# jQuery Guide

## 5. jQuery

### 5.1 jQuery Introduction

#### 5.1.1 Use of jQuery

* **Simplifies DOM Manipulation**: It makes it easier to select, traverse, and manipulate HTML elements.
* **Handles Events Efficiently**: Simplifies attaching and managing events.
* **Cross-Browser Compatibility**: Ensures the code works consistently across different web browsers.
* **AJAX Support**: Offers easy methods to perform asynchronous HTTP requests.
* **Example**:
* $(document).ready(**function** () {  
   $("#hideButton").click(**function** () {  
   $("#content").hide();  
   });  
   $("#showButton").click(**function** () {  
   $("#content").show();  
   });  
  });

#### 5.1.2 Difference Between jQuery and JavaScript

* **JavaScript**: The core scripting language used for making web pages interactive.
* **jQuery**: A library that simplifies complex JavaScript operations like DOM manipulation and event handling.
* **Example**:
  + **JavaScript**:
  + document.querySelector("#myElement").style.color = "red";
  + **jQuery**:
  + $("#myElement").css("color", "red");

#### 5.1.3 HTML/CSS Methods of jQuery

* .html(): Gets or sets the HTML content of an element.
* $("#myDiv").html("<p>Updated Content</p>");
* .css(): Applies or retrieves CSS styles for an element.
* $("p").css("font-size", "18px");
* .addClass() and .removeClass(): Add or remove class(es) from an element.
* $("#myElement").addClass("highlight");  
  $("#myElement").removeClass("highlight");
* .attr(): Sets or gets the value of an attribute.
* $("#link").attr("href", "https://www.example.com");

#### 5.1.4 jQuery Selector

* **Basic Selectors**:
* $("p"); *// Selects all <p> elements*  
  $(".myClass"); *// Selects elements with the class 'myClass'*  
  $("#myId"); *// Selects the element with the id 'myId'*
* **Attribute Selectors**:
* $("input[type='text']"); *// Selects all input elements with type 'text'*
* **Filter Selectors**:
* $("li:first"); *// Selects the first <li> element*  
  $("li:even"); *// Selects all even <li> elements*

### 5.2 Events of jQuery

#### 5.2.1 Basic Events

* .click(): Triggered when an element is clicked.
* $("#myButton").click(**function** () {  
   alert("Button clicked!");  
  });
* .dblclick(): Triggered when an element is double-clicked.
* $("#myButton").dblclick(**function** () {  
   alert("Button double-clicked!");  
  });
* .mouseenter() and .mouseleave(): Detect mouse entering or leaving an element.
* $("#myDiv")  
   .mouseenter(**function** () {  
   $(**this**).css("background-color", "yellow");  
   })  
   .mouseleave(**function** () {  
   $(**this**).css("background-color", "white");  
   });

#### 5.2.2 How to Fire Event Programmatically

* **Trigger an Event**:
* $("#myButton").trigger("click");

#### 5.2.3 Custom Logic on Event Fire

* **Example**: Custom event handling logic.
* $("#inputField").on("keyup", **function** () {  
   **let** inputLength = $(**this**).val().length;  
   **if** (inputLength > 5) {  
   $("#warning").text("Input too long!");  
   } **else** {  
   $("#warning").text("");  
   }  
  });

### 5.3 jQuery Validation

#### 5.3.1 Basic Validation

* **Example**: Check if a text input is empty.
* $("#submitButton").click(**function** () {  
   **if** ($("#username").val() === "") {  
   alert("Username is required!");  
   }  
  });

#### 5.3.2 Validation with jQuery Validator

* **Example**: Using jQuery Validation Plugin.
* $("#myForm").validate({  
   rules: {  
   username: "required",  
   email: {  
   required: **true**,  
   email: **true**,  
   },  
   },  
   messages: {  
   username: "Please enter your username",  
   email: "Please enter a valid email address",  
   },  
  });

### 5.4 jQuery Functions: map(), grep(), extend(), each(), merge(), etc.

* .map(): Creates a new array with the results of calling a function on every array element.
* **let** numbers = [1, 2, 3];  
  **let** doubled = $.map(numbers, **function** (num) {  
   **return** num \* 2;  
  });  
  console.log(doubled); *// [2, 4, 6]*
* .grep(): Filters an array based on a specified condition.
* **let** numbers = [1, 2, 3, 4, 5];  
  **let** evenNumbers = $.grep(numbers, **function** (num) {  
   **return** num % 2 === 0;  
  });  
  console.log(evenNumbers); *// [2, 4]*
* $.extend(): Merge the contents of two or more objects into the first object.
* **let** obj1 = { name: "John" };  
  **let** obj2 = { age: 30 };  
  $.extend(obj1, obj2);  
  console.log(obj1); *// { name: "John", age: 30 }*
* .each(): Iterates over an array or object, executing a function for each matched element.
* $.each([1, 2, 3], **function** (index, value) {  
   console.log("Index: " + index + ", Value: " + value);  
  });
* $.merge(): Merges the contents of two arrays into the first array.
* **let** arr1 = [1, 2];  
  **let** arr2 = [3, 4];  
  **let** merged = $.merge(arr1, arr2);  
  console.log(merged); *// [1, 2, 3, 4]*

### 5.5 Regular Expressions in jQuery

* **Example**: Using Regex to select elements containing a specific pattern.
* $("input[value\*='pattern']").css("border", "1px solid red");

### 5.6 Callback Functions

* **Definition**: A function executed after another function completes.
* **Example**:
* **function** displayMessage() {  
   alert("Hello, world!");  
  }  
  $("#myButton").click(displayMessage); *// displayMessage is the callback*

### 5.7 Deferred & Promise Object

* **Definition**: Objects that represent the eventual completion (or failure) of an asynchronous operation.
* **Example**:
* $.get("data.json")  
   .done(**function** (data) {  
   console.log("Data received: ", data);  
   })  
   .fail(**function** () {  
   console.error("Error loading data");  
   });

### 5.8 AJAX

#### 5.8.1 What is AJAX?

* **AJAX (Asynchronous JavaScript and XML)**: A method to update parts of a web page without reloading the whole page.

#### 5.8.2 Use of AJAX

* **Example**: Loading content asynchronously.
* $.ajax({  
   url: "https://api.example.com/data",  
   method: "GET",  
   success: **function** (data) {  
   $("#content").html(data);  
   },  
   error: **function** () {  
   alert("Error fetching data");  
   },  
  });

#### 5.8.3 How to Send Data with AJAX Request

* **Example**: Using POST to send data.
* $.post("submit.php", { name: "John", age: 30 }, **function** (response) {  
   console.log("Server response: ", response);  
  });

#### 5.8.4 Difference Between GET, POST, PUT, DELETE Methods

* **GET**: Retrieve data from the server.
* **POST**: Send new data to the server.
* **PUT**: Update existing data on the server.
* **DELETE**: Remove data from the server.

#### 5.8.5 JSON Data

* **Definition**: JavaScript Object Notation, a format for structuring data.
* **Example**:
* **let** jsonData = { name: "John", age: 30 };  
  console.log(JSON.stringify(jsonData)); *// Converts object to JSON string*

#### 5.8.6 Serialization & De-Serialization

* **Serialization**: Converting an object into a string format.
* **let** serializedData = JSON.stringify({ key: "value" });
* **De-Serialization**: Converting a string back to an object.
* **let** deserializedData = JSON.parse(serializedData);

### 5.9 Document Ready Function

* **Definition**: Ensures that the code runs only after the DOM is fully loaded.
* **Example**:
* $(document).ready(**function** () {  
   console.log("DOM is fully loaded!");  
  });