To fetch the data from the database, we can modify the JavaScript code to send an AJAX request to the server. Assuming the server returns the JSON data for the selected states, here's the updated JavaScript code to fetch the data and display it:

html

<!DOCTYPE html>

<html>

<head>

<title>State Dropdown</title>

<link rel="stylesheet" type="text/css" href="//fonts.googleapis.com/css?family=Open+Sans" />

<style>

/\* CSS styles here (same as in your original HTML) \*/

</style>

</head>

<body>

<h1>State Dropdown</h1>

<div class="dropdown">

<select class="stateDropdown">

<option value="" disabled selected>Select a state</option>

<option value="AL">Alabama</option>

<!-- Add all 50 states as options here -->

<option value="WY">Wyoming</option>

</select>

</div>

<div id="checkboxesContainer">

<div class="checkbox-container">

<!-- Checkbox labels here (same as in your original HTML) -->

</div>

</div>

<button class="add-on-btn" onclick="addOn()">Add On</button>

<script>

function handleCheckbox(checkbox) {

// Function code here (same as in your original JavaScript)

}

function addOn() {

const currentStateDropdown = document.querySelector('.stateDropdown');

const currentState = currentStateDropdown.value;

if (currentState) {

// Create the AJAX request

const xhr = new XMLHttpRequest();

xhr.onreadystatechange = function() {

if (xhr.readyState === XMLHttpRequest.DONE) {

if (xhr.status === 200) {

const data = JSON.parse(xhr.responseText);

displayData(data); // Function to display fetched data

} else {

console.error('Error fetching data from the server.');

}

}

};

// Replace 'server\_url' with the actual URL of your server endpoint that handles the database query

xhr.open('GET', 'server\_url?state=' + currentState, true);

xhr.send();

}

}

function displayData(data) {

// Clear the existing containers

const checkboxesContainer = document.getElementById('checkboxesContainer');

while (checkboxesContainer.firstChild) {

checkboxesContainer.removeChild(checkboxesContainer.firstChild);

}

// Create and populate new containers with fetched data

for (const stateData of data) {

const newContainer = document.createElement('div');

newContainer.className = 'container';

const newDropdown = currentStateDropdown.cloneNode(true);

newDropdown.value = stateData.stateName;

newContainer.appendChild(newDropdown);

const newCheckboxContainer = document.createElement('div');

newCheckboxContainer.className = 'checkbox-container';

const checkboxLabels = ['Grid Exist', 'New Grid', 'Approve Grid'];

for (const label of checkboxLabels) {

const checkboxLabel = document.createElement('label');

checkboxLabel.className = 'checkbox';

checkboxLabel.innerHTML = `

<input type="checkbox" name="gridOption" value="${label.replace(' ', '').toLowerCase()}">

<span class="checkmark"></span> ${label}

`;

newCheckboxContainer.appendChild(checkboxLabel);

}

for (const checkbox of newCheckboxContainer.querySelectorAll('input[type="checkbox"]')) {

checkbox.addEventListener("click", () => handleCheckbox(checkbox));

}

newContainer.appendChild(newCheckboxContainer);

checkboxesContainer.appendChild(newContainer);

}

}

</script>

</body>

</html>

In this code, the addOn() function sends an AJAX request to the server with the selected state as a query parameter. The server should handle the query, retrieve the relevant data from the database, and respond with a JSON array of objects, each containing a "stateName" field.

The displayData(data) function receives the JSON data from the server, clears the existing containers, and then creates and populates new containers with the fetched data. Each container contains a dropdown with the state name and the same set of checkboxes as in your original code.

Remember to replace 'server\_url' in the AJAX request with the actual URL of your server endpoint that handles the database query and returns the JSON data. Also, ensure that the server-side code is set up to handle the request, query the database, and respond with the appropriate JSON data.

make a json file and put mock data in it

ChatGPT

Sure! Here's a JSON file with mock data:

json

{

"states": [

{

"stateName": "Alabama",

"data": {

"Grid Exist": true,

"New Grid": false,

"Approve Grid": true

}

},

{

"stateName": "Alaska",

"data": {

"Grid Exist": false,

"New Grid": true,

"Approve Grid": false

}

},

{

"stateName": "Arizona",

"data": {

"Grid Exist": true,

"New Grid": true,

"Approve Grid": true

}

},

{

"stateName": "Arkansas",

"data": {

"Grid Exist": false,

"New Grid": false,

"Approve Grid": false

}

},

{

"stateName": "California",

"data": {

"Grid Exist": true,

"New Grid": false,

"Approve Grid": true

}

},

{

"stateName": "Colorado",

"data": {

"Grid Exist": true,

"New Grid": true,

"Approve Grid": false

}

},

{

"stateName": "Connecticut",

"data": {

"Grid Exist": false,

"New Grid": true,

"Approve Grid": true

}

},

{

"stateName": "Delaware",

"data": {

"Grid Exist": true,

"New Grid": false,

"Approve Grid": false

}

},

{

"stateName": "District of Columbia",

"data": {

"Grid Exist": false,

"New Grid": true,

"Approve Grid": true

}

},

{

"stateName": "Florida",

"data": {

"Grid Exist": true,

"New Grid": true,

"Approve Grid": true

}

}

// Add more states with their respective data here

]

}