Міністерство освіти і науки України

Черкаський державний технологічний університет

Звіт

про виконання лабораторної роботи №1 з навчальної дисципліни

«Програмування WEB додатків»

Виконав: студент   
 Групи ПЗС-2044

Оратівський С.І.

Перевірив:

Метелап В.В.

Черкаси

2022

**Тема: Jquery**

**Варіант: Магазин книжковий**

1. Зверстати сторінку з використанням Jquery. Далі необхідно передбачити CRUD з даними, які будуть заноситись в БД.

Посилання на коміт: [SerhiiOrSoratik/internet-shop-Web-Application-Programming at f352539f9d5503d90f781f0711acae43f461a6ed (github.com)](https://github.com/SerhiiOrSoratik/internet-shop-Web-Application-Programming/tree/f352539f9d5503d90f781f0711acae43f461a6ed)

Frontend/Index.js

$.ajax({  
 headers: {"Accept": "application/json"},  
 type: 'GET',  
 url: "http://localhost:3000/product",  
 crossDomain: true,  
 success: function (data) {  
 data.forEach(el => {  
 $('#catalog').append(`  
 <div style="border: 1px solid black; width: 20%; min-width: 300px; height: auto; margin-bottom: 25px">  
 <p style="color: gray; font-size: 12px">${el.availability ? ' ' : 'Немає в наявності'}</p>  
 <p>${el.caption}</p>  
 <button onclick="removeById(${el.id})" style="transform: translate(150px, -50px); position: absolute; height: 25px; width: 25px">x</button>  
 <button onclick="editBook(${el.id})" style="transform: translate(120px, -50px); position: absolute; height: 25px; width: 25px">✎</button>  
 <p>${el.price} грн</p>  
 <p style="word-wrap: break-word; color: gray">${el.description}</p>  
 </div>  
 `)  
 })  
 }  
});  
  
$(***document***).ready(function (e) {  
 $("#createForm").on('submit', (function (e) {  
 e.preventDefault();  
 $.ajax({  
 headers: {  
 'Accept': 'application/json',  
 },  
 url: "http://localhost:3000/product",  
 dataType: 'json',  
 type: "POST",  
 data: $('#crForm').serialize(),  
 success: function (data) {  
 ***location***.reload();  
 },  
 });  
 }));  
  
 $("#updateForm").on('submit', (function (e) {  
 e.preventDefault();  
 $.ajax({  
 headers: {  
 'Accept': 'application/json',  
 },  
 url: "http://localhost:3000/product/" + $('#upd\_id').attr("value"),  
 dataType: 'json',  
 type: "PUT",  
 data: $('#updtForm').serialize(),  
 success: function () {  
 ***location***.reload();  
 },  
 });  
 }));  
});  
  
const removeById = (id) => {  
 $.ajax({  
 url: "http://localhost:3000/product/" + id,  
 type: 'DELETE',  
 success: function () {  
 ***location***.reload();  
 }  
 });  
}  
  
function openForm() {  
 $("#createForm").attr("style", "display:block");  
}  
  
function hideForm(formName) {  
 const form = $("#createForm");  
 form.css("style", "display:none");  
 form.empty();  
}  
  
function hideUpdateForm() {  
 const form = $("#updateForm");  
 form.css("style", "display:none");  
 form.empty();  
}  
  
const editBook = (id) => {  
 const form = $("#updateForm");  
 form.append(`  
 <form id="updtForm" method="put" style="width: 320px; padding: 10px; background-color: #999999; display: flex; flex-direction: column; gap: 25px;">  
 <h3>Edit book</h3>  
 <input id="upd\_id" name="id" required value="${id}" style="display: none" readonly/>  
 <input id="upd\_caption" name="caption" placeholder="Enter caption" required"/>  
 <input id="upd\_price" type="number" name="price" placeholder="Enter price" required"/>  
 <input id="upd\_description" name="description" placeholder="Enter description""/>  
 <div style="display: flex; justify-content: space-between">  
 <input style=" background: #656565; background: -webkit-gradient(linear, 0 0, 0 bottom, from(#656565), to(#444)); background: -moz-linear-gradient(#656565, #444); background: linear-gradient(#656565, #444);  
 padding: 16px 20px; border: none; cursor: pointer; width: 45%; opacity: 0.8;" type="submit" value="Edit">  
 <button style=" background: #656565; background: -webkit-gradient(linear, 0 0, 0 bottom, from(#656565), to(#444)); background: -moz-linear-gradient(#656565, #444); background: linear-gradient(#656565, #444);  
 padding: 16px 20px; border: none; cursor: pointer; width: 45%; opacity: 0.8;" onclick="{hideUpdateForm()}">Cancel</button>  
 </div>  
 </form>`  
 )  
 form.attr("style", "display:block");  
}

Frontend/Index.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Books</title>  
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>  
 <script src="index.js"></script>  
 <link rel="stylesheet" href="styles.css">  
</head>  
<body>  
<div id="header">  
 <h1>Book shop</h1>  
 <button id="addBook" onclick="openForm()">Add book</button>  
</div>  
  
<div class="form-popup" id="createForm">  
 <form id="crForm" method="post" enctype="application/json" class="form-container">  
 <h3>Add book</h3>  
 <div class="form-group">  
 <label for="cr\_caption">Caption</label>  
 <input id="cr\_caption" name="caption" placeholder="Enter caption" required/>  
 </div>  
 <div class="form-group">  
 <label for="cr\_price">Price</label>  
 <input id="cr\_price" type="number" name="price" placeholder="Enter price" required/>  
 </div>  
 <div class="form-group">  
 <label for="cr\_description">Description</label>  
 <input id="cr\_description" name="description" placeholder="Enter description"/>  
 </div>  
 <div style="display: flex; justify-content: space-between">  
 <input class="btn" type="submit" value="Upload">  
 <button class="btn" value="Cancel" onclick="hideForm()">Cancel</button>  
 </div>  
 </form>  
</div>  
  
<div class="form-popup" id="updateForm">  
</div>  
  
<div id="container">  
 <div id="catalog"></div>  
</div>  
</body>  
  
</html>

Backend/app.js

const express = require('express');  
const ***app*** = express();  
const ***router*** = require('./routes')  
  
const cors = require('cors');  
  
***app***.use(cors())  
***app***.use(express.***urlencoded***({ extended: true }));  
***app***.use(***router***)  
  
module.exports = ***app***;

Backend/server.js

const ***app*** = require('./app');  
const ***sequalize*** = require('./sequelize');  
const ***modelDb*** = require('./modelsDb')  
  
const start = async () => {  
 try {  
 await ***sequalize***.authenticate();  
 await ***sequalize***.sync();  
 ***app***.listen(3000);  
 } catch (e) {  
 ***console***.log(e);  
 }  
}  
  
start();

Backend/sequalize.js

const {Sequelize} = require('sequelize');  
  
module.exports = new Sequelize('internet\_shop', 'postgres', 'root', {  
 host: 'localhost',  
 dialect: 'postgres'  
});

Backend/models.js

const ***sequelize*** = require('./sequelize');  
const {DataTypes} = require('sequelize');  
  
const product = sequelize.define('products', {  
 id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},  
 caption: {type: DataTypes.STRING, allowNull: false},  
 availability: {type: DataTypes.BOOLEAN, allowNull: false},  
 price: {type: DataTypes.INTEGER, allowNull: false},  
 description: {type: DataTypes.STRING, allowNull: true},  
});  
  
module.exports = {  
 product  
}

Backend/routes/productsRoute

const Router = require('express');  
const ***productsRoute*** = require('../controllers/productsController');  
const ***router*** = new Router();  
  
  
***router***.get('/', (req, res) => {  
 ***productsRoute***.getProducts(req, res);  
})  
  
***router***.post('/', (req, res) => {  
 ***productsRoute***.createProduct(req, res);  
});  
  
***router***.put('/:id', (req, res) => {  
 ***productsRoute***.updateProduct(req, res);  
})  
  
***router***.delete('/:id', (req, res) => {  
 ***productsRoute***.deleteProduct(req, res);  
})  
  
module.exports = ***router***;

Backend/controllers/productsController

const ***productModel*** = require('../models/productsService');  
  
class Products {  
  
 async getProducts(req, res) {  
 try {  
 const products = await ***productModel***.getProducts();  
 res.status(200);  
 res.json(products);  
 } catch {  
 ***console***.log('Bad request')  
 res.status(400);  
 res.end('Bad request');  
 }  
 }  
  
 async createProduct(req, res) {  
 try {  
 const newProduct = await ***productModel***.createProduct(req, res);  
 res.status(201);  
 res.json(newProduct);  
 } catch {  
 ***console***.log('Bad request')  
 res.status(400);  
 res.end('Bad request');  
 }  
 }  
  
 async updateProduct(req, res) {  
 try {  
 res.json(await ***productModel***.updateProduct(req));  
 res.status(200);  
 } catch {  
 ***console***.log('Bad request')  
 res.status(400);  
 res.end('Bad request');  
 }  
 }  
  
 async deleteProduct(req, res) {  
 try {  
 const id = req.params.id;  
 res.status(***productModel***.deleteProduct(id));  
 res.end();  
 } catch {  
 ***console***.log('Bad request')  
 res.status(400);  
 res.end('Bad request');  
 }  
  
 }  
  
}  
  
module.exports = new Products();

Backend/models/productsService

const ***model*** = require('../modelsDb');  
  
class productsModel {  
  
 getProducts() {  
 return ***model***.product.*findAll*();  
 }  
  
 async createProduct(req) {  
 return await ***model***.product.*create*({  
 caption: req.body.caption,  
 availability: true,  
 price: req.body.price,  
 description: req.body.description,  
 });  
 }  
  
 async updateProduct(req) {  
 return await ***model***.product.*update*(  
 req.body,  
 {where: {id: req.params.id}}  
 )  
 }  
  
 deleteProduct(id) {  
 ***model***.product.*destroy*({  
 where: {  
 id: id  
 }  
 });  
 return 201;  
 }  
}  
  
module.exports = new productsModel()

1. Переробити завдання з попередньої лабораторної (створити нову версію) з урахуванням нового матеріалу (з використанням Promise). Також передбачити вигрузку та збереження інформації з бази в різних форматах (json, xml, csv).

Посилання на коміт: [SerhiiOrSoratik/internet-shop-Web-Application-Programming at ca3b561f989c69a4180ea92970d00be92519f8a0 (github.com)](https://github.com/SerhiiOrSoratik/internet-shop-Web-Application-Programming/tree/ca3b561f989c69a4180ea92970d00be92519f8a0)

Frontend/index.js

const getById = id => {  
 const getByIdPromise = new ***Promise***(resolve => {  
 $.ajax({  
 url: "http://localhost:3000/product/" + id,  
 type: 'get',  
 success: data => {  
 resolve(data);  
 },  
 });  
 })  
 return getByIdPromise.then(data => {  
 return data;  
 })  
}

const downloadScvFile = async id => {  
 const data = await getById(id);  
 const csv = convertToCSV(data);  
 let element = ***document***.createElement('a');  
 element.setAttribute('href', 'data:application/json;charset=utf-8,' + encodeURIComponent(csv));  
 element.setAttribute('download', `${data.caption}.csv`);  
 element.click();  
}  
  
const convertToCSV = data => {  
 const keys = ***Object***.keys(data);  
 const values = ***Object***.values(data);  
 let result = keys.join(',');  
 result += '\n';  
 result += values.join(',')  
 return result;  
}  
  
const downloadJsonFile = async id => {  
 const data = await getById(id);  
 const json = ***JSON***.stringify(data);  
 let element = ***document***.createElement('a');  
 element.setAttribute('href', 'data:application/json;charset=utf-8,' + encodeURIComponent(json));  
 element.setAttribute('download', `${data.caption}.json`);  
 element.click();  
}

Також на бекенді було додано новий ендпоінт

Backend/models/productsService

async getProductById(id) {  
 return await model.product.*findOne*({  
 where: {  
 id: id  
 }  
 });  
}

1. Імплементувати пошук

Посилання на коміт: [SerhiiOrSoratik/internet-shop-Web-Application-Programming at e6343359118ac0db5c3a3197badbe6b2e6af6b80 (github.com)](https://github.com/SerhiiOrSoratik/internet-shop-Web-Application-Programming/tree/e6343359118ac0db5c3a3197badbe6b2e6af6b80)

Frontend/index.html

<div style="width: 50%">  
 <input style="width: 15%; height: 88%" id="search"/>  
 <button class="form\_button" style="width: 15%" id="searchButton" onclick="search()">Search</button>  
</div>

Frontend/index.js

const search = () => {  
 const searchValue = $("#search")[0].value;  
  
 const search = new ***Promise***(resolve => {  
 if (searchValue) {  
 $.ajax({  
 url: "http://localhost:3000/product/search/" + searchValue,  
 type: 'get',  
 success: data => {  
 ***console***.log(data)  
 resolve(data);  
 },  
 });  
 } else {  
 $.ajax({  
 headers: {"Accept": "application/json"},  
 type: 'GET',  
 url: "http://localhost:3000/product",  
 crossDomain: true,  
 success: data => {  
 resolve(data);  
 }  
 })  
 }  
 })  
 return search.then(data => {  
 render(data);  
 })  
}

Backend/routes/productsRoute

router.get('/search/:searchValue', (req, res) => {  
 productsRoute.search(req, res);  
})

Backend/Controllers/productsController

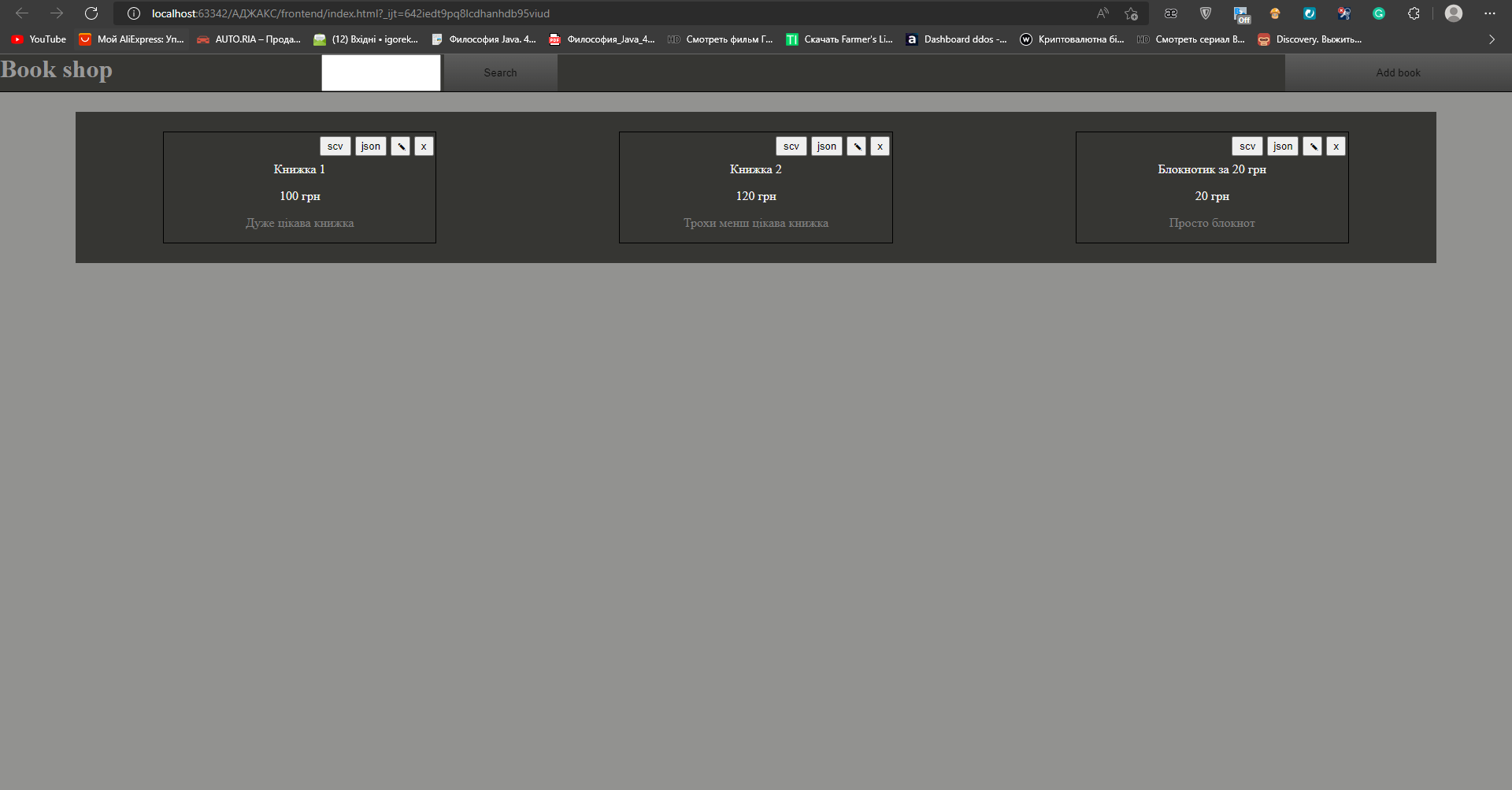
async search(req, res) {  
 try {  
 const products = await productModel.search(req.params.searchValue);  
 res.status(200);  
 res.json(products);  
 } catch (e) {  
 ***console***.log(e);  
 res.status(400);  
 res.end('Bad request');  
 }  
}

Backend/models/productsService

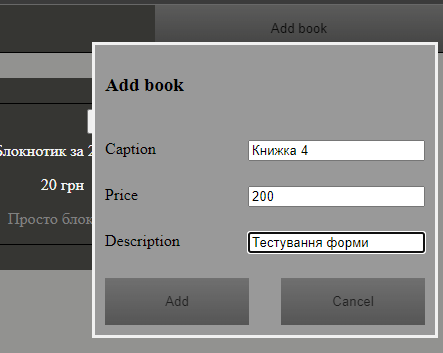
async search(searchValue) {  
 return await model.product.*findAll*({  
 where: {  
 caption: {  
 [Op.like]: searchValue + '%'  
 }  
 }  
 });  
}

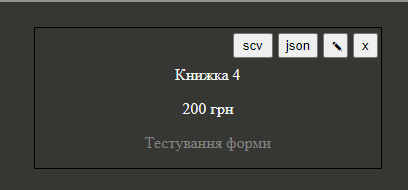
Огляд результатів

Сторінка проекту

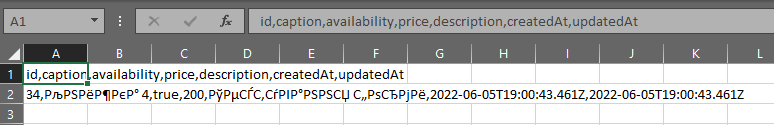


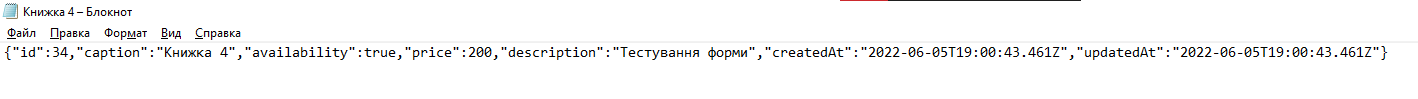
Додавання нової книжки



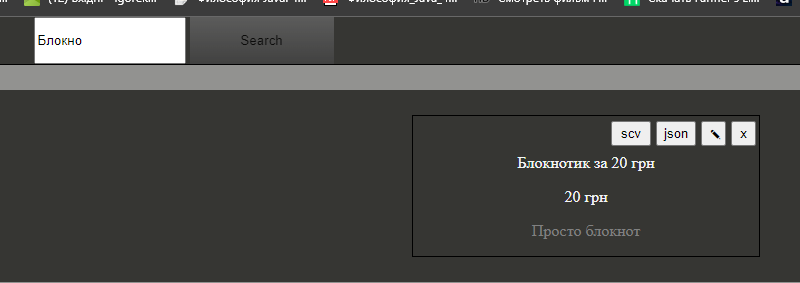


Файли скачані після натискань на кнопки csv та json





Перевірка роботи пошуку



Питання:

1. Що таке JQuery?

jQuery - це бібліотека JavaScript для маніпулювання об’єктною моделлю документа, яка в основному використовується для управління та подорожі по документах HTML, обробки подій веб-елементів, спеціальних ефектів на комунікації DOM та Ajax та розробки декількох браузерів JavaScript.

1. Головна перевага JQuery?

Зменшення кількості кода

1. Основна відмінність JS від JQuery?

Js – це мова програмування, а JQuery це бібліотека

1. JQuery використовується на фронтенді чи на бекенді?

На фронтенді

1. Як почати роботу з JQuery?

Підключити за допомогою посилання або скачаного файлу

1. Що означає знак долара в JQuery?

Це селектор який допомагає вибрати потрібний нам тег

1. Чи можна використовувати кілька бібліотек разом з JQuery?

Так

1. Яка різниця між JQuery та JQuery.min?

JQuery.min спрощена версія де видалені деякі можливості

1. Що таке селектор в JQuery?

Селектор це річ, яка допомагає шукати елементи на сторінці

1. Які селектори в JQuery найшвидші?

Найшвидший селектор по ID

1. Чи швидший JQuery за js?

Ні тому, що Jquery використовує функції js під капотом

1. Поясніть функціональність методу Ajax ().

Метод Ajax () надсилає асинхронні HTTP-запити на сервер і отримує дані від сервера. Метод Ajax () цінний, оскільки пропонує можливість вказувати як зворотні виклики успіху, так і невдачі.

1. Поясніть функціональність методу ajax load ().

Метод load () надсилає запити HTTP для завантаження даних у вигляді HTML або текстового вмісту з сервера і поміщає повернені дані у вибрані елементи DOM.

1. Розробка подій jQuery Ajax

Методи Ajax запускають обробник подій, що призводить до подій jQuery Ajax

1. Поясніть функціональність методу ajaxComplete ().

AjaxComplete () отримує виклик незалежно від того, чи був запит успішним або призвів до помилки, і отримується повний зворотний виклик, навіть для синхронних запитів.

1. Поясніть функціональність методу ajaxStart ().

Подія ajaxStart () - глобальна подія, яка запускається в результаті запуску запиту Ajax, за умови, що в даний час не виконуються інші запити Ajax.

1. Диференціюйте методи find () та дочірні ().

.find () та .children () використовуються для пошуку дочірнього елемента відповідних елементів DOM, .find () переходить на будь-який рівень вниз, тоді як .children () рухається на один рівень вниз, щоб знайти елемент

1. Поясніть переваги методів jQuery Ajax.

Ajax може запитувати та отримувати дані із сервера без перезавантаження сторінки за допомогою DOM та JavaScript.

1. Розрізняти методи завантаження () та document.ready () jQuery.

Різниця між методами onload () та document.ready () полягає в тому, що метод onload () JavaScript буде викликаний лише після того, як усі об'єкти у веб-документі будуть повністю відображені / завантажені.

1. Поясніть функціональність методу connect () jQuery.

jQuery connect () - це плагін, який використовується для підключення / прив'язки функції до іншої функції шляхом призначення обробника. За допомогою цієї функції ми можемо використовувати подію елемента DOM.

1. Поясніть можливість використання методу jQuery HTML для документів HTML та XML.

Метод jQuery HTML недоступний у документах XML, він працює лише для документів HTML.

1. Поясніть функціональність плагіна Data Table для jQuery.

Таблиця даних - це плагін jQuery, який при застосуванні до записів відображається у вигляді таблиці.

Ми можемо сортувати дані як за окремими, так і за кількома стовпцями, шукати конкретний запис, додавати пагінацію, записи на сторінку та переміщатися між ними в таблиці. Таблицю даних можна застосувати до статичних даних, масиву, даних у форматі JSON, а також відповіді AJAX.

1. Визначте різницю між .detach () та .remove () jQuery.

Метод detach () jQuery видаляє вибраний елемент, проте зберігає дані та події. Метод .remove () jQuery видаляє елементи, дані, а також події.

1. Поясніть функціональність фільтра jQuery.

.filter () у jQuery перевірить наявність відповідного елемента, і атрибут може бути доданий до відповідного елементу.

1. Поясніть функціональність методу .each () у jQuery.

jQuery .each () використовується для ітерації / переліку елемента DOM, який присутній в об’єкті jQuery.

1. Поясніть функціональність jQuery.noConflict.

Метод noConflict () jQuery застосовується для вирішення конфліктів, коли нам потрібно використовувати фреймворки, крім jQuery.

1. Розрізняти $ (this) та this у jQuery.

$ (this) - це об’єкт jQuery, тоді як це загальне посилання на об’єкт JavaScript, використовуючи це, ми можемо посилатись на елемент DOM у документі HTML.

$ (this) посилається на батьківський об'єкт, тоді як це стосується елемента DOM, який у випадку масиву представляє об'єкт із методом .each (), який відображає поточну ітерацію

1. Яке використання методу serialize () у jQuery?

Він серіалізує значення форми, щоб його серіалізовані значення могли бути використані в рядку запиту URL-адреси під час надсилання запиту AJAX.

.serialize () метод jQuery повертає введені значення HTML-форми у вигляді рядка.

1. Яке використання методу val () у jQuery?

Метод .val () допомагає знайти значення атрибута елемента HTML

1. Як jQuery зберігає дані, що стосуються елемента?

Метод jQuery.data () допомагає приєднувати будь-який тип даних до елементів DOM, вільний від витоків пам'яті. jQuery гарантує, що дані видаляються разом з елементами DOM, видаленими методами jQuery.