**JQUERY (John Resig)**

The JavaScript library that prevents our fingers from breaking. It's a library, so a bunch of code that somebody else wrote that you can incorporate into your own projects and use to improve your own projects or to make your life simply a lot easier. DOM of our website, we're selecting the document and we're querying for the selector h1. document.queySelector(“h1”); So we're looking for the h1 element in our web page. Now if we were to use the jQuery library, then we can get rid of all of that and simply say jQuery("h1"). And if you want to be even shorter, the shorthand way to write jQuery is simply a dollar sign. So $("h1") does exactly the same as document.querySelector ("h1").

$("h1").css("color","red");

*//selecting the h1 element and changing its css properties mainly color to red.*

*//Note: add the jQuery cdn before the js script otherwise it will not work.*

 <script *src*="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>

    <script *src*="./index.js" *charset*="utf-8"></script>

Now the other way that it might mess up is that if you had both of the script tags inside the head section of your website, then when you refresh your website you'll notice that again, it doesn't work. And in this case, it's not even giving you a proper error. And you have to be really careful about including the CDN for our jQuery library in the head section of your website, because what might happen is that before we've actually finished loading up the entire jQuery library, our index.js might already be trying to access it. So one of the ways that you can safeguard against this is to check whether if our jQuery library is ready, and we do that by using the dollar sign to select our document and then we use the ready() method and we have a callback function in here.

$(document).ready(function(){

    $("h1").css("color","red");

});

And that means that once our document is ready then it's going to call this code. So now if we hit Save and we refresh, then you can see our code now works because we waited until our jQuery library has finished loading and then we access the h1 to change its color.

As discussed we can use a $ instead of document.querySelector using jQuery.

console.log($("h1").css("color"));

this will tell about the details of h1 color property . And the output will be shown in the console as rgb(0,0,0) which signifies it to be a black color. Similary we can do this with font=-size and any properties too.

.big-title

{

    font-size: 5rem;

    color: yellow;

    font-family: cursive;

}

$("h1").addClass("big-title");



so here we are trying to say that it is advisable to style things in css rather than doing the styling stuffs in js we can add the properties in js. This is done in order to seregate codes and make our styling and functionalities separately.

 **Dynamic Styles:** If styles need to be applied based on user interaction or data, JavaScript might be necessary.

 **Complex Animations:** For intricate animations that require calculations or timing functions, JavaScript can be used in conjunction with CSS animations.

In summary, using CSS for styling is the recommended approach for better performance, maintainability, and reusability. JavaScript can be used for dynamic styling or complex animations when CSS alone isn't sufficient.

As we have seen that we can addClass , similary we can remove the class using

$("h1").removeClass("big-title");

which will remove the properties of the big-title and our h1 element will be now



also we can add and remove multiple classes just add a space between the classes.

$("h1").addClass("big-title margin-50");

.hasClass it returns a Boolean value i.e. true or false it checks wether the element has the class or not . Ex:

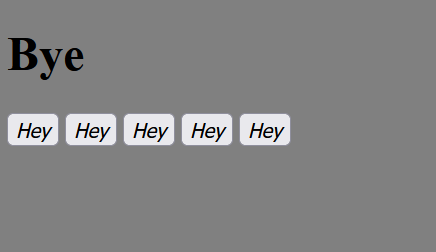


$("h1").text("Bye");

*// it changes the text of the element h1 also suppose if there are many h1's then it will change their text too not just one h1 but the entire h1 element present.*

$("button").html("<em>Hey</em>");

*//so this .html is the shortform of innerHTML and throught we can change the html as well as we have done earlier*



if we use here .text inplace of .html then it will assume it to be a text and it will print <em>Hey</em> and ignore all the html tags.

Now by this point, you would have noticed that most of the things that you do with jQuery are implemented using methods, and we’re accessing it through the dot notation.

You're absolutely right! The jQuery .attr() method is a versatile tool for working with attributes of HTML elements. Here's a breakdown of how it's used for setting and getting attributes, along with the example you provided:

**Setting Attributes:**

The .attr() method can be used to set the value of an attribute on one or more selected elements. It takes two arguments:

1. **attributeName (String):** The name of the attribute you want to set.
2. **value (String or Function):** The value to assign to the attribute. You can also pass a function that calculates the value based on the current element.

**Example:**

JavaScript

$("img").attr("src", "new\_image.jpg");

This code selects all <img> elements and sets their src attribute to "new\_image.jpg". This will replace the existing image source with the new one.

**Getting Attributes:**

The .attr() method can also be used to retrieve the current value of an attribute. In this case, it only returns the value for the first matched element.

**Example:**

JavaScript

var currentHref = $("a").attr("href");

console.log(currentHref);

This code selects the first <a> element and retrieves the value of its href attribute. The value is then stored in the currentHref variable and logged to the console.

**Dynamic Image Loading (Your Example):**

In your example, $("a").attr("src"); retrieves the src attribute value of the first <a> element. This is likely used in a scenario where you want to dynamically load an image based on a user action or data fetched from an external source.

Here's a possible scenario:

HTML

<a id="imageLink" href="#">Click to load image</a>

JavaScript

$("#imageLink").click(function() {

var imageUrl = "image\_from\_data.jpg"; // Image URL obtained dynamically

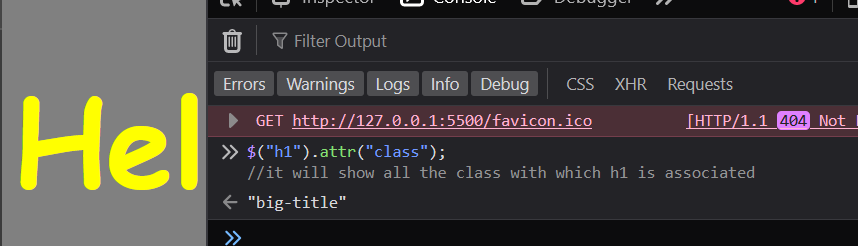
$("img").attr("src", imageUrl);

});

In this example, clicking the <a> element triggers a JavaScript function. The function retrieves the image URL (which could be from user input, database, etc.) and assigns it to the src attribute of the <img> element using .attr(). This dynamically loads the image based on user interaction.

$("h1").attr("class");

*//it will show all the class with which h1 is associated*



*// to change the color from yellow to purple using pure js*

for(var i = 0 ; i <  5 ; i++ )

    {

        document.querySelectorAll("button")[i].addEventListener("click",function()

    {

        document.querySelector("h1").style.color = "purple";

    });

    }

*// so in js we have to write this bunch of code first we have to run the for loop to iterate through all the button and add the function to it that when we click the button it should change the color of the h1 element from yellow to purple*

*// now trying to implement this idea using jQuery*

$("button").click(function()

{

    $("h1").css("color","purple");

});

Your analysis of both approaches is spot on! Here's a breakdown of the code and the advantages of jQuery in this scenario:

**Pure JavaScript Approach:**

The code iterates through all buttons (document.querySelectorAll("button")) using a loop (0 to 4). For each button, it adds an event listener (addEventListener) that changes the h1 element's color to purple when clicked.

**Advantages:**

* Works without external libraries.

**Disadvantages:**

* Repetitive code for each button.
* Can become cumbersome for many buttons.

**jQuery Approach:**

The jQuery code is more concise and efficient. It selects all buttons with $("button") and attaches a single click event listener using .click(). Inside the listener, it sets the h1 element's color to purple using .css("color", "purple").

**Advantages:**

* More concise and readable code.
* Single event listener for all buttons.
* Easier to maintain for many buttons.

**Summary:**

Both approaches achieve the same result of changing the h1 color on button click. However, jQuery offers a cleaner and more scalable solution, especially when dealing with multiple buttons.

Here are some additional points to consider:

* **Event Delegation:** With jQuery, you can use event delegation to attach a single listener to a parent element (like the container holding all buttons) and handle clicks on child buttons efficiently.
* **Specificity:** Make sure your CSS selector for the h1 element is specific enough to avoid unintended styling conflicts.

If you're using jQuery in your project, it's generally recommended to use its methods for event handling and DOM manipulation for better readability and maintainability.

**CHALLENGE EXERCISE**: to show that whenever I press or tap on the key on the website it should display on the website which will be enclosed within h1 tag.

$(document).keydown(function(*event*) {

*// Get the pressed key character*

    var pressedKey = *event*.key;

*// Update the h1 content with the pressed key*

    $("h1").text(pressedKey);

*// or we can simply write $("h1").text(event.key);*

  });

and the below code shows the key pressed in console

$(document).keydown(function(*event*)

{

    console.log(*event*.key);

});

$("h1").on("mouseover",function()

{

    $("h1").css("color","purple");

});

when we hover our mouse on the screen it will change the color of the h1 content.

You've got the right idea about the mouseover event! Here's a breakdown of the code, the difference between click and on, and some additional considerations:

**Hovering to Change Color:**

JavaScript

$("h1").on("mouseover", function() {

$("h1").css("color", "purple");

});

This code uses jQuery's .on() method to attach a mouseover event listener to the <h1> element. When the user hovers their mouse over the <h1> element, the function inside the curly braces is executed. This function changes the h1 element's color to purple using .css("color", "purple").

**Difference Between click and on:**

* **click:** This is a shorthand method specifically for attaching a click event listener. It's equivalent to $(".element").on("click", function() { ... }). Clicking on the element triggers the function.
* **on:** This is a more versatile method for attaching various event listeners to elements. You can specify different events like mouseover, mouseout (when mouse leaves), keydown, keyup, and more within the quotes after "on".

In your example, mouseover is used to detect hovering, while .click would be used to detect a mouse click on the <h1> element.

**Additional Considerations:**

* You can use .on() to attach multiple event listeners to the same element. For example, you could combine mouseover and mouseout to change the color on hover and back to the original color on mouseout.
* Remember to include the jQuery library in your HTML file for these methods to work.

By understanding the difference between click and on, you can create more interactive web pages with various user interactions.

$("h1").before("<button>New</button>");

*// it adds the button before h1 tag*

$("h1").after("<button>New</button>");

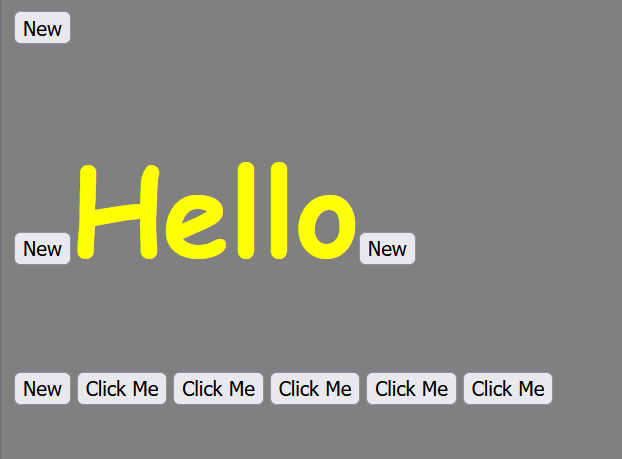
*// it adds the button after h1 tag*

$("h1").prepend("<button>New</button>");

*// it adds the button before the Content of the h1 tag and at the starting of the h1 tag i.e. <h1> <button>New</button>*

$("h1").append("<button>New</button>");

*// it adds the button after the Content of the h1 tag and before the closing tag of h1 i.e. <h1> <button>New</button>*



$("button").remove();

*// it will remove all the buttons from the webpage*

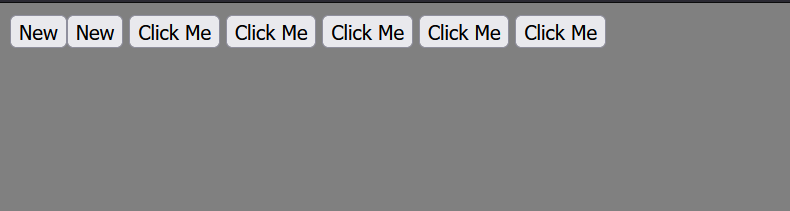


$("button").on("click",function(){

    $("h1").hide();

});

*// this will hide the content of the h1 and will remove it too from our webPage*



similarly there are many more methods like show , fadeOut , fadeIn , toogle , fadeToggle , slideUp , slideDown : it can be used to dropdown menu

You're absolutely right! The code you provided hides the <h1> element when the button is clicked. However, there's a distinction between hiding and removing elements:

* **Hiding:** The element is still present in the HTML but becomes invisible. You can use .show() to bring it back.
* **Removing:** The element is completely removed from the DOM (Document Object Model) and the HTML code.

Here's a breakdown of the jQuery methods you mentioned and their uses for creating dynamic and interactive web pages:

**Hiding and Showing:**

* .hide(): Hides the matched elements.
* .show(): Makes previously hidden elements visible again.
* .toggle(): Toggles the visibility of the matched elements (hide if visible, show if hidden).

**Fading Effects:**

* .fadeOut(): Gradually fades out the opacity of the matched elements to 0, making them transparent over time.
* .fadeIn(): Gradually fades in the opacity of the matched elements from 0 to 1, making them visible over time.
* .fadeToggle(): Toggles between .fadeOut() and .fadeIn().

**Sliding Effects:**

* .slideUp(): Hides the matched elements by sliding them up (reducing their height).
* .slideDown(): Shows the matched elements by sliding them down (increasing their height) from their hidden state.

**Use Cases:**

* **Dropdown Menus:** You can use .slideDown() to reveal a dropdown menu when the user hovers over a button and .slideUp() to hide it when they hover out.
* **Accordions:** Combine .slideToggle() with click events to create accordion menus where clicking a header expands or collapses its corresponding content.
* **Image Galleries:** Use .fadeOut() and .fadeIn() to create a smooth transition effect between images in a slideshow.
* **Show/Hide Content:** Dynamically show or hide content based on user actions (like clicking a button) using .show() and .hide().
* **Tabbed Content:** Employ these methods to switch between tabs, fading out the inactive content and fading in the selected content.

**Additional Tips:**

* You can control the animation speed of these methods by passing a value in milliseconds (e.g., .fadeOut(500)) or using named durations like "slow" or "fast".
* Consider using CSS transitions or animations for smoother and more customizable effects.

By effectively using these methods, you can enhance user experience and create dynamic interactions on your website.

**1. jQuery.animate()**

* **Functionality:** This method allows you to gradually change CSS properties (like opacity, width, height) of selected elements over a specified duration.
* **Advantages:**
  + Easier to use for simple animations without complex keyframes.
  + Offers built-in options for easing functions (how the animation progresses over time).
* **Disadvantages:**
  + Limited animation capabilities compared to CSS animations.
  + Can be less performant for complex animations.

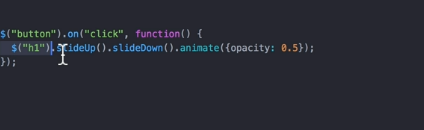
**Example:**

JavaScript

$("h1").animate({ opacity: 0.5 }, 1000); // Fade h1 to 50% opacity over 1 second

Now, the thing to remember about the animate method is that in between the curly braces, you can only add the CSS rules that have a numeric value. So that means that you can’t animate to something like a color, right? You can't change the color to red. And when I run this you can see that we get an error. Red is not defined. And this is because it's very difficult to create that kind of twin animation that gradually progresses from one style to another using the animate method. So you have to stick to things that have a numeric value. For example a margin of 20 pixels, that it can do. Or if you wanted to make it a percentage, then you can include it as a string. So 20% margin which makes it much bigger. But essentially make sure that this second value in the CSS that you're trying to animate to is a number value.

method chaining. So here first it will slideUp the conetent of the h1 and then slideDown the content of h1 and then it will make it opacity to 50% i.e. it will make the content a bit more lighter.