Promise Object

Published 08/07/2015 01:48 AM | Updated 07/03/2020 12:44 AM | Answer Id: 43804

SuiteCloud Platform SuiteScript SuiteScript 2.0 API Reference SuiteScript 2.0 Global Objects Promise Object

Note: The content in this help topic pertains to SuiteScript 2.0.

In SuiteScript 2.0, all client scripts support the use of Promises. With Promises, developers can write asynchronous code that is intuitive and efficient. SuiteScript 2.0 provides promise APIs for selected modules (see SuiteScript 2.0 Promise APIs). In addition, you can create custom Promises in all client scripts (see Custom Promises).

A promise is a JavaScript object that represents the eventual result of an asynchronous process. After this object is created, it serves as a placeholder for the future success or failure of the operation. During the period of time that the promise object is waiting, the remaining segments of the script can execute.

A Promise holds one of the following values:

- `fulfilled` – The operation is successful.
- `rejected` – The operation failed.
- `pending` – The operation is still in progress and has not yet been fulfilled or rejected.

When it is first created, a Promise holds the value `pending`. After the associated process is complete (from success or failure), the value changes to `fulfilled` or `rejected`. A success or failure callback function attached to the Promise is called when the process is complete. Note that a Promise can only succeed or fail one time. When the value of the Promise updates to fulfilled or rejected, it cannot change.

For additional information regarding Promises, see https://www.promisejs.org/.

SuiteScript 2.0 Promise APIs

SuiteScript 2.0 provides client-side promise APIs. For supported modules members and additional API information, see SuiteScript 2.0 Modules.

Important: Although these modules as a whole are supported in client and server-side scripts, their promise APIs are supported only in client scripts.

The available promise APIs are named so that they correspond with their synchronous counterparts. The distinction is that the promise APIs have names that are suffixed with `.promise`. For example, the `search.create(options)` API has a promise version named `search.create.promise(options)`.

The following is a basic example of how to use a promise API in a client script.

```
/**
* @NAPIVersion 2.0
*/
define(['N/search'],
    function(search)
    {
        function  doSomething()
        {
            search.create.promise({
                type: 'salesorder'
            })
            .then(function(result) {
                log.debug("Completed: " + result);
                //do something after completion
            })
            .catch(function(reason) {
                log.debug("Failed: " + reason)
                //do something on failure
            });
        }
        return
        {
            pageInit: doSomething
        }
    }
);
```

Copy

Ask a Question

Submit a Case

For lower severity cases, please submit a case via Contact Support Online to initiate research and testing on your issue.

Contact Support Online

Ask a Guru

Case Severity Level Definitions

Case Status Definitions

Urgent Questions

Call Us

For higher severity cases that significantly affect your use of NetSuite, please contact NetSuite Support by phone

Contact Support by Phone

NetSuite Support Phone Menu Routing Options

NetSuite Issue Resolution Process

Severities and Status Definitions for Non-ERC Defects

Log in to Release Preview

Release Preview phone #

North American Toll Free:

1-877-774-4271

International:

1-925-948-1084

Additional Resources

- Help Topic Weekly Updates
- Tips
- Videos
- Training Course Catalog
- Training Webinars & Events
- Contact NetSuite Training

Recently Viewed Answers

- Using Internal IDs, External IDs, and References
- External IDs Overview
- SAML Roles and Permissions FAQ
- Opt-In for Release Preview in Test Drive/Demo Account (TSTDRV), but No Account Is Created

This example demonstrates how to chain promises created with promise APIs.

```
/**
* @NAPIVersion 2.0
*/
define(['N/search'],
    function(search)
    {
        function doSomething()
        {
            var filter = search.createFilter({
                name: 'mainline',
                operator: search.Operator.IS,
                values:['T']
            });
            search.create.promise({
                type: 'salesorder',
                filters:[filter]
            })
            .then(function(searchObj) {
                return searchObj.run().each.promise(
                    function(result, index){
                    //do something
                })
            })
            .then(function(result) {
                log.debug("Completed: " + result)
                //do something after completion
            })
            .catch(function(reason) {
                log.debug("Failed: " + reason)
                //do something on failure
            });
        })
        return
        {
            pageInit: doSomething
        }
    }
);
```

Copy

Custom Promises

The following example shows a custom Promise. Custom Promises do not utilize the SuiteScript 2.0 promise APIs.

```
/**
* @NAPIVersion 2.0
*/
define(function(){
    function doSomething(addresses){
        var promise = new Promise(function(resolve, reject){
                var url = 'https://your.favorite.maps/api/directions?start=' + addresses.start + '&end=' + addresses.end,
                    isAsync = true,
                    xhr = new XMLHttpRequest();

                xhr.addEventListener('load', function (event) {
                    if (xhr.readyState === 4) {
                        if (xhr.status === 200) {
                            resolve(xhr.responseText );
                        }
                        else {
                            reject( xhr.statusText );
                        }
                     }
                });
                xhr.addEventListener('error', function (event) {
                    reject( xhr.statusText );
                });
                xhr.open('GET', url, isAsync);
                xhr.send();
            });

        return promise;
    }

    return {
        lookupDirections: doSomething
    };
});
```

Copy

Related Topics

- SuiteScript 2.0 Global Objects
- define Object
- require Function
- log Object
- util Object
- toString()
- JSON object