# Learn jQuery: Introduction

## jQuery streamlines dynamic behavior

*jQuery* is a JavaScript library that streamlines the creation of dynamic behavior with predefined methods for selecting and manipulating DOM elements. It offers a simplified approach to implementing responsiveness and requires fewer lines of code to assign behaviors to DOM elements than traditional JavaScript methods.

```
//Selecting DOM elements and adding an event listener in JS
const menu = document.getElementById('menu');
const closeMenuButton = document.getElementById('close-menu-
button');
closeMenuButton.addEventListener('click', () => {
    menu.style.display = 'none';
});

//Selecting DOM elements and adding an event listener in
jQuery
$('#close-menu-button').on('click', () =>{
  $('#menu').hide();
});
```

## jQuery document ready

JavaScript code runs as soon as its file is loaded into the browser. If this happens before the DOM (Document Object Model) is fully loaded, unexpected issues or unpredictable behaviors can result.

jQuery's `.ready()` method waits until the DOM is fully rendered, at which point it invokes the specified callback function.

```
$(document).ready(function() {
  // This code only runs after the DOM is loaded.
  alert('DOM fully loaded!');
});
```

## jquery object variables start with

*jQuery objects* are typically stored in variables where the variable name begins with a `$` symbol. This naming convention makes it easier to identify which variables are jQuery objects as opposed to other JavaScript objects or values.

```
//A variable representing a jQuery object
const $myButton = $('#my-button');
```

## jQuery CDN Import

*jQuery* is typically imported from a CDN (Content Delivery Network) and added at the bottom of an HTML document in a `<script>` tag before the closing `</body>` tag.

The jQuery `<script>` tag must be listed before linking to any other JavaScript file that uses the jQuery library.

```
<html>
  <head></head>
  <body>
    <!-- HTML code -->

    <!--The jQuery library-->
    <script src="https://code.jquery.com/jquery-
3.3.1.min.js"></script>

    <!--Site-specific JavaScript code using jQuery-->
    <script src="script.js"></script>
  </body>
</html>
```

# Learn jQuery: Effects

## jquery slideToggle method

The jQuery `.slideToggle()` effect method combines the effects of the `.slideDown()` and `.slideUp()` methods. This causes the element to alternate between gradually appearing by sliding down and gradually disappearing by sliding up every time the event listener is triggered.

The `.slideToggle()` method takes an optional parameter that specifies the duration of the effect. If not specified, the default value is 400 milliseconds.

```
// The '#menu' element will alternate between
// gradually sliding down and gradually sliding up
// every time the '#menu-button' is clicked.
$('#menu-button').on('click', () => {
  $('#menu').slideToggle();
});
```

## jquery fadeIn effect method

The jQuery `.fadeIn()` effect method causes a hidden element to gradually appear on the page. The method takes *optional* parameters.

The first optional parameter specifies the duration of the fading effect. The default value is 400 milliseconds.

The second parameter specifies the name of an “easing” function that determines the speed of the fading effect at different points in the animation. The default value is ‘swing’, where the fade-in effect is slower at the beginning/end and faster in the middle.

```
// The '#menu' element will gradually appear
// on the page when the '#menu-button' is clicked.
$('#menu-button').on('click', () => {
  // equivalent to $('#menu').fadeIn(400, 'swing')
  $('#menu').fadeIn();
});
```

## jquery fadeOut effect method

The jQuery `.fadeOut()` effect method causes an element and the space it was occupying to gradually disappear from the page. The method takes *optional* parameters.

The first optional parameter specifies the duration of the fading effect. The default value is 400 milliseconds.

The second parameter specifies an “easing” function that determines the speed of the fading effect at different points in the animation. The default value is ‘swing’, where the fade-out effect is slower at the beginning/end and faster in the middle.

```
// The '#menu' element will gradually disappear from
// the page when the '#menu-button' is clicked.
$('#menu-button').on('click', () => {
  // equivalent to $('#menu').fadeOut(400, 'swing')
  $('#menu').fadeOut();
});
```

## jquery show effect

The jQuery `.show()` effect method causes an element, assuming it is hidden, to instantly appear on the page.

```
//Instantly reveals the '#menu' element when the '#show-menu-button' is clicked.
$('#show-menu-button').on('click', () => {
  $('#menu').show();
});
```

## jquery toggle effect

The jQuery `.toggle()` effect method combines the effects of the `.hide()` and `.show()` methods. Every time the event listener is triggered, the element will alternate between displayed on the page and hidden from the page.

```
//The '#menu' element will alternate between being displayed and hidden every time the '#menu-button' is clicked.
$('#menu-button').on('click', () =>{
  $('#menu').toggle();
});
```

## jquery effects

*jQuery Effects* are jQuery object methods used to add animation and dynamic behavior to page elements. Effects can be used to show or hide elements, fade elements in and out, and more.

```
// The .show() effect causes the #menu element to
// appear once the #menu-button element is clicked.
$('#menu-button').on('click', event => {
  $('#menu').show();
});
```

## jquery fadeToggle method

The jQuery `.fadeToggle()` effect method combines the effects of the `.fadeIn()` and `.fadeOut()` methods. The element will alternate between gradually disappearing and appearing every time the event listener is triggered. This method takes optional parameters.

The first optional parameter specifies the duration of the fading effect. The default value is 400 milliseconds.

The second parameter specifies the name of an “easing” function that determines the speed of the fading effect at different points in the animation. The default value is ‘swing’, where the fading effect is slower at the beginning/end and faster in the middle.

```
// The '#menu' element will alternate between
// gradually disappearing and gradually appearing
// on the page when the '#menu-button' is clicked.
$('#menu-button').on('click', () => {
  // equivalent to $('#menu').fadeToggle(400, 'swing')
  $('#menu').fadeToggle();
});
```

## jquery slideUp method

The jQuery `.slideUp()` effect method causes an element and the space it was occupying to gradually disappear from the page by sliding up its content.
This method takes an optional parameter that specifies the duration of the effect in milliseconds. If not specified, the default duration is 400 milliseconds.

```
// The '#menu' element will gradually disappear
// from the page by sliding up its content when
// the '#menu-button' is clicked.
$('#menu-button').on('click', () => {
  // slide up over half a second
  $('#menu').slideUp(500);
});
```

## jquery hide effect

The jQuery `.hide()` effect method causes an element and the space it was occupying to disappear instantly from the page. When executed, the browser will render the HTML as if the hidden element does not exist.

```
//The '#menu' element will disappear instantly from the page
when the '#hide-menu-button' is clicked.
$('#hide-menu-button').on('click', () => {
  $('#menu').hide();
});
```

## jquery slideDown method

The jQuery `.slideDown()` effect method causes a hidden element to gradually appear on the page by sliding down its content.
This method takes an optional parameter that specifies the duration of the effect in milliseconds. If not specified, the default value of 400 milliseconds is used.

```
// The '#menu' element will gradually appear on
// the page by sliding down its content when the
// '#menu-button' element is clicked.
$('#menu-button').on('click', () => {
  // menu appears over 400ms duration
  $('#menu').slideDown();
});
```

# Learn jQuery: Event Handlers

## jquery on event listeners

jQuery `.on()` event listeners support all common browser *event types* such as mouse events, keyboard events, scroll events and more.

```
// A mouse event 'click'
$('#menu-button').on('click', () => {
  $('#menu').show();
});

// A keyboard event 'keyup'
$('#textbox').on('keyup', () => {
  $('#menu').show();
});

// A scroll event 'scroll'
$('#menu-button').on('scroll', () => {
  $('#menu').show();
});
```

## jquery event object

In a jQuery event listener, an *event object* is passed into the event handler callback when an event occurs. The *event object* has several important properties such as `type` (the event type) and `currentTarget` (the individual DOM element upon which the event occurred).

```
// Hides the '#menu' element when it has been clicked.
$('#menu').on('click', event => {
  // $(event.currentTarget) refers to the '#menu' element
that was clicked.
  $(event.currentTarget).hide();
});
```

## jquery event.currentTarget attribute

In a jQuery event listener callback, the `event.currentTarget` attribute only affects the individual element upon which the event was triggered, even if the listener is registered to a group of elements sharing a class or tag name.

```
// Assuming there are several elements with the
// class 'blue-button', 'event.currentTarget' will
// refer to the individual element that was clicked.
$('.blue-button').on('click', event => {
  $(event.currentTarget).hide();
});
```

## jquery on method chaining

*jQuery* `.on()` event listener methods can be chained together to add multiple events to the same element.

```
// Two .on() methods for 'mouseenter' and
// 'mouseleave' events chained onto the
// '#menu-button' element.
$('#menu-button').on('mouseenter', () => {
  $('#menu').show();
}).on('mouseleave', () => {
  $('#menu').hide();
});
```

## jquery on method

The jQuery `.on()` method adds *event listeners* to jQuery objects and takes two arguments: a string declaring the event to listen for (such as 'click'), and the event handler callback function. The `.on()` method creates an event listener that detects when an event happens (for example: when a user clicks a button), and then calls the *event handler* callback function, which will execute any defined actions after the event has happened.

```
//The .on() method adds a 'click' event to the '#login'
element. When the '#login' element is clicked, the callback
function will be executed, which reveals the '#login-form'
to the user.

$('#login').on('click', () => {
  $('#login-form').show();
});
```

# Learn jQuery: Style Methods

## jQuery .animate

The jQuery `.animate()` method animates the transition of CSS properties to a new state.

The `.animate()` method can take a JavaScript object as its first argument, and the second optional parameter can be a number (in terms of milliseconds) or certain string keywords. The property names must be in *camelCase*, and all the values must be strings. If no second argument is provided, the default animation time will be `400` milliseconds.

```
$('.tile').on('mouseenter', event => {
  $('.tile-text').animate({
    color: '#FFFFFF',
    backgroundColor: '#000000'
  }, 300); // The animation will take place over 300
milliseconds
});
```

## jQuery .css method

The jQuery method `.css()` can be used to set CSS property values. To change a single CSS property, the method takes two string arguments: the first is the desired CSS property name and the second is the new value.

If the CSS property is not already applied to the object, it will add the new property and set it to the provided value. If the property already exists for the object, it will update the existing value of the property to the new value.

```
// Hovering over the '.button' element will change the
// text color to red.
$('.button').on('mouseenter', event => {
  $('.text').css('color', 'red');
});
```

## jQuery .toggleClass

The jQuery `.toggleClass()` method combines the effects of the `.addClass()` and `.removeClass()` methods. The string argument passed to `.toggleClass()` is the name of the desired class to toggle and has no period preceding it.

If the class is not currently present on the selected element(s), then `.toggleClass()` will add the class to it. If the class is currently present on the element(s), then `.toggleClass()` will remove the class from them.

```
$('.choice').on('click', () => {
  $('.choice').toggleClass('highlighted');
});
```

## jQuery .css modify multiple properties

The jQuery `.css()` method can be used to modify multiple CSS properties of an element. To change multiple CSS properties, the method takes a JavaScript object with key-value pairs, where the key is the desired CSS property name (written in camelCase), and the value is a string representing the new value.

In the example code block, the CSS properties `color`, `background-color`, and `font-size` will be modified for all `<h2>` tags.

```
$('h2').css({
  color: 'blue',
  backgroundColor: 'gray',
  fontSize: '24px'
});
```

## jQuery removeClass

The jQuery `.removeClass()` method removes a class from a selected element. The string argument passed into the `.removeClass()` method is the name of the desired class to be removed, and has no period preceding it.

```
// When the user's mouse exits the '.button' element,
// the 'button-active' class is removed from it.
$('.button').on('mouseleave', event => {
  event.currentTarget.removeClass('button-active');
});
```

## jQuery addClass

The jQuery `.addClass()` method adds a class to a selected element. The string argument passed into `.addClass()` is the name of the new class and has no period preceding it, as it is implied to be a class name.

It allows for changes to styles while keeping all style declarations in separate CSS.

```
// The 'button-active' class is added to the '.button'
elements
// when a user's mouse enters the '.button' elements.
$('.button').on('mouseenter', () => {
  $('.button').addClass('button-active');
});
```

# Learn jQuery: Traversing the DOM

## jQuery children

The jQuery `.children()` method returns all child elements of a selected parent element.

This method only applies to the direct children of the parent element, and not deeper descendents.

In the example code, `$('.parent').children()` would select all the `.item` elements.

```
<div class="parent">
  <div class="item">Child 1</div>
  <div class="item">Child 2</div>
  <div class="item">Child 3</div>
</div>
```

## jQuery .parent

The jQuery `.parent()` method returns the parent element of a jQuery object.

```
<ul>ul <!-- this is the parent of li's one, two, six and ul three -->
  <li class="one">li</li>
  <li class="two">li</li>
  <ul class="three"> <!-- this is the parent of li's four and five -->
    <li class="four">li</li>
    <li class="five">li</li>
  </ul>
  <li class="six">li</li>
</ul>
```

## jQuery .next

The jQuery `.next()` method targets the next element that shares the same parent element as the original element.

In the following HTML code, the element returned by `$('.two').next()` would be `<li class="three">Item three</li>`.

```
<ul>
  <li class="one">Item one</li>
  <li class="two">Item two</li>
  <li class="three">Item three</li>
</ul>
```

## jQuery .find()

In jQuery, the `.find()` method will find and return all descendent elements that match the selector provided as an argument.
This code block shows a snippet of HTML that has a simple shopping list. Using jQuery, the list items inside the shopping list can be selected. The `listItems` variable will be a jQuery object that contains the two list items from the shopping list.

```
/*
In HTML:
<ul id='shopping-list'>
        <li class='list-item'>Flour</li>
  <li class='list-item'>Sugar</li>
</ul>
*/

// jQuery:
const listItems = $('#shopping-list').find('.list-item');
```

## jQuery .siblings

The jQuery `.siblings()` method targets all of the sibling elements of a particular element.
`.siblings()` can be used to add a `selected` class to an element on click and remove it from all of its sibling elements, ensuring that only one element appears as “selected” at one time.

```
$('.choice').on('click', event => {
  // Remove the 'selected' class from any siblings
  $(event.currentTarget).siblings().removeClass('selected');
  // Adds 'selected' class to that element only.
  $(event.currentTarget).addClass('selected');
});
```

## jQuery .closest

The jQuery `.closest()` method travels up through the DOM tree to find the first (and closest) ancestor element matching a selector string.