Creating Smart Templates

Michael Van Sickle

@vansimke

Introduction

- Pipelines
- Built-in functions
- Custom functions
- Logical tests
- Looping

Pipelines

```
{{command1 command2 command3}}
```

Output must have one or two values.
Second must be and error type.

```
{{.Title}}
```

◂ Data command

```
{{template “content”}}
```

◂ Function with one argument

```
type Data struct {}
func (d Data) SayMsg(m string) string {
    return m
}
```

```
{{.SayMsg “Hello World!”}}
```

◂ Method with one argument

Pipelines

```
{{ command1 command2 | command3 }}
```

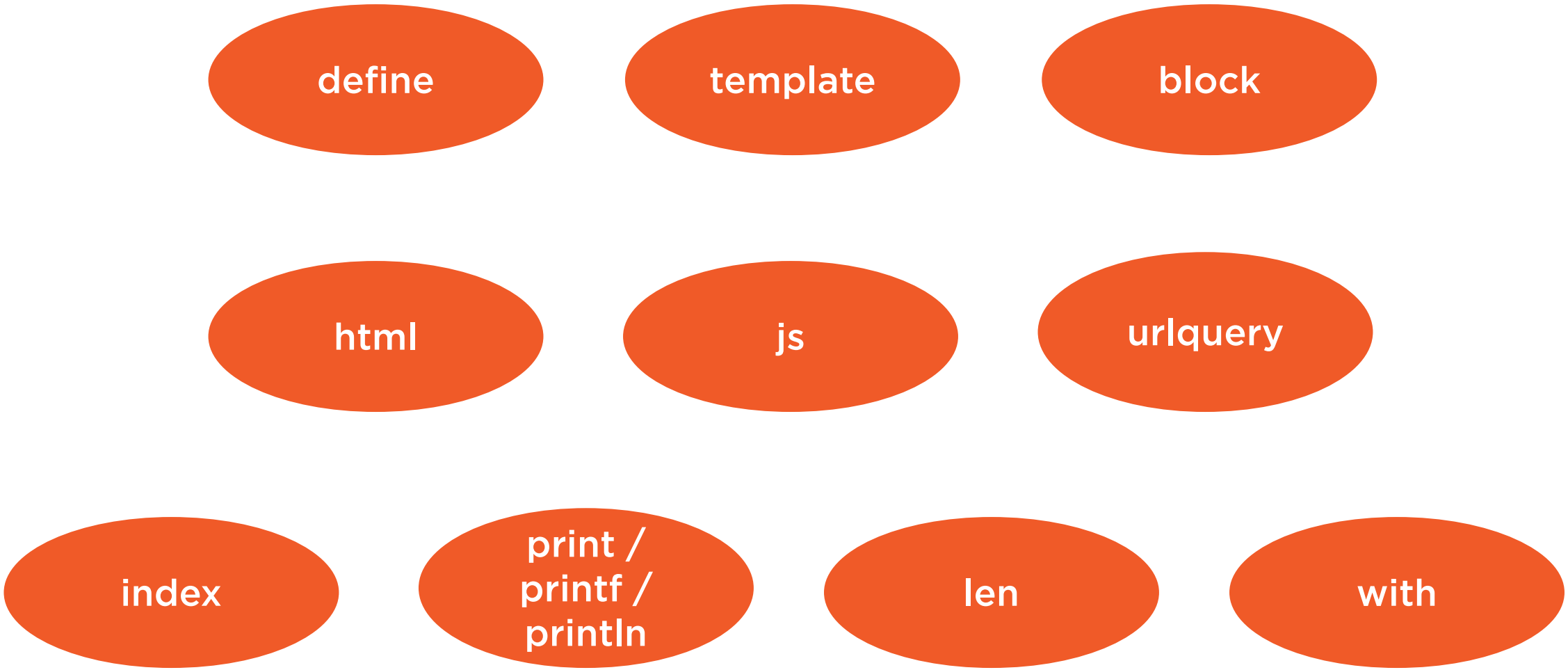
Pipelines

```
{{ command1 command2 | command3 }}
```

Pipe operator

Pass result of previous command as last argument of next command

Built-in Functions



Custom Functions

```
template.Funcs(funcMap FuncMap) *Template

type FuncMap map[string]interface{}

template.New(“”).Funcs(funcMap).Parse(...)
```

Acceptable values

- Function that returns a single value
- Function that returns a single value and an error type

If Blocks

```
{{if pipeline}}
    T1
{{end}}
```

- T1 prints if pipeline results in non-empty value

Empty values

false	zero
nil	empty collection

```
{{if pipeline}}
    T1
{{else}}
    T2
{{end}}
```

- T1 prints if pipeline is non-empty, otherwise T2 prints

```
{{if pipeline}}
    T1
{{else if pipeline}}
    T2
{{end}}
```

- T1 prints if first pipeline is non-empty, otherwise T2 prints if second pipeline is non-empty

Logical Operators

eq / ne	lt / gt	le / ge
and	or	not

All arguments are evaluated!

```
{{range pipeline}}
    T1
{{end}}
```

```
{{range pipeline}}
    T1
{{else}}
    T2
{{end}}
```

Range Blocks

- Pipeline must be array, slice, map or channel
- Data context of T1 is the current collection item
- T1 executed if pipeline is non-empty, otherwise T2 is executed

Summary

- Pipelines
- Built-in functions
- Custom functions
- Logical tests
- Looping