Click

Click

Click

Click

Click

Click

Click

Click

Click

Click

Keep Choosing TypeScript

LLM output is not "new machine code"

LLM output is not "new machine code"

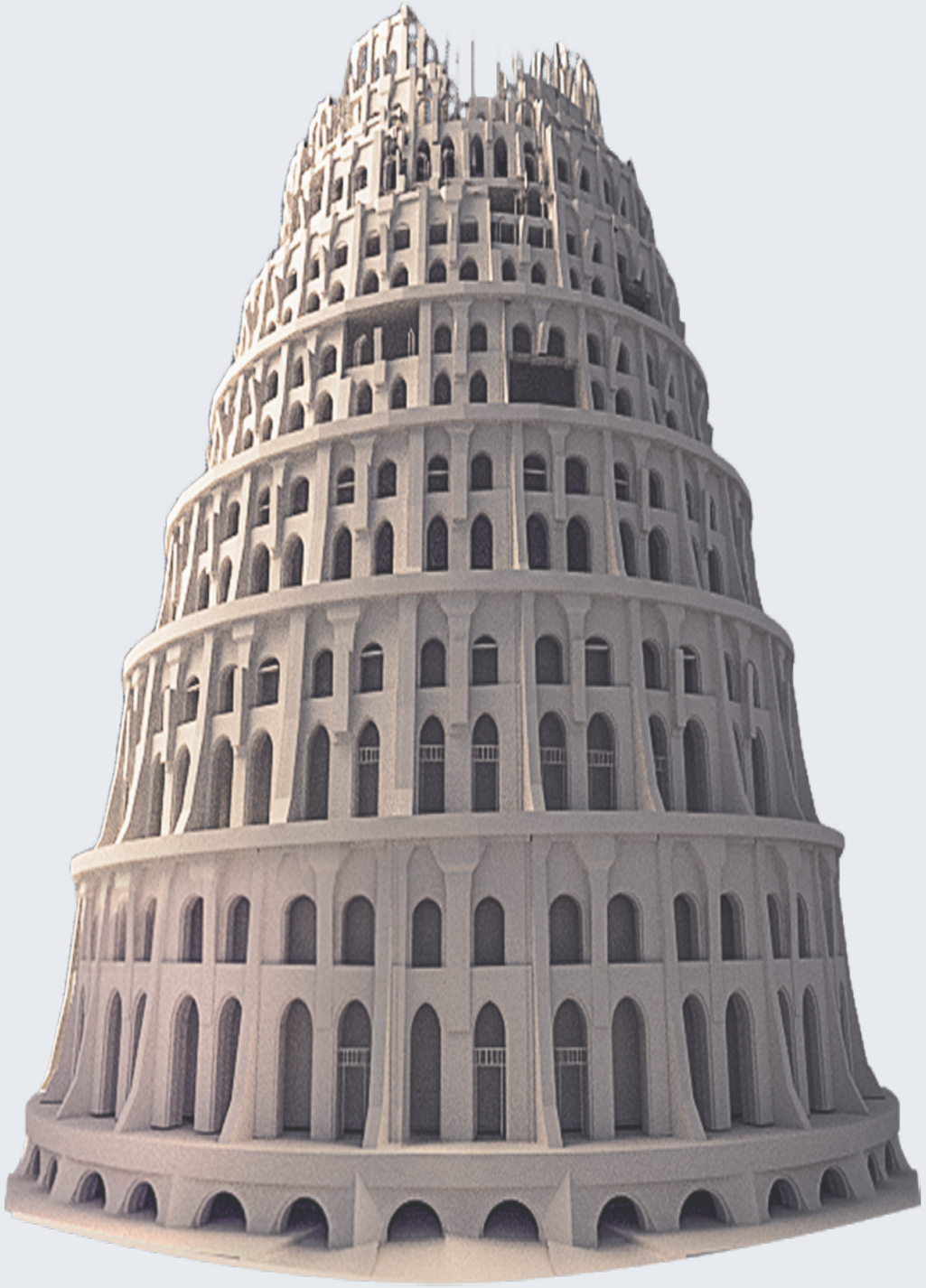
You should learn a language well

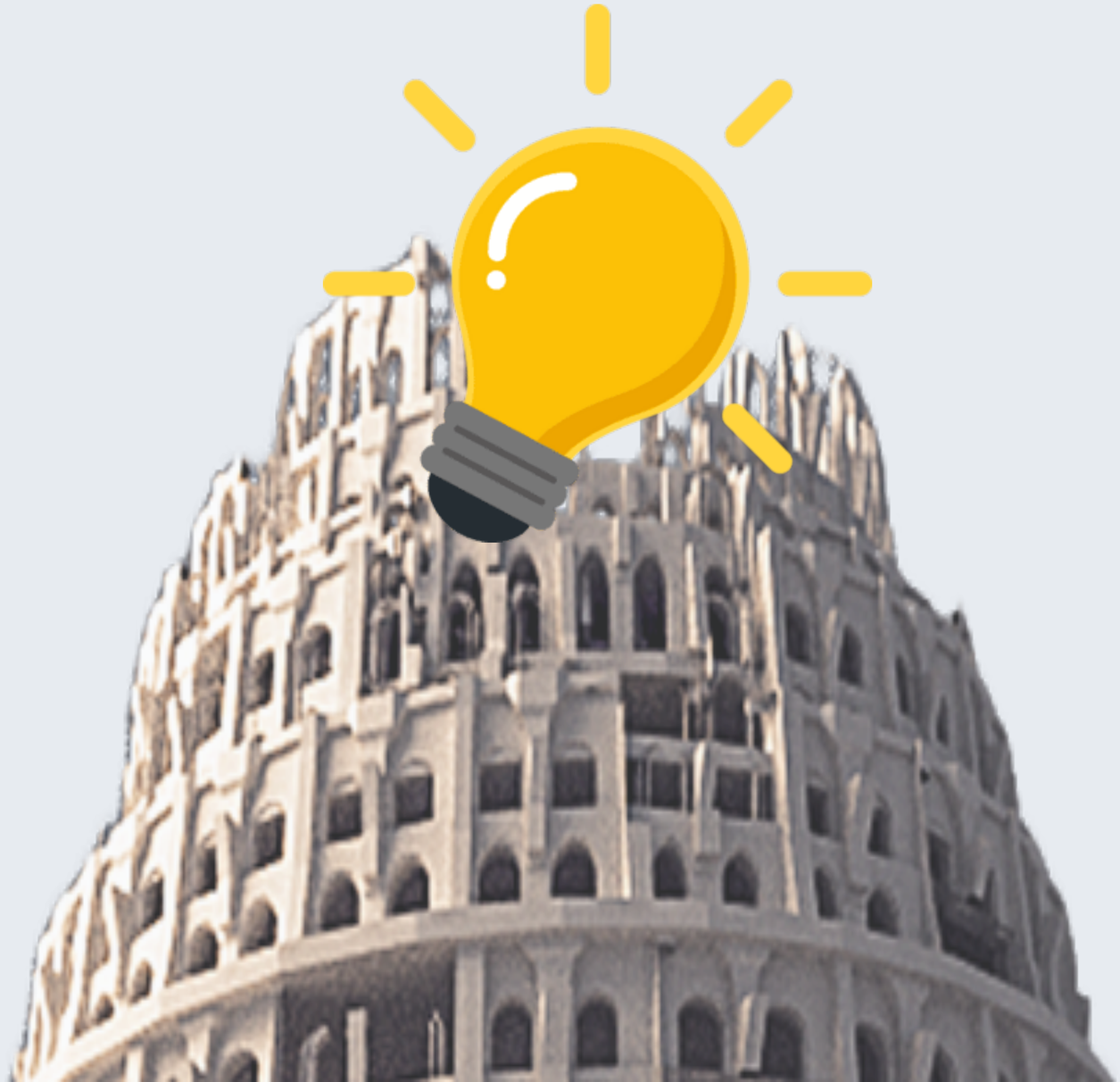
LLM output is not "new machine code"

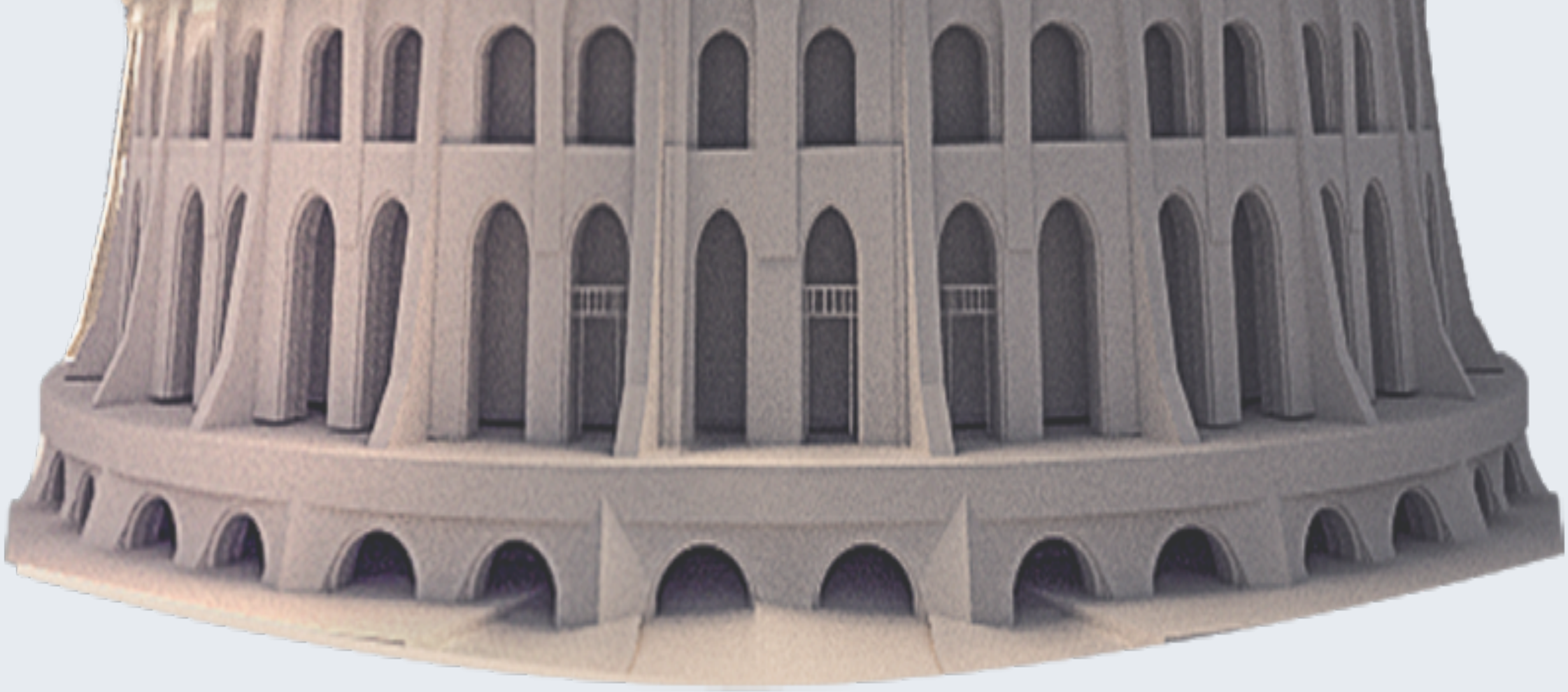
You should learn a language well

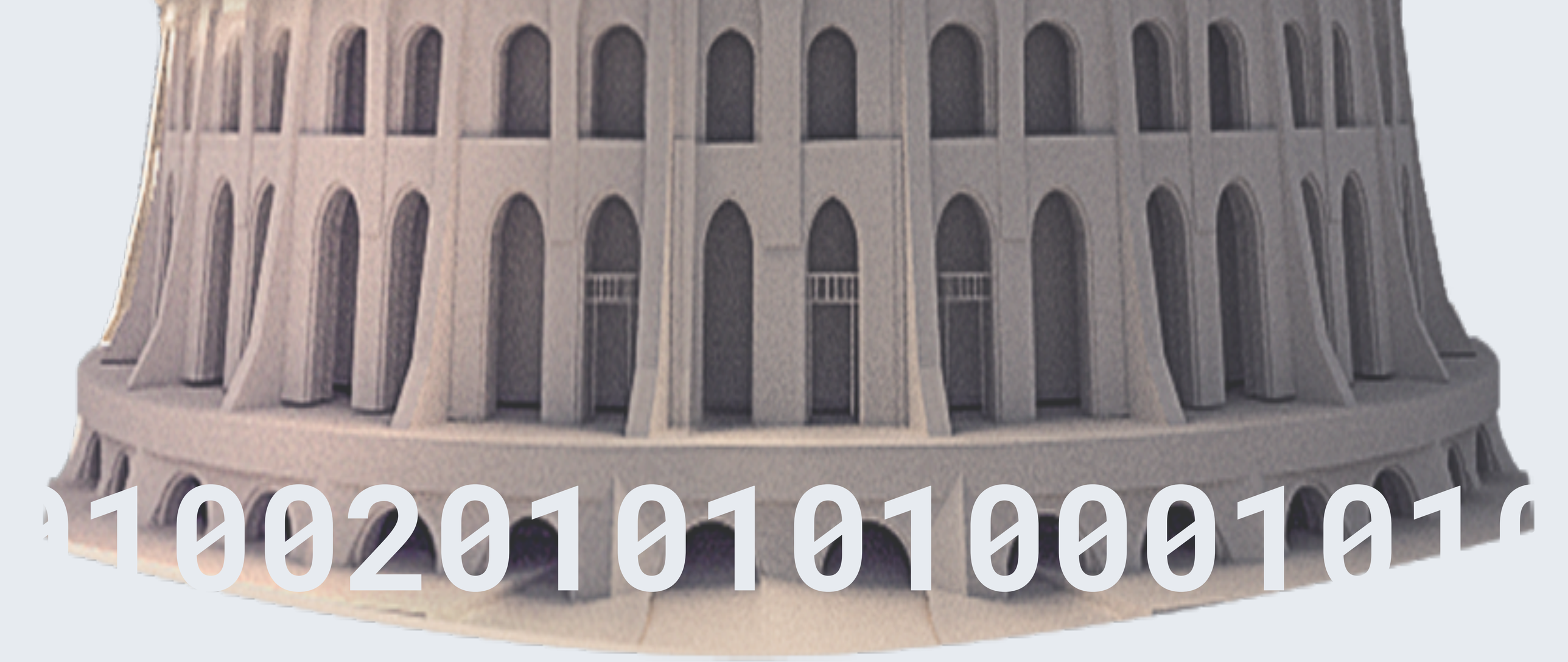
TypeScript is the perfect choice!

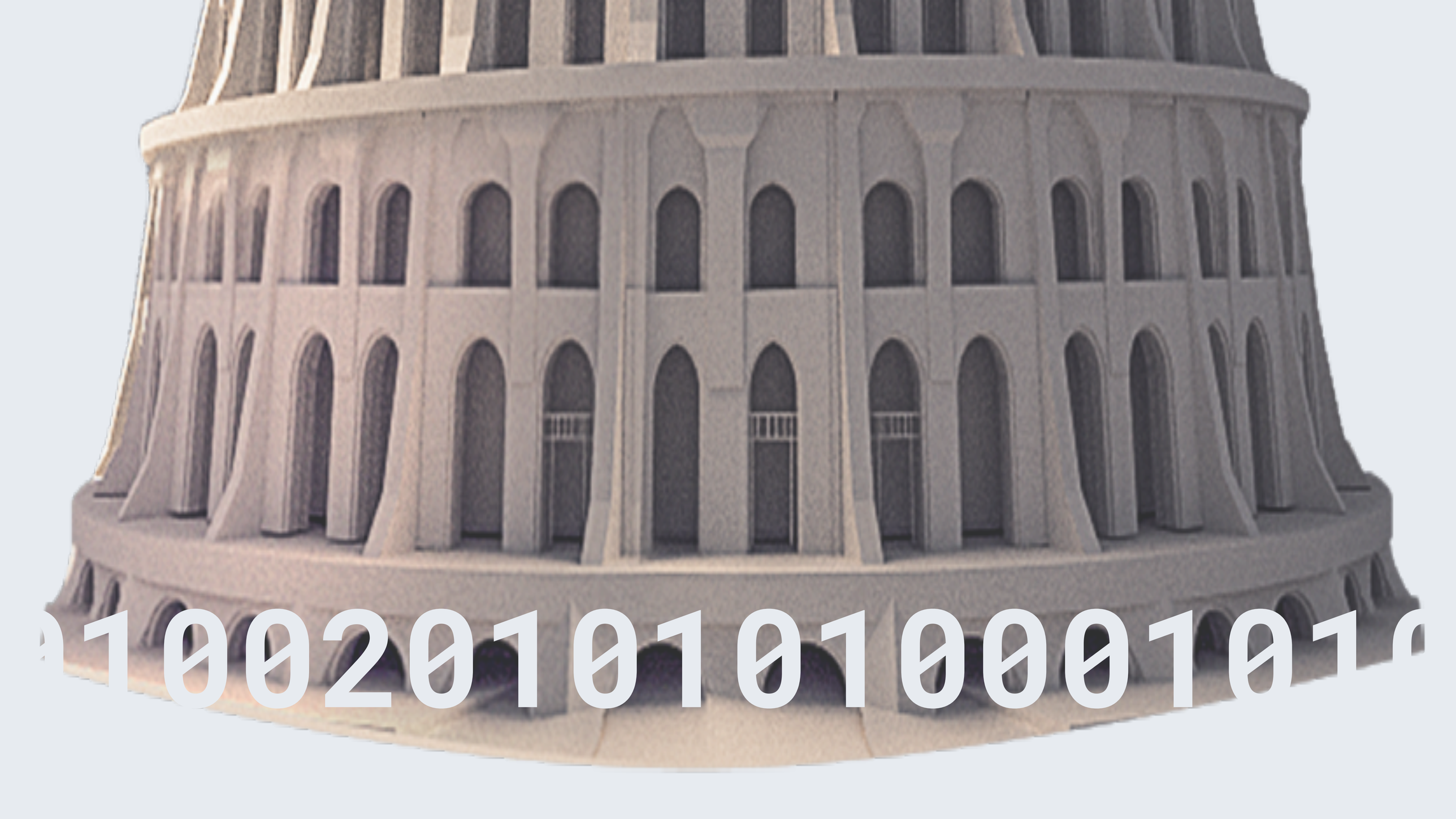
Why care about the language?







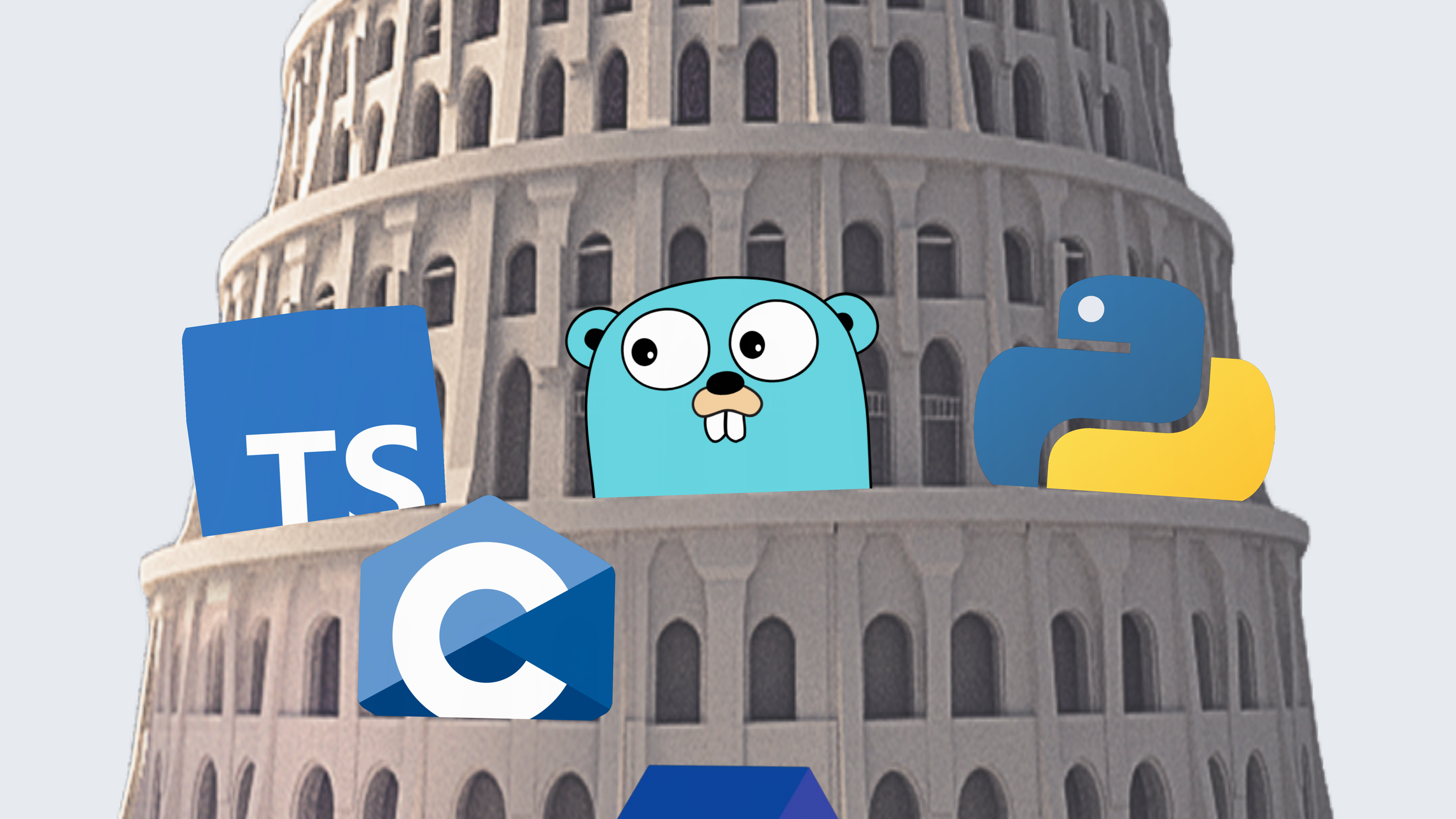

1002010101000101

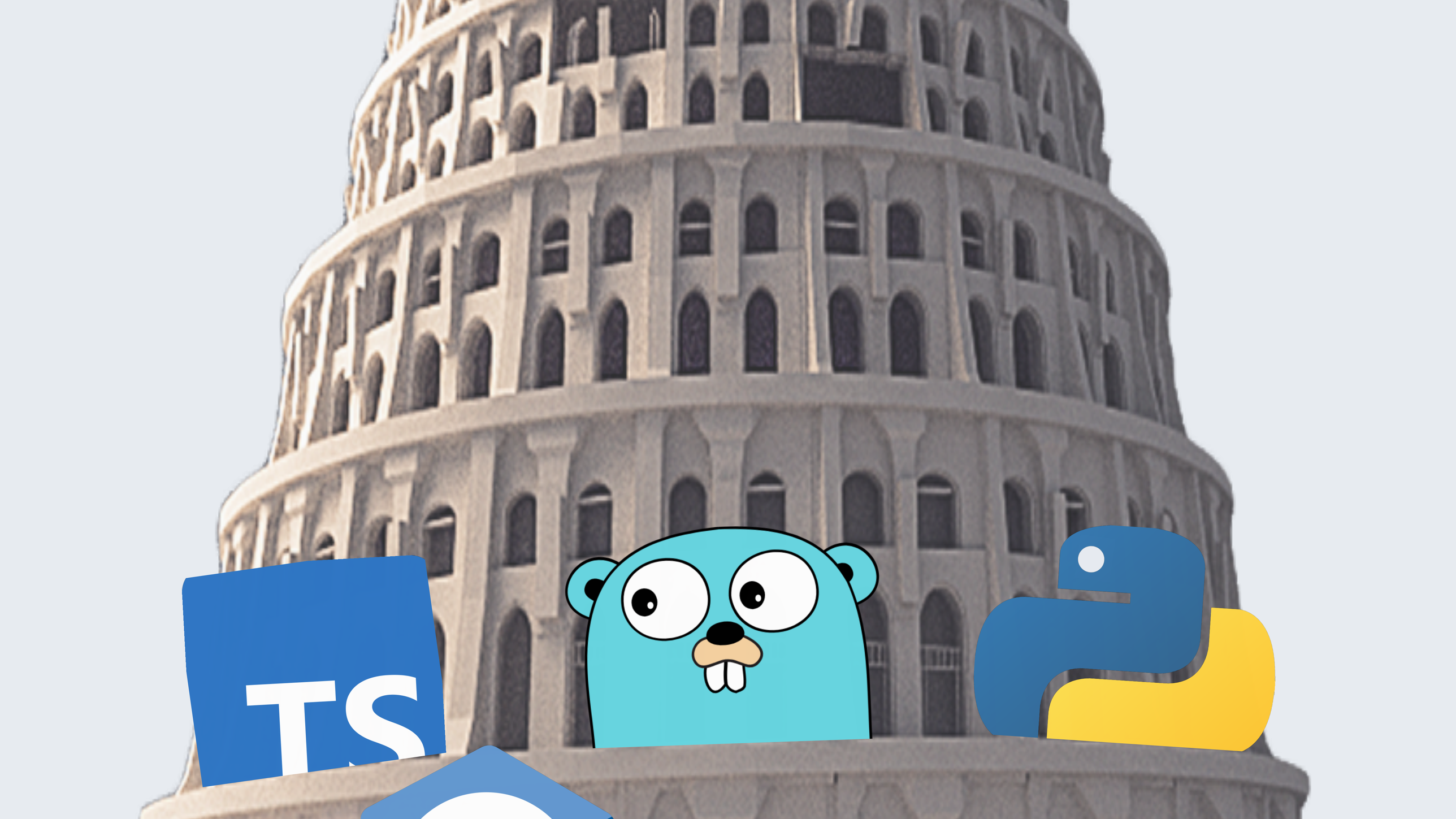

1002010101000101


ASM
1002010101000101


ASM


ASM

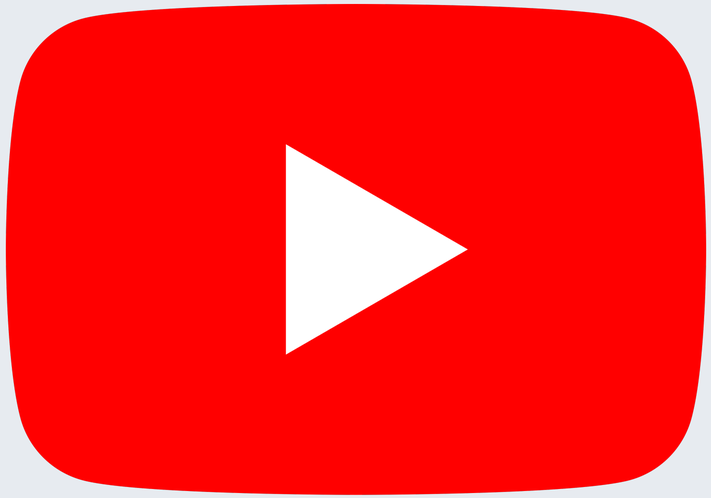

TS
C

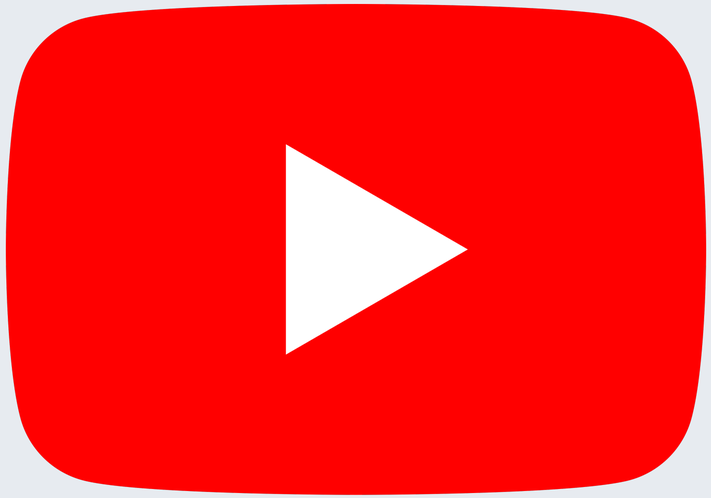
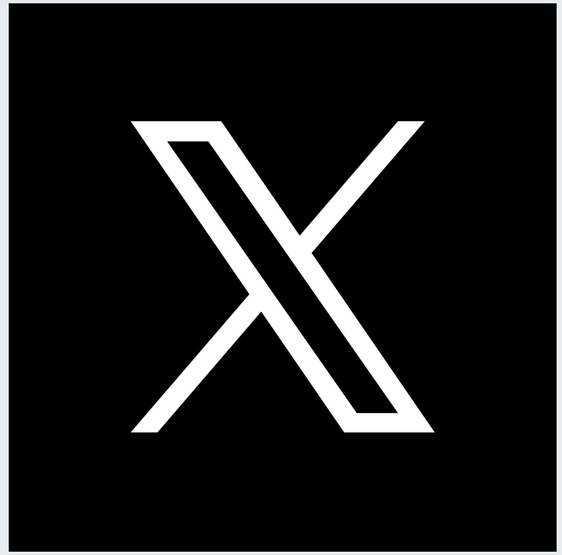

TS

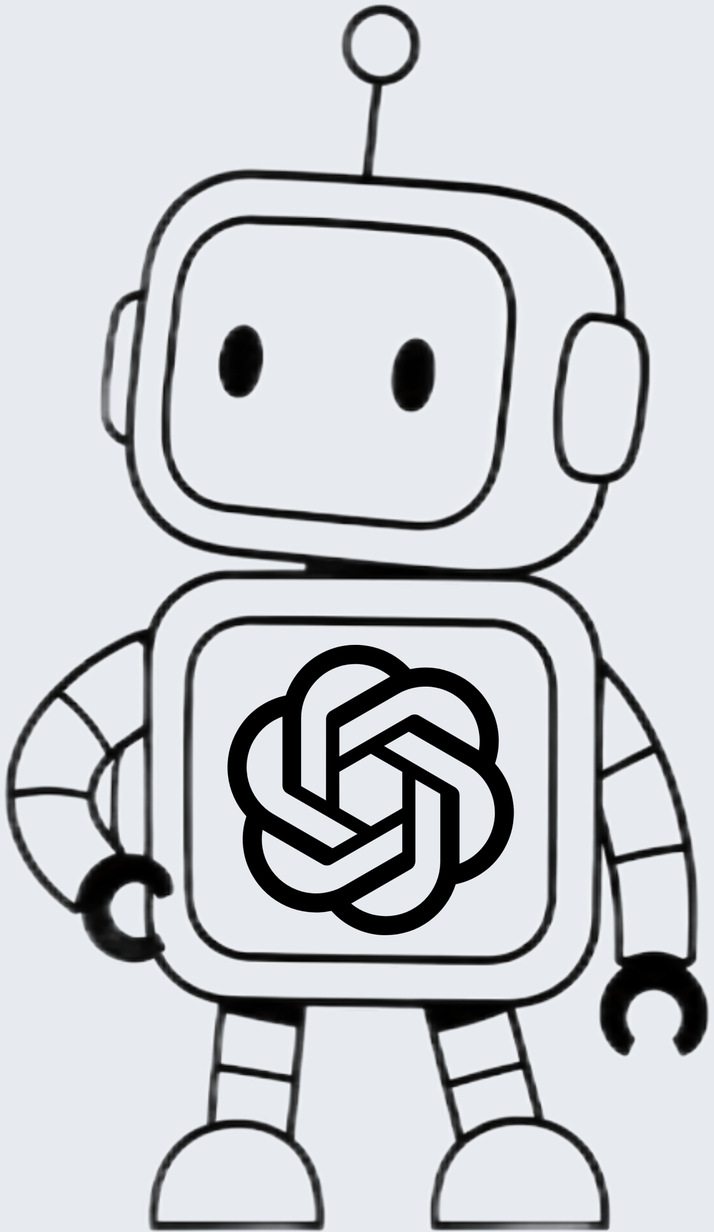

TS

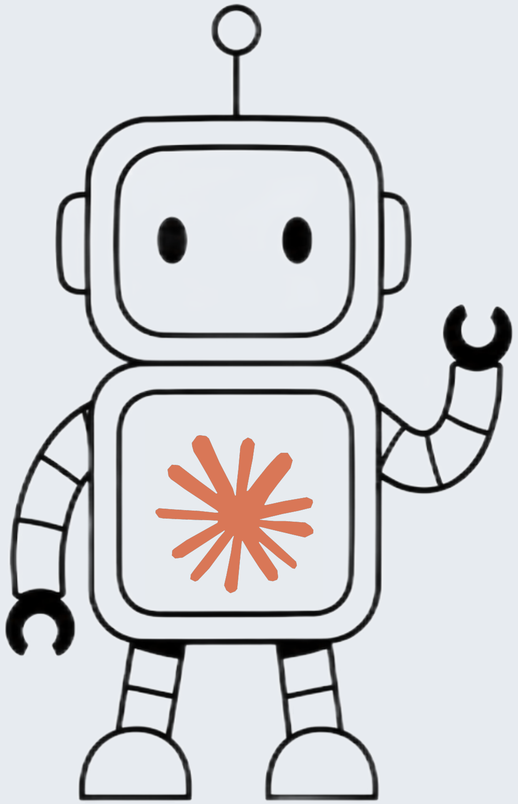

TS
ASM
01002010101000101010

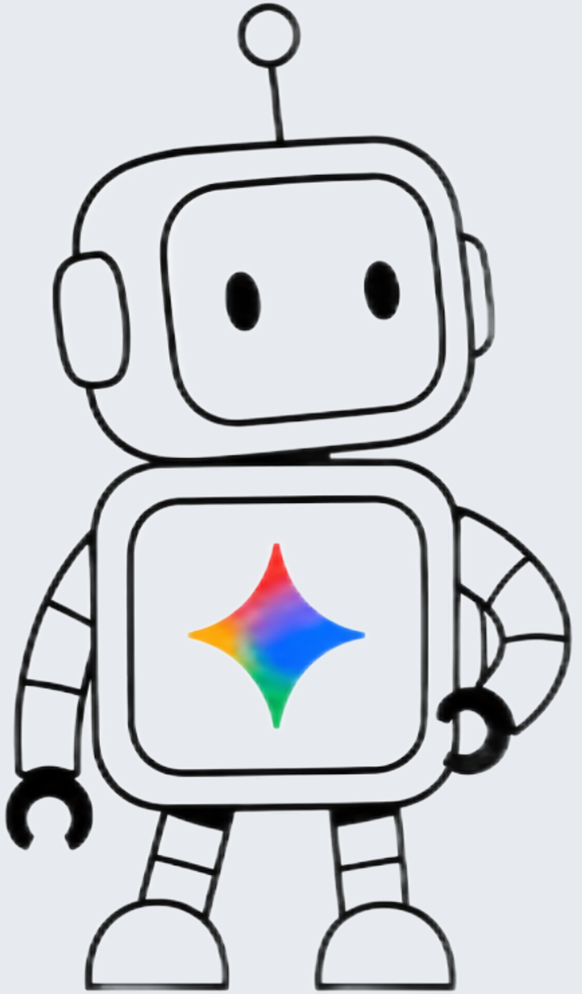

TS
C



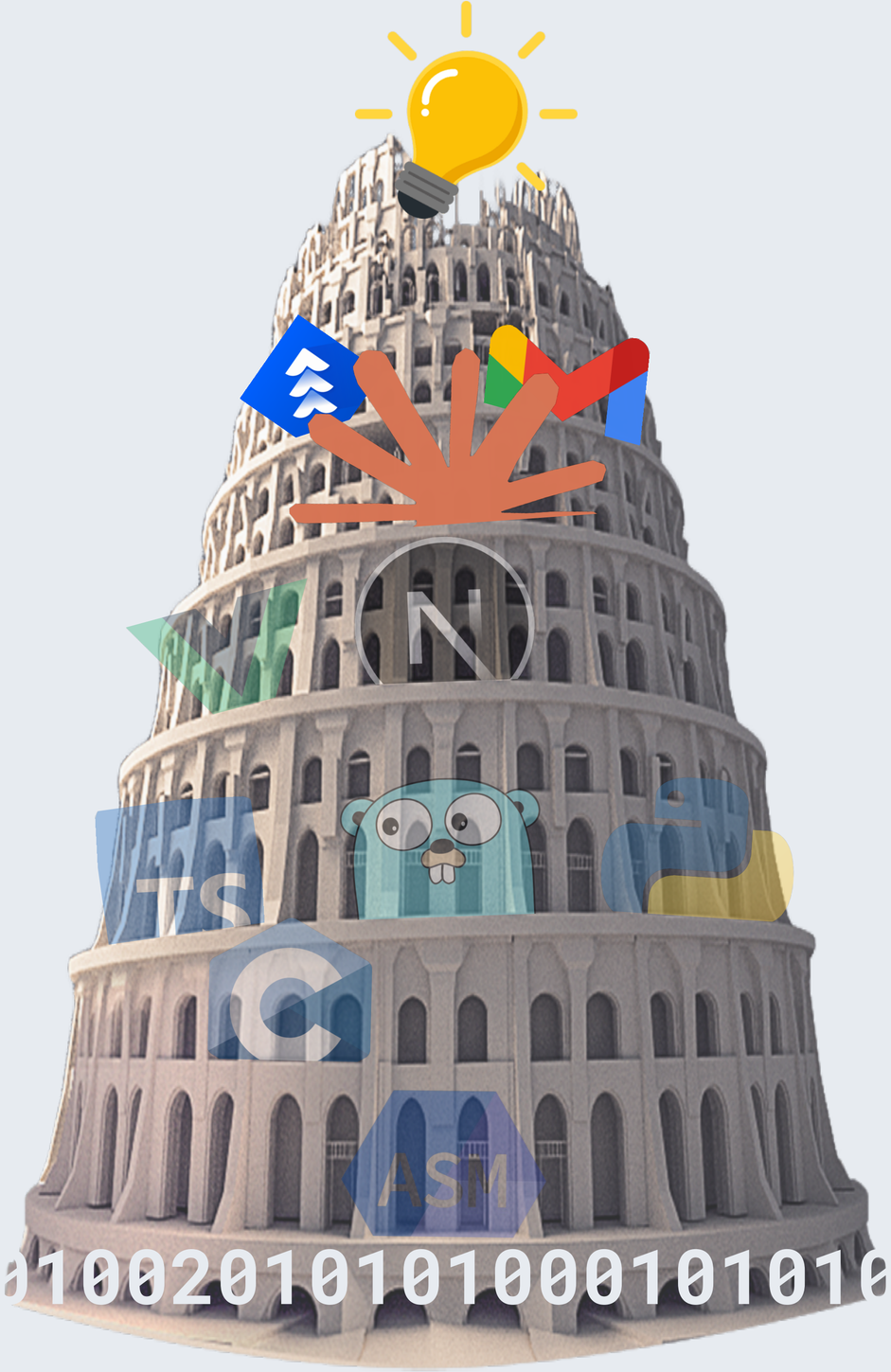


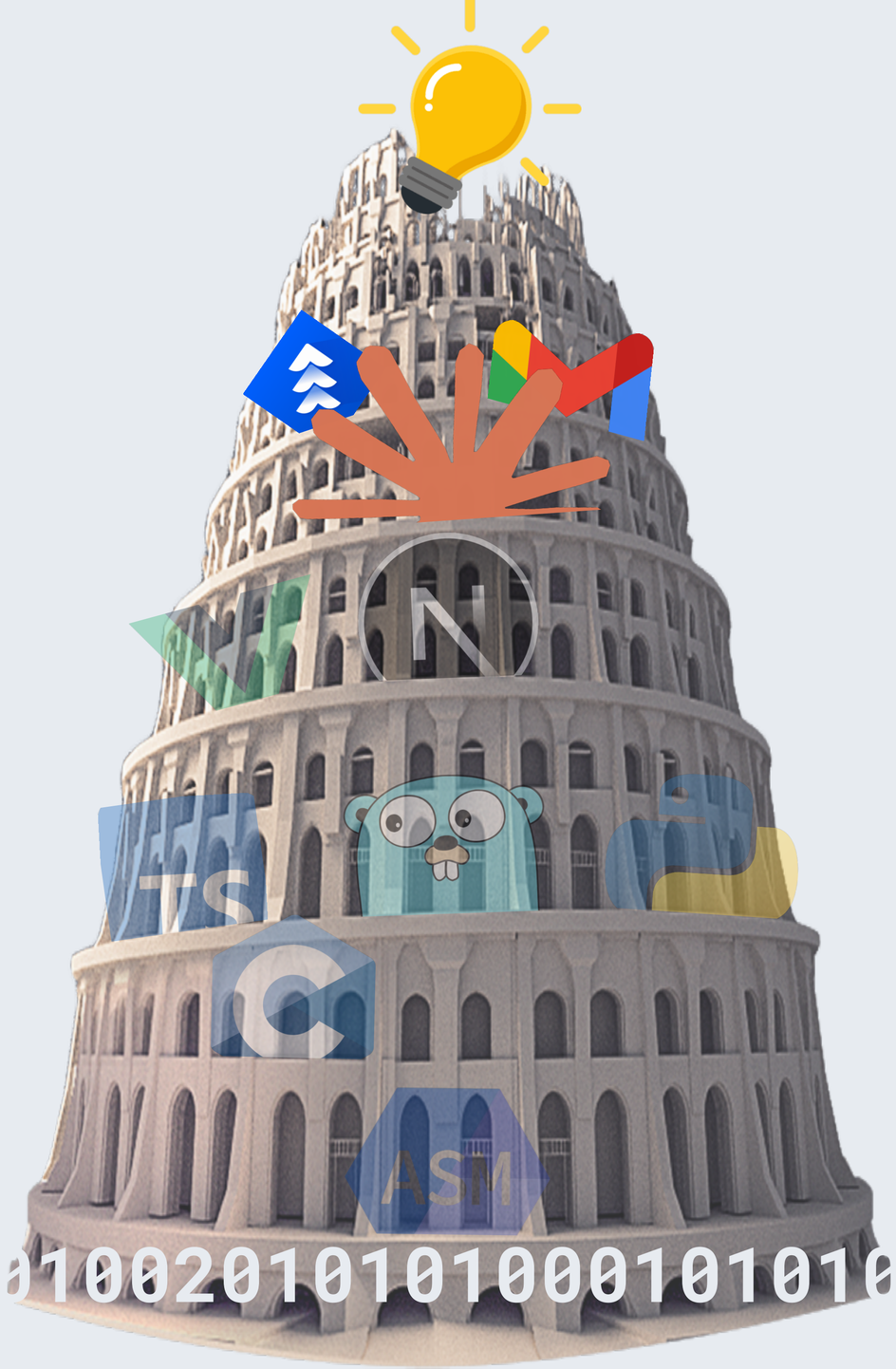


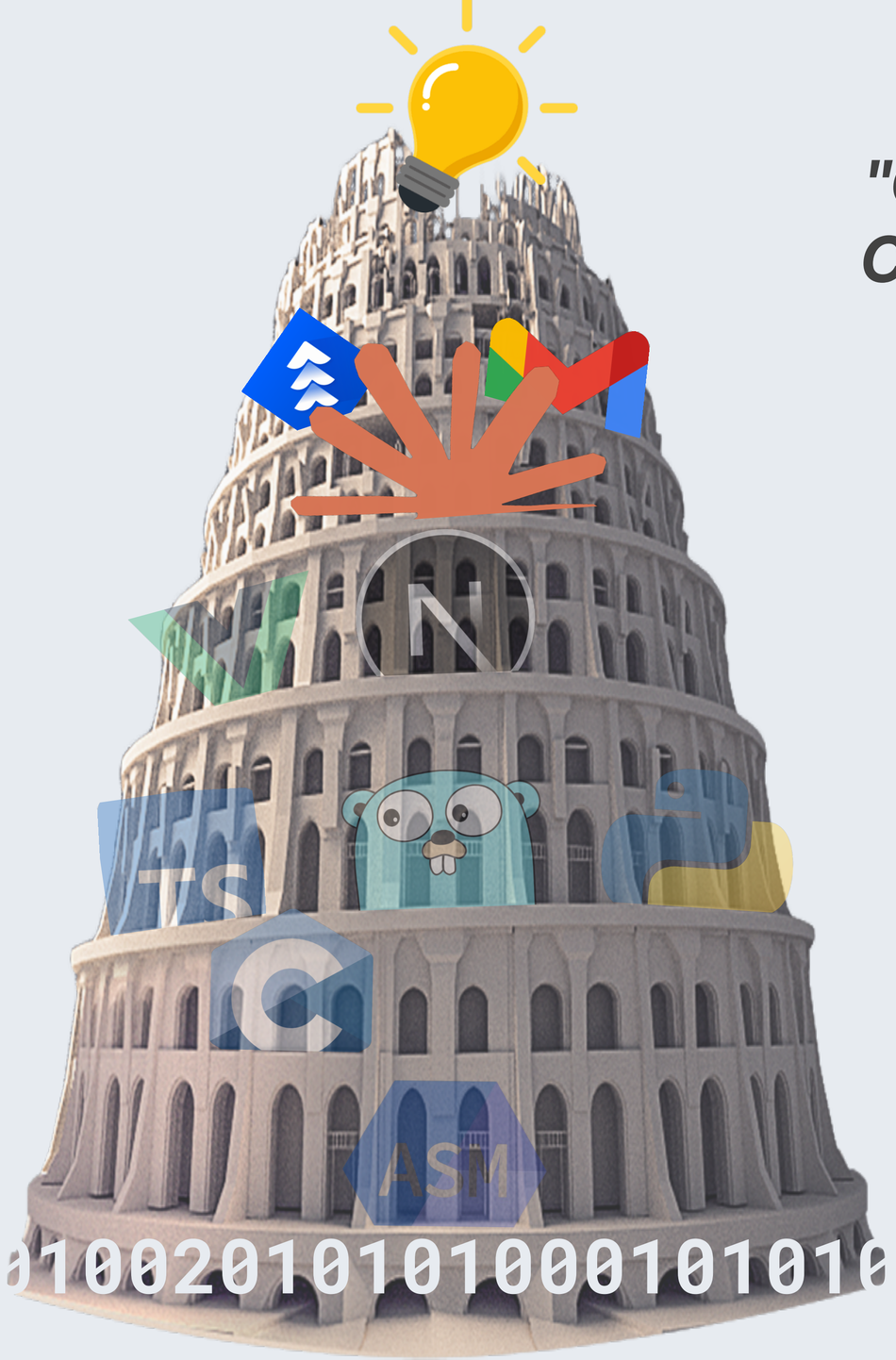



TS
ASM
01002010101000101010


TS
ASM
0100201010100010101

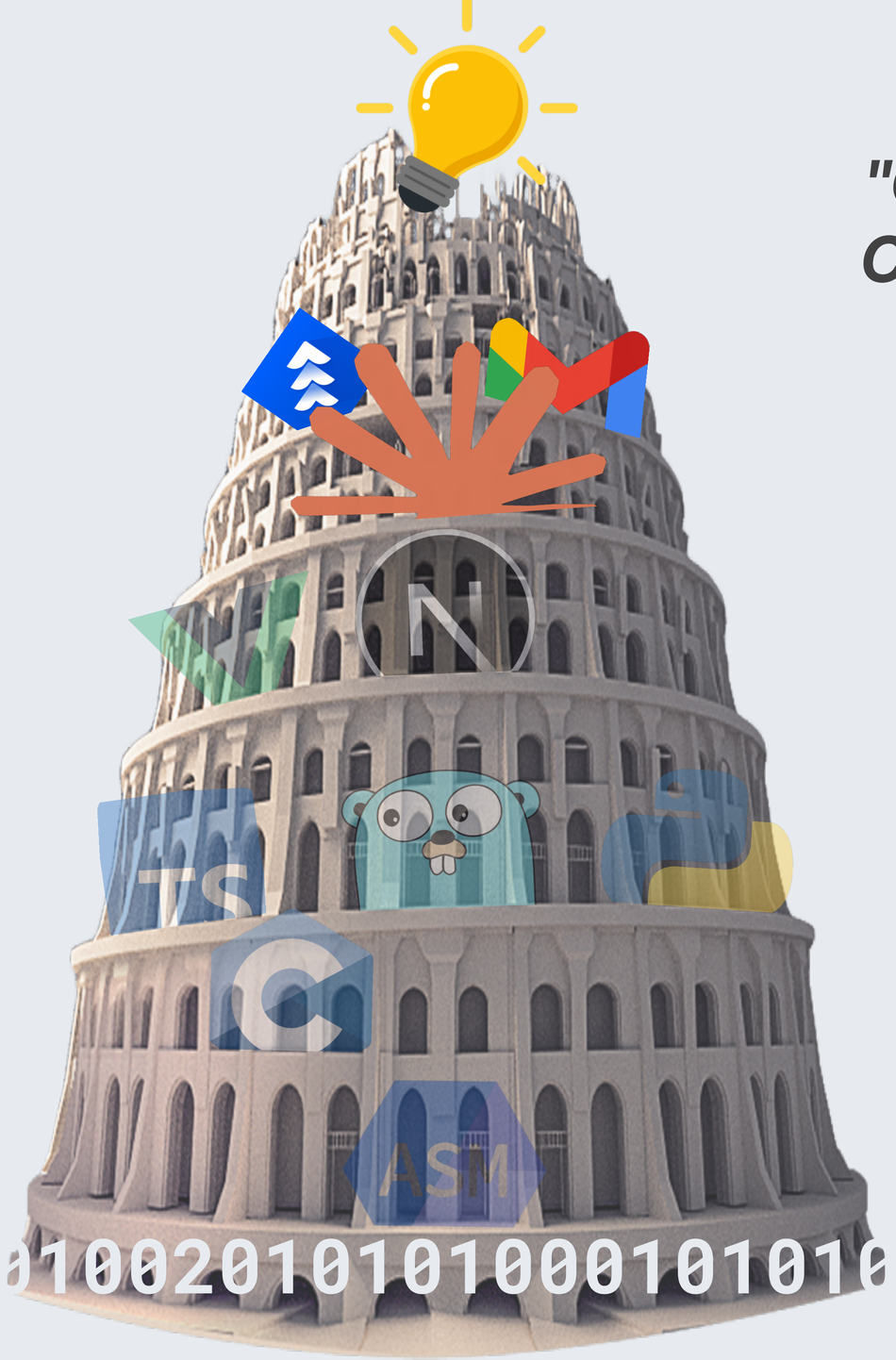

N
TS
C
ASM
01002010101000101010


TS
ASM
01002010101000101010



"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
01002010101000101010

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

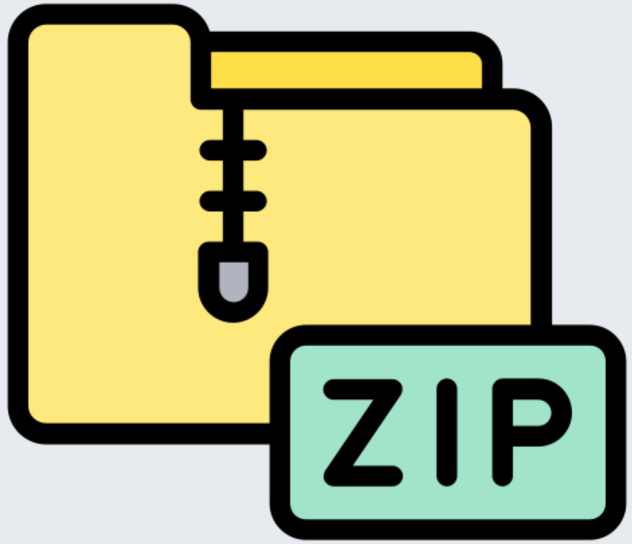
TypeScript

Haskell

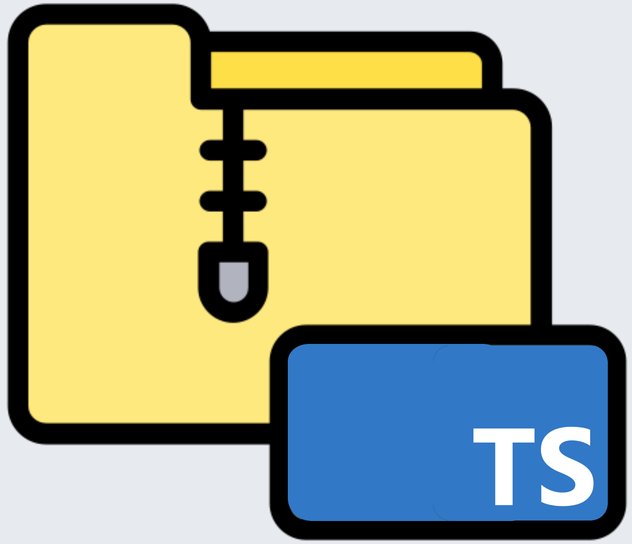

TS
ASM
01002010101000101010

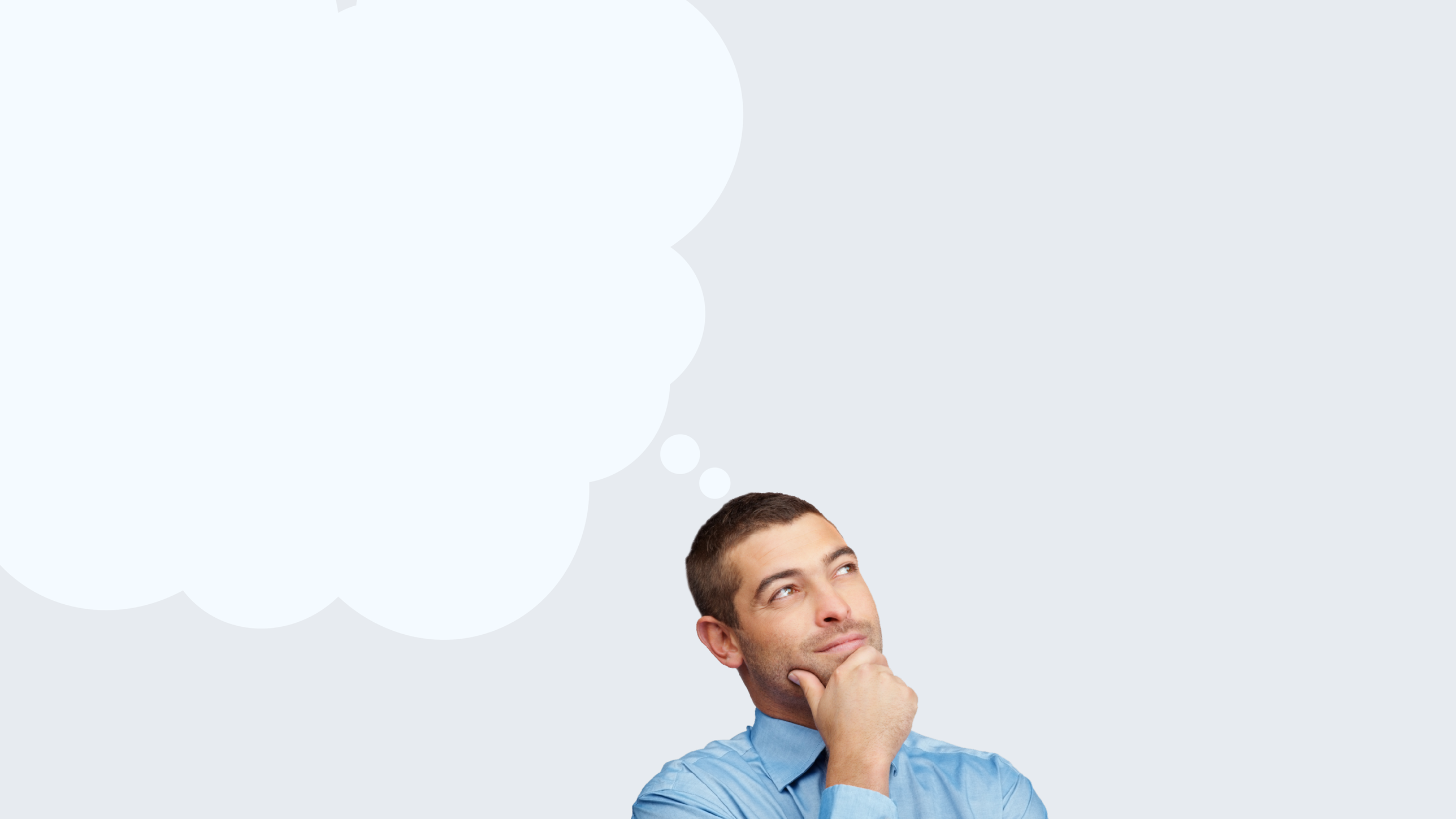
But why TypeScript?

Software engineering is **compression**!



Software engineering is compression!





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

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

n appropriate

base if we

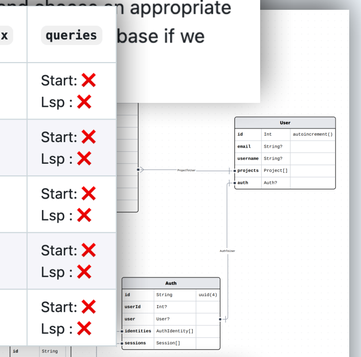
Before:

```
$ 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 : ❌

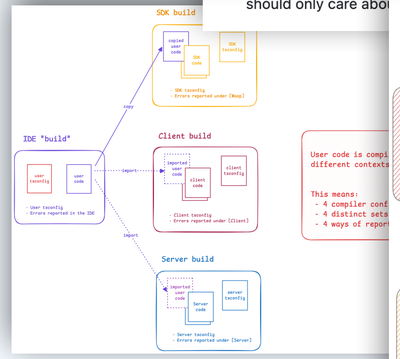


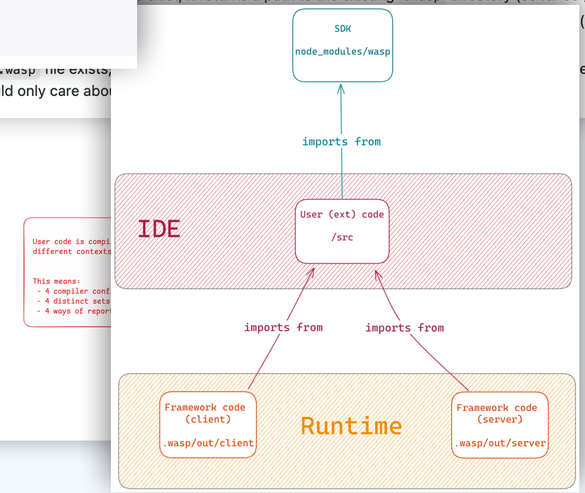
s exist, it returns a path to the existing `.wasp` directory (so far so good)

(this is the

e implies it

should only care abo

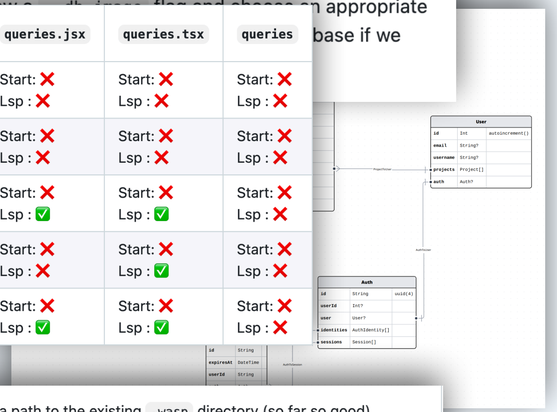




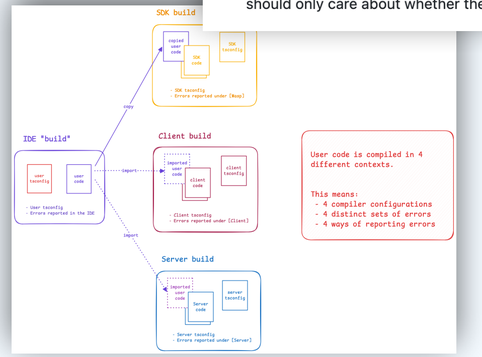
`run-wasp-app dev` should then allow the `--db-type` to be either `SQLite`, `MySql`, or `PostgreSQL`.

- 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 default base if we decide

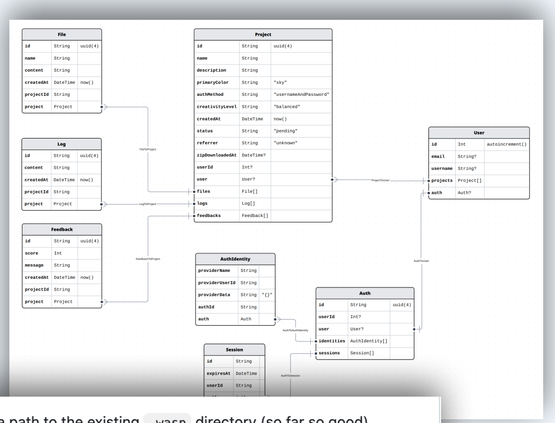
Real file name (right) Imported as (down)	`queries.js`	`queries.ts`	`queries.jsx`	`queries.tsx`	`queries`
`queries.js`	Start: ✅ Lsp : ✅	Start: ✅ Lsp : ✅	Start: ❌ Lsp : ❌	Start: ❌ Lsp : ❌	Start: ❌ Lsp : ❌
`queries.ts`	Start: ✅ Lsp : ❌	Start: ✅ Lsp : ❌	Start: ❌ Lsp : ❌	Start: ❌ Lsp : ❌	Start: ❌ Lsp : ❌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



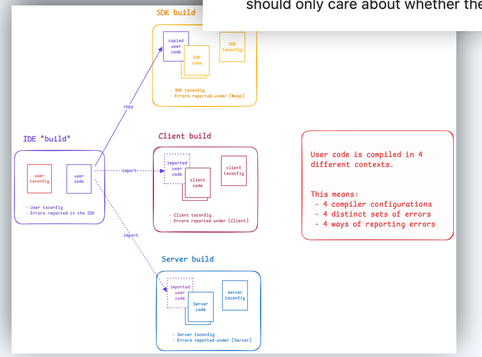
- If a `.wasp` directory does exist, it returns a path to the existing `.wasp` directory (so far so good)
- If the `.wasp` directory doesn't exist, it returns a path to the non-existing `.wasp` directory (this is the first weird part)
- If a `.wasp` file exists, it returns an error saying that `.wasp`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

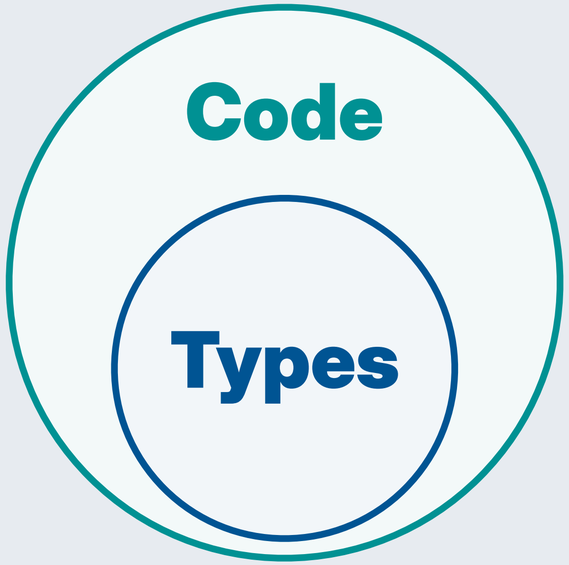


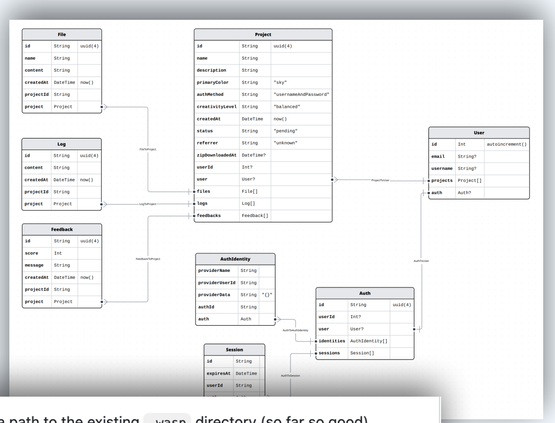
Types



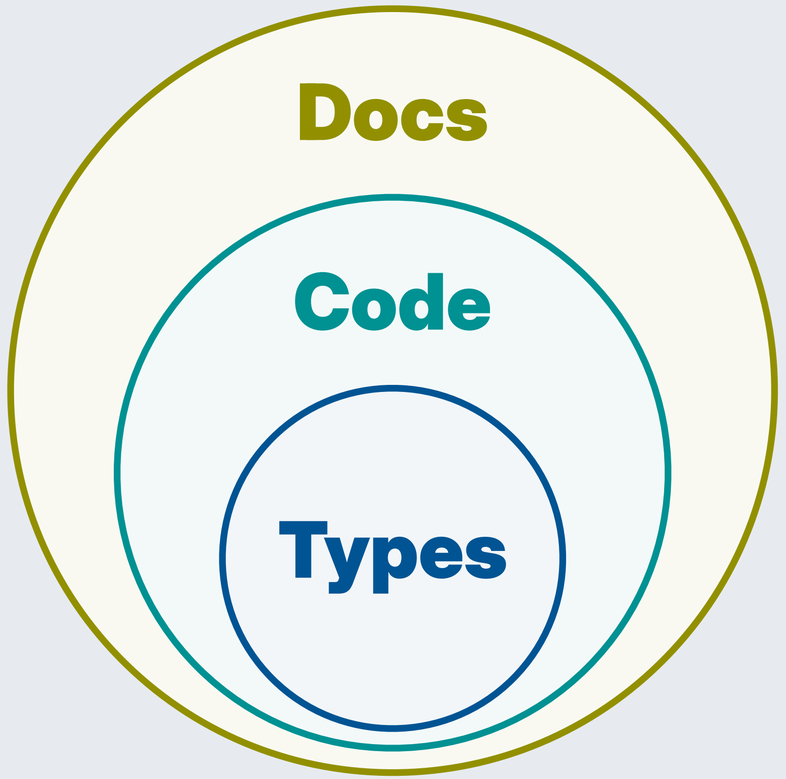
- If a `.wasp` directory does exist, it returns a path to the existing `.wasp` directory (so far so good)
- If the `.wasp` directory doesn't exist, it returns a path to the non-existing `.wasp` directory (this is the first weird part)
- If a `.wasp` file exists, it returns an error saying that `.wasp`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

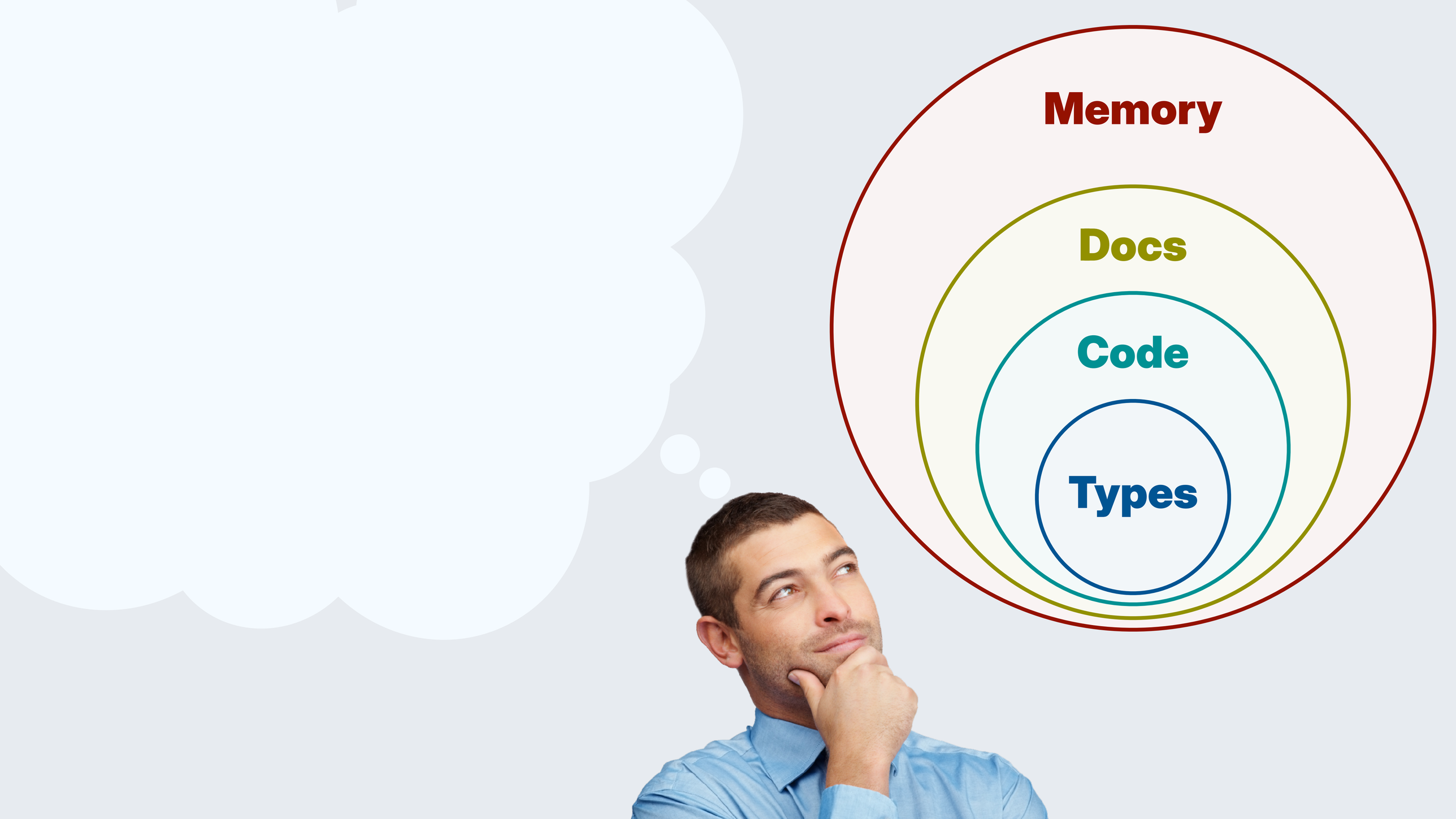


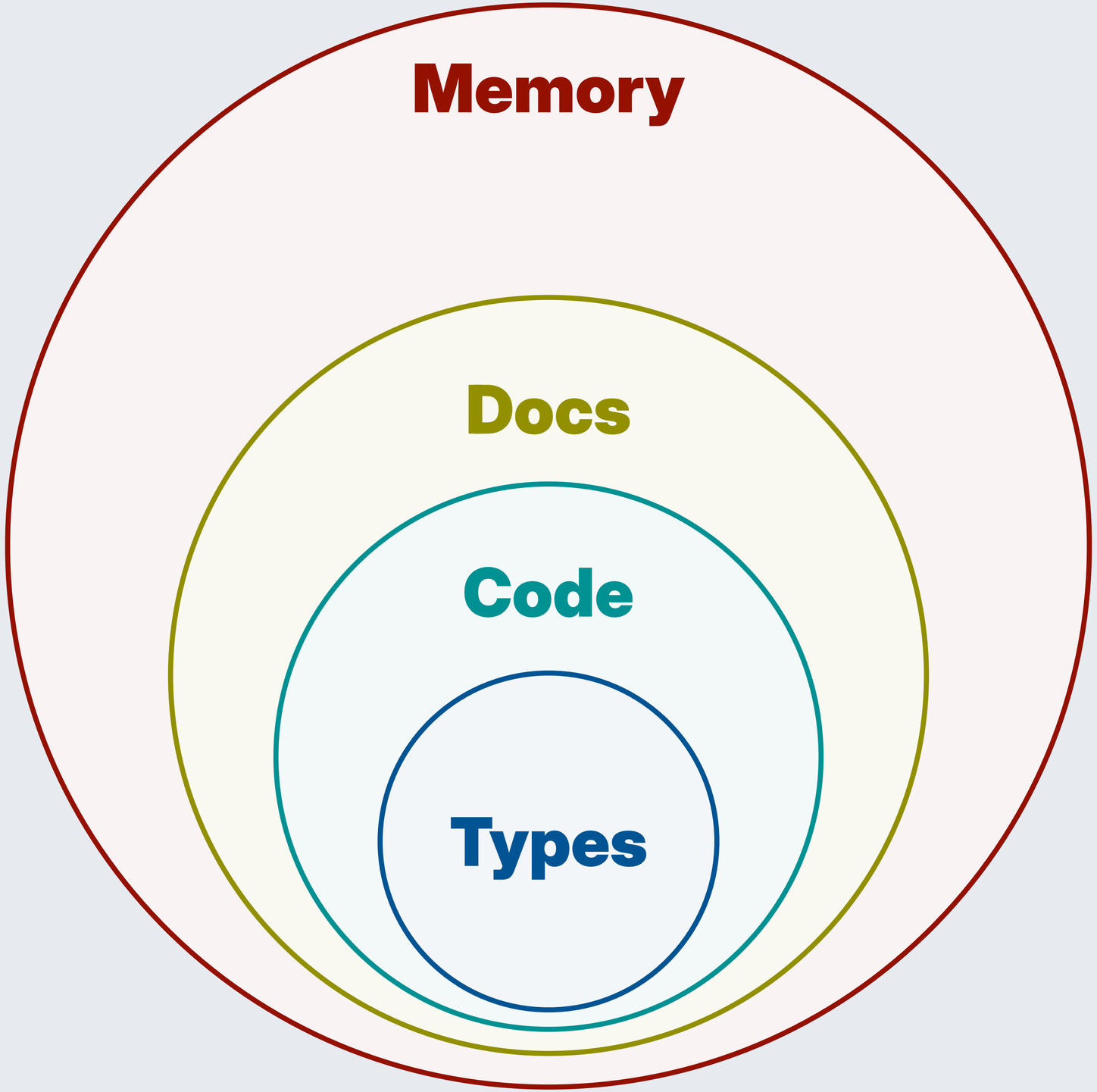


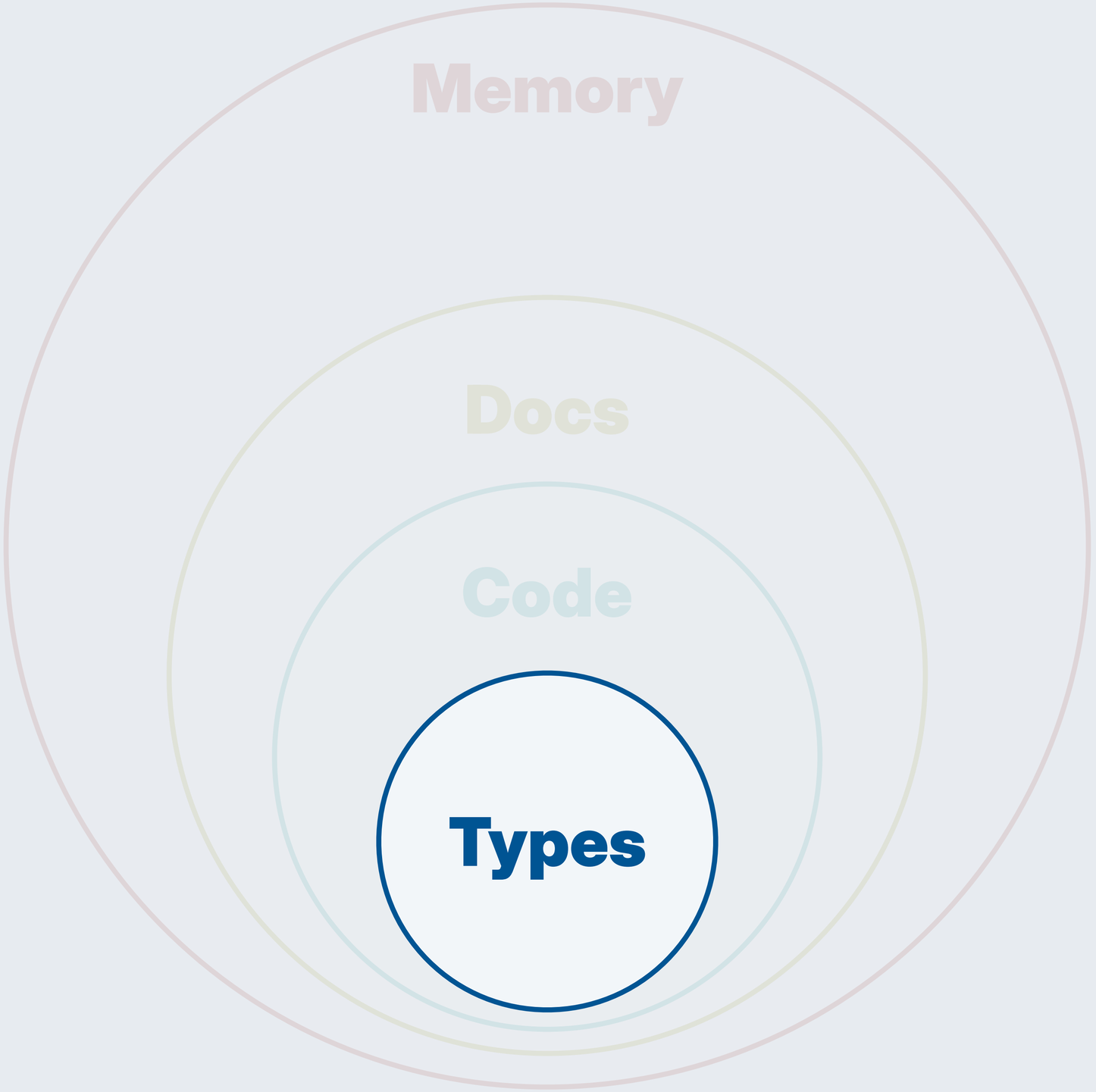


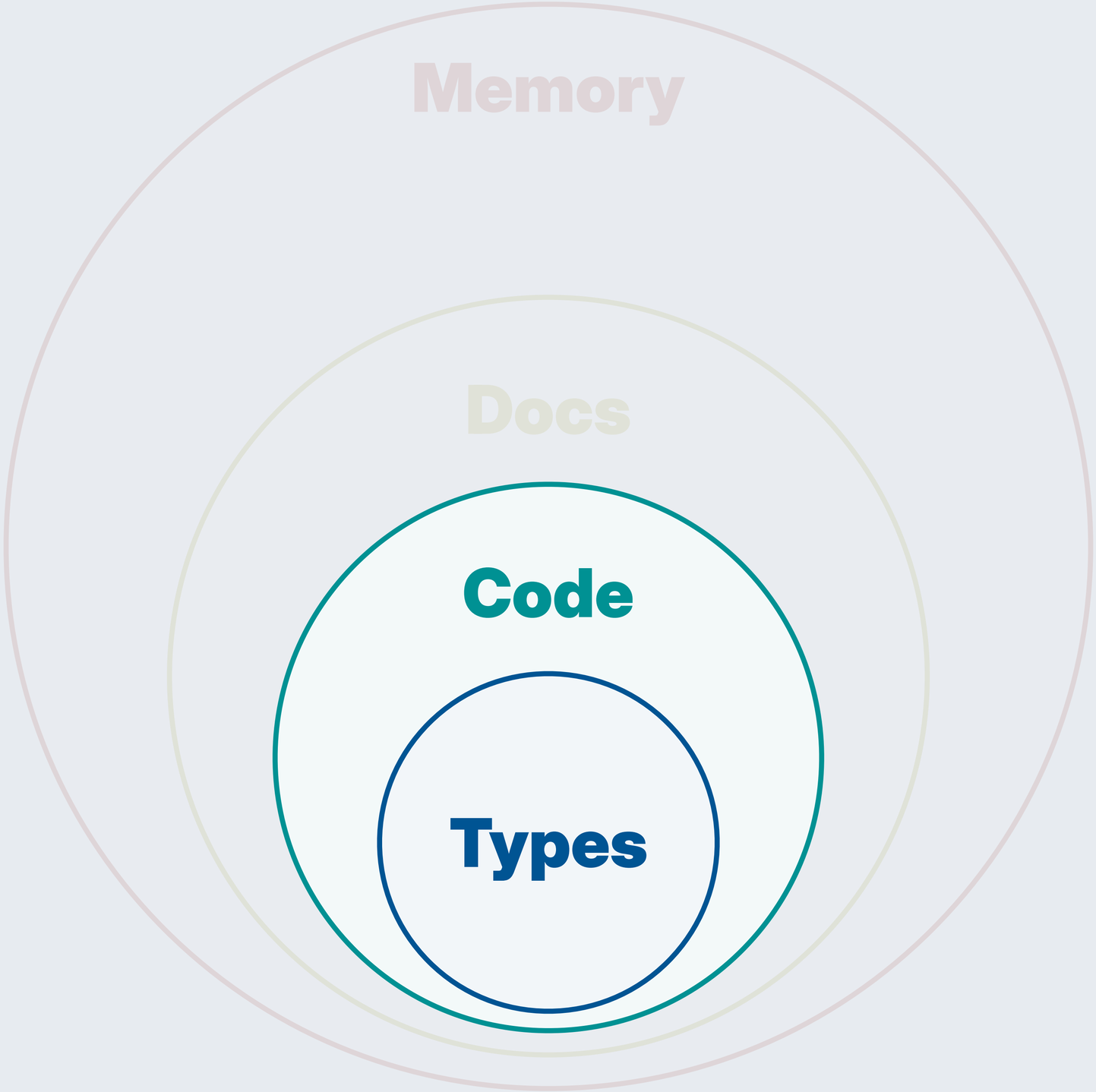
- If a `.wasp` directory does exist, it returns a path to the existing `.wasp` directory (so far so good)
- If the `.wasp` directory doesn't exist, it returns a path to the non-existing `.wasp` directory (this is the first weird part)
- If a `.wasp` file exists, it returns an error saying that `.wasp`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

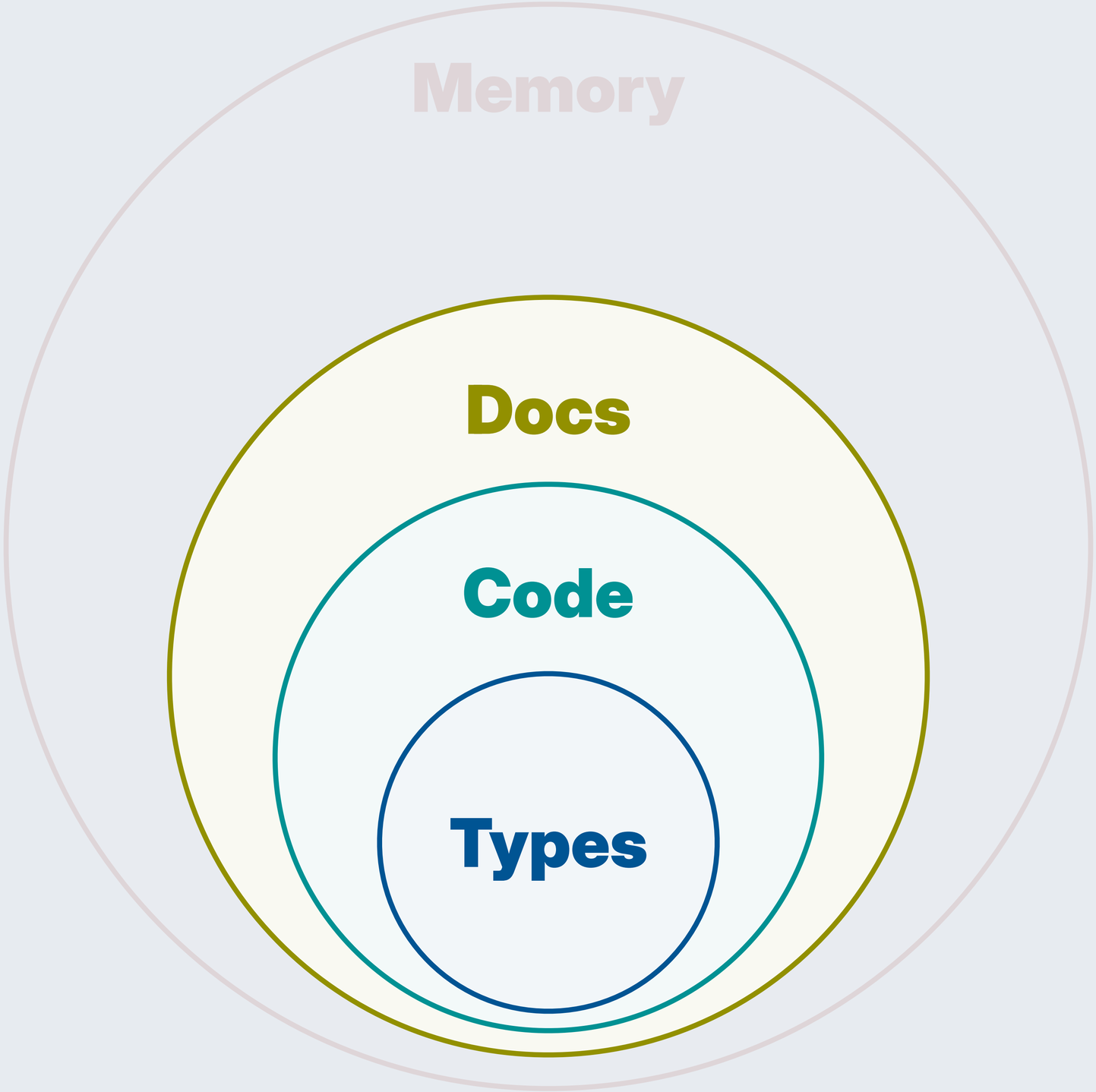


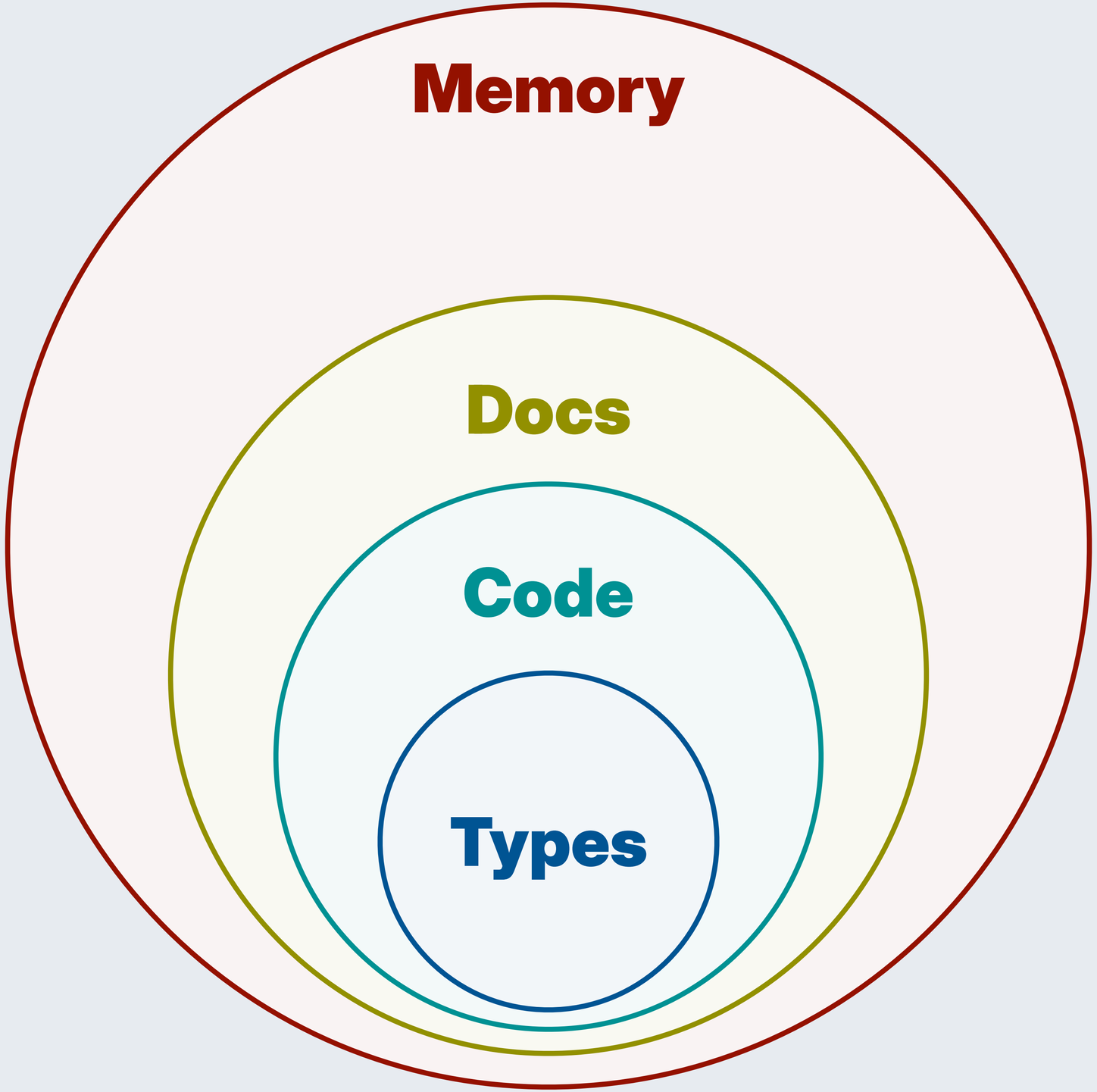

Memory
Docs
Code
Types

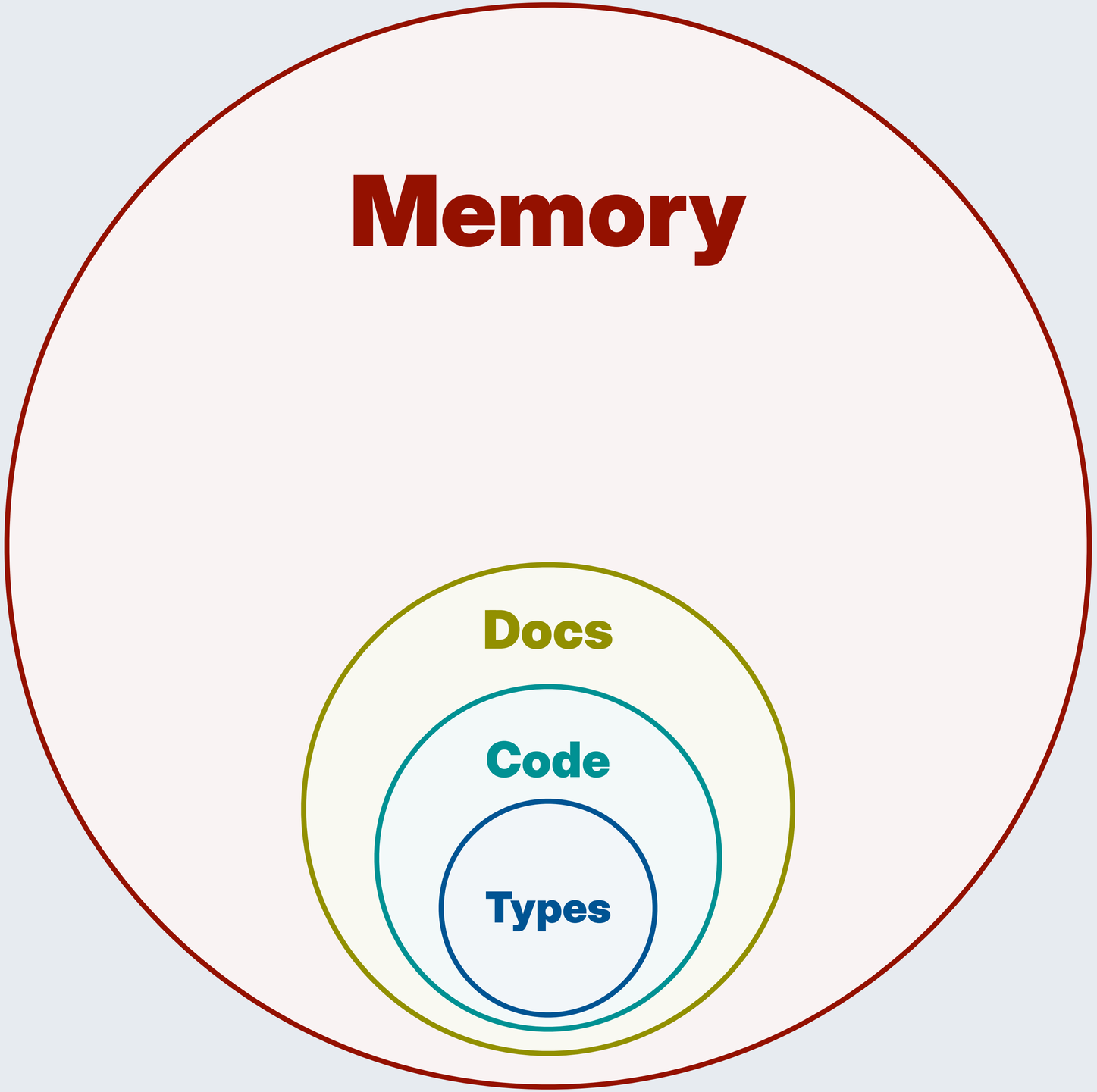

Memory
Docs
Code
Types

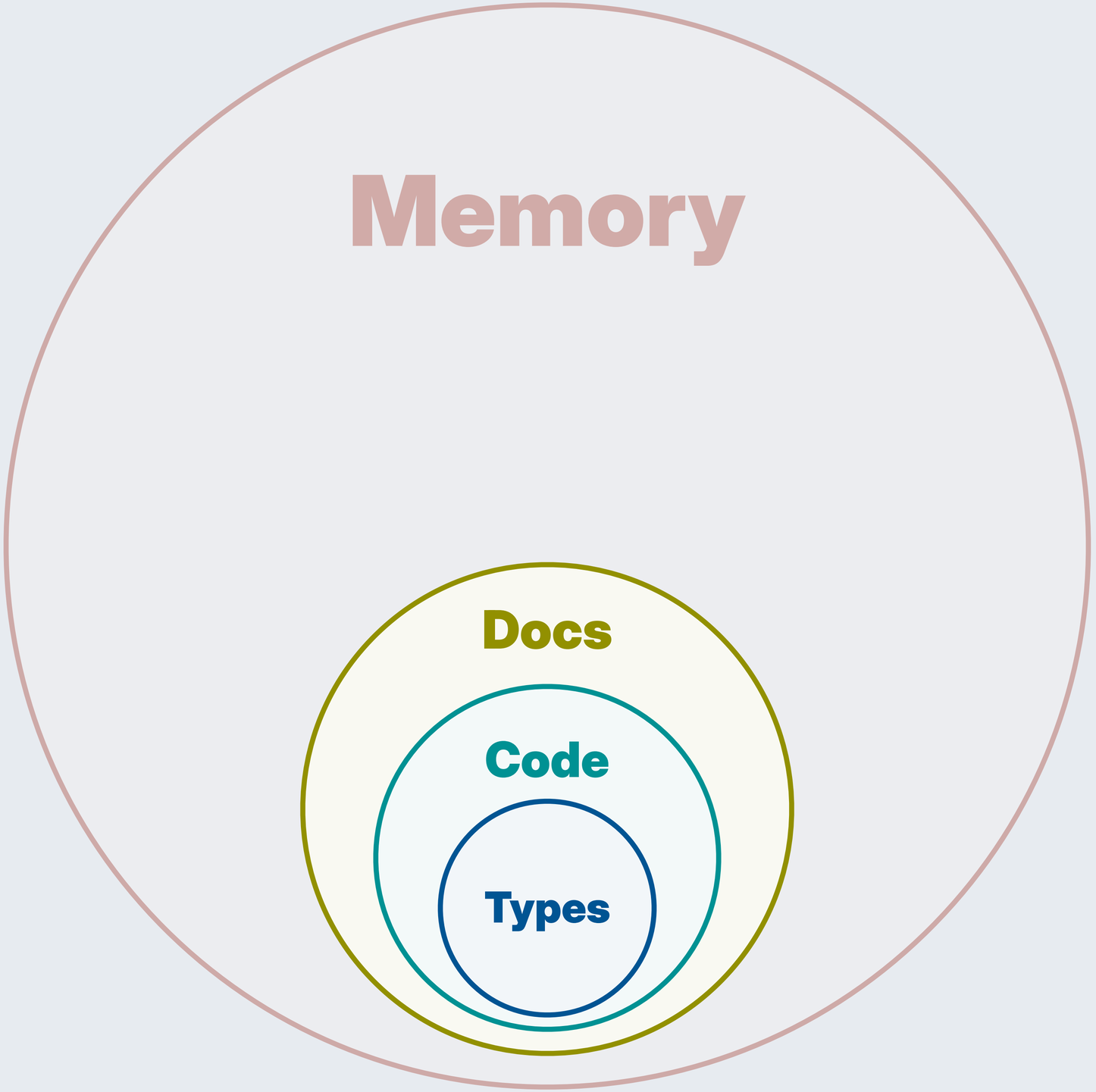

Memory
Docs
Code
Types


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 assumptions.
No rambling.
Just excellent 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 assumptions.
No rambling.
Just excellent documentation.

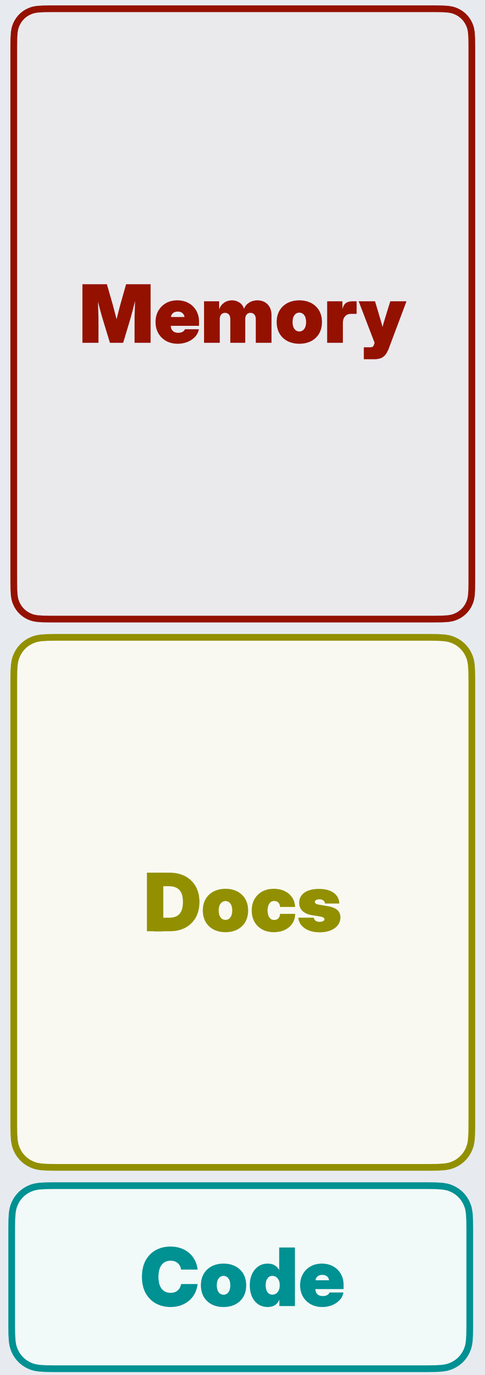
Please document the code like a
principal engineer. Make no mistakes.

✻ 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 assumptions.
No rambling.
Just excellent documentation.



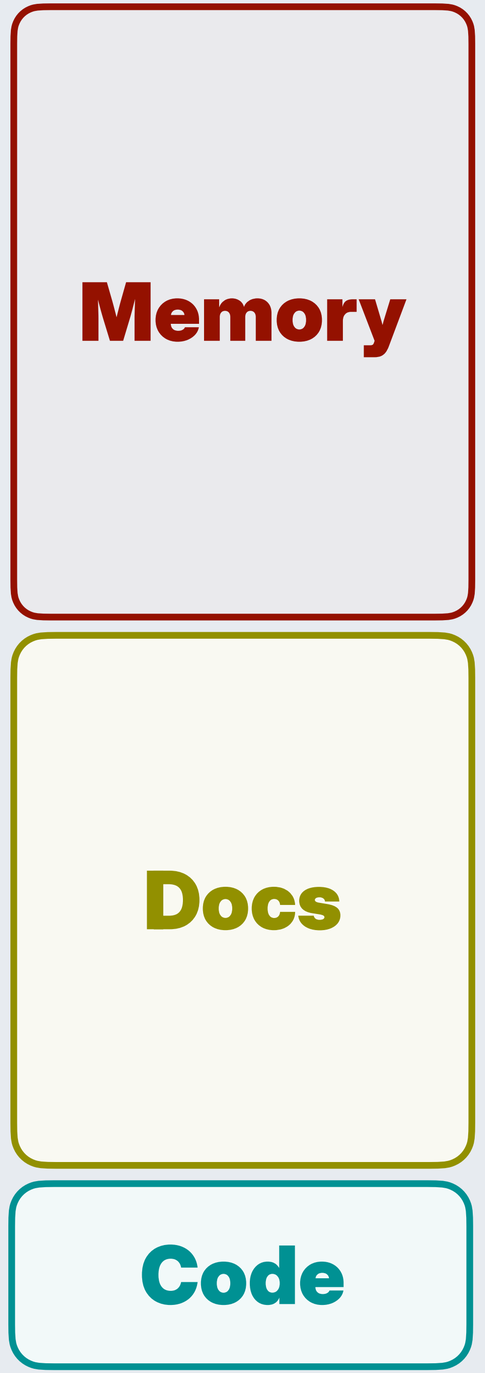
Please document the code like a
principal engineer. Make no mistakes.

✻ 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 assumptions.
No rambling.
Just excellent 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)
}
```

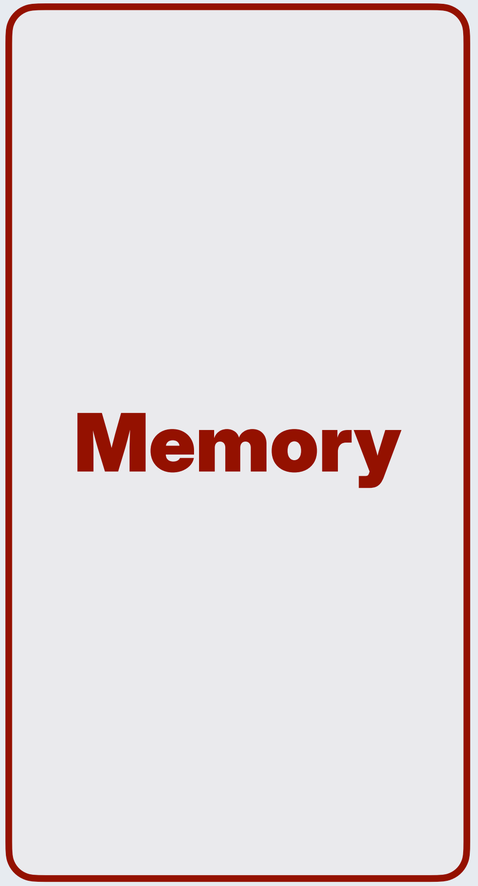
```
serve({ kind: 0xCOFFEE })
```

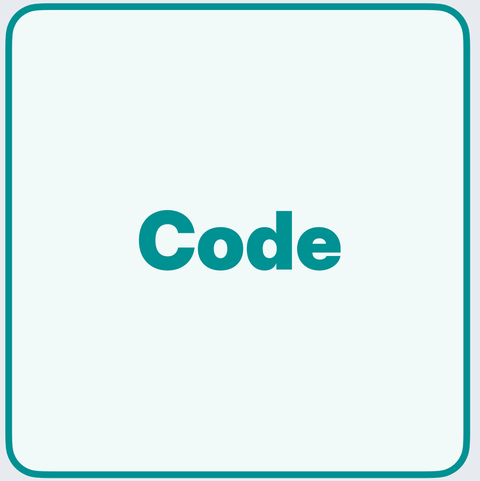
```
function serve({ kind, beans, shots, milk }) {
    grindBeans(beans)
    if (kind === "drip") pourDrip()
    else pullShots(beverage.shots)
    if (beverage.milk)
        pourMilk(beverage.milk)
}
```

```
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)
}
```

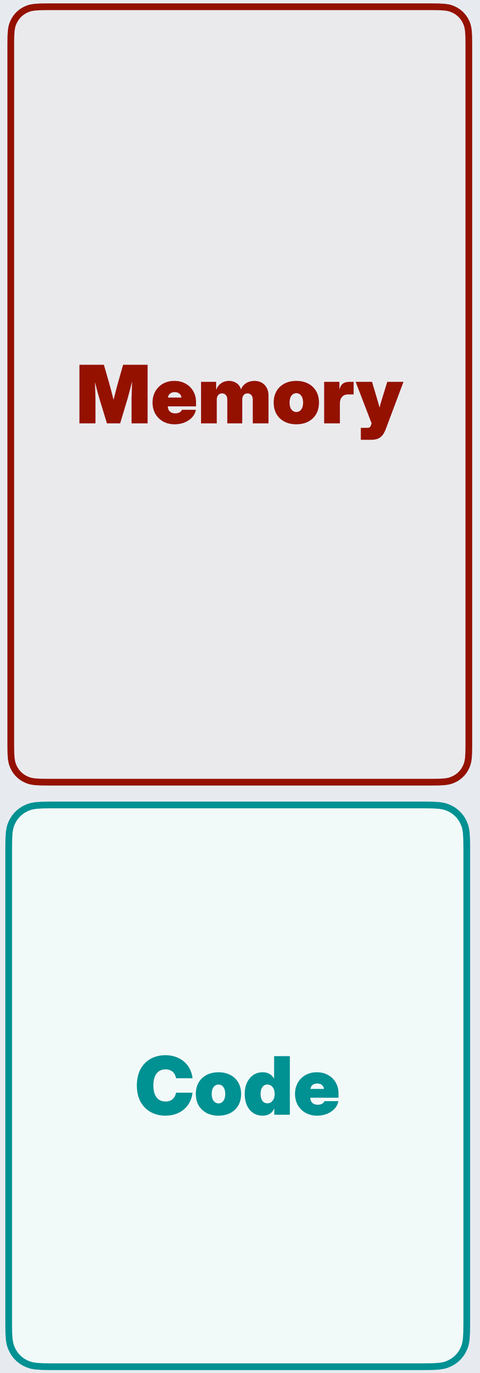
```
serve({ kind: 0xCOFFEE })
serve({ beans: 100 })
serve({ beans: "robusta"})
```





```
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" })
```

Memory

Code

```
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!

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"
}
```

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"
}
```

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"
}
```

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 EspressoBasedCoffe =
  | 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"
```

Drip coffee is served **with or without** 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 EspressoBasedCoffe =
  | StraightEspresso
  | EspressoWithMilk

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

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

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

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

type EspressoBasedCoffe =
  | 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
  | EspressoBasedCoffe
```

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

type EspressoBasedCoffe =
  | 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
  | EspressoBasedCoffe
```

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

```ts
type EspressoBasedCoffe =
  | 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
  | EspressoBasedCoffe
```

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

What kinds of coffee exist?

```ts
type EspressoBasedCoffe =
  | 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
  | EspressoBasedCoffe
```

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

What kinds of coffee exist?

```ts
type EspressoBasedCoffe =
  | 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
  | EspressoBasedCoffe
```

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

What kinds of coffee exist?

```ts
type EspressoBasedCoffe =
  | 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
  | EspressoBasedCoffe
```

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

What kinds of coffee exist?

```ts
type EspressoBasedCoffe =
  | 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
  | EspressoBasedCoffe
```

Can I have tea, or a cold brew?

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

What kinds of coffee exist?

```ts
type EspressoBasedCoffe =
  | 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
  | EspressoBasedCoffe
```

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

```ts
type EspressoBasedCoffe =
  | 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,
}
```

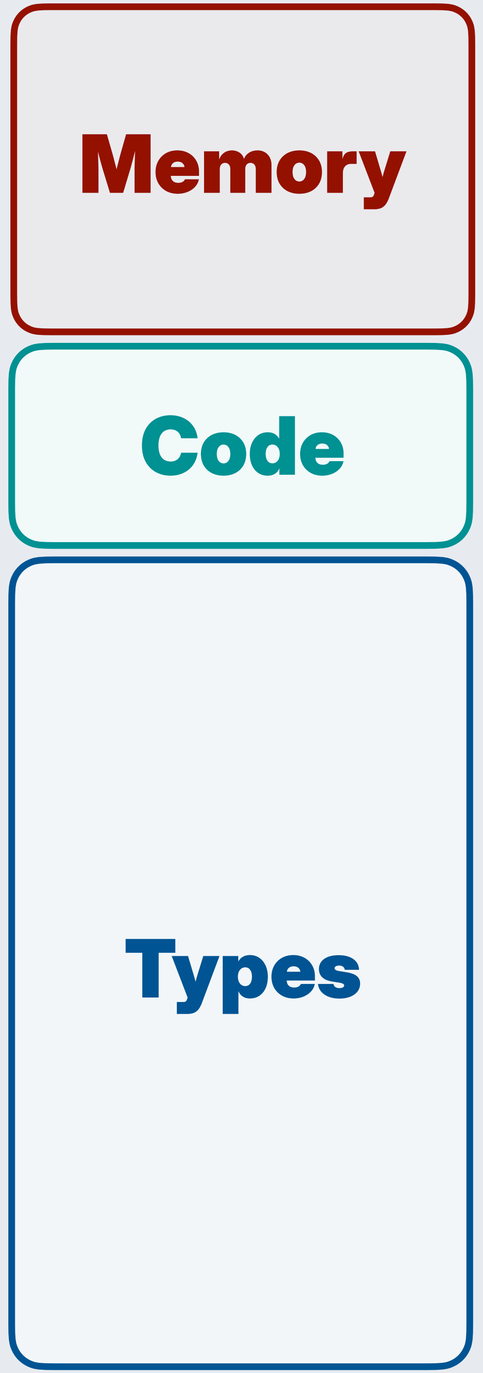
```ts
type Espresso = {
  coffee: Coffee
  shots: 1 | 2
}
```

```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" })
```

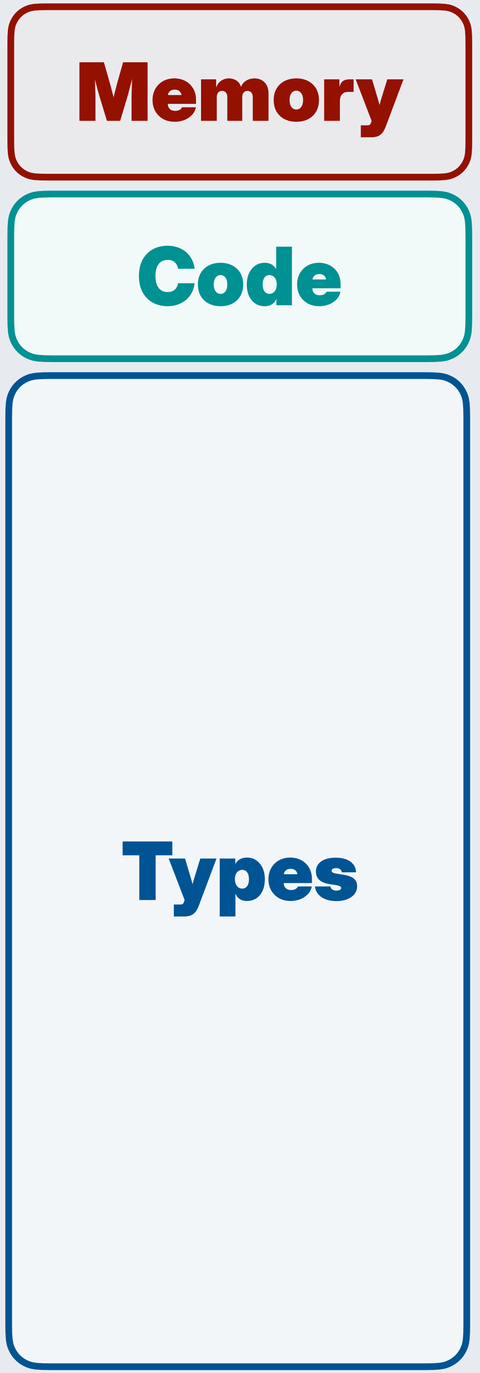
```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```



```
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" })
```

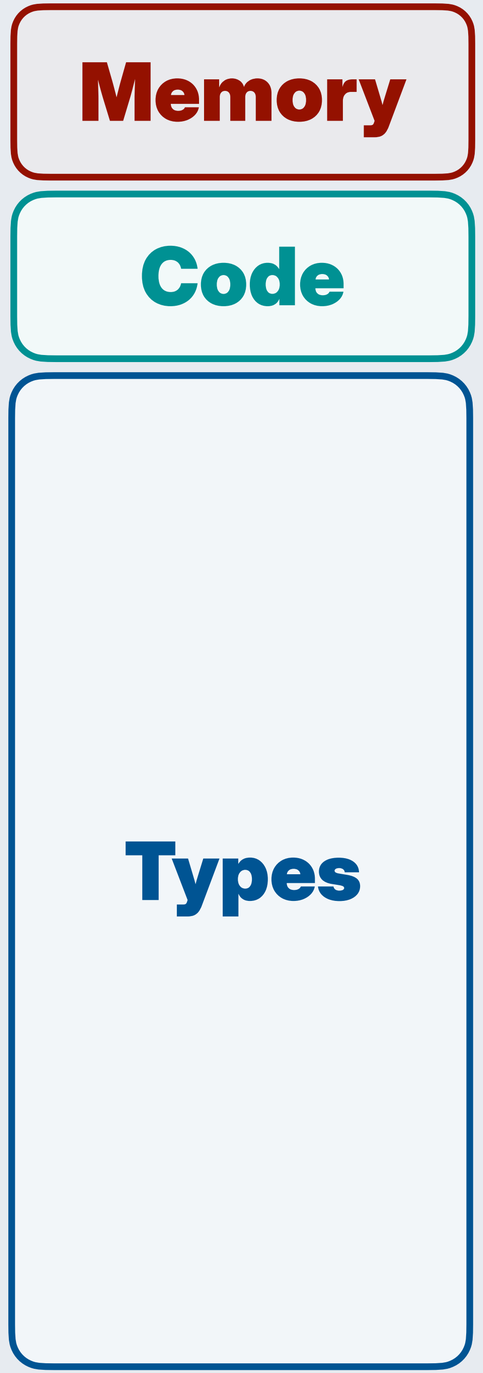
```
serve({ kind: "espresso", shots: 2, beans: "arabica" })
serve({ kind: "drip", beans: "robusta", milk: "oat"})
```



```
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
  | EspressoBasedCoffe
```

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

```ts
type EspressoBasedCoffe =
  | 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"
}
```

Branded types

The Result type

Branded types

Ensuring exhaustiveness

The Result type

Branded types

Ensuring exhaustiveness

The Result type

Smart constructors

Branded types

Ensuring exhaustiveness

The Result type

Smart constructors

Branded types

Edge type safety

Ensuring exhaustiveness

The Result type

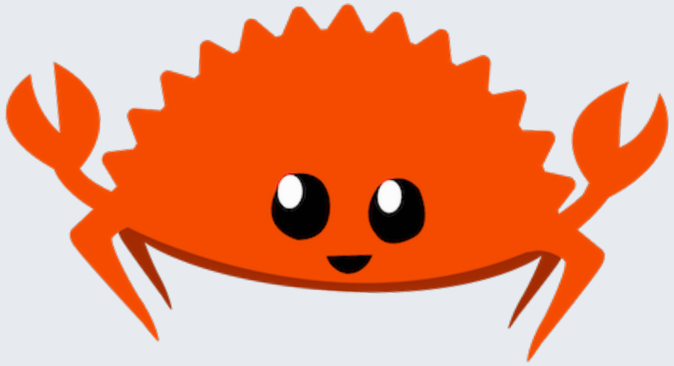
Smart constructors

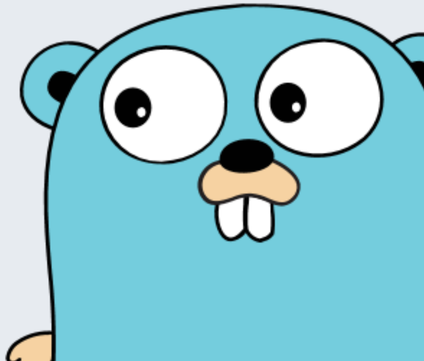
Branded types

Edge type safety

What about the competition?

What about the competition?





It gets better!

It gets better!

Modern and safer defaults

Rewrite in Go

Better errors

Removed legacy baggage

Most popular language

Choosing TypeScript matters more than ever

Choosing TypeScript matters more than ever

Choosing TypeScript matters more than ever

Thank You!

Resources