T1_Unit 1,2,3 For only reference-variations possible

Create HTTP webpage on which Home page display “Welcome to Log in page” in blue color and font size must be 32px, Login page shows one HTML file from static URL having Form with detail for Username, Password, submit and reset button, Gallery page reflect one Image “hello.jpg” and any other page shows “Page Not found”. Write all necessary files to perform task. (Image already exist in same folder)

```
var h=require("http");
var ps=require("fs");
var u=require("url");
var addr="http://localhost:8080/form.html";
var q=u.parse(addr,true);
data=ps.readFileSync("."+q.pathname);
data2=ps.readFileSync("./02.jpg")

var server=h.createServer(
   function(req,res)
   {
if(req.url=="/")
{
   res.writeHead(200,{"content-type":"text/html"});
   res.write("<h1 style='color:blue'> Welcome to Log in page </h1>");
   res.end();
}
else if(req.url=="/student")
{
            res.writeHead(200,{"content-type":"text/html"}); //plain shows code as it is
            res.write(data);
            res.end();
}
else if(req.url=="/gallery")
{
            res.writeHead(200,{"content-type":"image/jpg"}); //plain shows code as it is
            res.write(data2);

            res.end();
}
else
{
   res.writeHead(404,{"content-type":"text/html"});
   res.write("<h1> Page Not found </h1>");
   res.end("Thanks");
  res.write("Bye");

}
   }
);
server.listen(5001);
```

```
console.log("Thanks for run");
```

Write a nodeJS script to fire an event named calculate which calculates the total marks of 5 subjects about of 25 marks and displays the total marks on console as an output. The calculate event fires another event name percentage which takes total marks as argument and percentage should get displayed in console.

```
const EventEmitter = require('events');
// Create an instance of EventEmitter
const eventEmitter = new EventEmitter();
// Listener for the 'calculate' event
eventEmitter.on('calculate', (marks) => {
    let totalMarks = 0;
    for (let i = 0; i < marks.length; i++) {
        totalMarks += marks[i];
    }
    console.log("Total Marks:", totalMarks);
    // Emit the 'percentage' event with total marks as argument
    eventEmitter.emit('percentage', totalMarks);
});
// Listener for the 'percentage' event
eventEmitter.on('percentage', (totalMarks) => {
    const percentage = (totalMarks / 125) * 100; // Assuming 25 marks for each of the 5
subjects
    console.log("Percentage:", percentage.toFixed(2) + "%");
});
// Example marks for 5 subjects (out of 25 each)
const marks = [25, 25, 25, 25, 25];

// Emit the 'calculate' event with marks as argument
eventEmitter.emit('calculate', marks);
```

Write a script of JSON array containing objects and display the same in the console:

```
// Create a JSON array containing
objects var jsonArray = [
  {
    "name":
    "John",
    "age": 25,
    "city": "New York"
  },
  {
    "name":
    "Alice",
    "age": 30,
    "city": "San Francisco"
```

```
  },
  {
    "name":
    "Bob",
    "age": 35,
    "city": "London"
  }
]
;

// Display the JSON array in the
console console.log(jsonArray);
```

Write a JSON script by entering user details of three different people having the same age group in the string format method. Print the following result in object form. (1) User Details (2) Name of 2nd person and his/her age

```
var json = `{
    "userDetails
    ": [
      {
        "name":
        "John", "age":
        25, "gender":
        "Male",
        "occupation": "Engineer"
      },
      {
        "name":
        "Emily",
        "age": 25,
        "gender": "Female",
        "occupation":
        "Doctor"
      },
      {
        "name": "Michael",
        "age": 25, "gender":
        "Male", "occupation":
        "Teacher"
      }
    ]
  }`

  var data = JSON.parse(json);
  var userDetails = data.userDetails;

  console.log("User Details:");
```

```
console.log(userDetails);

var secondPerson = userDetails[1];
var name = secondPerson.name;
var age = secondPerson.age;

console.log("Name of 2nd person:"+name);
console.log("Age of 2nd person:"+age);
```

Write a JSON script to store information related to books based on their id, topic,edition and author. (Minimum details of three books having id=1,2,3)

```
{
  "books": [
    {
      "id": 1,
      "topic": "Fiction",
      "edition": "1st",
      "author": "John Doe"
    },
    {
      "id": 2,
      "topic": "Science",
      "edition": "3rd",
      "author": "Jane Smith"
    },
    {
      "id": 3,
      "topic": "History",
      "edition": "2nd",
      "author": "Robert Johnson"
    }
  ]
}
```

Write a function 'FirstAndLast' that takes in an array, and returns an object with: 1) the first element of the array as the object's key, and 2) the last element of the array as that key's value. (Example input: ['ABC', 'DEF', 'Employee', 'Manager'] output: ABC : 'Manager')

```
function FirstAndLast(array) {
  if (Array.isArray(array) && array.length > 0) {
    var result = {};
    result[array[0]] = array[array.length - 1];
    return result;
  } else {
    return null;
  }
}
var inputArray = ['ABC', 'DEF', 'Employee', 'Manager'];
var outputObject = FirstAndLast(inputArray);
console.log(outputObject);
```

Write a JSON script to define Name,DOB,Age and birthplace of one person. Then print his birthdate in console as well as in chrome after clicking birthdate button

```
<!DOCTYPE html>
<html>
<head>
  <title>Person's Birthdate</title>
  <script>
    function printBirthdate() {
      var jsonData = {
      "person": {
          "Name": "John Doe",
          "DOB": "1990-05-18",
          "Age": 33,
          "birthplace": "New York City"
        }
      };

      var dob = new Date(jsonData.person.DOB);
      console.log("Birthdate:", dob.toDateString());

      var birthdateElement = document.getElementById("birthdate");
      birthdateElement.innerHTML = "Birthdate: " + dob.toDateString();
    }
  </script>
</head>
```

```
<body>
  <button onclick="printBirthdate()">Show Birthdate</button>
  <p id="birthdate"></p>
</body>
</html>
```

Write a script to define two JSON objects named as “division1” and “division2” having an array to store 5 names of students along with their roll number and list of subjects they opted.

```
var division1 = {
  "students": [
    {
      "name": "John Doe",
      "rollNumber": "D1-001",
      "subjects": ["Mathematics", "Science", "English"]
    },
    {
      "name": "Jane Smith",
      "rollNumber": "D1-002",
      "subjects": ["History", "Geography", "Spanish"]
    },
    {
      "name": "Mike Johnson",
      "rollNumber": "D1-003",
      "subjects": ["Physics", "Chemistry", "Biology"]
    }
  ]
};

var division2 = {
  "students": [
    {
      "name": "Sarah Thompson",
      "rollNumber": "D2-001",
      "subjects": ["Mathematics", "Science", "English"]
    },
    {
      "name": "Michael Brown",
      "rollNumber": "D2-002",
      "subjects": ["History", "Geography", "Spanish"]
    }
  ]
```

```
};

console.log("Division 1:", division1);
console.log("Division 2:", division2);
```

Write a JS to store an array of objects having height and name. display name and the height of the person with the highest height.

```
var people = [
  { "name": "John", "height": 175 },
  { "name": "Jane", "height": 160 },
  { "name": "Mike", "height": 185 },
  { "name": "Emily", "height": 191 },
  { "name": "Alex", "height": 190 }
];

var tallestPerson = people[0]; // Assume the first person is the tallest initially

for (var i = 1; i < people.length; i++) {
  if (people[i].height > tallestPerson.height) {
    tallestPerson = people[i];
  }
}

console.log("Person with the highest height:");
console.log("Name: " + tallestPerson.name);
console.log("Height: " + tallestPerson.height);
```

Write a script to define two JSON objects named as “division1” and “division2” has an array to store names of students. These name should be sorted alphabetically in the object and should be be written to the file. At last, both division objects should be visible with names sorted alphabetically in file.

```
const fs = require('fs');

// Define the division1 and division2 objects with student names
const division1 = {
  students: ['John', 'Jane', 'Mike', 'Emily', 'Alex']
};

const division2 = {
  students: ['Sarah', 'Michael', 'Emma', 'William', 'Olivia']
};
```

```
// Sort the student names alphabetically within each division
division1.students.sort();
division2.students.sort();

// Combine the division objects
const divisions = {
  division1,
  division2
};

// Convert the divisions object to JSON format
const jsonData = JSON.stringify(divisions);

// Write the JSON data to a file named "divisions.json"
fs.writeFile('divisions.json', jsonData, 'utf8', (err) => {
  if (err) {
    console.error('An error occurred while writing the file:', err);
  } else {
    console.log('The divisions data has been written to divisions.json');
  }
});
```

Write a JSON object which contains an array of 3 objects. Each object contains 2 properties name and age. Now, sort an array values by age in descending order. Print name in terminal as per the sorted age.

```
var data = {
  "people": [
    { "name": "John", "age": 25 },
    { "name": "Jane", "age": 30 },
    { "name": "Mike", "age": 20 }
  ]
};

// Sort the array of objects by age in descending order
data.people.sort((a, b) => b.age - a.age);

// Print the names in the terminal as per the sorted age
data.people.forEach((person) => {
console.log(person.name);
});
```

Write one JSON string with date property (yyyy-mm-dd) and print date in India standard time

```
var data = {
  "people": [
    { "name": "John", "age": 25 },
    { "name": "Jane", "age": 30 },
    { "name": "Mike", "age": 20 }
  ]
};

// Sort the array of objects by age in descending order
data.people.sort((a, b) => b.age - a.age);

// Print the names in the terminal as per the sorted age
data.people.forEach((person) => {
console.log(person.name);
});
```

// Write a Node.js program to CRUD operation of file management.
// 1)Create folder named "Hello".
// 2) Create file in it named abc.txt and enter data in to it.
// 3) Add more data at last in file.
// 4)Read data without getting buffer data at first.
// 5)rename file
// 6)Delete both file and folder.

```
const fs = require('fs');
try {
  // 1) Create folder named "Hello"
  fs.mkdirSync('Hello');
  console.log('Folder "Hello" created successfully.');

  // 2) Create file in it named abc.txt and enter data into it.
  fs.writeFileSync('Hello/abc.txt', 'Initial data in abc.txt');
  console.log('File "abc.txt" created and data added successfully.');

  // 3) Add more data at last in file.
  fs.appendFileSync('Hello/abc.txt', '\nAdditional data in abc.txt');
  console.log('Additional data added to "abc.txt" successfully.');

  // 4) Read data without getting buffer data at first.
  const data = fs.readFileSync('Hello/abc.txt', 'utf8');
  console.log('Data in "abc.txt":', data);
```

```
  // 5) Rename file
  fs.renameSync('Hello/abc.txt', 'Hello/xyz.txt');
  console.log('File "abc.txt" renamed to "xyz.txt" successfully.');

  // 6) Delete both file and folder.
  fs.unlinkSync('Hello/xyz.txt');
  fs.rmdirSync('Hello');
  console.log('File and folder deleted successfully.');
} catch (err) {
  console.error(err);
}
```

// Write a Node.Js program to sort an interger array, where all element are available in a file separated by white space. Print sorted array elements on node.js server.

```
const fs = require('fs');
const http = require('http');

// Read the file and retrieve the array elements
const fileData = fs.readFileSync('array.txt', 'utf8');
const array = fileData.trim().split(' ').map(Number);

// Sort the array
array.sort((a, b) => a - b);

// Create a Node.js server to display the sorted array elements
const server = http.createServer((req, res) => {
res.writeHead(200, { 'Content-Type': 'text/plain' });
res.end(`Sorted Array: ${array.join(' ')}`);
});

// Start the server const
port = 3000;
server.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

Write Node.Js code that display "Hello" with increasing font-size in interval of 50 miliseconds in blue color. When font-size reaches to 50 pixel it should stop.

```
<html>
```

```
<head>
    <style>
        p{
            color:blue;
        }
    </style>
</head>
<body>
    <p id="p1"> Hello</p>
    <button onclick="fun()">font-size</button>
    <script>
    font="10";
    function fun()
    {
        setInterval(
            ()=>
            {
                if(font<=50)
                {
                    font++

                    document.getElementById("p1").style.fontSize=font+"px";
                    document.getElementById("p1").style.color="red"
                }
            },50
        );
    }
</script>
</body>
</html>
```

Write node.js script to copy content of one file to the other file. data should be fetched from source.txt and insert to destination.txt

```
const fs = require('fs');
  // Read the content from the source file
  const data = fs.readFileSync('source.txt', 'utf8');

  // Write the content to the destination file
  fs.writeFileSync('destination.txt', data);

  console.log('Content copied successfully from source to destination.');
```

Write node.js script to print “Welcome Admin” on home page of server. If

user request for second page it display “This is second page” in italic fontstyle and if any other request is requested it shows “Page not found” message.

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/html' });

  if (req.url === '/') {
    res.write('<h1>Welcome Admin</h1>');
  } else if (req.url === '/second') {
    res.write('<em>This is second page</em>');
  } else {
    res.write('<h1>Page not found</h1>');
  }

  res.end();
});

const port = 3000;
server.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

Write a script to display sum of two numbers which are passed as an argument in function named Total. the sum should display in any html element. Use callback as third argument in Total function

```
<!-- // Write a script to display sum of two numbers which are passed as an
// argument in function named Total. the sum should display in any html
// element. Use callback as third argument in Total function -->

<!DOCTYPE html>
<html>
<head>
  <title>Sum Calculation</title>
</head>
<body>
  <h1>Sum Calculation</h1>
  <div id="result"></div>
  <script>
    function Total(num1, num2, callback) {
      const sum = num1 + num2;
      callback(sum);
    }
```

```
    function displayResult(result) {
      const outputElement = document.getElementById('result');
      outputElement.textContent = 'Sum: ' + result;
    }
      Total(5, 7, displayResult);

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

Write a script to define two JSON objects named as “division1” and “division2” having an array to store names of students. These names should be sorted alphabetically in the object and should be written to the file. At last, both division objects should be visible with names sorted alphabetically in File.

```
const fs = require('fs');

const division1 = {
  name: 'division1',
  students: ['John', 'Alice', 'David']
};

const division2 = {
  name: 'division2',
  students: ['Emily', 'Bob', 'Catherine']
};

// Sort the student names alphabetically within each division
division1.students.sort();
division2.students.sort();

const divisions = [division1, division2];

const data = JSON.stringify(divisions, null, 2);


  fs.writeFileSync('divisions.json', data);
  console.log('Data written to file successfully.');
```

Write a node.js script to write contents to the file in original manner. Delete file after finishing writing

```
const fs = require('fs');
```

```
const filePath = 'file.txt';
const content = 'This is the new content to be added.\n';

  // Append the content to the file
  fs.appendFileSync(filePath, content);
  console.log('Content written to file successfully.');
  // Delete the file
  fs.unlinkSync(filePath);

  console.log('File deleted successfully.');
```

Write a node.js script to write contents to the file in original manner. Delete file after finishing writing

```
const fs = require('fs');
const filePath = 'file.txt';
const content = 'This is the new content to be added.\n';
  // Append the content to the file
  fs.appendFileSync(filePath, content);
  console.log('Content written to file successfully.');
  // Delete the file
  fs.unlinkSync(filePath);
  console.log('File deleted successfully.');
```

Write a node.js script to jump on a specific code by specifying path on address bar of browser.

```
const http = require('http');

// Create an HTTP server
const server = http.createServer((req, res) => {
  // Get the URL path from the request
  const path = req.url;

  // Jump to specific code based on the path
  if (path === '/home') {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Welcome to the home page!');
  } else if (path === '/about') {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('This is the about page');
  } else {
    res.writeHead(404, { 'Content-Type': 'text/plain' });
  res.end('Page not found');
  }
```

```
});

// Start the server and listen on port 3000
server.listen(3000, () => {
  console.log('Server is running on http://localhost:3000');
});
```

Write a node.js script to print query string of URL on console as well as on file using ES6 Callback.

```
const fs = require('fs');
const url = require('url');

// Example URL with query string
const urlString = 'http://example.com/?param1=value1&param2=value2';

// Parse the URL and extract the query string
const parsedUrl = new URL(urlString);
const queryString = parsedUrl.search;

// Print the query string on the console
console.log('Query String:', queryString);


  // Write the query string to a file synchronously using a callback
  fs.writeFileSync('output.txt', queryString);
  console.log('Query string written to file successfully.');
```

Write a node.js script to load a simple.html file on nodejs web server & prints its contents as an html content.

```
const http = require('http');
const fs = require('fs');

const server = http.createServer((req, res) => {
  try {
    // Read the HTML file synchronously
    const data = fs.readFileSync('simple.html', 'utf8');

    // Set the content type to HTML
    res.setHeader('Content-Type', 'text/html');

    // Send the HTML content as the response
```

```
    res.end(data);
  } catch (err) {
    console.error('Error reading HTML file:', err);
    res.statusCode = 500;
    res.end('Internal Server Error');
  }
});

const port = 3000;
server.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```

Write a node.js script to find all prime no.s between 1-50 using external module having a function checkPrime(). This function returns Boolean value on the basis of a no. is prime or not prime. Write all necessary .js files.

// primeUtils.js

```
// Function to check if a number is prime
function checkPrime(num) {
  if (num <= 1) {
    return false;
  }

  for (let i = 2; i <= Math.sqrt(num); i++) {
    if (num % i === 0) {
      return false;
    }
  }

  return true;
}

module.exports = {
  checkPrime,
};
```

findprime.js

```
const { checkPrime } = require('./primeUtils');

// Find and print prime numbers between 1 and 50
for (let num = 1; num <= 50; num++) {
  if (checkPrime(num)) {
    console.log(num);
  }
```

}

Write a node.js script using event handling to consider an errorneous triangle to find area. Take fix values of all three sides.
(1) If any of the side is negative, then print the message “Sides must be positive” using event handler.
(2) If perimeter of triangle is negative then print the message “Perimeter must be positive” using event handler.
(3) Both above messages must be printed in sequence.

```
const EventEmitter = require('events');
const ee=new EventEmitter();

class Triangle {
  constructor(side1, side2, side3) {

    this.side1 = side1;
    this.side2 = side2;
    this.side3 = side3;
  }

  calculateArea() {
    if (this.side1 <= 0 || this.side2 <= 0 || this.side3 <= 0) {
      ee.emit('error', 'Sides must be positive');
      return;
    }

    const perimeter = this.side1 + this.side2 + this.side3;
    if (perimeter < 0) {
      ee.emit('error', 'Perimeter must be positive');
      return;
    }

    const semiPerimeter = perimeter / 2;
    const area = Math.sqrt(
    semiPerimeter *
        (semiPerimeter - this.side1) *
        (semiPerimeter - this.side2) *
        (semiPerimeter - this.side3)
    );

ee.emit('areaCalculated', area);
  }
  }
```

```
// Create an instance of the Triangle class
const triangle = new Triangle(3, 4, 5);

// Event handler for errors
ee.on('error', (errorMessage) => {
console.error(errorMessage);
});

// Event handler for area calculation
ee.on('areaCalculated', (area) => {
console.log('Area:', area);
});

// Calculate the area of the triangle
triangle.calculateArea();
```

Write a node.js script to create two listeners for a common event call their respective callbacks. Print number of events associated with an emitter. Remove one of the listeners & call remaining listners again. Print number of remaining listners also.

```
const EventEmitter = require('events');

// Create an instance of EventEmitter
const emitter = new EventEmitter();

// Listener 1
const listener1 = () => {
  console.log('Listener 1 called');
};

// Listener 2
const listener2 = () => {
  console.log('Listener 2 called');
};

// Attach the listeners to the 'commonEvent' event
emitter.on('commonEvent', listener1);
emitter.on('commonEvent', listener2);

// Get the number of events associated with the emitter const eventCount
= emitter.listenerCount('commonEvent'); console.log('Number of events
associated with the emitter:', eventCount);
```

```
// Emit the 'commonEvent' event
emitter.emit('commonEvent');

// Remove listener1
emitter.removeListener('commonEvent', listener1);

// Get the number of remaining listeners
const remainingListeners = emitter.listenerCount('commonEvent');
console.log('Number of remaining listeners:', remainingListeners);

// Emit the 'commonEvent' event again
emitter.emit('commonEvent');
```

Write a Node.js script to create a class student by assigning name & result in form of members. Create one member function named as topper of X which returns topper student object. Details of this topper student should be printed on file as well as on console.

```
const fs = require('fs');

class Student {
  constructor(name, result) {
  this.name = name;
  this.result = result;
  }

  static topperOfX(students) {
    // Find the topper student based on the result
    let topper = students[0];
    for (let i = 1; i < students.length; i++) {
      if (students[i].result > topper.result) {
        topper = students[i];
      }
    }

    return topper;
  }

  static printDetails(student) {
    console.log('Topper Details:');
    console.log('Name:', student.name);
    console.log('Result:', student.result);
```

```
    fs.writeFileSync('topper.txt', `Name: ${student.name}\nResult: ${student.result}`);
    console.log('Topper details saved to topper.txt');
  }
}

// Create an array of student objects
const students = [
  new Student('John', 90),
  new Student('Alice', 95),
  new Student('Bob', 80),
];

// Get the topper student
const topper = Student.topperOfX(students);

// Print the details of the topper student
Student.printDetails(topper);
```

Write a Node.js script to create a class person by assigning name & age in form of members. Create one member function named as elder of X which returns elder person object. Details of this elder person should be printed on file as well as on console.

```
const fs = require('fs');

class Person {
  constructor(name, age) {
  this.name = name;
  this.age = age;
  }

  elderOfX(people) {
    // Find the elder person based on the age
    let elder = people[0];
    for (let i = 1; i < people.length; i++) {
      if (people[i].age > elder.age) {
        elder = people[i];
      }
    }

    return elder;
  }

  printDetails(person) {
```

```
    console.log('Elder Person Details:');
    console.log('Name:',  person.name);
    console.log('Age:', person.age);

    fs.writeFileSync('elder.txt', `Name: ${person.name}\nAge: ${person.age}`);
    console.log('Elder person details saved to elder.txt');
  }
}

// Create an array of person objects
const people = [
  new Person('John', 35),
  new Person('Alice', 40),
  new Person('Bob', 50),
];

// Create an instance of the Person class
const personInstance = new Person();

// Get the elder person
const elderPerson = personInstance.elderOfX(people);

// Print the details of the elder person
personInstance.printDetails(elderPerson);
```

Write a node.js script to create a class time & assign members hour, minute& second. Create two objects of time class & add both the time objects so that it should return is third time object. Third time object should have hour, minute & second such that after addition if second exceeds 60 then minute value should be incremented. If minute exceeds 60 then hour value should be Incremented.

```
class Time {
  constructor(hour, minute, second) {
    this.hour = hour;
    this.minute = minute;
    this.second = second;
  }

  add(otherTime) {
    const totalSeconds = this.second + otherTime.second;
    const additionalMinutes = Math.floor(totalSeconds / 60);
    const remainingSeconds = totalSeconds % 60;

    const totalMinutes = this.minute + otherTime.minute + additionalMinutes;
```

```
    const additionalHours = Math.floor(totalMinutes / 60);
    const remainingMinutes = totalMinutes % 60;

    const totalHours = this.hour + otherTime.hour + additionalHours;

    return new Time(totalHours, remainingMinutes, remainingSeconds);
  }
}

// Create two time objects
const time1 = new Time(10, 25, 40);
const time2 = new Time(2, 35, 20);

// Add the two time objects
const result = time1.add(time2);

// Print the result
console.log('Result:', result);
```

Write a node.js script to create calculator using external module having a function add(), sub(), mul(), div(). This function returns result of calculation. Write all necessary .js files.

Calculator.js

```
exports.add = (a, b) => {
return a + b;
};

exports.sub = (a, b) => {
  return a - b;
};

exports.mul = (a, b) => {
  return a * b;
};

exports.div = (a, b) => {
  return a / b;
};
```

86.js

```
const calculator = require('./calculator');

const num1 = 10;
```

```
const num2 = 5;

const sum = calculator.add(num1, num2);
console.log(`${num1} + ${num2} = ${sum}`);

const difference = calculator.sub(num1, num2);
console.log(`${num1} - ${num2} = ${difference}`);

const product = calculator.mul(num1, num2);
console.log(`${num1} * ${num2} = ${product}`);

const quotient = calculator.div(num1, num2);
console.log(`${num1} / ${num2} = ${quotient}`);
```

Create an event emitter instance and register a couple of callbacks.

```
// Create an event emitter instance and register a couple of callbacks.
const EventEmitter = require('events');

// Create an instance of EventEmitter
const myEmitter = new EventEmitter();

// Register a callback for the 'event1' event
myEmitter.on('event1', () => {
console.log('Event 1 occurred');
});

// Register another callback for the 'event1' event
myEmitter.on('event1', () => {
console.log('Another callback for event 1');
});

// Emit the 'event1' event
myEmitter.emit('event1');

// Register a callback for the 'event2' event
myEmitter.on('event2', () => {
console.log('Event 2 occurred');
});

// Emit the 'event2' event
myEmitter.emit('event2');
```

Explain node js events with appropriate example with all event methods.

```
const EventEmitter = require('events');

// Create a new event emitter instance
const myEmitter = new EventEmitter();

// Register an event listener using `on` method
myEmitter.on('greet', () => {
console.log('Hello!');
});

// Register a one-time event listener using `once` method
myEmitter.once('greet', () => {
  console.log('This will only be called once');
});

// Emit the 'greet' event
myEmitter.emit('greet');

// Get the count of listeners for an event using `listenerCount` method
const listenerCount = EventEmitter.listenerCount(myEmitter, 'greet');
console.log(`Number of listeners for 'greet' event: ${listenerCount}`);

// Get all the event names for an emitter using `eventNames` method
const eventNames = myEmitter.eventNames();
console.log('Event names:', eventNames);

// Check if an emitter has a listener for a specific event using `listenerCount` method
const hasListener = myEmitter.listenerCount('greet') > 0;
console.log(`Does 'greet' event have a listener? ${hasListener}`);

// Remove a specific event listener using `off` method
const listener = () => {
  console.log('This listener will be removed');
};
myEmitter.on('greet', listener);
myEmitter.off('greet', listener);

// Remove all event listeners for a specific event using `removeAllListeners` method
myEmitter.removeAllListeners('greet');

// Emit the 'greet' event again
```

myEmitter.emit('greet');

Write a node js script to demonstrate Chalk module.

```
const chalk = require('chalk');

// Basic colors console.log(chalk.red('This is
red text')); console.log(chalk.green('This is
green text')); console.log(chalk.blue('This is
blue text'));

// Background colors
console.log(chalk.bgYellow.black('This has a yellow background and black text'));
console.log(chalk.bgCyan.white('This has a cyan background and white text'));

// Styled text
console.log(chalk.bold('This is bold text'));
console.log(chalk.underline('This is underlined text'));
console.log(chalk.inverse('This has inverted colors'));

// Combining styles
console.log(chalk.bold.blue('This is bold and blue text'));

// Chaining styles
console.log(chalk.red.bgYellow.bold('This has chained styles'));

// Disabling Chalk chalk.enabled = false;
console.log(chalk.red('This will not be styled'));
```

Explain validator in NPMjs. const validator = require('validator');

```
const email = 'test@example.com';
const url = 'https://www.example.com';
const number = '12345';

console.log('isEmail:', validator.isEmail(email));
console.log('isURL:', validator.isURL(url));
console.log('isNumeric:', validator.isNumeric(number));
console.log('isAlpha:', validator.isAlpha('Hello'));
console.log('isAlphanumeric:', validator.isAlphanumeric('Hello123'));
console.log('isLowercase:', validator.isLowercase('hello'));
```

```
console.log('isUppercase:', validator.isUppercase('HELLO'));
console.log('isDate:', validator.isDate('2022-01-01'));
```