Using Templating

Alex Schultz
SOFTWARE ENGINEER | AWS ML HERO
@AlexCSchultz

Overview

- Templating Basics
- Pipelines
- Looping
- Global Functions and Operators
- Custom Functions

Templates

Templates

```
type BlogPost struct {
  Header string
  Message string
}
```

```
<html>
  ...
  <h1>{{.Header}}</h1>
  <div>
   <p>{{.Message}}<p>
  </div>
  ...
</html>
```

Template Packages

text/template

Base functionality for working with templates in Go

html/template

Same interface but with added security for HTML output

template.New

```
func New(name string) *Template
```

Template.Parse

```
func (t *Template) Parse(text string) (*Template, error)
```

Template.Execute

```
func (t *Template) Execute(wr io.Writer, data interface{}) error
```

main.go

```
import “html/template”
type BlogPost struct {
  Header string
  Message string
}
func main() {
  post := BlogPost{“First Post!”, “Hello World”}
  tmpl, _ := template.New(“post”).Parse(`<h1>{{.Header}}</h1><p>{{.Message}}</p>`)
  tmpl.Execute(os.Stdout, post)
}
```

<h1>First Post!</h1> Hello World

Pipelines

Command or sequence of commands

- Simple value (argument)
- Function or method call
- Can accept arguments

Pipelines

```
{{ “Hello” }}

{{ 1234 }}

{{ .Message }}
```

```
{{ println “Hi” }}

{{ .SayHello }}

{{ .SaySomething “Bye” }}
```

Pipeline Chaining

```
{{ .SaySomething “Hello” }}

{{ “Hello” | .SaySomething }}

{{ “Hello” | .SaySomething | printf “%s %s” “World” }}
```

Pipeline Looping

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

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

Pipeline Looping

```
{{ range $index, $element := pipeline }}
```

main.go

```
import “html/template”



tmpl := “{{range .}}{{.}}{{end}}”
func main() {
  items := []string{“one”, “two”, “three”}
  tmpl, _ := template.New(“tmplt”).Parse(tmpl)
  err := tmpl.Execute(os.Stdout, post)
}
```

onetwothree

Template Functions

Function	Example
and	{{if and true true true}} {{end}}
or	{{if or true false true}} {{end}}
index	{{index . 1}}
len	{{len .}}
not	{{if not false}}
print, printf, println	{{println “hey”}}

https://golang.org/pkg/text/template#hdr-Functions

Template Operators

```
eq      arg1 == arg2

ne      arg1 != arg2

lt      arg1 < arg2

le      arg1 <= arg2

gt      arg1 > arg2

ge      arg1 >= arg2
```

Template.Funcs

```
func (t *Template) Funcs(funcMap FuncMap) *Template

type FuncMap map[string]interface{}
```

Template Functions

Return a single value

Return a single value, or an error

main.go

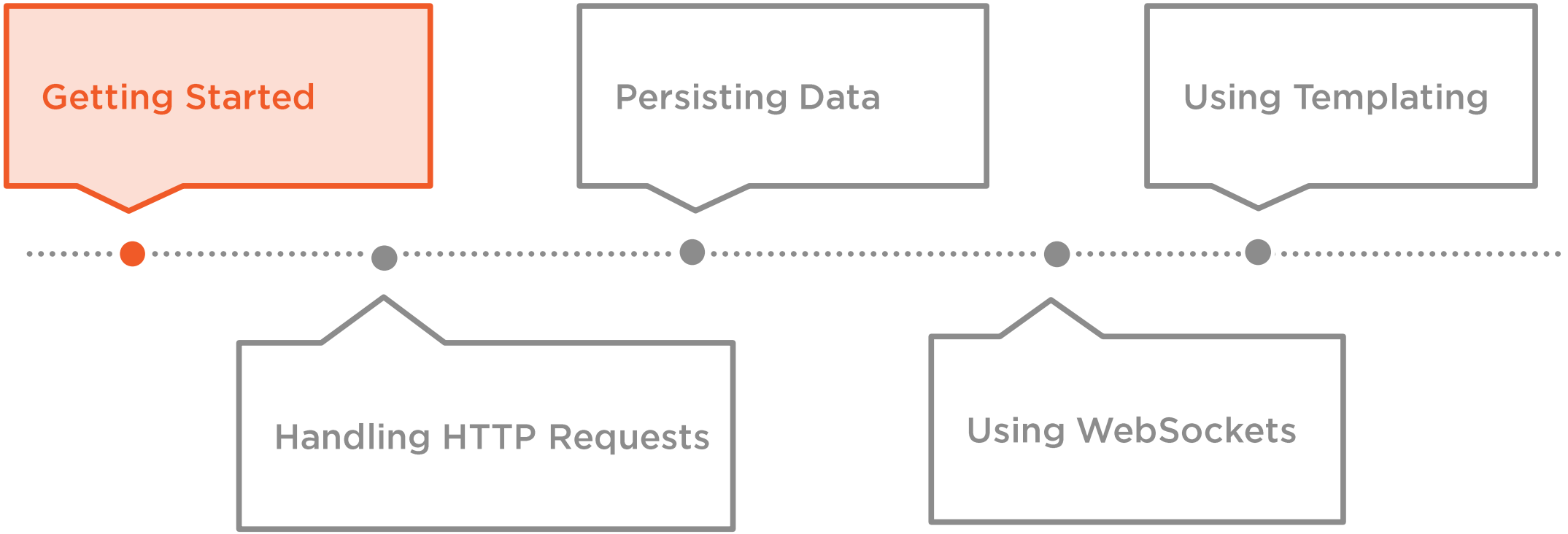
```
import “html/template”


tmpl := “{{range &index, $element := .}}{{if mod $index 2}}{{.}}{{end}}{{end}}”
func main() {
  items := []string{“one”, “two”, “three”}
  fm := template.FuncMap{“mod”: func(i, j int) bool {return i%j == 0 }}
  tmpl, _ := template.New(“tmplt”).Funcs(fm).Parse(tmpl)
  err := tmpl.Execute(os.Stdout, post)
}
```

onethree

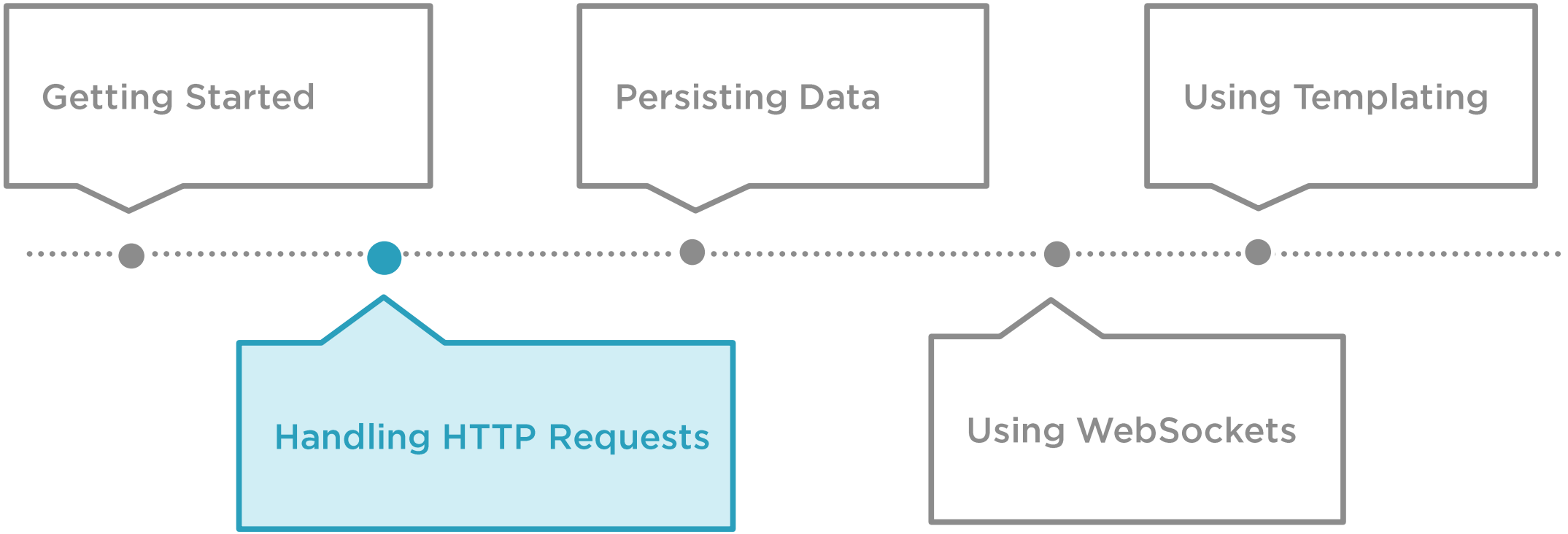
Summary

- Templating Basics
- Pipelines
- Looping
- Global Functions and Operators
- Custom Functions

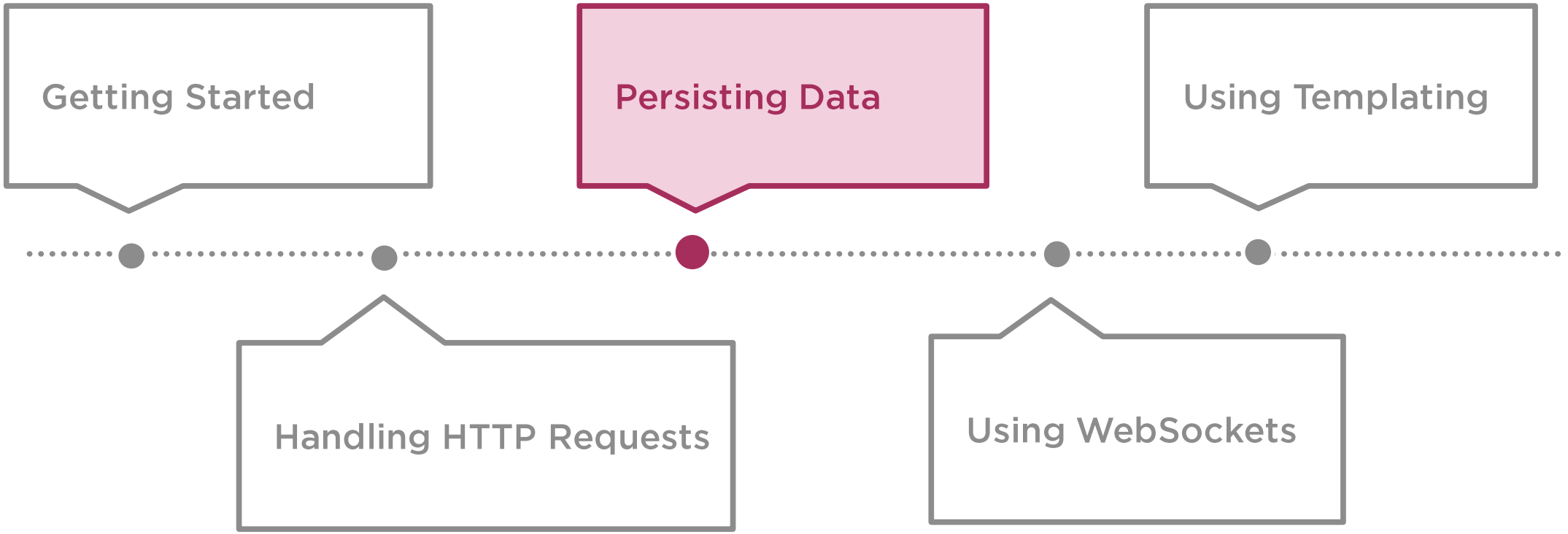
Course Summary



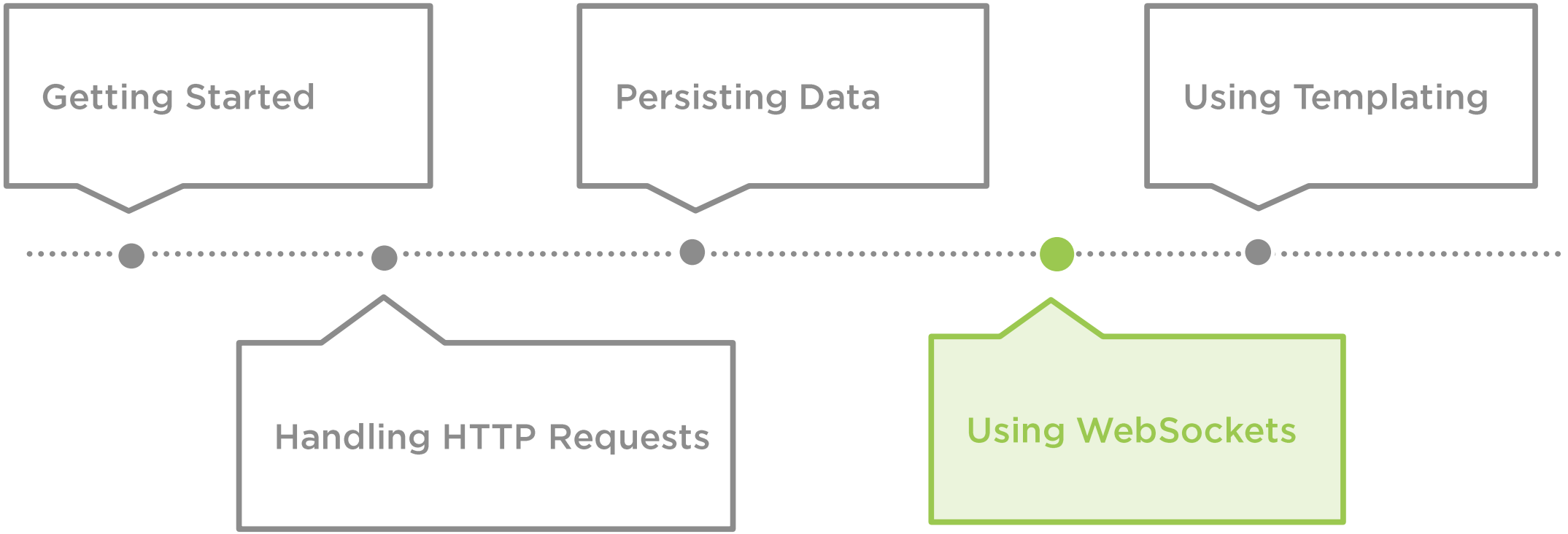
Course Summary



Course Summary



Course Summary



Course Summary