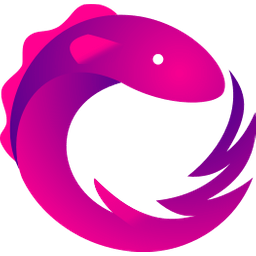
RxJS Tutorial



RxJS tutorial provides basic and advanced concepts of Reactive Extension for Javascript and Reactive programming. In this tutorial, you will get an overview of Reactive programming, Reactive extension for JavaScript, and its programming concepts, functionalities of RxJS etc. in an easy and simple way. Our RxJS tutorial is designed for beginners and professionals.

RxJS is a [JavaScript](https://www.javatpoint.com/javascript-tutorial) library that can used with other JavaScript libraries and frameworks such as [Angular](https://www.javatpoint.com/angularjs-tutorial), ReactJS, Vue.js, Node.js etc. It is supported by JavaScript and also with TypeScript.

Our RxJS tutorial includes all topics of reactive programming, RxJS Latest Updates, RxJS Observables, RxJS Operators, RxJS working etc.

Introduction to RxJS

RxJS is an acronym that stands for **Reactive Extension for JavaScript**. It is a JavaScript library that is used to work with Reactive programming that deals with asynchronous data calls, callbacks and event-based programs. RxJS can be used with other JavaScript libraries such as Angular, [ReactJS](https://www.javatpoint.com/reactjs-tutorial), [Vue.js](https://www.javatpoint.com/vue-js), [Node.js](https://www.javatpoint.com/nodejs-tutorial) etc. It is supported by JavaScript and also with [TypeScript](https://www.javatpoint.com/typescript-tutorial).

What is Reactive Programming?

Reactive Programming is a declarative programming paradigm that concerns data streams and the propagation of changes. It is another different way to built software applications. In Reactive Programming, we can easily express static such as arrays or dynamic such as event emitters data streams. It also specifies that an existed inferred dependency within the associated execution model can facilitate the automatic propagation of the changed data flow.

**For example:**

In the imperative programming declaration, you have seen that **a:=b+c** would simply mean that a is being assigned as the result of **b+c** in the instant the expression is evaluated. You can change the values of b and c later, and there would be no effect on the value of a. On the other hand, in reactive programming, the value of a is automatically updated whenever the values of **b or c** is changed, without the program having to re-execute the statement **a:=b+c.** In simple means, it determines that **a** is always assigned with updated values of **b+c**.

What is RxJS?

The official website of RxJS has defined it as a library for composing asynchronous and event-based programs by using observable sequences. It provides one core type, the Observable, satellite types (Observer, Schedulers, Subjects) and operators inspired by Array#extras (map, filter, reduce, every, etc.) to allow handling asynchronous events as collections.

RxJS is a JavaScript library that has introduced the concept of "reactive programming" to the web. In general terms, you can say that RxJS is a Reactive programming language. It provides way to write assembly lines in your software applications. It facilitates you to write software that can be reusable, configurable, and asynchronous.

Features of RxJS

Following is the list of essential concepts or primary features of RxJS that are used to solve async event management:

**Observable**

The observable is used to represent the idea of an invokable collection of future values or events. It can be defined as a function that creates an observer and attaches it to the source where values are expected. The observable examples are clicks, mouse events from a dom element or an Http request, etc.

**Observer**

It is a collection of callbacks that are used to specify how to listen to values delivered by the Observable. It can be specified as an object with next(), error() and complete() methods, that will be executed when you have to interact with observable.

**Subscription**

It is used to represent the execution of an Observable. It is primarily used to cancel the execution. For example, when the observable is created, you have to subscribe to it to execute the observable. It can also be used to cancel the execution.

**Operators**

Operators are simple functions used to enable a functional programming style to deal with collections with operations like map, filter, concat, flatMap, etc. It takes observable as input and gives the output also in the form of observable.

**Subject**

A subject is equivalent to an EventEmitter and used to multicast a value or event to multiple Observers. For example: Consider a button with an event listener; the function attached to the event will be called whenever the user will click on the button.

**Schedulers**

The Schedulers are centralized dispatchers that are used to control concurrency. They are used to control the execution when the subscription has to start and notified.

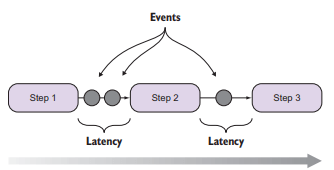
When to use RxJS?

RxJS is preferred to use when your project contains lots of async task handling. It is the right choice for these conditions. RxJS is loaded by default with the Angular project.

Why use RxJS?

We have used many JavaScript async libraries such as [JQuery](https://www.javatpoint.com/jquery-tutorial), Async.js, Q.js, or others, and everyone has their own preference, but they all have some drawbacks. [RxJS](https://www.javatpoint.com/rxjs) is preferred overall because by combining the functional and reactive programming paradigms, RxJS will help you resolve the following issues:

* Most used control flow structures such as for and while loops don't work well together with asynchronous functions because they are not async compatible. So, they are unaware of wait time or latency between iterations.
* Error-handling strategies become complicated when you begin nesting try/catch blocks within each callback. RxJS provides an error-handling approach from a functional perspective.
* The business logic is tightly coupled within the nested callback structure, and it makes the code complex. The more nested your code is, the harder it will be in terms of readability and complexity. RxJs provides an ideal way to create reusable and modular components in order to have loosely coupled business logic that can be maintained and unit tested independently.
* Excessive use of closures is not a good practice in programming, but functions in JavaScript create a closure around the scope in which they are declared. When you nest them, you have to be concerned about the state of the variables passed in as arguments and the state of all external variables surrounding each function declaration, causing side effects to occur. These side effects increase the state's cognitive load and make it difficult to keep track of the code. RxJS provides a way to deal with the side effects of this problem.
* RxJS provides an ideal and easy mechanism to cancel events cleanly after some predetermined amount of time. Implementing your cancellation mechanism can be very challenging and error prone even with the help of third-party libraries.
* Good quality of the responsive design is always throttling a user's interaction with any UI components so that the system isn't unnecessarily overloaded. RxJS tells you how to use throttling and debouncing to get rid of unnecessary overload. To get a manual solution for achieving this is very hard to get right and involve functions that access data outside their local scope, which breaks your entire program's stability.



The above image shows the program execution in RxJS. RxJS treats asynchronous data flows with a programming model that resembles a simple chain of sequential steps.

Advantages and Disadvantages of RxJS

Let's see the advantages and disadvantages of RxJS. First, we see the advantages or pros of RxJS.

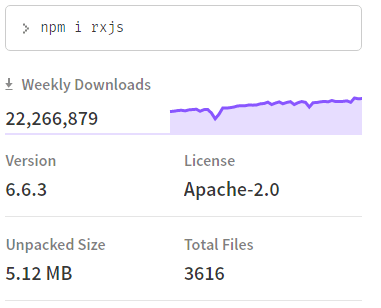
Advantages of RxJS

The following is a list of the advantages of using [RxJS](https://www.javatpoint.com/rxjs):

**RxJS is growing rapidly**

RxJS is a functional and powerful tool integrated into a wide range of frameworks. It makes reactive programming more appealing and approachable. It is growing and evolving rapidly. The following facts can clearly show its immense popularity:

* It has more than 2.4 million dependent repositories on GitHub.
* It has more than 16 thousands of dependent packages on NPM.
* The number of weekly downloads from NPM has exceeded 22 million.



**Flexible to Use**

RxJS can easily be used with other [Javascript](https://www.javatpoint.com/javascript-tutorial) libraries and frameworks such as [Angular](https://www.javatpoint.com/angularjs-tutorial), [ReactJS](https://www.javatpoint.com/reactjs-tutorial), [Vue.js](https://www.javatpoint.com/vue-js), [Nodejs](https://www.javatpoint.com/nodejs-tutorial) etc. It is supported by JavaScript and also with [TypeScript](https://www.javatpoint.com/typescript-tutorial).

**A vast and helpful library**

RxJS is an incredible and vast library when it comes to the handling of async tasks. RxJS uses observables to work with reactive programming that deals with asynchronous data calls, callbacks and event-based programs.

**High-quality API**

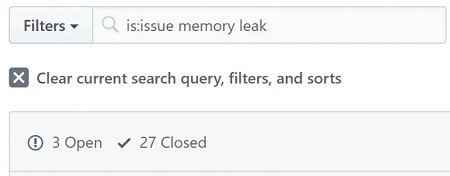
RxJS provides a graceful API that makes it easy to describe a complex stream of data compact and precise. RxJS comes with a comprehensive kit of standard entities such as Subjects, Observables, and operators which undertakes all the hard work and makes it possible. Using this method hides the complexity of the process from a developer and allows him to focus more on the product's logic. Due to this high-quality API, it becomes possible for a developer to significantly simplify the workflow with asynchronous data flows, and he/she can save plenty of time and effort for other tasks.

**Makes Programming Easy**

RxJS provides a vast collection of operators in mathematical, transformation, utility, filtering, conditional, error handling, join categories that make programming easy and simple when used with reactive programming.

**Provides good optimization and memory leaks protection**

RxJS has a solid background that assures developers of high quality of optimization. Many developers have participated in testing and enhancing this library, so the problems associated with memory leaks have already been detected and handled correctly. You can see it in the following image:



**Extensibility**

RxJS provides a comprehensive kit of standard operators, but sometimes, it may be necessary to eject repeating algorithms into new RxJS operators. The extensibility provided by RxJS can achieve that.

**Small and concise footprint**

RxJS provides a high-quality optimization and has a modular architecture. These characteristics make it small, and the production bundle contains only those parts that are used.

**No third-party dependencies**

RxJS does not depend on any third-party. It has a self-sufficient library, which doesn't import other parties of dependencies. Due to this reason, the RxJS project's size is optimal and depends on its libraries that it uses in the project.

According to NPM, currently, RxJS has only one dependency - tslib. This single dependency is used to provide Typescript support.



The lack of dependencies also has some advantage that it is easy to adopt the RxJS itself as a dependency for other packages. For example, API SDKs, utility packages, server-side programs, webhook implementations, front-end event managers etc.

**Big and Responsive Community**

This is one of the most significant advantages of this library that it has a diversified and large community. Because of such a responsive community, RxJS grows even stronger and more popular. The participants of this community help each other to solve the problems and questions on StackOverflow and Gitter.

**Documentation**

Reactive programming and RxJS are not very simple to learn, but they provide extensive documentation to deal with this problem. The official website of RxJS provides community-driven resources that give an in-depth explanation of concepts, best practices, patterns, and examples, and helping users enhance theoretical knowledge and skills.

**Regular updates**

It is also an essential benefit of RxJS is that it is properly maintained. It provides regular updates on GitHub that makes it a good and reliable codebase. It provides regular and consistent updates to improve the overall work of RxJS, making it even more powerful, reliable, and convenient.

Disadvantages of RxJS

There are many advantages to this library that make it great for various purposes, but we should keep in mind that it has some disadvantages. The following is a list of the disadvantages of using RxJS:

**RxJS requires Data immutability**

First, we understand that the need for data immutability in RxJS is not exactly a requirement, but it is instead a concept that came from functional programming, and the reactive paradigm works best and most optimal with the combination of functional programming.

**Testing and Debugging is Difficult**

In RxJS, debugging the code with observables is little difficult. Due to some specific reactive programming features, a developer must be an expert of additional tools and techniques to test the RxJS code.

**Strict typing issues**

The only dependency of RxJS is tslib, which provides Typescript support. The Typescript provides a set of advantages such as strong typing, autocomplete, etc. but also provides some wrong usage of access modifiers (private/public), which makes internal methods accessible from the outside.

**Use of Observables**

To using Observables, you have to wrap your full code under the observables.