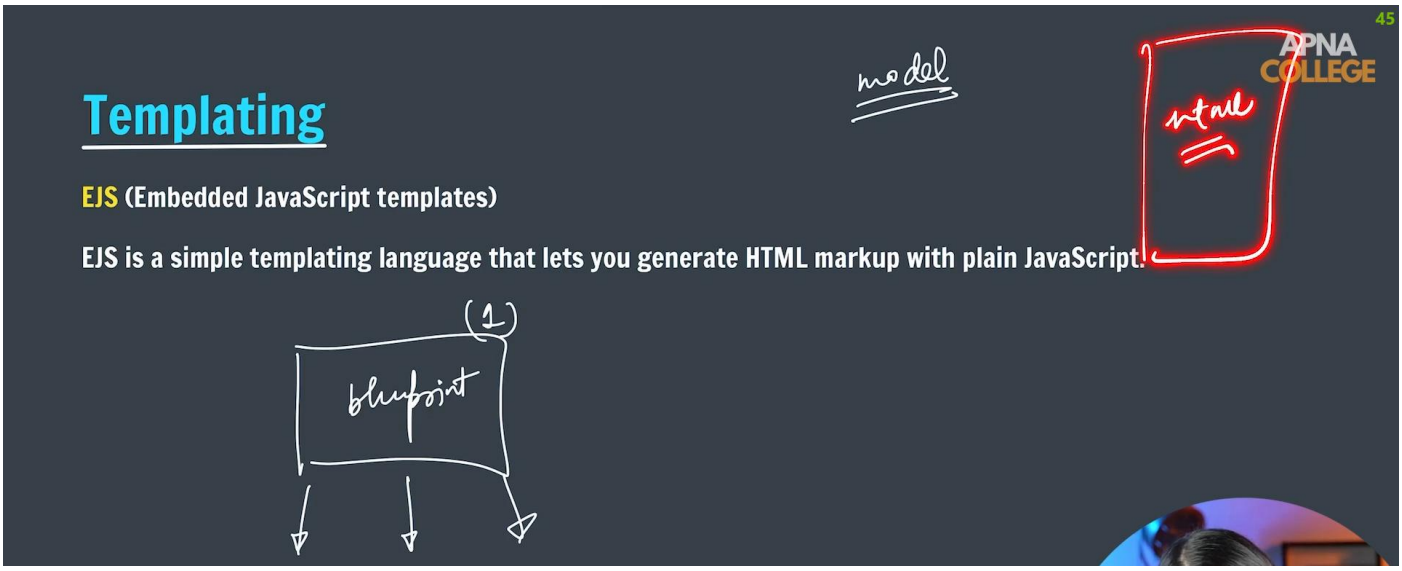
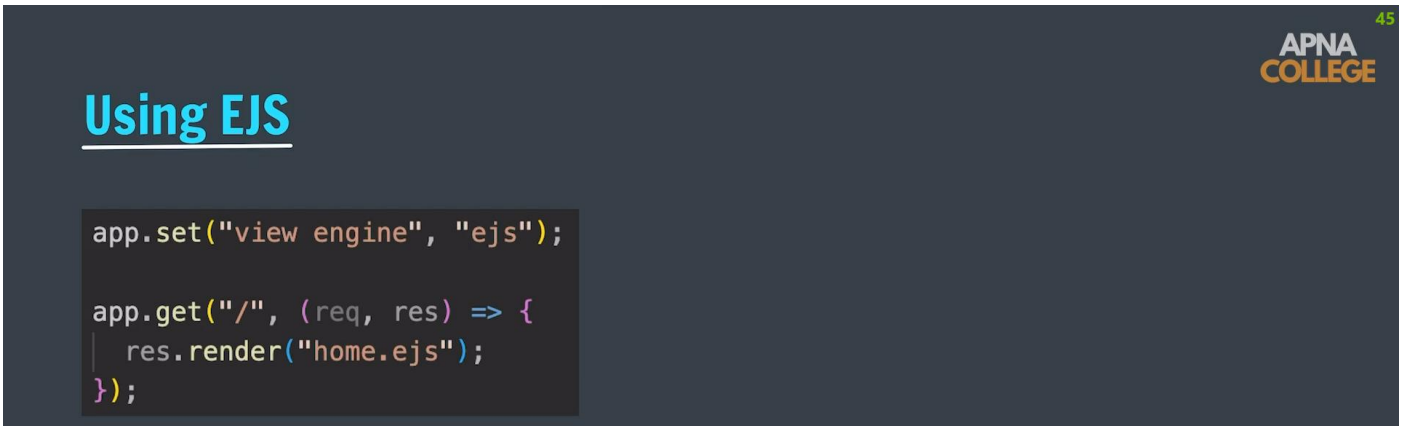
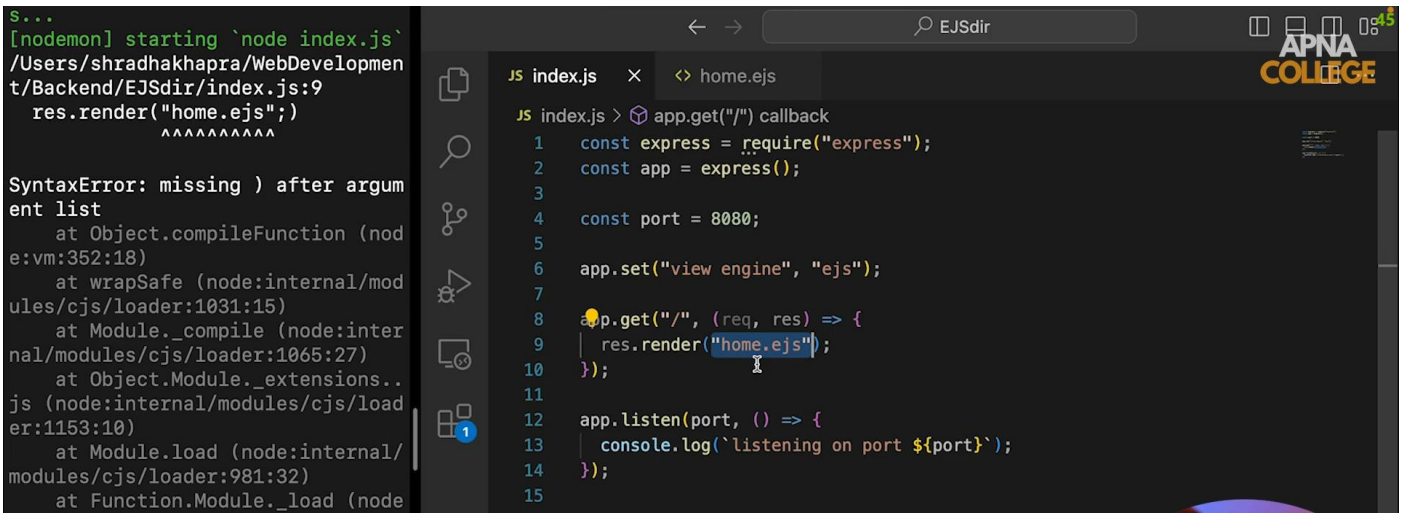
EJS




```
shradhakhapra@Shradhas-MacBook-Air
 EJSdir % npm init -y
```



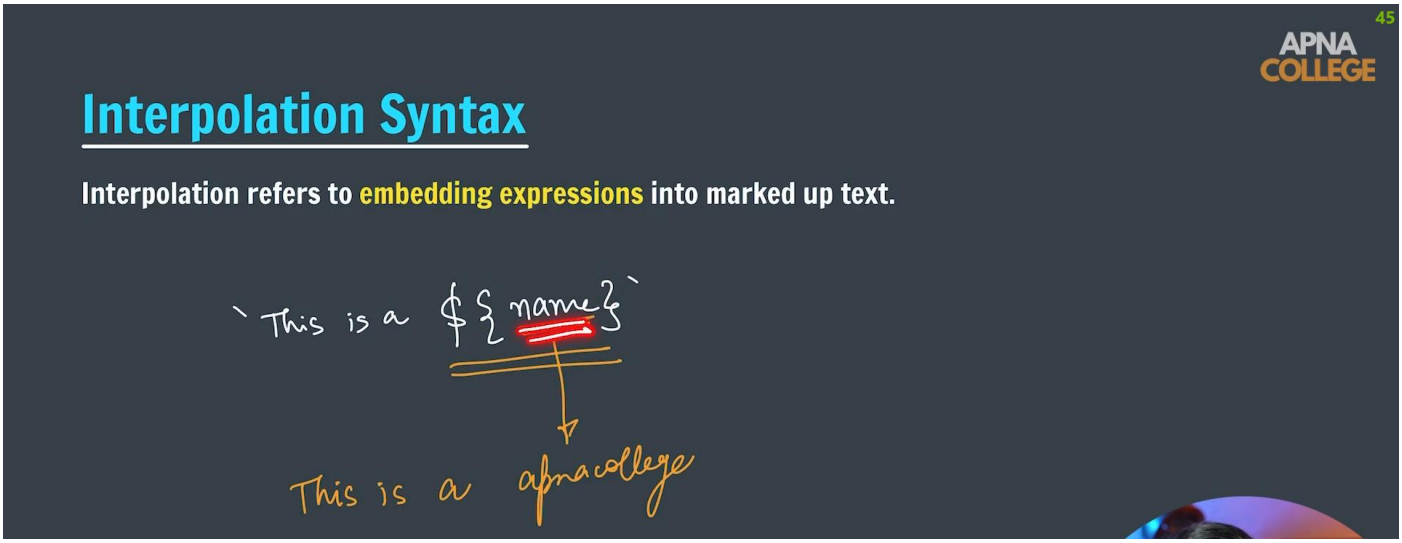


Express by default searches for the views folder and then search for home.ejs inside it. So we need to create folder called views.

Writing .ejs is optional.

Views folder is searched from where we have run our server.

So then home.ejs will not be found if the server is run from outside the folder. So to solve this we have to add these 2 lines in our code.



```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.(
    <title>Home Page</title>
  </head>
  <body>
    <h1>This is the home page 1 * 2</h1>
    <h3><%= ["hello", "bonjour", "namaste"][2] %></h3>
    <p>
      Lorem ipsum dolor sit, amet consectetur adipisicing elit. Unde,
      voluptatibus, recusandae ea laudantium accusamus ab enim necessita
      odit est nisi ut debitis beatae quasi possimus tempora dicta dist
      modi quibusdam.
    </p>
    <button>click me!</button>
  </body>
</html>
```

<%= whatever you write in between its value is calculated and then printed as HTML string %>

We can write JavaScript inside these tags.

```javascript
app.get("/rolldice", (req, res) => {
  let diceVal = Math.floor(Math.random() * 6) + 1;
  res.render("rolldice.ejs", { num: diceVal });
});
```

This can also be written as

```javascript
app.get("/rolldice", (req, res) => {
  let diceVal = Math.floor(Math.random() * 6) + 1;
  res.render("rolldice.ejs", { diceVal });
});
```

We send the diceVal as an object to that file where we can access it.

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <title>Roll Dice</title>
  </head>
  <body>
    <h1>Dice gave value : <%= num %></h1>
  </body>
</html>
```


APNA COLLEGE

Conditional Statements in EJS

Adding conditions inside EJS

```
<% if(diceVal == 6) { %>
<h2>Nice! Roll dice again.</h2>
<% } %>
```


APNA COLLEGE

Loops in EJS

```
<% for(user of followers) { %>
<li><%= user %></li>
<% } %>
```

Instagram page with EJS

const instaData = require("./data.json");

```
      comments: 19
    },
    {
      image: 'https://images.unsplash.com/photo-1577023311546-cdc07a8454d9?ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8Nnx8Y2F0c3xlbnwwfHwwfHx8MA%3D%3D&auto=format&fit=crop&w=800&q=60',
      likes: 1065,
      comments: 200
    }
  ]
}
{
  name: 'dogs',
  followers: 75000,
  following: 150,
  posts: [
    {
      image: 'https://images.unsplash.com/photo-1598133894008-61f7fdb8cc3a?ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8Nnx8ZG9nc3xlbnwwfHwwfHx8MA%3D%3D&auto=format&fit=crop&w=800&q=60',
      likes: 3000,
      comments: 41
    },
    {
      image: 'https://images.unsplash.com/photo-1552053831-71594a276
```

EJSdir

instagram.ejs ● index.js

views > instagram.ejs > html > body > ?

```
5     <meta name="viewport" content="width=device-width, initial-scale=1.
6     <title>Instagram</title>
7   </head>
8   <body>
9     <h2>This page belongs to @<%= data.name %></h2>
10    <button>Follow</button>
11    <button>Message</button>
12
13    <p>
14      Followers : <%= data.followers %>      Follow
15      data.following %>
16    </p>
17    <hr />
18
19    <% for(let post of data.posts) { %>
20    <img src="<%= post.image %>" alt="some img" />
21    <p>
22      Likes : <%= post.likes %>     
23      post.comments %>
24    </p>
25    <% } %>
26  </body>
27 </html>
28
```



```
app.get("/ig/:username", (req, res) => {
  let { username } = req.params;
  const instaData = require("./data.json");
  const data = instaData[username];
  if (data) {
    res.render("instagram.ejs", { data });
  } else {
    res.render("error.ejs");
  }
});
```


Includes → subtemplates

```
<%- include("includes/head.ejs"); %>
```

```
shradhakhapra@Shradhas-MacBook-Air
 EJSdir % nodemon index.js
[nodemon] 3.0.1
[nodemon] to restart at any time,
enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,
mjs,cjs,json
[nodemon] starting `node index.js`
listening on port 8080
```

```
instagram.ejs   error.ejs   rolldice.ejs
views > instagram.ejs > ? > body > ? > ? > ?
5     <button>Message</button>
6
7     <p>
8       Followers : <%= data.followers %>      Follow
9       data.following %>
10    </p>
11    <hr />
12
13    <% for(let post of data.posts) { %>
14    <img src="<%= post.image %>" alt="some img" />
15    <p>
16      Likes : <%= post.likes %>      comments : <%=
17      post.comments %>
18    </p>
19    <% } %>
20
21    <%- include("includes/footer.ejs") %>
22  </body>
23  <script src="/app.js"></script>
24 </html>
```

Serving Static Files

middlewares

app.use(express.static(folder_name))

```
app.use(express.static(path.join(__dirname, "public")));
```

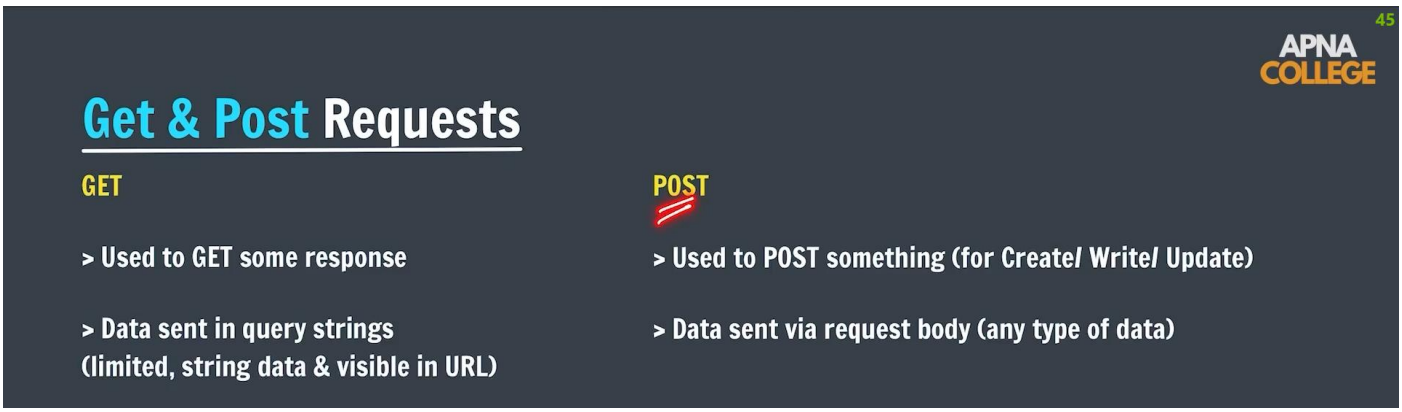
When we want to send css and js along with html as response.

We should have a folder by the name public for serving static files just like we have views folder for templates.

```
const btns = document.querySelectorAll("button");

for (btn of btns) {
  btn.addEventListener("click", () => {
    console.log("button was clicked");
  });
}
```

```
app.use(express.static(path.join(__dirname, "/public/js")));
app.use(express.static(path.join(__dirname, "/public/cs")));
app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "/views"));
```

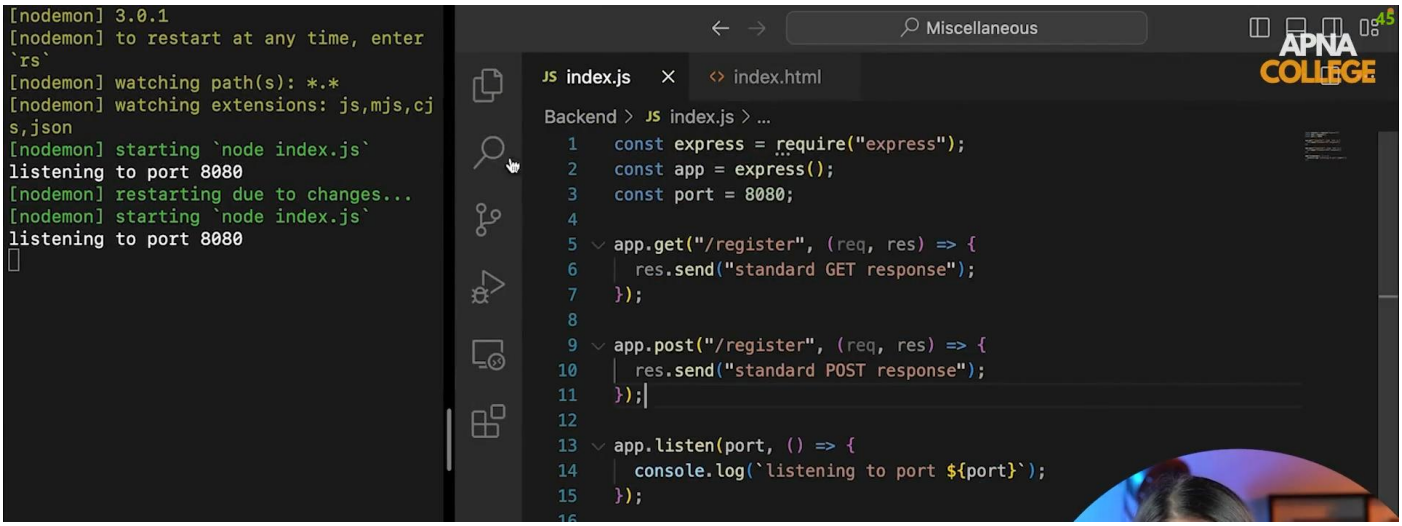
GET vs POST



In get request when we submit the data we can see the data as query string in the URL while in POST we do not see the data submitted and it is sent as the body.

Browsers have a limit on the size of the URL they can show and also showing data in the url is not safe.

By POST method we can send data of any type format like string, json etc.



```
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
listening to port 8080
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
listening to port 8080
```

Frontend > index.html > html > body > form

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale
    <title>GET & POST Requests</title>
  </head>
  <body>
    <h3>GET Request Form</h3>
    <form method="get" action="http://localhost:8080/register">
      <input placeholder="enter username" name="user" type="text" />
      <input placeholder="enter password" name="password" type="pass
      <button>Submit</button>
    </form>
    <hr />
    <h3>POST Request Form</h3>
    <form method="post" action="http://localhost:8080/register">
      <input placeholder="enter username" name="user" type="text" />
      <input placeholder="enter password" name="password" type="pass
      <button>Submit</button>
    </form>
  </body>
</html>
```

This is how you take that data out.

```
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
listening to port 8080
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
listening to port 8080
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
listening to port 8080
```

Backend > index.js > app.get("/register") callback

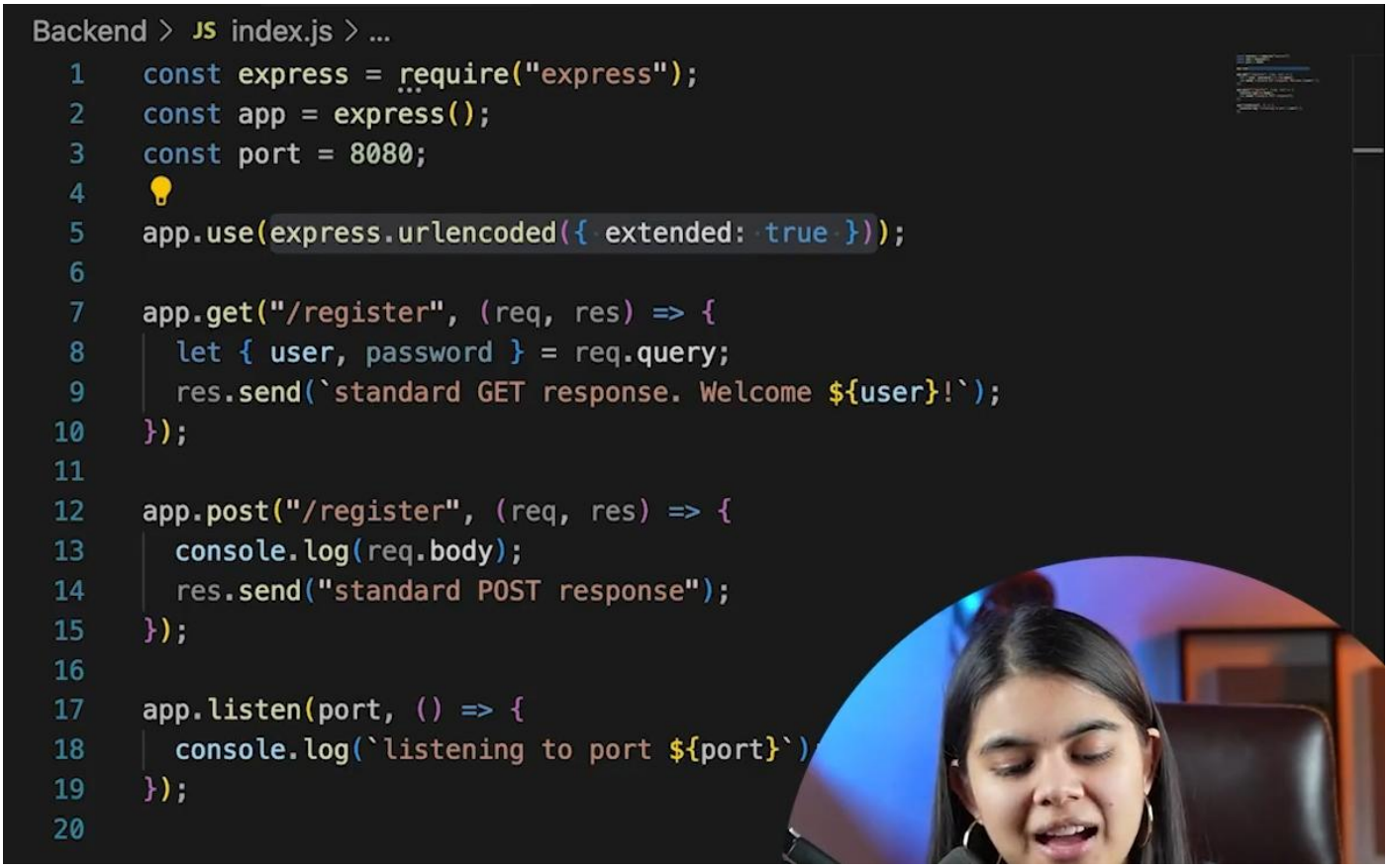
```js
const express = require("express");
const app = express();
const port = 8080;

app.get("/register", (req, res) => {
  let { user, password } = req.query;
  res.send(`standard GET response. Welcome ${user}!`);
});

app.post("/register", (req, res) => {
  res.send("standard POST response");
});

app.listen(port, () => {
  console.log(`listening to port ${port}`);
});
```

Handling Post requests

- Set up POST request route to get some response
- Parse POST request data

```
app.use(express.urlencoded({ extended: true }));
app.use(express.json());
```

If we are sending the data in json format then we have to write the second line.



The line 13 will print undefined if you don't write the 5th line.

The 5th line is a use function which caters all types of requests and it uses a middleware in it which is used to decode the data being received for express to read it and print it. So we have to include this line always.

We cant read req.body because we have to specify which type of data we are trying to parse and then only we can read it. That is why it was showing undefined.