```
$("element subelement[attribute=value]:filter").doSomething(parameters)
```

Importing

```
<head>
        <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
</head>
```

Selecting Elements

Select By	Example
ID	$(“#header”)
Class	$(“.updated”)
Tag Name	$(“table”)
Combination	$(“table.user-list”) or $(“#footer ul.menu li”)
Basic Filters	:first, :last, :even, :odd
Content Filters	:empty , :contains(text), :has(selector)
Attribute Filters	[attribute], [attribute=value], [attribute!=value]
Forms	:input, :text, :submit, :password, :enabled, :checked

Actions

Action	Example
DOM Manipulation	before(), after(), append(), appendTo()
Attributes	addClass(), css(), attr(), html(), val(), text()
Events	click(), on(), bind(), unbind(), live()
Effects	hide(), fadeOut(), toggle(), animate()
AJAX	load(), get(), ajax(), post(), getJSON()

Events

`$("span#message").click(function(event){…});`
OR
`$("span#message").on("click", function(event) {…});`

Without function, it becomes a manual triggering of the event: `$("span#message").click()` clicks the span with ID message.

- `this` attribute is used in a event handler function to reference the element that's been selected

```
$( "p" ).click(function() {
        var htmlString = $( this ).html();
}
```

- `hide()` / `show()`: Hide or show elements.
- `toggle()`: Toggle visibility of elements.
- `fadeIn()` / `fadeOut()`: Fade in or out by changing opacity.
- `slideToggle()`: Toggle visibility with a sliding effect.
- `animate()`: Create custom animations by changing CSS properties over time.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>jQuery Demo</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <style>
    #message {
      padding: 20px;
      background-color: lightblue;
      border: 1px solid blue;
      display: none; /* Hidden initially */
    }

    .hidden {
      display: none;
    }

    .animated-box {
      width: 200px;
      height: 200px;
      background-color: lightgreen;
      margin-top: 20px;
    }
  </style>
</head>
<body>

  <button id="trigger">Click to Toggle Visibility</button>
  <button id="fadeToggle">Fade In/Out</button>
```

```
  <button id="slideToggle">Slide In/Out</button>
  <button id="animateBox">Animate Box</button>
  <button id="showMessage">Show Message</button>
  <button id="toggleClass">Toggle Class (Hidden)</button>

  <span id="message">This is a clickable span message!</span>

  <div class="animated-box"></div>

  <script>
    $(document).ready(function() {

      // Toggle visibility of the message using .click()
      $("#trigger").click(function() {
        $("span#message").toggle();  // Toggle visibility of the message
      });

      // Fade the message in/out using .fadeIn() and .fadeOut()
      $("#fadeToggle").click(function() {
        $("span#message").fadeToggle(1000);  // Toggle fade visibility
      });

      // Slide the message in/out using .slideToggle()
      $("#slideToggle").click(function() {
        $("span#message").slideToggle(1000);  // Toggle slide visibility
      });

      // Animate a box to move left and change its size
      $("#animateBox").click(function() {
        $(".animated-box").animate({
          left: '+=100px', // Move box to the right by 100px
          height: '+=50px', // Increase height by 50px
          width: '+=50px'  // Increase width by 50px
        }, 1000);
      });

      // Show the message when the Show Message button is clicked
      $("#showMessage").click(function() {
        $("span#message").show(1000);  // Show the message with a fade-in effect
      });

      // Toggle the 'hidden' class using .toggleClass()
      $("#toggleClass").click(function() {
        //$("span#message").toggleClass("hidden");  doesn't work because the other
functions add inline styles which take precedence
        $("span#message").css('display', 'none');
      });

      // Using `this` in an event handler to get the clicked element's content
      $("span#message").click(function() {
        var htmlString = $(this).html();
        alert("You clicked on: " + htmlString);  // Display the content of the
clicked span
```

```
      });

    });
  </script>

</body>
</html>
```

Promises

Producing Code (Creates the promise):

```
let fetchData = new Promise(function(resolve, reject) {
    let data = "data fetched";

    if (data) {
        resolve(data); // Data is successfully fetched
    } else {
        reject("Error fetching data");
    }
});
```

Consuming Code (Waits for the result):

```
fetchData.then(function(result) {
    console.log(result); // Logs "data fetched"
}).catch(function(error) {
    console.error(error); // Logs "Error fetching data" if rejected
});
```

1. `.then()` Method:

- The `.then()` method is primarily used to handle the resolution (success) of a promise.
- It takes two arguments:
  - The first function is called when the promise is resolved.
  - The second function is called when the promise is rejected (although this is less common and not best practice).

2. `.catch()` Method:

- The `.catch()` method is specifically designed to handle rejections (failures) of a promise.
- It can be used after a `.then()` to catch any errors that occurred in the promise chain.

Promise Object

- Two properties -> state and result

State	Result
Pending	Undefined
Fulfilled	Value
Rejected	Error

- Cannot be accessed directly, promise methods need to be used