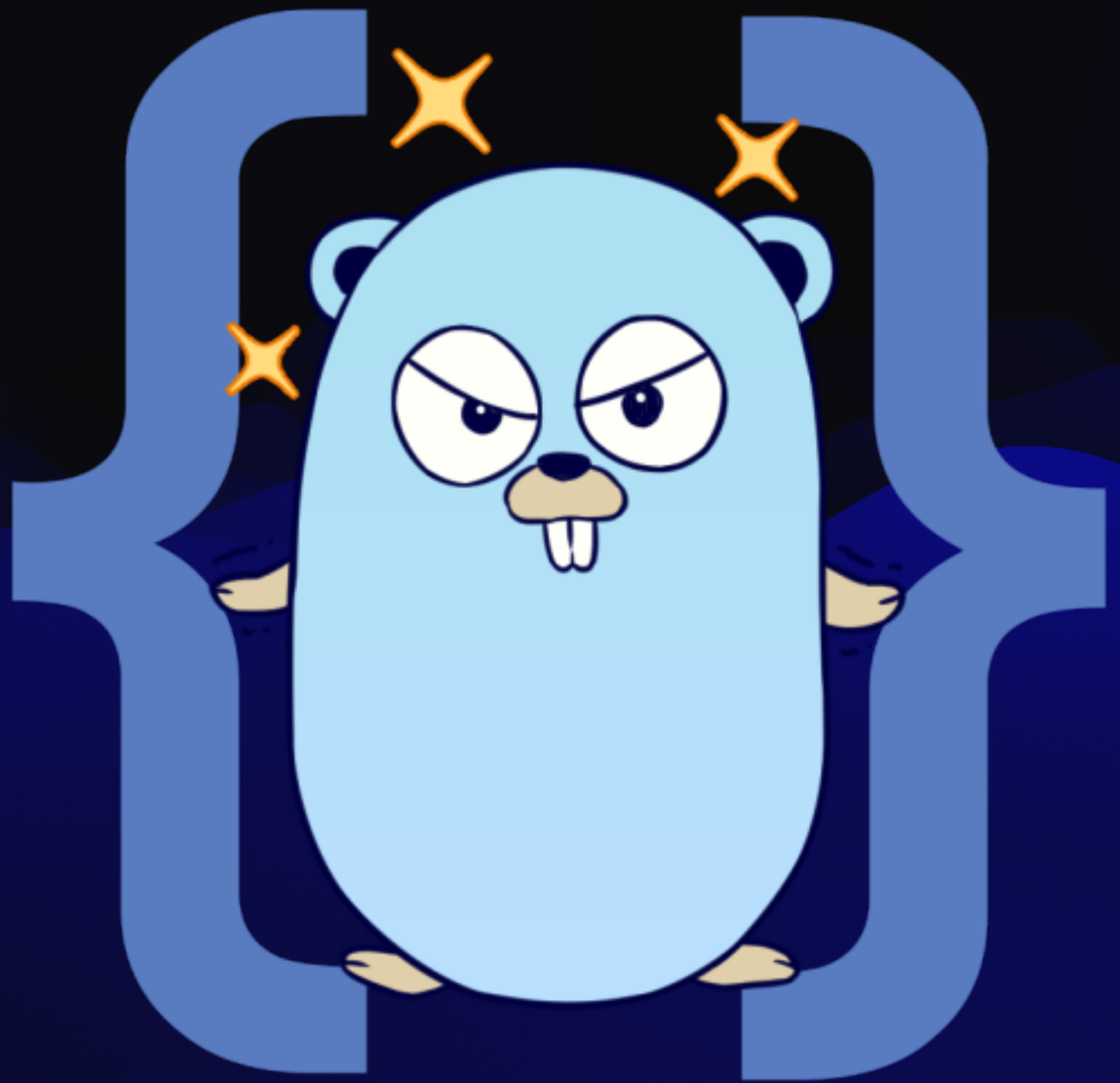
JSON

Why do I have to suffer

Nick

JSON now, JSON future

You can’t escape Javascript



https://github.com/bytedance/sonic

JSON is not that convenient in Go

Put down the pitchforks, I haven’t yet given you enough reasons to hate me

- You have 2 choices
  - json.NewDecoder
  - json.Unmarshal
- Slow*
- Limited options customization or none at all.

Let’s start simple

```go
var data struct {
    key string `json:"the_key"`
}

jsonData := []byte(`{"the_key": "Value"}`)
err := json.Unmarshal(jsonData, &data)

fmt.Printf("%v\n", data)
fmt.Printf("%s\n", err)
```

Struct Tags are strings

Obviously… duh???

- What’s not obvious is that different tools parse struct tags differently.
- There is no spec regarding how these strings should be structured nor is there any validation.
- So you could be writing mess like this and be ok.
- Oh and you can’t break struct tags to multi line, that will not work as reflection expects one whole line :)

```
type IAmAtThePayphone struct {
    TryingToCallHome int32 `json:"integration_type_id"
validate:"gte:1"`
}
```

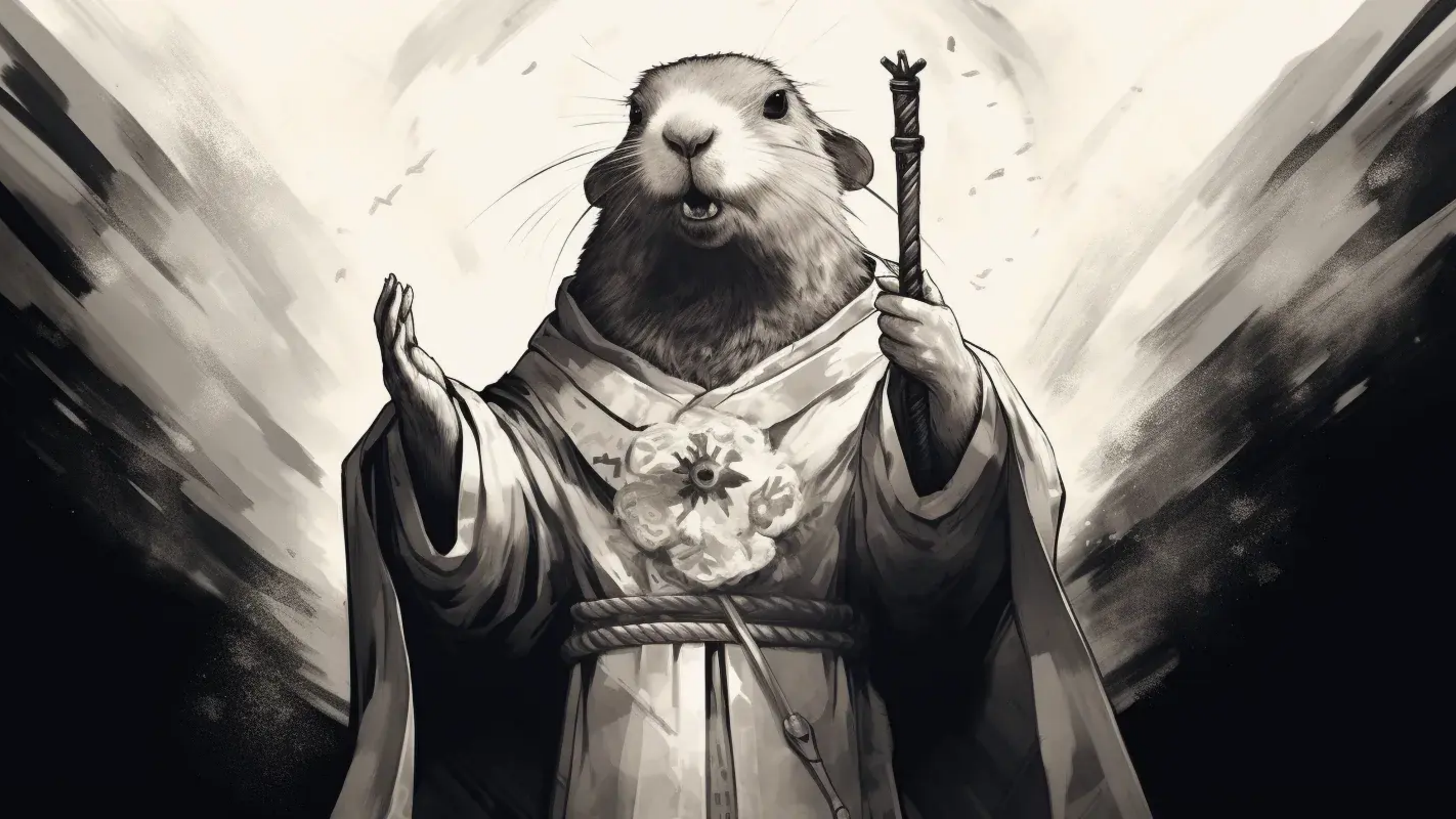
Oh v2 and struct tags are case insensitive as a bonus (ﾉ °□°)ﾉ ⌒ ┻━┻

```go
var data struct {
    Key string `json:"k"`
}
jsonData := []byte(`{"k": "Value", "k": "Hello"}`)
```

Ok we get it...

In no particular order

- When marshaling Go maps to json, they are alphabetically sorted.
- Byte slices are encoded as base64 strings
- json.Decoder loves `{}{"Name": “Bob"}`
- You have to do struct embedding for nested objects
  - Unless you are mad enough to do `map[string]any` and call it pseudo-Python
    - (ಠ ʖ̯ ಠ) what’s wrong with you?
- JS is all floats, so good luck marshaling/unmarshaling uint, big.Int and staying consistent :3

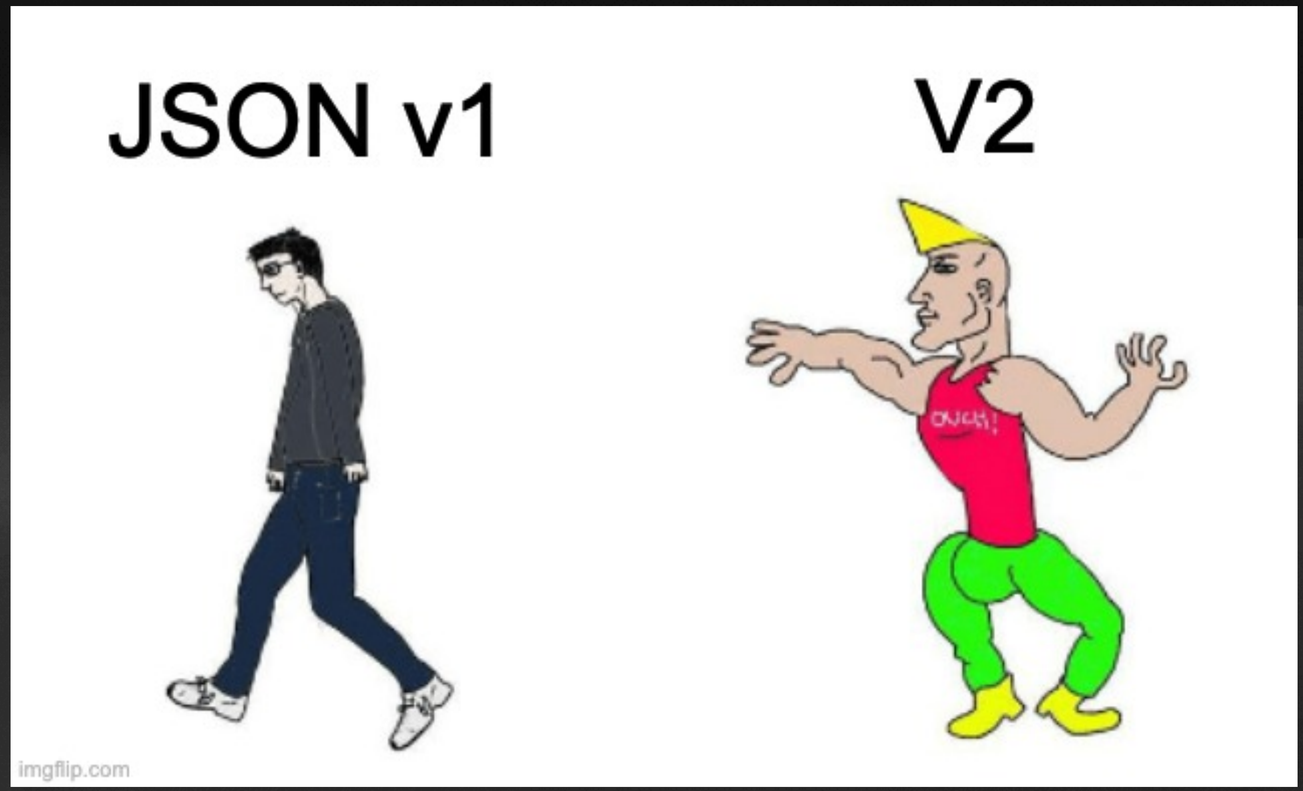


https://github.com/golang/go/discussions/63397

Or just type in google jsonv2 golang…

Some Highlights

- Expand options that can be passed into Decoder for greater control
- Case sensitive matching
- Nil maps/slices are marshaled into empty object/array
- Error on trying to serialized struct with only unexported members.
- Omitzero



Backwards compatible

io.EOF

Questions?