# Introduction

* Node.js is a javascript runtime built on Chrome’s V8 javascript engine

1. Node runs on a server, not in a browser (backend not frontend)
2. The console is the terminal window
3. Global object instead of window object

### server.js

| console.log(global) |
| --- |

### Output

| <ref \*1> Object [global] {  global: [Circular \*1],  queueMicrotask: [Function: queueMicrotask],  clearImmediate: [Function: clearImmediate],  setImmediate: [Function: setImmediate] {  [Symbol(nodejs.util.promisify.custom)]: [Getter]  },  structuredClone: [Getter/Setter],  clearInterval: [Function: clearInterval],  clearTimeout: [Function: clearTimeout],  setInterval: [Function: setInterval],  setTimeout: [Function: setTimeout] {  [Symbol(nodejs.util.promisify.custom)]: [Getter]  },  atob: [Getter/Setter],  btoa: [Getter/Setter],  performance: [Getter/Setter],  fetch: [AsyncFunction: fetch],  crypto: [Getter]  } |
| --- |

1. Has common core modules that we will explore
   * These modules relate to the operating system, file system, and other things that we can do on the server
2. Common JS modules instead of ES6 modules

## OS and path core modules

### server.js

| const os = require('os')  const path = require('path')  console.log(os.type())  console.log(os.version())  console.log(os.homedir())  console.log(\_\_dirname)  console.log(\_\_filename)  console.log(path.dirname(\_\_filename))  console.log(path.basename(\_\_filename))  console.log(path.extname(\_\_filename))  console.log(path.parse(\_\_filename)) |
| --- |

### Output

| Windows\_NT  Windows 10 Home  C:\Users\kotag  C:\Users\kotag\OneDrive\Desktop\javascript testing  C:\Users\kotag\OneDrive\Desktop\javascript testing\server.js  C:\Users\kotag\OneDrive\Desktop\javascript testing  server.js  .js  {  root: 'C:\\',  dir: 'C:\\Users\\kotag\\OneDrive\\Desktop\\javascript testing',  base: 'server.js',  ext: '.js',  name: 'server'  } |
| --- |

## Custom Modules and import/export options

### server.js

| const math = require('./math')  const {subtract} = require('./math')  const {multiply} = require('./math')  const {divide} = require('./math')  console.log(math.add(2, 3))  console.log(subtract(2, 3))  console.log(multiply(2, 3))  console.log(divide(2, 3)) |
| --- |

### math.js

| exports.add = (a, b) => a + b;  exports.subtract = (a, b) => a - b;  exports.multiply = (a, b) => a \* b;  exports.divide = (a, b) => a / b;  // const add = (a, b) => a + b;  // const subtract = (a, b) => a - b;  // const multiply = (a, b) => a \* b;  // const divide = (a, b) => a / b;  //module.exports = {add, subtract, multiply, divide} //{} not [] |
| --- |

### Output

| 5  -1  6  0.6666666666666666 |
| --- |

1. Missing some JS APIs like fetch

# Reading and Writing Files

* Node JS documentation - https://nodejs.org/en/
  + Node JS website => DOCS => LTS => File system
* fs - file system

## Async example

### server.js

| const fs = require('fs')  fs.readFile('starter.txt', 'utf8', (err, data) => {  if(err) throw err  console.log(data)  // console.log(data.toString())  })  console.log(`Hello...`) //printed before the data is printed  // exit on uncaught errors  process.on('uncaughtException', err => {  console.error(`There was an uncaught error: ${err}`)  process.exit(1)  }) |
| --- |

### Output

| Hello...  Hello my name is Dakota! |
| --- |

## Path Module

### server.js

| const fs = require('fs')  const path = require('path')  fs.readFile(path.join(\_\_dirname, 'files', 'starter.txt'), 'utf8', (err, data) => {  if(err) throw err  console.log(data)  // console.log(data.toString())  })  console.log(`Hello...`) //printed before the data is printed  // exit on uncaught errors  process.on('uncaughtException', err => {  console.error(`There was an uncaught error: ${err}`)  process.exit(1)  }) |
| --- |

### Output

| Hello...  Hello my name is Dakota! |
| --- |

## Write a file

### server.js

| //path, what you want to write to the file, err  fs.writeFile(path.join(\_\_dirname, 'files', 'reply.txt'), 'Nice to meet you.', (err) => { //reply isn't a file that is already made, 'utf8' is by default,  if(err) throw err  console.log('Write Complete')  }) |
| --- |

* We now have a new file in our files folder called ‘reply.txt’

## Updating a file

### server.js

| //path, what you want to add to the file, err  //test isn't a file that is already made  fs.appendFile(path.join(\_\_dirname, 'files', 'test.txt'), 'testing text', (err) => {  if(err) throw err  console.log('Append Complete')  }) |
| --- |

## Nested operations

### server.js

| fs.writeFile(path.join(\_\_dirname, 'files', 'reply.txt'), 'Nice to meet you.', (err) => {  if(err) throw err  console.log('Write Complete')  // \n creates a new line  fs.appendFile(path.join(\_\_dirname, 'files', 'reply.txt'), '\n\n Yes it is', (err) => {  if(err) throw err  console.log('Append Complete')  })  }) |
| --- |

## Callback

### server.js

| fs.writeFile(path.join(\_\_dirname, 'files', 'reply.txt'), 'Nice to meet you.', (err) => {  if(err) throw err  console.log('Write Complete')  fs.appendFile(path.join(\_\_dirname, 'files', 'reply.txt'), '\n\nYes it is', (err) => {  if(err) throw err  console.log('Append Complete')  //renamed the reply file to newReply  fs.rename(path.join(\_\_dirname, 'files', 'reply.txt'), path.join(\_\_dirname, 'files', 'newReply.txt'), (err) => {  if(err) throw err  console.log('Rename Complete')  })  })  }) |
| --- |

* First it wrote, then it appended, then it renamed

## fsPromises

### server.js

| const fsPromises = require('fs').promises //importing the file system promises instead of just fs  const path = require('path')  const fileOps = async () => {  try {  const data = await fsPromises.readFile(path.join(\_\_dirname, 'files', 'starter.txt'), 'utf8')  console.log(data)  } catch (err) {  console.error(err)  }  }  fileOps() |
| --- |

### Output

| Hello my name is Dakota! |
| --- |

## Replace Callback with Async Await

### server.js

| const fileOps = async () => {  try {  const data = await fsPromises.readFile(path.join(\_\_dirname, 'files', 'starter.txt'), 'utf8')  console.log(data)  await fsPromises.writeFile(path.join(\_\_dirname, 'files', 'promiseWrite.txt'), data) //writes the file  await fsPromises.appendFile(path.join(\_\_dirname, 'files', 'promiseWrite.txt'), '\n\nNice to meet you') //appends the file  await fsPromises.rename(path.join(\_\_dirname, 'files', 'promiseWrite.txt'), path.join(\_\_dirname, 'files', 'promiseComplete.txt')) //rename  const newData = await fsPromises.readFile(path.join(\_\_dirname, 'files', 'promiseComplete.txt'), 'utf8')  console.log(newData)  } catch (err) {  console.error(err)  }  }  fileOps() |
| --- |

## Deleting a file

### server.js

| const fileOps = async () => {  try {  const data = await fsPromises.readFile(path.join(\_\_dirname, 'files', 'starter.txt'), 'utf8')  console.log(data)  await fsPromises.unlink(path.join(\_\_dirname, 'files', 'starter.txt')) //deletes starter.txt  await fsPromises.writeFile(path.join(\_\_dirname, 'files', 'promiseWrite.txt'), data) //writes the file  await fsPromises.appendFile(path.join(\_\_dirname, 'files', 'promiseWrite.txt'), '\n\nNice to meet you') //appends the file  await fsPromises.rename(path.join(\_\_dirname, 'files', 'promiseWrite.txt'), path.join(\_\_dirname, 'files', 'promiseComplete.txt')) //rename  const newData = await fsPromises.readFile(path.join(\_\_dirname, 'files', 'promiseComplete.txt'), 'utf8')  console.log(newData)  } catch (err) {  console.error(err)  }  }  fileOps() |
| --- |

## Stream data listener

* Readstream and Writestream for large files

### stream.js

| const fs = require('fs')  const rs = fs.createReadStream('./files/lorem.txt', {encoding: 'utf8'})  const ws = fs.createWriteStream('./files/new-lorem.txt')  //listens for the data  rs.on('data', (dataChunk) => {  ws.write(dataChunk)  }) |
| --- |

## Piping data streams

### stream.js

| const fs = require('fs')  const rs = fs.createReadStream('./files/lorem.txt', {encoding: 'utf8'})  const ws = fs.createWriteStream('./files/new-lorem.txt')  rs.pipe(ws) //more efficient than stream data listeners |
| --- |

## Creating and checking a Directory

### dir.js

| const fs = require('fs')  if (!fs.existsSync('./new')){ //if it doesn't exist create it  fs.mkdir('./new', (err) => {  if (err) throw err  console.log('Directory created')  }) //mkdir - make directory  } |
| --- |

## Removing a Directory

### dir.js

| const fs = require('fs')  if (fs.existsSync('./new')){ //if it does exist remove it  fs.rmdir('./new', (err) => {  if (err) throw err  console.log('Directory removed')  }) //rmdir - remove directory  } |
| --- |

# NPM Node Package Manager Modules