BACK-END DEVELOPMENT

WEEK 5 – React + Axios Integration

CADT
IDT

After Finishing This Lecture:

- What is Axios
- Axios basics: **GET**/**POST** from client
- **useEffect**, **useState** for loading data
- React component
- Basic form submission
- Displaying and deleting data
- Authorization in Request header

What is Axios

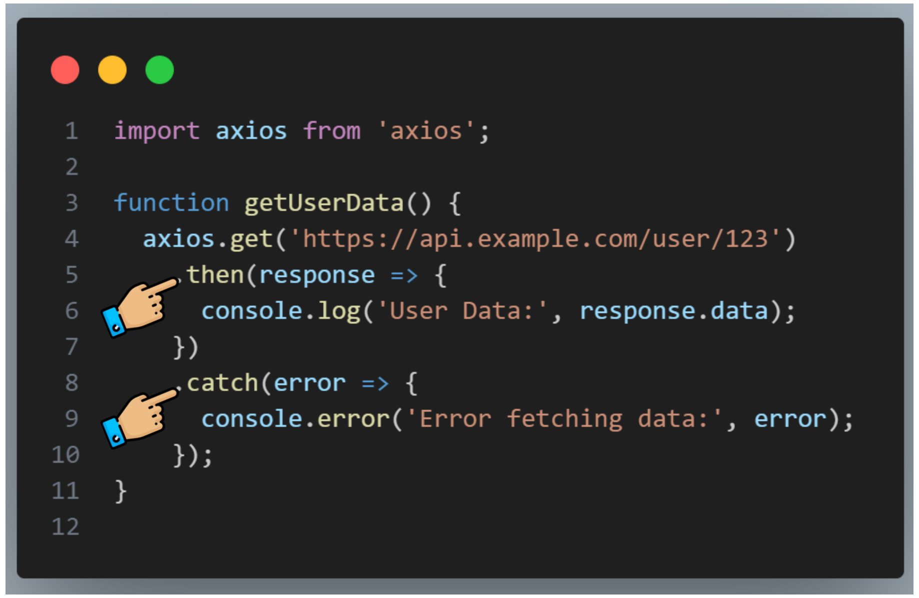
🔍 Introduction to Axios

Axios is a promise-based HTTP client for JavaScript, used to make requests to external resources (like APIs). It works in both the browser and Node.js environments.

Axios is commonly used in frontend applications (like **React**, **Vue**, **Angular, etc.**) and backend services (Node.js) to handle API requests.

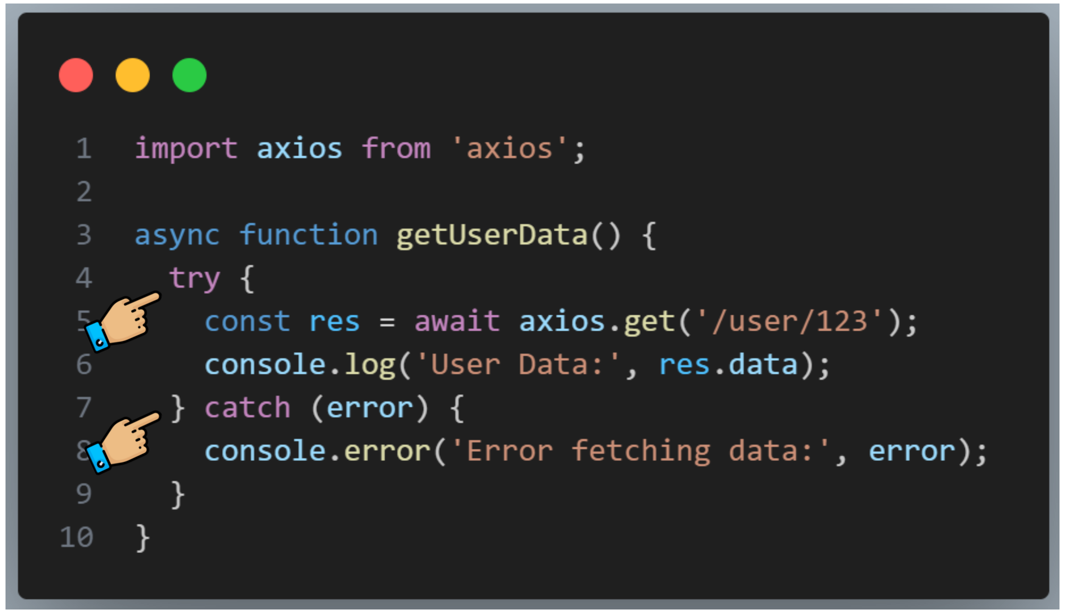
AXIOS as a Promise

- ❑ Using **Axios** with `.then()` and `.catch()`



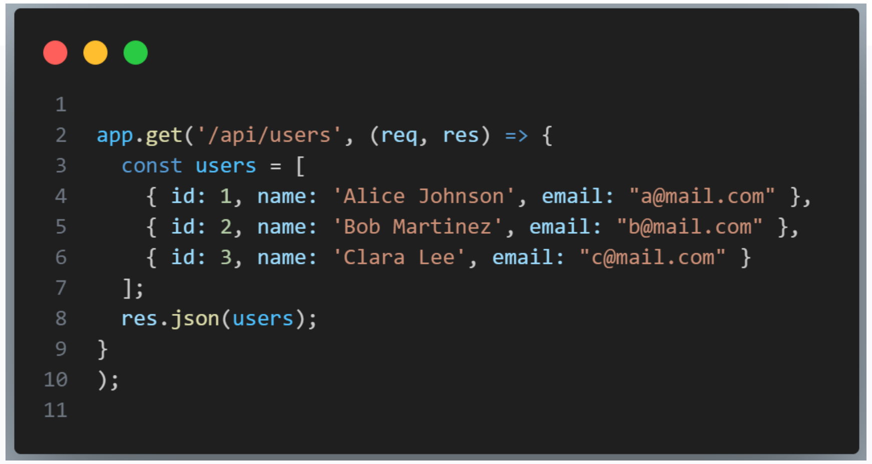
```
import axios from 'axios';

function getUserData() {
  axios.get('https://api.example.com/user/123')
    .then(response => {
      console.log('User Data:', response.data);
    })
    .catch(error => {
      console.error('Error fetching data:', error);
    });
}
```

- ❑ Using **Axios** with `async/await` and `try/catch`

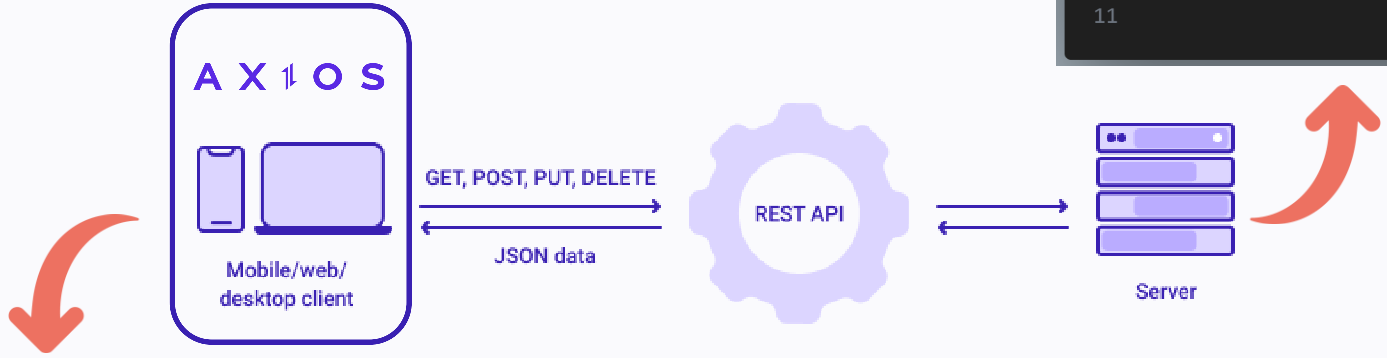


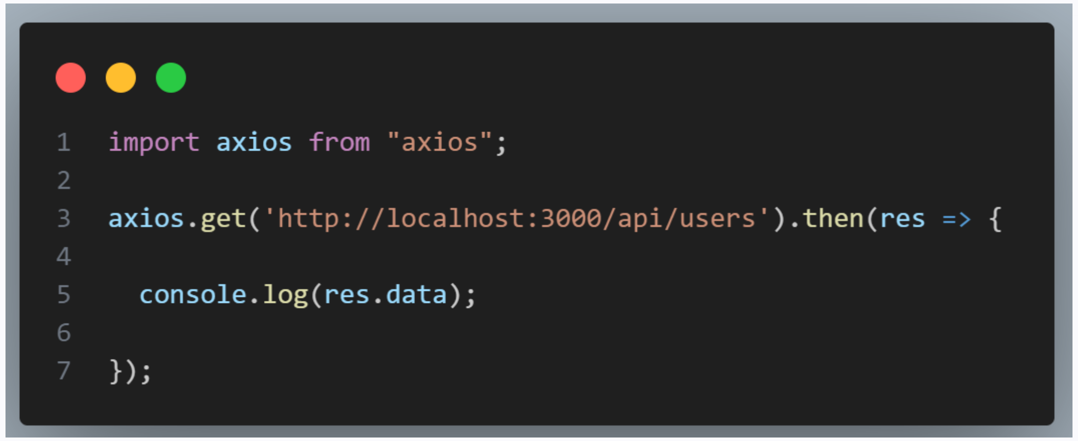
```
import axios from 'axios';

async function getUserData() {
  try {
    const res = await axios.get('/user/123');
    console.log('User Data:', res.data);
  } catch (error) {
    console.error('Error fetching data:', error);
  }
}
```

Understanding Axios



```
app.get('/api/users', (req, res) => {
  const users = [
    { id: 1, name: 'Alice Johnson', email: "a@mail.com" },
    { id: 2, name: 'Bob Martinez', email: "b@mail.com" },
    { id: 3, name: 'Clara Lee', email: "c@mail.com" }
  ];
  res.json(users);
}
);
```





```
import axios from "axios";

axios.get('http://localhost:3000/api/users').then(res => {

  console.log(res.data);

});
```

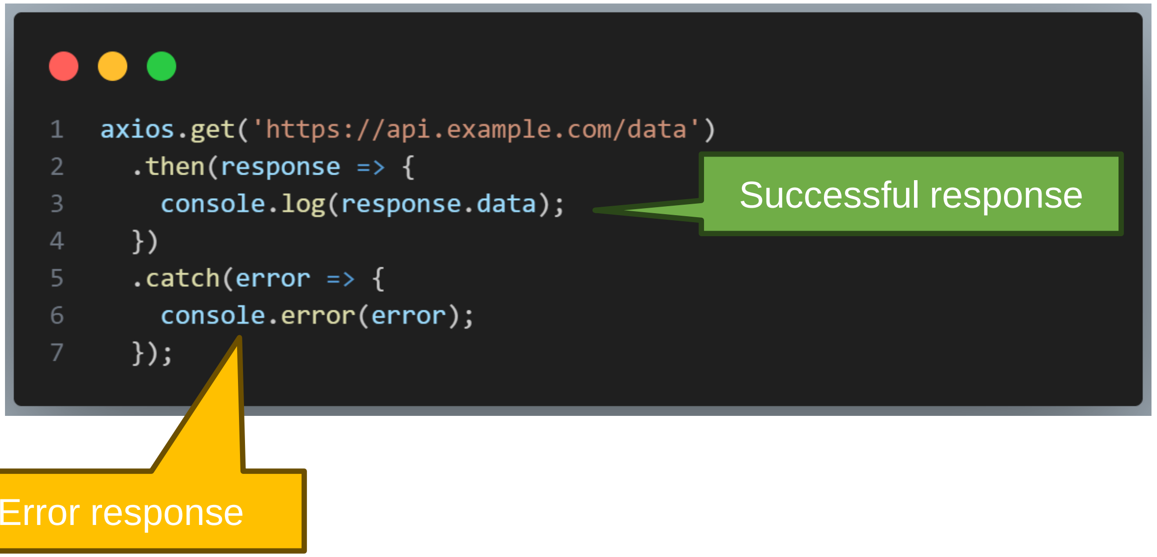
Installing Axios

Axios allows developers to easily send **HTTP** requests using methods like **GET**, **POST**, **PUT**, and **DELETE** in NodeJS applications.

- **Installing Axios**

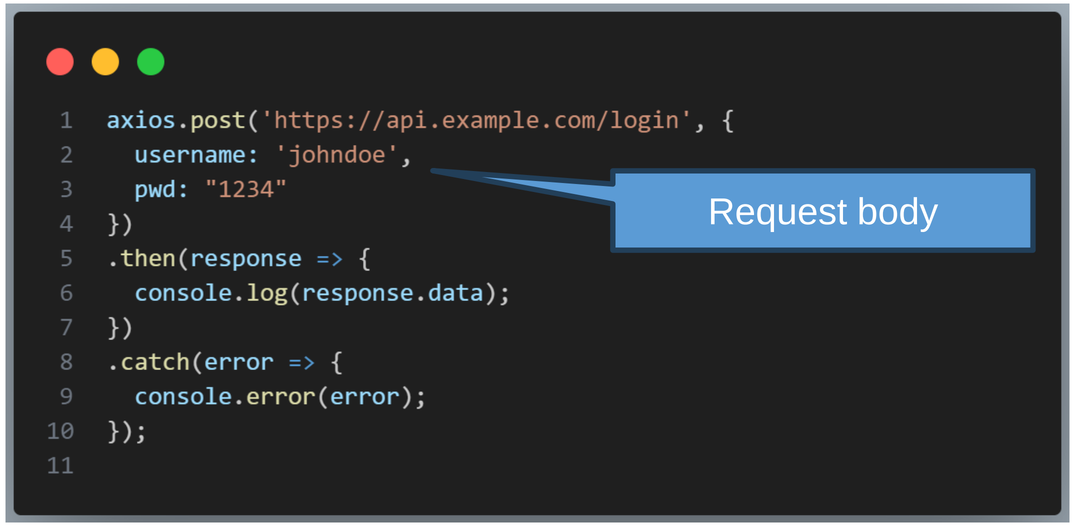
```
$: npm install axios
```

- **GET Request**



```
axios.get('https://api.example.com/data')
  .then(response => {
    console.log(response.data);
  })
  .catch(error => {
    console.error(error);
  });
```

- **POST Request**



```
axios.post('https://api.example.com/login', {
  username: 'johndoe',
  pwd: "1234"
})
.then(response => {
  console.log(response.data);
})
.catch(error => {
  console.error(error);
});
```

Axios usage

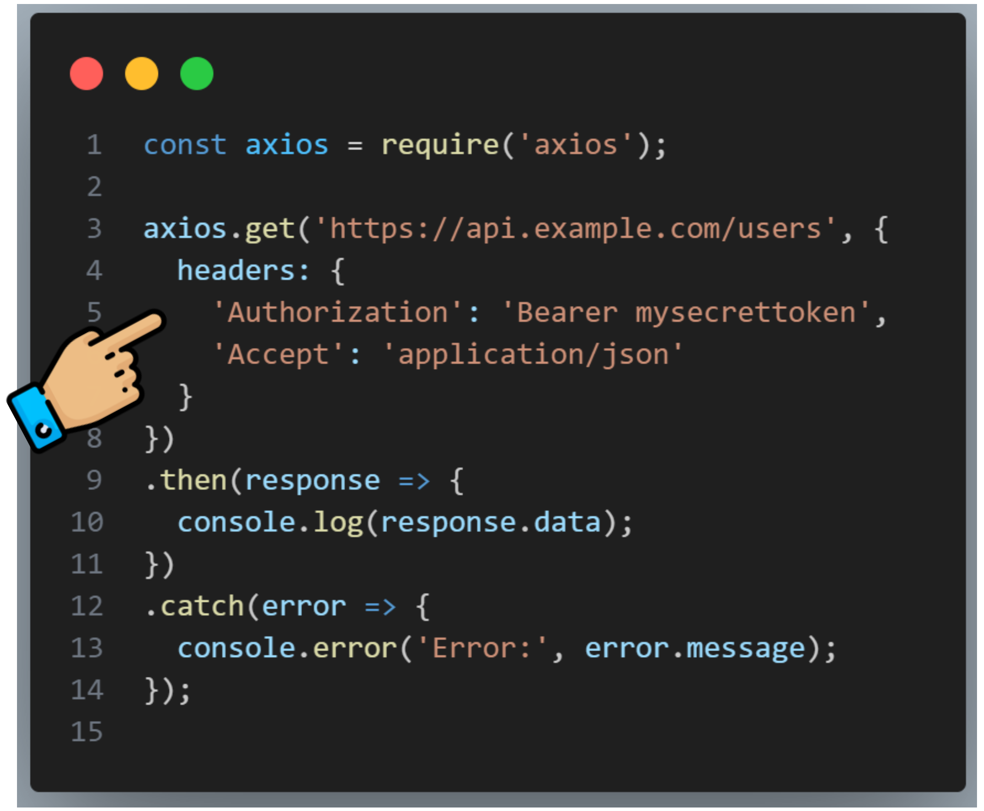
Axios Requests with **Headers** in Node.js

What about passing Headers in a request

HTTP headers are key-value pairs sent with HTTP requests and responses. They carry metadata about the request or the response.

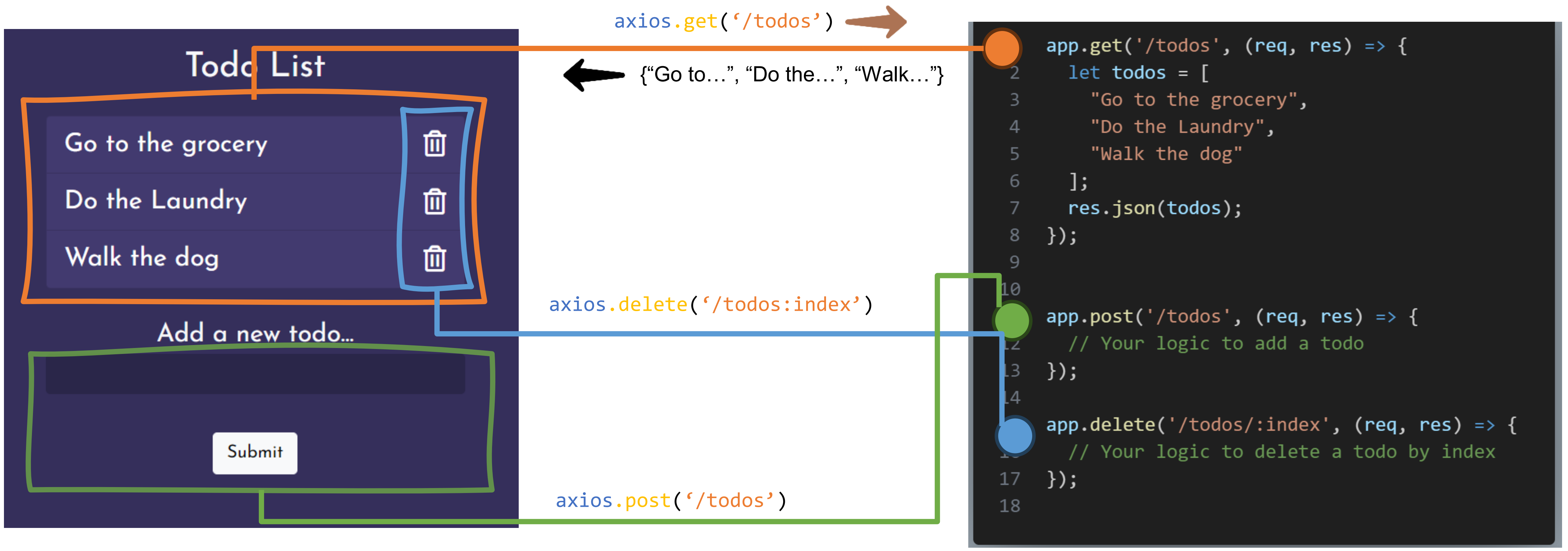
Common uses:

- Sending authentication tokens (e.g., Authorization)
- Specifying data formats (e.g., Content-Type)
- Controlling caching
- Setting custom headers for your API



```javascript
const axios = require('axios');

axios.get('https://api.example.com/users', {
  headers: {
    'Authorization': 'Bearer mysecrettoken',
    'Accept': 'application/json'
  }
})
.then(response => {
  console.log(response.data);
})
.catch(error => {
  console.error('Error:', error.message);
});
```

How to Do an API Integration?



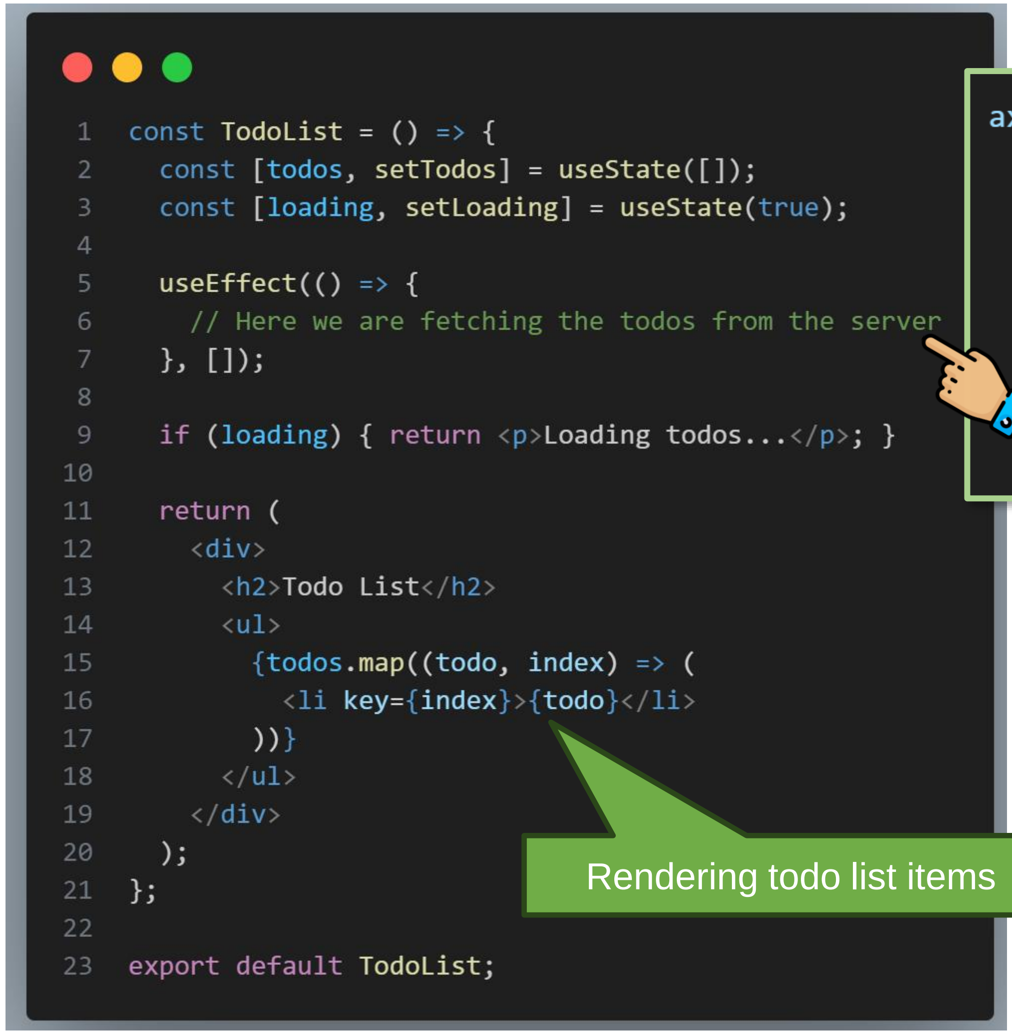
```
app.get('/todos', (req, res) => {
  let todos = [
    "Go to the grocery",
    "Do the Laundry",
    "Walk the dog"
  ];
  res.json(todos);
});


app.post('/todos', (req, res) => {
  // Your logic to add a todo
});

app.delete('/todos/:index', (req, res) => {
  // Your logic to delete a todo by index
});
```

(My Web App)

(Backend APIs)

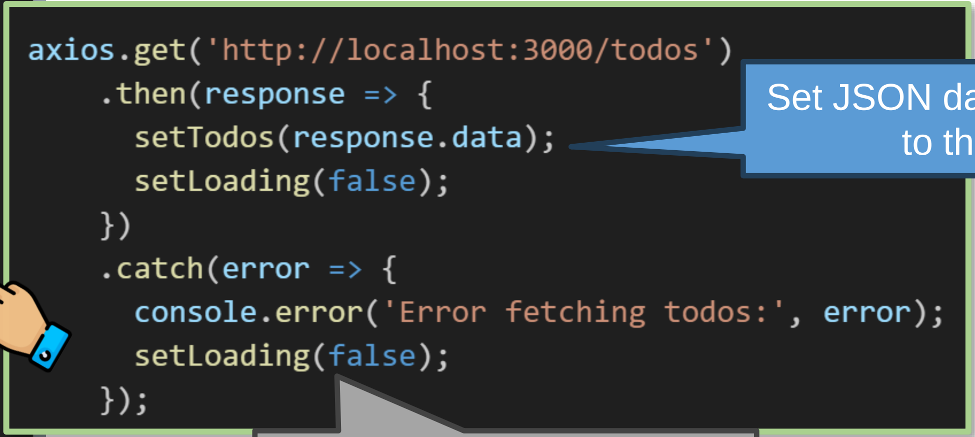
ReactJS

useEffect, useState for loading “Todo List”



```
const TodoList = () => {
  const [todos, setTodos] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    // Here we are fetching the todos from the server
  }, []);

  if (loading) { return <p>Loading todos...</p>; }

  return (
    <div>
      <h2>Todo List</h2>
      <ul>
        {todos.map((todo, index) => (
          <li key={index}>{todo}</li>
        ))}
      </ul>
    </div>
  );
};

export default TodoList;
```

A component in ReactJS



```
axios.get('http://localhost:3000/todos')
    .then(response => {
      setTodos(response.data);
      setLoading(false);
    })
    .catch(error => {
      console.error('Error fetching todos:', error);
      setLoading(false);
    });
```

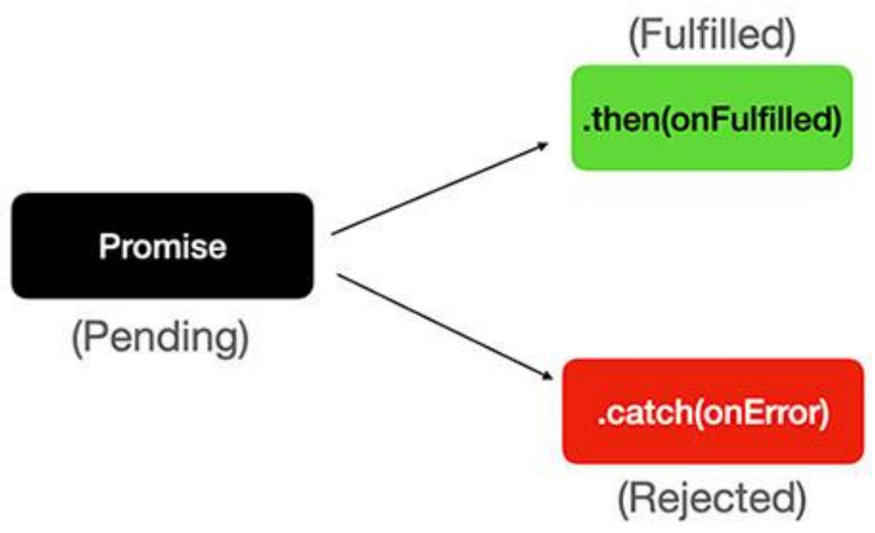
Set JSON data from an API to the state

Loading status

Rendering todo list items

ReactJS

useEffect, useState for loading “Todo List”



Loading Data from APIs

Several actions should be taken:

In response **(Todos successfully loaded)**

1. Update the UI with the Fetched Data
2. Hide Any Loading Indicators
3. Clear Any Previous Error Messages
4. Saving into cache

In error response:

1. Show an Error Message to the User
2. Show a button to reload/fetch the data again

ReactJS

useEffect, useState for loading “Todo List”

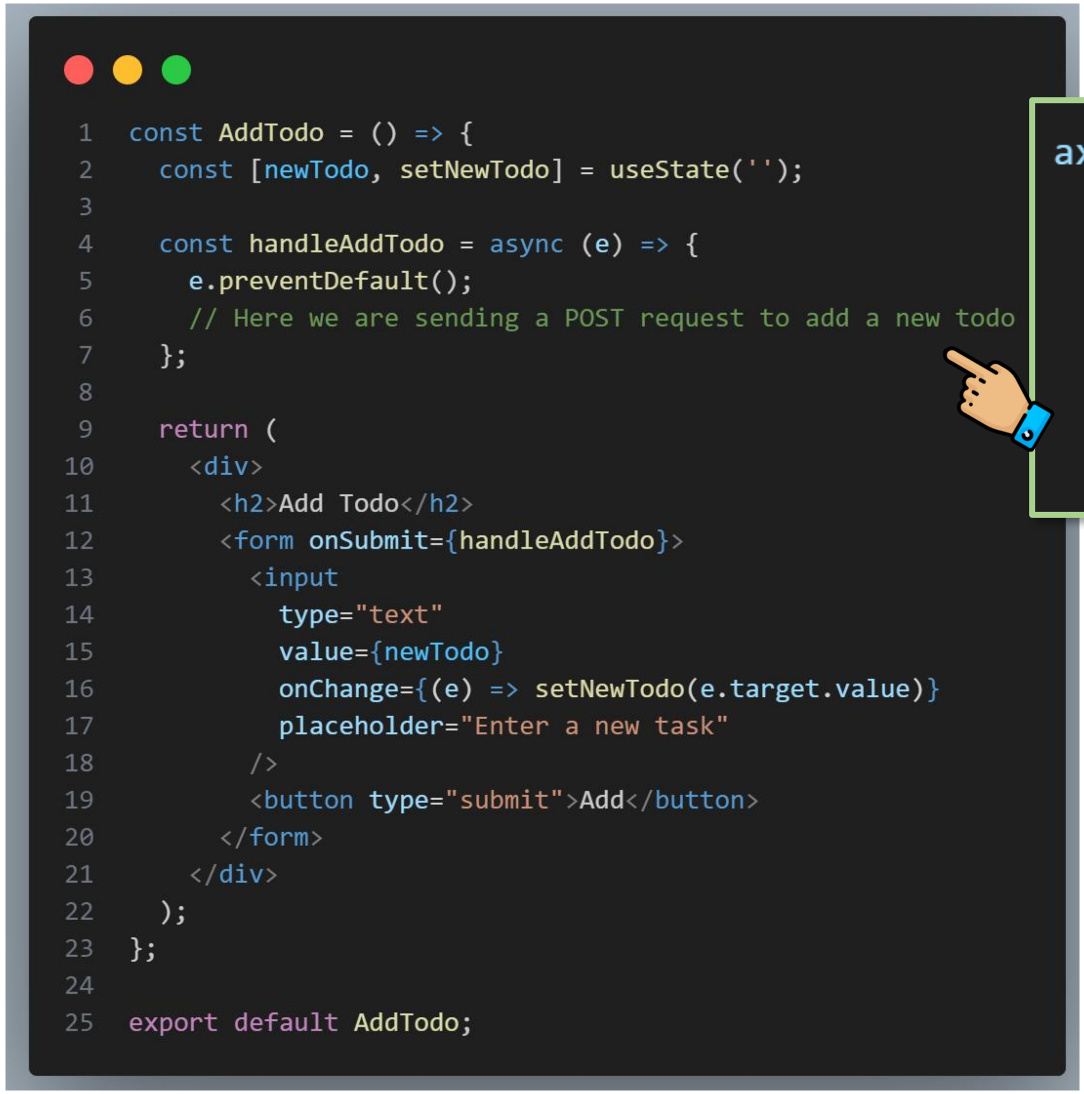
Loading Data from APIs

🔍 General Use Cases

Use Case	Description
Product Listings	Fetch product data from an e-commerce backend to display on the store page.
User Profile	Get user details (name, avatar, preferences) to display on a dashboard or profile page.
Blog Posts / News Feed	Load posts or articles from a content API to show on a blog/news site.
Weather Info	Fetch live weather data from an external API (like OpenWeatherMap).
Search Results	Submit a search query to the server and display the matching results.
Comments Section	Load comments related to a specific post or item.
Chat Messages	Fetch recent messages when entering a chat room or conversation thread.
Analytics Dashboard	Display charts and stats using data from analytics APIs.
Task / To-Do List	Load tasks assigned to a user from a task management backend.
Image Gallery	Fetch image metadata (URLs, titles) from a server or cloud service (e.g., Cloudinary).
Many more…	

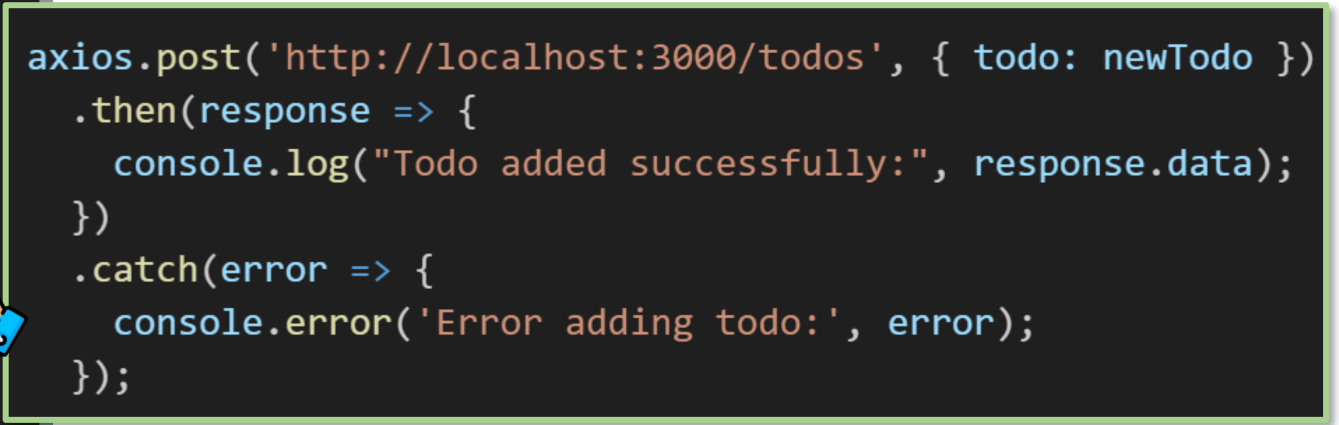
ReactJS

useState for creating a new “Todo”



```
const AddTodo = () => {
  const [newTodo, setNewTodo] = useState('');

  const handleAddTodo = async (e) => {
    e.preventDefault();
    // Here we are sending a POST request to add a new todo
  };

  return (
    <div>
      <h2>Add Todo</h2>
      <form onSubmit={handleAddTodo}>
        <input
          type="text"
          value={newTodo}
          onChange={(e) => setNewTodo(e.target.value)}
          placeholder="Enter a new task"
        />
        <button type="submit">Add</button>
      </form>
    </div>
  );
};

export default AddTodo;
```

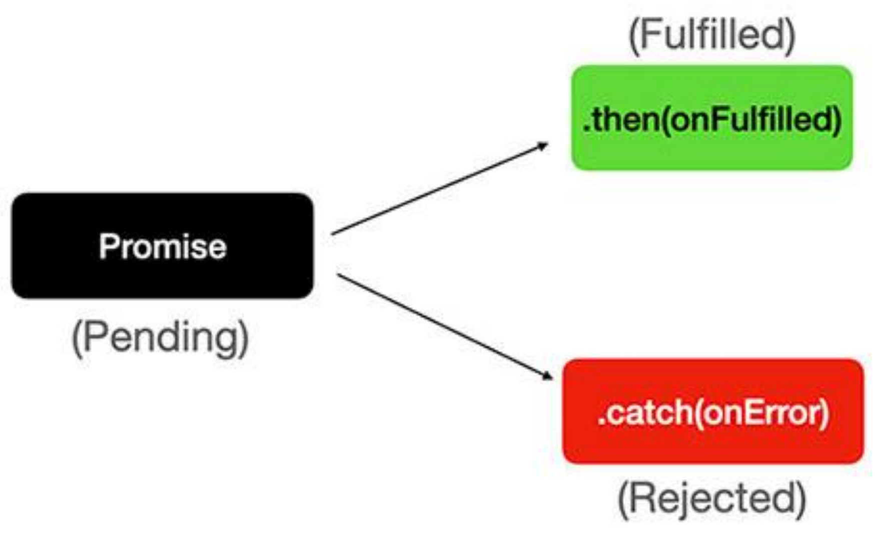
A component in ReactJS



```
axios.post('http://localhost:3000/todos', { todo: newTodo })
  .then(response => {
    console.log("Todo added successfully:", response.data);
  })
  .catch(error => {
    console.error('Error adding todo:', error);
  });
```

ReactJS

useState for creating a new "Todo"



Creating data in APIs

Several actions should be taken:

In response (A Todo successfully added/created)

1. Clear the input field
2. Update the Todo List in the UI
3. Show Feedback (Toast or Message) – (Eg. "A new Todo is added")

In error response:

1. Show a Clear Error Message to the User
2. Allow the User to Retry

ReactJS

useState for creating a new “Todo”

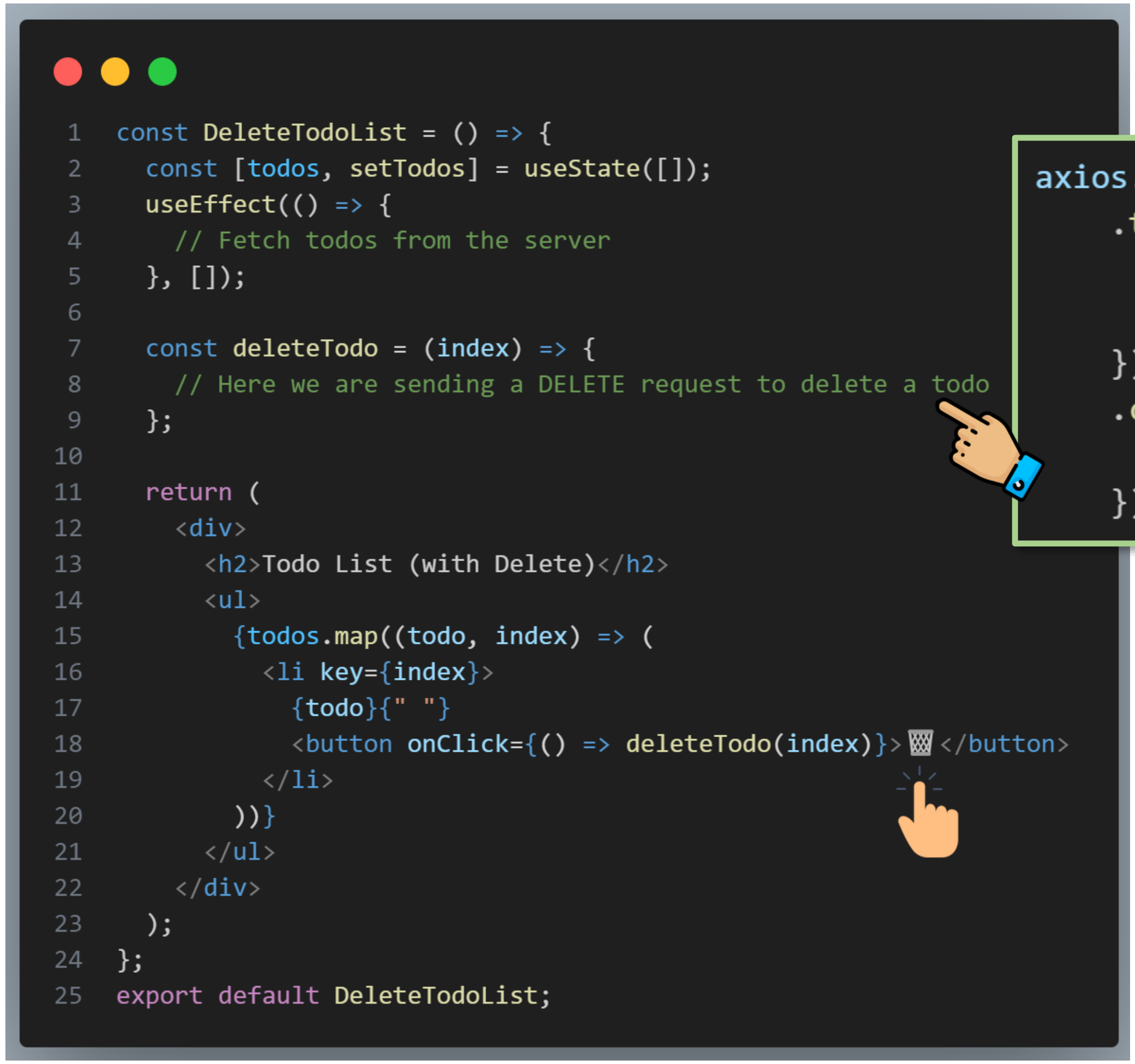
Creating data in APIs

🔍 General Use Cases

Use Case	Description
Sign Up / Register User	Submit new user data to create an account.
Add Product to Store	Admin adds a new product with name, price, and image.
Create New Blog Post	Author submits a new article via a form.
Submit Feedback / Contact Form	User fills out and sends a message to the site.
Add Task to To-Do List	User adds a new task to a personal task manager.
Post a Comment	User comments on a blog post or video.
Upload Photo or File	File input uploads images/documents to the server.
Create Booking/Reservation	User makes a booking for a hotel, event, or service.
Create Invoice	In a billing system, create and store new invoice data.
Many more…	

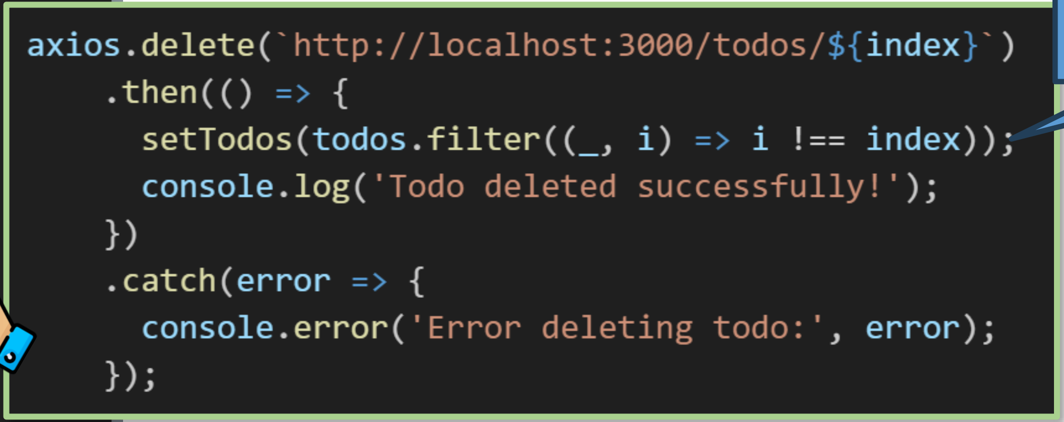
ReactJS

Deleting a “Todo” by an INDEX



```
const DeleteTodoList = () => {
  const [todos, setTodos] = useState([]);
  useEffect(() => {
    // Fetch todos from the server
  }, []);

  const deleteTodo = (index) => {
    // Here we are sending a DELETE request to delete a todo
  };

  return (
    <div>
      <h2>Todo List (with Delete)</h2>
      <ul>
        {todos.map((todo, index) => (
          <li key={index}>
            {todo}{" "}
            <button onClick={() => deleteTodo(index)}>🗑</button>
          </li>
        ))}
      </ul>
    </div>
  );
};
export default DeleteTodoList;
```

A component in ReactJS

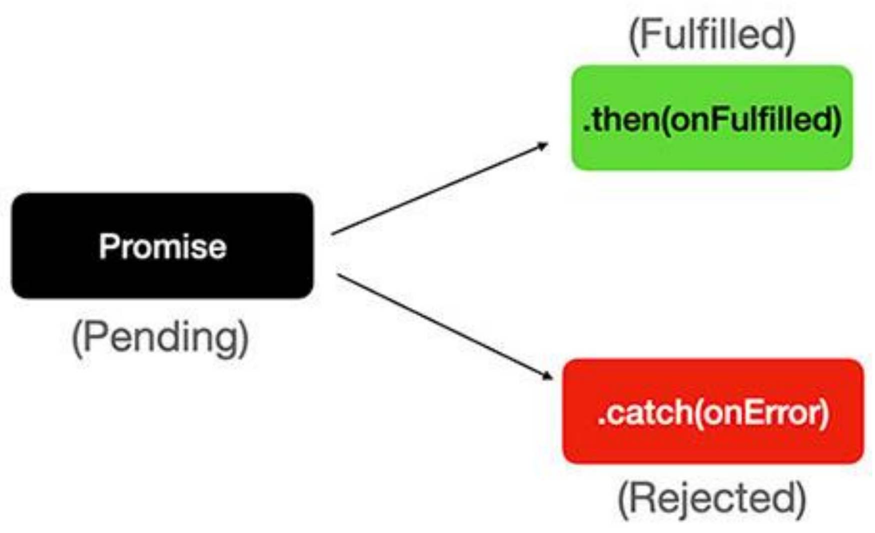


```
axios.delete(`http://localhost:3000/todos/${index}`)
    .then(() => {
      setTodos(todos.filter((_, i) => i !== index));
      console.log('Todo deleted successfully!');
    })
    .catch(error => {
      console.error('Error deleting todo:', error);
    });
```

Remove the deleted todo

ReactJS

Deleting a “Todo” by an INDEX



Deleting data in APIs

Several actions should be taken:

In response (A Todo successfully removed/deleted)

1. Update the UI Immediately (Re-fetch the List, remove the deleted one)
2. Show a Success Message
3. Reset Any Loading States

In error response:

1. Show a Clear Error Message to the User

ReactJS

Deleting a “Todo” by an INDEX

Deleting data in APIs

🔍 General Use Cases

Use Case	Description
Delete User Account	Remove a user and their data from the system.
Remove Product	Admin deletes a product from inventory.
Delete a Comment	User deletes their comment or admin moderates content.
Remove Task	User deletes a to-do item.
Cancel Booking	User cancels an upcoming reservation.
Clear Notifications	Delete read or dismissed notifications.
Remove Item from Cart	Delete an item from the shopping cart.
Delete Uploaded File	Remove an image or document from the server.
Many more…	

Authentication

Basic understanding authentication check
in between ReactJS and ExpressJS

How to pass an Authentication Bearer token from a React frontend to an Express.js backend, using Axios and Authorization headers.

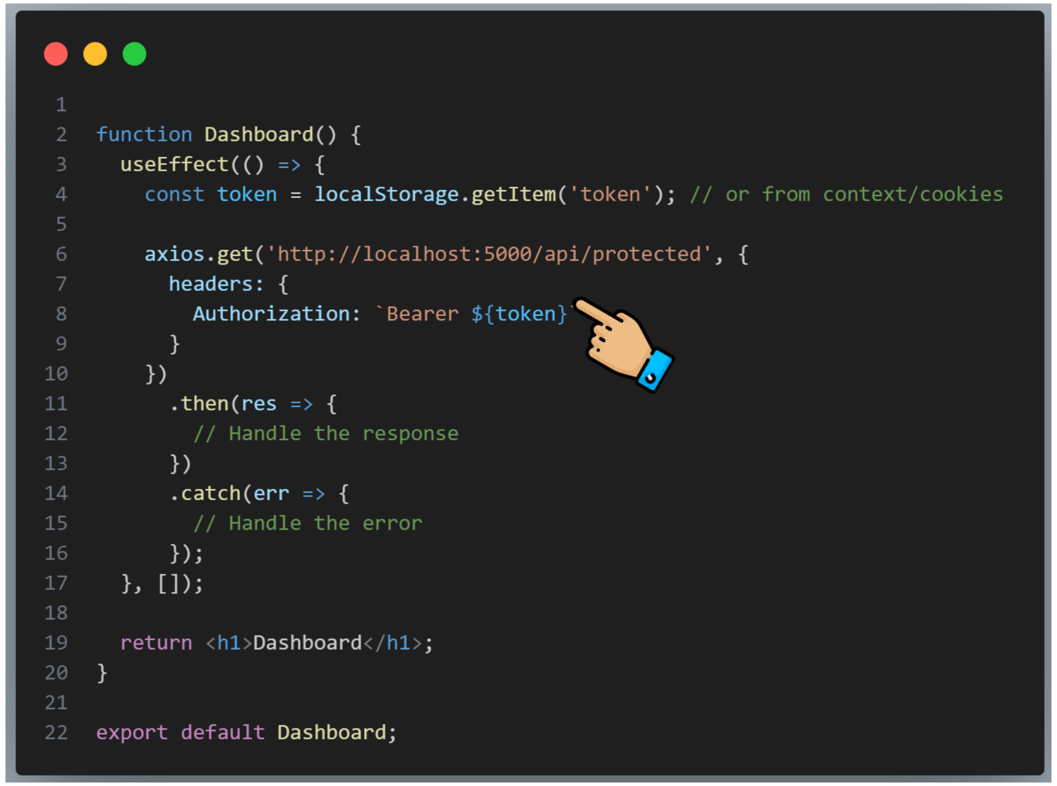
- ✓ ***Frontend (ReactJS):*** *Sends token using Axios.*
- ✓ ***Backend (ExpressJS):*** *Validates the token using middleware.*

Authentication

Basic understanding authentication check
in between ReactJS and ExpressJS

Let's implement it…;

A component in ReactJS



```

function Dashboard() {
  useEffect(() => {
    const token = localStorage.getItem('token'); // or from context/cookies

    axios.get('http://localhost:5000/api/protected', {
      headers: {
        Authorization: `Bearer ${token}`
      }
    })
      .then(res => {
        // Handle the response
      })
      .catch(err => {
        // Handle the error
      });
  }, []);

  return <h1>Dashboard</h1>;
}

export default Dashboard;
```

✅ **Token** typically comes from ***localStorage***, ***sessionStorage***, ***cookie***, or ***React Context*** after login.

What is Bearer Token Authorization?

Bearer Token Authorization is a way for a client (like a frontend app) to prove its identity to a server by including a special token in the HTTP request headers. You can find following format of it.

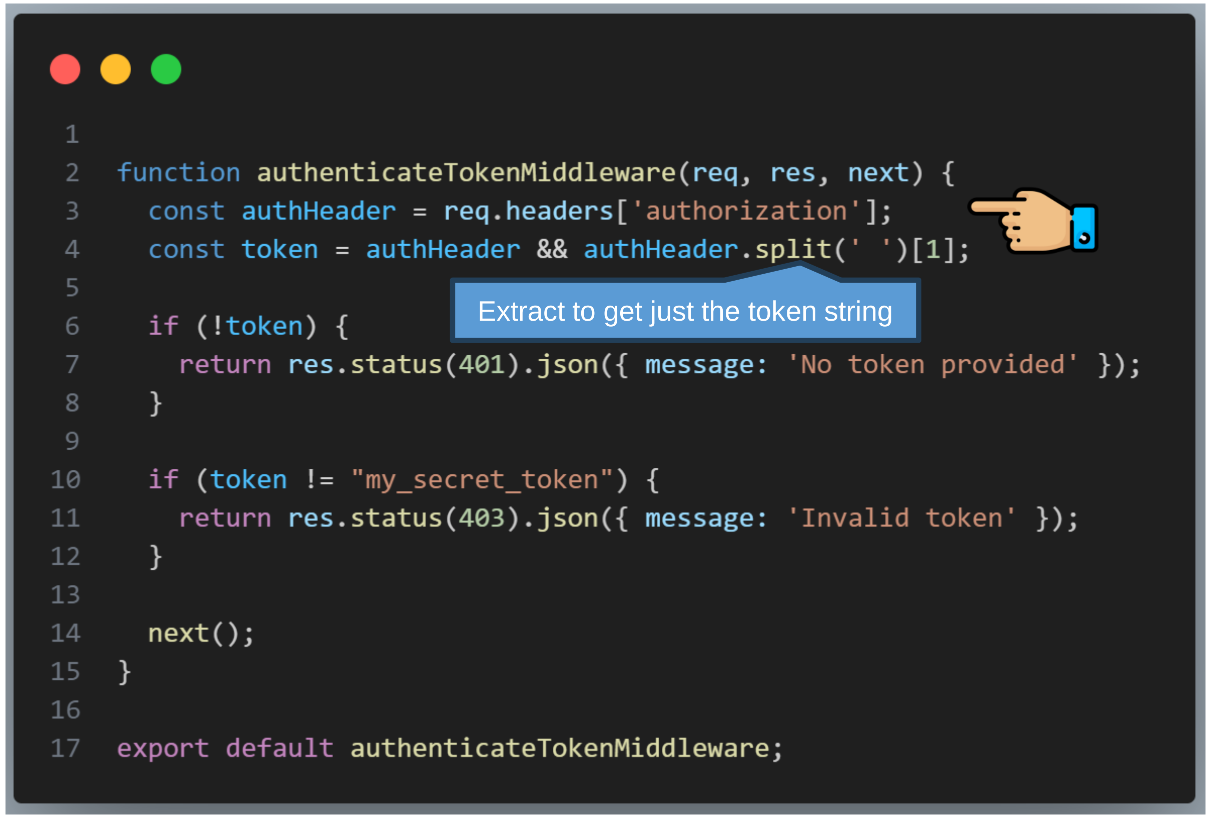
```
Authorization: Bearer <token>
```

Authentication

Basic understanding authentication check
in between ReactJS and ExpressJS

Let's implement it…;

An API middleware in ExpressJS



```
function authenticateTokenMiddleware(req, res, next) {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];

  if (!token) {
    return res.status(401).json({ message: 'No token provided' });
  }

  if (token != "my_secret_token") {
    return res.status(403).json({ message: 'Invalid token' });
  }

  next();
}

export default authenticateTokenMiddleware;
```

Extract to get just the token string

1. **App-Level Middleware**

```
const app = express();
app.use(authenticateTokenMiddleware);

app.get('/api/public', (req, res) => {
  res.send('Even this route is now protected!');
});

app.listen(5000, () => console.log('Server is running'));
```

2. **Group Route Level (Router-Level Middleware)**

```
const app = express();

app.use('/api/protected', authenticateTokenMiddleware);
app.get('/api/protected', (req, res) => {
  res.json({ message: 'Main protected route' });
});

app.listen(5000, () => console.log('Server is running'));
```

3. **Route Level (Per Route)**

```
app.get('/api/protected', authenticateTokenMiddleware, (req, res) => {
  res.json({ message: 'This is protected data', user: req.user });
});
```

Reflections

Questions

Which HTTP method is used in Express.js to delete a todo item by its index?

A) GET
B) POST
C) PUT
D) DELETE

Questions

In React, which hook is commonly used to fetch data on component mount?

A) useState
B) useEffect
C) useContext
D) useReducer

Questions

What is the purpose of setTodos(response.data) in a React component after fetching todo data?

A) To clear the todo list
B) To update the UI with fetched todos
C) To send data to the server
D) To delete all todos

Questions

When *adding a new todo* using Axios in React, which status code usually indicates a successful creation?

A) 200
B) 201
C) 400
D) 500

Questions

What should be done immediately after a todo is successfully deleted in the React state?

A) Reload the entire page
B) Remove the deleted todo from local state
C) Clear all todos from the list
D) Disable the delete button permanently

Questions

If an Axios request to add a todo fails due to no server response, what kind of feedback should the app show?

A) "Todo added successfully!“
B) "No response from server. Please check your connection.“
C) "Invalid todo item.“
D) No feedback is necessary

Questions

What does the Express.js route handler for POST /todos expect to receive in the request body?

A) An array of todos
B) A todo string under the key *todo Eg. {todo: “my task”}*
C) An ID to delete
D) Nothing, it uses query parameters

Questions

In React, what is a good practice after successfully adding a todo?

A) Keep the input field unchanged
B) Clear the input field for the next entry
C) Reload the entire page
D) Disable the add button permanently

Questions

Why might you disable the submit button while an Axios request is in progress?

A) To prevent duplicate submissions
B) To speed up the request
C) To clear the form automatically
D) To disable the whole form permanently

Additional learnings

- AXIOS Guideline: https://axios-http.com/docs/intro
- How To Use Axios with React : https://www.digitalocean.com/community/tutorials/react-axios-react

- **ReactJS: Todo List**

  https://www.youtube.com/watch?v=9wiWzu_tRB0&ab_channel=BroCode

- **ExpressJS: Todo List**

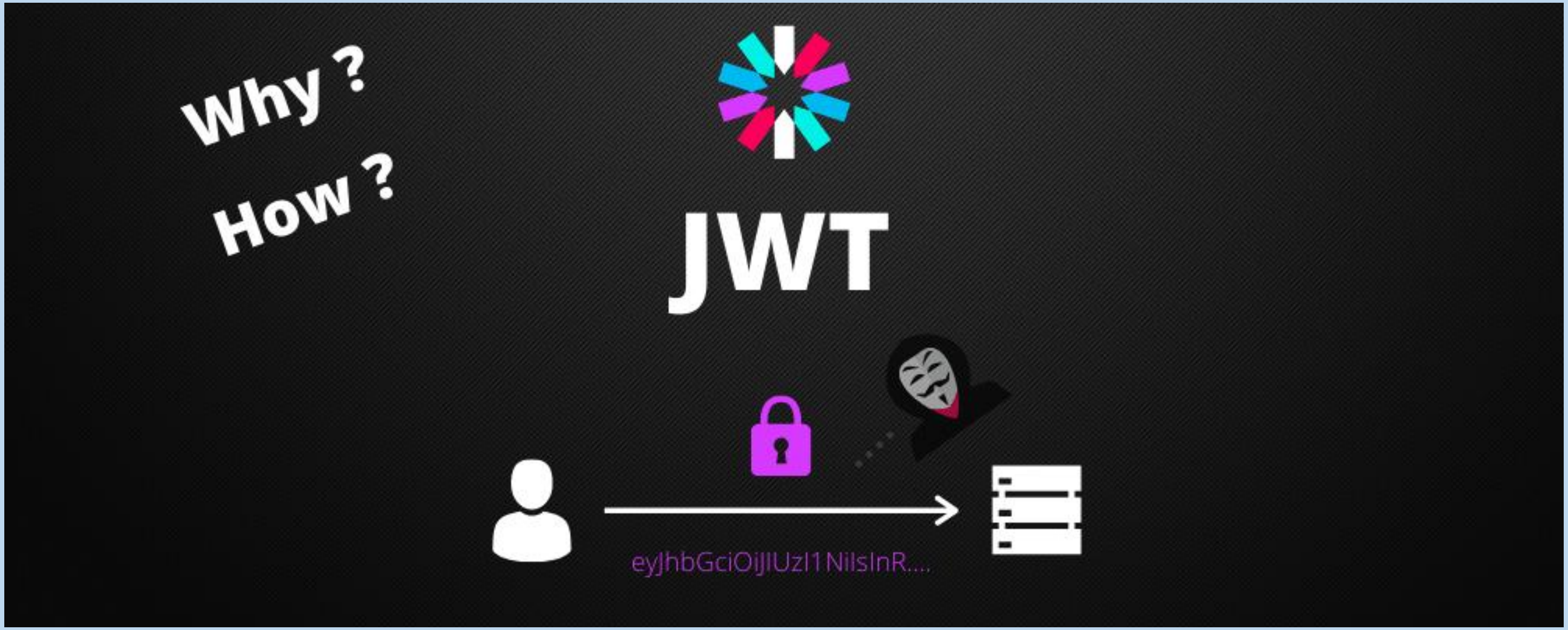
  https://www.youtube.com/watch?v=2u8VAAyvFv0&ab_channel=TheNormieProgrammer

WHAT WE HAVE LEARNT

- What is Axios
- Axios basics: **GET**/**POST** from client
- **useEffect**, **useState** for loading data
- React component
- Basic form submission
- Displaying and deleting data
- Authorization in Request header

WHAT TO LEARN

JWT: JSON Web Tokens



https://www.youtube.com/watch?v=7Q17ubqLfaM