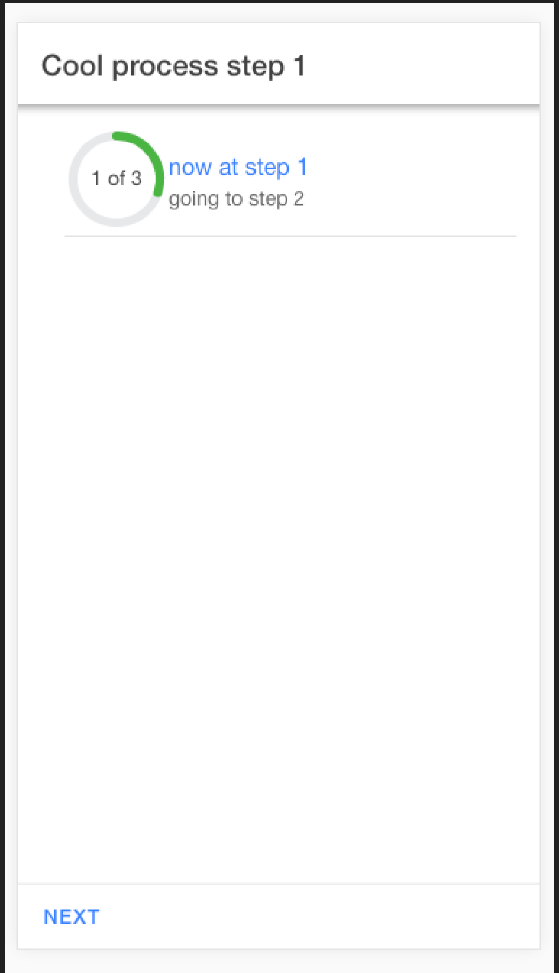
Reusable Components and rxjs Observables in Ionic-Angular

Thou Shall not duplicate code, for it is sin to the software gods

Why write reusable components?

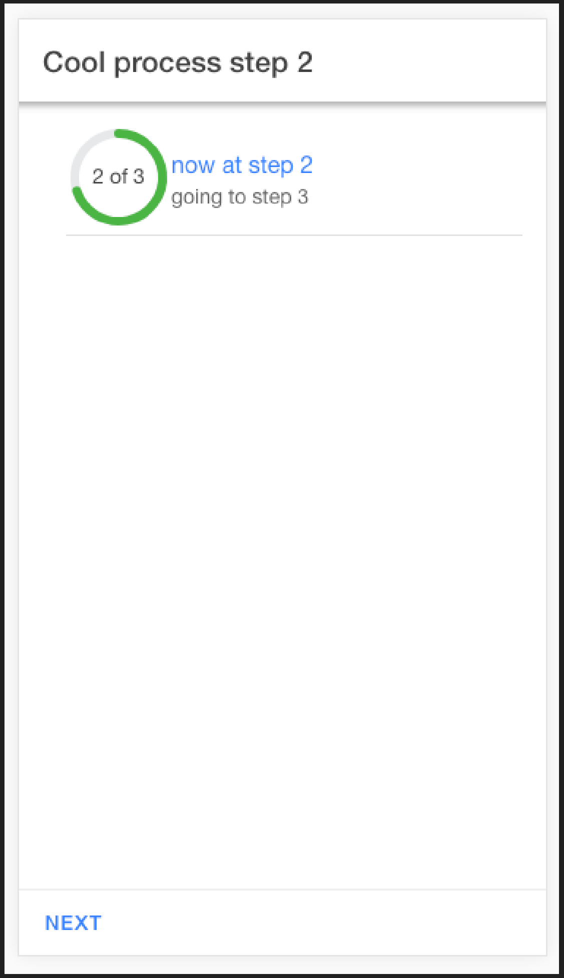
- Apps generally repeat layouts
- Save yourself a ton of headache while debugging(if you need to make changes you make them in one place)
- Ability to extend functionality based on use cases or new requirements when app grows

Lets actually write one !


Cool process step 1

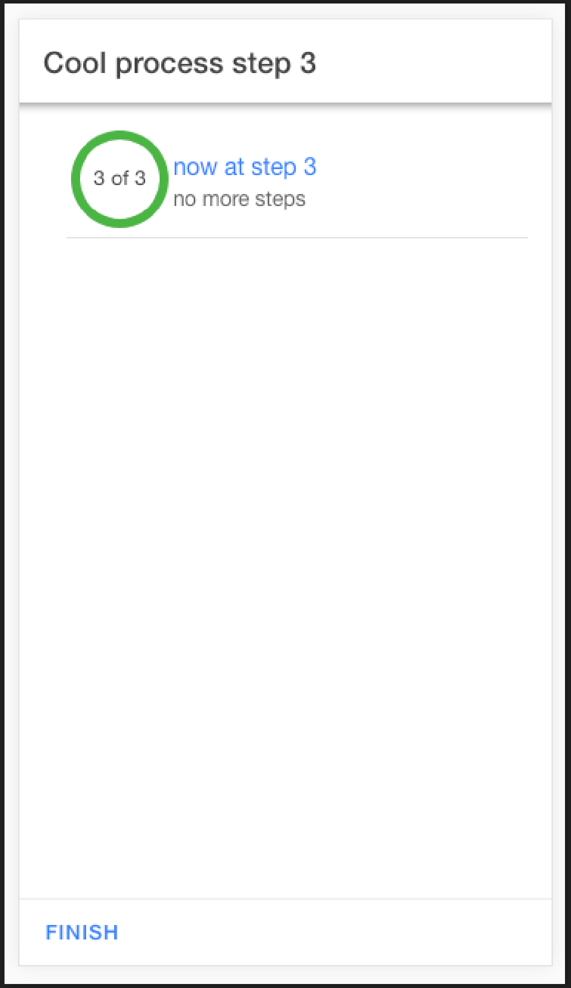
1 of 3 now at step 1
going to step 2

NEXT


Cool process step 2

2 of 3 now at step 2
going to step 3

NEXT


Cool process step 3

3 of 3 now at step 3
no more steps

FINISH

Rxjs Observables with Reusable Components

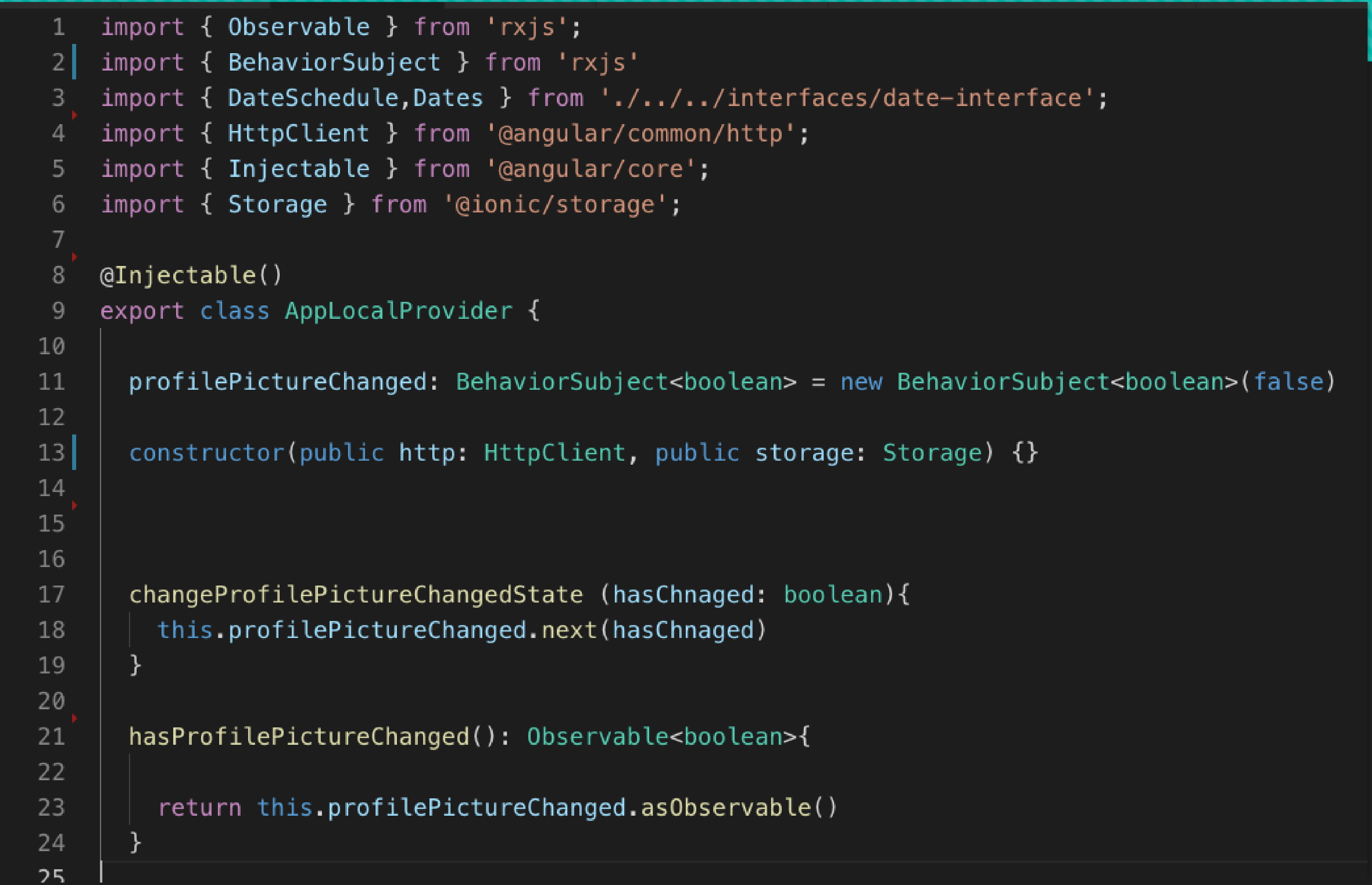
- What do Observables
  - Observables watch for state changes (to put it as simple as possible)

  Scenario

  Updating a user profile photo that lives inside a reusable component that is being called. From a side menu and profile screen

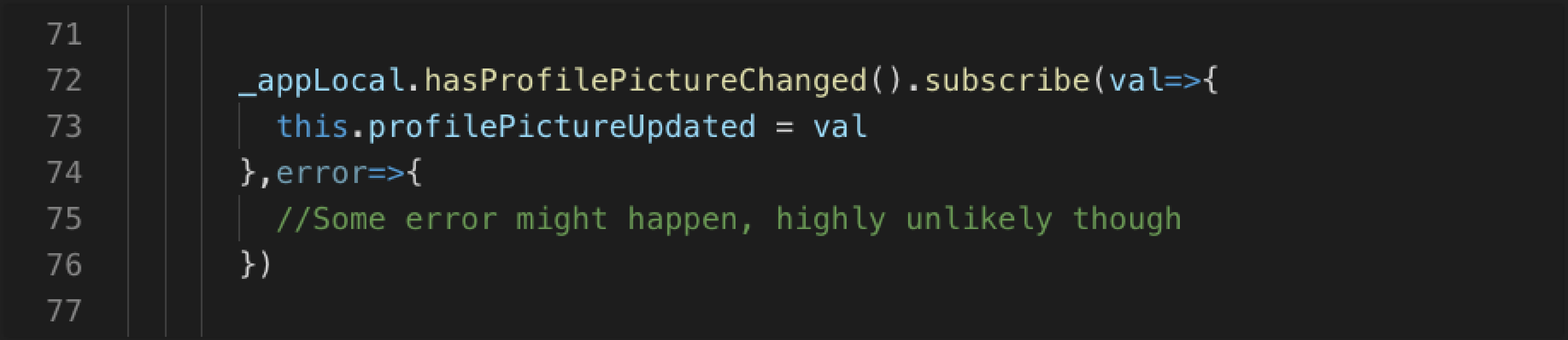
  Possible solution ??? (any takers)

Doing is it with Rxjs observables



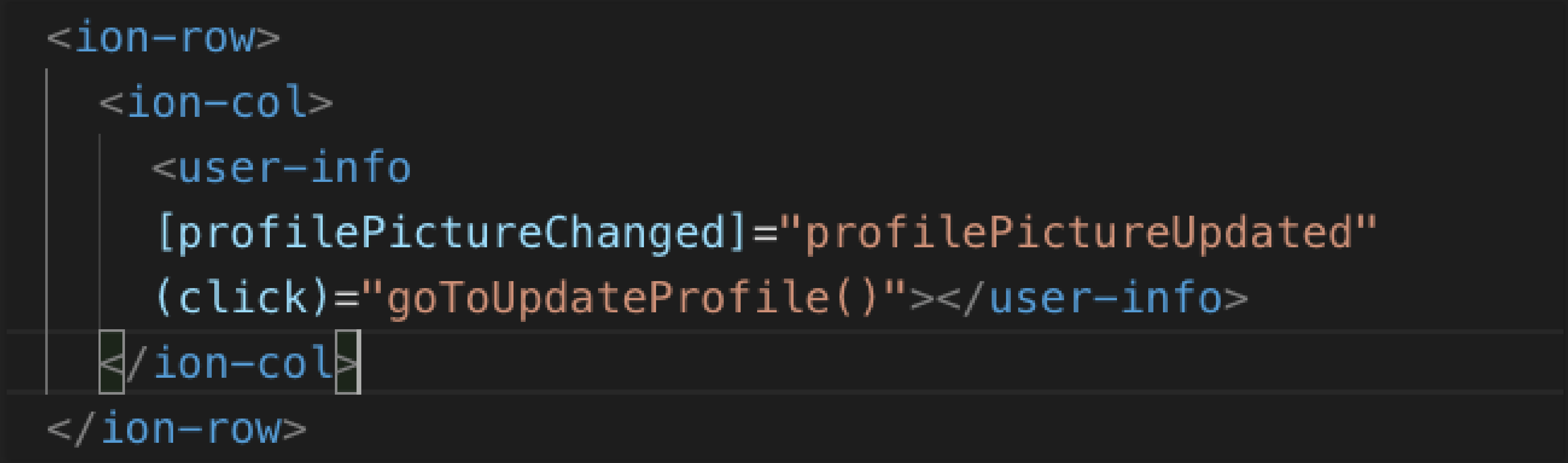
```
import { Observable } from 'rxjs';
import { BehaviorSubject } from 'rxjs'
import { DateSchedule,Dates } from './../../interfaces/date-interface';
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Storage } from '@ionic/storage';

@Injectable()
export class AppLocalProvider {

  profilePictureChanged: BehaviorSubject<boolean> = new BehaviorSubject<boolean>(false)

  constructor(public http: HttpClient, public storage: Storage) {}



  changeProfilePictureChangedState (hasChnaged: boolean){
    this.profilePictureChanged.next(hasChnaged)
  }

  hasProfilePictureChanged(): Observable<boolean>{

    return this.profilePictureChanged.asObservable()
  }

```

Cont’d



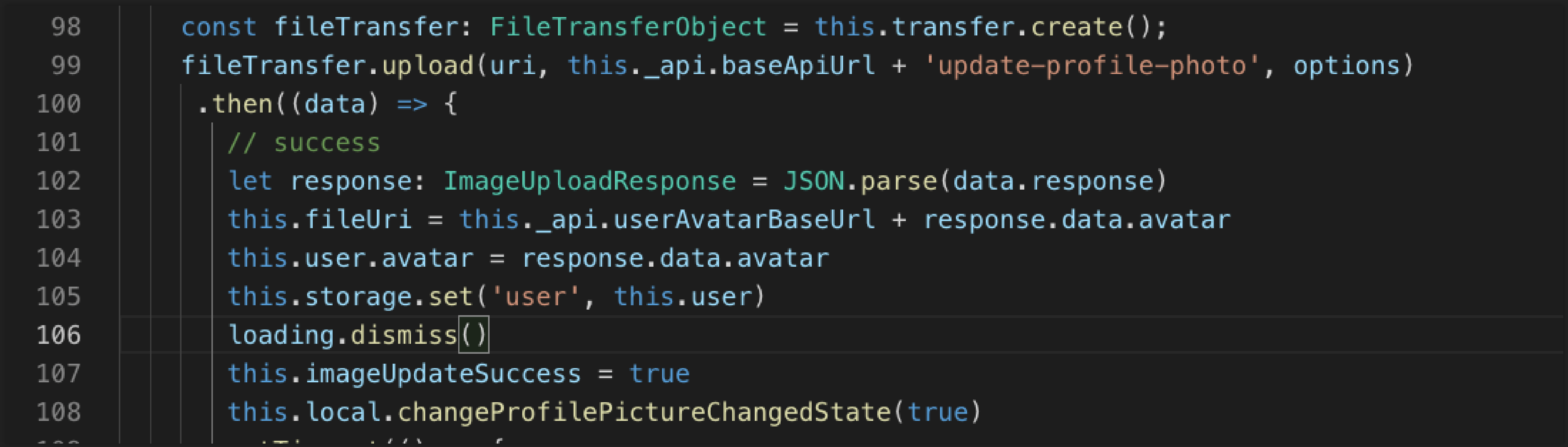
```

_appLocal.hasProfilePictureChanged().subscribe(val=>{
  this.profilePictureUpdated = val
},error=>{
  //Some error might happen, highly unlikely though
})

```

Cont’d



```
<ion-row>
  <ion-col>
    <user-info
    [profilePictureChanged]="profilePictureUpdated"
    (click)="goToUpdateProfile()"></user-info>
  </ion-col>
</ion-row>
```

Changing the state



```
    const fileTransfer: FileTransferObject = this.transfer.create();
    fileTransfer.upload(uri, this._api.baseApiUrl + 'update-profile-photo', options)
     .then((data) => {
       // success
       let response: ImageUploadResponse = JSON.parse(data.response)
       this.fileUri = this._api.userAvatarBaseUrl + response.data.avatar
       this.user.avatar = response.data.avatar
       this.storage.set('user', this.user)
       loading.dismiss()
       this.imageUpdateSuccess = true
       this.local.changeProfilePictureChangedState(true)
```