# Introduction to Htmx with Go

FRANCIS LAVOIE

# WHO AM I

- Frontend lead
- I work with JS/TS, React, Nodes and a bit of Go

- www.github.com/francisl
- www.twitter.com/Flavoie
- www.linkedin.com/in/lavoiefrancis/
- https://ferlab.bio

# What is htmx?

- htmx gives you access to AJAX, CSS Transitions, WebSockets and Server Sent Events directly in HTML, using attributes, so you can build modern user interfaces with the simplicity and power of hypertext

- htmx is small (~14k min.gz'd), dependency-free, extendable, IE11 compatible & has reduced code base sizes by 67% when compared with react

# Why htmx?

- Do more with less
- Play wells with others
- Lean & mean
- Still responsive, but relies on backend
- Fast

# When to use Htmx

## Probably all your project

- Simple to complex

## The exception will be

- Requires a lot of client side states and interactivities
  - Game
  - Editor (image, video)
  - Graphic design
- Requires offline support

# Why Go

- Run fast
- Reliable
- Easy to understand
- Productive
- Great choice of libraries

# What will we cover

## We will cover

- Htmx basic functionalities
- How Htmx works
- Create a basic application

## We will not cover

- Htmx more advance features
- Htmx Edges cases
- Go specifics, concurrencies, ...
- Beautiful UI

# Getting started - Requirements

- Go compiler
- Git

# Getting started - New project

- `mkdir gohtmx`
- `cd gohtmx`
- `go mod init gohtmx`
- `go install github.com/cosmtrek/air@latest`
- `go get github.com/a-h/templ`
- `go get github.com/labstack/echo/v4`
- `touch main.go`

Or

- `git clone git@github.com:francisl/htmx-go-intro.git -b intro-1`

# Getting started – The Go server

```
package main

import (
	"github.com/labstack/echo/v4"
	"htmx-go-intro/views"
)

func main() {
	e := echo.New()
	e.Static("/public", ”public")
	e.GET("/*", func(c echo.Context) error {
		component := views.Ticket()
		return component.Render(c.Request().Context(), c.Response().Writer)
	})
	e.File("/", "public/index.html")
	e.Logger.Fatal(e.Start(":4000"))
}
```

# Getting started - Frontend

- `mkdir public public/css public/js`
- `touch public/index.html`
- `touch public/css/style.css`
- `wget https://cdn.jsdelivr.net/npm/@picocss/pico@2/css/pico.min.css -O public/css/pico.min.css`
- `wget https://unpkg.com/htmx.org@1.9.10 -O public/js/htmx.js`

Or

- `git clone git@github.com:francisl/htmx-go-intro.git -b intro-1`

# Getting started - Index Page

```
<!doctype html>
<html lang="en">
    <head>
        <title>htmx-intro</title>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <script src="public/js/htmx.js"></script>
        <link rel="stylesheet" href="public/css/pico.min.css" />
    </head>
    <body>
        <div>
            <button hx-get="/person" hx-swap="outerHTML">Poke Person</button>
        </div>
    </body>
</html>
```

# Getting started – ticket.templ

```
package views

import "strconv"


templ Ticket(count int) {
     <div>
          if count > 1 {
               { strconv.Itoa(count) } Available Tickets
          } else if count == 1 {
               { strconv.Itoa(count) } Available Ticket
          } else {
               No Available Ticket
          }
     </div>
}
```

# Templ compositions

```
package main

templ heading() {
    <h1>Heading</h1>
}

templ layout(contents templ.Component) {
    <div id="heading">
        @heading()
    </div>
    <div id="contents">
        @contents
    </div>
}
```

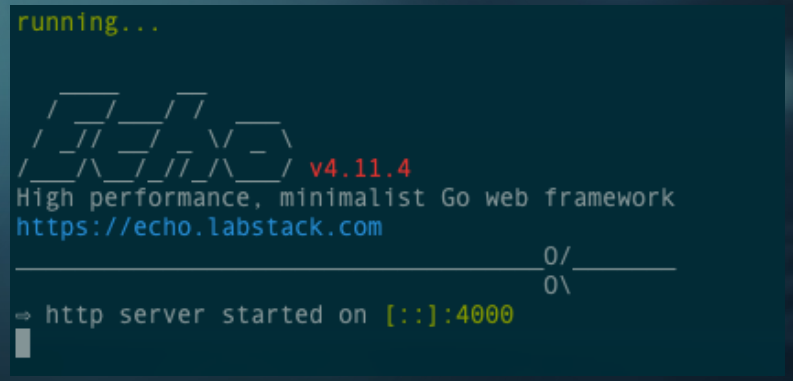
```
templ paragraph(contents string) {
    <p>{ contents }</p>
}


templ root() {
    @layout(paragraph("Dynamic
contents"))
}
```

# Getting started – Run the dev server

## Run Air to hot reload the go server

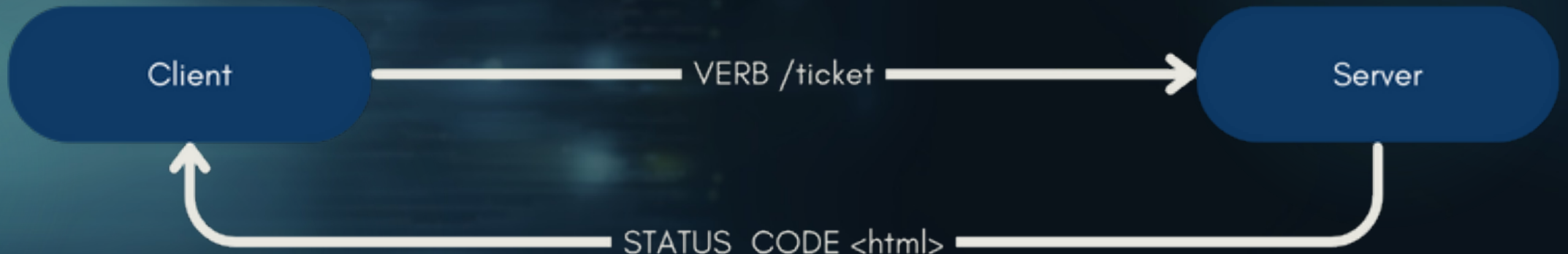
- air




## Run the templ compiler

- templ generate -watch

# Htmx Flow



# Htmx hx-target

```
<input
      type="text"  name="q"
      hx-get="/search"
      hx-trigger="keyup delay:500ms changed"
      hx-target="#search-results"
      placeholder="Search..."  >
 <div id="search-results"></div>
```

Options:

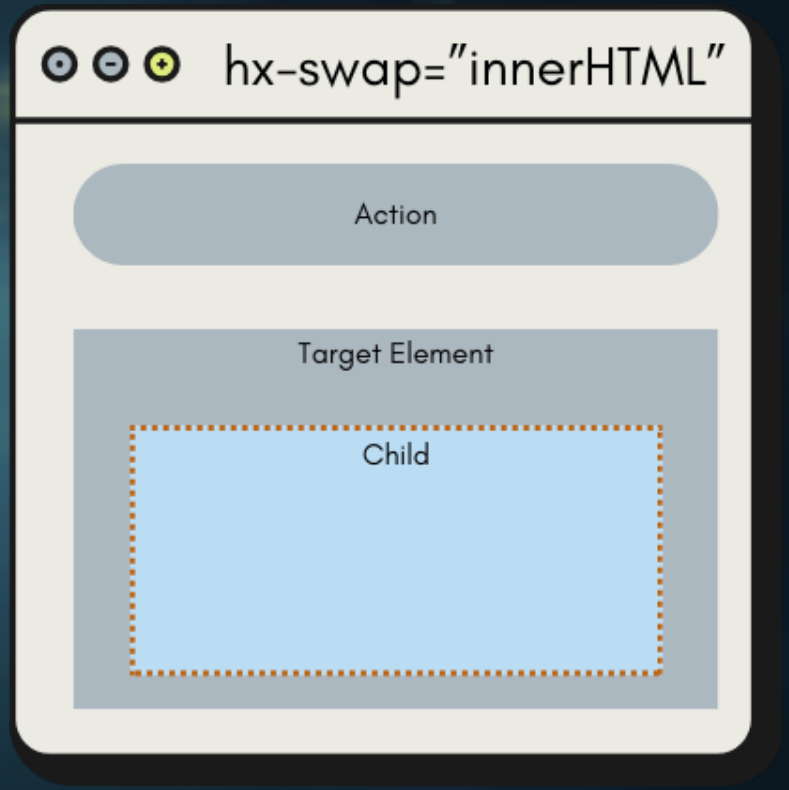
- Default or “this” keyword replace the element
- Any css selector (ids, classes, etc)

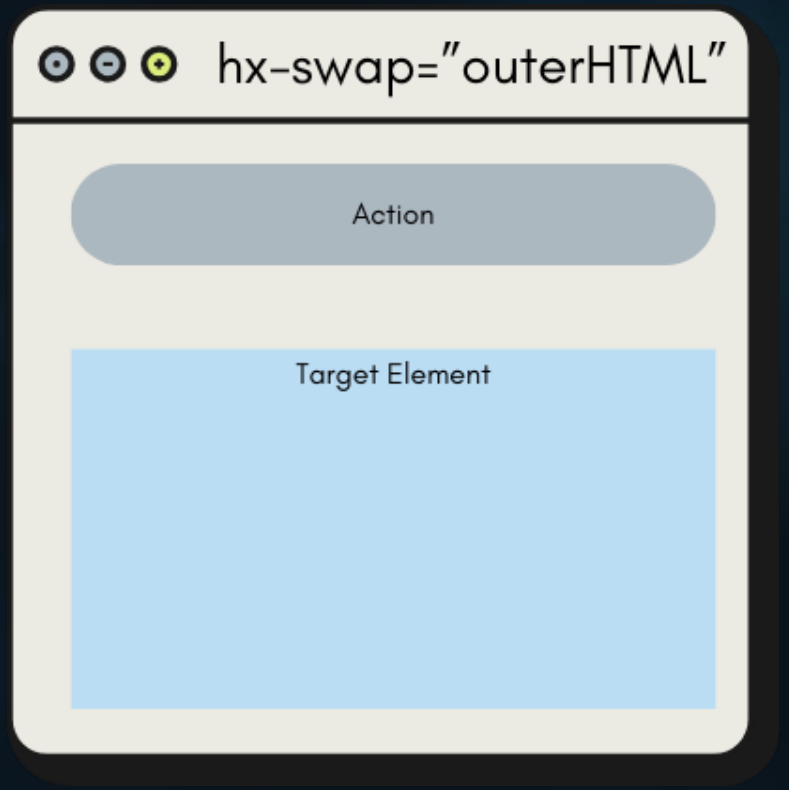
# Htmx hx-swap

```
<input
      type="text"  name="q"
      hx-get="/search"
      hx-trigger="keyup delay:500ms changed"
      hx-target="#search-results"
      placeholder="Search..."  >
 <div id="search-results"></div>
```

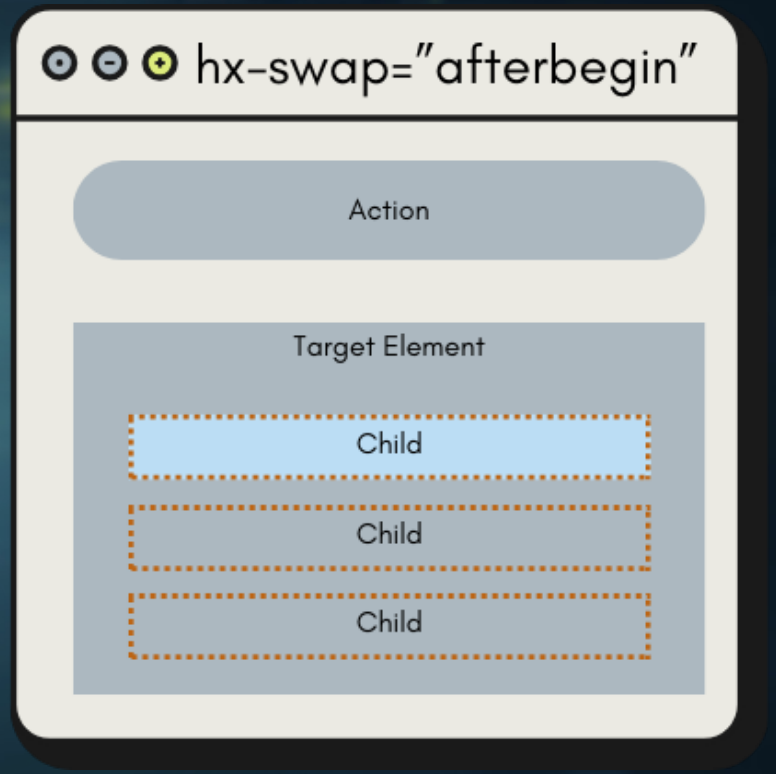
- innerHTML – (default)
- outerHTML
- beforebegin
- afterbegin
- beforeend
- afterend
- delete
- none

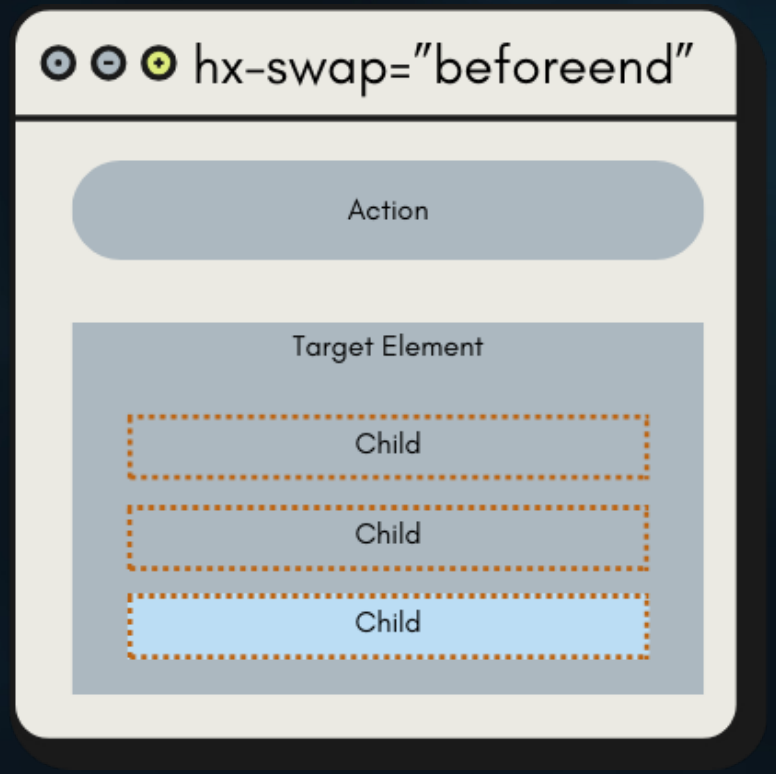
# Htmx swap innerHTML & outerHTML



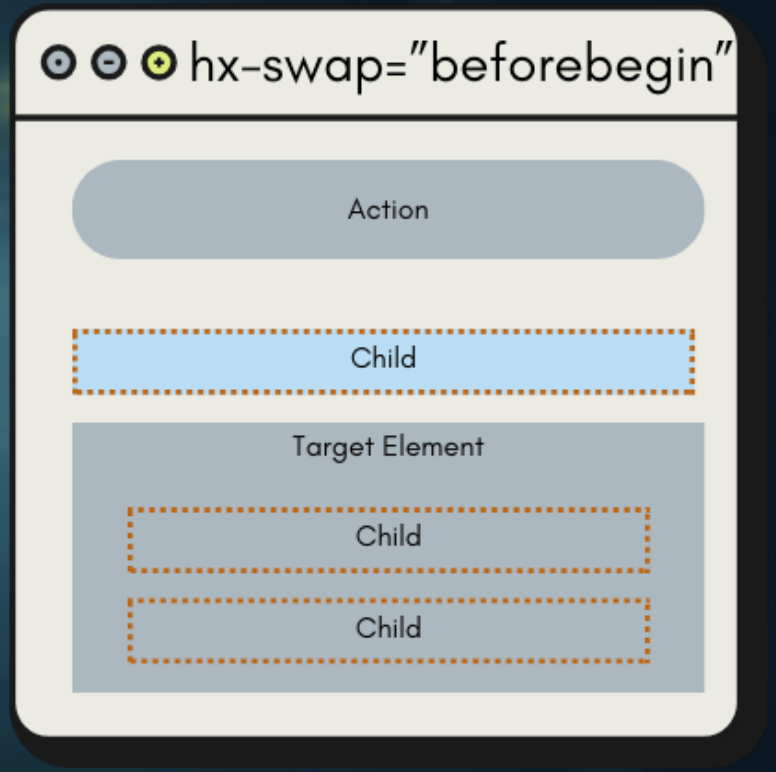



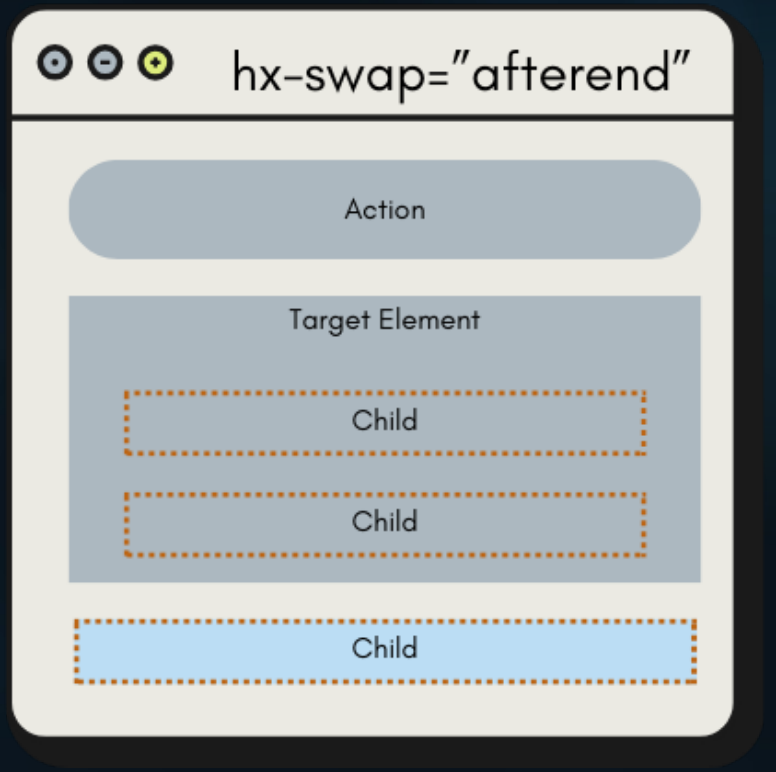
# Htmx swap afterbegin & beforeend



# Htmx swap beforebegin & afterend



# hx-* demo

```
git clone git@github.com:francisl/htmx-go-intro.git -b intro-2
```

# Htmx hx-trigger

## Modifiers

- once
- changed
- delay:<timing declaration>
- throttle:<timing declaration>
- from:<Extended CSS selector>
- And more

## Standard Events

- hx-trigger="click"
- hx-trigger="click[ctrlKey]"
- hx-trigger="click[checkGlobalState()]"

https://htmx.org/attributes/hx-trigger/

# Demo – Create a Review Form

# Htmx with React

## Htmx template

```
<!DOCTYPE html>
<html lang="en">
<body>
     <header></header>
     <div id="main-content" />
     <footer id="main-footer">footer</footer>
</body>
<script
     type="text/javascript”
     src="http://domain/static/js/bundle.js">
</script>
</html>
```

## React

```
ReactDOM.render(
     <React.StrictMode>
     <App lang="en" />
     </React.StrictMode>,
     document.getElementById('main-content'),
);
```

# Build Desktop Application


Build beautiful cross-platform applications using Go

https://wails.io

# Use web components


Reusable frontend components

# Thank You!

# Links

- Htmx https://htmx.org
- Go Language https://go.dev/dl/
- Air : Go hot reload https://github.com/cosmtrek/air
- Templ : Excellent template engine https://templ.guide/
- Echo :  web framework https://echo.labstack.com
- Wails : https://wails.io
- Lit : https://lit.dev

# Further Reading

- Hypermedia systems https://hypermedia.systems/hypermedia-reintroduction/
- Hypertext https://en.wikipedia.org/wiki/Hypertext