Angular

Workshops part 3

How to inform Parent?

If your web developer and practice tells you – by event, you are right.

Parent could listen its child/s for events and when events happen react.

There are some other options (based on architecture and what you want to achieve) but we focus on @Output now.

How to inform Parent?

How we manage that?

In the book components create a html hock to bind some method – optional,

```
(click)="delete(book)"
```

Declare event emitter book component

```
@Output() deleted = new EventEmitter<
Book>();
```

Implement method with emitted event inside

```
delete(book: Book) {this.deleted.emit
(book);}
```

Catch the event and react in parent

```
deleteBook(book: Book) {
    this.books = this.books.filter(
(b) => book.id !== book.id );
  }
```

Also in html

```
(deleted)="deleteBook($event)"
```

Introduction to Reactive Extensions

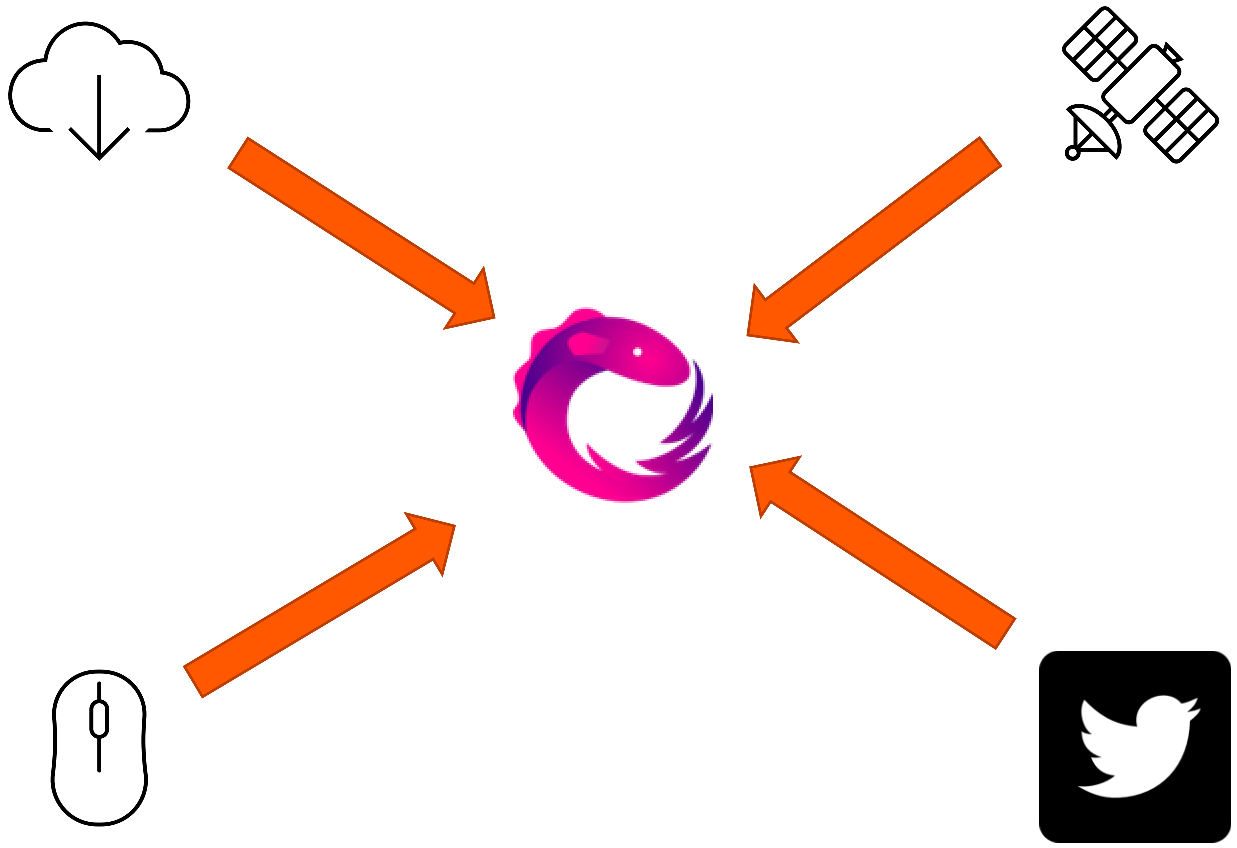
avanade

Reactive programming

In computing, **reactive programming** is a declarative programming paradigm concerned with data streams and the propagation of change.

WIKIPEDIA
The Free Encyclopedia

Reactive Extensions



- RX.NET
- RxJava
- **RxJS**
- RxScala
- …

http://reactivex.io/

Basic components of Rx

- Observables
  - Observer/Observable + Iterator pattern
  - Emit events or values
  - You can subscribe to start observe
- Operators
  - allows you to transform, combine, manipulate, and work with the sequences of items emitted by Observables
  - Can be chained

Observables + Angular

- RxJS is installed by default by Angular during project scaffolding as many Angular features uses RxJS „under the hood”
- Observables by convention are marked by $ at the end of the Observable name. We can also specified what we expect from observable. For Example in this case we expect array of users.

```
books$: Observable<Book[]>;
```

To be ready for http coming!

We will prepare our component to handle data coming from API.

To do that we need to create observable by using „of” or „from” operator.

OF and FROM

There is slight difference between those two, but we won’t dig into that now.

Lets use of()

of(['string1', 'string2', ... 'stringN']);

```
of(books.sort(this.compareName));
```

And now we can subscribe by async pipe

```
div *ngFor="let book of books$ | async">
  <app-book [childBook]="book" (deleted)="delete($event)"></app-book>
</div>
```

Mind that book component doesn’t care about where the data come from.

STEP by STEP

We are changing everything step by step.

It helps everyone to follow and understand all concepts.

Subscribing to observable is needed because data we will get will be asynchronous.

If you can imagine how to do it better we will definitelly do later.

We are ready to be asynchronous

So let’s do it!

We will create our services to serve some business logic.

What is business requirement? For sure we need CRUD!

Thank you

avanade