Events y Callbacks

Las funciones son objetos

Callbacks

```
function myFunc(fn) { … }

…

myFunc(function () {

alert(‘ahora me ejecutaron’);

})
```

Veámoslo con un ejemplo

```
function suma(num1, num2) {
  return num1 + num2;
}

function resta(num1, num2) {
  return num1 - num2;
}

function division(num1, num2) {
  return num1 / num2;
}

function multiplicacion(num1, num2) {
  return num1 * num2;
}

```

```
function calculadora(cuenta, num1, num2) {
  return cuenta(num1, num2);
}
```

```
21  calculadora(suma, 1, 3); // 4
22  calculadora(resta, 1, 3); // -2
23  calculadora(division, 10, 2); // 5
24  calculadora(multiplicacion, 5, 5); // 25
25
```

Callbacks asincrónicos

ejemplo_callbacks2.js

```
/**
 * Ejecutar luego de
 * una tarea asincrónica
 */

var postId = '1234567890';

obtenerLikes(postId, function (likes) {
  alert("Se encontraron" + likes.length + " likes");
});

function obtenerLikes(postId, fn) {
  setTimeout(function () {
    fn([ '5501a7a7d93d972100f61145', '5372e27f7ff0ce200085d7e4', '53c400472e600532008089a3' ])
  }, 5000);
}

```

Eventos

```
1  /**
2   * Ejemplo de click en un botón
3   */
4
5  // <button type="submit" id="myButton">Alert!</button>
6
7  var button = document.getElementById('myButton');
8
9  button.addEventListener('click', function (event) {
10   alert('Me hicieron click!');
11 })
12
```

```
/**
 * Ejemplo event con jQuery
 */

// <button type="submit" id="myButton">Alert!</button>

$('button').click(function () {
  alert('me hicieron click');
})

```

```
11  $('button').on('click', function () {
12    alert('me hicieron click');
13  })
```

```
17  $('.product button.like').click(function (ev) {
18    $(this)
19      .closest('.product')
20      .addClass('liked')
21  })
22
```

```
23  // Múltiples eventos, mismo handler
24  $( "input" ).on(
25      "click change",
26      function() {
27          console.log( "Me hicieron click o me cambiaron el texto!" );
28      }
29  );
30
```

```
31  // Múltiples eventos con diferentes handlers
32  $( "p" ).on({
33    "click": function() { console.log( "me hicieron click!" ); },
34    "mouseover": function() { console.log( "me pasaron el mouse por arriba!" ); }
35  });
```

Métodos de Eventos

Browser Events

- `.resize()`
- `.scroll()`

Document Loading

- `.ready()`

Event Handler Attachment

- `.off()`
- `jQuery.proxy()`
- `.unbind()`
- `.on()`
- `.trigger()`
- `.undelegate()`
- `.one()`
- `.triggerHandler()`

Event Object

- `event.currentTarget`
- `event.stopPropagation()`
- `event.type`
- `event.preventDefault()`
- `event.target`

Form Events

- `.blur()`
- `.select()`
- `.change()`
- `.submit()`
- `.focus()`

Keyboard Events

- `.focusin()`
- `.keypress()`
- `.focusout()`
- `.keyup()`
- `.keydown()`

Mouse Events

- `.click()`
- `.focusout()`
- `.mouseenter()`
- `.mouseout()`
- `.dblclick()`
- `.hover()`
- `.mouseleave()`
- `.mouseover()`
- `.focusin()`
- `.mousedown()`
- `.mousemove()`
- `.mouseup()`

Sacando eventos

```
37  // Sacar los handlers bindeados
38
39  // Saca todos los handlers bindeados
40  // al evento click de los elementos <p> de la página
41  $( "p" ).off( "click" );
```

```
46  var foo = function() { console.log( "foo" ); };
47  var bar = function() { console.log( "bar" ); };
48
49  $( "p" ).on( "click", foo ).on( "click", bar );
50  $( "p" ).off( "click", bar ); // foo todavía se sigue ejecutando
```

Namespaces

```
52  // Usando Namespaces
53
54  $( "p" ).on( "click.myNamespace", function() { /* ... */ } );
55  $( "p" ).off( "click.myNamespace" );
56  $( "p" ).off( ".myNamespace" ); // Saca todos los eventos del namespace
57
```

Forms

```
60  $("form").on("submit", function(event) {
61
62    // Prevenir que el form haga post
63    event.preventDefault();
64
65    // Hacemos lo que queremos, como loguear el evento
66    console.log(event);
67
68    var action $(this).attr('action');
69
70    // Hacer el request
71    $.ajax(action, { /* ... */ })
72  });
```

Propagación de eventos

```
<html>
<body>
  <div id="container">
    <ul id="list">
      <li><a href="/products/1">Producto #1</a></li>
      <li><a href="/products/2">Producto #2</a></li>
      <li><a href="/products/3">Producto #3</a></li>
      <li><a href="/products/4">Producto #4</a></li>
    </ul>
  </div>
</body>
</html>
```

```
79  $("#list a").on("click", function(event) {
80    event.preventDefault();
81    console.log($( this ).text());
82  });
```

```
85  $("#list").append("<li><a href='http://shop-here.com/products/5'>Producto #5</a></li>");
```

Propagación o “burbujeo” de eventos

```
<html>
<body>
  <div id="container">
    <ul id="list">
      <li><a href="/products/1">Producto #1</a></li>
      <li><a href="/products/2">Producto #2</a></li>
      <li><a href="/products/3">Producto #3</a></li>
      <li><a href="/products/4">Producto #4</a></li>
    </ul>
  </div>
</body>
</html>
```

- <a>
-
- <ul #list>
- <div #container>
- <body>
- <html>
- document root

```
88  // Attachear un evento delegado
89  $("#list").on( "click", "a", function(event) {
90    event.preventDefault();
91    console.log($( this ).text());
92  });
```

```
// Attach a delegated event handler
$( "#list" ).on( "click", "a", function( event ) {
    var elem = $( this );
    if ( elem.is( "[href^='http']" ) ) {
        elem.attr( "target", "_blank" );
    }
});
```

```
103  // Refinamos el selector
104  $("#list").on("click", "a[href^='http']", function(event) {
105    $( this ).attr("target", "_blank");
106  });
```

Trigger

```
108 // Disparando eventos manualmente
109
110 $('#myButton').trigger('click')
111 $('#myButton').click()
```