Go: Creating Your First Web API

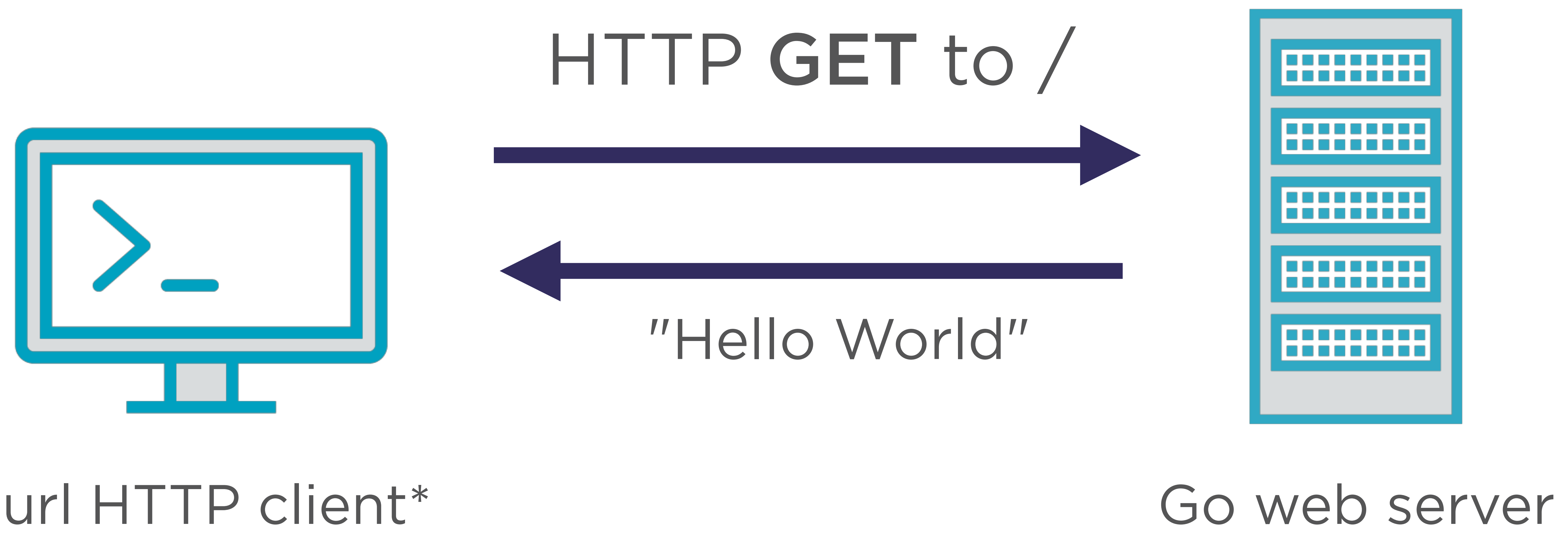
WRITING A SIMPLE WEB API ENDPOINT IN GO

Carlos Souza
PLURALSIGHT AUTHOR
@caike

What we'll learn

- How to write a request handler
- How to map a URL path to a handler
- How to bind to a network port

What we'll build



* https://curl.haxx.se/download.html

The request handler

The request handler function is responsible for handling incoming web requests.

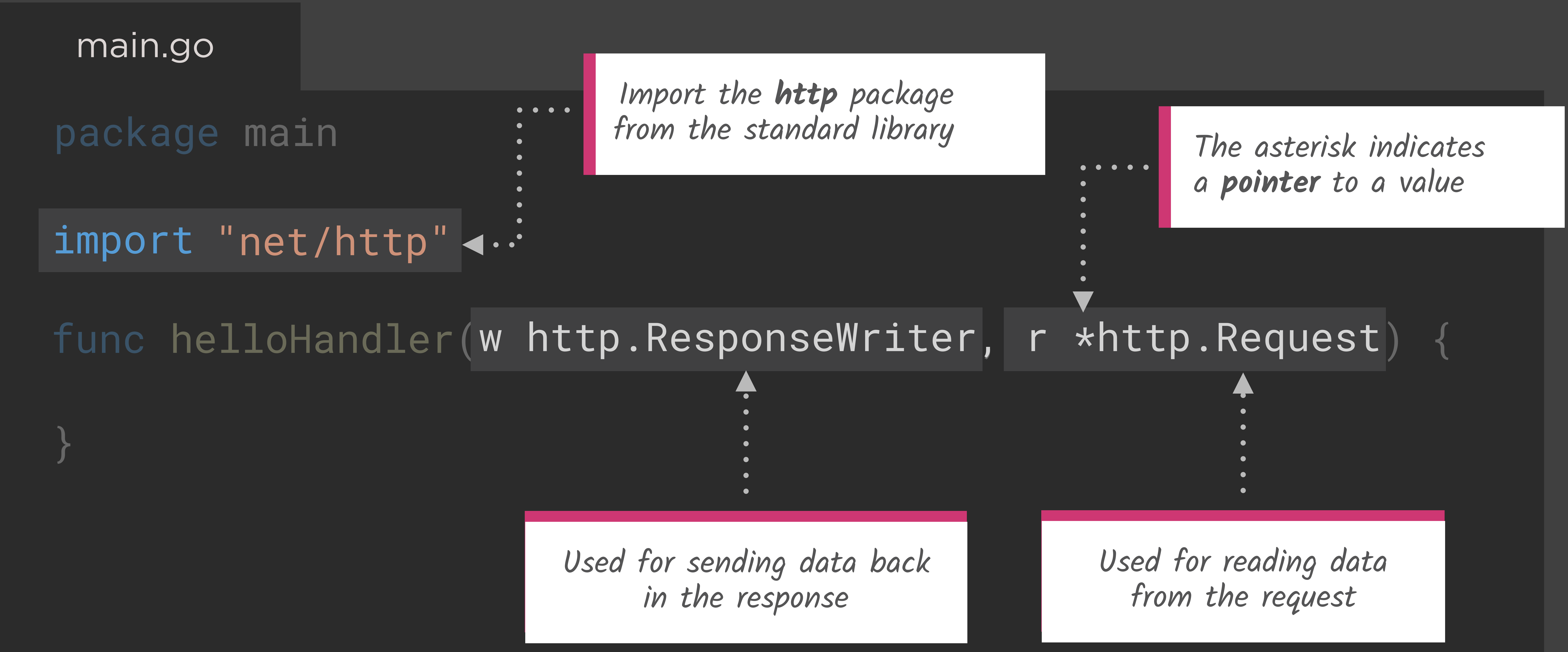
main.go

```go
package main


func helloHandler(                              ,                    ) {

}
```

Can be called anything

The request handler

The function takes two arguments: the **response writer** and **the request.**



main.go

```go
package main

import "net/http"

func helloHandler(w http.ResponseWriter, r *http.Request) {

}
```

*Import the **http** package from the standard library*
*The asterisk indicates a **pointer** to a value*
Used for sending data back in the response
Used for reading data from the request

Writing the response

We use the Fprintf() function to write data to the **response writer.**

main.go

```go
...
import (
  "fmt"
  "net/http"
)

func helloHandler(w http.ResponseWriter, r *http.Request) {
  fmt.Fprintf(w, "Hello world\n")
}
```

*Import the **fmt** package from the standard library*

*Takes the **response writer** as its first argument*

New line character makes output more readable

Current progress

✓ – How to write a request handler

– How to map a URL path to a handler

– How to bind to a network port

Routing requests

the second argument

The HandleFunc() function routes **URL paths** to **function handlers.**

the first argument

main.go

```
...
import (
  "fmt"
  "net/http"
)
func main() {
  http.HandleFunc("/", helloHandler)
}
func helloHandler(w http.ResponseWriter, r *http.Request) {
  fmt.Fprintf(w, "Hello world\n")
}
```

Routes the root path...

...to our handler function.

Listening on the network

The ListenAndServe() function starts the server and listens on the given port.

main.go

```
...
import (
  "fmt"
  "net/http"
)
func main() {
  http.HandleFunc("/", helloHandler)
  http.ListenAndServe(":8080",     )
}
func helloHandler(w http.ResponseWriter, r *http.Request) {}
```

Port number must be a string starting with a colon ":"

The Multiplexer

The HTTP multiplexer function dispatches URL paths to function handlers.

main.go

```
...
import (
  "fmt"
  "net/http"
)
func main() {
  http.HandleFunc("/", helloHandler)
  http.ListenAndServe(":8080", nil)
}
func helloHandler(w http.ResponseWriter, r *http.Request) {}
```

*Typically **nil**, in which case a default implementation is used. Most times, you will NOT need to set this value.*

Making requests

The **curl** tool, available in most systems, makes HTTP requests from the command line.

main.go

```go
...
import (
  "fmt"
  "net/http"
)
func main() {
  http.HandleFunc("/", helloHandler)
  http.ListenAndServe(":8080", nil )
}
func helloHandler( ) {   }
```

Console

```
$ curl localhost:8080

> Hello World
```

What we’ve learned

- ✓ – How to write a request handler
- ✓ – How to map a URL path to a handler
- ✓ – How to bind to a network port