Converting to Ajax

Introduction

The website that we are building has a very major flaw on the home page. Every time we post something on the website it refreshes.

To overcome this, we are going to incorporate AJAX into our website. Whenever we submit the form for creating a post, we will submit it via AJAX, get the response in the form of a JSON and append it to the page.

Creating a Post:: Sending Data

We will be going to use JQUERY AJAX for the website.

- In **{ layout.ejs }** paste the link **{ <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>** } to include jquery just below noty link.
- We have to create a JS file that fetches the data from the form and sends it in the JSON format to the action.
- In the **js folder** inside the **assets** folder, we create a file **{home_posts.js}.**
- Include the script **{home_posts.js}** inside the **{home.ejs}** file to load the script.
- While creating a post, we need two things -
    - A function that handles the submission of the post
    - A function that receives the data of the created post and displays it.
- Create a function that sends the data to the controller action in **{home_posts.js}**
- Whenever the form is submitted, we don't want it to submit own its own. Hence, we will use the parameter **preventDefault**.
- Submit the form manually using AJAX.

- Send in the data that we create the post for and serialize it. (serializing means to convert the form data into JSON { key-value pair }).
- Once we have submitted the form we will receive it in the { **post_controller.js }** and view the data in the **{ post_controller.js }.**
- Check whether the request is AJAX or not. The type of an AJAX request is - > **{ XML HTTP Request } (XHR).**
- If the request is AJAX then we have to return JSON with a status.
- Alongside data, the general format of interacting when we are sending data back by JSON is to include a message. The message can be that **"Post is created"**.

{home_posts.js}

```
{
    // method to submit the form data for new post using AJAX
    let createPost = function(){
        let newPostForm = $('#new-post-form');

        newPostForm.submit(function(e){
            e.preventDefault();

            $.ajax({
                type: 'post',
                url: '/posts/create',
                data: newPostForm.serialize(),
                success: function(data){
                   console.log(data);
                }, error: function(error){
                    console.log(error.responseText);
                }
            });
        });
    }


    // method to create a post in DOM


    createPost();
}
```

{ post_controller.js }

```
module.exports.create = async function(req, res){
    try{
        let post = await Post.create({
            content: req.body.content,
            user: req.user._id
        });

        if (req.xhr){
            return res.status(200).json({
                data: {
                    post: post
                },
                message: "Post created!"
            });
        }

        req.flash('success', 'Post published!');
        return res.redirect('back');

    }catch(err){
        req.flash('error', err);
        return res.redirect('back');
    }

}
```

***EXTRA**: You can look at the link below to know about JQuery*

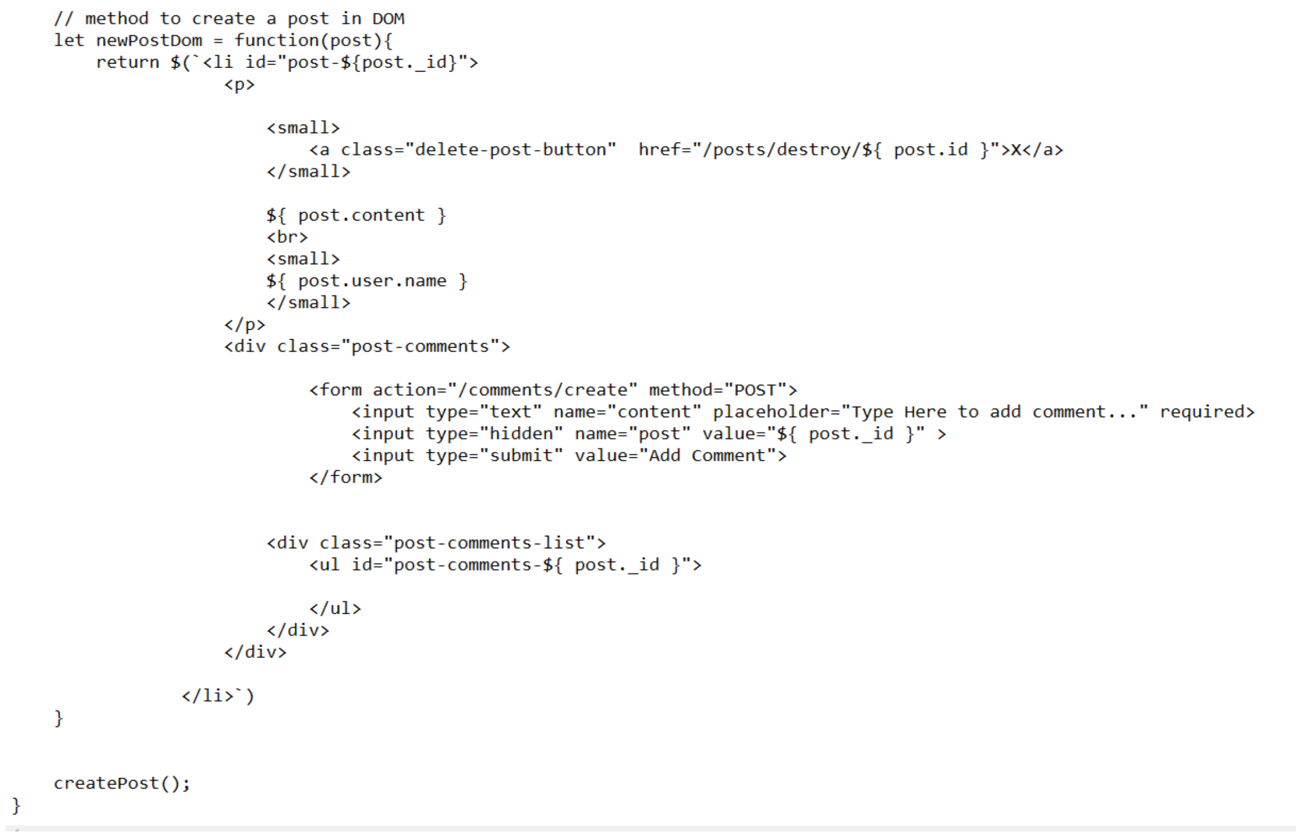
https://code.jquery.com/

Creating a Post:: Receiving Data

- For deleting the post we need some id that can uniquely identify the post. This will come in handy when we progress in this lecture.
- Add class to the delete button so that we can give it common styling and also attach click listeners to it.
- For creating the post in the DOM, we need a function that will help us in converting the text of HTML into a JQuery object.
- Create the function and pass the post data that we have received and return the text of HTML.
- The data that has been displayed has another key data that needs to be sent as a post.
- Prepend it to the list of posts that are displayed on the screen.

Thus, we have successfully created a post, displayed on the screen with AJAX. Our page is not getting refreshed anymore while creating a new post.

{home_posts.js}



```
    // method to create a post in DOM
    let newPostDom = function(post){
        return $(`<li id="post-${post._id}">
                    <p>

                        <small>
                            <a class="delete-post-button"  href="/posts/destroy/${ post.id }">X</a>
                        </small>

                        ${ post.content }
                        <br>
                        <small>
                        ${ post.user.name }
                        </small>
                    </p>
                    <div class="post-comments">

                            <form action="/comments/create" method="POST">
                                <input type="text" name="content" placeholder="Type Here to add comment..." required>
                                <input type="hidden" name="post" value="${ post._id }" >
                                <input type="submit" value="Add Comment">
                            </form>


                        <div class="post-comments-list">
                            <ul id="post-comments-${ post._id }">

                            </ul>
                        </div>
                    </div>

                </li>`)
    }


    createPost();
}
```

Deleting a Post

Post is now created and displayed using AJAX without the need to refresh. Now, let us try to do the following things:

1. Sort the post according to the nearest post which was created in time first **{Reverse chronological order}.**

2. Deleting the post.

Sorting the Post

- In **{ home_controller }** we need to add a sort function for the post.

Deleting the Post

- For deleting the post we need two things - To be able to send the data to the controller to delete it, and once we get confirmation all we need to do is remove that post from the DOM.
- Let's assume that we will be getting data that has the **id** of the post that is deleted. (we will be sending it in the URL)
- We have just created a function to delete id from the DOM but we have not received the post id from the server & handled the AJAX request.

Revising

- We went to our **{home_posts.js}** and created a function that sends the post id to be deleted, blocks the natural behavior of the delete link, and sends it via AJAX parallelly.
- When the function sends the AJAX request it { function }also receives some data that is the post id and it will be removed.
- The function that will be sending the AJAX request we need to populate the delete link argument and put it on the delete link.

{home_posts.js}

```
// method to delete a post from DOM
let deletePost = function(deleteLink){
    $(deleteLink).click(function(e){
        e.preventDefault();

        $.ajax({
            type: 'get',
            url: $(deleteLink).prop('href'),
            success: function(data){
                $(`#post-${data.data.post_id}`).remove();
            },error: function(error){
                console.log(error.responseText);
            }
        });

    });
}
```

{ post_controller.js }

```
module.exports.destroy = async function(req, res){

    try{
        let post = await Post.findById(req.params.id);

        if (post.user == req.user.id){
            post.remove();

            await Comment.deleteMany({post: req.params.id});


            if (req.xhr){
                return res.status(200).json({
                    data: {
                        post_id: req.params.id
                    },
                    message: "Post deleted"
                });
            }

            req.flash('success', 'Post and associated comments deleted!');

            return res.redirect('back');
        }else{
            req.flash('error', 'You cannot delete this post!');
            return res.redirect('back');
        }

    }catch(err){
        req.flash('error', err);
        return res.redirect('back');
    }

}
```

Assignment

Your task is now to add the following features to your social media website.

- Display notifications of creation and deletion using **NOTY**.
- Add/ delete comments dynamically with **AJAX** and then move on to **NOTY.**

Summary

- We created a JS file that sends the form data via AJAX.
- Using **preventDefault** we have prevented the default behavior to get submitted and we send the data to the server via AJAX parallelly asynchronously.
- On the server-side we used **request.XHR** and we checked whether the request is an AJAX request or not.
- If it was the **AJAX** request we have sent data in JSON format and displayed it using a function.
- For the delete link to be activated, we created another function that sends the delete request via AJAX. Once we get a successful response, we remove that element from the DOM.