M.SATHISH KUMAR
E0119052

FINAL ASSESSMENT

CSE 220 - WEB PROGRAMMING & SCRIPTING

1. Write a Javascript program to compute the sum and product of an array of integers

```
<html>
<head>
<title> Sum and product of an array </title>
</head>
<body>
<script>
  var array = [10, 20, 10, 5, 2]
      sum = 0,
      product = 1,
  for (i=0; i < array.length; i+=1)
      {
        sum += array[i];
            product* = array[i];
      }
  console.log ("Sum of array :" + sum)
  console.log ("product of array :" + product)
</script> </body> </html>
```

Sample Output:

Sum of array : 47

Product of array : 20000

2. Javascript program to compare two objects to determine if first one contains equivalent property values to second one :

```
<! DOCTYPE HTML>
<html>
 <head>
 <title> Comparing two objects
 </head>
 <body>
 <script>
  ~~const matches = (obj, source) =>~~
  const user = (obj, source) =>
       Object.keys(source).every(key =>
            obj.hasOwnProperty(key) && obj[key] === source
                                                [key]));
console.log(user([{ colour: black, salary: 10,000,
                    native: India},
                  { colour: white, native: India}]));
```

Console.log (user {f colour : white, salary : 15,000, nature : US$},

{ salary : 15,000 , nature : US$}));

</script>

</body>

</html>

Output :

False

True

---

3. ★ In the code snippet, function is called before its declaration and function will be initialized during the compilation phase itself. After that num = 10 is assigned

★ According to JS engine, variable declared with var will be both declared and initialized during compilation phase.

★ Here first call the function and pass the variable num that is now having value undefined then the function will print value of num.

For this code output is "undefined"

4. Javascript programs using spread operator:

a) Concatenating elements of two array:

```
const fruits = [`Apple', `Orange', 'Kiwi']
const vegetalls = [`Tomato', `Potato', `Cabbage']

Const items = [...fruits, ...vegetables];
Console.log(items);
```

Output:

['Apple', `Orange', `Kiwi', `Tomato', `Potato', `Cabbage']

b) Copying elements from one array to another:

```
var fruits = ['Apple', 'Orange', 'Kiwi']
var vegetables: ['Tomato', 'Potato', 'Cabbage']
fruits.push(...vegetables);
Console.log(fruits)
```

Output:

['Apple', 'Orange', 'Kiwi', 'Tomato', 'Potato', 'Cabbage']

5. Write a program to create a collection in MongoDB database

```
var Mongo = require ('mongodb'). MongoClient;
var url = "mongodb://localhost:27017";
Mongo. connect (url, function (err, db)
{
  if (err) throw err;
    var db = db. db ("MyDatabase")
    db. createCollection ("Fruits", function (err, res)
    {
      if (err) throw err;
        console.log ("Collection created succesfully");
        db. close ();
    });
});
```

Prerequisites: MongoDB, Node

Output: ! Database named Fruits created in MongoDB through Node

6. Commands to run Mongo DB shell and Mongo DB server

* To run mongo DB Shell : mongo
* To run mongo DB server : mongod

* To run mongo DB shell : mongo
* To run mongo DB server : mongod

7. Creating a package.json file:

* It will be defaultly created on a node project
* It must be created in the root of the project
* npm init will create a package.json
* To update new package also in package.json npm, init will be useful
* npm init will update new installed packages in package.json

8. Installing and Importing Express TS Framework :

Installation :

* npm install express

A bunch of files will be created with basic connections, package.json , node_modules

* Importing :

Opening index.js in src, and type the below code snippet

```
import express from 'express';
const app = express()
app.listen(8080, () =>
        console.log("Importing express and listening
                    on port 8080")
);
```

* Checking :

In console, type npm start

* Output :

Importing express and listening on port 8080

9) Server program to define routes for Express JS application:

```
Const http = require ("http")
Const express = require ("express")
Const app = express ();
app.get ("/welcome/:message", (req, res, next) =>
    {
      if (req.params.message === "happy")
        {
          res.status (200).json ({
            message : "Routing successfully",
            word : req.params.message,
          })
          else
          {
            res.status (400).json ({message : "Bad request"});
          }
});
```

```
app.post("/post", (req, res) =>
    { res.status(200).json({
        message: "You are Posting",
    });
});

app.patch("/patch", (req, res) =>
    {
      res.status(200).json({
          message: "You are patching",
      });
    });

app.delete("/delete", (req, res) =>
    {
      res.status(200).json({
          message: "You are deleting",
      });
    });
```

```
app.use((req, res) =>          // Handling errors
    {
    res.status(404).json({
        message: "Handling Errors",
    });
});

const server = http.createServer(app);
const port = process.env.PORT || 3000;
server.listen(port);
```

10.

```
const express = require('express')
const app = express();
const port = 3000;
const socket = require('socket.io');
var io = socket.listen(app.listen(port));
io.on("connection", function(socket) =>
    {
    console.log("Connected to socket")
```

10.

```
var app = require ('express');
var http = require ('http'). Server (app);
var io = require ('socket.io') (http);

app. get ('/', function (req, res)
    {
    res. sendfile ('index. html');
    });

io. on ('connection', function (socket)
    {
    console. log ('User is connected');

    socket. on ('disconnect', function ()
        {
        console. log ('User is disconnected');
        });
    });

http. listen (3000, function (){
    console. log (3000, function ()
        {
        console. log ('listening at port 3000')
    });
```

71 ❀ Chat application :

★ ~~Server.js~~ : index.js :

```
let app = require ('express')();
let http = require ('http').Server(app);
let io = require ('socket.io')(http);
app.get ('/', (req, res) => {
    res.sendfile (__dirname + '/index.html');
});
io.on ('connection', (socket) =>
    {
    io.emit ('noOfConnections', Object.keys
                (io.sockets.connected).length)

socket.on ('disconnect', () => {
        console.log ('disconnected')
        io.emit ('noOfConnections', Object.keys
                (io.sockets.connected).length)
    })
socket.on ('chat-message', (msg) => {
    socket.broadcast.emit ('chat-message', msg)
});
```

```
socket.on('joined', (name) => {
        socket.broadcast.emit('joined', name)
    })
socket.on('left', (name) =>
    {
    socket.broadcast.emit('left', name)
    }
socket.on('typing', (data) =>
    {
    socket.broadcast.emit('typing', data)
    })
    })
http.listen(process.env.PORT || 3000, () =>
    {
    console.log('Server is started at port 3000')
    })
```

A index.html

```
<html>
<head>
 <title> Welcome to Vue APP </title>
</head>
<body>
 <div class = "container" id = "app">
   <div class :" col-md-4 >
       <h1>{{ name }} </h1>
    </p>
   </div>
<form @ submit.prevent = "Set Name" >
      <label > Set ur Name : </label>
      <input type = "text"  v-model = "name" >
      <input type = "submit" value = "Add" >
   </div>
</form>
```

```
<div class="card bg-info" v-if="joined">
    My chat APP
    <span class>{{connectionCount}} <\span>
    members : {{ online }}
  <ul class = list-group >
     <small v-if="typing" >
      <i > {{ typing }} is typing <\i>
     <\small>
    <li class = list-group-item" v-for= message in messages>
        <span : class "{ float left : {message} }">
        < small > {{ message.by }} <\small>
        {{ message. message }}
      <\span>
    <\b>
  <\ul>
  < input type = "text" placeholder = "type new"
                    v-model = "New message" >
<\form>
```

```
<script src="/socket.io/socket.io.js"></script>
<script>
    var socket= io()
    let appp = new Vue({
     el : '#app';
     data : {
          new message : null,
          messages : [ ]
          typing : false,
          Online : [ ]
          name : null,
          ready : false,
           info : [ ]
           Connection count : 0,
       },
methods {
    send ( ) {
        socket.emit ('chat-message',{message, this.message,
                                     user:this.name })
        this . newmessage = null
    },
```

```
setName (){
    socket.emit ('Joined', this.Name)
    this.ready = true
  }
},

watch: {
  newmessage (value)
    {
    value ? socket.emit ('typing', this.name)
},
created ()
  {
  saket.on ('chat message', (data) => {
    this.message.push ({message: data.message})

  socket.on ['typing', (data =>
    {console.log (data)
    this.typing = data
  })

});

</script>
</body> </html>
```