Gin 1 Fundamentals

Introduction

Michael Van Sickle

@vansimke

Version Check

This version was created by using:

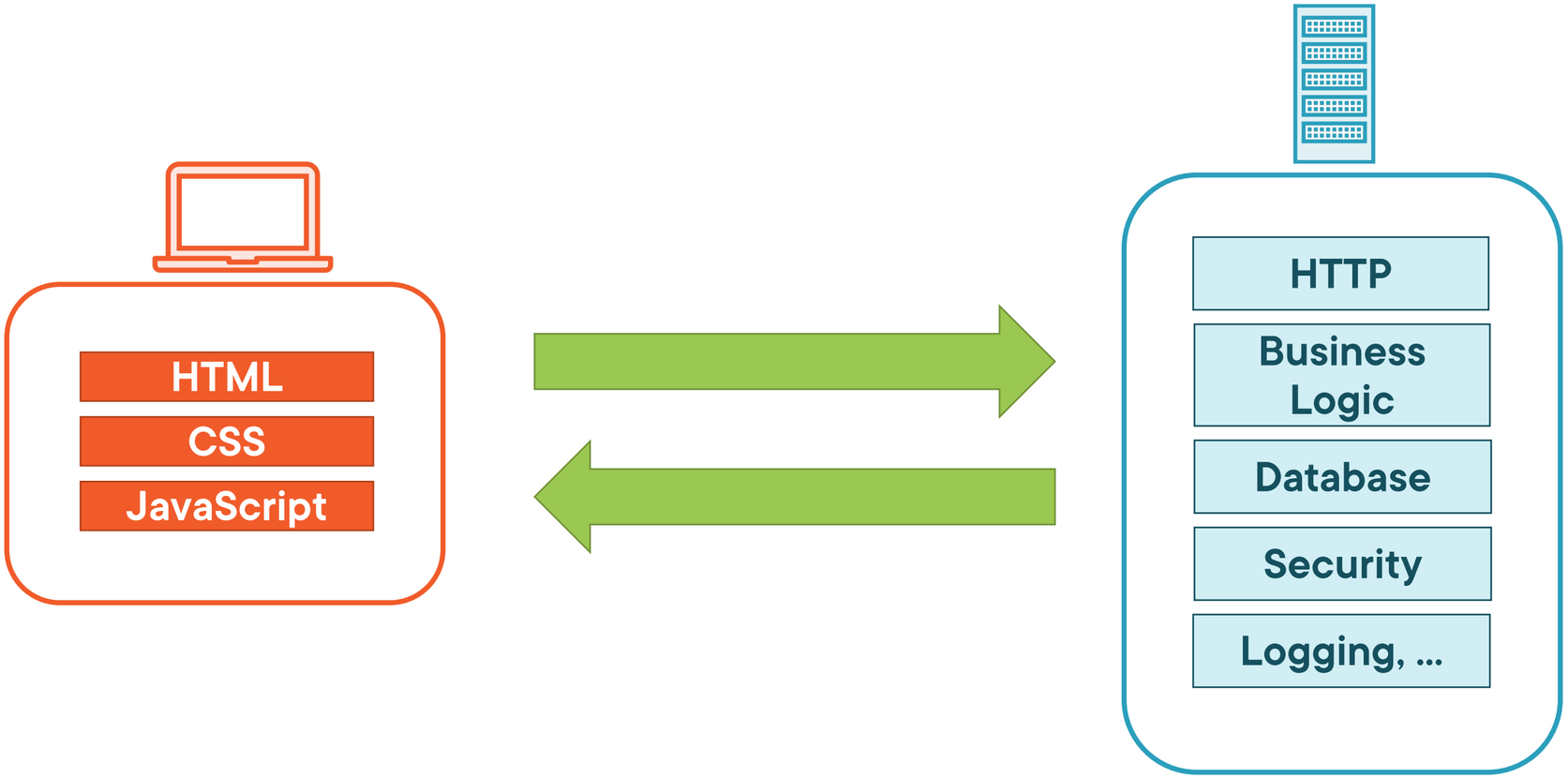
- Gin 1.7.7

Version Check

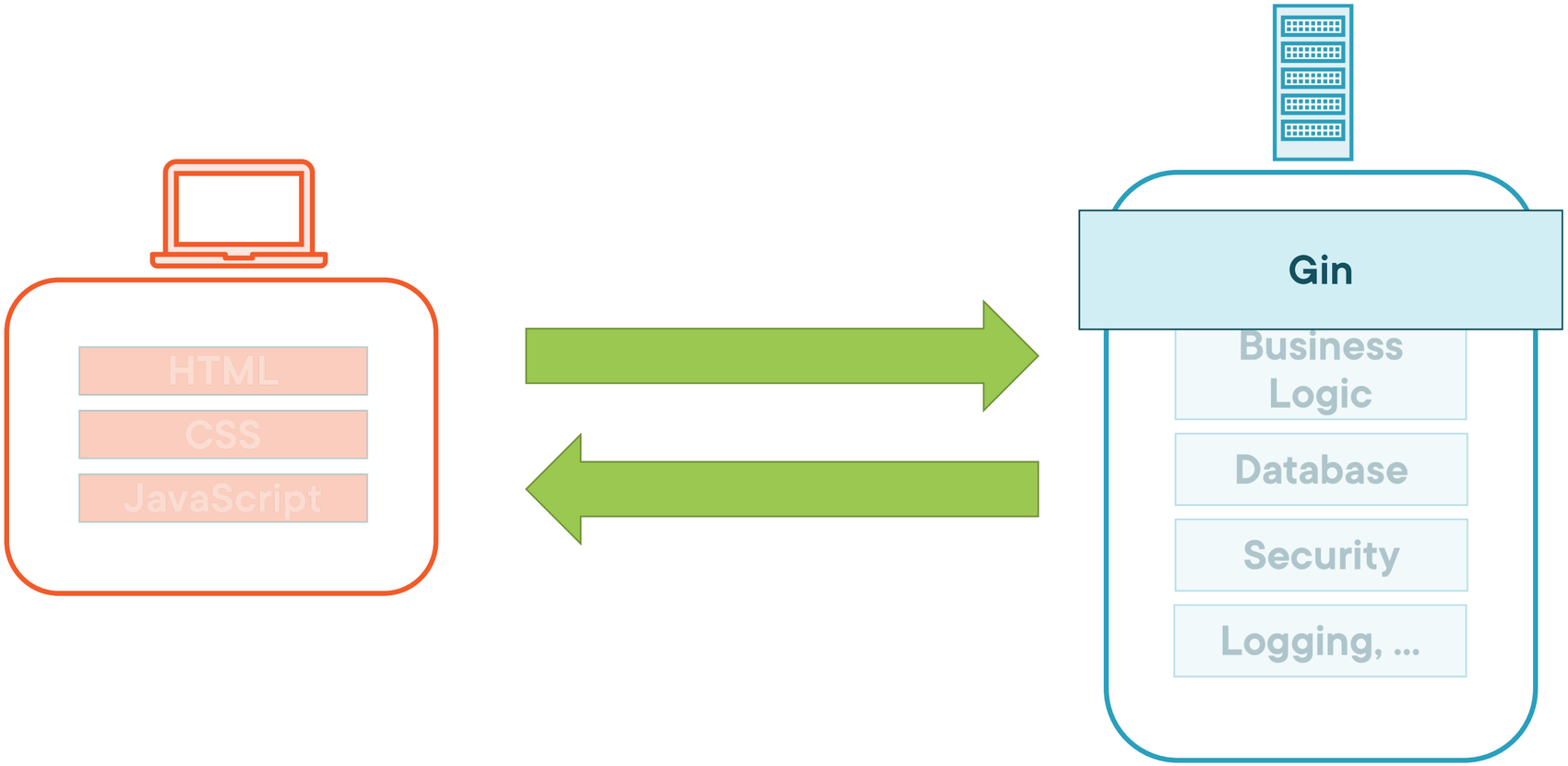
This course is 100% applicable to:

- Gin 1.6 through Gin 1.7

Parts of a Web Application



Parts of a Web Application



Advantages of Gin - JSON

```
// standard library
data, err := json.Marshal(obj)
if err != nil {
    // handler error
}
response.Write(data)
response.WriteStatus(http.StatusOk
```

Gin

```
c.JSON(http.StatusOK, obj)
```

Advantages of Gin - Routing

```
// standard library
routePattern := regexp.MustCompile(`^\/users\/(\S+?)$`)
func handler(w http.ResponseWriter, r *http.Request) {
    if matches := routePattern.FindStringSubmatch(r.URL.Path); len(matches) > 0 {
        switch r.Method {
        case http.MethodGet:
            // handle get request (finally!)
        default:
            w.WriteStatus(http.StatusMethodNotSupported)
        }
    } else {
        w.WriteStatus(http.StatusNotFound)
    }
}
```

Gin

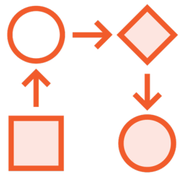
```
c.GET("/users/:username",
  func(c *gin.Context) {
    // handle get request
  }
)
```

Characteristics of Gin

Fast and flexible routing

Comprehensive data binding and validation API

Extensive rendering options



Large collection of middleware

Built-in crash recovery and error management capabilities

Prerequisites

```html
<!DOCTYPE html>
<html lang="en">
    <head></head>
    <body></body>
</html>
```

HTML
CSS

```go
package main
func main() {
    fmt.Println("Hello, world!")
}
```

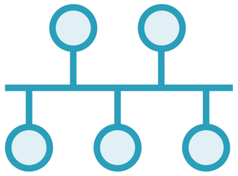
Familiarity with Go language

Course Overview

Introduction

Generating Responses

Routing Requests



Using Middleware

Working with Requests

Testing Applications