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

You can hide an HTML element in JavaScript by changing its CSS style. There are a few common methods to do this. Here's one using the `style` property:

```
// Assuming you have an HTML element with an ID like "myElement"
var element = document.getElementById("myElement");
element.style.display = "none"; // This will hide the element
```

Alternatively, you can use the `style` property to change other CSS properties, such as `visibility` or `opacity, to hide elements in different ways:

```
element.style.visibility = "hidden"; // Hides the element but preserves the space it occupies
element.style.opacity = 0; // Makes the element transparent, effectively hiding it
```

If you're using jQuery, you can hide elements more conveniently:

```
// Using jQuery to hide an element by ID
$("#myElement").hide();
```

The `hide()` function in jQuery will set the `display` property to "none" by default, but it can be configured to use other hiding mechanisms as well.

2. How to select all textboxes using jQuery selector?
Hint : Without using classname / id

You can select all textboxes without using class names or IDs by using the `:input` selector in combination with the `type` attribute. Textboxes have a `type` attribute set to "text," so you can select them using this combination. Here's how you can do it:

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

This code will select all input elements with the `type` attribute set to "text," which includes textboxes.

If you want to perform actions on these selected textboxes, you can loop through them using `.each()`. For example:

```
$(":input[type='text']").each(function() {
  // Do something with each textbox
  console.log($(this).val()); // This prints the value of each textbox
});
```

This code will loop through all the selected textboxes and perform the specified action for each one.

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

In jQuery programming, there is no significant difference between "this" and "$this" in terms of core functionality. Both refer to the current DOM element being operated on within a jQuery context. However, the usage and conventions around these two can differ based on coding styles and preferences.

1. `this`: "this" is a keyword in JavaScript and represents the current context or object. When used within a jQuery event handler, it typically refers to the DOM element that triggered the event. For example:

```
$("button").click(function() {
  // 'this' refers to the clicked <button> element
  $(this).text("Clicked!");
});
```

2. `$this`: "$this" is not a standard construct in jQuery, but it could be used as a variable name to store a reference to the current jQuery-wrapped DOM element for easier access, especially in situations where you may need to use it multiple times within a function. For example:

```
$("button").click(function() {
  var $this = $(this); // Storing the jQuery object for convenience
  $this.text("Clicked!");
  // You can use $this throughout the function
  $this.css("background-color", "red");
});
```

The use of "$this" can make your code more readable and can help prevent unnecessary rewrapping of the same element, but it's a matter of coding style and is not a requirement. You can choose to use "this" directly or "$this" based on your preferences and the specific needs of your code.