**What are deferred and promise object in jQuery?**

In this article, we will see the deferred and promise object in jQuery, along with understanding their basic implementation and the differences between them.

**Deferred objects**are a fundamental part of the jQuery library, used for handling asynchronous requests and operations. They provide a way to trigger callbacks when an operation is either completed or failed and can be chained together to create complex asynchronous flows. Deferred objects are a representation of an operation that may not have been completed yet. They can be used to trigger callbacks when the operation is either completed or failed and can be chained together to create complex asynchronous flows.

**Syntax:** The following is the syntax for creating a deferred object and resolving it.

var deferred = $.Deferred();

deferred.resolve();

**Approach 1:**Here, we have an HTML page with a button and a message element. When the user clicks the button, we want to resolve a deferred object and display a message. To achieve this, we first include the jQuery library in the HTML page and then use the *$(document).ready()* function to ensure that the code will execute when the DOM is fully loaded. Inside the $(document).ready() function, we create a deferred object using the *$.Deferred() function*. This creates an object that can be resolved at a later time. Next, we bind a click event to the button using the *$(‘#button’).click()* function. This function will execute when the user clicks the button. Inside the click event handler, we use the deferred.resolve() function to resolve the deferred object. Finally, we bind a done callback function to the deferred object using the deferred.done() function. This function will execute when the deferred object is resolved. Inside the done callback function, we use the *$(‘#message’).text(*) function to update the text of the message element with the string “Deferred object resolved”. When the user clicks the button, the click event handler will execute and the deferred object will be resolved. This will trigger the callback function to execute is done, causing the message to be displayed.

**Example:**In this example, we will see how to resolve a deferred object when the user clicks a button using jQuery.

$(document).ready(function () {

            // Create deferred object

            var deferred = $.Deferred();

            // Bind click event to button

            $('#button').click(function () {

                  // Resolve deferred object

                deferred.resolve();

            });

            // Bind done callback to deferred object

            deferred.done(function () {

                // Display message when deferred object

                // is resolved

                $('#message').text(

                    'Deferred object resolved'

                );

            });

        });

A**Promise object** in jQuery is a representation of the result of a deferred operation. It allows you to access the result of a deferred operation once it has been completed, and to attach callbacks to be executed when the operation is completed or fails

**Syntax:**

var def = $.Deferred();

var promise = def.promise();

**Approach 2:**Here, we create a deferred object def using $.Deferred() and a promise object promise using the promise() method of the deferred object. We then use the done() method of the promise object to attach a callback that will be executed when the promise is resolved, and the fail() method to attach a callback that will be executed when the promise is rejected. We also added two buttons with ids “resolve” and “reject” respectively. When the “resolve” button is clicked, the resolve() method of the deferred object is called, which in turn triggers the done() callback attached to the promise object, and shows the resolved data in the div with id “output”. And when the “reject” button is clicked, the reject() method of the deferred object is called, which in turn triggers the fail() callback attached to the promise object, and shows the rejected message in the div with id “output”.

**Example:**In this example, we will look at how to resolve a promise object when the user clicks a button using jQuery.

$(document).ready(function () {

            var def = $.Deferred();

            var promise = def.promise();

            promise.done(function (data) {

                console.log("Promise resolved with ", data);

                $("#output").html("Promise resolved with " + data);

            });

            promise.fail(function () {

                console.log("Promise rejected");

                $("#output").html("Promise rejected");

            });

            $("#resolve").click(function () {

                def.resolve("example data");

            });

            $("#reject").click(function () {

                def.reject();

            });

        });

Deferred and promise objects are both fundamental concepts in jQuery for handling asynchronous operations. While they are closely related, there are a few key differences between them.

