Programming for Cloud

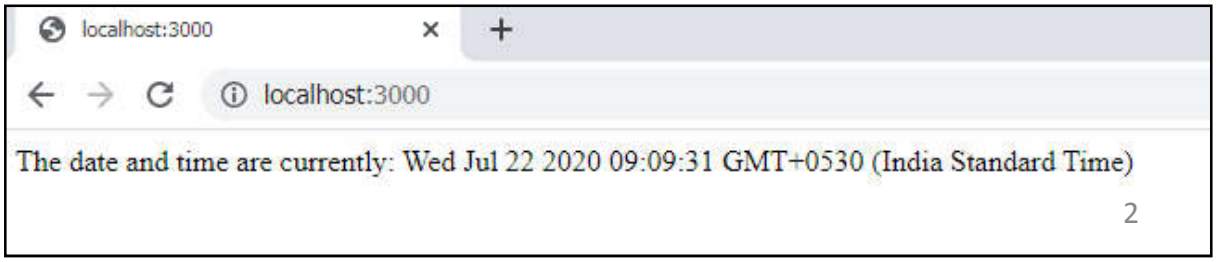
Dr. R. Karthi

Create Module in Node

```
// create modules in node
// use the file - myfirstmodule.js
Nodepgm1.js
var http = require('http');
var dt = require('./myfirstmodule');
var server = http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write("The date and time are currently: " + dt.myDateTime()
    );
  res.end();
});
server.listen(3000);
```

myfirstmodule.js

```
exports.myDateTime = function()
{
    return Date();
}
```



NPM

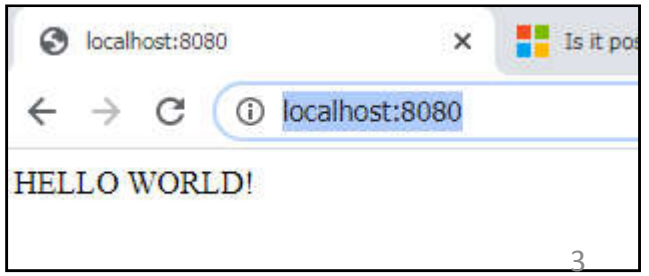
- npm is the standard package manager for Node.js.
- npm program is installed when you install Node.js
- npm install packages with all the required dependencies for the package

npm install upper-case – save

```
E:\CloudComputing\2020\Class Material\programs>npm install upper-case --save
+ upper-case@2.0.1
updated 1 package and audited 2 packages in 1.882s
found 0 vulnerabilities
```

```
// using NPM
var http = require('http');
var uc = require('upper-case');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write(uc.upperCase("Hello World!"));
  res.end();
}).listen(8080);
```

```
E:\CloudComputing\2020\Class Material\programs>node nodepgm1.js
```



Express

- **Web application framework** (**WAF**) is a software framework that is designed to support the development of web applications.
- Frameworks are libraries that help you develop your application faster and smarter.
- Common WAF are: Ruby on Rails, Django, ASP.NET, AngularJS, Express.
- Express is the most popular *Node* web framework

Installing Express and creating an application

- Step 1 : Create a directory to hold your application and make that your working directory.

  $ mkdir exp1

  $ cd exp1

- Step 2: Use the npm init command to create a package.json file for your application.

  $ npm init

  This command prompts you for a number of things, such as the name and version of your application. Hit RETURN to accept the defaults for most of them, with the following exception:

  entry point: **app.js**

  Enter app.js as the name of the main file and hit RETURN

- Step 3: Install Express in the exp1 directory and save it in the dependencies list.

  $ npm install express -- save

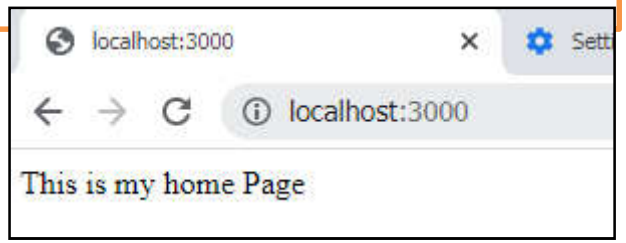
- Step4: Verify express installation

  $ npm list

Express programs

app.js

```
var express = require('express')
var app = express();                    // Creates an Express application
app.get('/',function(req,res)           // Route & method
{
    res.send(" This is my home Page");
}
);
app.listen(3000);
```




```
E:\CloudComputing\2020\Class Material\express_programs\exp1>node  app.js
```

Express programs

```
app.js
var express = require('express')
var app = express();                    // Creates an Express application
app.get('/',function(req,res)           // Route & method
{
    res.send(" This is my home Page");
}
);
app.listen(3000);
```

Routing **refers to determining how an application responds to a client request.**
Route has
- **endpoint, which is a URI (or path)**
- **HTTP request method (GET, POST, and so on)**

Route definition structure: app.METHOD(PATH, HANDLER)

Routing

- Route definition takes the following structure:
  app.METHOD(PATH, HANDLER) Where:
  - **app** is an instance of express.
  - **METHOD** is an HTTP request method, in lowercase.
  - **PATH** is a path on the server.
  - **HANDLER** is the function executed when the route is matched.

```
app.get ( '/' , function (req, res) {
      res.send('Hello World!')
}
)
app.post ( '/user' , function (req, res) {
      res.send('Got a POSt request at /user')
}
)
```

Multiple Route

```
var express = require('express');
var app = express();
app.get(` / ',function(req,res)
{
    res.send(" this is my home page");
});
 app.get(` /hh ',function(req,res)
{
  res.send(" this is my contact page");
});
app.get(` /sample ',function(req,res)
{
    console.log(req.url);
    res.sendFile(__dirname + '/demofile1.html');  // HTML file is sent to the browser
})
app.listen(3000);
```

Using Get Method

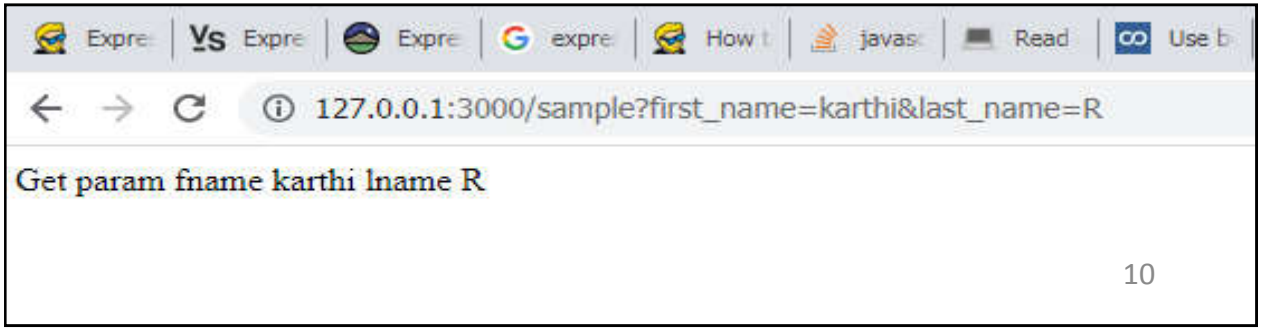
index_get_param_example.html

```
<html>
<body>
<form action="http://127.0.0.1:3000/sample"
method="GET">
First Name:<input type="text" name="first_name"
    >  <br>
LastName: <input type="text" name="last_name">

<input type="submit" value="Submit">
</form>
</body>
</html>
```

get_query_parse.js

```
var express = require('express')
var app = express();
app.get (`/sample` , function(req,res)
{
    let fname = req.query.first_name;
    let lname = req.query.last_name;
    res.send("Get param fname " + fname +
"  lname " + lname );
}

);
```

E:\exp1>node get_query_parse.js



Using POST Method

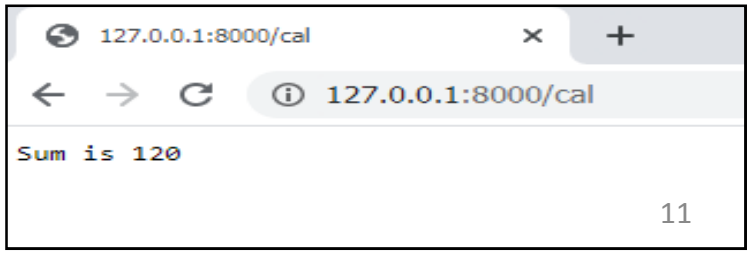
index_post_param_example.html

```
<html>
<body>
<form action="http://127.0.0.1:8000/cal
    " method="POST">
num 1: <input type="text" name="N1">  <
    br>
num 2: <input type="text" name="N2">  <
    br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

E:\exp1>node post_query_parse.js

post_query_parse.js

```
var express = require('express');
var app = express();
var bodyParser = require('body-parser');
var urlencodedParser = bodyParser.urlencoded(
{extended: false});
app.post('/cal',urlencodedParser, function(req, res)
{
        number1 = req.body.N1,
        number2 = req.body.N2
sum = parseInt(number1,10) + parseInt(number2,
10);
        res.end("Sum is " + sum)
}
);
app.listen(8000);
```



Full Application Development

- Client send a request

  HTML Forms page with post / get methods

- Server receives the request and process the data

  Node runs a java script program to process request and send the response

- Client receives the response

  HTML page display the response

Thank You