Certainly! Let's go through each line of the provided code and explain what it does:

**signUp Function**

javascript

Copy code

function signUp(event) {

// prevents page from refreshing

event.preventDefault();

* The signUp function is triggered by an event (usually a form submission).
* event.preventDefault(); stops the default form submission behavior, preventing the page from refreshing.

javascript

Copy code

// to activate spinner on button when clicked

const getSpin = document.querySelector(".spin");

getSpin.style.display = "inline-block";

* Selects the spinner element with the class spin.
* Displays the spinner by setting its display style to inline-block.

javascript

Copy code

const getName = document.getElementById("name").value;

const getEmail = document.getElementById("email").value;

const getPassword = document.getElementById("password").value;

const getConfirmPassword = document.getElementById("confirmPassword").value;

* Retrieves the values from the input fields with IDs name, email, password, and confirmPassword.

javascript

Copy code

if (getName === "" || getEmail === "" || getPassword === "" || getConfirmPassword === "") {

Swal.fire({

icon: 'info',

text: 'All fields are required!',

confirmButtonColor: '#2D85DE'

})

getSpin.style.display = "none";

}

* Checks if any of the input fields are empty.
* If any field is empty, displays an alert using Swal.fire and hides the spinner.

javascript

Copy code

if (getConfirmPassword !== getPassword) {

Swal.fire({

icon: 'info',

text: 'Passwords do not match',

confirmButtonColor: '#2D85DE'

})

getSpin.style.display = "none";

}

* Checks if the passwords do not match.
* If they don't match, displays an alert using Swal.fire and hides the spinner.

javascript

Copy code

else {

// converted to formdata

const signHeader = new FormData();

signHeader.append("name", getName);

signHeader.append("email", getEmail);

signHeader.append("password", getPassword);

signHeader.append("password\_confirmation", getConfirmPassword);

* Creates a new FormData object.
* Appends the name, email, password, and confirm password values to the FormData object.

javascript

Copy code

const signMethod = {

method: 'POST',

body: signHeader

}

const url = "https://accosmart.com.ng/yorubalearning/api/register\_admin";

* Prepares a request object with the method POST and the FormData as the body.
* Sets the URL for the API endpoint to register a new admin.

javascript

Copy code

fetch(url, signMethod)

.then(response => response.json())

.then(result => {

console.log(result)

if (result.status === "success") {

Swal.fire({

icon: 'success',

text: `${result.message}`,

confirmButtonColor: '#2D85DE'

})

// delay for 5 seconds

setTimeout(() => {

location.href = "index.html"

}, 5000)

}

else {

Swal.fire({

icon: 'danger',

text: `${result.message}`,

confirmButtonColor: '#2D85DE'

})

getSpin.style.display = "none";

}

})

.catch(error => console.log('error', error));

}

}

* Sends the request to the server using fetch.
* If the response is successful (status === "success"), displays a success message and redirects to index.html after 5 seconds.
* If not, displays an error message and hides the spinner.
* Catches any errors that occur during the fetch and logs them to the console.

**logIn Function**

javascript

Copy code

function logIn(event) {

event.preventDefault();

// to activate spinner on button when clicked

const getSpin = document.querySelector(".spin");

getSpin.style.display = "inline-block";

const getEmail = document.getElementById("email").value;

const getPassword = document.getElementById("password").value;

if (getEmail === "" || getPassword === "" ) {

Swal.fire({

icon: 'info',

text: 'All fields are required!',

confirmButtonColor: '#2D85DE'

})

getSpin.style.display = "none";

}

else {

// converted to formdata

const logHeader = new FormData();

logHeader.append("email", getEmail);

logHeader.append("password", getPassword);

const logMethod = {

method: 'POST',

body: logHeader

}

const url = "https://accosmart.com.ng/yorubalearning/api/admin\_login";

fetch(url, logMethod)

.then(response => response.json())

.then(result => {

console.log(result)

localStorage.setItem("admin", JSON.stringify(result))

if (result.hasOwnProperty("email")) {

location.href = "dashboard.html"

}

else {

Swal.fire({

icon: 'danger',

text: 'All fields are required!',

confirmButtonColor: '#2D85DE'

})

getSpin.style.display = "none";

}

})

.catch(error => console.log('error', error));

}

}

* Similar to signUp, this function handles the login process.
* It collects email and password values, checks if they are not empty, and sends them to the login API endpoint.
* If login is successful, stores the result in localStorage and redirects to dashboard.html.
* If login fails, displays an error message.

**getDashData Function**

javascript

Copy code

function getDashData() {

const getSpin = document.querySelector(".pagemodal");

getSpin.style.display = "block";

const category = document.getElementById("category")

const lm = document.getElementById("learnmat")

const sc = document.getElementById("subCat")

const quiz = document.getElementById("quiz")

const student = document.getElementById("student")

const adminId = document.getElementById("adminId")

const getToken = localStorage.getItem("admin");

const myToken = JSON.parse(getToken);

const token = myToken.token;

const dashHeader = new Headers();

dashHeader.append("Authorization", `Bearer ${token}`);

const dashMethod = {

method: 'GET',

headers: dashHeader

}

const url = `${baseUrl}admin\_dashboardapi`;

fetch(url, dashMethod)

.then(response => response.json())

.then(result => {

console.log(result)

category.innerHTML = result.total\_number\_of\_categories;

lm.innerHTML = result.total\_number\_of\_learningmaterial;

sc.innerHTML = result.total\_number\_of\_subcategories;

quiz.innerHTML = result.total\_number\_of\_quize;

student.innerHTML = result.total\_number\_of\_students;

adminId.innerHTML = `Hello ${result.admin\_name}`

adminId.style.color = "#2D85DE";

adminId.style.fontSize = "20px";

getSpin.style.display = "none";

})

.catch(error => console.log('error', error));

}

* Fetches and displays data for the admin dashboard.
* It uses a token from localStorage to authorize the request.
* Populates various elements on the dashboard with the fetched data.

**studentModal Function**

javascript

Copy code

function studentModal(event) {

event.preventDefault();

const student = document.querySelector(".allstudent");

const modal = document.getElementById("dash-modal");

modal.style.display = "block";

const getToken = localStorage.getItem("admin");

const myToken = JSON.parse(getToken);

const token = myToken.token;

const dashHeader = new Headers();

dashHeader.append("Authorization", `Bearer ${token}`);

const getMethod = {

method: 'GET',

headers: dashHeader

}

let data = [];

const url = `${baseUrl}top\_three\_students`;

fetch(url, getMethod)

.then(response => response.json())

.then(result => {

console.log(result)

if (result.length === 0) {

student.innerHTML = "No Records Found";

}

else {

result.map((item) => {

data += `

<div class="search-card">

<div class="d-flex justify-content-between">

<p>Name:</p>

<p>${item.name}</p>

</div>

<div class="d-flex justify-content-between">

<p>Email:</p>

<p>${item.email}</p>

</div>

<div class="d-flex justify-content-between">

<p>Phone Number:</p>

<p>${item.phone\_number}</p>

</div>

<div class="d-flex justify-content-between">

<p>Total Score:</p>

<p>${item.total\_score}</p>

</div>

<div class="d-flex justify-content-between">

<p>Position:</p>

<p>${item.position}</p>

</div>

</div>

`

student.innerHTML = data;

})

}

})

.catch(error => console.log('error', error));

}

* Opens a modal displaying the top three students.
* Uses a token from localStorage to authorize the request.
* Populates the modal with student information fetched from the server.

**closeDashModal Function**

javascript

Copy code

function closeDashModal() {

const modal = document.getElementById("dash-modal");

modal.style.display = "none";

}

* Closes the dashboard modal.

**getAllStudents Function**

javascript

Copy code

function getAllStudents() {

const table = document.getElementById("table-id");

const getSpin = document.querySelector(".pagemodal");

getSpin.style.display = "block";

const getToken = localStorage.getItem("admin");

const myToken = JSON.parse(getToken);

const token = myToken.token;

const dashHeader = new Headers();

dashHeader.append("Authorization", `Bearer ${token}`);

const getMethod = {

method: 'GET',

headers: dashHeader

}

let data = [];

const url = `${baseUrl}get\_all\_students`;

fetch(url, getMethod)

.then(response => response.json())

.then(result => {

console.log(result)

if (result.length === 0) {

table.innerHTML = "No Records Found!";

getSpin.style.display = "none";

}

else {

result.map((item) => {

data += `

<tr>

<td>${item.name}</td>

<td>${item.email}</td>

<td>${item.phone\_number}</td>

<td>${item.position}</td>

<td>${item.total\_score}</td>

</tr>

`

table.innerHTML = data;

getSpin.style.display = "none";

})

}

})

.catch(error => console.log('error', error));

}

* Fetches and displays a list of all students in a table.
* Uses a token from localStorage to authorize the request.
* Populates the table with student information fetched from the server.

**createCategory Function**

javascript

Copy code

function createCategory(event) {

event.preventDefault();

const getSpin = document.querySelector(".spin");

getSpin.style.display = "inline-block";

const getName = document.getElementById("cat").value;

const getImage = document.getElementById("imcat").files[0];

if (getName === "" || getImage === "") {

Swal.fire({

icon: 'info',

text: 'All fields are required!',

confirmButtonColor: '#2D85DE'

})

getSpin.style.display = "none";

}

else {

const getToken = localStorage.getItem("admin");

const myToken = JSON.parse(getToken);

const token = myToken.token;

const dashHeader = new Headers();

dashHeader.append("Authorization", `Bearer ${token}`);

const catData = new FormData();

catData.append("name", getName);

catData.append("image", getImage);

const catMethod = {

method: 'POST',

headers: dashHeader,

body: catData

}

const url = `${baseUrl}create\_category`;

fetch(url, catMethod)

.then(response => response.json())

.then(result => {

console.log(result)

if (result.status === "success") {

Swal.fire({

icon: 'success',

text: `${result.message}`,

confirmButtonColor: '#2D85DE'

})

setTimeout(() => {

location.reload();

}, 5000)

}

else {

Swal.fire({

icon: 'warning',

text: `${result.message}`,

confirmButtonColor: '#2D85DE'

})

getSpin.style.display = "none";

}

})

.catch(error => console.log('error', error));

}

}

* Handles the creation of a new category.
* Collects category name and image, checks if they are not empty, and sends them to the create category API endpoint.
* If the category is created successfully, displays a success message and reloads the page after 5 seconds.

**getCatList Function**

javascript

Copy code

function getCatList() {

const scrol = document.querySelector(".scroll-object");

const getToken = localStorage.getItem("admin");

const myToken = JSON.parse(getToken);

const token = myToken.token;

const dashHeader = new Headers();

dashHeader.append("Authorization", `Bearer ${token}`);

const catMethod = {

method: 'GET',

headers: dashHeader,

}

let data = [];

const url = `${baseUrl}category\_list`;

fetch(url, catMethod)

.then(response => response.json())

.then(result => {

console.log(result)

if (result.length === 0) {

scrol.innerHTML = "No Records Found";

}

else {

result.map((item) => {

data += `

<div class="search-card">

<img src="${item.image}" alt="${item.name}">

<p>${item.name}</p>

<div class="text-right">

<button class="update-button">Update</button>

<button class="delete-button">Delete</button>

</div>

</div>

`

scrol.innerHTML = data;

})

}

})

* Fetches and displays a list of categories.
* Uses a token from localStorage to authorize the request.
* Populates the scrollable element with category information fetched from the server.

These functions together form a basic frontend for managing user authentication and administrative tasks, such as managing categories and displaying data on a dashboard.

2/2

Top of Form

Bottom of Form

Sure, here's an explanation of each line of the code:

```javascript

const getToken = localStorage.getItem("admin");

```

1. This line retrieves an item called `"admin"` from the browser's local storage.

2. `localStorage` is a web storage object that allows you to save key/value pairs in a web browser.

3. `getItem("admin")` accesses the value associated with the key `"admin"` in local storage and returns it as a string.

```javascript

const myToken = JSON.parse(getToken);

```

1. This line parses the JSON string stored in `getToken` into a JavaScript object.

2. `JSON.parse(getToken)` converts the string retrieved from local storage back into an object. This is necessary because `localStorage` stores data as strings.

```javascript

const token = myToken.token;

```

1. This line extracts the `token` property from the `myToken` object.

2. `myToken.token` accesses the value of the `token` property from the parsed object.

```javascript

const dashHeader = new Headers();

```

1. This line creates a new `Headers` object.

2. `Headers` is a web API interface used to create and manage HTTP headers, which are key/value pairs that describe the specifics of an HTTP request or response.

```javascript

dashHeader.append("Authorization", `Bearer ${token}`);

```

1. This line adds a new header to the `dashHeader` object.

2. `dashHeader.append("Authorization", `Bearer ${token}`)` sets the `Authorization` header to `Bearer ${token}`.

3. `Bearer ${token}` is the format used for passing the token in the Authorization header, where `Bearer` is the type of token and `${token}` is the actual token value obtained earlier. This is typically used in API requests for authentication purposes.