การเชื่อมต่อ Front-End และ Backend

1. แนวคิดพื้นฐานของ Front-End และ Backend

- **Front-End**: ส่วนที่ผู้ใช้มองเห็นและโต้ตอบ (HTML, CSS, JavaScript)
- **Backend**: ส่วนประมวลผลและจัดการข้อมูล (Node.js, PHP, Python, ฐานข้อมูล)
- **API (Application Programming Interface)**: ช่องทางที่ Front-End ติดต่อกับ Backend

---

2. วิธีการเชื่อมต่อ Front-End กับ Backend

2.1 การใช้ Fetch API (Vanilla JavaScript)

ใช้ fetch() เพื่อส่งคำขอ (Request) ไปยัง Backend และรับข้อมูลกลับมา (Response)

ตัวอย่าง: การดึงข้อมูลจาก Backend (GET Request)

```
js

CopyEdit

fetch('http://localhost:3000/products')

  .then(response => response.json()) // แปลง JSON เป็น Object

  .then(data => console.log(data))   // แสดงผลข้อมูล

  .catch(error => console.error('Error:', error));
```

ตัวอย่าง: การส่งข้อมูลไปยัง Backend (POST Request)

```
js

CopyEdit

fetch('http://localhost:3000/products', {

  method: 'POST',

  headers: {

    'Content-Type': 'application/json'

  },

  body: JSON.stringify({ name: 'สินค้าใหม่', price: 100 })
```

```
})
.then(response => response.json())
.then(data => console.log('Success:', data))
.catch(error => console.error('Error:', error));
```

2.2 การใช้ Axios (React.js)

Axios เป็นไลบรารีที่ช่วยจัดการ HTTP Request ได้ง่ายขึ้น

ติดตั้ง Axios

sh

CopyEdit

```
npm install axios
```

ตัวอย่างการใช้ Axios (GET Request)

js

CopyEdit

```
import axios from 'axios';

axios.get('http://localhost:3000/products')
  .then(response => console.log(response.data))
  .catch(error => console.error('Error:', error));
```

ตัวอย่างการใช้ Axios (POST Request)

js

CopyEdit

```
axios.post('http://localhost:3000/products', { name: 'สินค้าใหม่', price: 100 })
  .then(response => console.log('Success:', response.data))
  .catch(error => console.error('Error:', error));
```

---

3. การสร้าง Backend ด้วย Node.js + Express.js

3.1 ติดตั้ง Express.js

sh

CopyEdit

npm init -y

npm install express cors body-parser

3.2 ตัวอย่างโค้ด Backend (Node.js + Express.js)

js

CopyEdit

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

const app = express();
const port = 3000;

app.use(cors());
app.use(bodyParser.json());

// จำลองข้อมูลสินค้า
let products = [
  { id: 1, name: 'สินค้า A', price: 100 },
  { id: 2, name: 'สินค้า B', price: 200 }
];
```

```
// API ดึงข้อมูลสินค้า
app.get('/products', (req, res) => {
  res.json(products);
});

// API เพิ่มสินค้าใหม่
app.post('/products', (req, res) => {
  const newProduct = { id: products.length + 1, ...req.body };
  products.push(newProduct);
  res.json(newProduct);
});

app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});
```

---

4. การเชื่อมต่อกับฐานข้อมูล (MySQL หรือ SQLite)

4.1 ติดตั้งไลบรารีสำหรับฐานข้อมูล

MySQL

sh

```
CopyEdit
npm install mysql2
```

SQLite

sh

CopyEdit

npm install sqlite3

4.2 ตัวอย่างการเชื่อมต่อ MySQL

js

CopyEdit

```
const mysql = require('mysql2');

const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'mydatabase'
});

db.connect(err => {
  if (err) throw err;
  console.log('Connected to MySQL');
});
```

4.3 ตัวอย่างการเชื่อมต่อ SQLite

js

CopyEdit

```
const sqlite3 = require('sqlite3').verbose();

const db = new sqlite3.Database('./database.db', sqlite3.OPEN_READWRITE, (err) => {
  if (err) return console.error(err.message);
```

```
  console.log('Connected to SQLite database.');
});
```

---

5. การทดสอบ API ด้วย Postman

- ใช้ Postman เพื่อทดสอบ API โดยลองส่ง Request ไปที่ http://localhost:3000/products
- ทดสอบ GET, POST, PUT, DELETE และดูผลลัพธ์

---

6. สรุปขั้นตอนการเชื่อมต่อ Front-End และ Backend

1. **Backend**: สร้าง API ด้วย Node.js + Express.js และเชื่อมต่อฐานข้อมูล
2. **Front-End**: ใช้ fetch() หรือ axios ติดต่อกับ API
3. **ทดสอบ**: ใช้ Postman หรือ Console เพื่อตรวจสอบข้อมูล