1. How to hide html element using JavaScript? jQuery?

To hide an HTML element using JavaScript, you can manipulate the element's `style` property to change its display to "none".

Example

Let's assume you have an HTML element with the ID "myElement" that you want to hide.

html

<!DOCTYPE html>

<html>

<body>

<p>This is a paragraph.</p>

<div id="myElement">This element will be hidden.</div>

<button onclick="hideElement()">Hide Element</button>

<script>

function hideElement() {

var elementToHide = document.getElementById("myElement");

elementToHide.style.display = "none";

}

</script>

</body>

</html>

```

jQuery:

jQuery makes it even simpler to hide elements using its `hide()` method.

Example:

Let's use the same "myElement" ID to demonstrate how you can hide an element using jQuery.

Include jQuery library in the HTML:

```html

<!DOCTYPE html>

<html>

<head>

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

</head>

<body>

<p>This is a paragraph.</p>

<div id="myElement">This element will be hidden.</div>

<button onclick="hideElement()">Hide Element</button>

<script>

function hideElement() {

$("#myElement").hide();

}

</script>

</body>

</html>

Both the JavaScript and jQuery methods achieve the same result, hiding the HTML element with the ID "myElement" when the corresponding button is clicked. The JavaScript method directly manipulates the `style` property of the element, while jQuery provides a shorthand method `hide()` to achieve the same effect.

2. How to select all textboxes using jQuery selector?

Hint : Without using classname / id

To select all textboxes without using a class or an ID, you can target them by their element type. In this case, textboxes are represented by the `<input>` element with the `type="text"` attribute. Using jQuery, you can select all textboxes by using the attribute selector.

Here is an example of how to select all textboxes using jQuery:

<!DOCTYPE html>

<html>

<head>

<title>Select All Textboxes using jQuery</title>

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

<script>

$(document).ready(function() {

// Select all textboxes using the attribute selector for input elements with type="text"

var allTextboxes = $('input[type="text"]');

// Adding a class for demonstration purposes - you can perform operations on selected elements here

allTextboxes.addClass('selectedTextboxes');

// Example: Change the border color of all selected textboxes

allTextboxes.css('border', '1px solid red');

});

</script>

</head>

<body>

<input type="text" placeholder="Textbox 1">

<input type="text" placeholder="Textbox 2">

<input type="text" placeholder="Textbox 3">

</body>

</html>

In this code, the `$('input[type="text"]')` selector targets all input elements that have the attribute `type="text"`. It selects all textboxes on the page and adds a class `selectedTextboxes` for demonstration purposes. You can perform any desired operations or apply any changes to the selected textboxes using jQuery methods once they are selected.

3. What is the difference between "this" and $(this) in jquery programming?

In jQuery programming, `this` and `$(this)` are related but serve different purposes:

- `this` in jQuery: Refers to the DOM element that is currently being operated on or the context in which a function is executed. It represents the raw, underlying DOM element without any jQuery methods or properties. `this` is a fundamental concept in JavaScript, and in the context of jQuery, it refers to the specific DOM element on which an event has occurred or the function is being executed.

Example:

javascript

$('button').click(function() {

// 'this' refers to the DOM element that triggered the click event

$(this).css('background-color', 'red');

});

In this example, within the click event handler function, `this` refers to the specific button DOM element that triggered the click, allowing you to manipulate it.

`$(this)` in jQuery:\*\* Represents the jQuery object that encapsulates the DOM element referred to by `this`. When `this` (referring to the DOM element) is wrapped with the jQuery `$()` function, it creates a jQuery object. This allows you to access and utilize jQuery methods and functions on the specific DOM element represented by `this`.

Example:

javascript

$('button').click(function() {

// $(this) wraps 'this' in a jQuery object, enabling the use of jQuery methods

$(this).css('background-color', 'red');

$(this).hide();

});

```

Here, `$(this)` creates a jQuery object, allowing you to apply jQuery methods like `.css()`, `.hide()`, and others to the element referenced by `this`.

In summary, `this` refers directly to the DOM element, while `$(this)` wraps that DOM element in a jQuery object, enabling the use of jQuery methods and functionalities on that element.