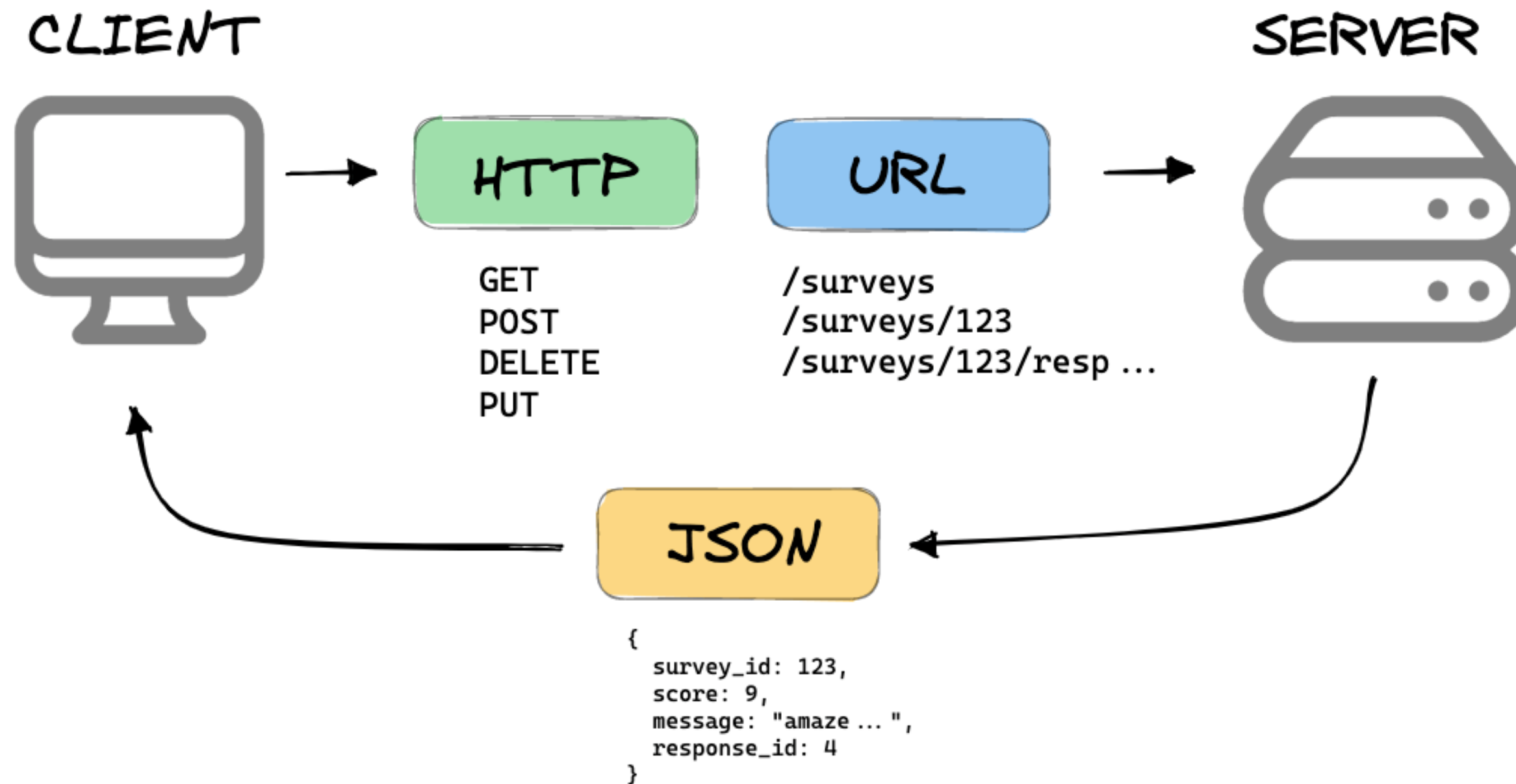




**A X ↕ O S**

# สิ่งที่ควรรู้ในส่วนขอ REST API

# REST API Flow



## HTTP Method

## HTTP Status Code

Get	ใช้สำหรับการเรียกดูข้อมูล	200 OK
Post	ใช้สำหรับเพิ่มข้อมูล	201 Created
Put	ใช้สำหรับแก้ไขข้อมูลทุก column	200 OK
Patch	ใช้สำหรับแก้ไขข้อมูลบาง column	200 OK
Delete	ใช้สำหรับลบข้อมูล	200 OK

# การส่งข้อมูลผ่าน REST API

## JSON

```
{ "title": "Hello Axios" }
```

## x-www-form-urlencoded

```
name=hoshi&age=18
```

## multipart/form-data

มักจะเอาไว้อัพโหลดไฟล์แต่ก็สามารถใช้ส่งข้อมูลแบบปกติได้เหมือนกัน

## Query params

```
http://localhost:3001/employees?id=1&name=abc
```

## Path params

```
http://localhost:3001/delete/1 ← ID
```

**A X ↕ O S**

# การติดตั้ง Axios

```
npm install axios
```

## การทำ Autocomplete

### CommonJS

```
const axios = require('axios').default;
```

### ESM

```
import axios from 'axios';
```

# การใช้งานเบื้องต้น

```
axios.get("http://localhost:3001/employees")
  .then(function (response) {
    // ถ้าทำงานสำเร็จ
    console.log(response);
  })
  .catch(function (error) {
    // ถ้าเกิด error
    console.log(error);
  })
  .finally(function () {
    // รันคำสั่งในนี้เสมอไม่ว่าจะสำเร็จหรือไม่ก็ตาม
    console.log("always executed");
  });
```



# คำสั่งที่ใช้งานทั่วไป

`axios.create(customConfig)`

`axios.get(url, config)`

`axios.post(url, data, config)`


`axios.put(url, data, config)`

`axios.patch(url, data, config)`

`axios.delete(url, config)`

`axios.interceptors.request.use`

`axios.interceptors.response.use`

`data = payload = claims` 

# axios

```
// แบบใส่ config เข้าไปใน function เลย
const getEmployees = () => {
  axios({
    method: "get",
    url: "http://localhost:3001/employees",
  })
  .then((res) => {
    setEmployeeList(res.data);
  })
  .catch((err) => {
    console.log("error: ", err);
  })
  .finally(() => {
    console.log("always executed");
  });
};
```

# GET

```
// แบบใช้ method ที่ทาง axios เตรียมไว้ให้
const getEmployees = () => {
  axios.get("http://localhost:3001/employees").then((res) => {
    setEmployeeList(res.data);
  });
};
```

# Query Params แบบที่ 1

```
const getEmployeeById = () => {  
  // แบบใส่ query params เข้าไปใน url  
  axios.get(`http://localhost:3001/employees?id=${id}`).then((res) => {  
    setEmployeeList(res.data);  
  });  
};
```

# Query Params แบบที่ 2

```
const getEmployeeById = () => {  
  // แบบใช้ options ใน axios  
  const options = {  
    params: {  
      id: id,  
    },  
  };  
  
  axios.get(`http://localhost:3001/employees`, options).then((res) => {  
    setEmployeeList(res.data);  
  });  
};
```

# POST ส่งข้อมูล JSON

```
const addEmployee1 = (event) => {  
  // ส่งข้อมูลแบบ JSON ไปที่ Server  
  const payload = {  
    name: name,  
    age: age,  
    country: country,  
    position: position,  
    wage: wage,  
  };  
  
  axios.post("http://localhost:3001/create", payload).then(() => {  
    setEmployeeList([  
      ...employeeList,  
      {  
        name: name,  
        age: age,  
        country: country,  
        position: position,  
        wage: wage,  
      },  
    ])  
  });  
}
```

# POST ส่งข้อมูล form-urlencoded

```
const addEmployee1 = (event) => {  
  // ส่งข้อมูลแบบ application/x-www-form-urlencoded โดยใช้ฟังก์ชัน URLSearchParams  
  const payload = new URLSearchParams({  
    name: name,  
    age: age,  
    country: country,  
    position: position,  
    wage: wage,  
  });  
  
  axios.post("http://localhost:3001/create", payload).then(() => {  
    setEmployeeList([  
      ...employeeList,  
      {  
        name: name,  
        age: age,  
        country: country,  
        position: position,  
        wage: wage,  
      },  
    ]);  
  });  
}
```

# POST ส่งข้อมูล form-data

```
const addEmployee1 = (event) => {  
  // ส่งข้อมูลแบบ multipart/form-data โดยใช้ฟังก์ชัน FormData  
  const payload = new FormData();  
  payload.append('name', name)  
  payload.append('age', age)  
  payload.append('country', country)  
  payload.append('position', position)  
  payload.append('wage', wage)  
  
  axios.post("http://localhost:3001/create", payload).then(() => {  
    setEmployeeList([  
      ...employeeList,  
      {  
        name: name,  
        age: age,  
        country: country,  
        position: position,  
        wage: wage,  
      },  
    ])  
  });  
}
```



# Auto Serialization to FormData, etc

```
const addEmployee1 = (event) => {  
  // ส่งข้อมูลแบบ multipart/form-data โดยให้ axios แปลง JSON เป็น FormData ให้อัตโนมัติ  
  const payload = {  
    name: name,  
    age: age,  
    country: country,  
    position: position,  
    wage: wage,  
  };  
  
  axios  
    .post("http://localhost:3001/create", payload, {  
      headers: {  
        "Content-Type": "multipart/form-data",  
      },  
    })  
    .then(() => {  
      setEmployeeList([  
        ...employeeList,  
        {  
          name: name,  
          age: age,  
          country: country,  
          position: position,  
          wage: wage,  
        },  
      ]);  
    });  
}
```


# PUT & PATCH

```
const updateEmployeeWage = (id) => {  
  // put หรือ patch ก็ได้ขึ้นอยู่กับว่า server ตั้งไว้เป็น method อะไร  
  axios  
    .put("http://localhost:3001/update", { wage: newWage, id: id })  
    .then((res) => {  
      setEmployeeList(  
        employeeList.map((val) => {  
          return val.id === id  
            ? {  
              id: val.id,  
              name: val.name,  
              country: val.country,  
              age: val.age,  
              position: val.position,  
              wage: newWage,  
            }  
            : val;  
        })  
      );  
    });  
};
```

# DELETE

Path Params

```
const deleteEmployee = (id) => {  
  axios.delete(`http://localhost:3001/delete/${id}`).then((res) => {  
    setEmployeeList(  
      employeeList.filter((val) => {  
        return val.id !== id;  
      })  
    );  
  });  
};
```



# Axios Instance

เราสามารถสร้าง instance เพื่อกำหนดค่า config เองได้ เช่น headers, baseURL

ประโยชน์ : เพื่อให้สามารถเรียก instance ที่สร้างพร้อมกับค่า config ไปใช้ได้เลยโดยไม่ต้องกำหนดค่า config ใหม่ทุกครั้งที่เราเรียกใช้ axios get, post, put, patch, delete

# Axios Instance

```
// Create Instance
const instance = axios.create({
  baseURL: process.env.REACT_APP_API_URL,
  timeout: 5000, // 5000ms => 5s
  withCredentials: true, // คำสั่งที่กำหนดให้ axios สามารถส่ง cookie ไปที่ server ได้
});
```

ข้อมูลเพิ่มเติม: <https://axios-http.com/docs/instance>

ข้อมูลเพิ่มเติม: [https://axios-http.com/docs/req\\_config](https://axios-http.com/docs/req_config)

# Global Variables

เป็นการกำหนดค่า config ให้กับ axios ทุกตัวที่ถูกเรียกใช้งาน

```
// Global axios
axios.defaults.baseURL = process.env.REACT_APP_API_URL;
axios.defaults.headers.common["Authorization"] = `Bearer ${token}`;
axios.defaults.headers.post["Content-Type"] = "application/x-www-form-urlencoded";
```

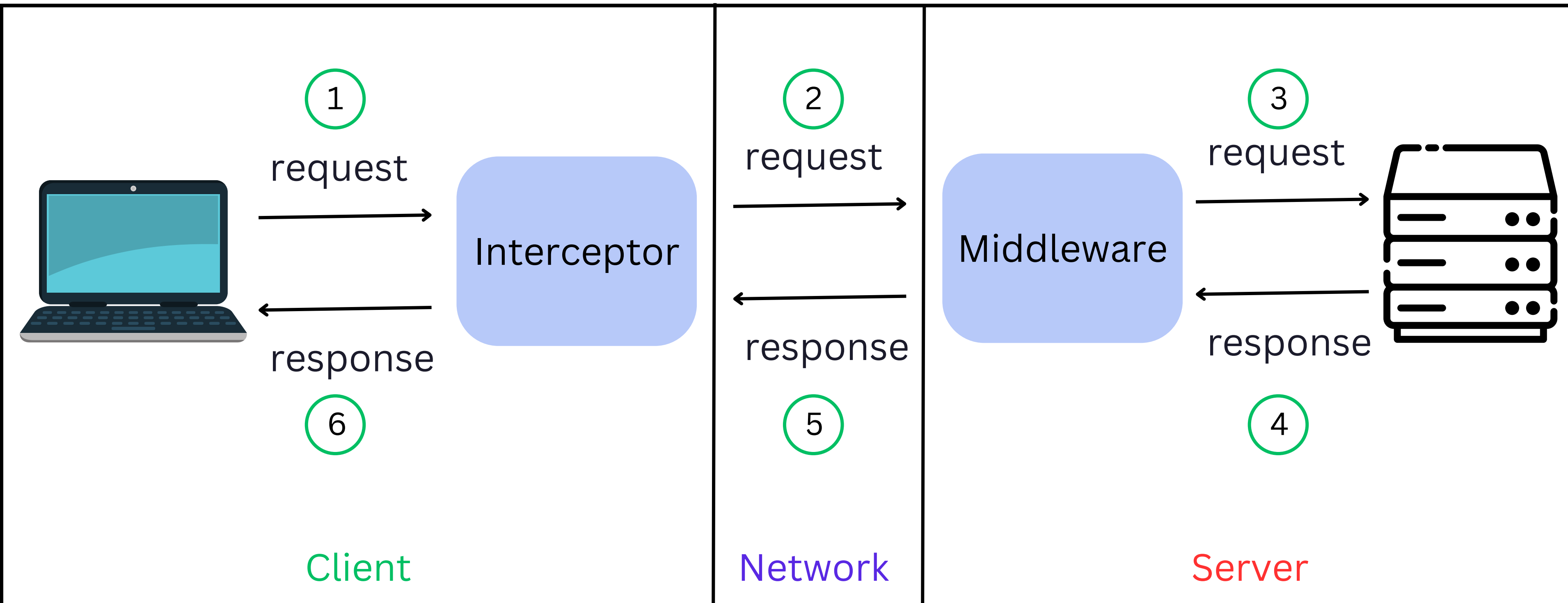
ข้อมูลเพิ่มเติม: [https://axios-http.com/docs/config\\_defaults](https://axios-http.com/docs/config_defaults)

# Interceptor

Interceptor ของ axios นั้นเป็น middleware ที่ดัก request และ response

ประโยชน์ : เพื่อให้เราสามารถเปลี่ยนแปลง config ก่อน Request หรือ แก้ไข response

# Interceptor





# Request Interceptor

```
// เพิ่ม request interceptor
axios.interceptors.request.use(
  (config) => {
    // ทำการเช็ค config บางอย่างก่อนที่จะส่ง request เช่น เช็ค token ก่อนส่ง request
    const apiKey = process.env.REACT_APP_API_KEY;
    if (apiKey) {
      config.headers["X-API-Key"] = apiKey; // ส่งไปใน header ในชื่อ X-API-Key
    }
    const token = localStorage.getItem("token");
    if (token) {
      config.headers["Authorization"] = `Bearer ${token}`; // ส่งไปใน header ในชื่อ Authorization
    }
    return config;
  },
  (error) => {
    return Promise.reject(error);
  }
);
```

# Response Interceptor

```
// เพิ่ม response interceptor
axios.interceptors.response.use(
  (response) => {
    // หากได้รับ status code ที่อยู่ในช่วง 2xx จะเข้ามาทำงานตรงนี้
    return response;
  },
  (error) => {
    // หากได้รับ status code ที่ไม่ได้ในช่วง 2xx จะเข้ามาทำงานตรงนี้
    if (error.response && error.response.status === 401) {
      console.log("Unauthorized");
      return;
    }
    return Promise.reject(error);
  }
);
```

# การจัดการ Error

```
// การดัก Error ใน axios
const getEmployees = () => {
  axios
    .get("http://localhost:3001/employees")
    .then((res) => {
      console.log(res.data);
    })
    .catch((err) => {
      if (err.response) {
        // ส่ง request ไปแล้วได้รับ response status code ที่ไม่ได้อยู่ในช่วง 2xx
        console.log(err.response.data);
        console.log(err.response.status);
        console.log(err.response.headers);
      } else if (err.request) {
        // ส่ง request แต่ไม่ได้รับ response กลับมา
        console.log(err.request);
      } else {
        // หากไม่เข้าเงื่อนไขไหนเลยจะเข้า case นี้
        console.log("Error:", err.message);
        console.log(err.config);
        console.log(err.toJSON()); // วัตถุข้อมูลเพิ่มเติมเกี่ยวกับ HTTP error
      }
    });
};
```

# axios ไม่ใช้ async/await

```
// axios ไม่ใช้ async/await
const getEmployees = () => {
  axios
    .get("http://localhost:3001/employees")
    .then((res) => {
      console.log("res", res);
    })
    .catch((err) => {
      console.log("error", err);
    });
  console.log("End");
};
```

## Output

End

res ▶ {data: Array(13), status: 200, statusText: 'OK', headers: AxiosHeaders, config: {...}, ...}

# axios ใช้ async/await

```
// axios ใช้ async/await
const getEmployees = async () => {
  await axios
    .get("http://localhost:3001/employees")
    .then((res) => {
      console.log("res", res);
    })
    .catch((err) => {
      console.log("error", err);
    });
  console.log("End");
};
```

## Output

```
res ▶ {data: Array(13), status: 200, statusText: 'OK', headers: AxiosHeaders, config: {...}, ...}
End
```

# axios ใช้ async/await

แต่มีการเรียกใช้ฟังก์ชัน 2 ฟังก์ชันในระดับเดียวกันโดยไม่ใช้ async/await

```
const getEmployees1 = async () => {  
  await axios  
    .get("http://localhost:3001/employees")  
    .then((res) => {  
      console.log("res", res);  
    })  
    .catch((err) => {  
      console.log("error", err);  
    });  
  console.log("End");  
};
```

```
const displayStart = () => {  
  console.log("Starting...");  
};
```

```
useEffectOnce(() => {  
  getEmployees1()  
  displayStart()  
});
```

## Output

```
Starting...  
res ▶ {data: Array(13), status: 200, statusText: 'OK', headers: AxiosHeaders, config: {...}, ...}  
End
```

# axios ใช้ async/await

แต่มีการเรียกใช้ฟังก์ชัน 2 ฟังก์ชันในระดับเดียวกันโดยใช้ async/await

```
const getEmployees1 = async () => {  
  await axios  
    .get("http://localhost:3001/employees")  
    .then((res) => {  
      console.log("res", res);  
    })  
    .catch((err) => {  
      console.log("error", err);  
    });  
  console.log("End");  
};
```

```
const displayStart = () => {  
  console.log("Starting...");  
};
```

```
useEffectOnce(async () => {  
  await getEmployees1()  
  await displayStart()  
});
```

## Output

```
res ▶ {data: Array(13), status: 200, statusText: 'OK', headers: AxiosHeaders, config: {...}, ...}  
End  
Starting...
```



# Concurrency ของ axios ไม่ใช้ async/await

```
// Concurrency ของ axios ไม่ใช้ async/await
const getEmployees = () => {
  const reqEmployee1 = axios.get("http://localhost:3001/employees");
  const reqEmployee2 = axios.get("http://localhost:3001/employees?id=11");

  Promise
    .all([reqEmployee1, reqEmployee2])
    .then(
      axios.spread((res1, res2) => {
        console.log("res1", res1);
        console.log("res2", res2);
      })
    )
    .catch((err) => {
      console.error(err);
    });

  console.log("End")
};
```

## Output

```
End
res1 ▶ {data: Array(13), status: 200, statusText: 'OK', headers: AxiosHeaders, config: {...}, ...}
res2 ▶ {data: Array(1), status: 200, statusText: 'OK', headers: AxiosHeaders, config: {...}, ...}
```



# Concurrency ของ axios ใช้ async/await

```
// Concurrency ของ axios ใช้ async/await
const getEmployees = async () => {
  const reqEmployee1 = axios.get("http://localhost:3001/employees");
  const reqEmployee2 = axios.get("http://localhost:3001/employees?id=11");

  await Promise.all([reqEmployee1, reqEmployee2])
    .then(
      axios.spread((res1, res2) => {
        console.log("res1", res1);
        console.log("res2", res2);
      })
    )
    .catch((err) => {
      console.error(err);
    });

  console.log("End");
};
```

## Output

```
res1 ▶ {data: Array(13), status: 200, statusText: 'OK', headers: AxiosHeaders, config: {...}, ...}
res2 ▶ {data: Array(1), status: 200, statusText: 'OK', headers: AxiosHeaders, config: {...}, ...}
End
```

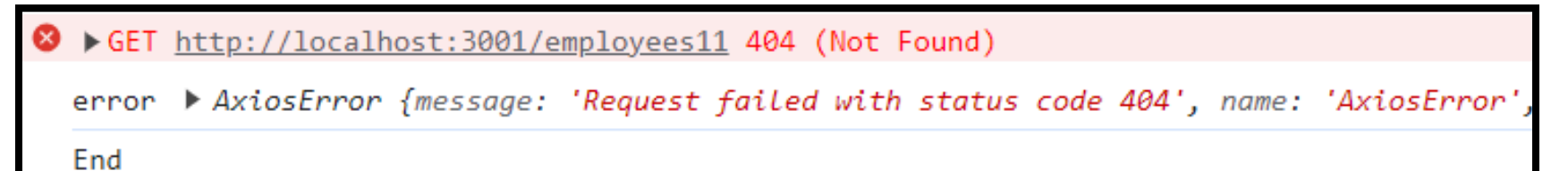
# Try & Catch ไม่ใช้ throw

```
// Concurency ของ axios ใช้ async/await และ try,catch
const getEmployees = async () => {
  try {
    const reqEmployee1 = axios.get("http://localhost:3001/employees11");
    const reqEmployee2 = axios.get("http://localhost:3001/employees?id=11");

    await Promise.all([reqEmployee1, reqEmployee2])
      .then(
        axios.spread((res1, res2) => {
          console.log("res1", res1);
          console.log("res2", res2);
        })
      )
      .catch((err) => {
        console.log("error", err);
        // throw err
      });

    console.log("End");
  } catch (error) {
    console.log("error2", error);
  }
};
```

## Output



A screenshot of a web browser's developer console. The top line shows a red error icon, a red arrow pointing to the right, and the text "GET http://localhost:3001/employees11 404 (Not Found)". Below this, there is a line of text: "error ▶ AxiosError {message: 'Request failed with status code 404', name: 'AxiosError'". At the bottom of the console, the word "End" is visible.

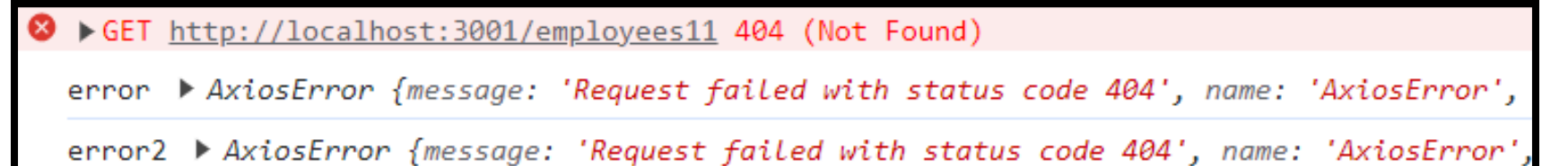
# Try & Catch ใช้ throw

```
// Concurrency ของ axios ใช้ async/await และ try,catch
const getEmployees = async () => {
  try {
    const reqEmployee1 = axios.get("http://localhost:3001/employees11");
    const reqEmployee2 = axios.get("http://localhost:3001/employees?id=11");

    await Promise.all([reqEmployee1, reqEmployee2])
      .then(
        axios.spread((res1, res2) => {
          console.log("res1", res1);
          console.log("res2", res2);
        })
      )
      .catch((err) => {
        console.log("error", err);
        throw err
      });

    console.log("End");
  } catch (error) {
    console.log("error2", error);
  }
};
```

## Output



The screenshot shows the browser console with two error messages. The first message is a red line indicating a GET request to `http://localhost:3001/employees11` failed with a 404 status (Not Found). The second message is a blue line showing an `AxiosError` object with the message `'Request failed with status code 404'` and name `'AxiosError'`. The third message is a blue line showing another `AxiosError` object with the same message and name, representing the second failed request.

```
✖ ▶ GET http://localhost:3001/employees11 404 (Not Found)
error ▶ AxiosError {message: 'Request failed with status code 404', name: 'AxiosError',
error2 ▶ AxiosError {message: 'Request failed with status code 404', name: 'AxiosError',
```