Template Engines

Static Middleware

```
const express = require('express')
const app = express();

app.get('/', (req, res) => {
    res.send('SOME HTML')
})

app.listen(3000, () => console.log('Server Started'))
```

If we inspect the plan text, we see that the browser automatically wrapped the entire text into a HTML. We know that whatever text we send will we wrapped into the html. So, if we send a html string, then,

```
const express = require('express')
const app = express();

app.get('/', (req, res) => {
    res.send(`<h1>My Heading</h1>`)
})

app.listen(3000, () => console.log('Server Started'))
```

The output would show ‘My Heading’ in the h1 style. So,

```
const express = require('express')
const app = express();

app.get('/', (req, res) => {
    const html = `
        <div>
            <h1>My Heading</h1>
            <ul>
                <li>One</li>
                <li>Two</li>
            </ul>
        </div>
    `

    res.send(html)
})
app.listen(3000, () => console.log('Server Started'))
```

We can see the output as,

My Heading

- One
- Two

Everything inside the body is exactly the same as we have defined the route.

Earlier we have hoisted our html using file protocol, but now we are using the server to see the html. The server is providing the content to the browser instead of us defining the content in a html file.

Let’s create another route;

```
app.get('/about', (req, res) => {
    const html = `
        <div>
            <h1>About Me</h1>
        </div>
    `
    res.send(html)
})
```

We can create multiple routes; static html content only, but we can define multiple such routes. So, now we have the home page and about page.

The user is not going to type the URL to go from home to about page. So, in the html we will create an anchor tag,

```
const express = require('express')
const app = express();

app.get('/', (req, res) => {
    const html = `
        <div>
            <h1>My Heading</h1>
            <ul>
                <li>One</li>
                <li>Two</li>
            </ul>
        </div>
        <a href='/about'>About Me</a>
    `
    res.send(html)
})

app.get('/about', (req, res) => {
    const html = `
        <div>
            <h1>About Me</h1>
        </div>
        <a href='/'>Home</a>
    `
    res.send(html)
})

app.listen(3000, () => console.log('Server Started'))
```

We can see the About Me in the /home page. When we click on it, we will go to the About page. Similarly, if we put the same thing in the /about route, as

`<a href='/'>Home</a>`

It will take us back to the Home page again.

If we give two routes in the contact page, then user can go from contact page to home page or about page.

```
const express = require('express')
const app = express();

app.get('/', (req, res) => {

    const html = `
        <div>
            <h1>My Heading</h1>
            <ul>
                <li>One</li>
                <li>Two</li>
            </ul>
        </div>
        <a href='/about'>About Me</a>
        <a href='/contact'>Contact</a>
    `
    res.send(html)
})

app.get('/about', (req, res) => {

    const html = `
```

```
        <div>
            <h1>About Me</h1>
        </div>
        <a href='/'>Home</a>
        <a href='/contact'>Contact</a>
    `
    res.send(html)
})

app.get('/contact', (req, res) => {

    const html = `
        <div>
            <h1>About Me</h1>
        </div>
        <a href='/'>Home</a>
        <a href='/about'>About</a>
    `
    res.send(html)
})

app.listen(3000, () => console.log('Server Started'))
```

This is not going to be scalable approach. The first level better approach is to create all the files. Let us create index.html and about.html page. We need to copy the content written in the const html = `` into the respective file names.

```
const express = require('express')
const app = express();

app.get('/', (req, res) => {
    res.send(html)
})

app.get('/about', (req, res) => {
    res.send(html)
})

app.listen(3000, () => console.log('Server Started'))
```

If we want to access home.html file from the address bar, it will show an error CANNOT GET /home.html. If we put home.html in the res.send, then it would show home.html. Express server is not able to configure itself, so we have to give a proper directory name i.e., we need to give the entire path from the local system.

```
const express = require('express')
const app = express();

app.get('/', (req, res) => {
     res.send('I:/Projects/AttainU/au16-lecture-coding-material/week22/d
ay-4/home.html')
})

app.get('/about', (req, res) => {
    res.send('/about.html')
})

app.listen(3000, () => console.log('Server Started'))
```

The second-level better approach would be, **`__dirname`**. This will always contain entire absolute path; wherever the file exits. The **`__dirname`** will get the entire absolute path name where we are using the variable. So, this gives us very easy way of handling. All we need to write,

```
const express = require('express')
const app = express();

app.get('/', (req, res) => {
    res.send(__dirname + '/home.html')
})

app.get('/about', (req, res) => {
    res.send(__dirname + '/about.html')
})

app.listen(3000, () => console.log('Server Started'))
```

There is another variable as **`__filename`** which has the name of the file. So, if we log

```
console.log('Server Started')

console.log('Server Started')
```

The **output** would show,

I:/Projects/AttainU/au16-lecture-coding-material/week22/day-4

I:/Projects/AttainU/au16-lecture-coding-material/week22/day-4/home.html

In one request response cycle, we can only send one file. So, there is another better approach for the same thing.

We are sending files, but the limitation is we cannot send css and html at the same time. So the first thing we need to understand is about express.static. It is an inbuilt middleware. Files which will not change randomly is called static files. We will create a folder PUBLIC and will put the about.html and home.html files in this folder. Now we need to tell our express server, that the files in the PUBLIC folder do not need to be send individually. So, we would write,

```
const express = require('express')
const app = express();

app.use(express.static('public'))

app.get('/', (req, res) => {
    res.send(__dirname + '/home.html')
})

app.get('/about', (req, res) => {
    res.send(__dirname + '/about.html')
})

app.listen(3000, () => console.log('Server Started'))
```

we can give the address and it will go directly to the page, even if we comment the res.send in both the routes. Our files in PUBLIC folder can be accessed even without the routes.

So, localhost:3000/home.html → home page and
    localhost:3000/about.html → about page and
    localhost:3000/css/style.css → style.css page opens

We can now create a js folder and inside we will save a file as browser.js having content, alert('ASD'). We link this file in our home.html page. All we have to do is in the html page,

```
<link rel="stylesheet" href="/css/style.css">
<script src="/js/browser.js"></script>
```

This is how we will serve the browser specific javascript and css. All we have to use is **`app.use(express.static("public"))`**

Never put anything private files like authentication related issues. Only files which you want everyone to access, only those should be kept in the public folder.