Practical observable usage

Contents

Here are some examples of domains in which observables are particularly useful.

Type-ahead suggestions

Observables can simplify the implementation of type-ahead suggestions. Typically, a type-ahead has to do a series of separate tasks:

- Listen for data from an input.
- Trim the value (remove whitespace) and make sure it's a minimum length.
- Debounce (so as not to send off API requests for every keystroke, but instead wait for a break in keystrokes).
- Don't send a request if the value stays the same (rapidly hit a character, then backspace, for instance).
- Cancel ongoing AJAX requests if their results will be invalidated by the updated results.

Writing this in full JavaScript can be quite involved. With observables, you can use a simple series of RxJS operators:

Typeahead

```
import { fromEvent } from 'rxjs';
import { ajax } from 'rxjs/ajax';
import { debounceTime, distinctUntilChanged, filter, map, switchMap } from
'rxjs/operators';


const searchBox = document.getElementById('search-box');

const typeahead = fromEvent(searchBox, 'input').pipe(
  map((e: KeyboardEvent) => (e.target as HTMLInputElement).value),
  filter(text => text.length > 2),
  debounceTime(10),
  distinctUntilChanged(),
  switchMap(() => ajax('/api/endpoint'))
);

typeahead.subscribe(data => {
```

```
 // Handle the data from the API
});
```

Exponential backoff

Exponential backoff is a technique in which you retry an API after failure, making the time in between retries longer after each consecutive failure, with a maximum number of retries after which the request is considered to have failed. This can be quite complex to implement with promises and other methods of tracking AJAX calls. With observables, it is very easy:

Exponential backoff

```
import { pipe, range, timer, zip } from 'rxjs';
import { ajax } from 'rxjs/ajax';
import { retryWhen, map, mergeMap } from 'rxjs/operators';

function backoff(maxTries, ms) {
 return pipe(
   retryWhen(attempts => zip(range(1, maxTries), attempts)
     .pipe(
       map(([i]) => i * i),
       mergeMap(i =>  timer(i * ms))
     )
   )
 );
}

ajax('/api/endpoint')
  .pipe(backoff(3, 250))
  .subscribe(data => handleData(data));

function handleData(data) {
  // ...
}
```

Version 10.0.10-local+sha.b32126c335.