** HTTP calls **

- Provide HTTP module/providers in the app config using provideHttpClient()
- inject the HttpClient service
- Use the http methods.

Step 1: In app.config.ts, add:

```
provideHttpClient(),  inside providers: [ .... ],
```

Step 2: inject it inside the service.ts we want to use.

i.e. inside the class:

```
http = inject(HttpClient);
          ↳ provides get, push, patch ...
```

Step 3: Use the client to do Http calls.

eg:
```
getTodosFromApi( ) {
    const url= ` .... `
    return this.http.get<Array<Todo>>(url);
}
```
returns an observable

Step4: Call the function in our component.

eg, from todos:

Inside the class component:

```
ngOnInit(): void {        accessing service class
                          function to get todos
    this.todoService.getTodosFromApi()
If error,       .pipe(
pipe it             catchError((err) => {
to do something         console.log(err);
                        throw err;
                    })
to call the     ).subscribe((todos) => {
Api and             this.todoItems.set(todos);
get response    });
    };
```

* Note:
The above method uses .subscribe which is used when the observable (data/response) is a stream and is persistant. If not, we can also use .toPromise()/ firstValueFrom().

Using firstValueFrom():

```
import { firstValueFrom } from 'rxjs';

async getUser() {
    try {                                                         url
        const res = await firstValueFrom(this.http.get('api/user'));
        :
        do something with res
    } catch(err) {
        console.error('Error:', err);
    }
}
```

We can also use .subscribe() like:

```
this.http.get('api/user').subscribe( {
    next: (res) => {
        :
    },
    error: (err) => console. - . . ,
    complete: () => console . . .
});
```

* Note:
Promises don't work with a stream of data, they will get one response and close the connection.