Write Basic Go Web Server

Lab learning outcomes

After completing this lab you should be able to

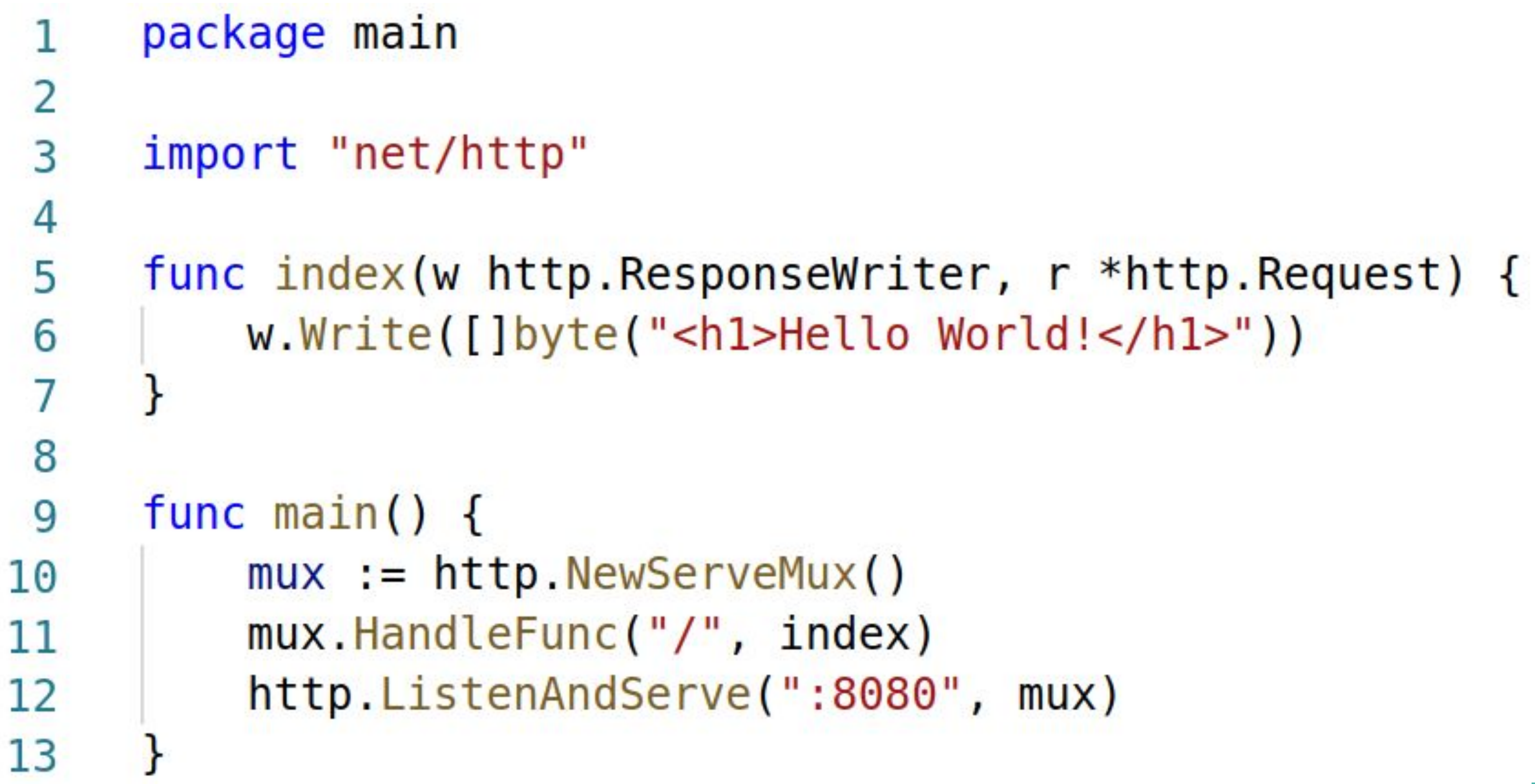
- Write **basic Go web server program** and explain the parts involved such as **`http`** and **`template`** packages; **`HandleFunc`**, **`multiplexer`**, **`ListenAndServe`**, **`FileServer`**
- Integrate Bootstrap with Go

Write Basic Go Web Server

Write the code shown in the next slide in a file called `server.go` and experiment the following things

1. Type `go run server.go` command at the terminal
   - The server should start listening at port `8080`
2. Open your web browser and go to http://localhost:8080
   - What did you see?
3. Insert the actual IP address of your machine's ethernet interface before the `:8080` text in your `server.go` file and restart your server
4. Repeat step 2 by replacing `localhost` with the your machine's IP address

Write Basic Go Web Server



```go
package main

import "net/http"

func index(w http.ResponseWriter, r *http.Request) {
    w.Write([]byte("<h1>Hello World!</h1>"))
}

func main() {
    mux := http.NewServeMux()
    mux.HandleFunc("/", index)
    http.ListenAndServe(":8080", mux)
}
```

Exercise

Remove line 10 in the previous code

Replace **`mux`** on line 11 with **`http`**

Replace **`mux`** on line 12 with **`nil`**

Restart the server and refresh your web browser and notice what happen and explain the reason why

Integrating Bootstrap with Go

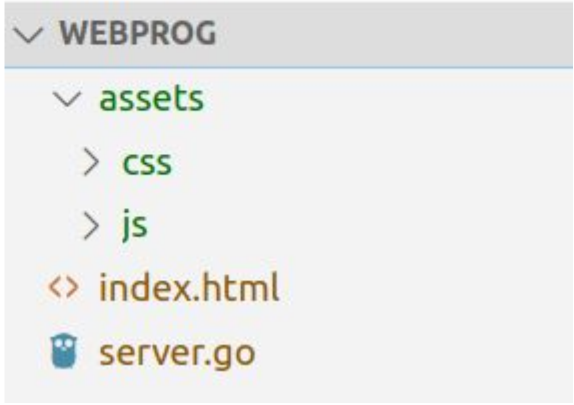
Go to the following url and download the **`Bootstrap`** library shown under the Compiled CSS and JS files section

https://getbootstrap.com/docs/4.3/getting-started/download/

In your web root directory (where the **`server.go`** file is stored) create a directory called **`assets`**

Extract the downloaded file

Place the **`css`** and **`js`** directories inside **`assets`**



```
WEBPROG
  assets
    css
    js
  index.html
  server.go
```

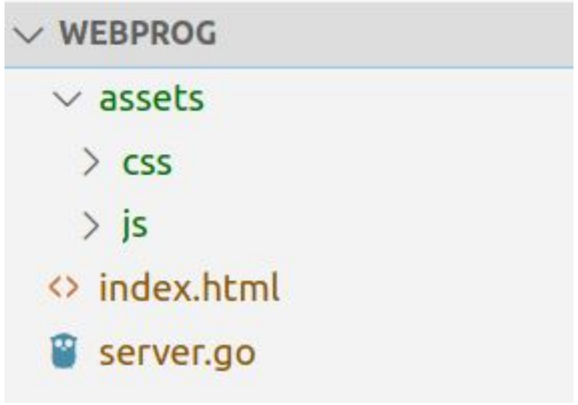
Integrating Bootstrap with Go

Bootstrap requires the `jquery` library

Go to the following URL to download the `jquery` file

> https://code.jquery.com/jquery-3.3.1.slim.min.js

Put it inside the `js` directory

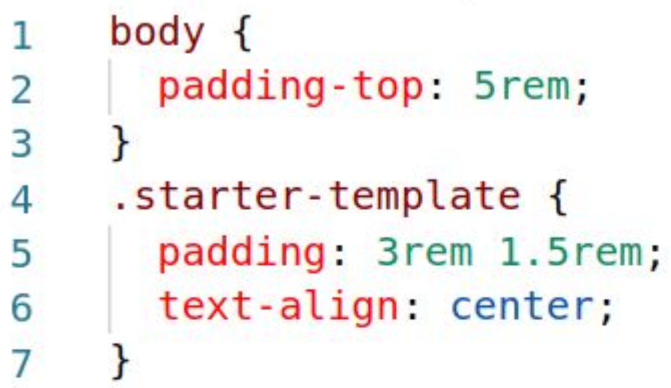


```
WEBPROG
  assets
    css
    js
  index.html
  server.go
```

Integrating Bootstrap with Go

A typical bootstrap file might also contain a custom `.css` file in addition to the ones provided by the Bootstrap library

Save the following CSS code in a file called `starter-template.css` file and put it inside the css directory



```css
body {
  padding-top: 5rem;
}
.starter-template {
  padding: 3rem 1.5rem;
  text-align: center;
}
```

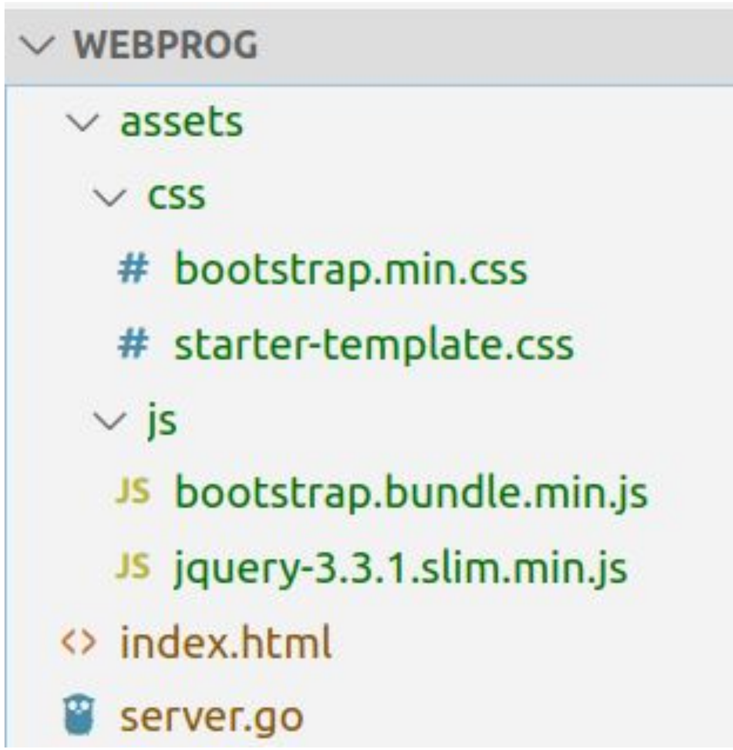
Integrating Bootstrap with Go

Now your directory structure should look like the one shown on the right side

You may want to delete the extra files found under the `css` and `js` directory except the ones shown here

Here notice that you have `assets` directory, `index.html`, and `server.go` files under the root (**WEBPROG**) directory

`css` and `js` directories are under the `assets` directory



```
WEBPROG
  assets
    css
      bootstrap.min.css
      starter-template.css
    js
      bootstrap.bundle.min.js
      jquery-3.3.1.slim.min.js
  index.html
  server.go
```

Integrating Bootstrap with Go

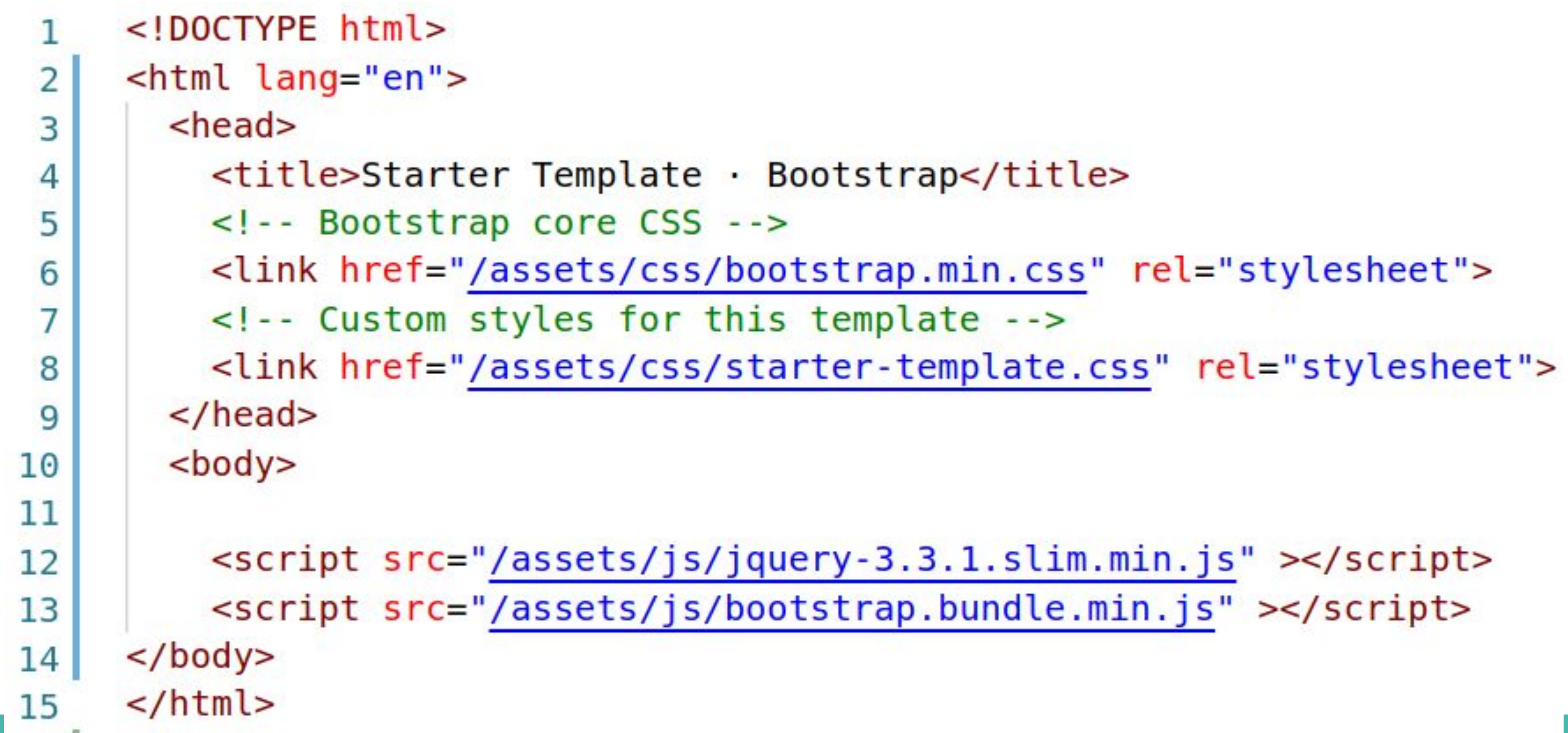
A typical Bootstrap web page might contain the following components as shown in the following slide

At the top inside the `head` section the Bootstrap and custom `.css` files are included

At the bottom before the closing `</body>` tag Bootstrap and custom `.js` files are included

Take note of the `paths` of the `.css` and `.js` files

Integrating Bootstrap with Go



```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Starter Template · Bootstrap</title>
    <!-- Bootstrap core CSS -->
    <link href="/assets/css/bootstrap.min.css" rel="stylesheet">
    <!-- Custom styles for this template -->
    <link href="/assets/css/starter-template.css" rel="stylesheet">
  </head>
  <body>

    <script src="/assets/js/jquery-3.3.1.slim.min.js" ></script>
    <script src="/assets/js/bootstrap.bundle.min.js" ></script>
</body>
</html>
```

Integrating Bootstrap with Go

The following slide shows an example Bootstrap page; you can save it in the `index.html` file

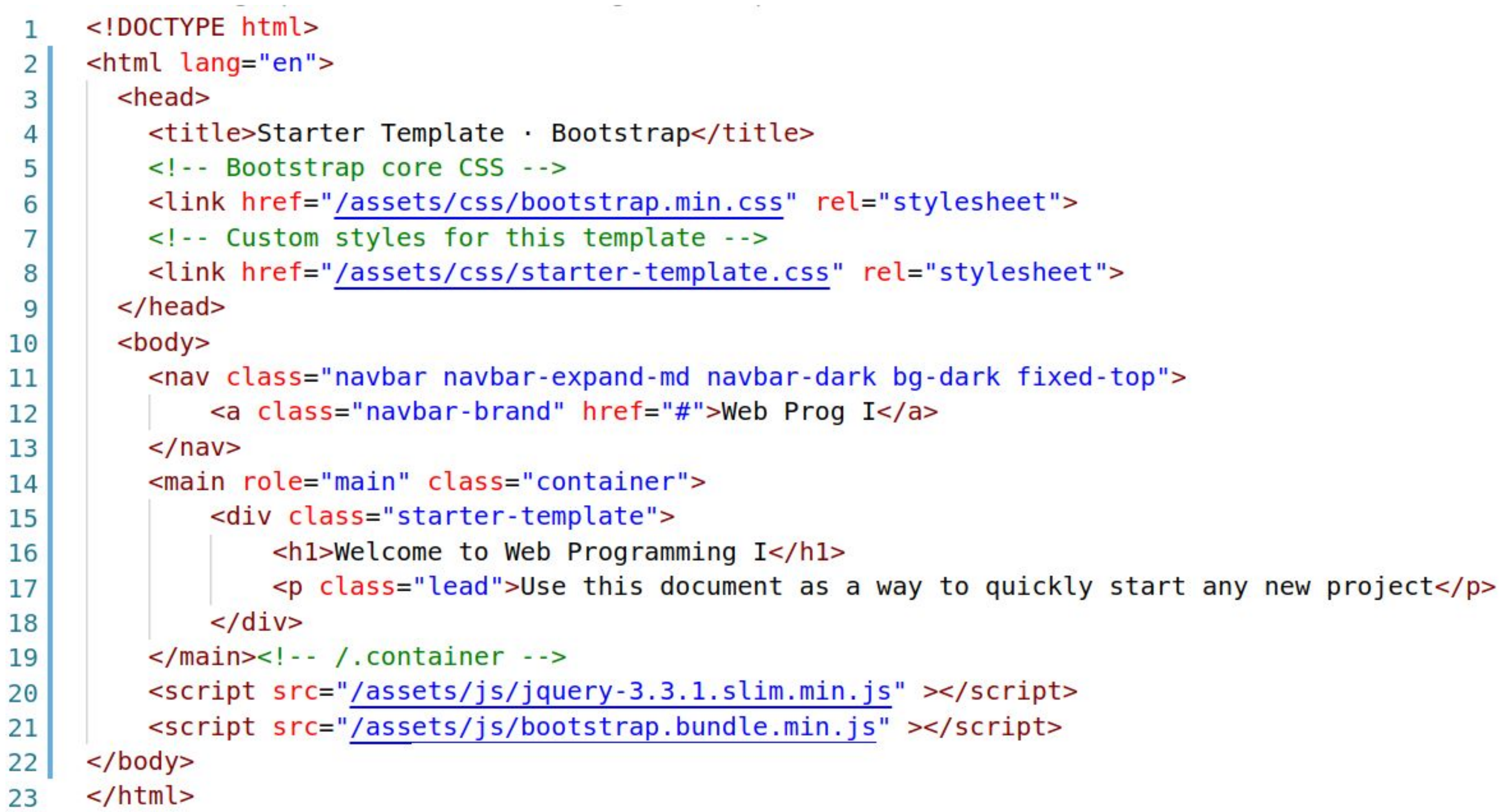
The page was adopted from the Bootstrap example pages (**Starter Template**) found in the following URL

> `https://getbootstrap.com/docs/4.3/examples/`

You can find the full code of the customized page in the following URL

> `https://github.com/betsegawlemma/web-prog-lab-03/blob/master/index.html`

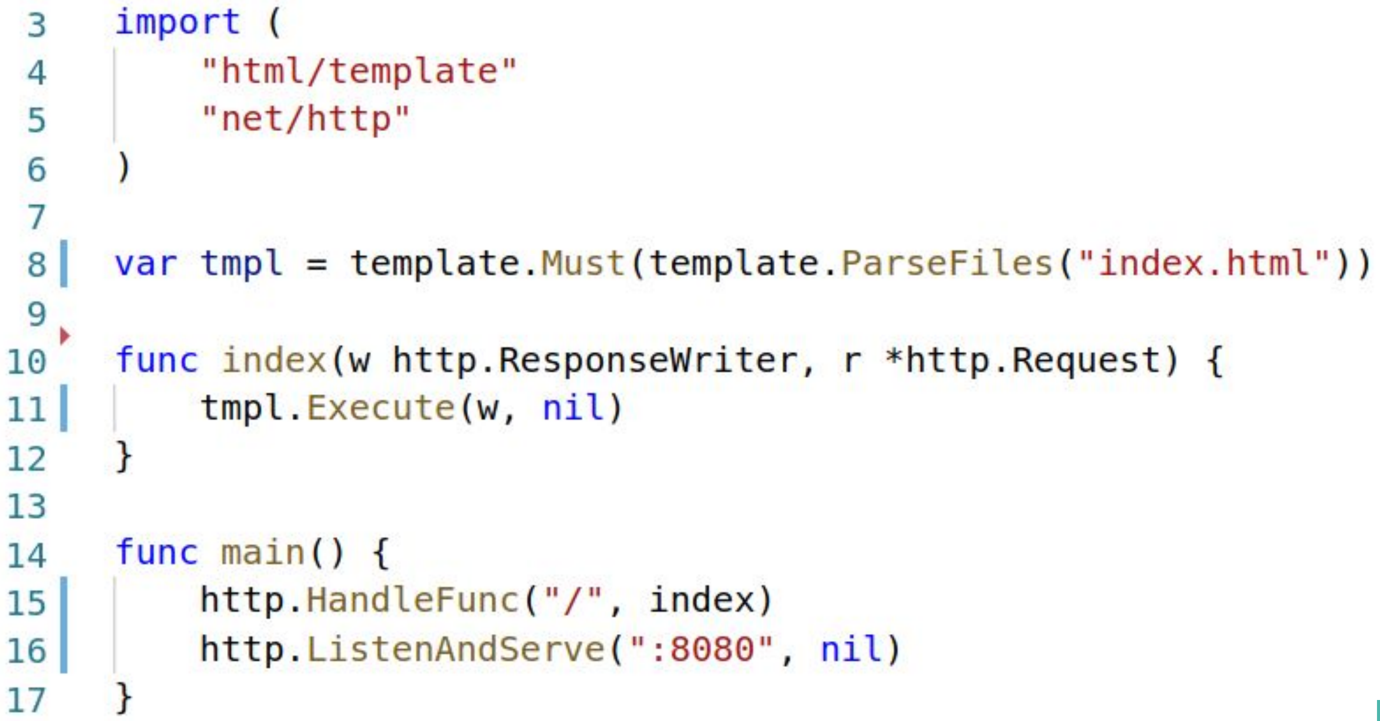
Integrating Bootstrap with Go



```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Starter Template · Bootstrap</title>
    <!-- Bootstrap core CSS -->
    <link href="/assets/css/bootstrap.min.css" rel="stylesheet">
    <!-- Custom styles for this template -->
    <link href="/assets/css/starter-template.css" rel="stylesheet">
  </head>
  <body>
    <nav class="navbar navbar-expand-md navbar-dark bg-dark fixed-top">
        <a class="navbar-brand" href="#">Web Prog I</a>
    </nav>
    <main role="main" class="container">
        <div class="starter-template">
            <h1>Welcome to Web Programming I</h1>
            <p class="lead">Use this document as a way to quickly start any new project</p>
        </div>
    </main><!-- /.container -->
    <script src="/assets/js/jquery-3.3.1.slim.min.js" ></script>
    <script src="/assets/js/bootstrap.bundle.min.js" ></script>
</body>
</html>
```

Using Go Templates

Use **`template`** package in order to serve **`HTML`** files

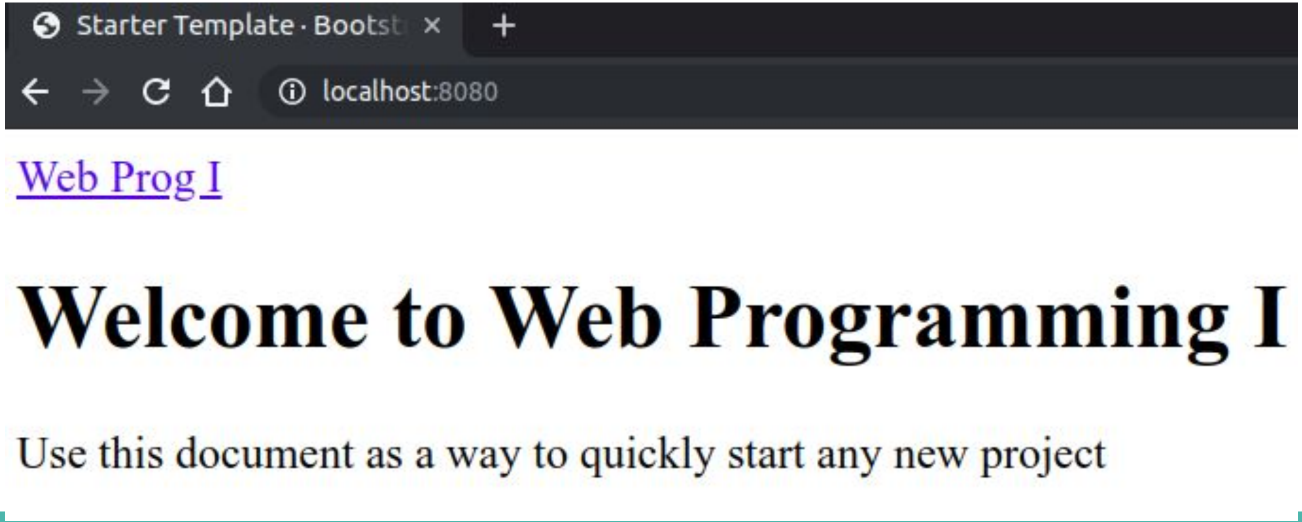


```go
import (
	"html/template"
	"net/http"
)

var tmpl = template.Must(template.ParseFiles("index.html"))

func index(w http.ResponseWriter, r *http.Request) {
	tmpl.Execute(w, nil)
}

func main() {
	http.HandleFunc("/", index)
	http.ListenAndServe(":8080", nil)
}
```

Using Go Templates

If you run your code now, you should see a page similar to the one shown below

Note that this page is not styled even though the css files were linked in the index.html page



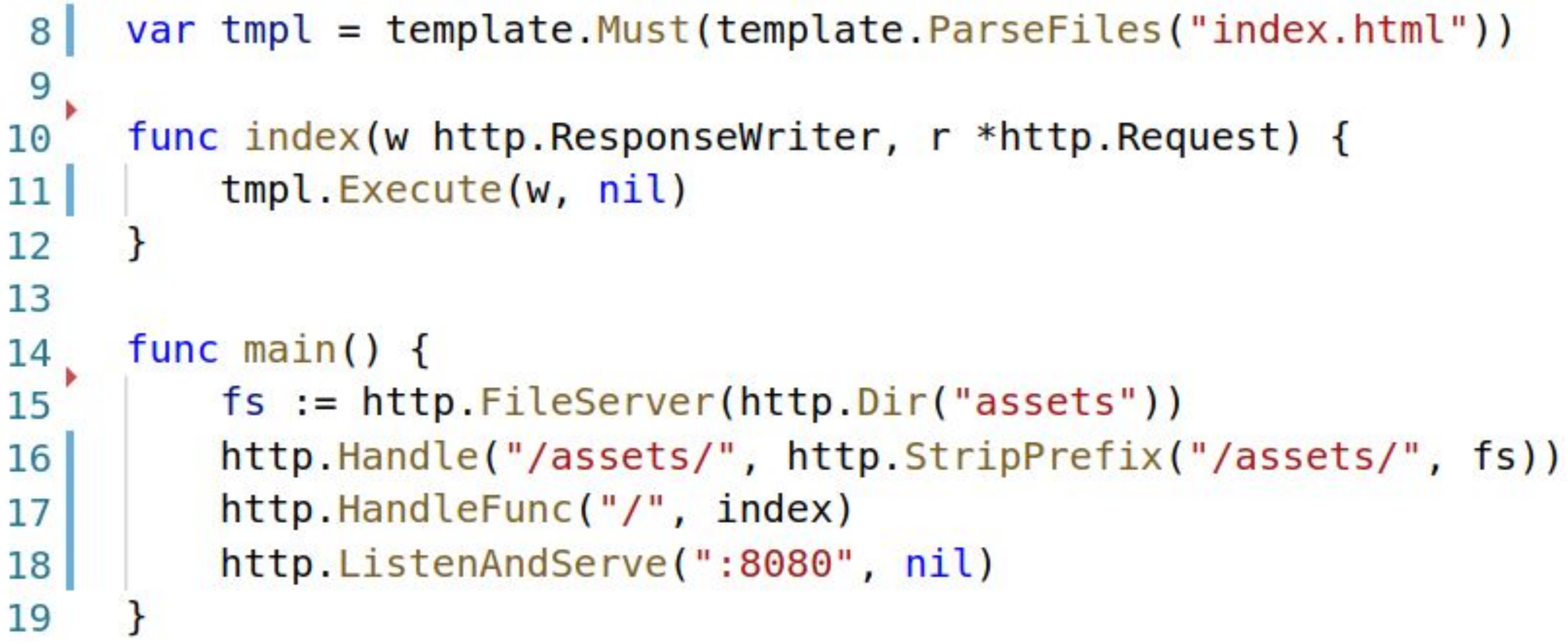
Web Prog I

Welcome to Web Programming I

Use this document as a way to quickly start any new project

Serving static files using `FileServer`

Note the added lines on line 15 and 16

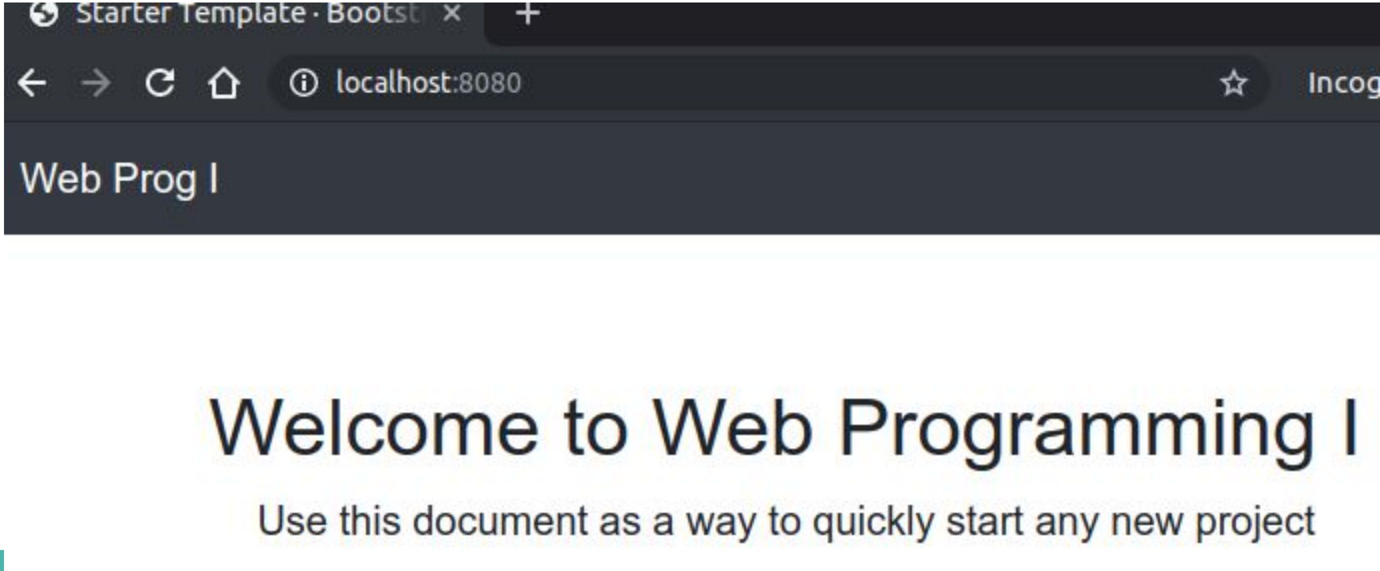


```go
 8  var tmpl = template.Must(template.ParseFiles("index.html"))
 9
10  func index(w http.ResponseWriter, r *http.Request) {
11      tmpl.Execute(w, nil)
12  }
13
14  func main() {
15      fs := http.FileServer(http.Dir("assets"))
16      http.Handle("/assets/", http.StripPrefix("/assets/", fs))
17      http.HandleFunc("/", index)
18      http.ListenAndServe(":8080", nil)
19  }
```

Serving static files using `FileServer`

If you refresh now you should see the following properly styled page

The full source code can be found here


Starter Template · Bootst
localhost:8080
Incog
Web Prog I
Welcome to Web Programming I
Use this document as a way to quickly start any new project