Handling Events

Dan Wahlin
www.pluralsight.com

Agenda

- jQuery Event Model Benefits
- Handling Events
- Binding to Events
- live() and delegate()
- Handling Hover Events

Handling Events using JavaScript

Question:

What type of JavaScript code do you write to handle a button click event?

Answer:

It depends on the browser!

Event Attachment Techniques

Most Browsers:

```
myButton.addEventListener('click', function() { },false);
```

Internet Explorer:

```
myButton.attachEvent('onclick', function() { });
```

jQuery Event Model Benefits

- Events notify a program that a user performed some type of action
- jQuery provides a cross-browser event model that works in IE, Chrome, Opera, FireFox, Safari and more
- jQuery event model is simple to use and provides a compact syntax

Agenda

- jQuery Event Model Benefits
- Handling Events
- Binding to Events
- live() and delegate()
- Handling Hover Events

jQuery Event Shortcut Functions

- **jQuery event shortcuts:**
  - click()
  - blur()
  - focus()
  - dblclick()
  - mousedown()
  - mouseup()
  - mouseover()
  - keydown(),
  - keypress()
  - See more at http://api.jquery.com/category/events

Handling Click Events

- .click(handler(eventObject)) is used to listen for a click event or trigger a click event on an element

```
$('#myID').click(function() {
   alert('The element myID was clicked');
});
```

Handling Click Events

- Raising a click event from within another function:

```
$('#otherID').click(function() {
   $('#myID').click();
});
```

- This would fire when the element otherID was clicked and raise the click event for myID

Agenda

- jQuery Event Model Benefits
- Handling Events
- Binding to Events
- live() and delegate()
- Handling Hover Events

Using bind()

- .bind(eventType, handler(eventObject)) attaches a handler to an event for the selected element(s)

```
$('#MyDiv').bind('click', function() {
    //Handle click event
});
```

.click() is the same as .bind('click')

Using unbind()

- .unbind(event) is used to remove a handler previously bound to an element:

  $('#test').click(handler); can be unbound using

  $("#test").unbind();

- Specific events can also be targeted using unbind():

  $('#test').unbind('click');

Binding Multiple Events

- bind() allows multiple events to be bound to one or more elements
- Event names to bind are separated with a space:

```
$('#MyDiv').bind('mouseenter mouseleave',
  function() {
     $(this).toggleClass('entered');
  }
);
```

Agenda

- jQuery Event Model Benefits
- Handling Events
- Binding to Events
- live() and delegate()
- Handling Hover Events

live() and delegate() Functions

- live() and delegate() allow new elements added into the DOM to automatically be "attached" to an event handler

live() –Allows binding of event handlers to elements that match a selector, including future elements. Events bubble up to the document object.

delegate() – Replacement for live() in jQuery 1.4. Attaches an event handler directly to the selector context.

Using live()

- Event handlers can be set using live()
- The document object handles events by default
- Works even when new objects are added into the DOM:

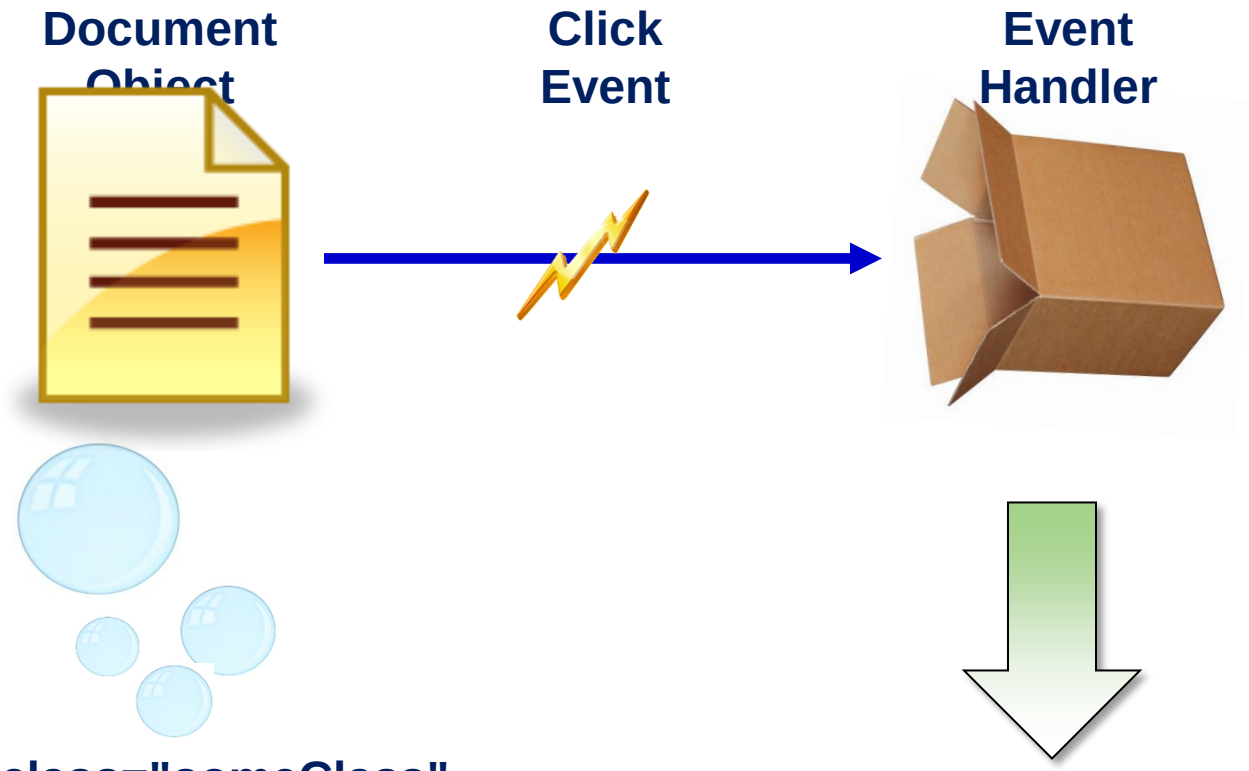
```
$('.someClass').live('click',
    someFunction);
```

Any element added with .someClass will have a click event

- Stop live event handling using die():

```
$('.someClass').die('click', someFunction);
```

How live() Works



Using delegate()

- Newer version of live() added in jQuery 1.4
- A context object (#Divs in the sample below) handles events by default rather than the document object
- Works even when new objects are added into the DOM:

```
$('#Divs').delegate('div','click',someFunction);
```

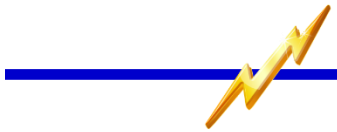
- Stop delegate event handling using undelegate()

How delegate() Works

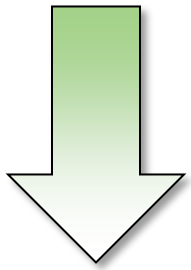
Context Object:
#Divs

Click
Event

Event
Handler



Click
Event



```
<span class="someClass" />
<p class="someClass" />
<div class="someClass" />
```

```
$('#Divs').delegate('.someClass','click',func
() { });
```

Agenda

- jQuery Event Model Benefits
- Handling Events
- Binding to Events
- live() and delegate()
- Handling Hover Events

Handling Hover Events

- Hover events can be handled using hover():

```
$(selector).hover(handlerIn, handlerOut)
```

- handlerIn is equivalent to mouseenter and handlerOut is equivalent to mouseleave

Using hover()

- This example highlights #target on mouseenter and sets it back to white on mouseleave

```
$('#target').hover(
    function(){
        $(this).css('background-color', '#00FF99');
    },
    function(){
        $(this).css('background-color', '#FFFFFF');
    }
);
```

Alternate Hover Example

- Another option is $(selector).hover(handlerInOut)
- Fires the same handler for mouseenter and mouseleave events
- Used with jQuery's toggle methods:

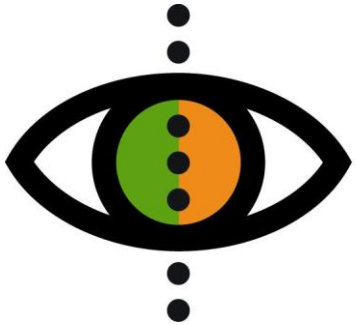
```
$('p').hover(function() {
    $(this).toggleClass('over');
});
```

This code will toggle the class applied to a paragraph element

Summary

- jQuery simplifies handling cross-browser event attachments
- Many built-in shortcut functions such as click() can be used
- bind() and unbind() provide a flexible way to work with different events
- live() and delegate() both allow future child elements to be wired to event handlers
- The hover() function provides a simple way to handle mouseenter and mouseleave events
- Combine hover() with toggleClass() to easily swap out CSS classes on an element

For more in-depth online developer training visit



pluralsight

see what you can learn

on-demand content from authors you trust