AngularFire2 Version 4 - Authentication Service

May 8, 2017 by Jeff Delaney

The latest release of AngularFire2 (v4), has introduced several major changes that affect past lessons posted on this site. In a nutshell, the new version breaks auth and database into their own modules, and also simplifies the API to align more closely with the Firebase Web API. I think these are very positive changes for the package, BTW.

The service below has been completely rewritten work with Angular4 and AngularFire2. Make sure to check out the official **AngularFire2 guide** for more background on this update.

Gist

```
import { Injectable } from '@angular/core';
import { AngularFireDatabaseModule, AngularFireDatabase, FirebaseListObservable
import { AngularFireAuth } from 'angularfire2/auth';
import { Router } from "@angular/router";
import * as firebase from 'firebase';


@Injectable()
export class AuthService {

  authState: any = null;

  constructor(private afAuth: AngularFireAuth,
              private db: AngularFireDatabase,
              private router:Router) {

            this.afAuth.authState.subscribe((auth) => {
              this.authState = auth
            });
          }

  // Returns true if user is logged in
  get authenticated(): boolean {
```

```
// Returns true if user is logged in
get authenticated(): boolean {
  return this.authState !== null;
}

// Returns current user data
get currentUser(): any {
  return this.authenticated ? this.authState : null;
}

// Returns
get currentUserObservable(): any {
  return this.afAuth.authState
}

// Returns current user UID
get currentUserId(): string {
  return this.authenticated ? this.authState.uid : '';
}

// Anonymous User
get currentUserAnonymous(): boolean {
  return this.authenticated ? this.authState.isAnonymous : false
}

// Returns current user display name or Guest
get currentUserDisplayName(): string {
  if (!this.authState) { return 'Guest' }
  else if (this.currentUserAnonymous) { return 'Anonymous' }
  else { return this.authState['displayName'] || 'User without a Name' }
}

//// Social Auth ////

githubLogin() {
  const provider = new firebase.auth.GithubAuthProvider()
  return this.socialSignIn(provider);
}

googleLogin() {
  const provider = new firebase.auth.GoogleAuthProvider()
  return this.socialSignIn(provider);
}

facebookLogin() {
  const provider = new firebase.auth.FacebookAuthProvider()
```

```
facebookLogin() {
  const provider = new firebase.auth.FacebookAuthProvider()
  return this.socialSignIn(provider);
}

twitterLogin(){
  const provider = new firebase.auth.TwitterAuthProvider()
  return this.socialSignIn(provider);
}

private socialSignIn(provider) {
  return this.afAuth.auth.signInWithPopup(provider)
    .then((credential) =>  {
        this.authState = credential.user
        this.updateUserData()
    })
    .catch(error => console.log(error));
}


//// Anonymous Auth ////

anonymousLogin() {
  return this.afAuth.auth.signInAnonymously()
  .then((user) => {
    this.authState = user
    this.updateUserData()
  })
  .catch(error => console.log(error));
}

//// Email/Password Auth ////

emailSignUp(email:string, password:string) {
  return this.afAuth.auth.createUserWithEmailAndPassword(email, password)
    .then((user) => {
      this.authState = user
      this.updateUserData()
    })
    .catch(error => console.log(error));
}

emailLogin(email:string, password:string) {
   return this.afAuth.auth.signInWithEmailAndPassword(email, password)
     .then((user) => {
```

```
    return this.afAuth.auth.signInWithEmailAndPassword(email, password)
      .then((user) => {
        this.authState = user
        this.updateUserData()
      })
      .catch(error => console.log(error));
}

// Sends email allowing user to reset password
resetPassword(email: string) {
  var auth = firebase.auth();

  return auth.sendPasswordResetEmail(email)
    .then(() => console.log("email sent"))
    .catch((error) => console.log(error))
}



//// Sign Out ////

signOut(): void {
  this.afAuth.auth.signOut();
  this.router.navigate(['/'])
}



//// Helpers ////

private updateUserData(): void {
// Writes user name and email to realtime db
// useful if your app displays information about users or for admin features

  let path = `users/${this.currentUserId}`; // Endpoint on firebase
  let data = {
                email: this.authState.email,
                name: this.authState.displayName
              }

  this.db.object(path).update(data)
  .catch(error => console.log(error));

}
```

```
    }
```