```
setTimeout(() => {
   console.log('hi test');
}, 1000);//ata global er modda ase
console.log(global);
console.log(__dirname);//aita global er modda nai
console.log(__filename);//aita global er modda nai



//module system
//people.js
const people = ['aff', 'sdff'];
module.exports = people;

//index.js
const people = ['chinki', 'minki'];
console.log(module);
module.exports = people;




//module export require ai gula to global er modda nai to kotha heke ashlo?











// Event emitter = event rase kora/event hoiyase
const EventEmitter = require('events');
const EventEmiter = require('events');
// EventEmiter=class
const emitter = new EventEmitter();
// raise an event or event hoilo or gotlo
emitter.emit('bellRing');

// ai tuko diya run korle  kisu hobe na ..karon event gotlo goter por k hobe boli nai
```

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

const EventEmiter = require('events');

// EventEmiter=class

const emitter = new EventEmitter();


//register a listener for bellRing event

// listen koralam

emitter.on('bellRing', ()=>{

   console.log('we need to run');

});

// raise an event or event hoilo or gotlo

emitter.emit('bellRing');



emitter.on('bellRing', () => {

   console.log('we need to run');

});
```

```
//ai code kaj korbe na


const EventEmitter = require('events');


const emitter = new EventEmitter();


//2 second por run korate chile

setTimeout(() => {

   emitter.emit('bellRing');

}, 2000);

emitter.on('bellRing', () => {

   console.log('we need to run');
});




const EventEmitter = require('events');

const emitter = new EventEmitter();
//register listener for belling event
emitter.on('bellRing', (period) => {
   console.log(` we need to run because ${period}`);
});
//rase an event
setTimeout(() => {
   emitter.emit('bellRing', 'second period ended');
}, 2000);
```

```
//event parametter pase

const EventEmitter = require('events');

const emitter = new EventEmitter();
//register listener for belling event
emitter.on('bellRing', ({period, text}) => {//obeject pelam distucture kore
   console.log(` we need to run because ${period}${text}`);
});
//rase an event
setTimeout(() => {
   emitter.emit('bellRing', {//object akare pase kortasi karon multiple parametter
      period: 'first',
      text: 'period ended',
   });
}, 2000);
```

```
//Extending event
//school.js
const EventEmitter = require('events');
class School extends EventEmitter {
    startPeriod() {
        console.log('class started');
        // rase an event
        setTimeout(() => {
            this.emit('bellRing', {
                period: 'first',
                text: 'period ended',
            });
        }, 2000);
    }
}
module.exports = School;

//index.js
const School = require('./school');

const school = new School();
// register listener for belling event
school.on('bellRing', ({ period, text }) => {
    // obeject pelam distucture kore
    console.log(` we need to run because ${period}${text}`);
});
// rase an event
school.startPeriod();
```

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

const server = http.createServer((req, res) => {
    if (req.url === '/') {
        res.write('hello');
        res.end();
    } else if (req.url === '/about') {
        res.write('hellofgf');
        res.end();
    } else {
        res.write('Not found');
        res.end();
    }
});
server.listen(33333);
console.log('listening on port 33333');
```

```
//path module

const path = require('path');
const myPath = 'G:All Nodejspraticeindex.js';
console.log(path.dirname(myPath));

//or
const path = require('path');
const myPath = path.dirname('G:\All Nodejs\pratice\index.js');
console.log(myPath);

const path = require('path');
const myPath = path.parse('G:\All Nodejs\pratice\index.js');
console.log(myPath);
```

```
/os module
const os = require('os');
console.log(os.freemem());
console.log(os.homedir());
console.log(os.cpus());




//fs module

const fs = require('fs');

fs.writeFileSync('myFile.txt', 'hello');
fs.appendFileSync('myFile.txt', 'hello');
const data = fs.readFileSync('myfile.txt');
console.log(data);
console.log(data.toString());

// callback
fs.readFile('myFile.txt', (err, data) => {
   console.log(data.toString());
});
console.log('what are you doing');


const fs = require('fs');
fs.writeFile('myFile', 'Hello', (err) => {
  if(err){
     console.log('Error');
  }else{
   console.log('Data Save');
  }
})
```

```
//read strem
const http = require('http');
const server = http.createServer((req, res) => {
  if (req.url === '/') {
    res.write('<html><head><title>Form</title></head>');
    res.write(
      '<body><form method="post" action="/process"><input name="message"/></form></body>',
    );
    res.end();
  } else if (req.url === '/process' && req.method === 'POST') {
    req.on('data', (chunk) => {
      console.log(chunk.toString());
    });
    res.write('Thank you');
    res.end();
  } else {
    res.write('Not found');
    res.end();
  }
});
server.listen(33000);
console.log('listening on port 33000');
```

```
//Read strem (strem way te data dicce)
const http = require('http');

const server = http.createServer((req, res) => {
  if (req.url === '/') {
    res.write('<html><head><title>Form</title></head>');
    res.write(
      '<body><form method="post" action="/process"><input name="message"/></form></body>',
    );
    res.end();
  } else if (req.url === '/process' && req.method === 'POST') {
    const body = []; //ekta ekta chunk ashtase
    req.on('data', (chunk) => {
      body.push(chunk);//er ashter por  body the chunk duktase
    });
    req.on('end', () => {
      console.log('strem finished');
      const parsedBody = Buffer.concat(body).toString();// body er modda ja pura  data ta ase shetake shob gula ke jug korlam buffer er modda raklam
      console.log(parsedBody);
      res.write('Thank you');
      res.end();
    });
    } else {
        res.write('Not found');
        res.end();
    }
});
server.listen(33000);
console.log('listening on port 33000');
```

```
//write strem
const fs = require('fs');

const ourReadStream = fs.createReadStream(`${__dirname}/bigdata.txt`);

const ourWriteStream = fs.createWriteStream(`${__dirname}/output.txt`);
ourReadStream.on('data', (chunk) => {
    ourWriteStream.write(chunk);
}); //2 line er bodole nicher aita o kora jai mane pipe

//ourWriteStream.pipe(ourWriteStream);
```

```
//pipe
//write strem
const fs = require('fs');

const ourReadStream = fs.createReadStream(`${__dirname}/bigdata.txt`);

const ourWriteStream = fs.createWriteStream(`${__dirname}/output.txt`);
ourWriteStream.pipe(ourWriteStream);
```

```
//pipe
// strem event ar shohoj upai pipeline
// read write er jamela korte hoitase na
const http = require('http');
const fs = require('fs');

const server = http.createServer((req, res) => {
   const myReadStrem = fs.createReadStream(`${__dirname}/bigdata.txt`, 'utf8');
   myReadStrem.pipe(res);
});
server.listen(3000);
console.log('Listem on port 3000')
```

//server create

```
// Title: Up Time monitoring application
// Description: A restful api to monitor up down time of user defined links
// Author: Maksuda Mila
// Date: 7/5/2021

//dependencies
const http = require('http');
const url = require('url');
const {StringDecoder} = require('string_decoder');
// app object
const app = {};

//app configuration
app.config = {
   port:3000,
};

//create server
app.createServer = () => {
    const server = http.createServer(app.handeReqRes);
    server.listen(app.config.port, () => {
        console.log(`listening to port ${app.config.port}`);
    });
};

//handlen request response
app.handeReqRes = (req, res) => {
//response handle
res.end('All of you Hello world') ;
};
// start the server
app.createServer();
```

//parsing request

//http://localhost:5555/ about? a = b & v= 3   (query string)(postman)

```
//index.js
// Title: Up Time monitoring application
// Description: A restful api to monitor up down time of user defined links
// Author: Maksuda Mila
// Date: 7/5/2021

//dependencies
const http = require('http');
const url = require('url');
const {StringDecoder} = require('string_decoder');
```

```
// app object
const app = {};

//app configuration
app.config = {
   port:3000,
};

//create server
app.createServer = () => {
    const server = http.createServer(app.handeReqRes);
    server.listen(app.config.port, () => {
        console.log(`listening to port ${app.config.port}`);
    });
};

//handlen request response
app.handeReqRes = (req, res) => {
 //request handle
    //get the url and parse it
    const parsedUrl = url.parse(req.url, true);
    // console.log(parsedUrl);
    const path = parsedUrl.pathname;
console.log(path);
const trimedPath = path.replace(/^\/+|\/+$/g, '');//.replace() = method
console.log(trimedPath);
const method = req.method.toLowerCase();
console.log(method);
const queryStingObject = parsedUrl.query;
console.log(queryStingObject);
const headerObject = req.headers;
const decoder = new StringDecoder('utf-8');
let realData = '';
req.on('data', (buffer) => {
    realData+= decoder.write(buffer);

});
req.on('end', () => {
    realData += decoder.end();
    console.log(realData);
    //response handle
res.end('All of you Hello world')  ;
});
};
// start the server
app.createServer();
```

Helpers/handleReqRes.js

```
      // Title: Handle Request Response

// Description: Handle Request and response
// Author: Maksuda Mila
// Date: 9/5/2021
```

```
//dependencies
const url = require('url');
const {StringDecoder} = require('string_decoder');
const routes = require('../routes');
const {notFoundHandler} = require('../handlers/routeHandlers/notFoundHandler')

//module scaffolding
const handler = {};
handler.handleReqRes = (req, res) => {
    //request handle
       //get the url and parse it
       const parsedUrl = url.parse(req.url, true);
       // console.log(parsedUrl);
       const path = parsedUrl.pathname;
   console.log(path);
   const trimedPath = path.replace(/^\/+|\/+$/g, '');//.replace() = method
   console.log(trimedPath);
   const method = req.method.toLowerCase();
   console.log(method);
   const queryStingObject = parsedUrl.query;
   console.log(queryStingObject);
   const headerObject = req.headers;

   const requestProperties = {
       parsedUrl,
       path,
       trimedPath,
       method,
       queryStingObject,
       headerObject,

   };

   const decoder = new StringDecoder('utf-8');
   let realData = '';

   const chosenHandler = routes[trimedPath] ? routes[trimedPath] : notFoundHandler;

   chosenHandler(requestProperties, (statusCode, payload) => {
   statusCode = typeof statusCode === 'number' ? statusCode : 500;
   payload = typeof payload === 'object' ? payload : {};
   const payloadString = JSON.stringify(payload);

   //return the final response
   res.writeHead(statusCode);
   res.end(payloadString);
   });

   req.on('data', (buffer) => {
       realData+= decoder.write(buffer);

   });
   req.on('end', () => {
       realData += decoder.end();
       console.log(realData);
       //response handle
   res.end('All of you Hello world')  ;
   });
   };
module.exports = handler;
```

router.js

```
// title: Routes
// Description: Application Routes
// Author: Maksuda mila
// Date: 10/5/2021

//dependencies
const {sampleHandler} = require('./handlers/routeHandlers/sampleHandler')
const routes = {
    'sample': sampleHandler,      //sample= property
};
```

```
module.exports = routes;
```

Handler/sampleHandler.js

```
// title: Sample HAndler
// Description: Sample HAndler
// Author: Maksuda mila
// Date: 10/5/2021

const handler ={};
handler.sampleHandler = (requestProperties, callback) => {
callback(200, {
    message: 'This is sample url'
});
};
module.exports = handler;
```

handler/notFoundHandler

```
// title: Not found handler
// Description: 404 Not found handler
// Author: Maksuda mila
// Date: 10/5/2021

const handler ={};
handler.notFoundHandler = (requestProperties, callback) => {
    callback(404, {
        message: 'Your request was not found'
    });
};
module.exports = handler;
```

routes.js

```
// title: Routes
// Description: Application Routes
// Author: Maksuda mila
// Date: 10/5/2021

//dependencies
const {sampleHandler} = require('./handlers/routeHandlers/sampleHandler')
const routes = {
    'sample': sampleHandler,      //sample= property
};

module.exports = routes;
```

//helpers/ environments.js

```
// Title: Environments
// Description: handle all enviromment related things
// Author: Maksuda Mila
// Date: 13/5/2021

//dependencies
```

```
//module scaffolding
const environments = {};

environments.staging = {
    port: 3000,
    envName:'staging'
};
environments.production = {
    port: 5000,
    envName: 'production'
}
//determine which environment was passes
const currentEnvironment = typeof(process.env.NODE_ENV) === 'string' ? process.env.NODE_ENV : 'staging';

// export corresponding environment object
const envirommentToExport = typeof(environments[currentEnvironment]) === 'object' ? environments[currentEnvironment] : e
nvironments.staging;
module.exports = envirommentToExport;
```

//lib/data.js

```
// dependencies
const fs = require('fs');
const path = require('path');

// module scaffolding
const lib = {};

// base directory of the data folder
lib.basedir = path.join(__dirname, '/../.data/');

// write data to file
lib.create = (dir, file, data, callback) => {
    // open file for writing
    fs.open(`${lib.basedir + dir}/${file}.json`, 'wx', (err, fileDescriptor) => {
        if (!err && fileDescriptor) {
            // convert data to stirng
            const stringData = JSON.stringify(data);

            // write data to file and then close it
            fs.writeFile(fileDescriptor, stringData, (err2) => {
                if (!err2) {
                    fs.close(fileDescriptor, (err3) => {
                        if (!err3) {
                            callback(false);
                        } else {
                            callback('Error closing the new file!');
                        }
                    });
                } else {
                    callback('Error writing to new file!');
                }
            });
        } else {
            callback('There was an error, file may already exists!');
        }
    });
};

// read data from file
lib.read = (dir, file, callback) => {
    fs.readFile(`${lib.basedir + dir}/${file}.json`, 'utf8', (err, data) => {
        callback(err, data);
    });
};

// update existing file
lib.update = (dir, file, data, callback) => {
    // file open for writing
    fs.open(`${lib.basedir + dir}/${file}.json`, 'r+', (err, fileDescriptor) => {
```

```javascript
        if (!err && fileDescriptor) {
            // convert the data to string
            const stringData = JSON.stringify(data);

            // truncate the file
            fs.ftruncate(fileDescriptor, (err1) => {
                if (!err1) {
                    // write to the file and close it
                    fs.writeFile(fileDescriptor, stringData, (err2) => {
                        if (!err2) {
                            // close the file
                            fs.close(fileDescriptor, (err3) => {
                                if (!err3) {
                                    callback(false);
                                } else {
                                    callback('Error closing file!');
                                }
                            });
                        } else {
                            callback('Error writing to file!');
                        }
                    });
                } else {
                    callback('Error truncating file!');
                }
            });
        } else {
            console.log(`Error updating. File may not exist`);
        }
    });
};

// delete existing file
lib.delete = (dir, file, callback) => {
    // unlink file
    fs.unlink(`${lib.basedir + dir}/${file}.json`, (err) => {
        if (!err) {
            callback(false);
        } else {
            callback(`Error deleting file`);
        }
    });
};

// list all the items in a directory
lib.list = (dir, callback) => {
    fs.readdir(`${lib.basedir + dir}/`, (err, fileNames) => {
        if (!err && fileNames && fileNames.length > 0) {
            const trimmedFileNames = [];
            fileNames.forEach((fileName) => {
                trimmedFileNames.push(fileName.replace('.json', ''));
            });
            callback(false, trimmedFileNames);
        } else {
            callback('Error reading directory!');
        }
    });
};

module.exports = lib;
```