

การเชื่อมต่อ

FRONTEND

กับ

BACKEND

FETCH API

Fetch API เป็นมาตรฐานสำหรับการร้องขอข้อมูลผ่านเน็ตเวิร์ค
ที่ให้เราสามารถ รับ-ส่ง ข้อมูลระหว่างเว็บ ได้จากเว็บเบราว์เซอร์
โดย Fetch API จะ return ค่า Promise กลับมาเสมอ

รูปแบบ syntax ของ fetch

แบบ **callback**

```
1 fetch(url)
2   .then(res => res.json())
3   .then(json => console.log(json))
4   .catch(error => console.log(error)) // handle the error
5
```

แบบ **async/await**

```
1 const res = await fetch(url)
2 const json = await res.json()
3 console.log(json)
4
```

ตัวอย่างการใช้งาน **fetch**

```
1 const url = 'https://jsonplaceholder.typicode.com/todos'
2 const options = {
3   method: 'POST',
4   headers: {
5     Accept: 'application/json',
6     'Content-Type': 'application/json;charset=UTF-8',
7   },
8   body: JSON.stringify({
9     a: 10,
10    b: 20,
11  }),
12 }
13 fetch(url, options)
14   .then((response) => response.json())
15   .then((data) => {
16     console.log(data)
17   })
18
```

ค่าที่ `fetch()` ส่งกลับมา เป็น object response จะมีค่าข้างในหลักๆ คือ

- **ok** - เป็นค่า true หรือ false
- **status** - ค่า HTTP status code ที่ได้ (ปกติ จะเป็น 200)
- **statusText** - ค่า default

“

ALWAYS REMEMBER

`fetch()` ไม่รองรับ Internet Explorer 11

หรือก่อนหน้า แต่ถ้าเป็น Browser ใหม่ๆ

สามารถใช้งานได้ปกติ

AXIOS

Axios คือ JavaScript Library

ที่ใช้สำหรับเชื่อมต่อข้อมูลกับ API ได้อย่างง่ายดาย ใช้งานได้บนเบราว์เซอร์
และ Node.js


การติดตั้ง Axios

ใช้ npm



```
1 npm install axios
```

ใช้ yarn



```
yarn add axios
```

ตัวอย่างการใช้งาน axios

```
1 import axios from 'axios';  
2  
3 async function getPosts() {  
4   const result = await axios.get('https://jsonplaceholder.typicode.com/posts')  
5   console.log(result.data) // แสดงค่าที่ส่งมาจาก API  
6   console.log(result.status) // แสดงสถานะ code ที่ส่งมาจาก API  
7   console.log(result.statusText) // แสดงสถานะ text ที่ส่งมาจาก API  
8 }  
9  
10 getPosts()
```


การใช้งาน **axios** ร่วมกับ
useState และ **useEffect**
เพื่อดึงข้อมูลจาก API

```
1 import React, { useState, useEffect } from 'react'
2 import axios from 'axios'
3
4 export default function PostList() {
5   const [posts, setPosts] = useState([])
6
7   useEffect(() => {
8     const fetchData = async () => {
9       const { data } = await axios.get(
10         'https://jsonplaceholder.typicode.com/posts'
11       )
12       setPosts(data)
13     }
14     fetchData()
15   }, [])
16 }
17 getPosts()
18
```

การใช้งาน `axios` ร่วมกับ `useEffect` และ `useState` เพื่อดึงข้อมูลจาก API

```
1 import React, { useEffect, useState } from 'react'
2 import axios from 'axios'
3
4 function Users() {
5   const [post, setPost] = useState([])
6
7   useEffect(() => {
8     axios.get('https://jsonplaceholder.typicode.com/users').then((data) => {
9       console.log(data)
10      setPost(data?.data)
11    })
12  }, [])
13
14  return (
15    <div>
16      Users
17      {post.map((item, i) => {
18        return (
19          <div key={i}>
20            <p>{item?.name}</p>
21          </div>
22        )
23      })}
24    </div>
25  )
26 }
27
28 export default Users
```

อิมพอร์ต `useEffect`, `useState` เพื่อใช้งาน
`Hooks`

อิมพอร์ต `axios` เพื่อใช้รับข้อมูลด้วยโปรโตคอล
HTTP

การใช้งาน `axios` ร่วมกับ `useEffect` และ `useState` เพื่อดึงข้อมูลจาก API

```
1 import React, { useEffect, useState } from 'react'
2 import axios from 'axios'
3
4 function Users() {
5   const [post, setPost] = useState([])
6
7   useEffect(() => {
8     axios.get('https://jsonplaceholder.typicode.com/users').then((data) => {
9       console.log(data)
10      setPost(data?.data)
11    })
12  }, [])
13
14  return (
15    <div>
16      Users
17      {post.map((item, i) => {
18        return (
19          <div key={i}>
20            <p>{item?.name}</p>
21          </div>
22        )
23      })}
24    </div>
25  )
26 }
27
28 export default Users
```

ประกาศตัวแปร `post` ไว้สำหรับเก็บข้อมูล และ
สร้างฟังก์ชัน `setPost()` ไว้สำหรับ set ค่าให้ตัว
แปร `post` ด้วย `useState hook` พร้อมด้วยค่า
เริ่มต้นเป็นอาร์เรย์เปล่า `[]`

การใช้งาน `axios` ร่วมกับ `useEffect` และ `useState` เพื่อดึงข้อมูลจาก API

```
1 import React, { useEffect, useState } from 'react'
2 import axios from 'axios'
3
4 function Users() {
5   const [post, setPost] = useState([])
6
7   useEffect(() => {
8     axios.get('https://jsonplaceholder.typicode.com/users').then((data) => {
9       console.log(data)
10      setPost(data?.data)
11    })
12  }, [])
13
14  return (
15    <div>
16      Users
17      {post.map((item, i) => {
18        return (
19          <div key={i}>
20            <p>{item?.name}</p>
21          </div>
22        )
23      })}
24    </div>
25  )
26 }
27
28 export default Users
```

ใช้เมธอด `axios.get()` เพื่อสร้าง request ไปขอข้อมูลจาก URL

จากนั้น ใช้เมธอด `then()` เพื่อรับข้อมูลไปเก็บไว้ในตัวแปร `post` ด้วย `setPost()`

การใช้งาน `axios` ร่วมกับ `useEffect` และ `useState` เพื่อดึงข้อมูลจาก API

```
1 import React, { useEffect, useState } from 'react'
2 import axios from 'axios'
3
4 function Users() {
5   const [post, setPost] = useState([])
6
7   useEffect(() => {
8     axios.get('https://jsonplaceholder.typicode.com/users').then((data) => {
9       console.log(data)
10      setPost(data?.data)
11    })
12  }, [])
13
14  return (
15    <div>
16      Users
17      {post.map((item, i) => {
18        return (
19          <div key={i}>
20            <p>{item?.name}</p>
21          </div>
22        )
23      })}
24    </div>
25  )
26 }
27
28 export default Users
```

ดูปแสดงผลข้อมูลอาเรย์ในตัวแปร `post` ด้วยเมธอด `map()` พร้อมแสดงค่า `item.name` ออกมา

การทำงาน axios ร่วมกับ useEffect และ useState

```
1 import React, { useEffect, useState } from 'react'
2 import axios from 'axios'
3
4 function Users() {
5   const [post, setPost] = useState([])
6
7   useEffect(() => {
8     axios.get('https://jsonplaceholder.typicode.com/users').then((data) => {
9       console.log(data)
10      setPost(data?.data)
11    })
12  }, [])
13
14  return (
15    <div>
16      Users
17      {post.map((item, i) => {
18        return (
19          <div key={i}>
20            <p>{item?.name}</p>
21          </div>
22        )
23      })}
24    </div>
25  )
26 }
27
28 export default Users
```

อิมพอร์ต **useEffect, useState** เพื่อใช้งาน
Hooks

อิมพอร์ต **axios** เพื่อใช้รับข้อมูลด้วยโปรโตคอล
HTTP

ใช้เมธอด **axios.get()** เพื่อสร้าง request ไปขอ
ข้อมูลจาก URL

ใช้เมธอด **then()** เพื่อรับข้อมูลไปเก็บไว้ในตัวแปร
post ด้วย **setPost()**

เปรียบเทียบการใช้งานระหว่าง `fetch` และ `axios`

fetch

```
1 const url = 'https://jsonplaceholder.typicode.com/todos'
2 const options = {
3   method: 'POST',
4   headers: {
5     Accept: 'application/json',
6     'Content-Type': 'application/json;charset=UTF-8',
7   },
8   body: JSON.stringify({
9     a: 10,
10    b: 20,
11  }),
12 }
13 fetch(url, options)
14   .then((response) => response.json())
15   .then((data) => {
16     console.log(data)
17   })
18
```

axios

```
1 const url = 'https://jsonplaceholder.typicode.com/posts'
2 const data = {
3   a: 10,
4   b: 20,
5 }
6 axios
7   .post(url, data, {
8     headers: {
9       Accept: 'application/json',
10      'Content-Type': 'application/json;charset=UTF-8',
11    },
12  })
13   .then(({ data }) => {
14     console.log(data)
15   })
16
```

เปรียบเทียบการใช้งานระหว่าง `fetch` และ `axios`

```
1 // node-fetch
2 fetch('https://your-domain.com/users/')
3   .then((res) => {
4     return res.json()
5   })
6
7 // axios
8 axios.get('https://your-
  domain.com/users/')
9   .then((response) => return response)
```

`axios` แปลงข้อมูลที่เป็น JSON
ให้เราโดยอัตโนมัติ

ส่วน `fetch` จะต้องมาทำ `response.json()` เอง

เปรียบเทียบการใช้งานระหว่าง `fetch` และ `axios`

```
1 // node-fetch
2 const handleErrors = (res) => {
3   if (!res.ok) {
4     throw Error(res.statusText)
5   }
6   return res
7 }
8 fetch('https://your-domain.com/500')
9   .then(handleErrors)
10  .then((response) => console.log('ok'))
11  .catch((error) => console.log(error))
12
```

`fetch` จัดการกับ `Error Handling` ด้วยการที่เราต้องเขียน function สำหรับ handle ขึ้นมาเอง

เปรียบเทียบการใช้งานระหว่าง `fetch` และ `axios`

`axios` จัดการกับ Error Handling ได้สะดวกกว่า โดยหากมี error จะทำการเอาจุด catch ให้เลยทันที

```
1 // axios
2 try {
3   return await axios.get('https://your-
  domain.com/500')
4 } catch (err) {
5   throw err
6 }
7
```

เปรียบเทียบการใช้งานระหว่าง `fetch` และ `axios`

```
1 // Add a request interceptor
2 axios.interceptors.request.use(
3   function (config) {
4     // Do something before request is sent
5     return config
6   },
7   function (error) {
8     // Do something with request error
9     return Promise.reject(error)
10  }
11 )
12
13 // Add a response interceptor
14 axios.interceptors.response.use(
15   function (response) {
16     // Any status code that lie within the range of 2xx cause this function to trigger
17     // Do something with response data
18     return response
19   },
20   function (error) {
21     // Any status codes that falls outside the range of 2xx cause this function to trigger
22     // Do something with response error
23     return Promise.reject(error)
24   }
25 )
```

Intercept request and response

`axios` มี Interceptor ให้อยู่ด้วย คุณสามารถปรับเปลี่ยน request ที่ได้รับมาก่อนจะโยนต่อหรือปรับเปลี่ยนข้อมูลที่จะ response ได้ก่อนที่จะทำการ response จริง







เปรียบเทียบการใช้งานระหว่าง `fetch` และ `axios`













```
1 const CancelToken = axios.CancelToken
2 const source = CancelToken.source()
3
4 axios
5   .get('/user/12345', {
6     cancelToken: source.token,
7   })
8   .catch(function (thrown) {
9     if (axios.isCancel(thrown)) {
10       console.log('Request canceled', thrown.message)
11     } else {
12       // handle error
13     }
14   })
15
16 axios.post(
17   '/user/12345',
18   {
19     name: 'new name',
20   },
21   {
22     cancelToken: source.token,
23   }
24 )
25
26 // cancel the request (the message parameter is optional)
27 source.cancel('Operation canceled by the user.')
28
```


Cancel Request

`axios` มีการ Provide การทำ Cancel request มาให้
เรียกใช้ได้

เบราว์เซอร์ที่รองรับการใช้งาน **axios**

					
Latest ✓	Latest ✓	Latest ✓	Latest ✓	Latest ✓	11 ✓

 Firefox	 Chrome	 IE	 Edge	 Safari
69  7 ✓	77  7 ✓	11  8.1 ✓	18  10 ✓	9  10.11 ✓
				10  10.12 ✓
				11  u ✓

TESTING POWERED BY
 SAUCE LABS

Final workshop

TodoList Frontend และ Backend

Link FE: <https://github.com/soncomqiq/todo-list-workshop-ep-5>

Link BE: <https://github.com/soncomqiq/to-do-list-backend-kbtg>