## Express GET request

- **Fetch data in JSON format:**

```
app.get('/process_get', function (req, res) {
response = {
      first_name:req.query.first_name,
      last_name:req.query.last_name
   };
   console.log(response);
   res.end(JSON.stringify(response));
})
```

- **Fetch data in paragraph format**

```
app.get('/get_example2', function (req, res) {
res.send('<p>Username: ' + req.query['first_name']+'</p><p>Lastname
: '+req.query['last_name']+'</p>');
})
```

## Express POST request

- we will first install the *body-parser*(for parsing JSON and url-encoded data) and multer(for parsing multipart/form data) middleware.
- This body-parser module **parses the JSON, buffer, string and URL encoded data** submitted using HTTP POST request.
- **body-parser extract *the entire body portion of an incoming request stream and exposes it* on req.body.**

- To install the *body-parser* and *multer*, go to your terminal and use
- **npm install --save body-parser multer**

- After importing the body parser and multer, we will use
- the **body-parser** for parsing json and x-www-form-urlencoded header requests,
- **multer** for parsing multipart/form-data.

## body-parser

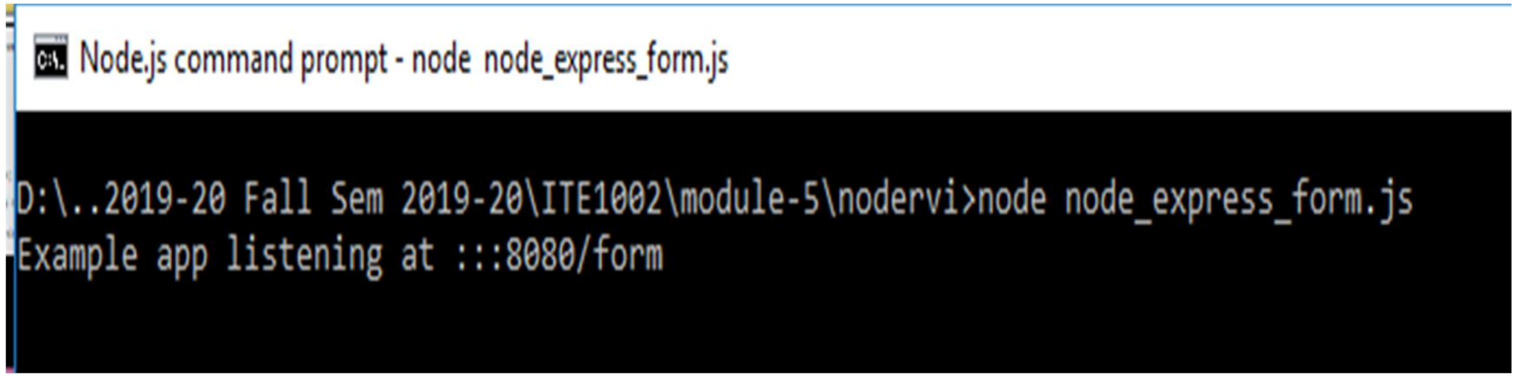
- The bodyParser object exposes various factories to create middlewares.(**npm install body-parser**)
- All middlewares will populate the **req.body** property with the parsed body when the Content-Type request header matches the type option, or an empty object ({}) if there was no body to parse, the Content-Type was not matched, or an error occurred.
- **bodyParser.json([options])**→Returns middleware that only parses json
- **bodyParser.text([options])**→Returns middleware that parses all bodies as a string.
- **bodyParser.urlencoded([options])**→Returns middleware that only parses urlencoded bodies
  - **extended** option →allows to choose between parsing the URL-encoded data with the querystring library (when false) or the qs library (when true).
  - The "extended" syntax allows for rich objects and arrays to be encoded into the URL-encoded format, allowing for a JSON-like experience with URL-encoded

## C:\Users\admin\node_express_form.js

```
var http = require("http");
var express = require('express');
var app = express();
var bodyParser = require('body-parser');
var urlencodedParser = bodyParser.urlencoded({ extended: true });
 // Running Server Details.
var server = app.listen(8080, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at %s:%s Port", host, port)});
```

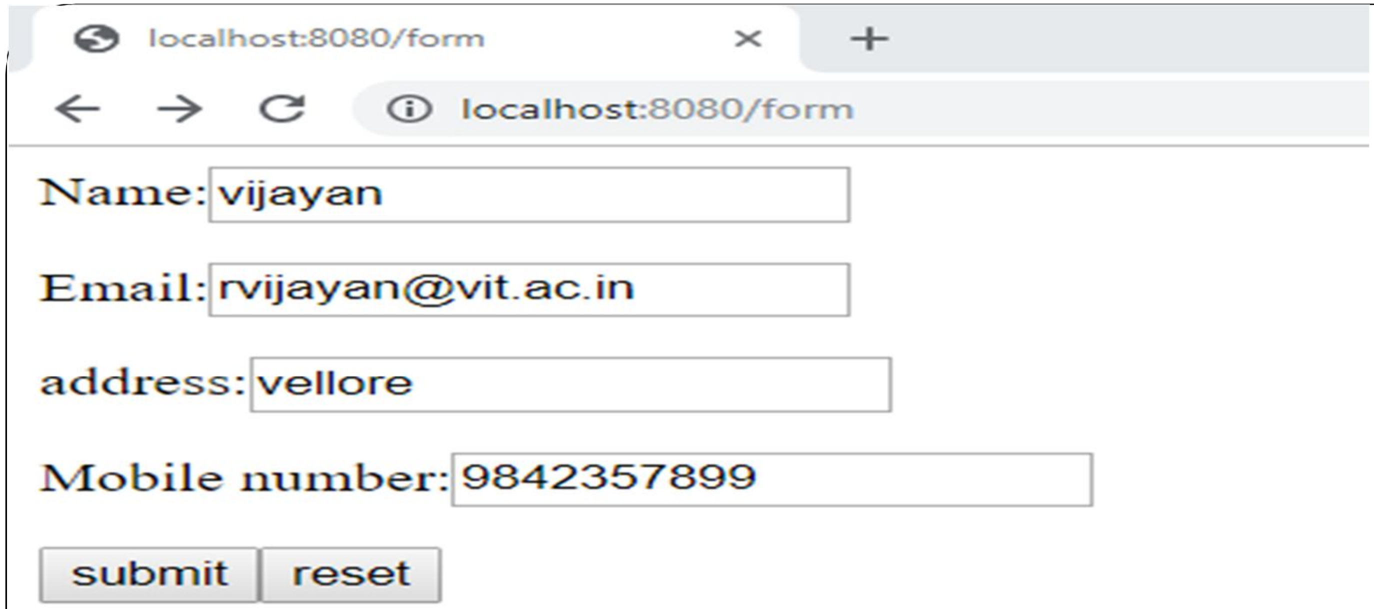
```
app.get('/form', function (req, res) {
 var html='';
 html +="<body>";
 html += "<form action='/thank'  method='post' name='form1'>";
 html += "Name:</p><input type= 'text' name='name'>";
 html += "Email:</p><input type='text' name='email'>";
 html += "address:</p><input type='text' name='address'>";
 html += "Mobile number:</p><input type='text'
name='mobilno'>";
 html += "<input type='submit' value='submit'>";
 html += "<INPUT type='reset'  value='reset'>";
 html += "</form>";
 html += "</body>";
 res.send(html);});
```

```
app.post('/thank', urlencodedParser, function (req, res){
 var reply='';
 reply += "Your name is" + req.body.name;
 reply += "Your E-mail id is" + req.body.email;
 reply += "Your address is" + req.body.address;
 reply += "Your mobile number is" + req.body.mobilno;
 res.send(reply); }); // response to the client browser window
```



Node.js command prompt - node node_express_form.js

```
D:\..2019-20 Fall Sem 2019-20\ITE1002\module-5\nodervi>node node_express_form.js
Example app listening at :::8080/form
```

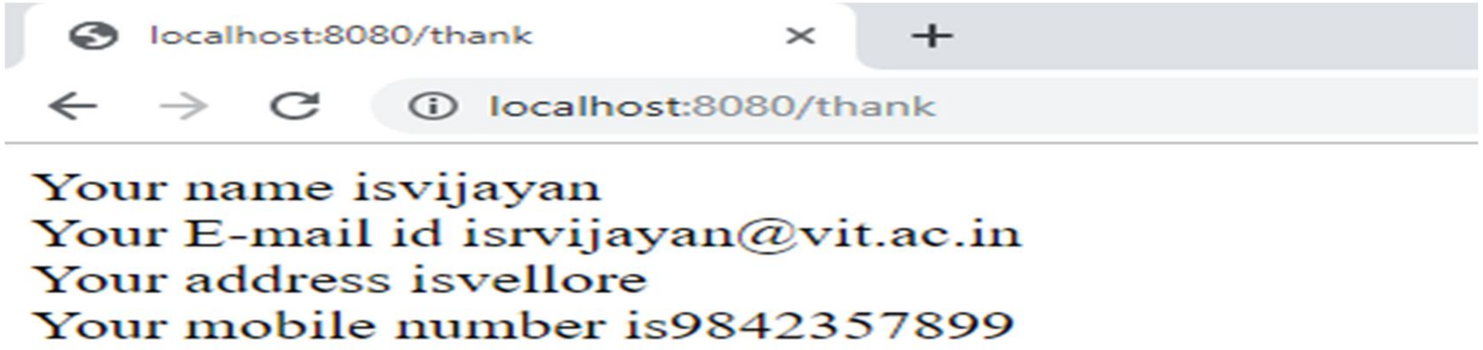


Name: vijayan

Email: rvijayan@vit.ac.in

address: vellore

Mobile number: 9842357899



```
Your name isvijayan
Your E-mail id isrvijayan@vit.ac.in
Your address isvellore
Your mobile number is9842357899
```

## Technologies\ITE1002\Programs\Node JS\home.html

- <html>
- <body>
- <h1>Hello</h1>
- <form action="/name" method="GET">
- <input type="text" name="t1">
- <button>SEND</button>
- </form>
- <form action="/add" method="GET">
- <input type="text" name="num1">
- <input type="text" name="num2">
- <button>ADD</button>
- </form>
- </body>
- </html>
-

## F:\Academic\Web Technologies\ITE1002\Programs\Node JS\index.js

```
let express = require('express')
let server = express()
let path=require('path')
server.get('/add', (req, res) => {
let n1=parseInt(req.query.num1); let n2=parseInt(req.query.num2);
let result=n1+n2; res.send('Addition:'+result);});
server.get('/home',(req,res)=>{
res.sendFile(path.join(__dirname,'home.html'))})
server.listen(3000);
console.log("Server is ready");
```

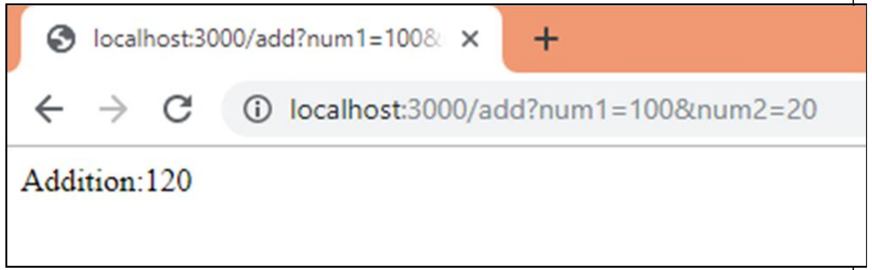
## In Visual Code

**F:\Academic\Web Technologies\ITE1002\Programs\Node JS\index.js**

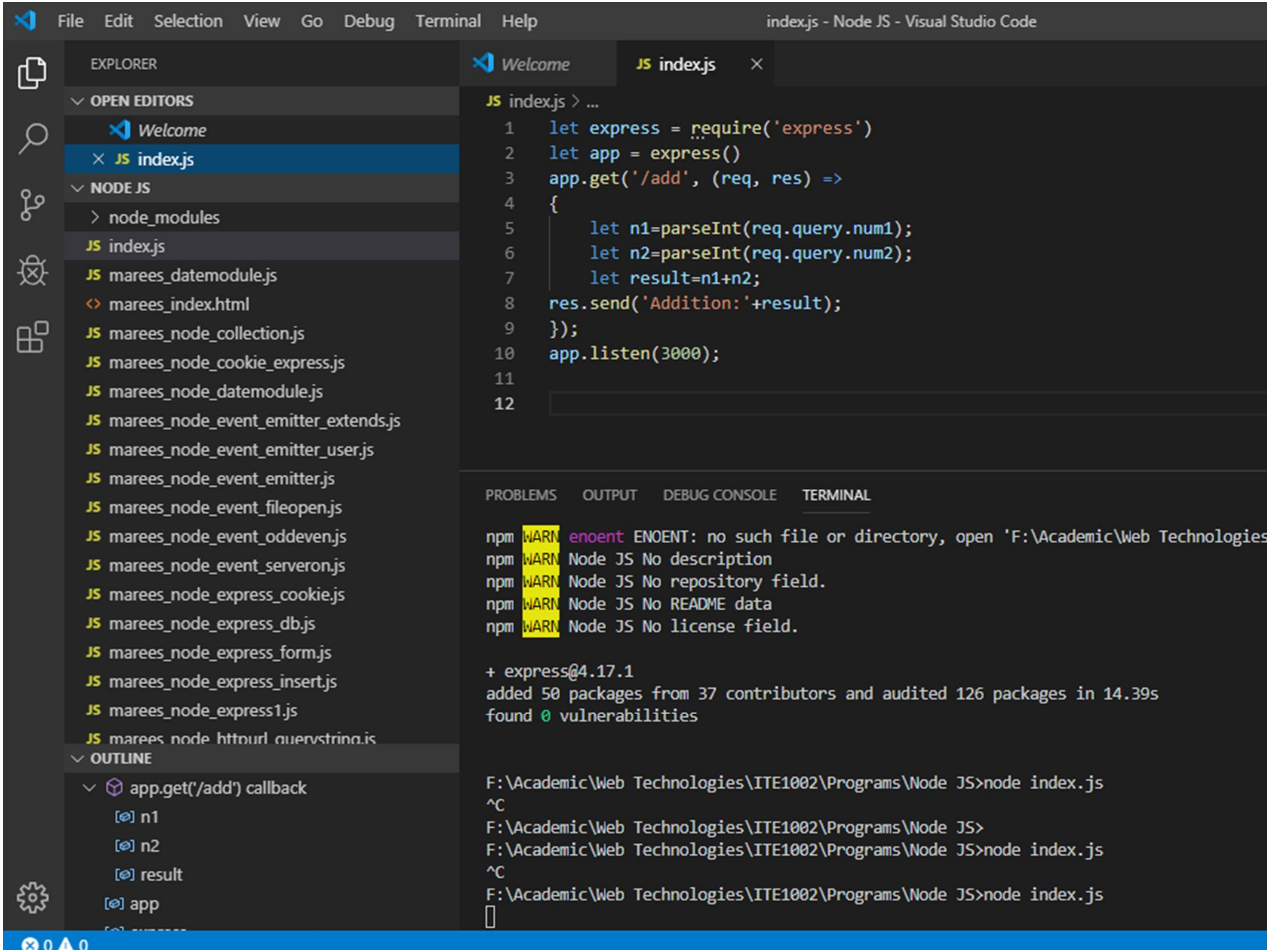
```
let express = require('express')
let app = express()
app.get('/add', (req, res) =>
{
let n1=parseInt(req.query.num1);
let n2=parseInt(req.query.num2);
let result=n1+n2;
res.send('Addition:'+result);
});
app.listen(3000);
```

Run **node index.js** in Terminal

[http://localhost:3000/add?num1=100&num2=20](http://localhost:3000/add?num1=100&num2=20) in chrome


localhost:3000/add?num1=100&
localhost:3000/add?num1=100&num2=20

```
Addition:120
```


File Edit Selection View Go Debug Terminal Help
index.js - Node JS - Visual Studio Code
EXPLORER
OPEN EDITORS
Welcome
index.js
NODE JS
node_modules
index.js
marees_datemodule.js
marees_index.html
marees_node_collection.js
marees_node_cookie_express.js
marees_node_datemodule.js
marees_node_event_emitter_extends.js
marees_node_event_emitter_user.js
marees_node_event_emitter.js
marees_node_event_fileopen.js
marees_node_event_oddeven.js
marees_node_event_serveron.js
marees_node_express_cookie.js
marees_node_express_db.js
marees_node_express_form.js
marees_node_express_insert.js
marees_node_express1.js
marees_node_httpurl_querystring.js
OUTLINE
app.get('/add') callback
n1
n2
result
app
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
let express = require('express')
let app = express()
app.get('/add', (req, res) =>
{
    let n1=parseInt(req.query.num1);
    let n2=parseInt(req.query.num2);
    let result=n1+n2;
res.send('Addition:'+result);
});
app.listen(3000);
```

```
npm WARN enoent ENOENT: no such file or directory, open 'F:\Academic\Web Technologies
npm WARN Node JS No description
npm WARN Node JS No repository field.
npm WARN Node JS No README data
npm WARN Node JS No license field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 126 packages in 14.39s
found 0 vulnerabilities


F:\Academic\Web Technologies\ITE1002\Programs\Node JS>node index.js
^C
F:\Academic\Web Technologies\ITE1002\Programs\Node JS>
F:\Academic\Web Technologies\ITE1002\Programs\Node JS>node index.js
^C
F:\Academic\Web Technologies\ITE1002\Programs\Node JS>node index.js
```

## Handling Cookie

- Cookies are simple, small files/data that are sent to client with a server request and stored on the client side. Every time the user loads the website back, this cookie is sent with the request. This helps us keep track of the user's actions.
- The following are the numerous uses of the HTTP Cookies –
  - Session management
  - Personalization(Recommendation systems)
  - User tracking
- To use cookies with Express, we need the cookie-parser middleware. To install it, use the following code –
- ***npm install --save cookie-parser***

- Now to use cookies with Express, we will require the **cookie-parser**. cookie-parser is a middleware which *parses cookies attached to the client request object*. To use it, we will require it in our **index.js** file; this can be used the same way as we use other middleware. Here, we will use the following code.

```
var cookieParser = require('cookie-parser');
app.use(cookieParser());
```

- cookie-parser parses Cookie header and populates **req.cookies** with an object keyed by the cookie names. To set a new cookie, let us define a new route in your Express app like –

```
var express = require('express');
var app = express();
app.get('/', function(req, res){
res.cookie('name', 'express').send('cookie set'); //Sets name =express
}); app.listen(3000);
```

- To check if your cookie is set or not, just go to your browser, fire up the console, and enter –

***console.log(document.cookie);***

- You will get the output like (you may have more cookies set maybe due to extensions in your browser) –

***"name = express"***

- The browser also sends back cookies every time it queries the server. To view cookies from your server, on the server console in a route, add the following code to that route.

***console.log('Cookies: ', req.cookies);***

- Next time you send a request to this route, you will receive the following output.

***Cookies: { name: 'express' }***

## Adding Cookies with Expiration Time

- You can add cookies that expire. To add a cookie that expires, just pass an object with property 'expire' set to the time when you want it to expire. For example,
- //Expires after 360000 ms from the time it is set.

*res.cookie(name, 'value', {expire: 360000 + Date.now()});*

- Another way to set expiration time is using **'maxAge'** property. Using this property, we can provide relative time instead of absolute time. Following is an example of this method.
- //This cookie also expires after 360000 ms from the time it is set.

***res.cookie(name, 'value', {maxAge: 360000});***

## Deleting Existing Cookies

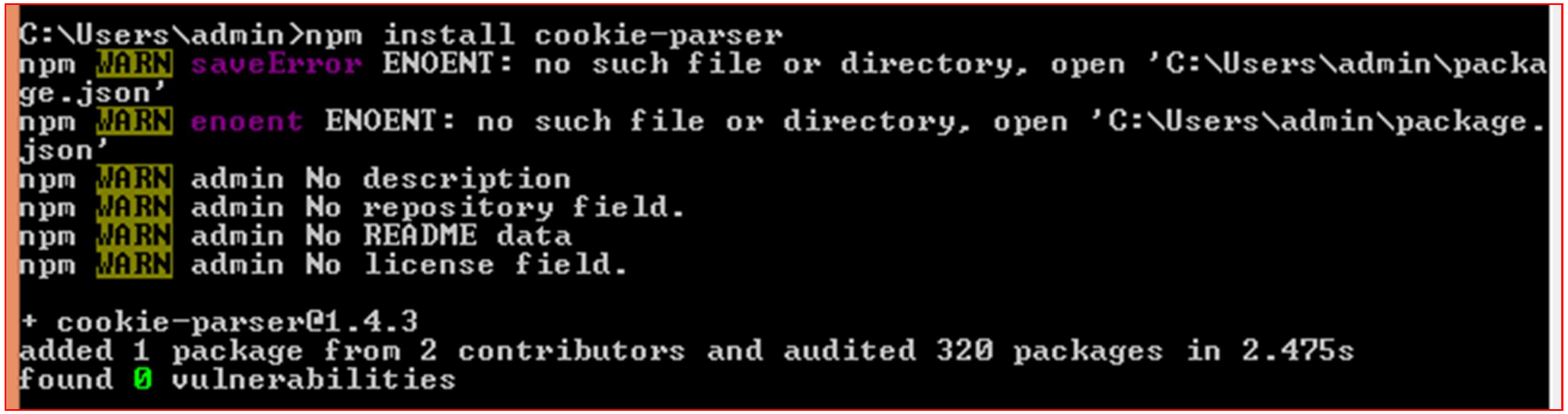
- To delete a cookie, use the clearCookie function. For example, if you need to clear a cookie named **foo**, use the following code.

```
var express = require('express');
var app = express();
app.get('/clear_cookie_foo', function(req, res){
res.clearCookie('foo');
res.send('cookie foo cleared'); });
app.listen(3000);
```

## C:\Users\admin\node_cookie_express.js

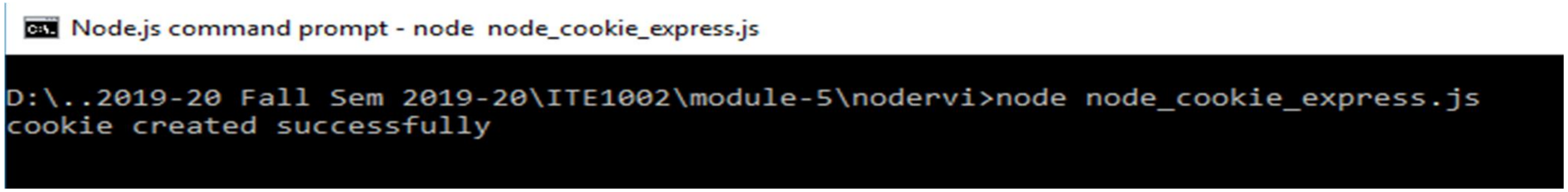
```
var express = require('express')
// npm install cookie-parser
var cookieParser = require('cookie-parser')
var app = express();
// need cookieParser middleware before we can do anything with
cookies
app.use(cookieParser());
// set a cookie
app.use(function (req, res, next) {
// check if client sent cookie
var cookie = req.cookies.cookieName; // RETRIEVE THE COOKIES
if (cookie === undefined) { // no: set a new cookie
var randomNumber=Math.random().toString();
randomNumber=randomNumber.substring(2,randomNumber.length);
res.cookie('cookieName',randomNumber, { maxAge: 900000, httpOnly:
true });
console.log('cookie created successfully');
} else
{ // yes, cookie was already present
console.log('cookie exists', cookie);
}next(); });                  // <-- important!
 app.get('/', function(req, res) {
res.send(JSON.stringify(req.cookies));
}) app.listen(8081)
```

## Install cookie-parser, running server and running client



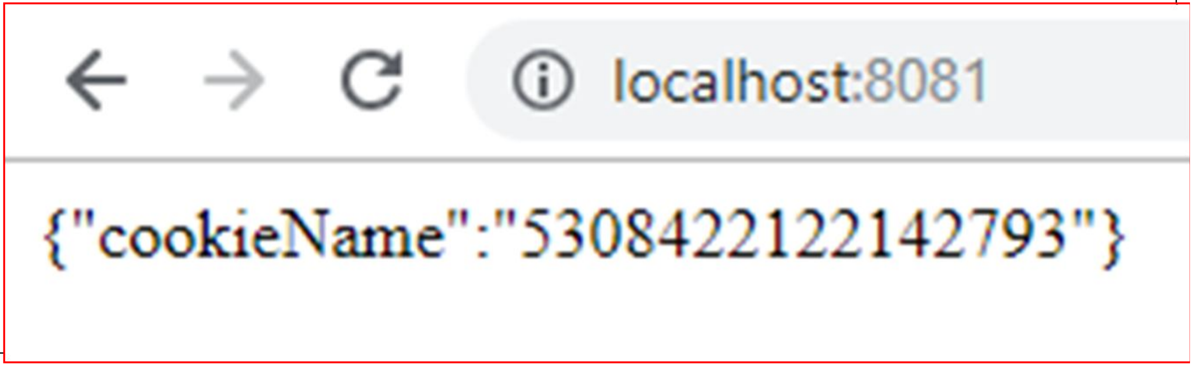
```
C:\Users\admin>npm install cookie-parser
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\admin\packa
ge.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\admin\package.
json'
npm WARN admin No description
npm WARN admin No repository field.
npm WARN admin No README data
npm WARN admin No license field.

+ cookie-parser@1.4.3
added 1 package from 2 contributors and audited 320 packages in 2.475s
found 0 vulnerabilities
```



Node.js command prompt - node node_cookie_express.js

```
D:\..2019-20 Fall Sem 2019-20\ITE1002\module-5\nodervi>node node_cookie_express.js
cookie created successfully
```



localhost:8081

{"cookieName":"5308422122142793"}

R Vijayan / Asso Prof / SITE / VIT

## Sessions

- Cookies are both readable and on the client side.
- Sessions solve exactly this problem.
- assign the client an ID and it makes all further requests using that ID. Information associated with the client is stored on the server linked to this ID.
- need the *Express-session*, so install it using the following code.

  **npm install --save express-session**
- put the **session** middleware handles all things for us, i.e.,
  - creating the session,
  - setting the session cookie and
  - creating the session object in **req** object.

```
var express = require('express');
var cookieParser = require('cookie-parser');
var session = require('express-session');
var app = express();
app.use(cookieParser());
app.use(session({secret: "Shh, its a secret!"}));
app.get('/', function(req, res){
   if(req.session.page_views){
      req.session.page_views++;
      res.send("You visited this page " + req.session.page_views + " times");
   } else {
      req.session.page_views = 1;
      res.send("Welcome to this page for the first time!");
   }
});app.listen(3000);
```

```
const express = require("express");
const session = require('express-session');
const app = express()
    // Port Number Setup
var PORT = process.env.port || 3000
   // Session Setup
app.use(session({
      // It holds the secret key for session
    //secret: 'Your_Secret_Key',
    secret:'343ji43j4n3jn4jk3n',
      // Forces the session to be saved
    // back to the session store
    resave: true,
      // Forces a session that is "uninitialized"
    // to be saved to the store
    saveUninitialized: true
}))
```

```
app.get("/", function(req, res){
        // req.session.key = value
    req.session.name = 'GeeksforGeeks'
    return res.send("Session Set")
})
   app.get("/session", function(req, res){
       var name = req.session.name
    return res.send(name)

    /* To destroy session you can use this function
     req.session.destroy(function(error){
        console.log("Session Destroyed")
    })*/

})
   app.listen(PORT, function(error){
    if(error) throw error
    console.log("Server created Successfully on PORT :", PORT)
})
```