Міністерство освіти і науки України

Черкаський державний технологічний університет

Звіт

про виконання лабораторної роботи №4 з навчальної дисципліни

«Програмування WEB додатків»

Виконав: студент   
 Групи ПЗС-2044

Оратівський С.І.

Перевірив:

Метелап В.В.

Черкаси

2022

**Тема: NodeJs**

**Завдання: Опрацювати Node Js на метаніті**

Кінцевий результат першого розділу

[SerhiiOrSoratik/internet-shop-Web-Application-Programming at 3db7caa9cf0e837b41faa60f2b8d84d13e5013b1 (github.com)](https://github.com/SerhiiOrSoratik/internet-shop-Web-Application-Programming/tree/3db7caa9cf0e837b41faa60f2b8d84d13e5013b1)

App.js

const http = require("http");  
http.createServer(function(request,response){  
  
 response.end("Hello NodeJS!");  
  
}).listen(3000, "127.0.0.1",function(){  
 *console*.log("Сервер начал прослушивание запросов на порту 3000");  
});

Кінцевий результат другого розділу

[SerhiiOrSoratik/internet-shop-Web-Application-Programming at bafc7ff0958521614caef62de2776d1f7b7398fc (github.com)](https://github.com/SerhiiOrSoratik/internet-shop-Web-Application-Programming/tree/bafc7ff0958521614caef62de2776d1f7b7398fc)

App.js

const fs = require("fs");  
const zlib = require("zlib");  
  
let readableStream = fs.createReadStream("hello.txt", "utf8");  
  
let writeableStream = fs.createWriteStream("hello.txt.gz");  
  
let gzip = zlib.createGzip();  
  
readableStream.pipe(gzip).pipe(writeableStream);

gretting.js

let currentDate = new ***Date***();  
  
***global***.date = currentDate;  
  
module.exports.getMessage = function(){  
 let hour = currentDate.getHours();  
 if(hour >16)  
 return "Добрый вечер, " + ***global***.name;  
 else if(hour >10)  
 return "Добрый день, " + ***name***;  
 else  
 return "Доброе утро, " + ***name***;  
}

user.js

function User(name, age){  
  
 this.name = name;  
 this.age = age;  
 this.displayInfo = function(){  
  
 ***console***.log(`Имя: ${this.name} Возраст: ${this.age}`);  
 }  
}  
User.prototype.sayHi = function() {  
 ***console***.log(`Привет, меня зовут ${this.name}`);  
};  
  
module.exports = User;

Кінцевий результат третього розділу

[SerhiiOrSoratik/internet-shop-Web-Application-Programming at e8169c8b3eea4be2dbbeaa8fd15f29ae69456421 (github.com)](https://github.com/SerhiiOrSoratik/internet-shop-Web-Application-Programming/tree/e8169c8b3eea4be2dbbeaa8fd15f29ae69456421)

App.js

const http = require("http");  
const fs = require("fs");  
  
http.createServer(function(request, response){  
  
 fs.readFile("index.html", "utf8", function(error, data){  
  
 let message = "Изучаем Node.js";  
 let header = "Главная страница";  
 data = data.replace("{header}", header).replace("{message}", message);  
 response.end(data);  
 })  
}).listen(3000);

about.html

<!DOCTYPE html>  
<html>  
<head>  
 <title>О сайте</title>  
 <meta charset="utf-8" />  
</head>  
<body>  
<h1>О сайте</h1>  
</body>  
<html>

index.html

<!DOCTYPE html>  
<html>  
<head>  
 <title>Главная</title>  
 <meta charset="utf-8"/>  
</head>  
<body>  
<h1>{header}</h1>  
<p>{message}</p>  
</body>  
<html>

Кінцевий результат четвертого розділу

[SerhiiOrSoratik/internet-shop-Web-Application-Programming at f3f3e09e2d0dc697a3d31e03e86f9407de752ba5 (github.com)](https://github.com/SerhiiOrSoratik/internet-shop-Web-Application-Programming/tree/f3f3e09e2d0dc697a3d31e03e86f9407de752ba5)

app.js

const express = require("express");  
const fs = require("fs");  
  
const app = express();  
const jsonParser = express.json();  
  
app.use(express.static(\_\_dirname + "/public"));  
  
const filePath = "users.json";  
app.get("/api/users", function(req, res){  
  
 const content = fs.readFileSync(filePath,"utf8");  
 const users = JSON.parse(content);  
 res.send(users);  
});  
// получение одного пользователя по id  
app.get("/api/users/:id", function(req, res){  
  
 const id = req.params.id; // получаем id  
 const content = fs.readFileSync(filePath, "utf8");  
 const users = JSON.parse(content);  
 let user = null;  
 // находим в массиве пользователя по id  
 for(var i=0; i<users.length; i++){  
 if(users[i].id==id){  
 user = users[i];  
 break;  
 }  
 }  
 // отправляем пользователя  
 if(user){  
 res.send(user);  
 }  
 else{  
 res.status(404).send();  
 }  
});  
// получение отправленных данных  
app.post("/api/users", jsonParser, function (req, res) {  
  
 if(!req.body) return res.sendStatus(400);  
  
 const userName = req.body.name;  
 const userAge = req.body.age;  
 let user = {name: userName, age: userAge};  
  
 let data = fs.readFileSync(filePath, "utf8");  
 let users = JSON.parse(data);  
  
 // находим максимальный id  
 const id = Math.max.apply(Math,users.map(function(o){return o.id;}))  
 // увеличиваем его на единицу  
 user.id = id+1;  
 // добавляем пользователя в массив  
 users.push(user);  
 data = JSON.stringify(users);  
 // перезаписываем файл с новыми данными  
 fs.writeFileSync("users.json", data);  
 res.send(user);  
});  
// удаление пользователя по id  
app.delete("/api/users/:id", function(req, res){  
  
 const id = req.params.id;  
 let data = fs.readFileSync(filePath, "utf8");  
 let users = JSON.parse(data);  
 let index = -1;  
 // находим индекс пользователя в массиве  
 for(var i=0; i < users.length; i++){  
 if(users[i].id==id){  
 index=i;  
 break;  
 }  
 }  
 if(index > -1){  
 // удаляем пользователя из массива по индексу  
 const user = users.splice(index, 1)[0];  
 data = JSON.stringify(users);  
 fs.writeFileSync("users.json", data);  
 // отправляем удаленного пользователя  
 res.send(user);  
 }  
 else{  
 res.status(404).send();  
 }  
});  
// изменение пользователя  
app.put("/api/users", jsonParser, function(req, res){  
  
 if(!req.body) return res.sendStatus(400);  
  
 const userId = req.body.id;  
 const userName = req.body.name;  
 const userAge = req.body.age;  
  
 let data = fs.readFileSync(filePath, "utf8");  
 const users = JSON.parse(data);  
 let user;  
 for(var i=0; i<users.length; i++){  
 if(users[i].id==userId){  
 user = users[i];  
 break;  
 }  
 }  
 // изменяем данные у пользователя  
 if(user){  
 user.age = userAge;  
 user.name = userName;  
 data = JSON.stringify(users);  
 fs.writeFileSync("users.json", data);  
 res.send(user);  
 }  
 else{  
 res.status(404).send(user);  
 }  
});  
  
app.listen(3000, function(){  
 console.log("Сервер ожидает подключения...");  
});

index.html

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8" />  
 <meta name="viewport" content="width=device-width" />  
 <title>Список пользователей</title>  
 <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet" />  
</head>  
<body>  
<h2>Список пользователей</h2>  
<form name="userForm">  
 <input type="hidden" name="id" value="0" />  
 <div class="form-group">  
 <label for="name">Имя:</label>  
 <input class="form-control" name="name" />  
 </div>  
 <div class="form-group">  
 <label for="age">Возраст:</label>  
 <input class="form-control" name="age" />  
 </div>  
 <div class="panel-body">  
 <button type="submit" class="btn btn-sm btn-primary">Сохранить</button>  
 <a id="reset" class="btn btn-sm btn-primary">Сбросить</a>  
 </div>  
</form>  
<table class="table table-condensed table-striped table-bordered">  
 <thead><tr><th>Id</th><th>Имя</th><th>возраст</th><th></th></tr></thead>  
 <tbody>  
 </tbody>  
</table>  
  
<script>  
 // Получение всех пользователей  
 async function GetUsers() {  
 // отправляет запрос и получаем ответ  
 const response = await fetch("/api/users", {  
 method: "GET",  
 headers: { "Accept": "application/json" }  
 });  
 // если запрос прошел нормально  
 if (response.ok === true) {  
 // получаем данные  
 const users = await response.json();  
 let rows = ***document***.querySelector("tbody");  
 users.forEach(user => {  
 // добавляем полученные элементы в таблицу  
 rows.append(row(user));  
 });  
 }  
 }  
 // Получение одного пользователя  
 async function GetUser(id) {  
 const response = await fetch("/api/users/" + id, {  
 method: "GET",  
 headers: { "Accept": "application/json" }  
 });  
 if (response.ok === true) {  
 const user = await response.json();  
 const form = ***document***.forms["userForm"];  
 form.elements["id"].value = user.id;  
 form.elements["name"].value = user.name;  
 form.elements["age"].value = user.age;  
 }  
 }  
 // Добавление пользователя  
 async function CreateUser(userName, userAge) {  
  
 const response = await fetch("api/users", {  
 method: "POST",  
 headers: { "Accept": "application/json", "Content-Type": "application/json" },  
 body: ***JSON***.stringify({  
 name: userName,  
 age: parseInt(userAge, 10)  
 })  
 });  
 if (response.ok === true) {  
 const user = await response.json();  
 reset();  
 ***document***.querySelector("tbody").append(row(user));  
 }  
 }  
 // Изменение пользователя  
 async function EditUser(userId, userName, userAge) {  
 const response = await fetch("api/users", {  
 method: "PUT",  
 headers: { "Accept": "application/json", "Content-Type": "application/json" },  
 body: ***JSON***.stringify({  
 id: userId,  
 name: userName,  
 age: parseInt(userAge, 10)  
 })  
 });  
 if (response.ok === true) {  
 const user = await response.json();  
 reset();  
 ***document***.querySelector("tr[data-rowid='" + user.id + "']").replaceWith(row(user));  
 }  
 }  
 // Удаление пользователя  
 async function DeleteUser(id) {  
 const response = await fetch("/api/users/" + id, {  
 method: "DELETE",  
 headers: { "Accept": "application/json" }  
 });  
 if (response.ok === true) {  
 const user = await response.json();  
 ***document***.querySelector("tr[data-rowid='" + user.id + "']").remove();  
 }  
 }  
  
 // сброс формы  
 function reset() {  
 const form = ***document***.forms["userForm"];  
 form.reset();  
 form.elements["id"].value = 0;  
 }  
 // создание строки для таблицы  
 function row(user) {  
  
 const tr = ***document***.createElement("tr");  
 tr.setAttribute("data-rowid", user.id);  
  
 const idTd = ***document***.createElement("td");  
 idTd.append(user.id);  
 tr.append(idTd);  
  
 const nameTd = ***document***.createElement("td");  
 nameTd.append(user.name);  
 tr.append(nameTd);  
  
 const ageTd = ***document***.createElement("td");  
 ageTd.append(user.age);  
 tr.append(ageTd);  
  
 const linksTd = ***document***.createElement("td");  
  
 const editLink = ***document***.createElement("a");  
 editLink.setAttribute("data-id", user.id);  
 editLink.setAttribute("style", "cursor:pointer;padding:15px;");  
 editLink.append("Изменить");  
 editLink.addEventListener("click", e => {  
  
 e.preventDefault();  
 GetUser(user.id);  
 });  
 linksTd.append(editLink);  
  
 const removeLink = ***document***.createElement("a");  
 removeLink.setAttribute("data-id", user.id);  
 removeLink.setAttribute("style", "cursor:pointer;padding:15px;");  
 removeLink.append("Удалить");  
 removeLink.addEventListener("click", e => {  
  
 e.preventDefault();  
 DeleteUser(user.id);  
 });  
  
 linksTd.append(removeLink);  
 tr.appendChild(linksTd);  
  
 return tr;  
 }  
 // сброс значений формы  
 ***document***.getElementById("reset").click(function (e) {  
  
 e.preventDefault();  
 reset();  
 })  
  
 // отправка формы  
 ***document***.forms["userForm"].addEventListener("submit", e => {  
 e.preventDefault();  
 const form = ***document***.forms["userForm"];  
 const id = form.elements["id"].value;  
 const name = form.elements["name"].value;  
 const age = form.elements["age"].value;  
 if (id == 0)  
 CreateUser(name, age);  
 else  
 EditUser(id, name, age);  
 });  
  
 // загрузка пользователей  
 GetUsers();  
</script>  
</body>  
</html>

**Питання:**

1) У чому різниця між Node.js та JavaScript?

Node.js - це інтерпретатор, а також середовище для JavaScript, яке в основному використовується для доступу або реалізації будь-яких процедур, що не блокують, для будь-якого типу ОС. Тут працює двигун Google Chrome.

Зважаючи на те, що JavaScript - це мова програм, яка використовується для будь-яких завдань на стороні клієнта для інтернет-додатків. Тут працює двигун Firefox, Safari, Google Chrome і т.д.

2) Що таке Node.js?

Node.js - це легка, але ефективна структура, створена на nbsp; Двигун JavaScript у Chrome. Він поєднує JavaScript прямо в код місцевого виробника. Він використовується в структурі серверних інтернет-додатків, а також розширює JavaScript API для забезпечення типових серверних можливостей. Люди використовують цю структуру для створення великих програм, таких як односторінкові програми, веб-сайти з потоковою передачею відеокліпів, а також різні інші онлайн-програми.

3) Забезпечте деякі переваги використання Node.js

Це швидко, оскільки створюється на движку Chrome JavaScript, що прискорює його збирання при реалізації коду.

Він є асинхронним, оскільки ніколи не очікує, що API поверне інформацію.

Він масштабується завдяки системі подій, яка дозволяє веб-серверу реагувати неблокуючим методом.

Це відкритий вихідний код, що забезпечує надзвичайне оточення з відкритим вихідним кодом, в результаті якого були створені деякі чудові компоненти, які додали ще більше можливостей для Node.js.

Node.js ніколи не блокує інформацію, тому що результат виходить частинами.

4) Які відмінності між Angular та Node.js?

Angular - це відкритий вихідний код, що використовується для інтернет-додатків, а &&Node.js, як правило, є кросплатформовим середовищем виконання для додатків.

Angular складений на TypeScript & & & Node.js складений такими мовами, як C, C++, а також JavaScript.

Angular в основному використовується для розробки односторінкових клієнтських інтернет-програм, а Node.js в основному використовується для розробки масштабованих, а також швидких серверних мережних додатків.

Angular - це структура інтернет-додатку, & Node.js має безліч nbsp; різні конструкції & NBSP; наприклад, Partial.js, Sails.js, а також Express.js і т.д.

Angular найкраще використовувати для розробки дуже енергійних, а також інтерактивних інтернет-застосунків, а & Node.js найкраще використовувати для створення & nbsp; завдання меншого розміру

Angular допомагає розділити програму на частини MVC, а & Node.js допомагає в & nbsp; розробка запитів до джерел даних &nbsp;

Angular добре використовується для розробки програм у реальному часі. & & Node.js добре використовується в тих випадках, коли він працює швидше, а також є більш масштабованим потрібно

5) Чому Node.js однопотоковий?

Він обслуговує однопоточну версію, а не звичайне виконання на основі потоків, щоб допомогти асинхронній обробці. Використовуючи асинхронну обробку, програма може бути реалізована набагато краще, а також набагато більш масштабується в тоннах Інтернету.

6) Як працює Node.js?

Node.js - це онлайн розробник, який працює в атмосфері v8, використовуючи JavaScript як мову сценаріїв. Він обслуговує однопоточну версію, а також неблокуючий введення-висновок, що забезпечує високу ціну, оскільки може керувати великою різноманітністю синхронізованих вимог. Оскільки він використовує HTTP-компонент, він може працювати на будь-якому автономному інтернет-сервері.

7) Де можна використовувати Node.js?

Node.js можна використовувати для розробки:

Мережеві програми

Інтернет-програми в реальному часі

Загальні об'єктивні програми

Дисперсне обладнання

8) У Node.js є безліч різних функцій API?

У Node.js можна знайти 2 види функцій API, зокрема паралельні, що перешкоджають функції, а також асинхронні, неблокуючі функції.

9) У чому різниця між асинхронною та неблокуючою функцією?

Асинхронні (не паралельні) функції - це ті, за допомогою яких ми можемо створювати асинхронні HTTP-запити, які не чекають на реакцію веб-сервера. Ці функції залишаються, щоб повернутися до запиту, на який він фактично отримав реакцію веб-сервера.

Неблокуючі функції використовуються щодо процедур введення-виводу. Вони швидко реагують на будь-яку пропоновану інформацію, а також підтримують роботу відповідно до вимог. Якщо рішення не повернуто, API негайно повертається з помилкою.

Додатково ознайомтесь: 20 найкращих запитань та відповідей з інтерв'ю Kubernetes, які вам потрібно знати у 2021 році

10) Поясніть шоу, орієнтовані на події

Це програмна стратегія, яка переважно використовує можливості для активації безлічі функцій. У деяких випадках це може бути натисканням клавіші

11) У контексті Node.js обговоріть REPL

REPL у Node.js означає ;Читати, оцінювати, публікувати, а також Лазівка. Він ілюструє створення комп'ютерної системи, такої як оболонка Unix/Linux чи домашня віконна консоль, або де можна підписати будь-який вид команди, і після цього система може відповісти з результатом. Node.js за замовчуванням завантажено з атмосферою REPL. REPL може реалізувати виконання завдань: Читання: він перевіряє дані, що вводяться користувачем, перетворює їх прямо в структуру даних JavaScript і після цього зберігає їх у пам'яті. Оцінка: він отримує та досліджує інформаційну структуру. Опублікувати: публікує найкращі результати. Лазейка: лазівка ​​у запропонованій команді доти, доки nbsp; CTRL+C натискається двічі.

12) Запишіть роботу, яка має виконуватися асинхронно, використовуючи лазівку для випадку?

Процедури введення/виведення Здоров'яний розрахунок Все, що потрібно зупинити

13) Чому Google використовує двигун V8 для Node.js?

Механізм виконання V8 перетворює код JavaScript прямо на код оригінального виробника, що призводить до швидкодії додатків. Цей двигун Chrome значно прискорює використання, а також процедури реагування.

14) Які переваги використання Node.js?

Як ми вже говорили раніше, Node.js є покращеним двигуном V8, завдяки чому його колекція швидко продає код. Сервер на основі Node.js не очікує, що API поверне інформацію, що вказує на асинхронність. Крім того, Node.js дуже масштабується, оскільки його система подій реагує неблокуючим чином. Крім того, у програмах Node.js немає буферизації. Протягом багатьох років фактично виникала велика спільнота відкритих ресурсів, яка додавала чудові компоненти, а також пропонує додаткові можливості для Node.js.

15) Що таке екзаменаційна піраміда?

Схема, що описує співвідношення кількості перевірок пристроїв, асиміляційних перевірок, а також наскрізного обстеження, яке необхідно скласти для ефективної структури завдання, називається пірамідою іспитів.

16) Які функції Express.js?

Express.js - це структура, яка допомагає відстежувати обмін інформацією між веб-сервером, а також курси серверних додатків. Ця структура, що адаптується, є покращеною версією Node.js, особливо в компоненті проміжного програмного забезпечення, званому attach. Отже компонент приєднання використовує компонент HTTP для взаємодії з Node.js. З цієї причини включення Express.js є дуже простим, якщо ви співпрацюєте з будь-яким компонентом проміжного програмного забезпечення на основі приєднання.

17) Яка функція компонента insist у Node.js?

Assert дозволяє вам складати іспити. У Node.js це внутрішній компонент, який надає набір перевірок тверджень для перевірки інваріантів. Ви можете використовувати код необхідності (наполягати) для різних інших програм.

18) Що таке зворотний виклик при першій помилці Node.js?

Зворотний виклик error-first у Node.js використовується для передачі помилок та інформації. Найперша специфікація, яка передає ці функції, має бути помилкою. Інші різні критерії – це прикріплена інформація. Якщо немає помилок або побоювань, ви можете продовжити обговорення.

var myPost = нова стаття ({title: 'Myexample'});

myPost.save (feature (err, myInstance) {

якщо (помилка)

{

// Виправляємо помилку, а також повертаємо

}

// продовжуємо з 'myInstance'

});

19) Яка мета документів package.json?

Документи package.json у Node.js включають метадані завдання. Це серце програми, де ви вказуєте вдома плану. Метадані документів можуть бути додатково класифіковані на такі категорії:

Резиденції, такі як ім'я завдання, сертифікат, письменник, варіант поточного компонента, зведення завдання тощо.

Ви також можете додати важливі деталі, що стосуються вашого завдання, створивши їх у документах package.json.

20) Що використовує module.exports?

Щоб змішати всі схожі коди прямо в єдиний пристрій коду, який може бути виконано шляхом переміщення всіх пов'язаних функцій прямо в певні документи, нам потрібно використовувати module.exports в Node.js. Наприклад, у вас є документ з ім'ям greet.js, який включає обидві функції, як показано нижче:

модуль.експорт = {

greetInHindi: feature() {

повернути «НАМАСТІ»;

},

greetInEnglish: feature() {

поверніть "Привіт";

}};

Модуль .exports & nbsp; надає 2 функції, які можна імпортувати в різні документи, використовуючи наведений нижче код:

var myGreets = need («./greet.js»);

myGreets.greetInHindi() // НАМАСТІ

myGreets.greetInEnglish() // Привіт

21) Що таке шаблон активатора в Node.js?

Шаблон активатора &nbsp; Node.js - це концепція неблокуючих процедур введення-виводу. Цей шаблон надає засіб навчання, який зв'язується з кожною процедурою введення-виводу, а також, як тільки створюється запит введення-виводу, він після цього відправляється в &nbsp; демультиплексор Цей демультиплексор є користувальницьким інтерфейсом попереджень, який може керувати паралелізмом у неблокуючому налаштуванні введення-виводу.

Це додатково допомагає впоратися з кожною вимогою у типі події та після цього помістити кожну подію у лінію. Відтак це призводить до створення Лінії подій. У той же час, у нас є лазівка ​​з нагоди, яка дублює випадки, що існують у Лінії подій.

Додатково перевірити: JavaScript проти. PHP: різниця між JavaScript та PHP

22) Що таке LTS-запуск Node.js?

LTS & nbsp; способи nbsp; Довгі & nbsp; Термін & nbsp; Варіант підтримки Node.js, у якому усунуті всі важливі проблеми з комахами з оновленнями безпеки, а також підвищена ефективність. Всі ці варіанти розраховані як мінімум на 18 місяців і зосереджені в основному на безпеці, а також на проблемах безпеки. Модифікації, внесені у варіації LTS, обмежуються лише боротьбою з комахами, npm, підвищення безпеки, документацією, а також підвищення ефективності.

23) Що ви дізнаєтеся за зворотним викликом?

Callback Heck ще називають пірамідою руйнування. Це шаблон, який виникає через серйозні вбудовані зворотні виклики, які не читаються. Зазвичай він включає кілька вбудованих функцій зворотного дзвінка, що ускладнюють перегляд або налагодження коду. Це в основному через неправильне виконання асинхронних міркувань.

async\_A (feature() {

async\_B (feature() {

async\_C (feature() {

async\_D (feature() {

});

});

});

});

24) Надайте вхідні дискусії async.queue.

У Node.js async.queue приймає як вхідні дані 2 обговорення. Вони складаються з функції завдання, і навіть цінності паралелізму.

25) Що таке libuv?

Це багатоплатформна допоміжна колекція яка в основному використовується для асинхронного введення-виводу. Спочатку він був створений для Node.js, сьогодні він додатково використовується в нашій системі, такий як pyuv, Luvit, Julia і т.д. . Деякі з важливих функцій libuv:

Випадки системи документів

Повнофункціональна лазівка ​​для заходів

Асинхронні виходи TCP та UDP

Асинхронні документи, а також системні процедури документування

Дитячі процедури

26) Потоки в Node.js

Потоки дозволяють переглядати та уточнювати величезні набори даних у Node.js. Ці елементи є інформаційними колекціями, які виглядають як рядки, а також діапазони. Вони дозволяють постійно переглядати інформацію з ресурсу або зв'язуватися з місцем. У Node.js існує 4 основних типи потоків: розбірливий, доступний для запису, дуплексний, а також перетворений.

27) Як саме ви переглядаєте та впроваджуєте документи в Node.js?

У Node.js є 2 методи для перегляду та реалізації документів - readFile та createReadStream. Основна різниця між ними полягає в тому, що readFile повністю буферизується, а createReadStream - частково. readFile() повертає зворотний зв'язок, коли всі документи вдавлюються прямо в бар'єр, зменшуючи час обробки.

З іншого боку, createReadStream розглядає процедуру як випадкову колекцію, розділяючи документи прямо на частини, а також повертаючи окремо дії. Таким чином, це дає надійний спосіб роботи з величезними інформаційними документами.

28) Які безпечні виконання пропонуються у Node.js?

Перевірка та обробка помилок – важливі заходи безпеки в Node.js.

29) Як дитячі рядки управляються в Node.js?

Ви, напевно, припустите, що, враховуючи, що Node.js є однопоточним, ви не можете використовувати дочірні рядки. Хоча процедура зазвичай не виявляє дочірні рядки, ви все ж таки можете реалізувати певні завдання асинхронного введення-виведення в історії за допомогою методу generate (). Тим самим не виключається лазівка ​​у центрі програми. Ви можете додатково складатися з ChildProcess, щоб чітко використовувати цю ідею багатопоточності.

30) Що використовує NODE\_ENV?

Якщо ваше завдання стосується ступеня виробництва, Node.js рекламує угоду про використання змінної NODE\_ENV для її позначення. Це допомагає зробити найкращий вибір під час виконання завдання. Більш того, коли ви вводите NODE\_ENV у виробництво, ваша програма працює швидше приблизно в 3 рази.