SRN | | | | | | | | | | | | |

PES University, Bengaluru

(Established under Karnataka Act No. 16 of 2013)

UE19CS204

DECEMBER 2020: END SEMESTER ASSESSMENT (ESA) B TECH 3rd SEMESTER

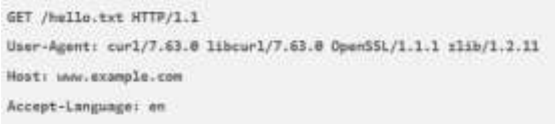
UE19CS204 – Web Technologies

Time: 3 Hrs	Answer All Questions	Max Marks: 100

1 a) Describe the structure of HTTP Request and Response message using an example. 4

Answer:

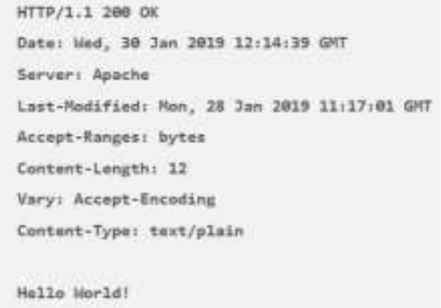
An HTTP request is just a series of lines of text that follow the HTTP protocol. A GET request might look like this:



```
GET /hello.txt HTTP/1.1
User-Agent: curl/7.63.0 libcurl/7.63.0 OpenSSL/1.1.1 zlib/1.2.11
Host: www.example.com
Accept-Language: en
```

This section of text, generated by the user's browser, gets sent across the Internet.

When an origin server receives an HTTP request, it sends an HTTP response:



```
HTTP/1.1 200 OK
Date: Wed, 30 Jan 2019 12:14:39 GMT
Server: Apache
Last-Modified: Mon, 28 Jan 2019 11:17:01 GMT
Accept-Ranges: bytes
Content-Length: 12
Vary: Accept-Encoding
Content-Type: text/plain

Hello World!
```

(2 marks each)

b) A web page has a form for online registration of a Tennis competition. 5

- The participants can be adults or children.
- Participant must be able to choose if (s)he has previously participated in the same event in one or more of the years 2019-20 /2018-19/2017-18
- The form should provide the controls to accept above mentioned details as well as the personal information like name and gender.
- The user must be redirected to the server script "/registration" when he submits the form.
- The details he enters must be visible in the address bar of his/her browser

Write the HTML code for the form with suitable form elements and default values.

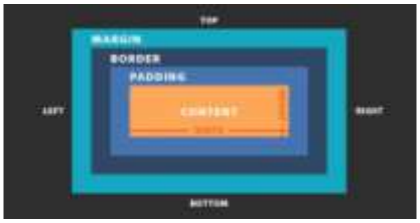
Answer:

Following should be present:

- <form action="/registration" method="get"> - **2 marks**
- Radio buttons for adult or child – **1 mark**
- Check box for previous participation – **1 mark**
- Text box for name and radio/text/select for gender – **1 mark**

c) With a neat diagram, explain the CSS Box model and its significance. 5

Answer:



CSS box model determines the size, position, and properties (color, background, border size, etc.) of each element. Every box is composed of four parts (or areas), defined by their respective edges: the content edge, padding edge, border edge, and margin edge. Every box has a content area and an optional surrounding margin, padding, and border.

(2 + 3 marks)

d) Write a JavaScript function to take a string as input parameter and reverse the case of every alphabet in the string. The function should return the modified string. Ex. changeCase (“HeLlo”) must return “hElLO”. 6

Answer:

```
function changeCase(str)
{
var UPPER = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
var LOWER = 'abcdefghijklmnopqrstuvwxyz';
var result = [];

  for(var x=0; x<str.length; x++)  {  // 1 mark
    if(UPPER.indexOf(str[x]) !== -1)  {  // Any approach to identify Uppercase – 1 mark
      result.push(str[x].toLowerCase());  // convert to Lowercase – 1 mark
    }
    else if(LOWER.indexOf(str[x]) !== -1)  { // Any approach to identify Lowercase – 1 mark
      result.push(str[x].toUpperCase());  // convert to Uppercase – 1 mark
    }
    else {
      result.push(str[x]);  //  add to result array/string – 1 mark
    }
  }
return (result.join(''));
}
```

2 a) A HTML page contains the following:

- A table exists with id “table1”
- A button “Add” with id “btn1”
- A div with id “display”

Add JavaScript to the webpage for the following specifications

• On clicking “Add” a new row with 2 cells are added to the table

• Populate each cell with a random number (between 1 and 200)

• If the cell contains an even number, mouse over should turn the cell green

• If the cell contains an odd number, mouse over should turn the cell red.

• If you click on a cell, the div will be populated with the cell content

Note: Write only vanilla JavaScript code. Assume that body tag has an onload=”init()” handler.

Answer:

```
function init(){
        document.getElementById(“btn1”).onclick = show; // 1 mark
        tbl1 = document.getElementById(“table1”);
        dsp = document.getElementById(“display”);
}
function show(){
        num1 =  Math.floor(Math.random()*200)+1;
        num2 =  Math.floor(Math.random()*200)+1; // 1 mark
        tr = document.createElement(“tr”);
        td1 = document.createElement(“td”);
        td2 = document.createElement(“td”);
        td1.innerHTML = num1; td2.innerHTML = num2; // 1 mark
        tr.appendChild(td1); tr.appendChild(td2); tbl1.appendChild(tr); // 1 mark
        td1.onmouseover=change; td2.onmouseover=change; // 1 mark
        td1.onclick = display; td2.onclick = display; // 1 mark
}
function change(event){
        if(event.target.innerHTML%2) // 1 mark
                event.target.style.backgroundColor = “red”;
        else
                event.target.style.backgroundColor = “green”;
}
function display(event){
        dsp.innerHTML = event.target.innerHTML; // 1 mark
}
```

8

b) Write briefly about the two methods of geolocation object.

Answer:

navigator.geolocation object allows you to access geo location through two primary functions:

- getCurrentPosition() : Returns the location of the visitor as a one-time snapshot
- watchPosition() : Returns the location of the visitor every time the location changes
- Both functions take the following parameters: Success callback function, Error callback function (optional) and Geo location options object (optional)

(2 marks each)

4

c) A web page has the following:

- A select element with options as Song names and value set to URL of an audio file

```
(ex. <select id=”input1”>
                <option value=”faded.mp3”>Faded by Alan Walker </option>
                …
     </select>)
```

- A button to fetch the audio asynchronously from the server
- A placeholder “div” where the audio fetched is displayed

Write JavaScript function as the click event handler for the button. The function should use the “fetch” API to asynchronously fetch the “selected” audio file from the server URL (specified in the value property) and display it in the div element.

Note:

- There should be only one audio element shown at any time
- Write only JavaScript code. HTML code is not required

Answer:

```
function fetchdata(){
        mydiv = document.querySelector("#container");
        fetch(document.querySelector(“#input1”).value) // 2 marks – fetch with 2  then
        .then(function(response){
                return response.blob(); // 1 mark
        })
        .then(function(data){
                aud = document.createElement("audio");
                aud.controls=true;
                aud.src=URL.createObjectURL(data); // 1 mark
                mydiv.firstChild.remove()
                mydiv.appendChild(aud); // 1 mark
        })
        .catch(function(error){
                console.log(error.message);
        })}
```

5

d) Name the methods/events supported by web workers for communication between worker and main threads. Show how they are used with an example.

Answer:

- Listen for messages, using the onmessage event listener function.
- Send messages via the postMessage() function.

```
worker.postmessage(“abc”); //send message to worker thread
worker.onmessage = function(event){ //receive message from worker
      console.log(“received message: “+event.data)
}
```

(1 + 1 + 1 marks)

3

3 a) What is the significance of key property? Describe with an example.

Answer:

- When returning an array or list of elements, the individual element should be uniquely identified by a key property
- The key property helps React identify each element in the list for rendering or updating etc.
- Not specifying the key property will display a warning on the console:

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) => <li>{number}</li>);
ReactDOM.render(
        <ul>{listItems}</ul>,
        document.getElementById('root')
);
```

(3 + 2 marks)

5

b) Write a component “Poster” that includes the movies poster image, a paragraph showing the movie title and another paragraph showing the movie director’s name. An App component renders multiple Poster components based on the values of image src, title and director name stored in an array of objects. The object members are src, title and director. The values in the array should be passed as properties to the Poster component. Note: Use “map” method or even a simple for loop to traversing through the array of values.

Answer:

```
Class Poster extends React.Component {
        render(){
                return (
                <div>
                        <img src={props.src}/> // 2 marks
                        <p>{props.title}</p>
                        <p>{props.director}</p>
                </div>
        )}};
Class App extends React.Component {
        render(){
                var mobj = [{…}]
                // 1 mark (map or any other approach to access all array elements
                acceptable)
                {mobj.map((m)=> {return (<Poster src={m.src} title={m.title}
                director={m.director} />)} // 1 mark
        )}}}
```

4

c) Explain the following methods of component life cycle:

- componentDidMount
- componentDidUpdate
- componentWillUnmount

Answer:

- componentDidMount
  - method that is executed after the React Component has been rendered (after the call to render method)
- componentDidUpdate
  - method that is executed after the React Component has been rendered (after an update to the component using setState or new props added)
- componentWillUnmount
  - method that is executed before the React Component is unmounted

(1 mark each)

3

d) In the game of Minecraft, a player can discover and extract raw materials to craft tools and items. For example, a player can build an axe using sticks and stones. Write a uncontrolled form (using only refs, without states), that has

- Two input elements to read the number of sticks and stones a player has (initial values set to 1)
- A compute button that displays the number of axes that can be made out of these resources.
- A placeholder div to display the number of axes

Note: For computation use the following logic. Three stones and two sticks can be used to build an axe. Assuming a player has 10 stones and 4 sticks, we have 3 sets of three sticks (10 / 3) and 2 sets of two stones (4 / 2), hence only 2 (minimum of these numbers) axes can be built.

8

Answer:

```
class Minecraft extends React.Component{
        constructor(props){
                super(props);
                this.setRef=(el)=>{ // 2 marks (two separate set ref methods accepted)
                        if(el.name=="height")
                                this.sticknum=el;
                        else if(el.name=="weight")
                                this.stonenum=el;
                }
                this.setOutputRef=(el)=>{this.output=el} // 1 mark
                this.handleSubmit=this.handleSubmit.bind(this);
        }
        handleSubmit=function(event){
                var axes;
                var stones = parseInt(this.sticknum.value);
                var sticks=this.stonenum.value-0;
                //axes = calculate number of axes // 1 marks
                var stonesset = stones / 3;
                var sticksset = sticks / 2;
                axes = Math.min(stonesset, sticksset);

                this.output.innerHTML="Number of Axes: "+axes; // 1 mark
                event.preventDefault();
        }
        render(){
          return(<div>
                <form onSubmit={this.handleSubmit}> // 1 mark
                  <label> Stick:</label>
                  <input name="stick" defaultValue='1' type="text" ref={this.setRef}/>
                  <label> Stone:  </label>
                  <input name="stone" defaultValue=’1' type="text" ref={this.setRef}/>
                  <input type="submit" value="submit"/>
                </form>
                <h2 ref={this.setOutputRef}/>
              </div> )}}
ReactDOM.render(
        <Minecraft/>,
        document.querySelector("#container")
)
```

// 2 input fields with defaultValue and ref – 2 marks (defaultValue missing – reduce 1 mark)

4 a) Write in brief the important features of Node JS. **4**

Answer:

Following are some of the important features of Node.js:

- **Asynchronous and Event Driven** − All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data.
- **Very Fast** − Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- **Single Threaded but Highly Scalable** − Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way
- **No Buffering** − Node.js applications never buffer any data. These applications simply output the data in chunks.
- **License** − Node.js is released under the MIT license

(Any 4 – 1 mark)

b) Write JavaScript code using NodeJS modules only (not Express JS) to accept a GET request in the form of http://server:8000/calc?op=add&op1=10&op2=20 and returns back the calculated output, in this case the addition of 10 and 20, which is 30. The operations supported by this basic calculator are addition, subtraction, multiplication and division. **8**

Answer:

```
var url = require('url');
var http = require('http');
var qs = require('querystring') // 1 mark

http.createServer(function(request,response){
   if(request.method=='GET'){  // 1 mark
           var myurl = url.parse(request.url) // 1 mark
           var query = myurl.query; // 1 mark
           var qobj = qs.parse(query); // 1 mark
           var res = 0;
           if(qobj.op==”add”) // 1 mark
                     res = qobj.op1 + qobj.op2;
           else if(qobj.op==”sub”)
                     res = qobj.op1 - qobj.op2;
           …
           response.writeHead(200,{'Content-type':'text/html'});  // 1 mark
           response.write(res); // 1 mark
    }
 })).listen(8000);
```

c) What are data streams in Node JS? Name the different types of streams with example of each. **4**

Answer:

A stream is an abstract interface for working with streaming data in Node.js. The stream module provides an API for implementing the stream interface.

- Writable: streams to which data can be written (for example, response on server).
- Readable: streams from which data can be read (for example, request on server).
- Duplex: streams that are both Readable and Writable (for example, net.Socket).
- Transform: Duplex streams that can modify or transform the data as it is written and read (for example, zlib.createDeflate()).

(2 + 2 marks)

d) Write Node JS code using the “readline” module to read every line in “input.txt”. Call the changeCase(str) function with the line read as the parameter and write the value returned on to the console. Note: You do not have to write the changeCase function.

Answer:

```
readline = require('readline'); // 1 mark
rl = readline.createInterface({ // 1 mark
        input:fs.createReadStream("test.txt"),
        output:process.stdout,
        terminal:false
})
rl.on('line',function(line){  // 1 mark
        var rev = changeCase(line);  // 1 mark
        console.log(rev);
})
```

4

5 a) What are Express middleware functions? Write briefly about the order of execution of middleware functions.

Answer:

Middleware functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application’s request-response cycle. These functions are used to modify req and res objects for tasks like parsing request bodies, adding response headers, etc.

The order in which Middleware in Express are executed is the order in which they are written/included in your file, given that the route matches also needs to be considered. The important thing is to call the next middleware in the order of execution using the “next();”

(2+1 marks – reduce 1 mark if next() is not mentioned)

3

b) Write server-side script in JavaScript to route requests for GET and POST requests for flight details. The details are stored in the mongodb database in the following format.

{from:”BLR”, to:”DEL”, dept:”12:25”, arrv:”14:25”, flnum:”6E-2428”}

The server script should support the following routes:

- GET /flights – return details of all flights
- GET /flights/:from/:to – return details of flight between specific airports
- POST /flights – save details of a flight and return a success or error message

Answer:

```
- app.get("/flights", function(req,res){
          …
          db.collection('student').find().toArray(function(err,objs){
                  res.send(objs)
          });});
- app.get("/flights/:from/:to", function(req,res){
          …
          db.collection('student').find({from:req.params.from,
          to:req.params.to}).toArray(function(err,objs){
                  res.send(objs)
          });});
- app.post("/flights", function(req,res){
          …
          db.collection('student').insertOne(req.body,function(err,objs){
                  res.send("Save Successful!!!")
          });});
```

(2 + 3 + 3 marks)

8

c) Ronaldino, a back end developer, has to create at least 4 -5 Express routes each for the resources “players”, “match” and “teams”. Since he is new to Express, he wants suggestions on how to structure his JavaScript code. Write an approach that can be taken by him to solve this problem.

Answer:

He has to use Express Routers to organize the routes in different JS files. And write the routes specific to each resource in these files.

```
var playersrouter = require("./players.js")
app.use("/players", playersrouter);
var matchrouter = require("./match.js")
app.use("/match", matchrouter);
var teamsrouter = require("./teams.js")
app.use("/teams", teamsrouter);
```

(1 + 2 marks)

3

d) Write a route called “picsupload” which receives multiple image files in a POST request. The route function should save each of them under the “pics” folder in the same directory as the JS file. Any error like, no files uploaded or saving files failed should be handled accordingly.

Note: Write only the server JS code. HTML/client code is not required.

Answer:

```
const express = require('express');
const fileUpload = require('express-fileupload'); // 1 mark
const app = express();

// default options
app.use(fileUpload()); // 1 mark

app.post('/picsupload', function(req, res) {
  if (!req.files || Object.keys(req.files).length === 0) { // 1 mark
    return res.status(400).send('No files were uploaded.');
  }
for(var i in req.files){ // 1 mark
  let picFile = req.files[i];
  // Use the mv() method to place the file somewhere on your server
  picFile.mv('pics/'+picFile.name, function(err) { // 1 mark
    if (err)
      return res.status(500).send(err); //1 mark
  });
}
 res.send('Files uploaded!');
});
```

6