

Basic syntax is: **`$(selector).action()`**

The Document Ready Event

You might have noticed that all jQuery methods in our examples, are inside a document ready event:

```
$(document).ready(function(){  
    // jQuery methods go here...  
});
```

This is to prevent any jQuery code from running before the document is finished loading (is ready).

It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.

Here are some examples of actions that can fail if methods are run before the document is fully loaded:

- Trying to hide an element that is not created yet
- Trying to get the size of an image that is not loaded yet

Tip: The jQuery team has also created an even shorter method for the document ready event:

```
$(function(){  
    // jQuery methods go here...  
});
```

Use the syntax you prefer. We think that the document ready event is easier to understand when reading the code.



ChatGPT

The difference between `$('.button')` and `$(":button")` lies in the elements they select:

1. `$('.button')`: This jQuery selector selects all `<button>` elements in the HTML document. It specifically targets elements with the `<button>` tag.
2. `$(":button")`: This jQuery selector selects all button elements, including `<button>` elements as well as `<input>` elements with the `type="button"` attribute. It targets any element that can act as a button, regardless of whether it's a `<button>` tag or an `<input>` tag.

In summary, `$('.button')` specifically selects `<button>` elements, while `$(":button")` selects both `<button>` elements and `<input>` elements with `type="button"`.



more Examples of jQuery selectors

Syntax	Description	Example
<code>\$("*")</code>	Selects all elements	Try it
<code>\$(this)</code>	Selects the current HTML element	Try it
<code>\$(".p.intro")</code>	Selects all <p> elements with class="intro"	Try it
<code>\$("p.first")</code>	Selects the first <p> element	Try it
<code>\$("ul li:first")</code>	Selects the first element of the first 	Try it
<code>\$("ul li:first-child")</code>	Selects the first element of every 	Try it
<code>\$("[href]")</code>	Selects all elements with an href attribute	Try it
<code>\$("a[target='_blank']")</code>	Selects all <a> elements with a target attribute value equal to "_blank"	Try it
<code>\$("a[target!='_blank']")</code>	Selects all <a> elements with a target attribute value NOT equal to "_blank"	Try it
<code>\$(".button")</code>	Selects all <button> elements and <input> elements of type="button"	Try it
<code>\$("tr:even")</code>	Selects all even <tr> elements	Try it
<code>\$("tr:odd")</code>	Selects all odd <tr> elements	Try it

Example

```
$( "p" ).on({
  mouseenter: function(){
    $(this).css("background-color", "lightgray");
  },
  mouseleave: function(){
    $(this).css("background-color", "lightblue");
  },
  click: function(){
    $(this).css("background-color", "yellow");
  }
});
```

[Try it Yourself »](#)

`.show()`, `.toggle()`, `.hide()`, `.css()`.

Note that `$('.parent div')` targets all the div that are Descendants of `.parent` while `$('.parent > div')` targets only the Direct child of `.parent`



ChatGPT

The ``contenteditable`` attribute is commonly used in various scenarios across web development to enable inline editing, rich-text editing, and interactive content creation. Here are some common use cases where ``contenteditable`` is employed:

1. **Rich-text Editors:** Many online platforms and content management systems (CMS) utilize ``contenteditable`` to create rich-text editors similar to desktop word processors. Users can format text, insert images, and create structured content directly within the web page.
2. **Inline Editing:** Websites often allow users to edit content inline, directly within the page, without navigating to a separate editing interface. Elements such as text blocks, titles, and descriptions can be made editable using ``contenteditable``, providing a seamless editing experience.
3. **Comments and Discussions:** Social media platforms, forums, and commenting systems often use ``contenteditable`` to enable users to write and edit comments or posts directly within the conversation thread.
4. **Interactive Widgets:** Interactive elements such as notes, to-do lists, and sticky notes can be made editable using ``contenteditable``, allowing users to modify their content dynamically.
5. **Document Editing:** Online document collaboration tools and productivity suites employ ``contenteditable`` to enable multiple users to edit documents simultaneously in real-time, similar to Google Docs.
6. **Custom Input Fields:** In some cases, developers use ``contenteditable`` to create custom input fields or interactive elements that require user input, such as form fields or dynamically generated content.

Overall, ``contenteditable`` offers a flexible and intuitive way to enable user interaction and content editing directly within web pages, enhancing the usability and interactivity of web applications.



Animations:

SlideToggle, fadeToggle, show, hide, toggle