Angular Observable vs Angular Promise: Differences, Uses & How To Build Them

[Desislava Dincheva](https://www.infragistics.com/community/blogs/p/contributor?un=AF7F258D1DD463598DE3F0B81CF693927950583F) / Thursday, April 7, 2022

When it comes to Angular, there are two main types of data management: using Observables or Promises with both being capable of managing asynchronous code in JavaScript. At first glance, Observables are seen as a more advanced alternative to Promises. But, in fact, they are quite different. From having separate APIs, through having different motives, to offering distinct ways for handling the async model in JavaScript, Angular Observables and Angular Promises have their own powers.

How to know which one is right for your Angular project? Let’s discuss them in detail, compare each implementation, and see the functions, the advantages and disadvantages. On top of this, I will demonstrate how to use Angular Observables and Angular Promises with our [low-code App Buildertm](https://www.infragistics.com/products/appbuilder)so you can handle asynchronous events easily and quickly.

Here is what the article will cover:

* + [What is a Promise in Angular?](https://www.infragistics.com/community/blogs/b/infragistics/posts/angular-observable-vs-angular-promise#h8sl1orww0f6m72wm126d0h53ftzc4)  
    - [How to create a Promise in Angular?](https://www.infragistics.com/community/blogs/b/infragistics/posts/angular-observable-vs-angular-promise#h14sl1orx10773lnhri9u5n719noe0p)
  + [What is an Observable in Angular?](https://www.infragistics.com/community/blogs/b/infragistics/posts/angular-observable-vs-angular-promise#h19sl1orx5vw4xs67i1usq50ddoh6zk)
    - [How to create an Observable in Angular?](https://www.infragistics.com/community/blogs/b/infragistics/posts/angular-observable-vs-angular-promise#h23sl1orxbsy1543y1r1i0y89g1cdlv5d)
  + [How to use Angular Observables and Promises in App Builder?](https://www.infragistics.com/community/blogs/b/infragistics/posts/angular-observable-vs-angular-promise#h26sl1orxkfq1l0qfq11jtnw3t1uh9ec5)
  + [Angular Observable vs Promise: What’s the difference?](https://www.infragistics.com/community/blogs/b/infragistics/posts/angular-observable-vs-angular-promise#h28sl1orxqws17owpyd17mk1fh2qs2z7)

[**TRY IGNITE UI FOR ANGULAR**](https://www.infragistics.com/products/ignite-ui-angular/download)

What Is Promise in Angular?

Promises in [Angular](https://www.infragistics.com/products/ignite-ui-angular) provide an easy way to execute asynchronous functions that use callbacks, while emitting and completing (resolving or rejecting) one value at a time. When using an Angular Promise, you are enabled to emit a single event from the API. Then, the controller (or the directive) takes on, registering up to three callbacks – success, error, and/or notifications.

There are four states of the Angular Promise:

* fulfilled - action is fulfilled
* rejected - action failed
* pending - action hasn’t succeeded or failed yet
* settled - action is either fulfilled or rejected

Something to remember is that Angular Promise is more passive compared to the Observable and cannot be cancelled once it is started. In other words, when you pass the callback to the Promise constructor (controller or directive), it will either resolve or reject the function.

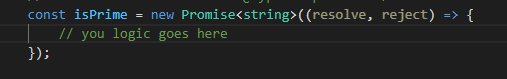
Now, there are several disadvantages with the Angular Promise.

* Promises aren’t cancellable and you have to wait until the callback returns error or success.
* Not suitable for centralized and predictable error handling as errors are sent to the child promise.
* They don’t provide any operations.
* Promises execute an async value only once.
* Cannot be used for multiple values over time.
* They become hard to manage with larger applications.
* They cannot be retried.

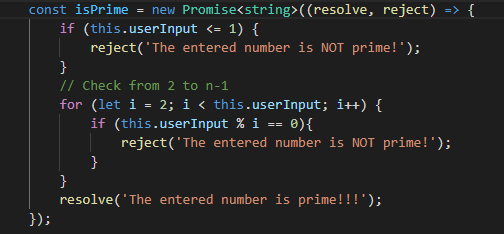
How To Create Promises in Angular?

To create a promise in Angular we just need to use 'new Promise(function)' syntax. The promise constructor takes function as a parameter and that inner function takes resolve and reject as a params.

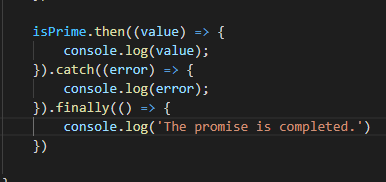
Let’s see how this looks in the code snippet below.



Now that we know how to create an Angular Promise, we will use it to implement a very simple scenario. The created Promise will help us validate whether the number is prime or not.



And finally, to put our method in use, we will handle the Promise in the following way.



What Is Observable in Angular?

Observables provide more capabilities to you as they let developers run functions asynchronously and synchronously. They represent a stream of values that can handle 0, 1, or multiple events, using the same API for each. The thing that I personally like about Angular Observables is the number of operators simplifying coding, including: retry(), or replay(), map, forEach, reduce and more.

Just like Angular Promises, Observables in Angular also have their disadvantages.

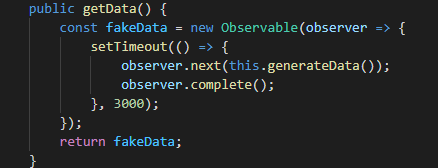
* Observables aren’t natively supported by browsers.
* They require RxJS operators to get a new tailored stream of values.
* Debugging the code with observables may be a bit more challenging.
* You must have the know-how of extra tools and techniques to test the RxJS code.

How To Create Observables in Angular?

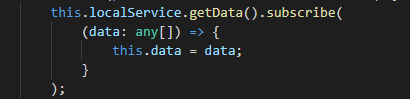
To create an Observable in Angular, first it is necessary to import Observable from rxjs, because as we learned above they are not natively supported.

importing angular observable

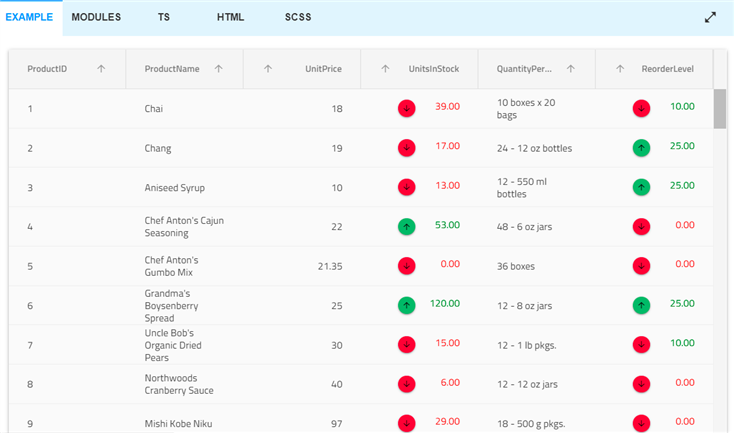
To create a new Observable we just need to use its constructor and to add the logic that will be executed upon subscription. In our case we will create an Observable that will return a great data source for our grid component.



After that what is left for us is to subscribe to the created Observable and bind the received data as a data source to our Grid component.



A great and more meaningful example of a use case for working with observables is using IgxGrid with Remote Data. As you can see in the example, only a small chunk of data is fetched initially when the component is rendered and on each scroll or data operation that is performed a new piece of the data is taken. This observable of type any[] is binded directly to the grid using async Angular pipe. The async pipe subscribes to an Observable or Promise and returns the latest value it has emitted.



[Click here to see the sample](https://www.infragistics.com/products/ignite-ui-angular/angular/components/grid/remote-data-operations#remote-virtualization-demo) and inspect the code right away.

How To Use Angular Observable and Promise in App Builder?

As we understand from the written above the most common scenario to use Observables is when we need to bind data to some component. To fetch data, we must make a http request using HttpClient. The best way to do it is to carry it out into a separate service. All HttpClient methods return an RxJS Observable of something. In general, an Observable can return multiple values over time but an Observable from HttpClient always emits a single value and then completes, never to emit again.

In the App Builder, binding a component to a data source is easy and intuitive. You currently have the following options to choose from in order to use one of our predefined data sources:

* Upload a JSON file or add a URL to a JSON file that contains your data.
* Add/upload a [Swagger](https://www.infragistics.com/community/blogs/b/infragistics/posts/app-builder-release-with-swagger-support-and-more) definition file where all endpoints of your server are described and you can directly get your real data form there.

After that, no matter which approach you choose, you can use the Data drop down from our props panel to bind the data to the component - for example to the Angular grid.

Now we can look at the [generated code in Angular](https://www.infragistics.com/community/blogs/b/jason_beres/posts/angular-code-generation-step-by-step-guide-using-indigo-design). We give you a separate service where the data fetching is performed using http get method.

And you can see in the MasterViewComponent ts file the subscription the data service performed. With this basis, you can conduct even more complicated scenarios whenever necessary.

Angular Observable vs Promise: What’s the Difference?

To present the differences between Observables and Promises in Angular, I decided to compose a table. This way you will be able to compare them in a more digestible and super concise way.

|  |  |
| --- | --- |
| **ANGULAR OBSERVABLES** | **ANGULAR PROMISES** |
| Can handle both synchronous and asynchronous events | Can only handle asynchronous data returns |
| Emit and complete 0, 1, or multiple values over a period of time (stream data in multiple pipeline) | Only emit one value at a time (have one pipeline) |
| Less passive, meaning | More passive, meaning |
| Offer operations like map, filter, reduce, retry(), or replay(), map, forEach, and more | Do not provide operators |
| Are cancellable, using the unsubscribe() method, meaning the listener doesn’t receive further values | Are not cancellable |
| Are lazy, i.e., the producer function triggers after users subscribe to the stream | Are eager, i.e., the Promise callback executes immediately and only once at the moment of its definition, without .then and .catch |
| Subscribers handle errors | Push errors to the child Promises |

Apparently, in the Angular Observable vs Promise comparison, the Observable wins. So, is there a good reason to use Promise, then? Absolutely.

Firstly, Observables come at a certain cost. As I mentioned before, they are not natively supported by browsers. Which means you must implement the RxJS library. And while it helps you create asynchronous or callback-based code, its bundle size can exceed 17.4kb.

Secondly, when your [Angular app](https://medium.com/ignite-ui/how-to-create-angular-app-from-scratch-in-10-minutes-infragistics-blog-fcd98c8b1b9c) is asynchronous by default, there’s no better way to handle all asynchronous functions and events than using Angular Promise. Isn’t that right?

Next, if you don’t want to use reactive style or don’t care about streams of values and cancellable events, then go for Promises in Angular.