in simple JavaScript we use add event listener to multiple buttons, we get the object data having details which button is click and we can perform any operation on that particular button.
But now I am learning Express js how can get to know which particular button is clicked and further I can perform any task as express js use route system...?

In Express.js, the concept of button clicks and event listeners is different from the client-side JavaScript approach. Express.js is a server-side framework for building web applications, and it doesn't directly handle button clicks like client-side JavaScript does.

In Express.js, you typically handle user interactions through HTTP requests and routes. When a user clicks a button in a web page, it triggers an HTTP request to the server. The server then uses the Express.js routing system to determine which route should handle that request.

Here's a simple example to illustrate how you can handle button clicks in Express.js:

1. Set up your Express.js server:

javascript Copy code

```javascript
const express = require('express');
const app = express();

app.listen(3000, () => {
  console.log('Server started on port 3000');
});
```

2. Define a route to handle button clicks:

javascript Copy code

```javascript
app.get('/button-click', (req, res) => {
  // Perform any desired task here
  res.send('Button clicked!');
});
```

3. In your HTML file, set up a button that triggers the HTTP request:

html Copy code

```html
<button onclick="window.location.href='/button-click'">Click me</button>
```

When the user clicks the button, it sends an HTTP GET request to the `**/button-click**` route on your server. The Express.js routing system then matches the route and triggers the corresponding callback function, where you can perform any desired task.

Remember to adapt the route and logic to your specific use case. This example uses a GET request, but you can also handle other HTTP methods like POST, PUT, or DELETE based on your requirements.

It's important to understand that Express.js is primarily a backend framework, and it's not directly responsible for handling client-side events like button clicks. It provides the server-side infrastructure for handling those events and responding to them appropriately.

Ok,
But I want some data of clicked button, as in client side we can perform

box.addEventListener("click", (p) => {
value = p.target.id}

here, we get element by p.targert.
Then what we can do for server side ?

‹ 2 / 2 ›

In Express.js, when handling a button click on the server side, you don't have direct access to the clicked button's data like you do in client-side JavaScript. The server only receives the HTTP request triggered by the button click.

However, you can pass data from the client-side to the server-side through the HTTP request. There are a few ways to achieve this:

1. Query parameters: You can include the button's data as a query parameter in the URL of the HTTP request. For example:

```javascript
app.get('/button-click', (req, res) => {
  const buttonData = req.query.data;
  // Perform any desired task with the buttonData
  res.send('Button clicked!');
});
```

   In your client-side JavaScript, you can construct the URL with the button's data as a query parameter and make the HTTP request to that URL.
2. Request body: If the button click requires more complex data, you can send it in the request body using a POST or PUT request instead of a GET request. Here's an example using the `body-parser` middleware to parse the request body:

```javascript
const express = require('express');
const bodyParser = require('body-parser');
const app = express();

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

app.post('/button-click', (req, res) => {
  const buttonData = req.body.data;
  // Perform any desired task with the buttonData
  res.send('Button clicked!');
});
```

   In your client-side JavaScript, you can use the `fetch` API or an AJAX library like Axios to send a POST request with the button's data in the request body.

These are just a couple of examples. Depending on your specific requirements, you can choose the appropriate method for passing the button's data from the client-side to the server-side.

ok, let me explain in detail.
I want to delete todolist item when someon click on checkbox. I am using ejs and MongoDb for database.
my ejs file is shown bellow :

```
<body>
  <div class="box" id="heading">
    <h1 id="heading"><%= list %></h1>
  </div>

  <div class="box">
```

```
    <% newItem.forEach((item)=>{ %>
      <div class="item">
        <input type="checkbox" class="task" onclick="window.location.href='/delItem'">
        <p ><%= item.listItem %></p>
      </div>
    <% }) %>

    <form action="/" method="post" class="item">
      <!-- <label for="">Enter your list</label> -->
      <input type="text" placeholder="Enter " name="listItem">
      <button type="submit" name="list" value=<%= list %> >+</button>
    </form>

  </div>
</body>
```

‹ 2 / 2 ›

To delete a todo list item when someone clicks on the checkbox, you can modify your EJS file and the Express.js server code. Here's how you can approach it:

1. Modify your EJS file:

html — Copy code

```
<body>
  <div class="box" id="heading">
    <h1 id="heading"><%= list %></h1>
  </div>

  <div class="box">
    <% newItem.forEach((item) => { %>
      <div class="item">
        <form action="/delItem" method="post">
          <!-- Use a hidden input field to pass the item's ID to the server -->
          <input type="hidden" name="itemId" value="<%= item._id %>">
          <input type="checkbox" class="task" onclick="this.parentNode.submit()">
          <p><%= item.listItem %></p>
        </form>
      </div>
    <% }) %>

    <form action="/" method="post" class="item">
      <input type="text" placeholder="Enter" name="listItem">
      <button type="submit" name="list" value=<%= list %>>+</button>
    </form>
  </div>
</body>
```

In the modified code, we added a hidden input field inside the form for each todo item. This input field holds the ID of the item from the database. When the checkbox is clicked, the corresponding form is submitted, sending the item's ID to the server.

2. Update your Express.js server code:

javascript — Copy code

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const mongoose = require('mongoose');
```

```
// Set up your database connection and model (assuming you have a TodoItem model)
mongoose.connect('mongodb://localhost:27017/todoDB', { useNewUrlParser: true, useUnifiedTopology: true });
const TodoItem = mongoose.model('TodoItem', { listItem: String });

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

// Handle the POST request to delete a todo item
app.post('/delItem', async (req, res) => {
  const itemId = req.body.itemId;
  try {
    // Delete the item from the database using the provided item ID
    await TodoItem.findByIdAndDelete(itemId);
    res.redirect('/');
  } catch (err) {
    res.status(500).send(err);
  }
});

// Your other routes and server setup

app.listen(3000, () => {
  console.log('Server started on port 3000');
});
```

In the server code, we added a route (`**/delItem**`) to handle the POST request when a checkbox is clicked. We extract the item ID from the request body (`**req.body.itemId**`) and use it to delete the corresponding item from the MongoDB database. After deleting the item, we redirect the user back to the home page (`**res.redirect('/')**`).

Note that you'll need to have a MongoDB database set up and a TodoItem model defined for this code to work properly. Adjust the code according to your database configuration and schema.

With these changes, when someone clicks the checkbox for a todo item, the corresponding item will be deleted from the database, and the user will be redirected back to the home page.