



PROJECTS

Ember

Ember Data

TAG: V2.10.0

MODULES

ember

ember-application

ember-debug

ember-extension-support

ember-glimmer

ember-htmlbars

ember-metal

ember-routing

ember-runtime

ember-testing

ember-views

NAMESPACES

Ember

Ember.AutoLocation

Ember.FEATURES

Ember.Instrumentation

Ember.Location

Ember.String

Ember.computed

Ember.inject

Ember.run

CLASSES

Backburner

BucketCache

Container

ContainerProxyMixin

Descriptor

Ember.ActionHandler

Ember.ActionSupport

Ember.Application

Ember.ApplicationInstance

Ember.ApplicationInstance.Boot...

Ember.Array

Ember.ArrayProxy

RSVP.Promise CLASS

DEFINED IN: `bower_components/rsvp/lib/rsvp/promise.js:30`MODULE: `ember`

Promise objects represent the eventual result of an asynchronous operation. The primary way of interacting with a promise is through its `then` method, which registers callbacks to receive either a promise's eventual value or the reason why the promise cannot be fulfilled.

TERMINOLOGY

`promise` is an object or function with a `then` method whose behavior conforms to this specification.

`thenable` is an object or function that defines a `then` method.

`value` is any legal JavaScript value (including undefined, a thenable, or a promise).

`exception` is a value that is thrown using the throw statement.

`reason` is a value that indicates why a promise was rejected.

`settled` the final resting state of a promise, fulfilled or rejected.

A promise can be in one of three states: pending, fulfilled, or rejected.

Promises that are fulfilled have a fulfillment value and are in the fulfilled state. Promises that are rejected have a rejection reason and are in the rejected state. A fulfillment value is never a thenable.

Promises can also be said to *resolve* a value. If this value is also a promise, then the original promise's settled state will match the value's settled state. So a promise that *resolves* a promise that rejects will itself reject, and a promise that *resolves* a promise that fulfills will itself fulfill.

BASIC USAGE:

```
1  var promise = new Promise(function(resolve, reject) {
2    // on success
3    resolve(value);
4
5    // on failure
6    reject(reason);
7  });
8
9  promise.then(function(value) {
10   // on fulfillment
11 }, function(reason) {
12   // on rejection
13 });
```

ADVANCED USAGE:

Promises shine when abstracting away asynchronous interactions such as `XMLHttpRequest`s.

```
1  function getJSON(url) {
2    return new Promise(function(resolve, reject){
3      var xhr = new XMLHttpRequest();
4
5      xhr.open('GET', url);
6      xhr.onreadystatechange = handler;
7      xhr.responseType = 'json';
8      xhr.setRequestHeader('Accept', 'application/json');
9      xhr.send();
10 }
```

- Ember.Binding
- Ember.Checkbox
- Ember.ClassNamesSupport
- Ember.Comparable
- Ember.Component
- Ember.ComputedProperty
- Ember.ContainerDebugAdapter
- Ember.Controller
- Ember.ControllerMixin
- Ember.Copyable
- Ember.CoreObject
- Ember.CoreView
- Ember.DataAdapter
- Ember.Debug
- Ember.DefaultResolver
- Ember.EachProxy
- Ember.Engine
- Ember.EngineInstance
- Ember.Enumerable
- Ember.Error
- Ember.EventDispatcher
- Ember.Evented
- Ember.Freezable
- Ember.HTMLBars
- Ember.HashLocation
- Ember.Helper
- Ember.HistoryLocation
- Ember.InjectedProperty
- Ember.LinkComponent
- Ember.Map
- Ember.MapWithDefault
- Ember.Mixin
- Ember.MutableArray
- Ember.MutableEnumerable
- Ember.Namespace
- Ember.NativeArray
- Ember.NoneLocation
- Ember.Object
- Ember.ObjectProxy
- Ember.Observable
- Ember.OrderedSet
- Ember.PromiseProxyMixin
- Ember.ProxyMixin
- Ember.Route
- Ember.Router
- Ember.Service
- Ember.TargetActionSupport
- Ember.Templates.helpers
- Ember.Test

```
11     function handler() {
12       if (this.readyState === this.DONE) {
13         if (this.status === 200) {
14           resolve(this.response);
15         } else {
16           reject(new Error('getJSON: `' + url + '` failed with status: [' + this.status + ']'));
17         }
18       }
19     };
20   });
21 }
22
23 getJSON('/posts.json').then(function(json) {
24   // on fulfillment
25 }, function(reason) {
26   // on rejection
27 });
```

Unlike callbacks, promises are great composable primitives.

```
1 Promise.all([
2   getJSON('/posts'),
3   getJSON('/comments')
4 ]).then(function(values){
5   values[0] // => postsJSON
6   values[1] // => commentsJSON
7
8   return values;
9 });
```

Index

Methods

Show: ☒ Inherited ☐ Protected ☐ Deprecated

METHODS

catch
finally

then

- Ember.Test.Adapter
- Ember.Test.QUnitAdapter
- Ember.TextArea
- Ember.TextField
- Ember.TextSupport
- Ember.View
- Ember.ViewMixin
- Function
- Libraries
- RSVP
- RSVP.EventTarget
- RSVP.Promise
- Registry
- RegistryProxyMixin
- RoutingService
- String
- Transition