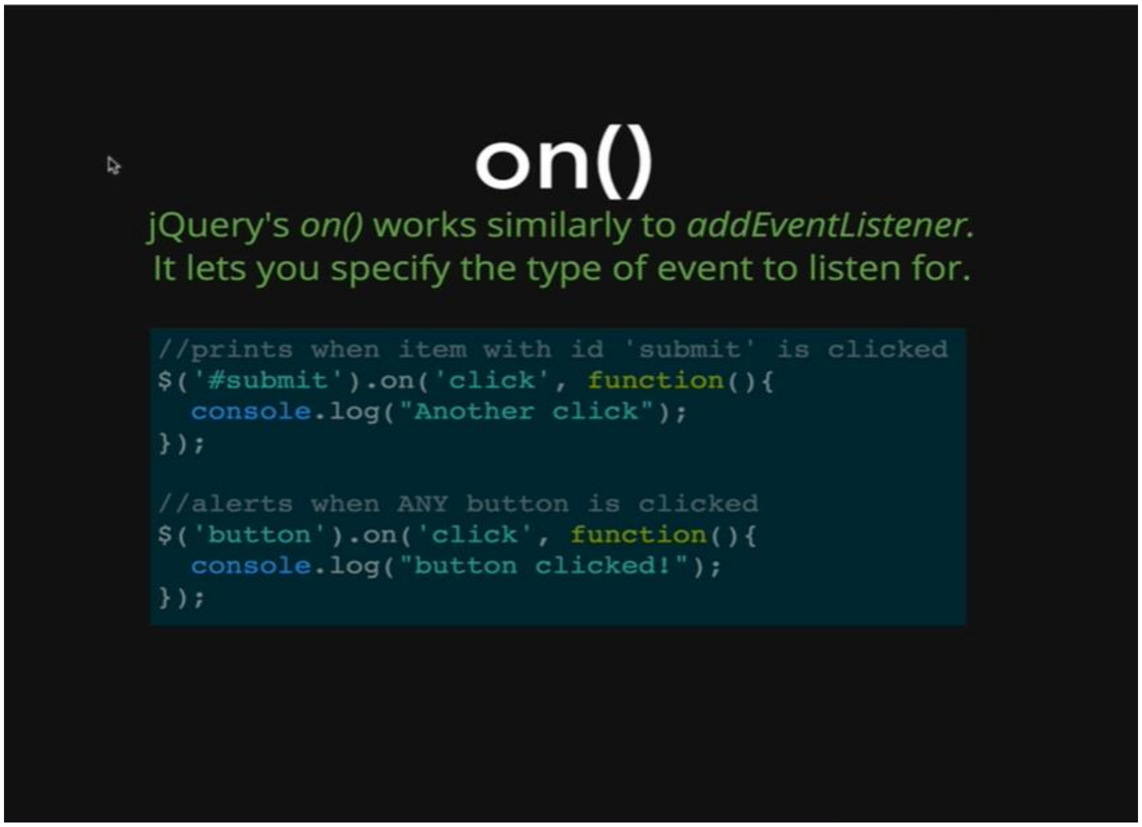
We have one last method that we are going to talk about. It the jQuery *On()* method that we will be using 99% of the time. Its by far the most used jQuery event method.

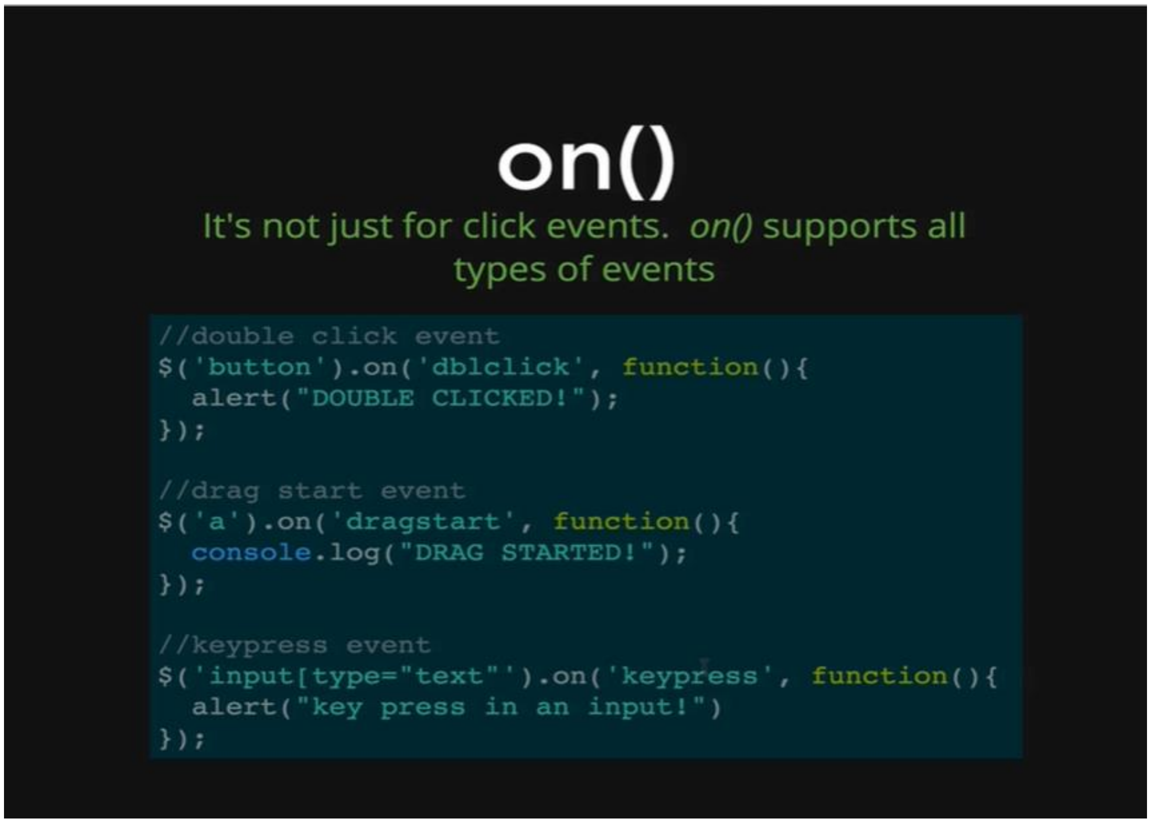


on()

jQuery's *on()* works similarly to *addEventListener*.
It lets you specify the type of event to listen for.

```
//prints when item with id 'submit' is clicked
$('#submit').on('click', function(){
  console.log("Another click");
});

//alerts when ANY button is clicked
$('button').on('click', function(){
  console.log("button clicked!");
});
```

The *On()* method is very similar to the Vanilla JS *addEventListener*. It behaves in a similar way where we give it the name of the event that we want to listen for. So, unlike *.click()* or .*keypress()* which only will fire on one type of an event, *On()* allow us to specify if we are doing a click, like at the example on the image, while clicking the class *submit,* the function gets executed.



## on()

It's not just for click events. *on()* supports all types of events

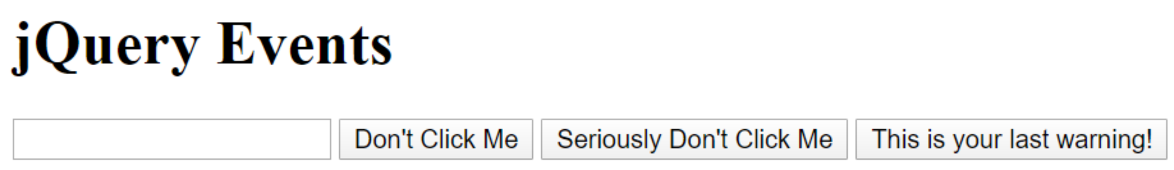
```
//double click event
$('button').on('dblclick', function(){
  alert("DOUBLE CLICKED!");
});

//drag start event
$('a').on('dragstart', function(){
  console.log("DRAG STARTED!");
});

//keypress event
$('input[type="text"']).on('keypress', function(){
  alert("key press in an input!")
});
```

The method also works when we are doing a double click, as we can see at the example on the image, on double-clicking the button, it executes the function. It can also work on drag start or a keypress, so it can take role of all the methods that we have seen before like the keypress. All the methods can be used by the *On()* method. It works pretty much the same way where we need to select something with the $ sign first.

Let’s demonstrate this on our HTML page.

```
<!DOCTYPE html>

<html>

  <head>

    <title>jQuery Events</title>

    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.4/jque
ry.min.js"></script>

  </head>

  <body>

    <h1>jQuery Events</h1>


      <input type= “text”>


    <button>Don't Click Me</button>

    <button>Seriously Don't Click Me</button>

    <button>This is your last warning!</button>

  </body>

</html>
```

*Output:*



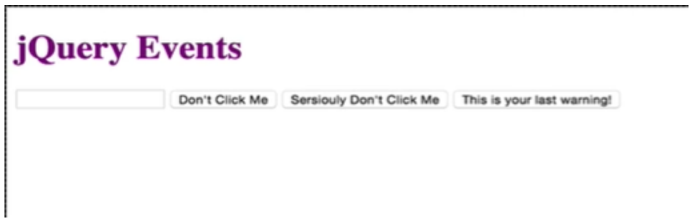
*Console.log*

*$("h1").on("click", function() {*

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

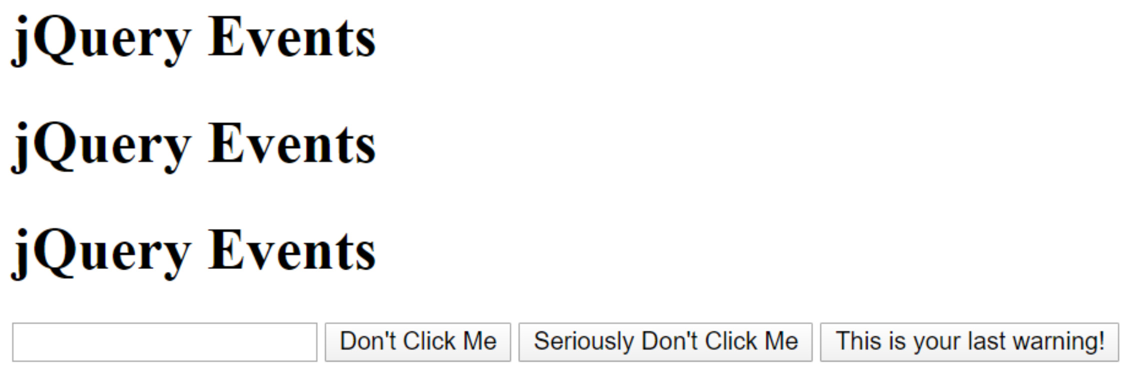
*});*

- ➔ *[          <h1>jQuery   Events</h1>]*


jQuery Events

As we can see, clicking on the h1 text (jQuery Events), it turns purple.

```
<!DOCTYPE html>

<html>

   <head>

      <title>jQuery Events</title>

      <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.4/jque
ry.min.js"></script>

   </head>

   <body>

      <h1>jQuery Events</h1>

     <h1>jQuery Events</h1>

     <h1>jQuery Events</h1>


         <input type= “text”>


      <button>Don't Click Me</button>

      <button>Seriously Don't Click Me</button>

      <button>This is your last warning!</button>

   </body>

</html>
```



jQuery Events

jQuery Events

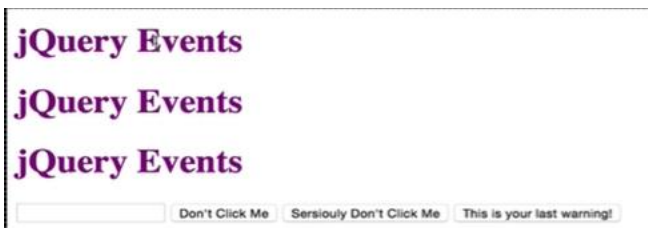
jQuery Events

As we can see we now have three h1 and on applying the same console code, if we click anyone of the h1 texts they all turn purple together.

*Console coding:*

```
$(“h1”).on(“click”, function() {
  $(“h1”).css(“color”, “purple”);
});
```

➔ *[ <h1>jQuery Events</h1>, <h1>jQuery Events</h1>, <h1>jQuery Events</h1>]*


jQuery Events

jQuery Events

jQuery Events

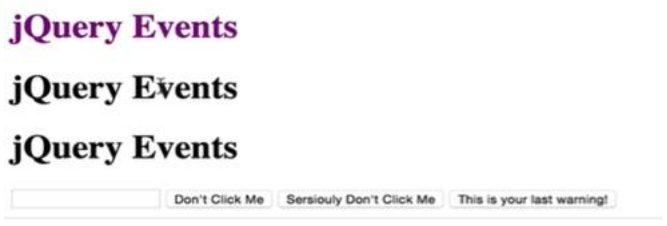
This is might be the feature that we want but most of the time we want the item to be purple that we click on. To do that we need to use the keyword *this* and remember it needs to be the jQuery wrapper for *this*. This *this* refers to the one h1 that was clicked on.

*Console coding:*

```
$(“h1”).on(“click”, function() {
 $(this).css(“color”, “purple”);
});
```

➔ *[ <h1>jQuery Events</h1>, <h1>jQuery Events</h1>, <h1>jQuery Events</h1>]*

Now clicking only, the one h1 will turn that h1 purple.


jQuery Events

jQuery Events

jQuery Events

Now we will test it *On()* event method on a keypress.

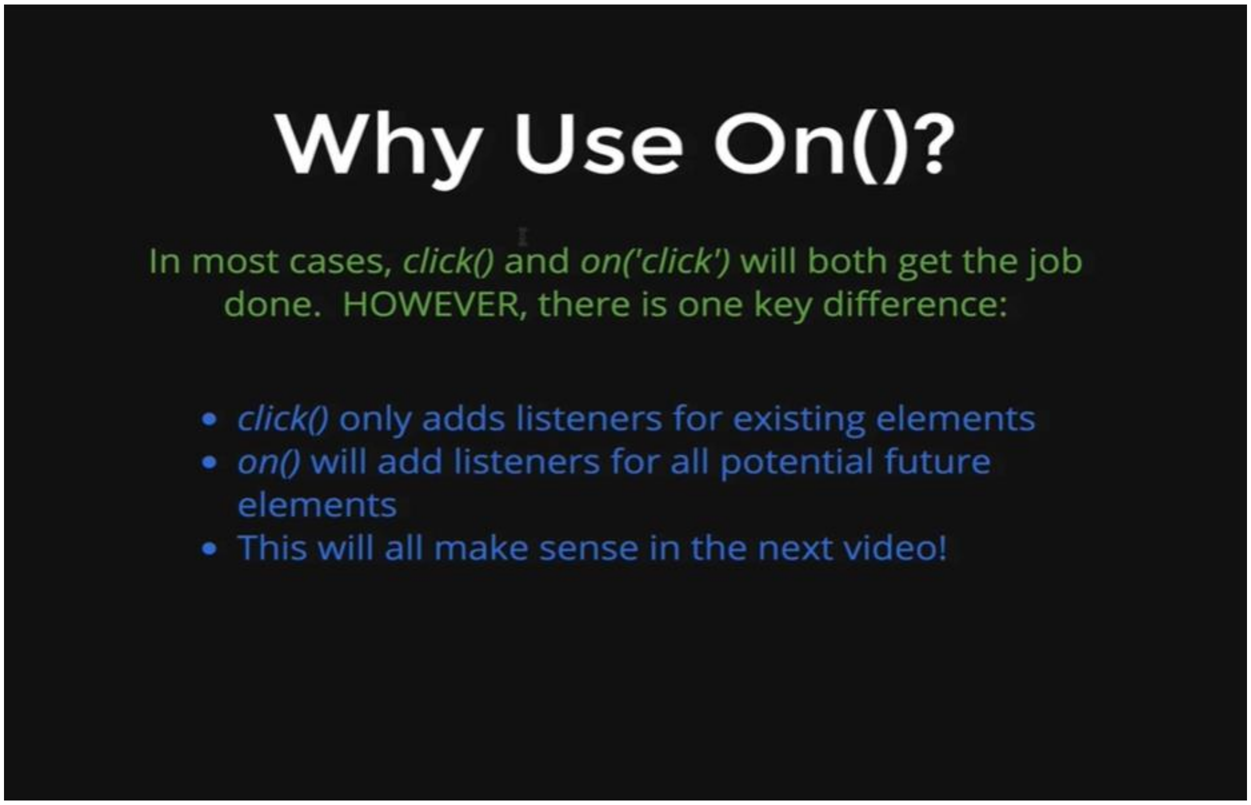
*Console coding:*

```
$(“input”).on(“keypress”, function() {
console.log(“Keypressed!”);
});
```

➔ *[ <input type= “text”> ]*

Now if we type any key on our input field we get the following output on our console.

```
Keypressed!
```



## Why Use On()?

In most cases, *click()* and *on('click')* will both get the job done. HOWEVER, there is one key difference:

- *click()* only adds listeners for existing elements
- *on()* will add listeners for all potential future elements
- This will all make sense in the next video!