jQuery

5.1 jQuery Introduction

5.1.1 Use of jQuery: -

jQuery is a popular JavaScript library that simplifies web development by providing a consistent and efficient way to work with HTML elements, handle events, make asynchronous requests, and create animations. It has been widely used in web development for years, although in recent years, native JavaScript has evolved to include many features and capabilities that were previously provided by jQuery.

- **DOM Manipulation:** jQuery is often used to select and modify HTML elements. For example, you can change the content of a div, hide/show elements, or change CSS styles with ease.
- **Event Handling:** jQuery simplifies attaching event handlers to elements, making it easier to respond to user actions like clicks and keyboard input.
- **Ajax Requests:** jQuery's Ajax functions allow you to fetch data from a server without refreshing the entire page, creating a smoother user experience.
- **Animation:** jQuery provides methods for creating animations and transitions, such as fading elements in and out or sliding them across the screen.
- **Form Validation:** It can be used for client-side form validation to provide immediate feedback to users when they submit forms.
- **Responsive Design:** jQuery can be used to create responsive web designs, adapting content and layout based on the user's device or screen size.

To use jQuery in a web project, you typically include the jQuery library in your HTML file using a '<script>' tag, either by hosting it on a content delivery network (CDN) or by downloading and hosting it locally.

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

5.1.2 Difference between jQuery and JavaScript: -

I. **Nature**:

- JavaScript: JavaScript is a programming language that is natively supported by web browsers. It is a core technology for web development and can be used to build entire web applications.
- jQuery: jQuery is not a programming language but a JavaScript library. It's written in JavaScript and provides a set of pre-written functions and methods that simplify common web development tasks.

II. **Abstraction**:

- JavaScript: JavaScript allows you to write code from scratch, giving you full control over your web application's behavior. It's a lower-level language and requires more code to accomplish certain tasks.
- jQuery: jQuery abstracts and simplifies many JavaScript tasks. It provides a high-level API for common operations, making it quicker and easier to achieve tasks like DOM manipulation, event handling, and Ajax requests.

III. **Syntax**:

- JavaScript: JavaScript has its own syntax and uses the native Document Object Model (DOM) API to manipulate HTML elements and perform various tasks.
- jQuery: jQuery uses a simplified syntax that often requires less code than raw JavaScript when performing common tasks. For example, selecting an element with jQuery is as simple as `$("#elementId"),` while in JavaScript, it typically involves `document.getElementById("elementId")`.

IV. Cross-Browser Compatibility:

- JavaScript: While JavaScript itself is a standard language, differences in how it's implemented by different web browsers can lead to compatibility issues. Developers often need to write browser-specific code or use polyfills to address these issues.
- jQuery: jQuery abstracts away many of the browser inconsistencies, providing a consistent API that works across various browsers, making it easier to write cross-browser compatible code.

V. Learning Curve:

- JavaScript: Learning JavaScript requires a deeper understanding of programming concepts, including data types, functions, loops, and more. It can be more challenging for beginners.
- jQuery: jQuery is often considered easier for beginners due to its simplified syntax and the fact that it handles many complex tasks behind the scenes.

VI. **Performance:**

- JavaScript: Writing JavaScript code directly allows for fine-grained control over performance optimization. Well-optimized JavaScript code can be highly efficient.
- jQuery: jQuery introduces a small overhead due to the library itself. While it's generally performant for common tasks, for highly optimized or specific tasks, writing plain JavaScript may be more efficient.

VII. **Flexibility:**

- JavaScript: JavaScript is more flexible and can be used for a wide range of development tasks, including building full-scale web applications and server-side development (Node.js).
- jQuery: jQuery is specialized for client-side web development tasks and doesn't provide the same level of flexibility as JavaScript for broader programming needs.

Feature	**JavaScript**	**jQuery**
Type	Programming language	JavaScript library
Purpose	To make web pages interactive	To make it easier to use JavaScript
Code length	Longer	Shorter
Cross-browser compatibility	More difficult	Easier
Learning curve	Steep	Gentle

5.1.3 HTML/CSS method of jQuery: -

Method	Description
addClass()	Adds one or more class names to selected elements
after()	Inserts content after selected elements
append()	Inserts content at the end of selected elements
appendTo()	Inserts HTML elements at the end of selected elements
attr()	Sets or returns attributes/values of selected elements
before()	Inserts content before selected elements
clone()	Makes a copy of selected elements
css()	Sets or returns one or more style properties for selected elements
detach()	Removes selected elements (keeps data and events)
empty()	Removes all child nodes and content from selected elements
hasClass()	Checks if any of the selected elements have a specified class name
height()	Sets or returns the height of selected elements
html()	Sets or returns the content of selected elements
innerHeight()	Returns the height of an element (includes padding, but not border)
innerWidth()	Returns the width of an element (includes padding, but not border)
insertAfter()	Inserts HTML elements after selected elements
insertBefore()	Inserts HTML elements before selected elements
offset()	Sets or returns the offset coordinates for selected elements (relative to the document)
offsetParent()	Returns the first positioned parent element
outerHeight()	Returns the height of an element (includes padding and border)
outerWidth()	Returns the width of an element (includes padding and border)
position()	Returns the position (relative to the parent element) of an element

prepend()	Inserts content at the beginning of selected elements
prependTo()	Inserts HTML elements at the beginning of selected elements
prop()	Sets or returns properties/values of selected elements
remove()	Removes the selected elements (including data and events)
removeAttr()	Removes one or more attributes from selected elements
removeClass()	Removes one or more classes from selected elements
removeProp()	Removes a property set by the prop() method
replaceAll()	Replaces selected elements with new HTML elements
replaceWith()	Replaces selected elements with new content
scrollLeft()	Sets or returns the horizontal scrollbar position of selected elements
scrollTop()	Sets or returns the vertical scrollbar position of selected elements
text()	Sets or returns the text content of selected elements
toggleClass()	Toggles between adding/removing one or more classes from selected elements
unwrap()	Removes the parent element of the selected elements
val()	Sets or returns the value attribute of the selected elements (for form elements)
width()	Sets or returns the width of selected elements
wrap()	Wraps HTML element(s) around each selected element
wrapAll()	Wraps HTML element(s) around all selected elements
wrapInner()	Wraps HTML element(s) around the content of each selected element

5.1.4 jQuery selector: -

Selector	Example	Selects
*	$("*")	All elements
#id	$("#lastname")	The element with id="lastname"
.class	$(".intro")	All elements with class="intro"
.class,.class	$(".intro,.demo")	All elements with the class "intro" or "demo"
element	$("p")	All <p> elements
el1,el2,el3	$("h1,div,p")	All <h1>, and <p> elements
:first	$("p:first")	The first <p> element
:last	$("p:last")	The last <p> element
:even	$("tr:even")	All even <tr> elements
:odd	$("tr:odd")	All odd <tr> elements
:first-child	$("p:first-child")	All <p> elements that are the first child of their parent
:first-of-type	$("p:first-of-type")	All <p> elements that are the first <p> element of their parent
:last-child	$("p:last-child")	All <p> elements that are the last child of their parent
:last-of-type	$("p:last-of-type")	All <p> elements that are the last <p> element of their parent
:nth-child(n)	$("p:nth-child(2)")	All <p> elements that are the 2nd child of their parent
:nth-last-child(n)	$("p:nth-last-child(2)")	All <p> elements that are the 2nd child of their parent, counting from the last child
:nth-of-type(n)	$("p:nth-of-type(2)")	All <p> elements that are the 2nd <p> element of their parent

:nth-last-of-type(n)	$("p:nth-last-of-type(2)")	All <p> elements that are the 2nd <p> element of their parent, counting from the last child
:only-child	$("p:only-child")	All <p> elements that are the only child of their parent
:only-of-type	$("p:only-of-type")	All <p> elements that are the only child, of its type, of their parent
parent > child	$("div > p")	All <p> elements that are a direct child of a element
parent descendant	$("div p")	All <p> elements that are descendants of a element
element + next	$("div + p")	The <p> element that are next to each elements
element ~ siblings	$("div ~ p")	All <p> elements that appear after the element
:eq(index)	$("ul li:eq(3)")	The fourth element in a list (index starts at 0)
:gt(no)	$("ul li:gt(3)")	List elements with an index greater than 3
:lt(no)	$("ul li:lt(3)")	List elements with an index less than 3
:not(selector)	$("input:not(:empty)")	All input elements that are not empty
:header	$(":header")	All header elements <h1>, <h2> ...
:animated	$(":animated")	All animated elements
:focus	$(":focus")	The element that currently has focus
:contains(text)	$(":contains('Hello')")	All elements which contains the text "Hello"
:has(selector)	$("div:has(p)")	All elements that have a <p> element
:empty	$(":empty")	All elements that are empty

:parent	$(":parent")	All elements that are a parent of another element
:hidden	$("p:hidden")	All hidden <p> elements
:visible	$("table:visible")	All visible tables
:root	$(":root")	The document's root element
:lang(language)	$("p:lang(de)")	All <p> elements with a lang attribute value starting with "de"
[attribute]	$("[href]")	All elements with a href attribute
[attribute=value]	$("[href='default.htm']")	All elements with a href attribute value equal to "default.htm"
[attribute!=value]	$("[href!='default.htm']")	All elements with a href attribute value not equal to "default.htm"
[attribute$=value]	$("[href$='.jpg']")	All elements with a href attribute value ending with ".jpg"
[attribute|=value]	$("[title|='Tomorrow']")	All elements with a title attribute value equal to 'Tomorrow', or starting with 'Tomorrow' followed by a hyphen
[attribute^=value]	$("[title^='Tom']")	All elements with a title attribute value starting with "Tom"
[attribute~=value]	$("[title~='hello']")	All elements with a title attribute value containing the specific word "hello"
[attribute*=value]	$("[title*='hello']")	All elements with a title attribute value containing the word "hello"
:input	$(":input")	All input elements
:text	$(":text")	All input elements with type="text"

:password	$(":password")	All input elements with type="password"
:radio	$(":radio")	All input elements with type="radio"
:checkbox	$(":checkbox")	All input elements with type="checkbox"
:submit	$(":submit")	All input elements with type="submit"
:reset	$(":reset")	All input elements with type="reset"
:button	$(":button")	All input elements with type="button"
:image	$(":image")	All input elements with type="image"
:file	$(":file")	All input elements with type="file"
:enabled	$(":enabled")	All enabled input elements
:disabled	$(":disabled")	All disabled input elements
:selected	$(":selected")	All selected input elements
:checked	$(":checked")	All checked input elements

5.2 Events of jQuery

5.3.1 Basic events: -

Method / Property	Description
bind()	Deprecated in version 3.0. Use the on() method instead. Attaches event handlers to elements
blur()	Attaches/Triggers the blur event
change()	Attaches/Triggers the change event
click()	Attaches/Triggers the click event
dblclick()	Attaches/Triggers the double click event
delegate()	Deprecated in version 3.0. Use the on() method instead. Attaches a handler to current, or future, specified child elements of the matching elements
die()	Removed in version 1.9. Removes all event handlers added with the live() method
error()	Removed in version 3.0. Attaches/Triggers the error event
event.currentTarget	The current DOM element within the event bubbling phase
event.data	Contains the optional data passed to an event method when the current executing handler is bound
event.delegateTarget	Returns the element where the currently-called jQuery event handler was attached
event.isDefaultPrevented()	Returns whether event.preventDefault() was called for the event object
event.isImmediatePropagationStopped()	Returns whether event.stopImmediatePropagation() was called for the event object
event.isPropagationStopped()	Returns whether event.stopPropagation() was called for the event object
event.namespace	Returns the namespace specified when the event was triggered
event.pageX	Returns the mouse position relative to the left edge of the document
event.pageY	Returns the mouse position relative to the top edge of the document
event.preventDefault()	Prevents the default action of the event

event.relatedTarget	Returns which element being entered or exited on mouse movement
event.result	Contains the last/previous value returned by an event handler triggered by the specified event
event.stopImmediatePropagation()	Prevents other event handlers from being called
event.stopPropagation()	Prevents the event from bubbling up the DOM tree, preventing any parent handlers from being notified of the event
event.target	Returns which DOM element triggered the event
event.timeStamp	Returns the number of milliseconds since January 1, 1970, when the event is triggered
event.type	Returns which event type was triggered
event.which	Returns which keyboard key or mouse button was pressed for the event
focus()	Attaches/Triggers the focus event
focusin()	Attaches an event handler to the focusin event
focusout()	Attaches an event handler to the focusout event
hover()	Attaches two event handlers to the hover event
keydown()	Attaches/Triggers the keydown event
keypress()	Attaches/Triggers the keypress event
keyup()	Attaches/Triggers the keyup event
live()	Removed in version 1.9. Adds one or more event handlers to current, or future, selected elements
load()	Removed in version 3.0. Attaches an event handler to the load event
mousedown()	Attaches/Triggers the mousedown event
mouseenter()	Attaches/Triggers the mouseenter event
mouseleave()	Attaches/Triggers the mouseleave event
mousemove()	Attaches/Triggers the mousemove event
mouseout()	Attaches/Triggers the mouseout event
mouseover()	Attaches/Triggers the mouseover event

mouseup()	Attaches/Triggers the mouseup event
off()	Removes event handlers attached with the on() method
on()	Attaches event handlers to elements
one()	Adds one or more event handlers to selected elements. This handler can only be triggered once per element
$.proxy()	Takes an existing function and returns a new one with a particular context
ready()	Specifies a function to execute when the DOM is fully loaded
resize()	Attaches/Triggers the resize event
scroll()	Attaches/Triggers the scroll event
select()	Attaches/Triggers the select event
submit()	Attaches/Triggers the submit event
toggle()	Removed in version 1.9. Attaches two or more functions to toggle between for the click event
trigger()	Triggers all events bound to the selected elements
triggerHandler()	Triggers all functions bound to a specified event for the selected elements
unbind()	Deprecated in version 3.0. Use the off() method instead. Removes an added event handler from selected elements
undelegate()	Deprecated in version 3.0. Use the off() method instead. Removes an event handler to selected elements, now or in the future
unload()	Removed in version 3.0. Use the on() or trigger() method instead. Attaches an event handler to the unload event

5.3.2 How to fire event programmatically: -

you can fire an event programmatically on an element using the trigger() method. This method allows you to simulate events such as clicks, keypresses, or custom events.

```
$("#elementId").trigger("click");
```

```
<script>
    $(document).ready(function(){
      $("#myButton").on("click", function() {
        alert("Button Clicked!");
      });

      $("#myButton").trigger("click");
    });
</script>
```

If you need to fire an event without bubbling it up to parent elements, you can use the .triggerHandler() method instead of the .trigger() method.

```
$("#myButton").triggerHandler("click");
```

5.3.3 Custom logic on event fire: -

To execute custom logic when an event is fired programmatically in jQuery, you can define a custom event handler. Here's an example that demonstrates how to trigger a custom event and execute custom logic when it's fired:

```
<!DOCTYPE html>
<html>
<head>
  <title>Custom Event Example</title>
</head>
<body>
  <button id="myButton">Click Me</button>

  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script>
    $(document).ready(function(){
      // Define a custom event handler
      $("#myButton").on("customEvent", function() {
        alert("Custom event triggered!");
      });

      // Programmatically trigger the custom event
      $("#myButton").click(function() {
        $("#myButton").trigger("customEvent");
      });
    });
  </script>
</body>
</html>
```

5.3 jQuery Validation

5.3.1 Basic validation: -

Basic validation in web development refers to the process of checking user input or data to ensure it meets certain criteria or rules before allowing it to be processed or submitted. Validation helps prevent incorrect or malicious data from being used in an application. Common types of basic validation include checking for required fields, data format (e.g., email or phone number), and length constraints.

```
<!DOCTYPE html>
<html>
<head>
  <title>Basic Form Validation</title>
</head>
<body>
  <h1>Contact Us</h1>
  <form id="contactForm" onsubmit="return validateForm()">
    <label for="name">Name:</label>
    <input type="text" id="name" required>
    <br>
    <label for="email">Email:</label>
    <input type="email" id="email" required>
    <br>
    <label for="message">Message:</label>
    <textarea id="message" required></textarea>
    <br>
    <input type="submit" value="Submit">
  </form>
```

```
    <script>
        function validateForm() {
            // Get form input values
            var name = document.getElementById("name").value;
            var email = document.getElementById("email").value;
            var message = document.getElementById("message").value;

            // Basic validation
            if (name === "" || email === "" || message === "") {
                alert("Please fill in all the required fields.");
                return false; // Prevent form submission
            }
            // You can add more complex validation here (e.g., email format)

            // If validation passes, the form will be submitted
            return true;
        }
    </script>
</body>
</html>
```

5.3.2 Validation with jQuery validator: -

To achieve validation using the jQuery Validation Plugin, you'll need to include the jQuery library and the jQuery Validation Plugin.

```
<!DOCTYPE html>
<html>
<head>
  <title>Form Validation with jQuery Validation Plugin</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/jquery.validation/1.19.3/jquery.validate.min.js"></sc
ript>
  <script>
    $(document).ready(function() {
      $("#contactForm").validate({
        rules: {
          name: "required",
          email: {
            required: true,
            email: true
          },
          message: "required"
        },
        messages: {
          name: "Please enter your name",
          email: {
            required: "Please enter your email",
            email: "Please enter a valid email address"
          },
```

```
                message: "Please enter your message"
            },
            submitHandler: function(form) {
                alert("Form submitted successfully");
                form.reset();
            }
        });
    });
  </script>
</head>
<body>
  <h1>Contact Us</h1>
  <form id="contactForm">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name">
    <br>
    <label for="email">Email:</label>
    <input type="email" id="email" name="email">
    <br>
    <label for="message">Message:</label>
    <textarea id="message" name="message"></textarea>
    <br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

To provide validation rules for various form elements and specify custom error messages, you can use a variety of rules and options. Here's a summary of some common rules and options you can use for form validation:

- **'required':** This rule specifies that the field is required and cannot be left empty.
- **'minlength'** and '**maxlength':** You can specify the minimum and maximum length of input, such as minlength: 5 and maxlength: 10.
- **'email':** This rule checks whether the input is a valid email address.
- **'url'**: Validates that the input is a valid URL.
- **'number':** Ensures that the input is a valid number.
- **'digits'**: Validates that the input contains only digits.
- **'equalTo'**: Compares the input with another field to ensure they are equal, such as confirming a password.
- **'min'** and '**max':** Specify the minimum and maximum numeric values allowed.
- **'range':** Defines a range of acceptable numeric values.
- **'pattern'**: Allows you to specify a regular expression pattern for validation.
- **'accept':** Validates the file type for file input elements.
- **'remote':** Checks the input against a remote server for validation.

5.3.3 jQuery Function map(),grep(),extends, each, merge etc.

i. **map():** This function is used to transform elements in an array or object and create a new array of the results.

```
var numbers = [1, 2, 3, 4, 5];
var squaredNumbers = $.map(numbers, function(num) {
   return num * num;
}); //[1,4,9,16,25]
```

ii. **grep():** It filters elements in an array or object based on a given condition and returns a new array.

```
var numbers = [1, 2, 3, 4, 5];
var evenNumbers = $.grep(numbers, function(num) {
   return num % 2 === 0;
}); //[2,4]
```

iii. **extend():** This function is used to merge the contents of one or more objects into the target object.

```
var object1 = { a: 1, b: 2 };
var object2 = { b: 3, c: 4 };
$.extend(object1, object2); //{ a: 1, b: 3, c: 4 }
```

iv. **each():** It is used to iterate over elements in an array or object and apply a function to each element.

```
var numbers = [1, 2, 3, 4, 5];
$.each(numbers, function(index, value) {
   console.log("Element at index " + index + " is " + value);
});
```

v. **merge():** This function is used to merge two arrays together into a single array.

```
var array1 = [1, 2, 3];

var array2 = [4, 5, 6];

var mergedArray = $.merge(array1, array2);

//[1,2,3,4,5,6]
```

vi. **$.trim():** Removes leading and trailing whitespace from a string.
vii. **$.inArray():** Checks if a value exists in an array.
viii. **$.isArray():** Determines if an object is an array.
ix. **$.isNumeric():** Checks if a value is a number.
x. **$.isEmptyObject():** Checks if an object is empty (has no properties).
xi. **$.type():** Returns the data type of a value (e.g., "string," "number," "object").
xii. **$.each():** Iterates over an object or array and execute a callbacks function for each element.
xiii. **$.isPlainObject():** Checks if a variable is a plain JavaScript object.
xiv. **$.makeArray():** Converts an object or array-like object to JavaScript array.
xv. **$.parseJSON():** Parse a JSON string into a JavaScript object.
xvi. **$.stringify():** Serialize a JavaScript object into a JSON string.

5.4 Regular expression in jQuery

Regular expressions are a sequence of characters that define a search pattern. They can be used to search for and manipulate text and are often used in data validation.

jQuery provides a number of methods for using regular expressions, such as the '.match()', '.replace()', and '.test()' methods.

jQuery itself doesn't provide its own implementation of regular expressions. Instead, jQuery relies on JavaScript's built-in support for regular expressions. Regular expressions in jQuery are used in the same way as regular expressions in plain JavaScript. You can use jQuery to manipulate the DOM, and JavaScript (including regular expressions) to work with the data contained in the DOM elements.

- **.match():** Matches the regular expression against the text of the element.

```
<p id="text">This is a test string.</p>
$(document).ready(function() {
  var match = $("#text").text().match(/test/);
  if (match) {
    alert("The match was found: " + match[0]);
  }
});
```

- **.replace():** Replaces all matches of the regular expression in the text of the element with the replacement string.

```
<p id="text">This is a test string.</p>
$(document).ready(function() {
  var replacedText = $("#text").text().replace("test", "**replaced**");
  $("#text").html(replacedText);
});
```

- **.test():** Tests whether the regular expression matches the text of the element.

```
<p id="text">This is a test string.</p>
$(document).ready(function() {
  var match = $("#text").text().test(/test/);
  if (match) {
    alert("The match was found.");
  }
});
```

5.5 Callback functions

A callback function in jQuery is a function that is passed as an argument to another function, to be executed only after the current function has completed its execution. Callback functions are useful for handling asynchronous events, such as when you need to execute code after an AJAX request has completed.

Callback functions can also be used to handle other types of events, such as click events, keyup events, and scroll events. For example, the following code uses a callback function to handle the click event of a button.

Callback functions are a powerful tool for handling asynchronous events in jQuery. By using callback functions, you can ensure that your code is executed in the correct order, even when the events that you are handling occur asynchronously.

- Callback functions can be nested, meaning that you can pass a callback function as an argument to another callback function. This can be useful for handling complex asynchronous tasks.
- Callback functions can be used to delay the execution of code. For example, you can use a callback function to execute code after a certain amount of time has elapsed, or after a certain event has occurred.
- Callback functions can be used to cancel the execution of code. For example, you can use a callback function to cancel an AJAX request if it takes too long to complete.

```
$('#myButton').click(function() {
  // This function is a callback for the button click event
  alert('Button clicked!');
});
```

The $.Callbacks() function is internally used to provide the base functionality behind the jQuery $.ajax() and $.Deferred() components. It can be used as a similar base to define functionality for new components.

$.Callbacks() supports a number of methods including callbacks.add(),callbacks.remove(), callbacks.fire() and callbacks.disable().

5.6 Deferred & Promise object

Deferred and Promise objects are used to manage asynchronous operations, handle asynchronous data, and provide a way to respond to the results of those operations. They are integral to working with asynchronous code, such as AJAX requests. jQuery's Deferred and Promise objects are similar to the Promises/A+ standard in modern JavaScript.

- **Deferred object: -**

A deferred object in jQuery is a special object that represents the eventual completion (or failure) of an asynchronous operation. Deferred objects can be used to manage the flow of control in asynchronous code, and to ensure that code is executed in the correct order, even when the asynchronous operations that it depends on complete in a different order.

- **resolve():** Resolves the deferred object with a value.
- **reject():** Rejects the deferred object with an error.
- **notify():** Notifies the promise object of the progress of the asynchronous operation.
- **promise():** Creates a promise object from the deferred object.

- **Promise object: -**

A promise object in jQuery is a special object that represents the eventual result (or failure) of an asynchronous operation. Promise objects are similar to deferred objects, but they are more lightweight and easier to use. Promise objects are also designed to be interoperable with other JavaScript promise libraries, such as the Promise library that is built into the ES6 standard.

- **then():** Attaches a callback function to the promise object to handle the successful completion of the asynchronous operation.
- **fail():** Attaches a callback function to the promise object to handle the failure of the asynchronous operation.
- **done():** Attaches a callback function to the promise object to handle the successful completion of the asynchronous operation. This method is deprecated in jQuery 3.0 and should be avoided.
- **always():** Attaches a callback function to the promise object to handle the successful completion or failure of the asynchronous operation.