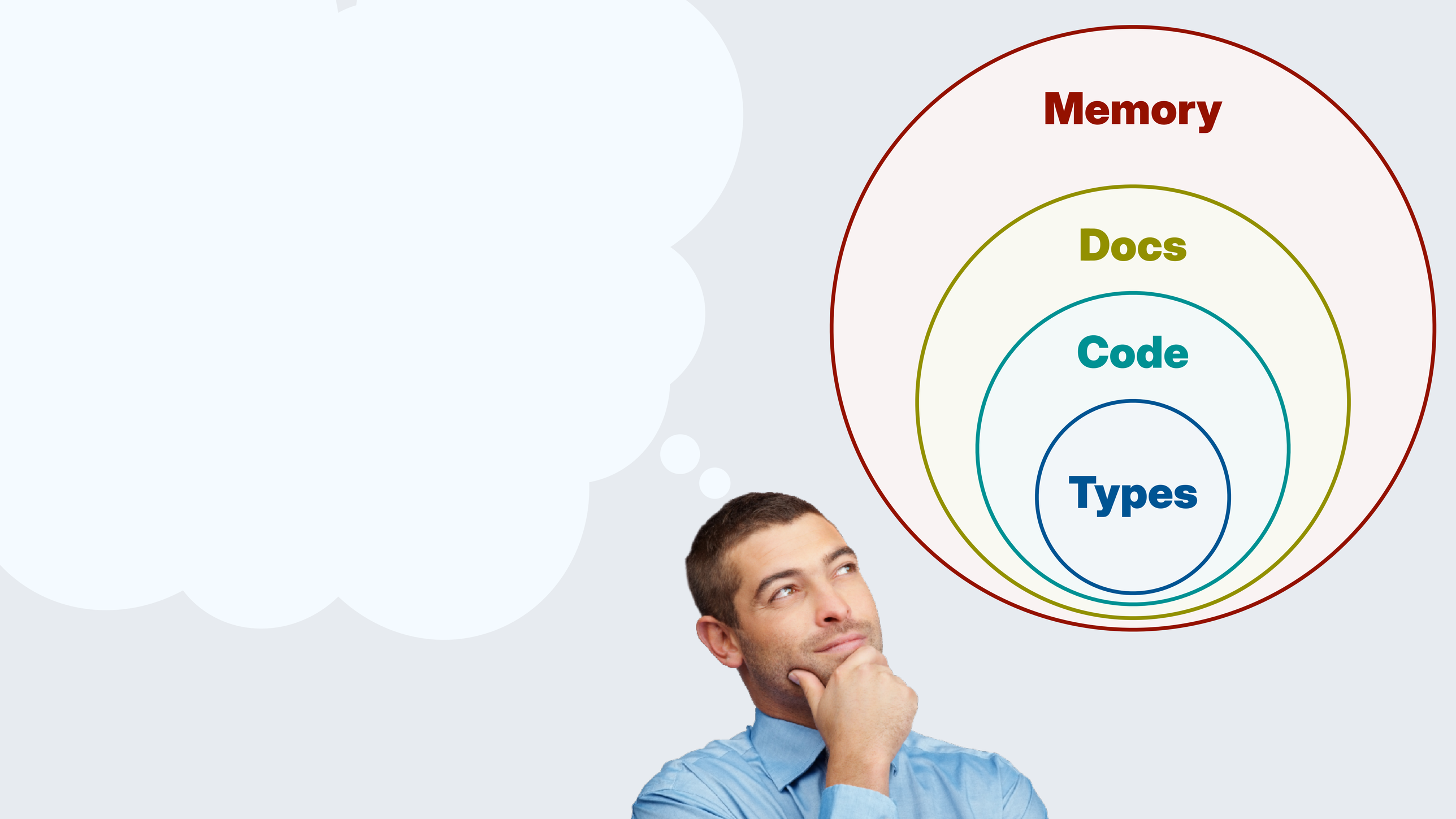


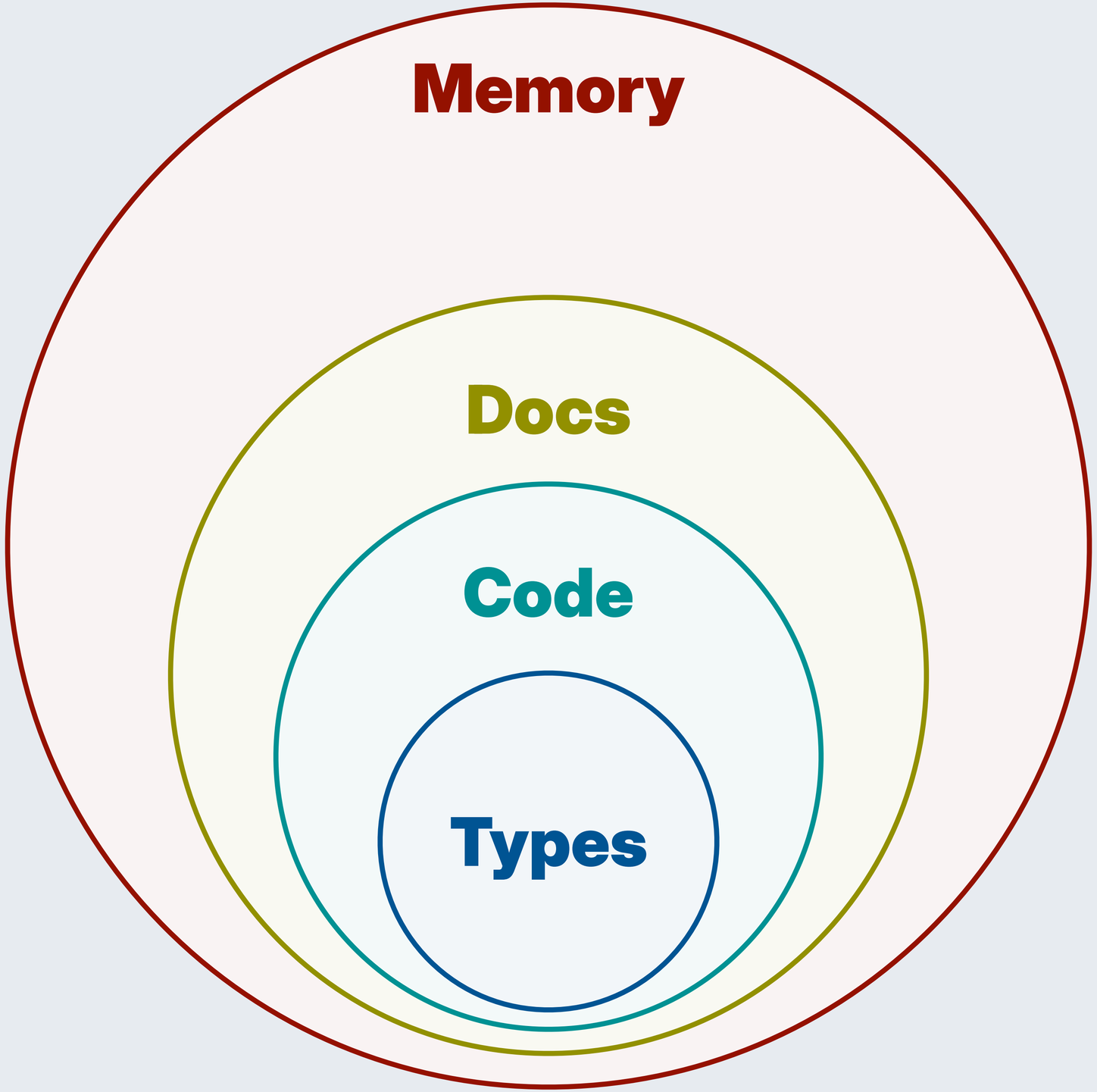


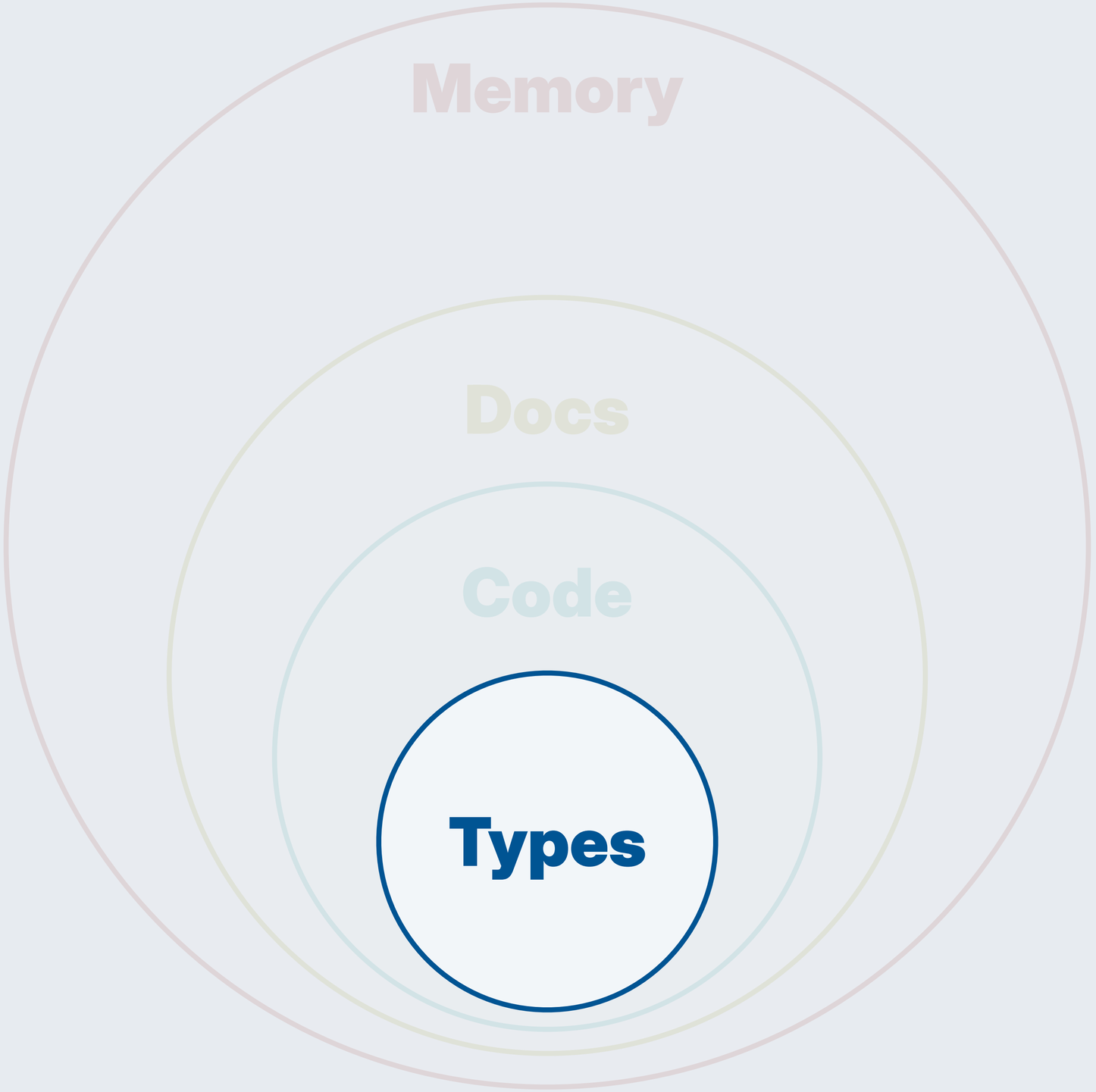
- If a `.wasp` directory does exist, it returns a path to the existing `.wasp` directory (so far so good)
- If the `.wasp` directory doesn't exist, it returns a path to the non-existing `.wasp` directory (this is the first weird part)
- If a `.wasp` file exists, it returns an error saying that `.wasp` is a file (but the function's name implies it should only care about whether the `.wasp` directory exists and report on that).

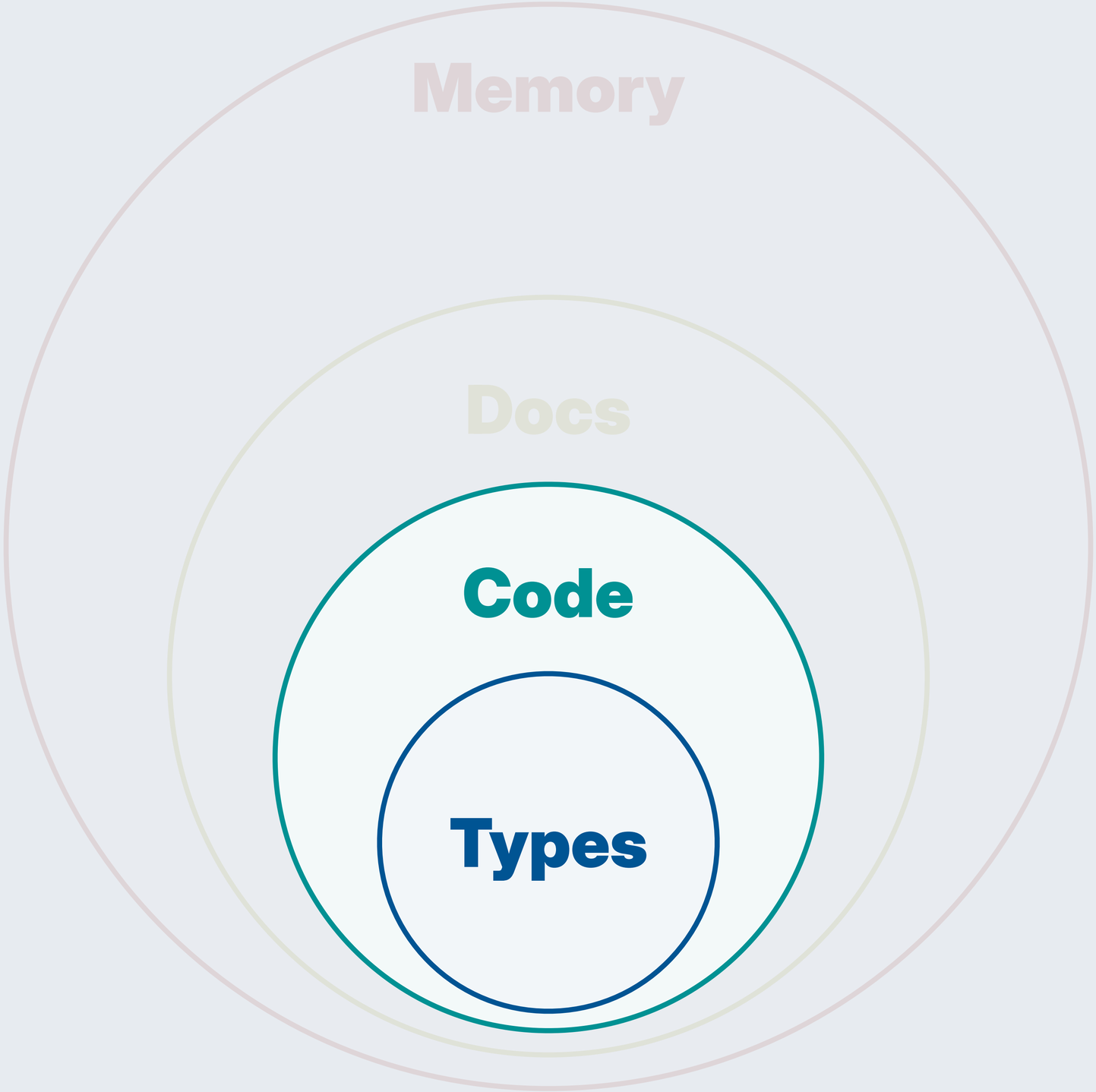


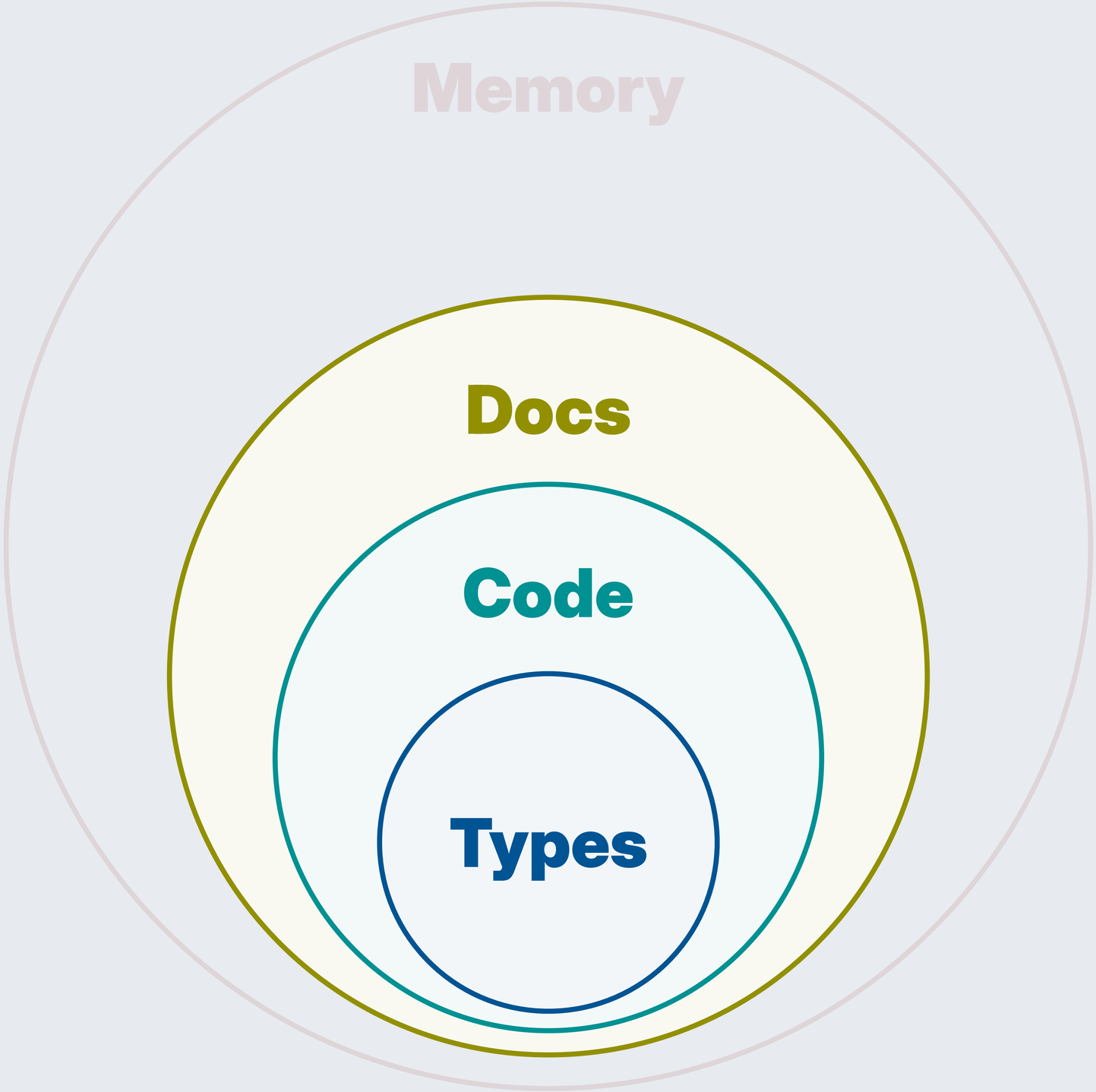


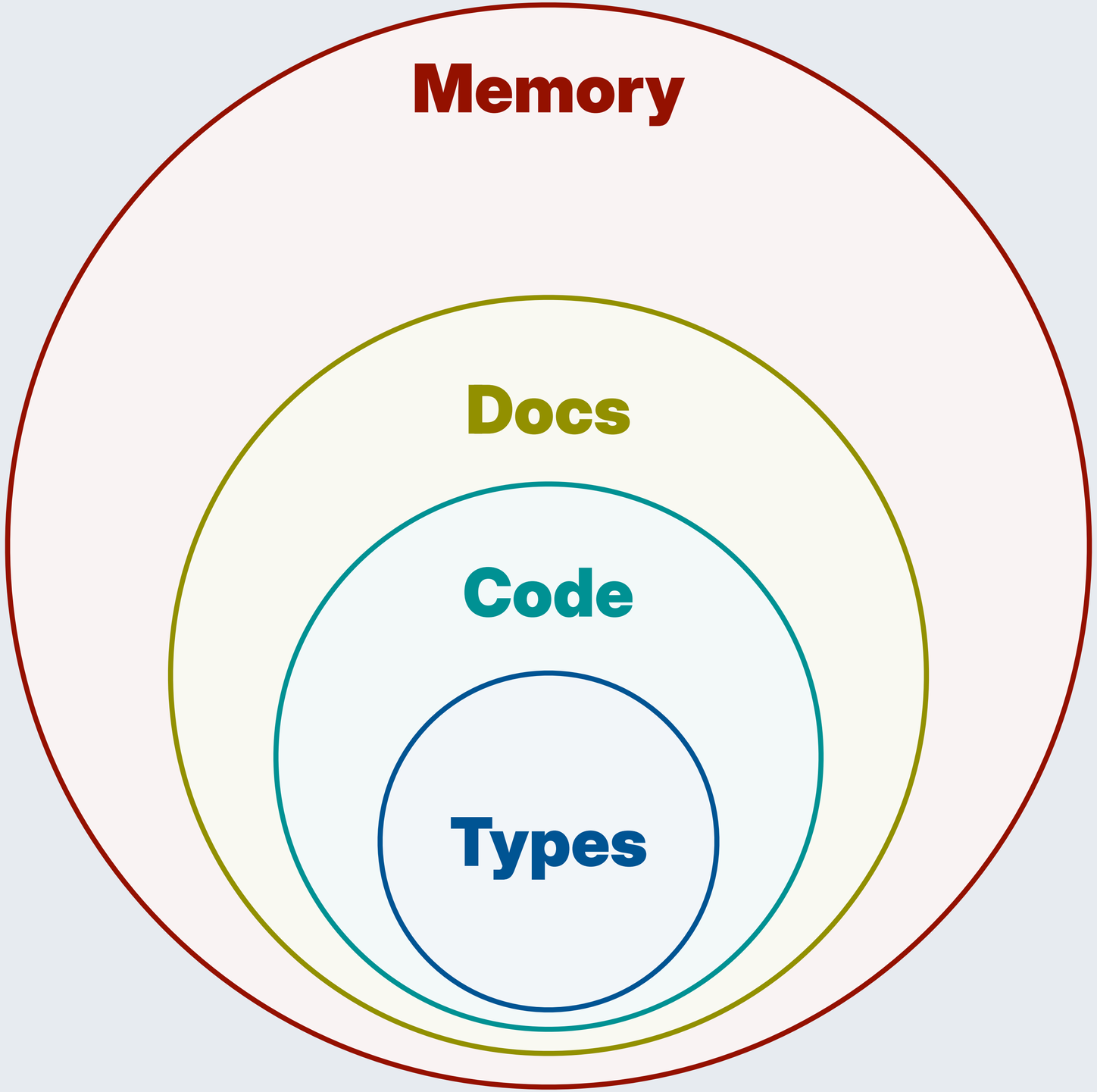

Memory
Docs
Code
Types

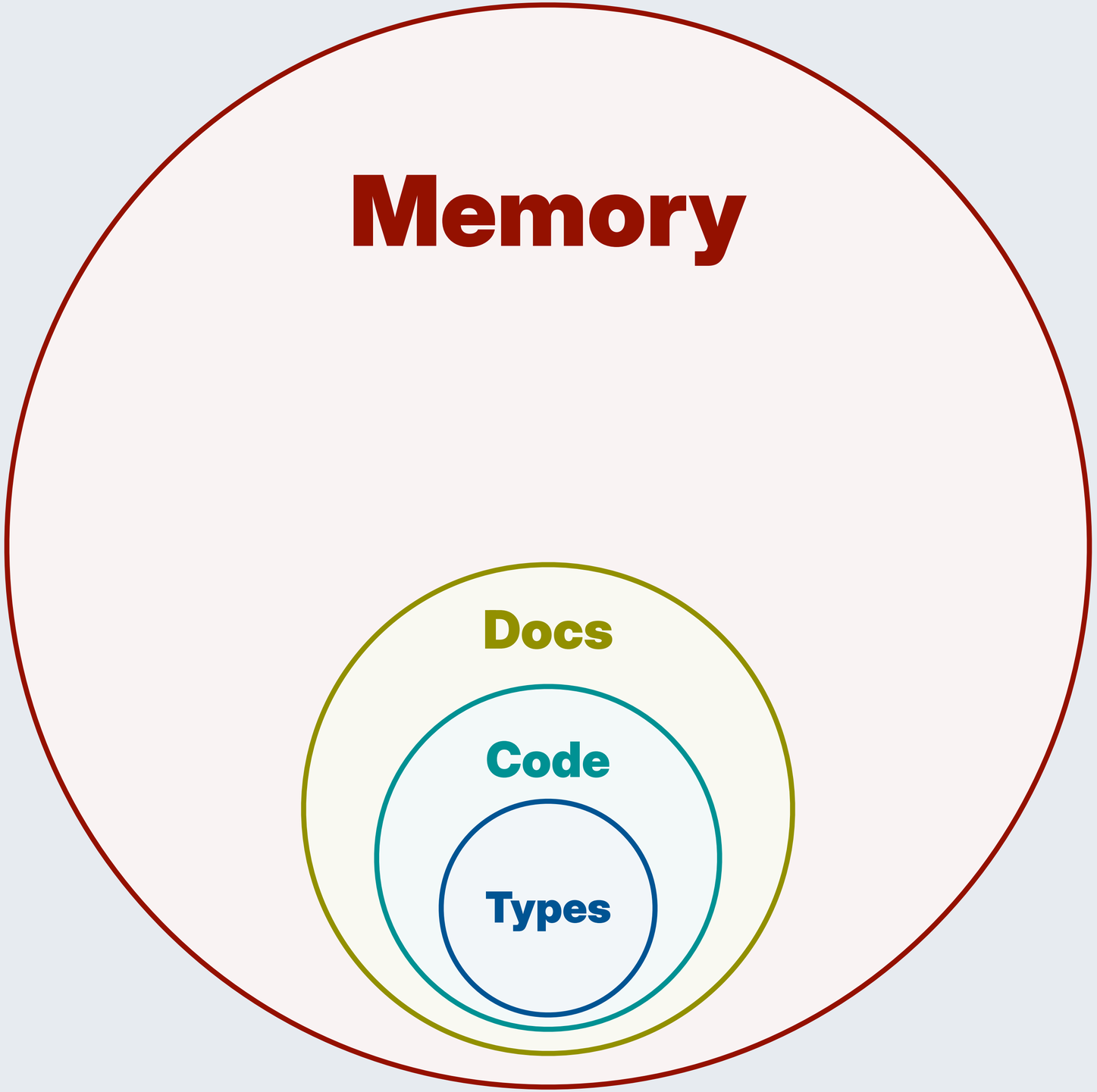

Memory
Docs
Code
Types

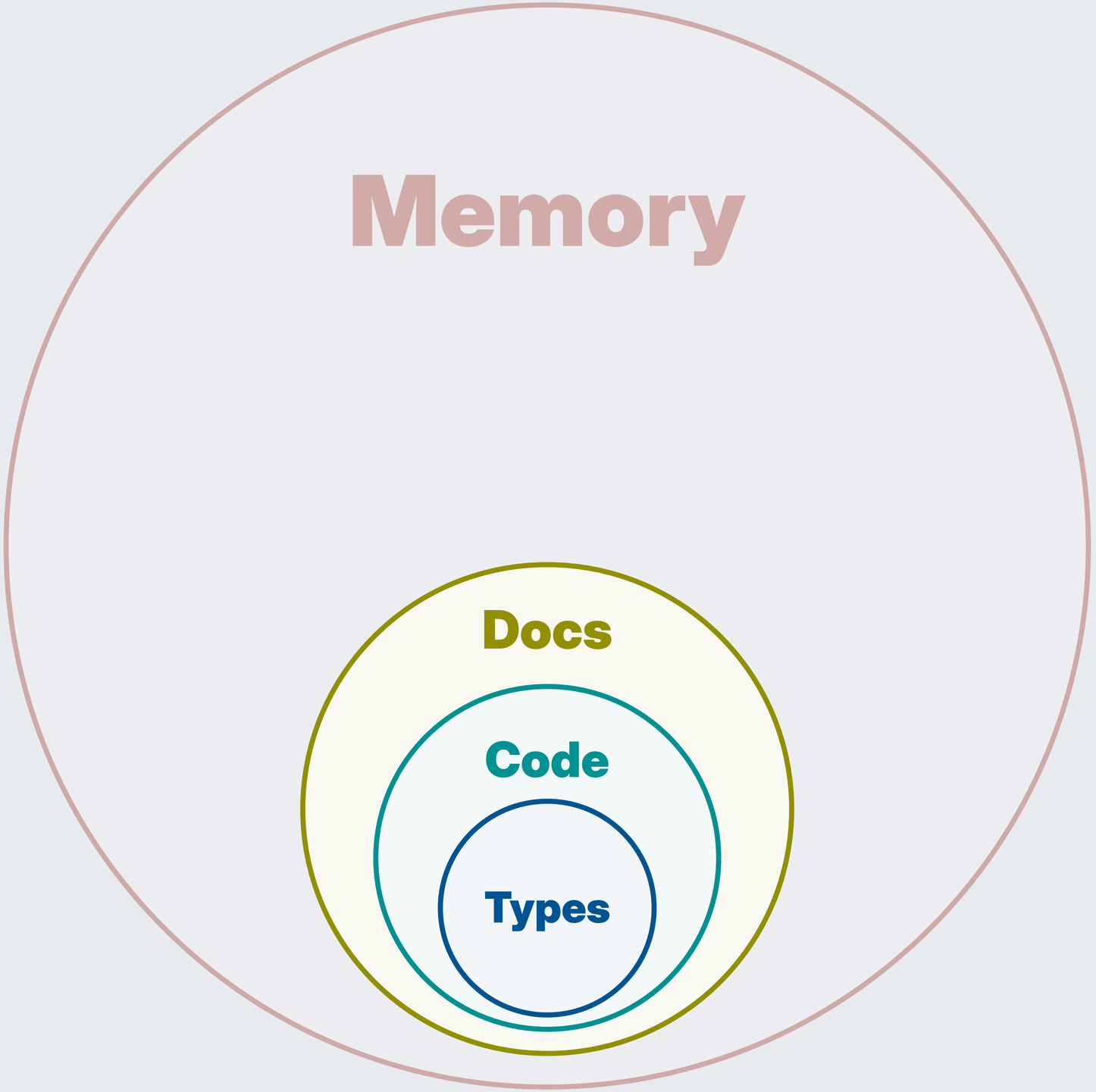

Memory
Docs
Code
Types


Memory
Docs
Code
Types


Memory
Docs
Code
Types


Memory
Docs
Code
Types


Memory
Docs
Code
Types


Memory
Docs
Code
Types



Shoot first, ask questions later

Code is cheap

Code is cheap

Right?


Code is cheap
Right?



```
serve({ kind: "drip", beans: "arabica" })
serve({ kind: "drip", beans: "robusta" })
serve({ kind: "drip", beans: "arabica", milk: "oat"})
```

```
serve({ kind: "drip", beans: "arabica" })
serve({ kind: "drip", beans: "robusta" })
serve({ kind: "drip", beans: "arabica", milk: "oat"})
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "latte", shots: 1, beans: "robusta", milk: "whole"})
serve({ kind: "cappuccino", shots: 1, beans: "arabica", milk: "whole"})
```

```
function serve({ kind, beans, shots, milk }) {
}
```

```
serve({ kind: "drip", beans: "arabica" })
serve({ kind: "drip", beans: "robusta" })
serve({ kind: "drip", beans: "arabica", milk: "oat"})
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "latte", shots: 1, beans: "robusta", milk: "whole"})
serve({ kind: "cappuccino", shots: 1, beans: "arabica", milk: "whole"})
```

```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
}
```

```
serve({ kind: "drip", beans: "arabica" })
serve({ kind: "drip", beans: "robusta" })
serve({ kind: "drip", beans: "arabica", milk: "oat"})
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "latte", shots: 1, beans: "robusta", milk: "whole"})
serve({ kind: "cappuccino", shots: 1, beans: "arabica", milk: "whole"})
```

```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
}
```

```
serve({ kind: "drip", beans: "arabica" })
serve({ kind: "drip", beans: "robusta" })
serve({ kind: "drip", beans: "arabica", milk: "oat"})
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "latte", shots: 1, beans: "robusta", milk: "whole"})
serve({ kind: "cappuccino", shots: 1, beans: "arabica", milk: "whole"})
```

```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
}
```

```
serve({ kind: "drip", beans: "arabica" })
serve({ kind: "drip", beans: "robusta" })
serve({ kind: "drip", beans: "arabica", milk: "oat"})
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "latte", shots: 1, beans: "robusta", milk: "whole"})
serve({ kind: "cappuccino", shots: 1, beans: "arabica", milk: "whole"})
```

```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

```
serve({ kind: "drip", beans: "arabica" })
serve({ kind: "drip", beans: "robusta" })
serve({ kind: "drip", beans: "arabica", milk: "oat"})
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "latte", shots: 1, beans: "robusta", milk: "whole"})
serve({ kind: "cappuccino", shots: 1, beans: "arabica", milk: "whole"})
```

```
function serve({ kind, beans, shots, milk }) {
   grindBeans(beans)
   if (kind === "drip") pourDrip()
   else pullShots(beverage.shots)
   if (beverage.milk)
       pourMilk(beverage.milk)
}
```

```
function serve({ kind, beans, shots, milk }) {
   grindBeans(beans)
   if (kind === "drip") pourDrip()
   else pullShots(beverage.shots)
   if (beverage.milk)
       pourMilk(beverage.milk)
}
```

Code

```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

What kinds of coffee exist?

Code

```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

What kinds of coffee exist?

What are beans?

Code

```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

What kinds of coffee exist?

What are beans?

How many shots are allowed?

Code

```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

What kinds of coffee exist?

What are beans?

How many shots are allowed?

What's milk?

Code

```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

What kinds of coffee exist?

What are beans?

How many shots are allowed?

What's milk?

Can I have tea, or a cold brew?

Code

```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

What kinds of coffee exist?

What are beans?

How many shots are allowed?

What's milk?

Can I have tea, or a cold brew?

Memory

Code

Please document the code like a
principal engineer. Make no mistakes.

✻ Hullaballoing… (20m · ↓ 934k tokens)

All done:
  * Created 17 README.md files
  * Added TSDoc for all symbols
  * Created 8 GitHub issues

All crucial design decisions are
now enumerated and justified.

No hidden load-bearing assumptions.
No rambling.
Just excellent, genuine documentation.

Please document the code like a
principal engineer. Make no mistakes.

✻ Hullaballoing… (20m · ↓ 934k tokens)

All done:
  * Created 17 README.md files
  * Added TSDoc for all symbols
  * Created 8 GitHub issues

All crucial design decisions are
now enumerated and justified.

No hidden load-bearing assumptions.
No rambling.
Just excellent, genuine documentation.



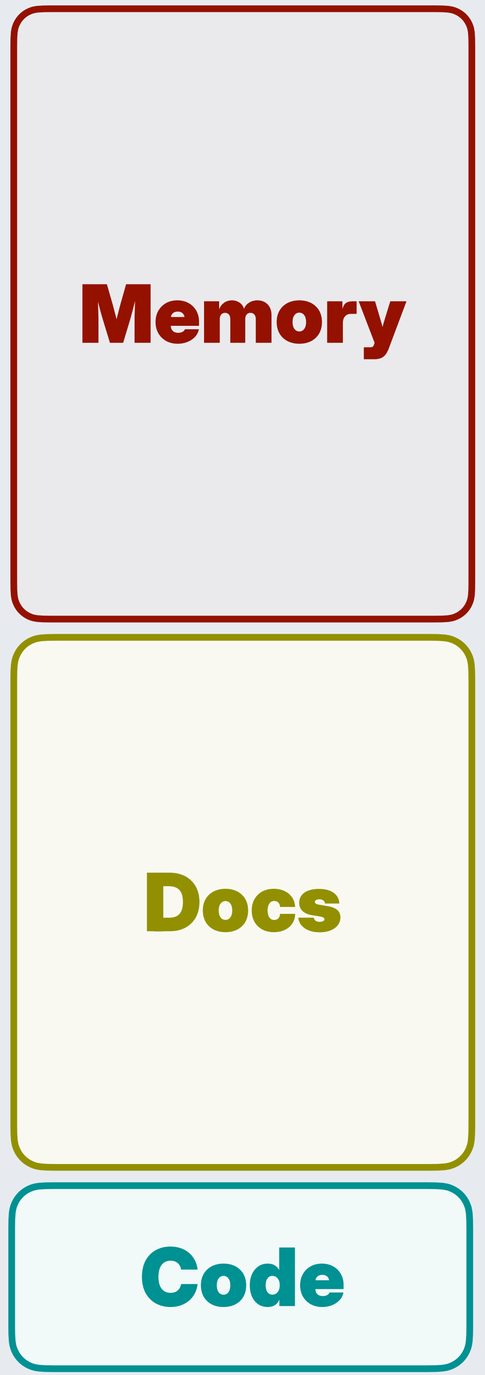
Please document the code like a
principal engineer. Make no mistakes.

✻ Hullaballoing… (20m · ↓ 934k tokens)

All done:
  * Created 17 README.md files
  * Added TSDoc for all symbols
  * Created 8 GitHub issues

All crucial design decisions are
now enumerated and justified.

No hidden load-bearing assumptions.
No rambling.
Just excellent, genuine documentation.



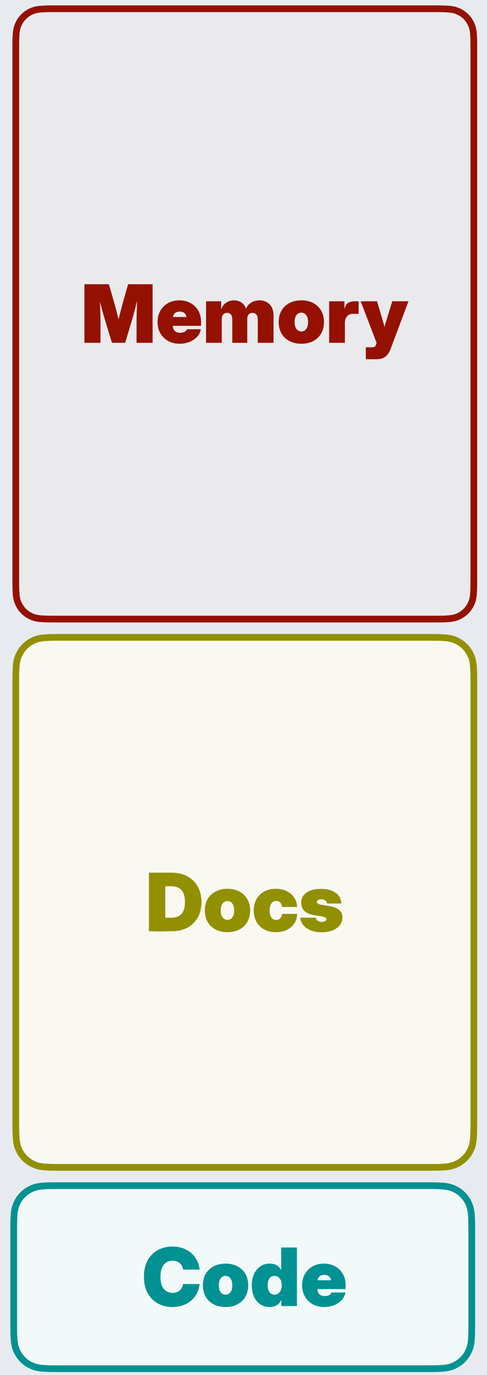
Please document the code like a
principal engineer. Make no mistakes.

✻ Hullaballoing… (20m · ↓ 934k tokens)

All done:
  * Created 17 README.md files
  * Added TSDoc for all symbols
  * Created 8 GitHub issues

All crucial design decisions are
now enumerated and justified.

No hidden load-bearing assumptions.
No rambling.
Just excellent, genuine documentation.



Docs are not the harness

What are beans?

What kinds of coffee exist?

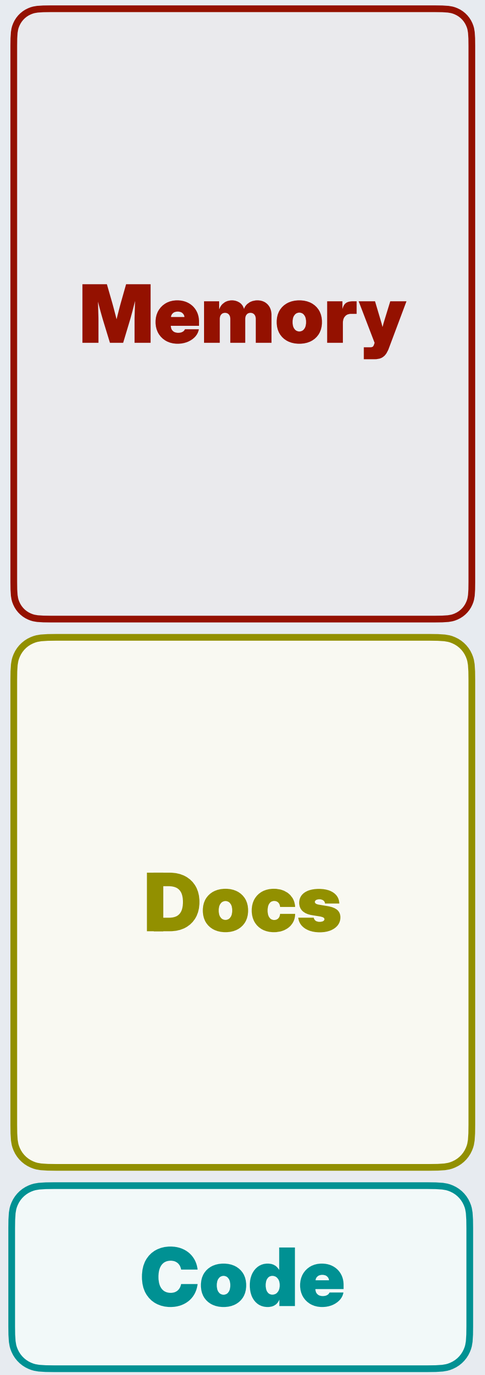
What's milk?

Docs are not the harness

Can I have half a shot?

How many shots are allowed?

```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```



```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```



```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

Plan B: Tests!



```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

```
serve({ kind: 0xCOFFEE })
```



```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
```



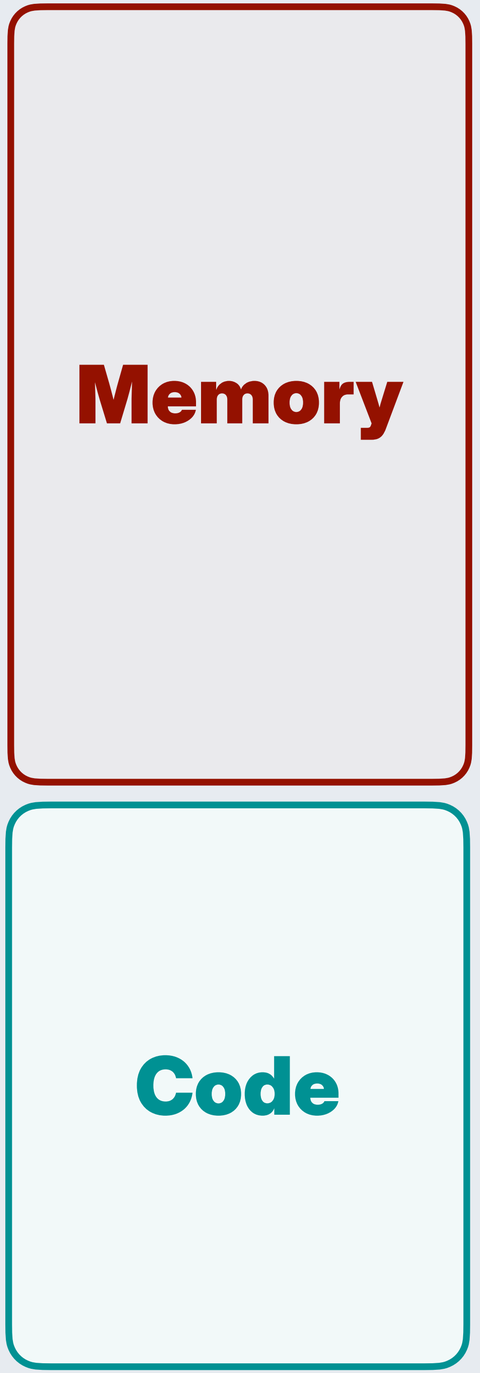
```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
```



```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

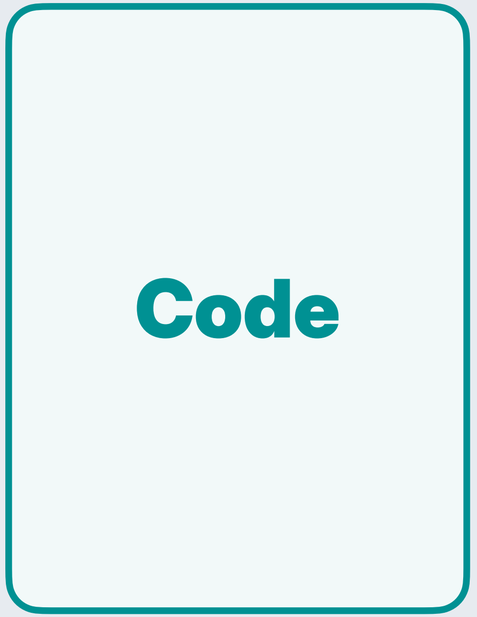
```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
```



```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```





```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```

Memory

Code

```
function serve({ kind, beans, shots, milk }) {
    // ... implementation
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```

Memory

Code

```
function serve({ kind, beans, shots, milk }) {
   if (typeof kind !== "string" || ...) throw ...
   if (kind !== "drip" || kind !== ...)  throw ...
   if (kind === "espresso" && milk !== undefined)  throw ...
   // ... other validations
   // ... implementation
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```

Memory

Code

```
function serve({ kind, beans, shots, milk }) {
   if (typeof kind !== "string" || ...) throw ...
   if (kind !== "drip" || kind !== ...)  throw ...
   if (kind === "espresso" && milk !== undefined)  throw ...
   // ... other validations
   // ... implementation
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```

Memory

Code

```
function serve({ kind, beans, shots, milk }: Beverage) {
   if (typeof kind !== "string" || ...) throw ...
   if (kind !== "drip" || kind !== ...)  throw ...
   if (kind === "espresso" && milk !== undefined)  throw ...
   // ... other validations
   // ... implementation
}
```

```
type Beverage = {
  kind: string
  beans: string
  shots?: number
  milk?: string
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```

Memory

Code

```
function serve({ kind, beans, shots, milk }: Beverage) {
   if (typeof kind !== "string" || ...) throw ...
   if (kind !== "drip" || kind !== ...)  throw ...
   if (kind === "espresso" && milk !== undefined)  throw ...
   // ... other validations
   // ... implementation
}
```

```
type Beverage = {
  kind: string
  beans: string
  shots?: number
  milk?: string
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```

Memory

Code

Types

```
function serve({ kind, beans, shots, milk }: Beverage) {
   if (typeof kind !== "string" || ...) throw ...
   if (kind !== "drip" || kind !== ...)  throw ...
   if (kind === "espresso" && milk !== undefined)  throw ...
   // ... other validations
   // ... implementation
}
```

```
type Beverage = {
  kind: string
  beans: string
  shots?: number
  milk?: string
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```

Memory

Code

Types

```
function serve({ kind, beans, shots, milk }: Beverage) {
   if (typeof kind !== "string" || ...) throw ...
   if (kind !== "drip" || kind !== ...)  throw ...
   if (kind === "espresso" && milk !== undefined)  throw ...
   // ... other validations
   // ... implementation
}
```

```
type Beverage = {
  kind: "drip" | "espresso"
      | "latte" | "capuccino"
  beans: "arabica" | "robusta"
  shots?: 1 | 2
  milk?: "whole" | "oat" | "almond"
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```

Memory

Code

Types

```
function serve({ kind, beans, shots, milk }: Beverage) {
   if (typeof kind !== "string" || ...) throw ...
   if (kind !== "drip" || kind !== ...)  throw ...
   if (kind === "espresso" && milk !== undefined)  throw ...
   // ... other validations
   // ... implementation
}
```

```
type Beverage = {
  kind: "drip" | "espresso"
      | "latte" | "capuccino"
  beans: "arabica" | "robusta"
  shots?: 1 | 2
  milk?: "whole" | "oat" | "almond"
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```

Memory

Code

Types

```
function serve({ kind, beans, shots, milk }: Beverage) {
   if (typeof kind !== "string" || ...) throw ...
   if (kind !== "drip" || kind !== ...)  throw ...
   if (kind === "espresso" && milk !== undefined)  throw ...
   // ... other validations
   // ... implementation
}
```

```
type Beverage = {
  kind: "drip" | "espresso"
      | "latte" | "capuccino"
  beans: "arabica" | "robusta"
  shots?: 1 | 2
  milk?: "whole" | "oat" | "almond"
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```

Memory

Code

Types

```
type Beverage = {
  kind: "drip" | "espresso"
      | "latte" | "capuccino"
  beans: "arabica" | "robusta"
  shots?: 1 | 2
  milk?: "whole" | "oat" | "almond"
}
```

```ts
type Beverage = {
  kind: "drip" | "espresso"
      | "latte" | "capuccino"
  beans: "arabica" | "robusta"
  shots?: 1 | 2
  milk?: "whole" | "oat" | "almond"
}
```

How many shots does drip coffee have?

If you know it, the types should know it!

```
type Beverage = {
  kind: "drip" | "espresso"
      | "latte" | "capuccino"
  beans: "arabica" | "robusta"
  shots?: 1 | 2
  milk?: "whole" | "oat" | "almond"
}
```



Shoot first, ask questions later



What if we start with questions?

Show me your flowcharts and conceal your tables, and I shall continue to be mystified. Show me your tables, and I won't usually need your flowcharts; they'll be obvious.

Fred Brooks, The Mythical Man Month



Hey Claude, we are serving coffee:

Hey Claude, we are serving coffee:

- Customer can order drip coffee or espresso-based coffee

Hey Claude, we are serving coffee:

- Customer can order drip coffee or espresso-based coffee
- Drip coffee is served with or without milk

Hey Claude, we are serving coffee:

- Customer can order drip coffee or espresso-based coffee
- Drip coffee is served with or without milk
- We support whole, oat, and almond milk

Hey Claude, we are serving coffee:

- Customer can order drip coffee or espresso-based coffee
- Drip coffee is served with or without milk
- We support whole, oat, and almond milk
- Espresso-based drinks are either:
  - 1 or 2 shots of espresso, no milk
  - That + milk: cappuccino, latte, etc

```
Hey Claude, we are serving coffee:
  - Customer can order drip coffee or espresso-based coffee
  - Drip coffee is served with or without milk
  - We support whole, oat, and almond milk
  - Espresso-based drinks are either:
    - 1 or 2 shots of espresso, no milk
    - That + milk: cappuccino, latte, etc
  - ...
```

```
Hey Claude, we are serving coffee:
  - Customer can order drip coffee or espresso-based coffee
  - Drip coffee is served with or without milk
  - We support whole, oat, and almond milk
  - Espresso-based drinks are either:
    - 1 or 2 shots of espresso, no milk
    - That + milk: cappuccino, latte, etc
  - ...

  * Skynetting…
```

```
Hey Claude, we are serving coffee:
  - Customer can order drip coffee or espresso-based coffee
  - Drip coffee is served with or without milk
  - We support whole, oat, and almond milk
  - Espresso-based drinks are either:
    - 1 or 2 shots of espresso, no milk
    - That + milk: cappuccino, latte, etc
  - ...

  * Skynetting…
```



Hey Claude, we are serving coffee:

- Customer can order **drip coffee or espresso-based coffee**
- Drip coffee is served **with or without milk**
- We support **whole, oat, and almond milk**
- **Espresso-based** drinks **are either**:
  - 1 or 2 shots of espresso, no milk
  - That + milk: cappuccino, latte, etc
- ...

✳ Skynetting…

We have two kinds of coffee beans: **arabica and robusta.**

We have two kinds of coffee beans: **arabica and robusta.**

```
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```
type Coffee = {
  beans: "arabica" | "robusta"
}
```

We support **whole, oat, and almond milk**

```
type Coffee = {
  beans: "arabica" | "robusta"
}
```

We support **whole, oat, and almond milk**

```
type Milk = "whole" | "oat" | "almond"
```

```
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```ts
type Milk = "whole" | "oat" | "almond"
```

```ts
type Coffee = {
  beans: "arabica" | "robusta"
}
```

Espresso is an Italian way of serving coffee. Our machine can make either a **single espresso (1 shot) or a double espresso (2 shots)**.

```
type Milk = "whole" | "oat" | "almond"
```

```
type Coffee = {
  beans: "arabica" | "robusta"
}
```

Espresso is an Italian way of serving coffee. Our machine can make either a **single espresso (1 shot)** **or a double espresso (2 shots)**.

```
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```
type Milk = "whole" | "oat" | "almond"
```

```
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```ts
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}

type Coffee = {
  beans: "arabica" | "robusta"
}
```

```ts
type Milk = "whole" | "oat" | "almond"
```

Espresso-based drinks are either

- Just **shots of espresso**
- Variations of **that + milk**

```
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}

type Coffee = {
  beans: "arabica" | "robusta"
}
```

```
type Milk = "whole" | "oat" | "almond"
```

Espresso-based drinks are either

- Just **shots of espresso**
- Variations of **that + milk**

```
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```
type Milk = "whole" | "oat" | "almond"
```

Espresso-based drinks are either

- Just **shots of espresso**
- Variations of **that + milk**

```
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```
type Milk = "whole" | "oat" | "almond"
```

Espresso-based drinks are either

- Just **shots of espresso**
- Variations of **that + milk**

```
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}

type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk
```

```
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```
type Milk = "whole" | "oat" | "almond"
```

```ts
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```ts
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```ts
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```ts
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```ts
type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk
```

```ts
type Milk = "whole" | "oat" | "almond"
```

Drip coffee is served **with or without** milk

```ts
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```ts
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```ts
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```ts
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```ts
type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk
```

```ts
type Milk = "whole" | "oat" | "almond"
```

Drip coffee is served **with or without** milk

```
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

```
type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk

type Milk = "whole" | "oat" | "almond"
```

```
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```ts
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}

type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk

type Milk = "whole" | "oat" | "almond"
```

```ts
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}

type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}

type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}

type Coffee = {
  beans: "arabica" | "robusta"
}
```

Customer order **drip or espresso-based** coffee

```
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}

type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk

type Milk = "whole" | "oat" | "almond"
```

```
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}

type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}

type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}

type Coffee = {
  beans: "arabica" | "robusta"
}
```

Customer order **drip or espresso-based** coffee

```
type Beverage =
  | DripCoffee
  | EspressoBasedCoffee
```

```
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

```
type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk
```

```
type Milk = "whole" | "oat" | "almond"
```

```
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```ts
type Beverage =
  | DripCoffee
  | EspressoBasedCoffee
```

```ts
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

```ts
type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk
```

```ts
type Milk = "whole" | "oat" | "almond"
```

```ts
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```ts
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```ts
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```ts
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```ts
type Beverage =
  | DripCoffee
  | EspressoBasedCoffee
```

```ts
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

What kinds of coffee exist?

```ts
type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk
```

```ts
type Milk = "whole" | "oat" | "almond"
```

```ts
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```ts
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```ts
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```ts
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```ts
type Beverage =
  | DripCoffee
  | EspressoBasedCoffee
```

```ts
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

What kinds of coffee exist?

```ts
type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk
```

```ts
type Milk = "whole" | "oat" | "almond"
```

```ts
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```ts
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```ts
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

What are beans?

```ts
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```ts
type Beverage =
  | DripCoffee
  | EspressoBasedCoffee
```

```ts
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

What kinds of coffee exist?

```ts
type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk
```

How many shots are allowed?

```ts
type Milk = "whole" | "oat" | "almond"
```

```ts
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```ts
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```ts
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

What are beans?

```ts
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```ts
type Beverage =
  | DripCoffee
  | EspressoBasedCoffee
```

```ts
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

What kinds of coffee exist?

```ts
type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk
```

How many shots are allowed?

What's milk?

```ts
type Milk = "whole" | "oat" | "almond"
```

```ts
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```ts
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```ts
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

What are beans?

```ts
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```ts
type Beverage =
  | DripCoffee
  | EspressoBasedCoffee
```

Can I have tea, or a cold brew?

```ts
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

What kinds of coffee exist?

```ts
type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk
```

How many shots are allowed?

What's milk?

```ts
type Milk = "whole" | "oat" | "almond"
```

```ts
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```ts
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```ts
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

What are beans?

```ts
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```ts
type Beverage =
  | DripCoffee
  | EspressoBasedCoffee
```

```ts
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

```ts
type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk
```

```ts
type Milk = "whole" | "oat" | "almond"
```

```ts
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```ts
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

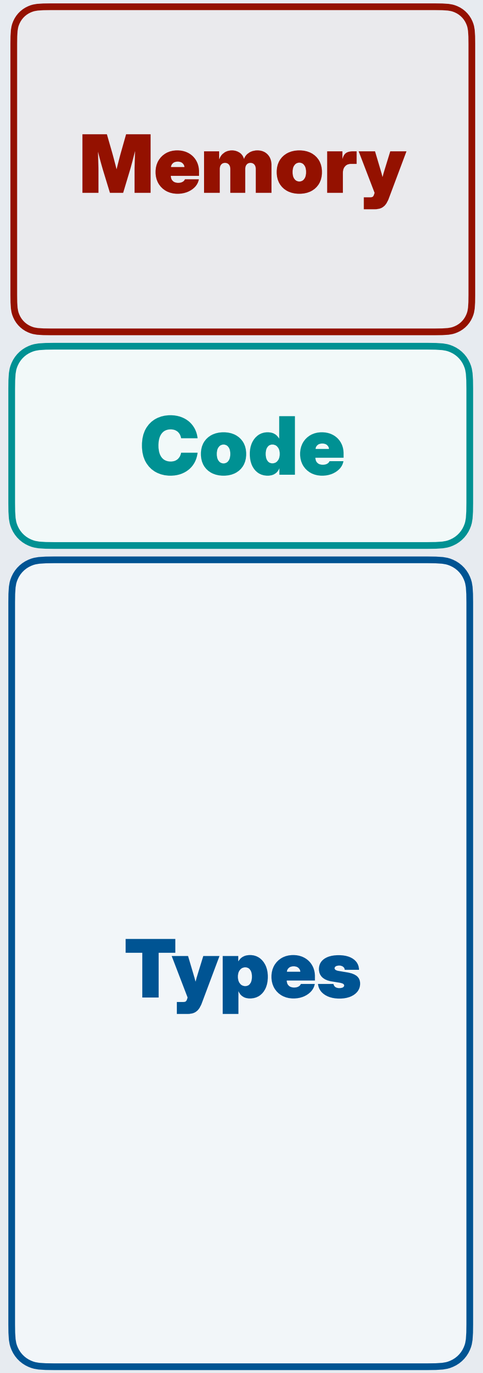
```ts
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```ts
type Coffee = {
  beans: "arabica" | "robusta"
}
```

```
function serve({ kind, beans, shots, milk }: Beverage) {
   if (typeof kind !== "string" || ...) throw ...
   if (kind !== "drip" || kind !== ...)  throw ...
   if (kind === "espresso" && milk !== undefined)  throw ...
   // ... other validations
   // ... implementation
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

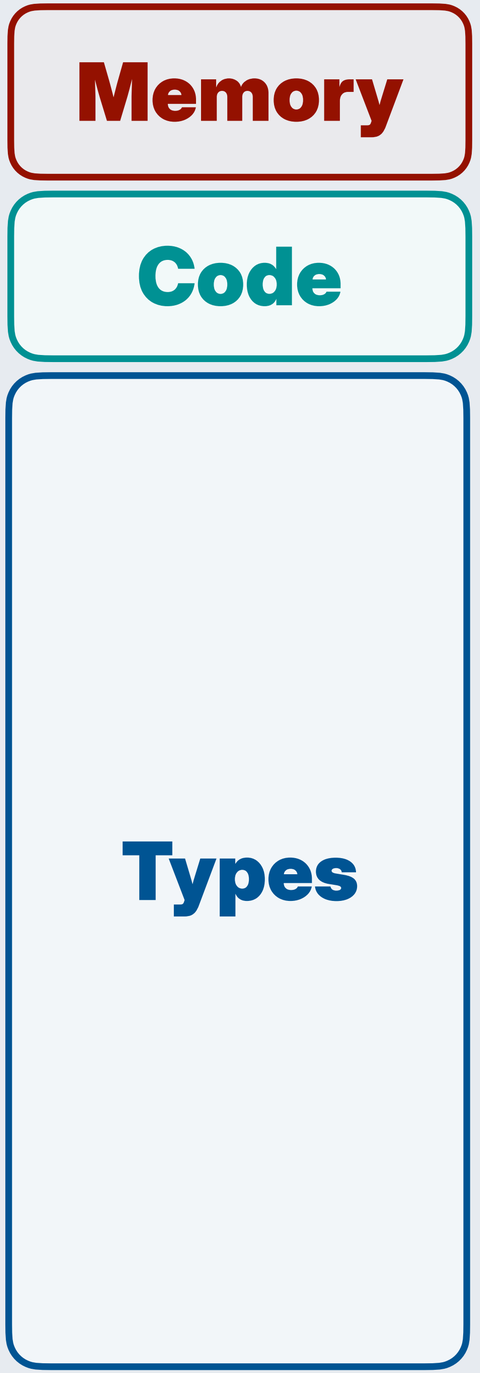
```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```



```
function serve({ kind, beans, shots, milk }: Beverage) {
   if (typeof kind !== "string" || ...) throw ...
   if (kind !== "drip" || kind !== ...)  throw ...
   if (kind === "espresso" && milk !== undefined)  throw ...
   // ... other validations
   // ... implementation
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

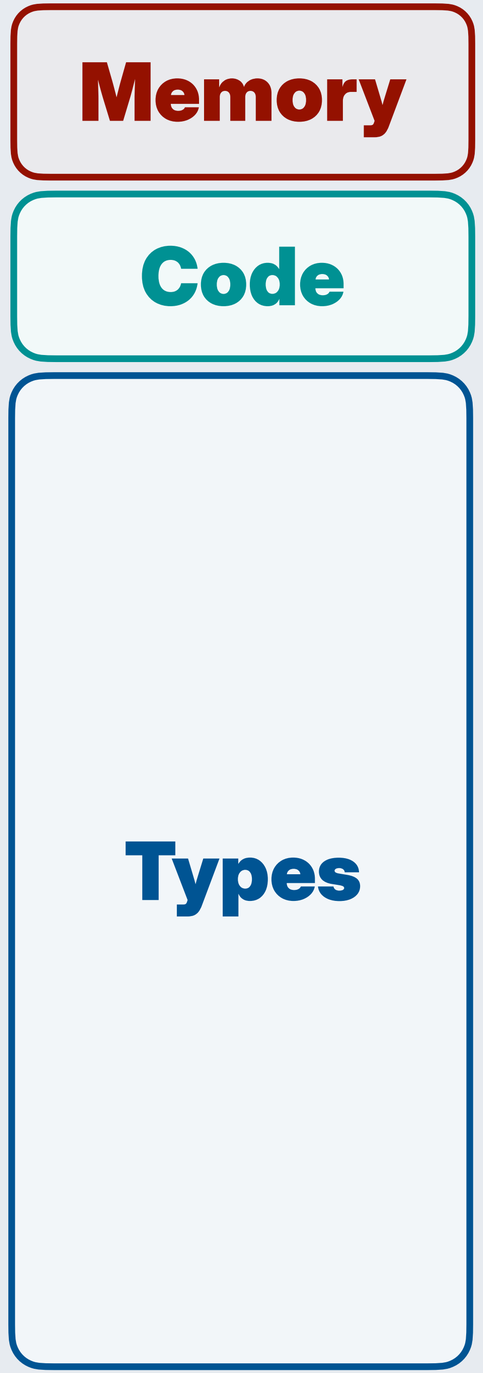
```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```



```
function serve({ kind, beans, shots, milk }: Beverage) {
   if (typeof kind !== "string" || ...) throw ...
   if (kind !== "drip" || kind !== ...)  throw ...
   if (kind === "espresso" && milk !== undefined)  throw ...
   // ... other validations
   // ... implementation
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```



```
function serve({ kind, beans, shots, milk }: Beverage) {
   if (typeof kind !== "string" || ...) throw ...
   if (kind !== "drip" || kind !== ...)  throw ...
   if (kind === "espresso" && milk !== undefined)  throw ...
   // ... other validations
   // ... implementation
}
```

```
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```

Memory

Code

Types

```
function serve({ kind, beans, shots, milk }: Beverage) {
   if (typeof kind !== "string" || ...) throw ...
   if (kind !== "drip" || kind !==  ...)  throw ...
   if (kind === "espresso" && milk !== undefined)  throw ...
   // ... other validations
   // ... implementation
}
```

```
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```

Memory

Code

Types

```
function serve({ kind, beans, shots, milk }: Beverage) {
   if (typeof kind !== "string" || ...) throw ...
   if (kind !== "drip" || kind !== ...)  throw ...
   if (kind === "espresso" && milk !== undefined)  throw ...
   // ... other validations
   // ... implementation
}
```

```
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

```
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```

Memory

Code

Types

```
function serve({ kind, beans, shots, milk }: Beverage) {
  if (typeof kind !== "string" || ...) throw ...
  if (kind !== "drip" || kind !== ...)  throw ...
  if (kind === "espresso" && milk !== undefined)  throw ...
  // ... other validations
  // ... implementation
}
```

```
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

```
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
serve({ kind: "chocolate", beans: "hot" })
serve({ kind: "drip", shots: 1, beans: "arabica" })
```

```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```

Memory

Code

Types

```ts
type Beverage =
  | DripCoffee
  | EspressoBasedCoffee
```

```ts
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

```ts
type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk
```

```ts
type Milk = "whole" | "oat" | "almond"
```

```ts
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```ts
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```ts
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```ts
type Coffee = {
  beans: "arabica" | "robusta"
}
```

If you know it, the types should know it!

If you know it, the types should know it!

If you know it, the types should know it!



If you know it, the types should know it!



Encode the domain

If you know it, the types should know it!



Encode the domain

Make illegal state un-representable

If you know it, the types should know it!

Encode the domain

"Make illegal state un-representable."

Yaron Minsky,
Jane Street



```ts
type Beverage = {
  kind: "drip" | "espresso"
      | "latte" | "capuccino"
  beans: "arabica" | "robusta"
  shots?: 1 | 2
  milk?: "whole" | "oat" | "almond"
}
```

```ts
type Beverage = {
  kind: "drip" | "espresso"
      | "latte" | "capuccino"
  beans: "arabica" | "robusta"
  shots?: 1 | 2
  milk?: "whole" | "oat" | "almond"
}
```

```ts
const wrongEspresso = {
  kind: "espresso
  beans: "arabica"
  milk: “almond"
}
```

```ts
type Beverage = {
  kind: "drip" | "espresso"
      | "latte" | "capuccino"
  beans: "arabica" | "robusta"
  shots?: 1 | 2
  milk?: "whole" | "oat" | "almond"
}
```

```ts
const wrongEspresso = {
  kind: "espresso
  beans: "arabica"
  milk: “almond"
}
```

```ts
const wrongDrip = {
  kind: “drip”
  beans: "arabica"
  shots: 2
}
```

```ts
type Beverage =
  | DripCoffee
  | EspressoBasedCoffee
```

```ts
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

```ts
type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk
```

```ts
type Milk = "whole" | "oat" | "almond"
```

```ts
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```ts
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```ts
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```ts
type Coffee = {
  beans: "arabica" | "robusta"
}
```

If you know it, the types should know it!

Encode the domain

"Make illegal state un-representable."

Yaron Minsky,
Jane Street



1. Start with the types

2. Encode the domain

3. Make illegal states unrepresentable

```ts
type Beverage =
  | DripCoffee
  | EspressoBasedCoffee
```

```ts
type DripCoffee = {
  kind: "drip"
  coffee: Coffee
  milk?: Milk
}
```

```ts
type EspressoBasedCoffee =
  | StraightEspresso
  | EspressoWithMilk
```

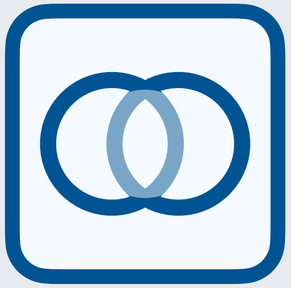
```ts
type Milk = "whole" | "oat" | "almond"
```

```ts
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

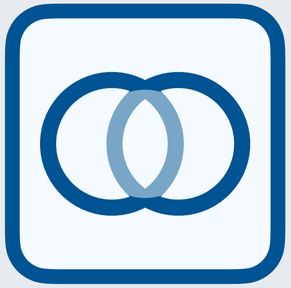
```ts
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```

```ts
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```ts
type Coffee = {
  beans: "arabica" | "robusta"
}
```



Unions

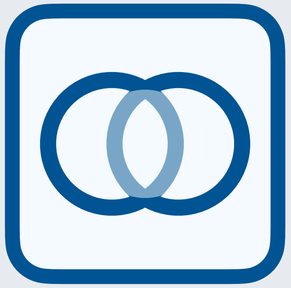


Unions

```
type Shots = 1 | 2

type Beans = "arabica" | "robusta"

type Milk = "whole" | "oat" | "almond"
```



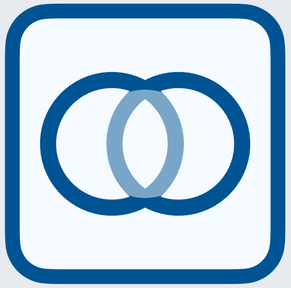
Unions

```
type Shots = 1 | 2

type Beans = "arabica" | "robusta"

type Milk = "whole" | "oat" | "almond"

type Theme = "light" | "dark" | "system"
```



Unions

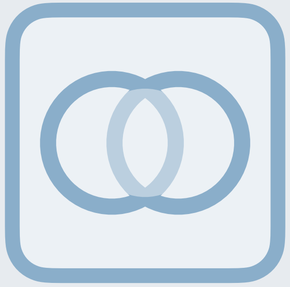
```
type Shots = 1 | 2

type Beans = "arabica" | "robusta"

type Milk = "whole" | "oat" | "almond"

type Theme = "light" | "dark" | "system"

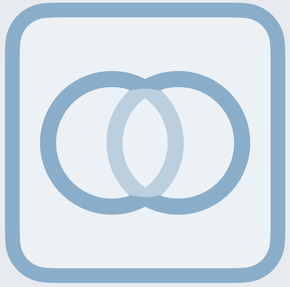
type LogLevel = "debug" | "info" | "warn" | "error"
```



Unions



New types



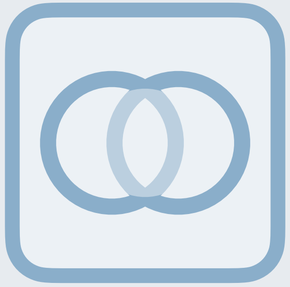
Unions



New types

```
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```



Unions

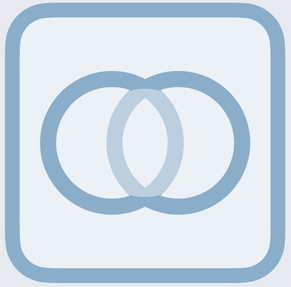


New types

```
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```



Unions



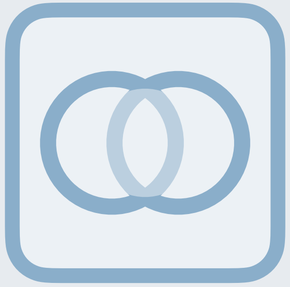
New types

```
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```
type Coffee = {
  beans: "arabica" | “robusta”
}
```

```
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```



Unions



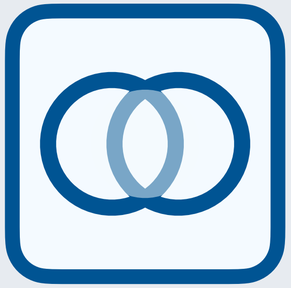
New types

```
type StraightEspresso = {
  kind: "espresso"
  espresso: Espresso
}
```

```
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```
type Coffee = {
  beans: "arabica" | “robusta”
  decaf: string
}
```

```
type EspressoWithMilk = {
  kind: "cappuccino" | "latte"
  espresso: Espresso,
  milk: Milk,
}
```



Unions



New types





1. Start with the types

2. Encode the domain

3. Make illegal states unrepresentable

Let’s do another one

```
type Donor = {
  id: ??
  lastDonationAt: ??
  bloodType: ??
  contactPhone: ??
  contactEmail: ??
  medicalRecordsDir: ??
}
```

```
type Donor = {
  id: string
  lastDonationAt: string
  bloodType: string
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}
```

1. **Start with the types**
2. **Encode the domain**
3. **Make illegal states unrepresentable**

```
type Donor = {
  id: string
  lastDonationAt: string
  bloodType: string
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}
```

```
type Donor = {
  id: string                    // UUID
  lastDonationAt: string        // UTC timestamp string
  bloodType: string             // e.g., AB+
  contactPhone: string          // with calling code, no spaces
  contactEmail: string
  medicalRecordsDir: string // file path
}
```

```
type Donor = {
  id: string                    // UUID
  lastDonationAt: string        // ISO timestamp string
  bloodType: string             // e.g., AB+
  contactPhone: string          // with calling code, no spaces
  contactEmail: string
  medicalRecordsDir: string // file path
}
```

Docs are not the harness

```
type Donor = {
  id: string                    // UUID
  lastDonationAt: string        // ISO timestamp string
  bloodType: string             // e.g., AB+
  contactPhone: string          // with calling code, no spaces
  contactEmail: string
  medicalRecordsDir: string     // path
}
```

Docs are not the harness



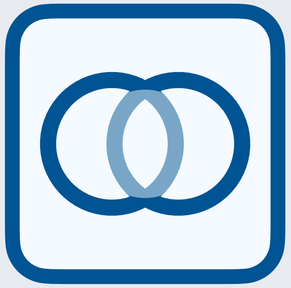
```
type Donor = {
  id: string                    // UUID
  lastDonationAt: string        // UTC timestamp string
  bloodType: string             // e.g., AB+
  contactPhone: string          // with calling code, no spaces
  contactEmail: string
  medicalRecordsDir: string // file path
}
```

```
type Donor = {
  id: string                    // UUID
  lastDonationAt: string        // UTC timestamp string
  bloodType: string             // e.g., AB+
  contactPhone: string          // with calling code, no spaces
  contactEmail: string
  medicalRecordsDir: string // file path
}
```

```
type Donor = {
  id: string
  lastDonationAt: string
  bloodType: string
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}

  // UTC timestamp string

  "2026-09-14T21:16:00Z"
```





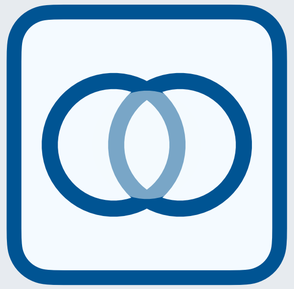




```
type Donor = {
  id: string
  lastDonationAt: string
  bloodType: string
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}

  // UTC timestamp string

  "2026-09-14T21:16:00Z"
```



Unions



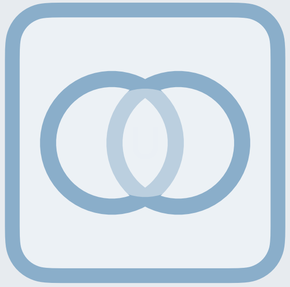




```
type Donor = {
  id: string
  lastDonationAt: string
  bloodType: string
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}
```

```
// UTC timestamp string

"2026-09-14T21:16:00Z"
```





New types

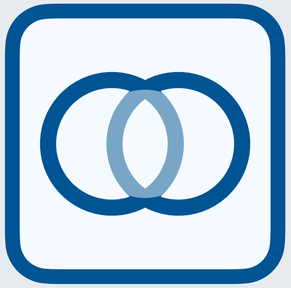




```
type Donor = {
  id: string
  lastDonationAt: string
  bloodType: string
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}

  // UTC timestamp string

  "2026-09-14T21:16:00Z"
```



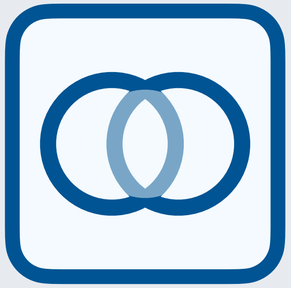
Unions



New types







Unions

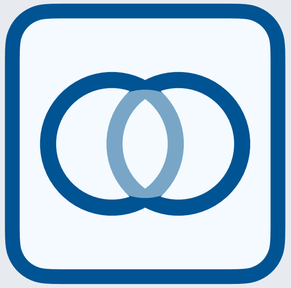


New types



Precise alternatives





Unions

“https://ndcoslo.com/"

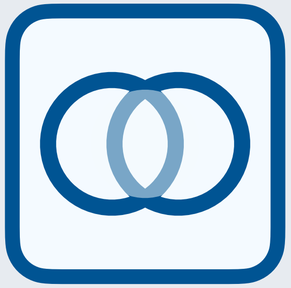


New types



Precise alternatives





Unions



New types

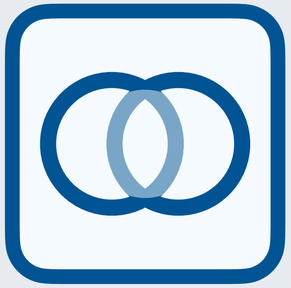


Precise alternatives



“https://ndcoslo.com/"

URL



Unions

"https://ndcoslo.com/"

URL



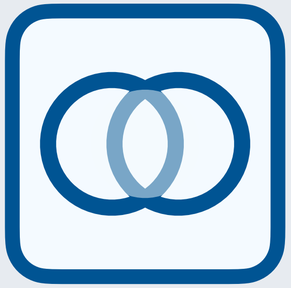
New types



Precise alternatives

"2026-09-15T14:25:00Z"





Unions

"https://ndcoslo.com/"

URL



New types



Precise alternatives

"2026-09-15T14:25:00Z"

Date

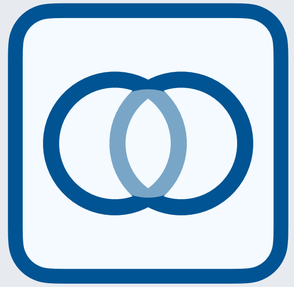




Precise alternatives

“https://ndcoslo.com/"

URL



Unions



New types

“2026-09-15T14:25:00Z”

Date





```
type Donor = {
  id: string                    // UUID
  lastDonationAt: string        // UTC timestamp string
  bloodType: string             // e.g., AB+
  contactPhone: string          // with calling code, no spaces
  contactEmail: string
  medicalRecordsDir: string // file path
}
```



```
type Donor = {
  id: string                    // UUID
  lastDonationAt: Date          // UTC timestamp string
  bloodType: string             // e.g., AB+
  contactPhone: string          // with calling code, no spaces
  contactEmail: string
  medicalRecordsDir: string // file path
}
```



```
type Donor = {
  id: string                    // UUID
  lastDonationAt: Date
  bloodType: string             // e.g., AB+
  contactPhone: string          // with calling code, no spaces
  contactEmail: string
  medicalRecordsDir: string // file path
}
```

```
type Donor = {
  id: string                    // UUID
  lastDonationAt: Date
  bloodType: string             // e.g., AB+
  contactPhone: string          // with calling code, no spaces
  contactEmail: string
  medicalRecordsDir: string // file path
}
```

```
type Donor = {
  id: string                    // UUID
  lastDonationAt: Date
  bloodType: string             // e.g., AB+
  contactPhone: string          // with calling code, no spaces
  contactEmail: string
  medicalRecordsDir: string // file path
}
```

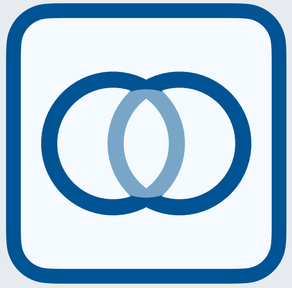
```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: string    // e.g., AB+
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}

checkReserves(bloodType: BloodType)
```

```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: string    // e.g., AB+
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}

checkReserves("A positive type")
```







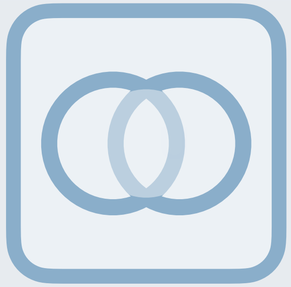



```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: string    // e.g., AB+
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}

checkReserves("A positive type")
```



Precise alternatives



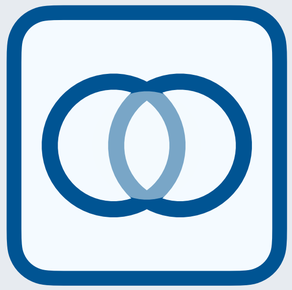




```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: string    // e.g., AB+
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}

checkReserves("A positive type")
```





Unions



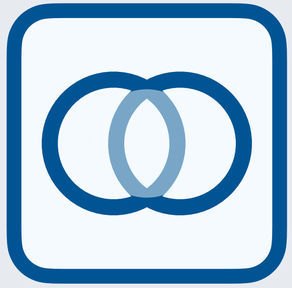



```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: string    // e.g., AB+
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}

checkReserves("A positive type")
```



```
type BloodType = 'A+'  | 'A-'  | 'B+' | 'B-'
               | 'AB+' | 'AB-' | 'O+' | 'O-'
```



Unions



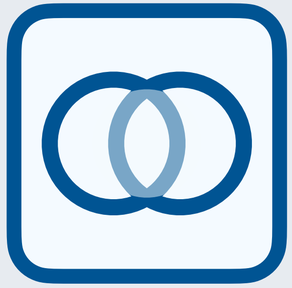
```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: BloodType // e.g., AB+
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}

checkReserves("A positive type")
```





```
type BloodType = 'A+'  | 'A-'  | 'B+' | 'B-'
               | 'AB+' | 'AB-' | 'O+' | 'O-'
```



Unions



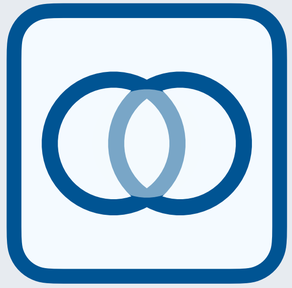
```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: BloodType // e.g., AB+
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}

checkReserves("A positive type")
```





```
type BloodType = 'A+'  | 'A-'  | 'B+' | 'B-'
               | 'AB+' | 'AB-' | 'O+' | 'O-'
```



Unions





```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: BloodType // e.g., AB+
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}

checkReserves("A+")
```

```
type BloodType = 'A+'  | 'A-'  | 'B+' | 'B-'
               | 'AB+' | 'AB-' | 'O+' | 'O-'
```

```
type BloodType = 'A+'  | 'A-'  | 'B+' | 'B-'
               | 'AB+' | 'AB-' | 'O+' | 'O-'
```

Are illegal states unrepresentable?

```
type BloodType = 'A+'  | 'A-'  | 'B+' | 'B-'
               | 'AB+' | 'AB-' | 'O+' | 'O-'
```

Are illegal states unrepresentable? Yes!

```
type BloodType = 'A+'  | 'A-'  | 'B+' | 'B-'
               | 'AB+' | 'AB-' | 'O+' | 'O-'
```

Are illegal states unrepresentable? Yes!

Did we encode the domain?

```
type BloodType = 'A+'  | 'A-'  | 'B+' | 'B-'
               | 'AB+' | 'AB-' | 'O+' | 'O-'
```

Are illegal states unrepresentable? Yes!

Did we encode the domain? Not quite...

```
type BloodType = 'A+'  | 'A-'  | 'B+' | 'B-'
               | 'AB+' | 'AB-' | 'O+' | 'O-'
```

Are illegal states unrepresentable? Yes!

Did we encode the domain? Not quite...

What's that plus sign?

```
type BloodType = 'A+'  | 'A-'  | 'B+' | 'B-'
               | 'AB+' | 'AB-' | 'O+' | 'O-'
```

Are illegal states unrepresentable? Yes!

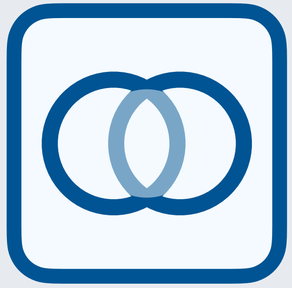
Did we encode the domain? Not quite...

What's that plus sign?

Why is there no AO or BO?



```
type BloodType = 'A+'  | 'A-'  | 'B+' | 'B-'
               | 'AB+' | 'AB-' | 'O+' | 'O-'
```

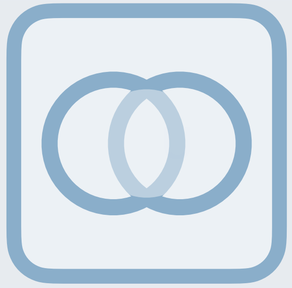


Unions









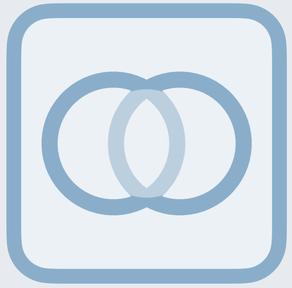


New types




```
type BloodType = 'A+'  | 'A-'  | 'B+' | 'B-'
               | 'AB+' | 'AB-' | 'O+' | 'O-'
```





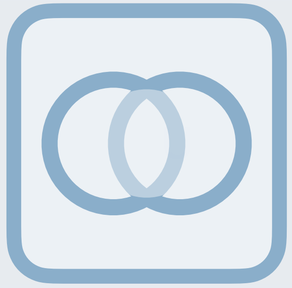


New types



```
type BloodType = {
    aboGroup: AboGroup
    rhFactor: RhFactor
}
```







New types



```
type BloodType = {
    aboGroup: AboGroup
    rhFactor: RhFactor
}


type AboGroup =
  'A' | 'B' | 'AB' | 'O'

type RhFactor =
  'positive' | 'negative'
```

```ts
type Donor = {
  id: string                    // UUID
  lastDonationAt: Date
  bloodType: BloodType          // e.g., AB+
  contactPhone: string          // with calling code, no spaces
  contactEmail: string
  medicalRecordsDir: string // file path
}
```

```ts
type BloodType = {
    aboGroup: AboGroup
    rhFactor: RhFactor
}
```

```ts
type AboGroup =
  'A' | 'B' | 'AB' | 'O'

type RhFactor  =
  ‘positive' | 'negative'
```

```ts
type Donor = {
  id: string                    // UUID
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: string          // with calling code, no spaces
  contactEmail: string
  medicalRecordsDir: string // file path
}
```

```ts
type BloodType = {
    aboGroup: AboGroup
    rhFactor: RhFactor
}
```

```ts
type AboGroup =
  'A' | 'B' | 'AB' | 'O'

type RhFactor  =
  ‘positive' | 'negative'
```

```ts
type Donor = {
  id: string                    // UUID
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: string          // with calling code, no spaces
  contactEmail: string
  medicalRecordsDir: string // file path
}

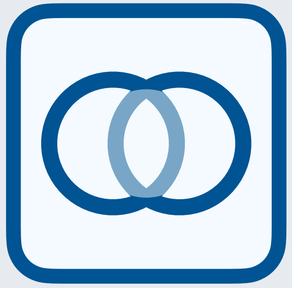
type BloodType = {
    aboGroup: AboGroup
    rhFactor: RhFactor
}

type AboGroup =
  'A' | 'B' | 'AB' | 'O'

type RhFactor  =
  ‘positive' | 'negative'
```

```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}
```





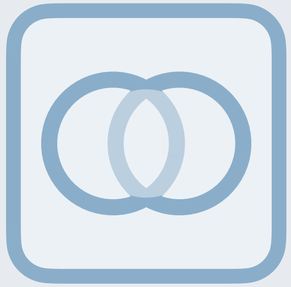




```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}
```



Precise alternatives

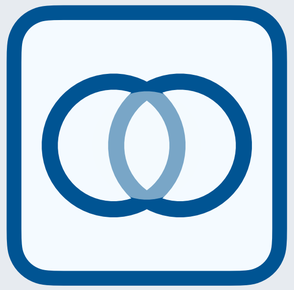






```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}
```





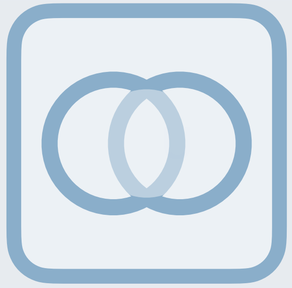
Unions






```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}
```



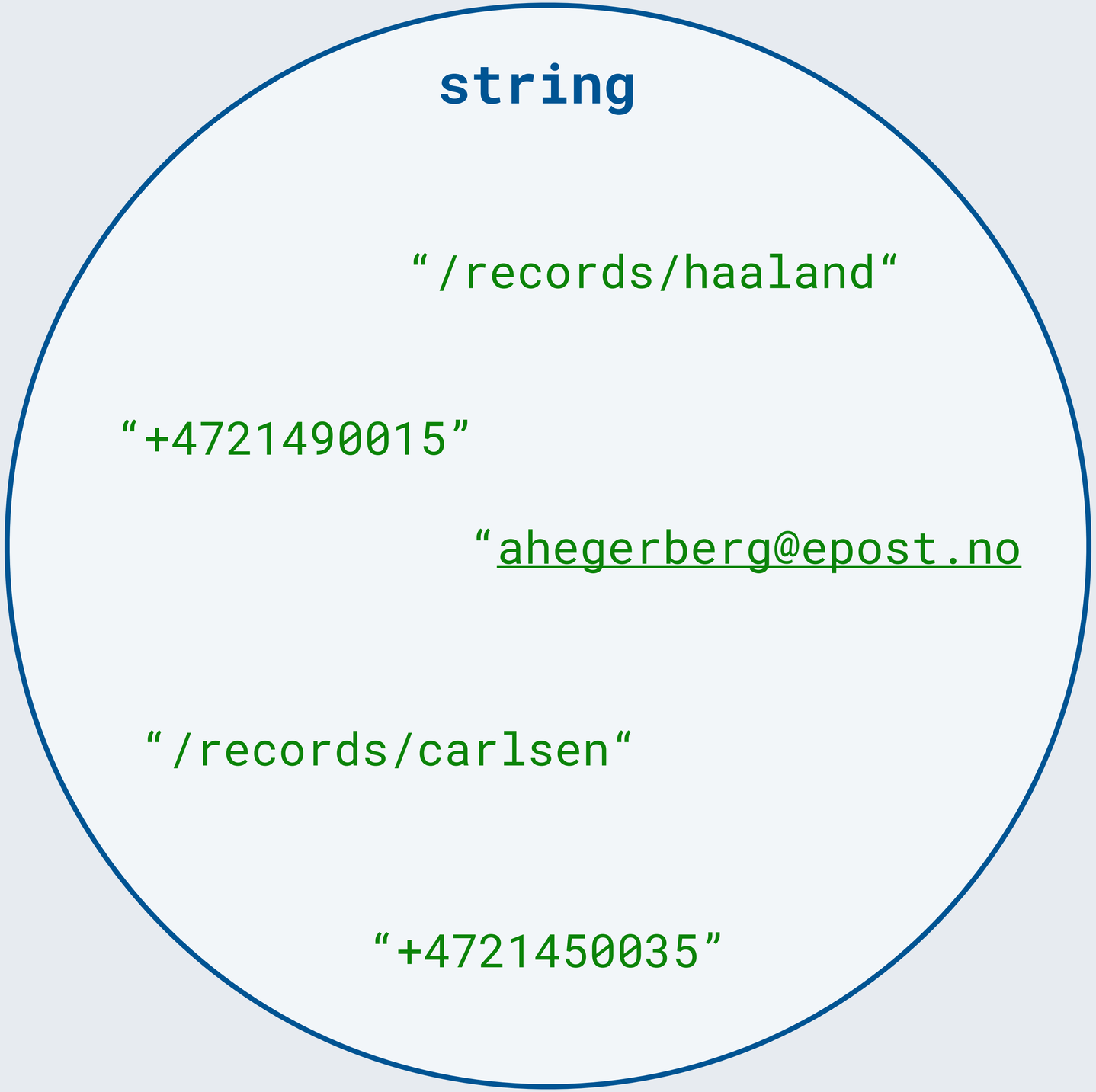




New types



```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}
```


string
“/records/haaland“
“+4721490015”
“ahegerberg@epost.no
“/records/carlsen“
“+4721450035”

```
type Donor = {
  // ...
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}
```

```
type Donor = {
  // ...
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}


function askForDonation(donor: Donor) {



}
```

```
type Donor = {
  // ...
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}



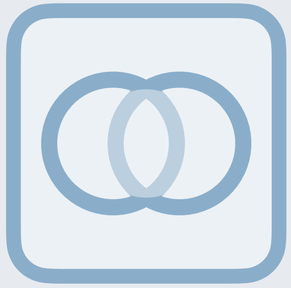
function askForDonation(donor: Donor) {
    askViaEmail(donor.contactPhone)
    askViaSMS(donor.contactEmail)
}
```

```
type Donor = {
  // ...
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}



function askForDonation(donor: Donor) {
    askViaEmail(donor.contactPhone)
    askViaSMS(donor.contactEmail)
}
```



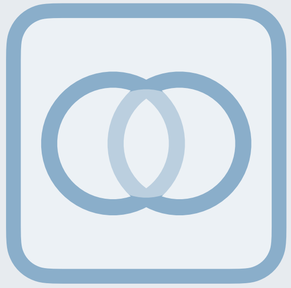






```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}
```









```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}
```

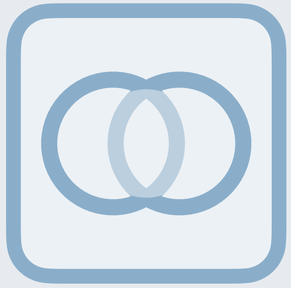






Branded types









Brands

```
type Donor = {
  id: string
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: string
  contactEmail: string
  medicalRecordsDir: string
}
```

string

“ahegerberg@epost.no

“+4721450035”

“/records/haaland“

“+4721490015”

“/records/carlsen“

```
type FilePath = string

type EmailAddress = string

type PhoneNumber = string
```

string

“ahegerberg@epost.no

“+4721450035”

“/records/haaland“

“+4721490015”

“/records/carlsen“

```
type FilePath = string & { _brand: "FilePath" }

type EmailAddress = string & { _brand: "Email" }

type PhoneNumber = string & { _brand: "PhoneNumber" }
```




```
type FilePath = string & { _brand: "FilePath" }

type EmailAddress = string & { _brand: "Email" }

type PhoneNumber = string & { _brand: "PhoneNumber" }
```

```ts
type FilePath = string & { _brand: "FilePath" }

type EmailAddress = string & { _brand: "Email" }

type PhoneNumber = string & { _brand: "PhoneNumber" }
```

```ts
type Branded<T, B> = T & { _brand: B }


type FilePath = string & { _brand: "FilePath" }

type EmailAddress = string & { _brand: "Email" }

type PhoneNumber = string & { _brand: "PhoneNumber" }
```

```ts
type Branded<T, B> = T & { _brand: B }


type FilePath = Branded<string, "FilePath">

type EmailAddress = Branded<string, "Email">

type PhoneNumber = Branded<string, "PhoneNumber">
```

```
type EmailAddress = string


function sendEmail(to: EmailAddress) {
    // ...
}



sendEmail(“filip@wasp.sh”)

sendEmail(“@@@@@”)

sendEmail(“I don’t know man”)
```

```ts
type EmailAddress = Branded<string, "Email">


function sendEmail(to: EmailAddress) {
    // ...
}
```

```ts
sendEmail(“filip@wasp.sh”)

sendEmail(“@@@@@”)

sendEmail(“I don’t know man”)
```

```ts
type EmailAddress = Branded<string, "Email">


function sendEmail(to: EmailAddress) {
    // ...
}



sendEmail(“filip@wasp.sh”)

sendEmail(“@@@@@”)

sendEmail(“I don’t know man”)
```

```ts
type EmailAddress = Branded<string, "Email">


function sendEmail(to: EmailAddress) {
    // ...
}



sendEmail(“filip@wasp.sh”)

sendEmail(“@@@@@”)

sendEmail(“I don’t know man”)
```

```ts
type EmailAddress = Branded<string, "Email">
```

```ts
function sendEmail(to: EmailAddress) {
    // ...
}
```

```ts
type EmailAddress = Branded<string, "Email">


function parseEmailAddress(emailInput: string): EmailAddress | null {
    // ...
    return parseOk ? parseResult as EmailAddress : null
}




function sendEmail(to: EmailAddress) {
    // ...
}
```

```
type EmailAddress = Branded<string, "Email">


function parseEmailAddress(emailInput: string): EmailAddress | null {
    // ...
    return parseOk ? parseResult as EmailAddress : null
}

function getEmailForUser(donorId: Donor[“id”]): EmailAddress {
    // ...
}


function sendEmail(to: EmailAddress) {
    // ...
}
```

```
type EmailAddress = Branded<string, "Email">


function parseEmailAddress(emailInput: string): EmailAddress | null {
    // ...
    return parseOk ? parseResult as EmailAddress : null
}

function getEmailForUser(donorId: Donor["id"]): EmailAddress {
    // ...
}


function sendEmail(to: EmailAddress) {
    // ...
}
```

Smart constructors

```
type FilePath = Branded<string, “FilePath">

type EmailAddress = Branded<string, "Email">

type PhoneNumber = Branded<string, "PhoneNumber">
```

```
type FilePath = Branded<string, “FilePath">

        parseFilePath(path: string): FilePath | null

type EmailAddress = Branded<string, "Email">

type PhoneNumber = Branded<string, "PhoneNumber">
```

```
type FilePath = Branded<string, “FilePath">
            parseFilePath(path: string): FilePath | null
            getDonorDirectory(donorId: string): FilePath

type EmailAddress = Branded<string, "Email">

type PhoneNumber = Branded<string, "PhoneNumber">
```

```
type FilePath = Branded<string, “FilePath">
        parseFilePath(path: string): FilePath | null
        getDonorDirectory(donorId: string): FilePath

type EmailAddress = Branded<string, "Email">
        parseEmail(possibleEmail: string): EmailAddress | null
        getDonorContactEmail(donorId: string): EmailAddress

type PhoneNumber = Branded<string, "PhoneNumber">
```

```
type FilePath = Branded<string, “FilePath">
        parseFilePath(path: string): FilePath | null
        getDonorDirectory(donorId: string): FilePath

type EmailAddress = Branded<string, "Email">
        parseEmail(possibleEmail: string): EmailAddress | null
        getDonorContactEmail(donorId: string): EmailAddress

type PhoneNumber = Branded<string, "PhoneNumber">
        parsePhoneNumber(numberString: string): PhoneNumber | null
        getDonorContactPhone(donorId: string): PhoneNumber
```

```
type FilePath = Branded<string, “FilePath">

type EmailAddress = Branded<string, "Email">

type PhoneNumber = Branded<string, "PhoneNumber">
```

```
type Donor = {
  id: string                    // UUID
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: PhoneNumber   // with calling code, no spaces
  contactEmail: EmailAddress
  medicalRecordsDir: FilePath // file path
}
```

```
type FilePath =
  Branded<string, “FilePath">

type EmailAddress =
  Branded<string, "Email">

type PhoneNumber =
  Branded<string, "PhoneNumber">
```

```
type BloodType = {
    aboGroup: AboGroup
    rhFactor: RhFactor
}

type AboGroup =
  ‘A' | 'B' | 'AB' | 'O'

type RhFactor =
  ‘positive' | 'negative'
```

```ts
type Donor = {
  id: string                    // UUID
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: PhoneNumber
  contactEmail: EmailAddress
  medicalRecordsDir: FilePath
}
```

```ts
type FilePath =
  Branded<string, “FilePath">

type EmailAddress =
  Branded<string, "Email">

type PhoneNumber =
  Branded<string, "PhoneNumber">
```

```ts
type BloodType = {
    aboGroup: AboGroup
    rhFactor: RhFactor
}

type AboGroup =
  ‘A' | 'B' | 'AB' | 'O'

type RhFactor =
  ‘positive' | 'negative'
```

```
type Donor = {
  id: string                    // UUID
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: PhoneNumber
  contactEmail: EmailAddress
  medicalRecordsDir: FilePath
}
```

```
type FilePath =
  Branded<string, "FilePath">

type EmailAddress =
  Branded<string, "Email">

type PhoneNumber =
  Branded<string, "PhoneNumber">
```

```
type BloodType = {
    aboGroup: AboGroup
    rhFactor: RhFactor
}

type AboGroup =
  ‘A' | 'B' | 'AB' | 'O'

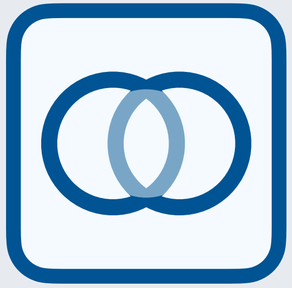
type RhFactor =
  ‘positive' | 'negative'
```

```
type Donor = {
  id: string   // UUID
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: PhoneNumber
  contactEmail: EmailAddress
  medicalRecordsDir: FilePath
}
```

```
type Donor = {
  id: string   // UUID
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: PhoneNumber
  contactEmail: EmailAddress
  medicalRecordsDir: FilePath
}
```

“dc02ccfe-ba4c-486a-995d-12b2beeca767"







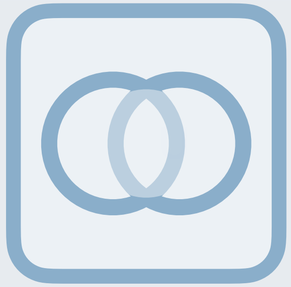


```
type Donor = {
  id: string   // UUID
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: PhoneNumber
  contactEmail: EmailAddress
  medicalRecordsDir: FilePath
}
```

“dc02ccfe-ba4c-486a-995d-12b2beeca767"



Precise alternatives



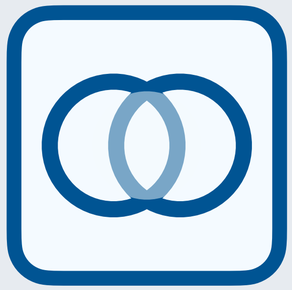




```
type Donor = {
  id: string   // UUID
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: PhoneNumber
  contactEmail: EmailAddress
  medicalRecordsDir: FilePath
}
```

“dc02ccfe-ba4c-486a-995d-12b2beeca767"





Unions

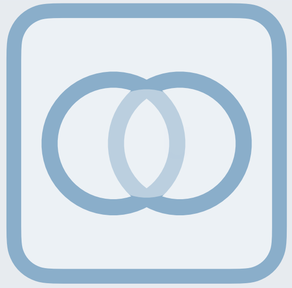




```
type Donor = {
  id: string   // UUID
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: PhoneNumber
  contactEmail: EmailAddress
  medicalRecordsDir: FilePath
}
```

“dc02ccfe-ba4c-486a-995d-12b2beeca767"







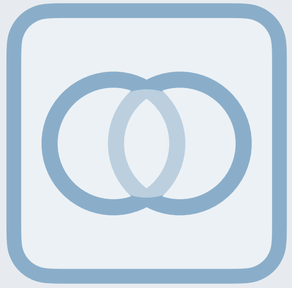
New types



```
type Donor = {
  id: string   // UUID
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: PhoneNumber
  contactEmail: EmailAddress
  medicalRecordsDir: FilePath
}
```

“dc02ccfe-ba4c-486a-995d-12b2beeca767"









Brands

```
type Donor = {
  id: string   // UUID
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: PhoneNumber
  contactEmail: EmailAddress
  medicalRecordsDir: FilePath
}
```

“dc02ccfe-ba4c-486a-995d-12b2beeca767"

```
type UUID =  Branded<string, “UUID">
```

```
type UUID = Branded<string, “UUID">
```

```
function getDonorById(id: UUID): Donor {
    // ...
}
```

```ts
type UUID = Branded<string, “UUID">

function generateUUID(): UUID {
    const uuid = crypto.randomUUID()
    return uuid as UUID
}
```

```ts
function getDonorById(id: UUID): Donor {
    // ...
}
```

```
type UUID = Branded<string, “UUID">

function generateUUID(): UUID {
    const uuid = crypto.randomUUID()
    return uuid as UUID
}

function getDonorIds(forType: BloodType): UUID[] {
    // ...
}


function getDonorById(id: UUID): Donor {
    // ...
}
```

```
type Donor = {
  id: UUID                          // UUID
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: PhoneNumber
  contactEmail: EmailAddress
  medicalRecordsDir: FilePath
}
```

```
type FilePath =
  Branded< string, "FilePath" >

type EmailAddress =
  Branded< string, "Email" >

type PhoneNumber =
  Branded< string, "PhoneNumber" >
```

```
type UUID =
  Branded<string, “UUID">
```

```
type BloodType = {
    aboGroup: AboGroup
    rhFactor: RhFactor
}

type AboGroup =
  ‘A' | 'B' | 'AB' | 'O'

type RhFactor =
  ‘positive' | 'negative'
```

```
type Donor = {
  id: UUID
  lastDonationAt: Date
  bloodType: BloodType
  contactPhone: PhoneNumber
  contactEmail: EmailAddress
  medicalRecordsDir: FilePath
}
```

```
type FilePath =
  Branded< string, "FilePath" >

type EmailAddress =
  Branded< string, "Email" >

type PhoneNumber =
  Branded< string, "PhoneNumber" >
```

```
type UUID =
  Branded<string, “UUID">
```

```
type BloodType = {
    aboGroup: AboGroup
    rhFactor: RhFactor
}

type AboGroup =
  ‘A' | 'B' | 'AB' | 'O'

type RhFactor =
  ‘positive' | 'negative'
```

What to make of this?

LLM output is not "new machine code"

Keep caring about the language

Pick one with a great type system

And learn it well

1. Start with the types

2. Encode the domain

3. Make illegal states unrepresentable



Precise alternatives



Unions

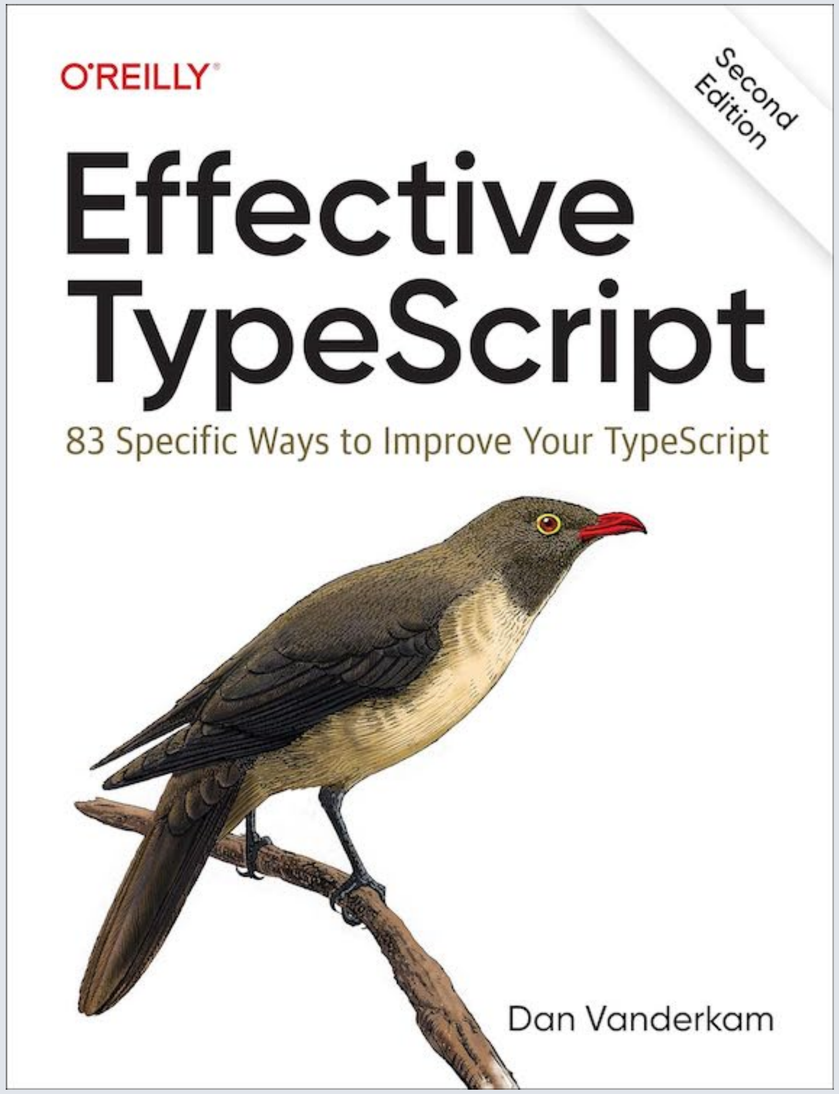


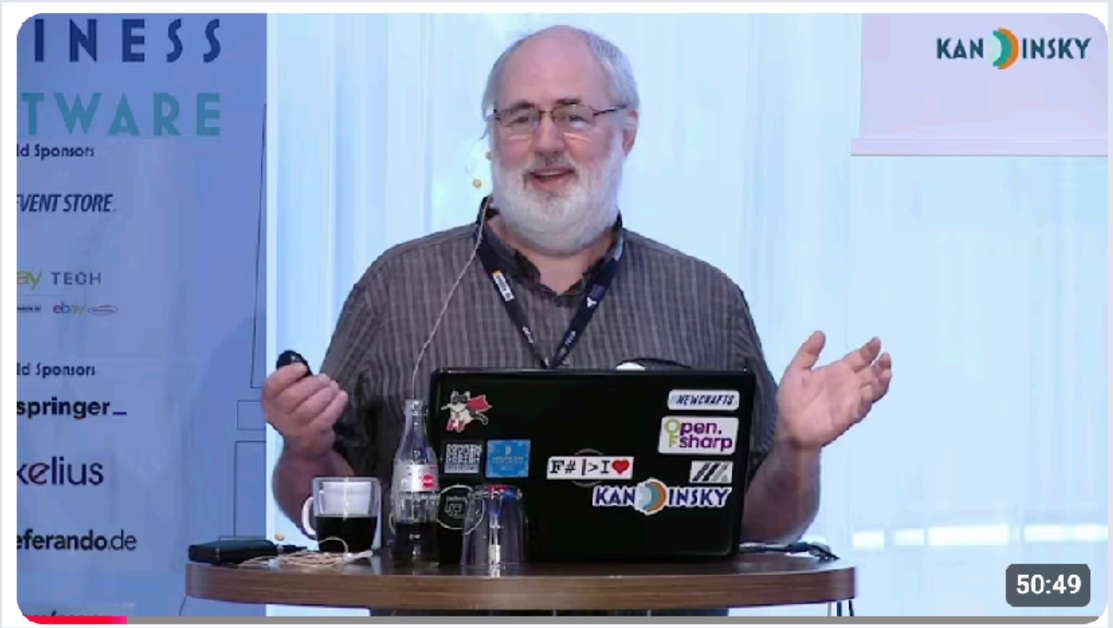
New types



Brands







Parse, don't validate

Effective ML

Types Are Worth The Typing

Types Are 

Worth The Typing

Types Are Still Worth The Typing

(Sometimes)

Feedback



Types Are Worth the Typing

Resources

