

RxJava Code Lab

<https://github.com/jraska/RxJava-Codelab>

Josef Raska (Android Ocean team)

Agenda

- Motivation
- Brief overview of ReactiveX
- **Let's code!**
- Transformations
- Combining
- Error handling
- Scheduling

Why this Code Lab?

- Rx is powerful tool
- Rx is trendy
- We use Rx and we want to use it more
- Different way of thinking
- Difficult to understand
- Lot of confusion

Why Rx?

```
interface UserManager {  
    User getUser();  
    void setName(String name);  
    void setAge(int age);  
}
```

Why Rx?

```
interface UserManager {  
    User getUser();  
    void setName(String name);  
    void setAge(int age);  
}
```

```
UserManager um = new UserManager();  
System.out.println(um.getUser());
```

Why Rx?

```
interface UserManager {  
    User getUser();  
    void setName(String name);  
    void setAge(int age);  
}
```

```
UserManager um = new UserManager();  
System.out.println(um.getUser());
```

```
um.setName("John Doe");  
System.out.println(um.getUser());
```

Why Rx?

```
interface UserManager {  
    User getUser();  
    void setName(String name); // <- now async  
    void setAge(int age);  
}
```

```
UserManager um = new UserManager();  
System.out.println(um.getUser());
```

```
um.setName("John Doe");  
System.out.println(um.getUser());
```

Why Rx?

```
interface UserManager {  
    User getUser();  
    void setName(String name, Runnable callback);  
    void setAge(int age, Runnable callback);  
}
```


Why Rx?

```
interface UserManager {  
    User getUser();  
    void setName(String name, Runnable callback);  
    void setAge(int age, Runnable callback);  
}
```

```
UserManager um = new UserManager();  
System.out.println(um.getUser());
```

```
um.setName( name: "John Doe", new Runnable() {  
    @Override public void run() {  
        System.out.println(um.getUser());  
    }  
});
```

Why Rx?

```
interface UserManager {  
    User getUser();  
    void setName(String name, Listener callback);  
    void setAge(int age, Listener callback);  
  
    interface Listener {  
        void success();  
        void failure(IOException e);  
    }  
}
```

Why Rx?

```
UserManager um = new UserManager();  
System.out.println(um.getUser());
```

```
um.setName( name: "John Doe", new UserManager.Listener() {  
    @Override public void success() {  
        System.out.println(um.getUser());  
    }  
  
    @Override public void failure(IOException e) {  
        // TODO: handle error  
    }  
});
```

Why Rx?

```
userManager.um = new UserManager();
System.out.println(um.getUser());

um.setName( name: "John Doe", new UserManager.Listener() {
    @Override public void success() {
        System.out.println(um.getUser());
    }

    @Override public void failure(IOException e) {
        // TODO: handle error
    }
});

um.setAge( age: 33, new UserManager.Listener() {
    @Override public void success() {
        System.out.println(um.getUser());
    }

    @Override public void failure(IOException e) {
        // TODO: handle error
    }
});
```

Why Rx?

```
userManager um = new UserManager();
System.out.println(um.getUser());

um.setName( name: "John Doe", new UserManager.Listener() {
    @Override public void success() {
        System.out.println(um.getUser());

        um.setAge( age: 33, new UserManager.Listener() {
            @Override public void success() {
                System.out.println(um.getUser());
            }

            @Override public void failure(IOException e) {
                // TODO: handle error
            }
        });
    }

    @Override public void failure(IOException e) {
        // TODO: handle error
    }
});
```

Why Rx?

```
private TextView textView;  
private UserManager um = new UserManager();  
  
void updateUserName() {  
    um.setName( name: "John Doe", new UserManager.Listener() {  
        @Override  
        public void success() {  
            runOnUiThread(new Runnable() {  
                @Override  
                public void run() {  
                    textView.setText(um.getUser().toString());  
                }  
            });  
        }  
    });  
  
    @Override  
    public void failure(IOException e) {  
        // TODO: handle error  
    }  
});
```

Why Rx?

Can you model your whole system synchronously?

- Removes the web of callbacks
- Declarative
- Simple asynchronous composing
- Threading abstraction
- Code more readable
- Less bugs

RxJava \subset ReactiveX

“ReactiveX is a library for composing asynchronous and event-based programs by using observable sequences.”

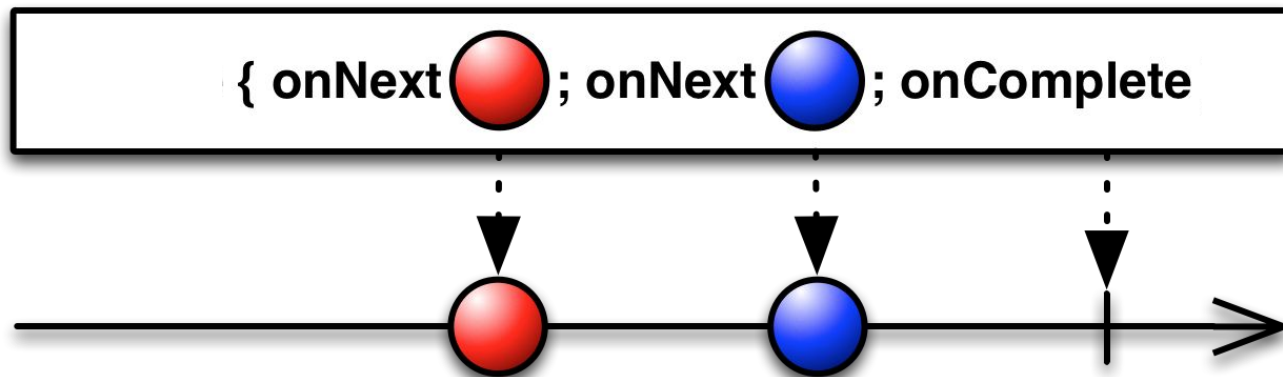
- Observer pattern
- Concepts language independent

Observable<T>

`onNext (T)`

`onError (Throwable)`

`onCompleted ()`



Observable is “dual” to Iterable

event	Iterable (pull)	Observable (push)
retrieve data	<code>T next()</code>	<code>onNext(T)</code>
discover error	throws <code>Exception</code>	<code>onError(Exception)</code>
complete	<code>!hasNext()</code>	<code>onCompleted()</code>

Why Rx?

```
interface UserManager {  
    User getUser();  
    void setName(String name, Listener callback);  
    void setAge(int age, Listener callback);  
}
```

```
interface Listener {  
    void success();  
    void failure(IOException e);  
}  
      ↓   ↓   ↓
```

```
interface UserManager {  
    Observable<User> getUser();  
    Observable<User> setName(String name);  
    Observable<User> setAge(int age);  
}
```

Why Rx?

```
userManager.um = new UserManager();
System.out.println(um.getUser());

um.setName( name: "John Doe", new UserManager.Listener() {
    @Override public void success() {
        System.out.println(um.getUser());
    }

    @Override public void failure(IOException e) {
        // TODO: handle error
    }
});

um.setAge( age: 33, new UserManager.Listener() {
    @Override public void success() {
        System.out.println(um.getUser());
    }

    @Override public void failure(IOException e) {
        // TODO: handle error
    }
});
```

Why Rx?

```
UserManager um = new UserManager();
```

```
um.setName("John Doe")
```

```
    .concatWith(um.setAge(33))
```

```
    .subscribe(System.out::println);
```

Why Rx?

```
userManager um = new UserManager();
System.out.println(um.getUser());

um.setName( name: "John Doe", new UserManager.Listener() {
    @Override public void success() {
        System.out.println(um.getUser());

        um.setAge( age: 33, new UserManager.Listener() {
            @Override public void success() {
                System.out.println(um.getUser());
            }

            @Override public void failure(IOException e) {
                // TODO: handle error
            }
        });
    }

    @Override public void failure(IOException e) {
        // TODO: handle error
    }
});
```

Why Rx?

```
UserManager um = new UserManager();
```

```
um.setName("John Doe")  
    .flatMap((user) -> um.setAge(33))  
    .subscribe(System.out::println);
```

Why Rx?

```
private TextView textView;  
private UserManager um = new UserManager();  
  
void updateUserName() {  
    um.setName( name: "John Doe", new UserManager.Listener() {  
        @Override  
        public void success() {  
            runOnUiThread(new Runnable() {  
                @Override  
                public void run() {  
                    textView.setText(um.getUser().toString());  
                }  
            });  
        }  
    });  
  
    @Override  
    public void failure(IOException e) {  
        // TODO: handle error  
    }  
});
```


Why Rx?

```
TextView textView;  
UserManager um = new UserManager();  
  
void updateUserName() {  
    um.setName("John Doe")  
        .subscribeOn(schedulers.io())  
        .observeOn(schedulers.mainThread())  
        .subscribe(user -> textView.setText(user.toString()));  
}
```

Codelab project

- RxJava 2
- Code playground is unit test
- Separate tasks for particular areas
- Each task has a solution in `solutions` package

Project setup

- Running unit tests with icon →

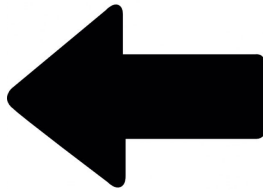
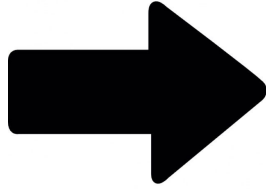
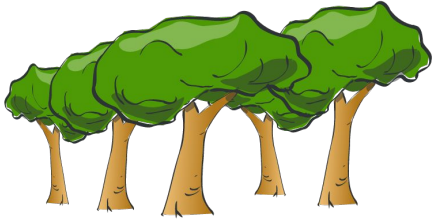


- Is everyone ready?
- Entertainment meanwhile:

<http://reactivex.io/intro.html>

<http://rxmarbles.com/>

Firewood Deploy Chain



Now let's make some furniture

