|  |
| --- |
| Java Script #04 - Async/Await, Fetch, Axios |

Image

**Keyword**

Synchronous - ทำงานทีละคำสั่งตามลำดับ

Asynchronous – ไม่งานพร้อมกัน (ไม่ต้องรอให้คำสั่งแรกทำงานเสร็จ)

Await – รอจนกว่าคำสั่งการทำงานจะเสร็จ

|  |  |  |
| --- | --- | --- |
| คุณสมบัติ | Fetch | Axios \*\*\* |
| การติดตั้ง | เป็น API ในตัวของเบราว์เซอร์ ไม่ต้องติดตั้ง | เป็นไลบรารีภายนอก ต้องติดตั้ง (ผ่าน CDN หรือ npm) |
| การจัดการ JSON | ต้องเรียก response.json() เพื่อแปลงข้อมูลด้วยตัวเอง | แปลงข้อมูล JSON ให้โดยอัตโนมัติ |
| การจัดการข้อผิดพลาด | ไม่จัดการข้อผิดพลาดของ HTTP (เช่น 404, 500) ต้องตรวจสอบ response.ok เอง | จัดการข้อผิดพลาดของ HTTP โดยอัตโนมัติ และส่งไปยัง catch |
| การเข้าถึงข้อมูล | ต้องรอ response.json() และเก็บข้อมูลในตัวแปร | เข้าถึงข้อมูลได้โดยตรงผ่าน response.data |
| การใช้งาน | มีความยืดหยุ่นสูง ใช้ได้ทั้งแบบ .then() และ async/await | นิยมใช้กับ async/await เพราะโค้ดจะสะอาดและอ่านง่ายมาก |

|  |
| --- |
| **การดึงข้อมูลจาก API แบบ fetch** |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>Fetch API Example</title>  </head>  <body>  <h1>โพสต์ล่าสุด</h1>  <button onclick="fetchAndDisplayPosts()">ดึงโพสต์</button>  <ul id="post-list"></ul>  </body>  </html>  <script>  // ประกาศฟังก์ชันเป็น async เพื่อให้สามารถใช้ await ได้  async function fetchAndDisplayPosts() {  const postList = document.getElementById('post-list');  // ใช้ try-catch เพื่อจัดการข้อผิดพลาด  try {  // 1. ส่งคำขอ GET ไปยัง API และรอ response  const response = await fetch('https://jsonplaceholder.typicode.com/posts?\_limit=5');  // 2. ตรวจสอบว่าการเรียก API สำเร็จหรือไม่ (สถานะ 200-299)  if (!response.ok) {  throw new Error(`การดึงข้อมูลล้มเหลว: ${response.status}`);  }  // 3. แปลงข้อมูล response เป็น JSON และรอจนกว่าจะเสร็จ  const posts = await response.json();  // 4. นำข้อมูลที่ได้มาแสดงผลบนหน้าเว็บ  posts.forEach(post => {  const listItem = document.createElement('li');  listItem.innerHTML = `<h3>${post.title}</h3><p>${post.body}</p>`;  postList.appendChild(listItem);  });  console.log('ดึงข้อมูลและแสดงผลโพสต์สำเร็จแล้ว!');  } catch (error) {  // 5. หากมีข้อผิดพลาดจะถูกจับที่นี่  console.error('เกิดข้อผิดพลาดในการทำงาน:', error.message);  postList.innerHTML = '<li>ไม่สามารถโหลดข้อมูลได้ โปรดลองใหม่อีกครั้ง</li>';  }  }  </script> |

|  |
| --- |
| **การดึงข้อมูลจาก API แบบ axios** |
| /\*\*  \* ขั้นตอนการทำงาน:  \* 1. แสดงข้อความ "กำลังโหลดข้อมูล..." ขณะรอผลลัพธ์จาก API  \* 2. ส่งคำขอ GET ด้วย axios.get()  \* 3. เมื่อได้รับข้อมูลสำเร็จ จะแสดงชื่อผู้ใช้แต่ละคนในรายการ  \* 4. หากเกิดข้อผิดพลาด จะแสดงข้อความแจ้งข้อผิดพลาดบนหน้าเว็บ  \*/  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Axios with HTML</title>  <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>  </head>  <body>  <h1>ตัวอย่างการใช้ Axios ใน HTML</h1>  <button onclick="fetchData()">ดึงข้อมูลผู้ใช้</button>  <ul id="user-list"></ul>    <script>  async function fetchData() {  const userList = document.getElementById('user-list');  userList.innerHTML = '<li>กำลังโหลดข้อมูล...</li>';  try {  // ใช้ axios.get() เพื่อส่งคำขอ GET  // axios จะพร้อมใช้งานเป็น global variable หลังจากเราแทรก script CDN  const response = await axios.get('https://jsonplaceholder.typicode.com/users');  const users = response.data;  // ล้างรายการเก่า  userList.innerHTML = '';    // แสดงข้อมูลผู้ใช้  users.forEach(user => {  const listItem = document.createElement('li');  listItem.textContent = user.name;  userList.appendChild(listItem);  });  } catch (error) {  console.error('เกิดข้อผิดพลาดในการดึงข้อมูล:', error.message);  userList.innerHTML = '<li>เกิดข้อผิดพลาดในการดึงข้อมูล</li>';  }  }  </script>  </body>  </html> |

**โจทย์**

สร้างหน้าเว็บเพื่อแสดงผลข้อมูลจาก API พร้อมตกแต่งให้เหมาะสม ตัวอย่าง API เช่น

* <https://github.com/public-apis/public-apis>
* <https://jsonplaceholder.typicode.com/>
* <https://api.nasa.gov/>
* <https://www.newsapi.ai/>
* <https://unsplash.com/developers>
* <https://randomuser.me/>
* <https://fakestoreapi.com/>
* <https://data.go.th/>

ประยุกต์การแสดงผลร่วมกับ Chart ต่างๆ เช่น

* Chart.js: <https://www.chartjs.org/>
* D3.js: <https://d3js.org/>
* Highcharts: <https://www.highcharts.com/>
* Plotly.js: <https://plotly.com/javascript/>

JavaScript Utility Libraries

* Lodash: <https://lodash.com/>
* Day.js**:** <https://day.js.org/>