IQRA Institute and Technologies

We will teach you web development, Magento development and ETL Testing
Taught by experience teachers.

JQuery

Table of Contents

1. Introduction to jQuery

What is jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It simplifies things like HTML document traversing and manipulation, event handling, animating, and AJAX, making it much easier to use JavaScript on your website.

Why use jQuery?

- Simplifies HTML DOM tree traversal and manipulation
- Simplifies event handling, animating, and AJAX
- Works across a multitude of browsers
- Extensible through plugins
- Large community and extensive documentation

jQuery vs Vanilla JavaScript

While modern JavaScript has evolved to include many features that jQuery pioneered, jQuery still offers advantages in terms of brevity and cross-browser compatibility.

Example:

```
// Vanilla JavaScript
document.getElementById('myElement').style.display = 'none';

// jQuery
$('#myElement').hide();
```

2. Getting Started

Including jQuery in Your Project

You can include jQuery in your HTML file using a CDN or by downloading the library.

Using a CDN:

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

Downloading and including locally:

```
<script src="path/to/jquery.min.js"></script>
```

Basic Syntax

The basic syntax of jQuery is:

```
$(selector).action()
```

- `$` defines/accesses jQuery
- `(selector)` finds HTML elements
- `action()` is the action to be performed on the element(s)

Document Ready

To ensure your code runs after the DOM is fully loaded:

```
$(document).ready(function() {
    // Your code here
});
```

```
// Shorthand
$(function() {
    // Your code here
});
```

3. Selectors

jQuery selectors allow you to select and manipulate HTML elements.

Basic Selectors

- `$("*")` - Selects all elements
- `$(this)` - Selects the current HTML element
- `$("p")` - Selects all `<p>` elements
- `$(".class")` - Selects all elements with class=“class”
- `$("#id")` - Selects the element with id=“id”

Attribute Selectors

- `$("[href]")` - Selects all elements with an href attribute
- `$("[href='#']")` - Selects all elements with href value equal to “#”
- `$("[href!='#']")` - Selects all elements with href value not equal to “#”
- `$("[href$='.pdf']")` - Selects all elements with href value ending in “.pdf”

Hierarchy Selectors

- `$("div p")` - Selects all `<p>` elements that are descendants of a `<div>` element
- `$("div > p")` - Selects all `<p>` elements where the parent is a `<div>` element
- `$("div + p")` - Selects the first `<p>` element that are placed immediately after `<div>` elements
- `$("div ~ p")` - Selects all `<p>` elements that are siblings of `<div>` elements

Filter Selectors

- `:first` - Selects the first matched element
- `:last` - Selects the last matched element
- `:even` - Selects even elements
- `:odd` - Selects odd elements
- `:eq(index)` - Selects the element at a specific index

Example:

```
$("p:first").css("background-color", "yellow");
$("li:odd").css("background-color", "lightgray");
```

4. DOM Manipulation

jQuery makes it easy to manipulate the Document Object Model (DOM).

Getting and Setting Content

- `text()` - Sets or returns the text content of selected elements
- `html()` - Sets or returns the content of selected elements (including HTML markup)
- `val()` - Sets or returns the value of form fields

Example:

```
// Get content
let txt = $("p").text();

// Set content
$("p").text("Hello World!");
$("div").html("<p>New content</p>");
$("input").val("New value");
```

Adding and Removing Elements

- `append()` - Inserts content at the end of the selected elements
- `prepend()` - Inserts content at the beginning of the selected elements
- `after()` - Inserts content after the selected elements
- `before()` - Inserts content before the selected elements
- `remove()` - Removes the selected elements
- `empty()` - Removes the child elements from the selected element

Example:

```
$("p").append(" <b>Appended text</b>");
$("p").prepend("<b>Prepended text</b> ");
$("img").after("After image");
$("img").before("Before image");
$("#div1").remove();
```

```
$("#div1").empty();
```

Manipulating Attributes

- `attr()` - Sets or returns the value of attributes
- `removeAttr()` - Removes the specified attribute

Example:

```
$("img").attr("src", "new-image.jpg");
$("a").attr("href", "https://www.example.com");
$("a").removeAttr("target");
```

CSS Manipulation

- `addClass()` - Adds one or more classes to the selected elements
- `removeClass()` - Removes one or more classes from the selected elements
- `toggleClass()` - Toggles between adding/removing classes from the selected elements
- `css()` - Sets or returns the style attribute

Example:

```
$("p").addClass("myClass");
$("p").removeClass("myClass");
$("p").toggleClass("myClass");
$("p").css("background-color", "yellow");
```

5. Events

jQuery provides methods to attach event listeners to elements.

Common Events

- `click()` - Click event
- `dblclick()` - Double-click event
- `mouseenter()` - Mouse enter event
- `mouseleave()` - Mouse leave event
- `keydown()` - Keydown event
- `keyup()` - Keyup event

- `submit()` - Form submit event
- `change()` - Input change event
- `resize()` - Browser window resize event
- `scroll()` - Scrolling event

Example:

```
$("button").click(function() {
  console.log("Button clicked!");
});

$("form").submit(function(event) {
  event.preventDefault();
  console.log("Form submitted!");
});
```

Event Object

The event object contains useful information about the triggered event.

Example:

```
$("button").click(function(event) {
  console.log(event.type);  // "click"
  console.log(event.which); // which mouse button was clicked
  console.log(event.target); // the clicked element
});
```

Event Delegation

Event delegation allows us to attach a single event listener to a parent element that will fire for all descendants matching a selector, whether those descendants exist now or are added in the future.

Example:

```
$("#parent").on("click", "button", function() {
  console.log("Button clicked!");
});
```

6. Effects and Animations

jQuery provides several built-in effects and the ability to create custom animations.

Basic Effects

- `hide()` and `show()` - Hides or shows the selected elements
- `toggle()` - Toggles between hiding and showing elements
- `fadeIn()` and `fadeOut()` - Fades in or fades out the selected elements
- `slideUp()` and `slideDown()` - Slides up or slides down the selected elements

Example:

```
$("#hide").click(function() {
  $("p").hide();
});

$("#show").click(function() {
  $("p").show();
});

$("#toggle").click(function() {
  $("p").toggle();
});
```

Custom Animations

The `animate()` method allows you to create custom animations for any numeric CSS property.

Example:

```
$("button").click(function() {
  $("div").animate({
    left: '250px',
    opacity: '0.5',
    height: '150px',
    width: '150px'
  });
});
```

Animation Queue

By default, jQuery creates a queue of animations. You can use the `stop()` method to stop animations before they finish.

Example:

```
$("#start").click(function() {
    $("div")
        .animate({height: "300px", opacity: "0.4"}, "slow")
        .animate({width: "300px", opacity: "0.8"}, "slow")
        .animate({height: "100px", opacity: "0.4"}, "slow")
        .animate({width: "100px", opacity: "0.8"}, "slow");
});

$("#stop").click(function() {
    $("div").stop();
});
```

7. AJAX

jQuery simplifies making AJAX requests to servers.

$.ajax() Method

The `$.ajax()` method is used to perform AJAX requests.

Example:

```
$.ajax({
    url: "https://api.example.com/data",
    method: "GET",
    dataType: "json",
    success: function(data) {
        console.log("Data received:", data);
    },
    error: function(xhr, status, error) {
        console.error("Error:", error);
    }
});
```

Shorthand Methods

jQuery provides several shorthand methods for common types of AJAX requests:

- `$.get()` - Perform a GET request
- `$.post()` - Perform a POST request

- `$.getJSON()` - Perform a GET request and expect JSON in return

Example:

```
$.get("https://api.example.com/data", function(data) {
   console.log("Data received:", data);
});

$.post("https://api.example.com/data", { name: "John", age: 30 }, function(data) {
   console.log("Data received:", data);
});
```

Loading Content

The `load()` method is a simple way to fetch data from the server and place it into the selected element.

Example:

```
$("#result").load("ajax/test.html");
```

8. Utilities

jQuery provides several utility functions to help with common programming tasks.

Array and Object Operations

- `$.each()` - Iterate over arrays or objects
- `$.map()` - Translate all items in an array or object to a new array
- `$.grep()` - Finds the elements of an array which satisfy a filter function

Example:

```
$.each([1, 2, 3], function(index, value) {
   console.log(index + ": " + value);
});

let numbers = [1, 2, 3, 4];
let squares = $.map(numbers, function(num) {
   return num * num;
});
```

Type Checking

- `$.isArray()` - Check if the argument is an array
- `$.isFunction()` - Check if the argument is a function
- `$.isNumeric()` - Check if the argument is numeric

Example:

```
console.log($.isArray([1, 2, 3]));  // true
console.log($.isFunction(function() {}));  // true
console.log($.isNumeric("123"));  // true
```

String Manipulation

- `$.trim()` - Remove whitespace from the beginning and end of a string

Example:

```
let str = "    Hello, World!    ";
console.log($.trim(str));  // "Hello, World!"
```

9. Plugins

jQuery's functionality can be extended through plugins. There are thousands of plugins available for various purposes.

Using Plugins

To use a plugin, include the plugin script after the main jQuery library:

```
<script src="jquery.min.js"></script>
<script src="jquery.plugin.js"></script>
```

Popular Plugins

- jQuery UI - Provides a set of user interface interactions, effects, widgets, and themes
- Slick - Fully responsive carousel
- Chosen - Makes long, unwieldy select boxes much more user-friendly
- DataTables - Adds advanced interaction controls to HTML tables

Creating a Simple Plugin

You can create your own jQuery plugins. Here's a simple example:

```
(function($) {
    $.fn.greenify = function() {
        this.css("color", "green");
        return this;
    };
}(jQuery));

// Usage
$("p").greenify();
```

10. Best Practices

Performance Optimization

- Cache jQuery selections
- Use chaining
- Use event delegation for dynamically created elements
- Minimize DOM manipulation

Example:

```
// Bad
$("button").on("click", function() {
    $(this).addClass("active");
    $(this).text("Clicked");
});

// Good
$("button").on("click", function() {
    $(this)
        .addClass("active")
        .text("Clicked");
});
```

Avoid Conflicts

Use `jQuery.noConflict()` to avoid conflicts with other libraries that use the `$` symbol:

```
jQuery.noConflict();
jQuery(document).ready(function($) {
```

```
    // Use $ here
});
```

Unobtrusive JavaScript

Separate your JavaScript from your HTML:

```
<!-- Bad -->
<a href="#" onclick="doSomething();">Click me</a>

<!-- Good -->
<a href="#" id="myLink">Click me</a>

<script>
$("#myLink").on("click", function(e) {
    e.preventDefault();
    doSomething();
});
</script>
```

Use HTTPS for CDN

When including jQuery from a CDN, use HTTPS to ensure security:

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

This concludes our comprehensive guide to jQuery for beginners. Remember, practice is key to mastering jQuery. Try out these concepts in your own projects, and don’t hesitate to consult the official jQuery documentation for more detailed information.