

Lab8

---

# Патерни, які я використовував

---

Дмитро Тукалевський

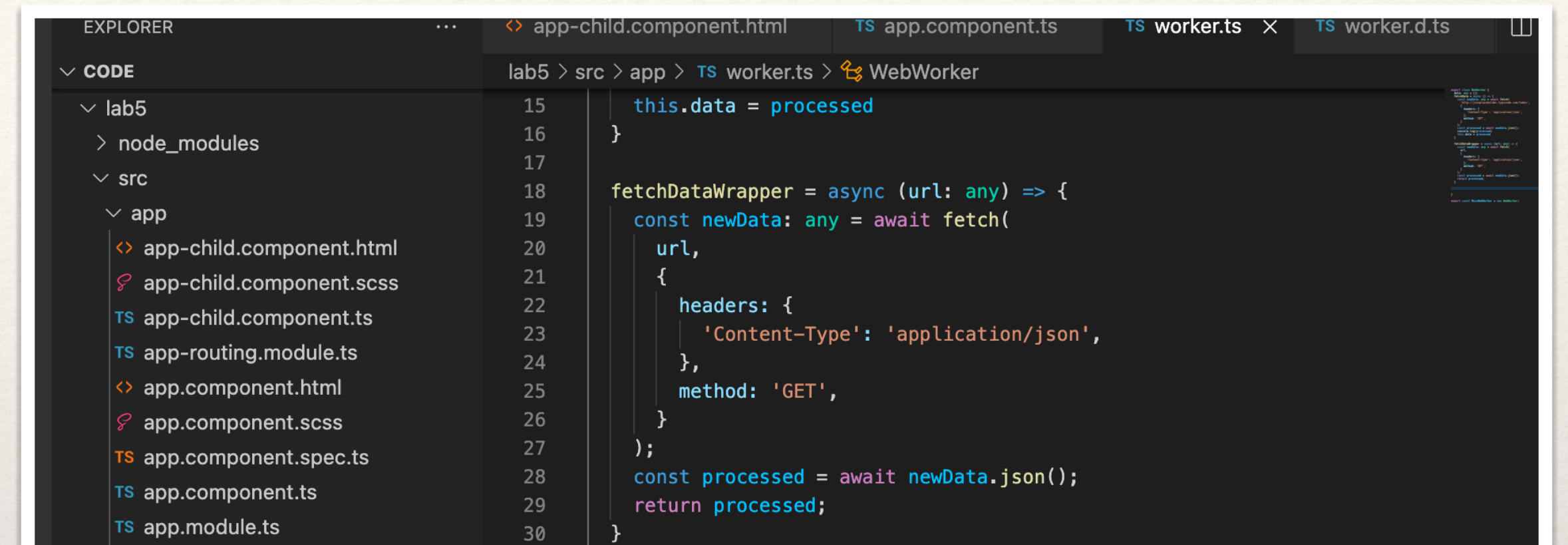


# Паттерн 1



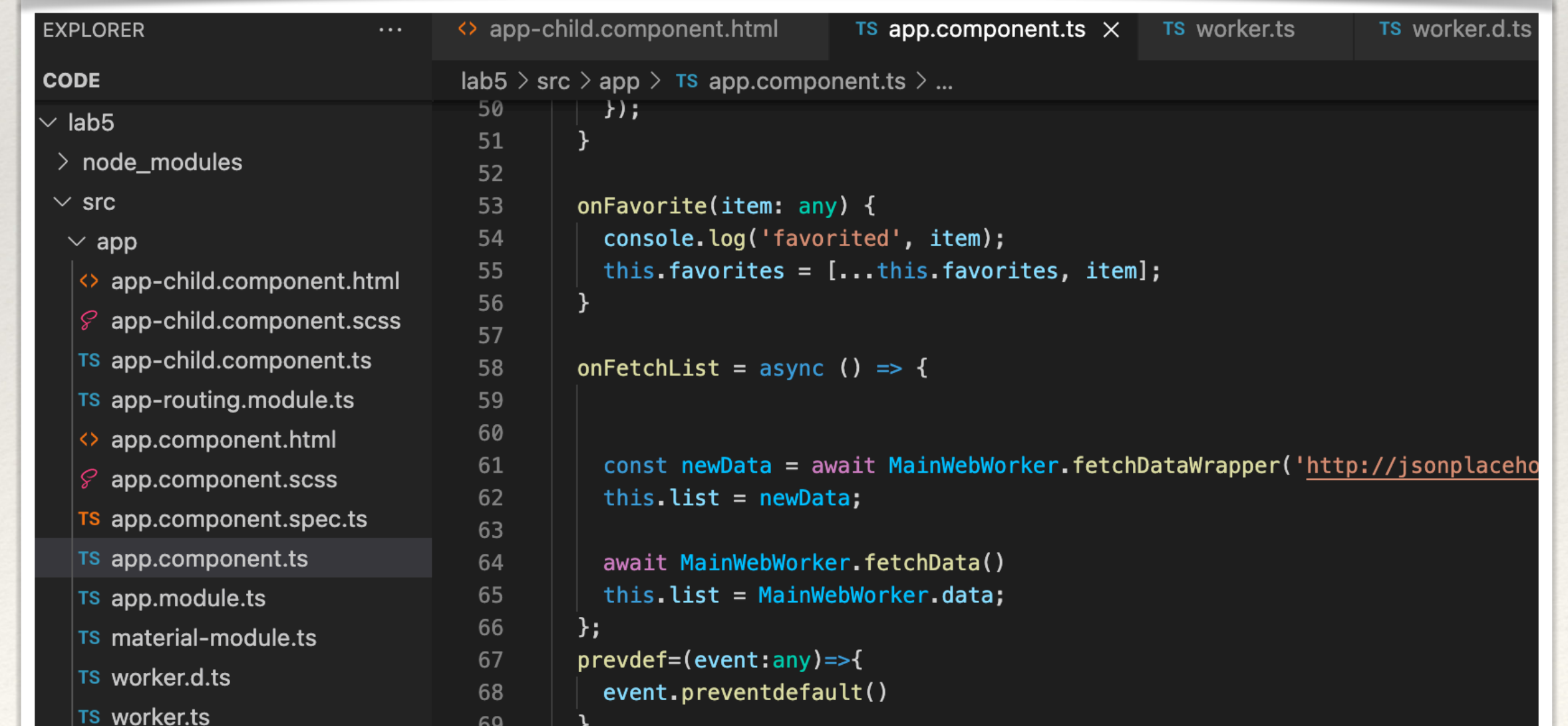
# Wrapper

- ❖ **Wrapper** - обертає код та додає йому додатковий функціонал.
- ❖ `fetchDataWrapper` отримує `url` з функції `onFetchList`, підставляє отримане посилання і обертає її в `get` запит.



```
EXPLORER  ...  app-child.component.html  TS app.component.ts  TS worker.ts  TS worker.d.ts
CODE
lab5
├── node_modules
└── src
    └── app
        ├── app-child.component.html
        ├── app-child.component.scss
        ├── TS app-child.component.ts
        ├── TS app-routing.module.ts
        ├── app.component.html
        ├── app.component.scss
        ├── TS app.component.spec.ts
        ├── TS app.component.ts
        └── TS app.module.ts

lab5 > src > app > TS worker.ts > WebWorker
15  this.data = processed
16  }
17
18  fetchDataWrapper = async (url: any) => {
19    const newData: any = await fetch(
20      url,
21      {
22        headers: {
23          'Content-Type': 'application/json',
24        },
25        method: 'GET',
26      }
27    );
28    const processed = await newData.json();
29    return processed;
30  }
```



```
EXPLORER  ...  app-child.component.html  TS app.component.ts  TS worker.ts  TS worker.d.ts
CODE
lab5 > src > app > TS app.component.ts > ...
50  });
51  }
52
53  onFavorite(item: any) {
54    console.log('favorited', item);
55    this.favorites = [...this.favorites, item];
56  }
57
58  onFetchList = async () => {
59
60    const newData = await MainWebWorker.fetchDataWrapper('http://jsonplaceholder
61    this.list = newData;
62
63    await MainWebWorker.fetchData()
64    this.list = MainWebWorker.data;
65  };
66
67  prevdef=(event:any)=>{
68    event.preventDefault()
69  }
```



Паттерн 2 та 3



# Singleton & Proxy

- ❖ **Singleton** - використання єдиного екземпляра класу для реалізації функціоналу у системі.
- ❖ `worker.ts` в якому знаходиться екземпляр класу, який пропускає через себе всі запити.
- ❖ Також використання цього класу для запиту реалізує паттерн **Proxy** відправляється запит як через посередника через цього екземпляра класу.
- ❖ Посередник може модифікувати запит як завгодно.

worker.ts — Code

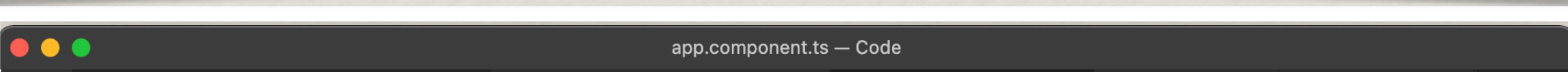
EXPLORER

CODE

- lab5
  - node\_modules
  - src
    - app
      - app-child.component.html
      - app-child.component.scss
      - app-child.component.ts
      - app-routing.module.ts
      - app.component.html
      - app.component.scss
      - app.component.spec.ts
      - app.component.ts
      - app.module.ts
      - material-module.ts
      - worker.d.ts
      - worker.ts
    - assets

lab5 > src > app > TS worker.ts > WebWorker > fetchData

```
1 export class WebWorker {
2   data: any = [];
3   fetchData = async () => {
4     const newData: any = await fetch(
5       'http://jsonplaceholder.typicode.com/todos',
6       {
7         headers: {
8           'Content-Type': 'application/json',
9         },
10        method: 'GET',
11      }
12    );
13    const processed = await newData.json();
14    console.log(processed)
15    this.data = processed
16  }
17 }
18 }
19
20 export const MainWebWorker = new WebWorker;
```



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure: 'lab5' contains 'node\_modules', 'src' (which contains 'app'). The 'app' directory is selected. The main editor area is titled 'app.component.ts — Code' and shows the following TypeScript code:

```
1 import { Component, Inject } from '@angular/core';
2 import { MainWebWorker } from './worker'
3 import {
4   MatDialog,
5   MatDialogRef,
```

```
TS app.component.spec.ts 55
TS app.component.ts 56
TS app.module.ts 57
TS material-module.ts 58
TS worker.d.ts 59
TS worker.ts 60
> assets 61
> environments 62
63
```

```
onFetchList = async () => {
  await MainWebWorker.fetchData()
  this.list = MainWebWorker.data;
};
prevdef=(event:any)=>{
  event.preventDefault()
}
```



Дякую за увагу