rxjs（reactive extention js）：通过Observable来编写异步和基于事件的程序：https://rxjs-dev.firebaseapp.com/

主要概念：

- Observable可观察对象
- Observer观察者：回调函数集合

```
const ob1 = new Observable((observer) => {
  setInterval(() => {
    // 请求成功，发送数据
    observer.next({success: true, data: 1});
    observer.next({success: true, data: 2});
    observer.next({success: true, data: 3});
    // 如果出错，执行error()
    // observer.error({success: false, data: 1});
    // 如果请求结束，执行complete()
    // observer.complete();
  }, 2000);
});
```

- Subscription订阅：表示Observable执行，可以用它取消Observable执行

```
const subscription = ob1.subscribe((result) => {
  console.log(result);
  // 通过订阅对象可取消
  subscription.unsubscribe();
});
```

- Operators操作符：操作集合的函数
  - filter 过滤
  - map 映射：数据格式转换加工

```
// 操作符
ob5.pipe(
  filter(n => n % 2 !== 0), // 过滤奇数
  map(n => n * n), // 求平方
  // ...
).subscribe(
  n => console.log(n)
);
```

- Subject主题：相当于事件派发器

创建Observable：

- new Observable(subscriber)

- Observable.create(subscriber)
- 通过Promise：
- 通过定时器
- 通过事件
- 通过已存在值

```
// 通过promise创建
const ob2 = fromPromise(fetch('assets/data.json'));
ob2.subscribe({
  next(resp) {
    console.log(resp);
  },
  error(error) {
    console.log(error);
  }
});

// 通过定时器构造
const ob3 = interval(1000).pipe(
  take(5)
);
ob3.subscribe(val => console.log('计数: ' + val));

// 通过事件构造
const ob4 = fromEvent(document.getElementById('p1'), 'click');
ob4.subscribe((evt: MouseEvent) => {
  console.log(evt.clientX + '-' + evt.clientY);

});

// 通过已存在的值
const ob5 = of(1, 2, 3, 4); // Observable<number>
const ob6 = of([1, 2, 3, 4]); // Observable<Array<number>>
const ob7 = of({foo: 'bar'}); // Observable<{foo:string}>
```

错误处理

- 方式一：ob.subscribe(next,error)
- 方式二：操作符catchError(error => of(...)),

> 使用了方式二，方式一error回调就不会执行了，数据会进入next流程

用户注册

图形验证码校验

- 图形验证码异步校验

```
@Directive({
```

```
  selector: '[appVerifyCodeImg]',
  providers: [
    {provide: NG_ASYNC_VALIDATORS, useExisting: CaptchaValidatorDirective, multi: true}
  ]
})
export class CaptchaValidatorDirective {

  constructor(private us: UserService) {
  }

  validate(control: AbstractControl): Promise<ValidationErrors | null> |
Observable<ValidationErrors | null> {
    // 得到数据结构是Observable<Result<string>>,
    // 但是当前函数需要Observable<ValidationErrors>
    return this.us.verifyCodeImg(control.value).pipe(// rxjs编程：函数式操作
      // map操作符用于数据类型转换
      map((r: Result<null>) => {
        // null说明校验通过
        return r.success ? null : {verifyCodeImg: true};
      }),
      // catchError操作符捕获可能出现的错误，如果出错校验就失败了，
      // 这里封装一个Observable<ValidationErrors>并返回
      catchError(e => of({verifyCodeImg: true}))
    );
  }
}
```

- 短信异步校验(同上)
- 注册

作业：

1.完成短信异步校验

2.复习巩固

3.预习路由跳转、传参等核心知识点预习