



FULL SAIL
UNIVERSITY

scripting for web applications

Finished Creative Brief is due at 9:00am
Turn in to dropbox as **lastname_firstname_CB.pdf**

jQuery events and
animation

SFW-2 DUE Dates

Scripting for Web Applications

Item	Due Dates
Branding / Logo	07/07/13 After Lab on the First Day
Creative Brief - Finished Document	07/16/13 Before Lecture 4
Site Prototype (<i>html/css</i>)	07/20/13 - After Last Lab of the 2nd Week
Development Milestone (<i>javascript</i>)	07/27/13 - Due End of Lab 7
Inclusion of 5 media center items	08/03/13 - Last Day of Class After Lab
Aesthetics & Usability (<i>finished site</i>)	08/03/13 - Last Day of Class After Lab
Functionality (<i>finished site</i>)	08/03/13 - Last Day of Class After Lab
Professionalism	The duration of the course
Class Participation	The duration of the course

```
$( "#nav > li" )...
```

```
$( "#nav a[data-id='001']" )...
```

```
$( "#nav li:first" )...
```

```
$( "#nav li:not(.active)" )...
```

use IDs for parent-level items (singular)

uses classes for repeatable elements (like lists/collections)

```
<ul id="nav">  
  <li></li> <li></li> <li></li> <li></li>  
</ul>
```

```
$("#nav li").css().filter(":odd").css().parent().css();
```

```
[  
<li>,<li>,<li>,<li>  
]
```

```
[  
<li>,<li>,  
]
```

```
[  
<ul id="nav"/>  
]
```

```
var html = '<a href="">Link</a>' ;  
  
$(html)  
  .appendTo('#nav')  
  .animate()  
;  
  
$( '#nav' )  
  .append(html)  
  .animate()  
;
```

append
appendTo
prepend
prependTo

after
insertAfter
before
insertBefore

replaceWith
replaceAll
clone

wrap
wrapAll
wrapInner

remove
empty

parent

wrap

sibling

before

target

after

child

prepend

append

web design and development
bachelor of science degree program





- Check Out How To Work With Events
 - Load up this site to try out using jQuery Events and see them applied:

<http://jsfiddle.net/sfw2/RTbYY/>

❖ DOM Event Model

- ❖ Last month you explored the **Event Model** of the DOM.
- ❖ There are several key parts to how the browser interprets and handles events, and you used JavaScript to assign and control those event actions.
- ❖ Let's re-examine the Event Model:

❖ DOM Event Model

- ❖ When a DOM element triggers an event, an “**event**” is always created
- ❖ We can create functions called “**handlers**” that are called to do something

❖ Anatomy of an Event

- ❖ Events have 2 components:
 - ❖ The DOM element we listen on
 - ❖ The function we assign to that listener

❖ jQuery Events

- ❖ So what does jQuery provide us?
 - ❖ Chainable methods for binding event handlers,
 - ❖ Allows multiple handlers to be bound to each event type,
 - ❖ Delegated event model,
 - ❖ Provides a cross-browser-compatible **event object**,
 - ❖ Provides cross-browser methods for canceling bubbling and browser-defaults

❖ jQuery Events (OLD School)

- ❖ There are a few different methods for event bindings in jQuery. The most basic is a method *type* where the *event name* is the name of the method itself.
- ❖ The argument is what function to use as the handler (*can be a reference, or a literal*)

Event Method	Example
<code>click(fn)</code>	<code>\$("a").click(<i>function(){};</i>);</code>
<code>mousemove(fn)</code>	<code>\$("a").mousemove(<i>function(){};</i>);</code>
<code>mouseup(fn)</code>	<code>\$("a").mouseup(<i>function(){};</i>);</code>
<code>mousedown(fn)</code>	<code>\$("a").mousedown(<i>function(){};</i>);</code>
<code>keyup(fn)</code>	<code>\$("input").keyup(<i>function(){};</i>);</code>
etc...	

.on()

- ▶ This is the preferable, source method for event binding
- ▶ Pre version 1.7 the command was .bind()

```
$(target).on( type, data, function )
```

type (string) eg - “click”

data (object) optional custom event data

fn (function) event handler

```
$("#link").on("click", {myvar:"test"}, function(e){  
    alert(e.data.myvar);  
    return false;  
});
```

event information

- ▶ Any function bound to an event will only receive 1 argument, **the event object**
- ▶ The **event object** contains information about what happened in the event

```
$("#mylink").on("click", function(e){  
    alert(e.type);  
    return false;  
});
```

web design and development
bachelor of science degree program



e.Property	Description
type	string: The name of the event type (ie- “click” or “mouseleave”)
target	object: DOM reference to the element that triggered the event. (<i>if a child element is the source, it will be the trigger</i>)
currentTarget	object: DOM reference to the current element in the bubbling chain. Note: <i>currentTarget is equal to the value of this</i>
relatedTarget	object: DOM reference for mouse event issues
timeStamp	number: Date timestamp of when the event was triggered (<i>in ms</i>)
which	number: Normalized key code to use instead of keyCode or charCode
pageX / pageY	number: The x/y event position, relative to the document page.
screenX / screenY	number: The x/y event position, relative to the client’s screen.
data	object: The custom event object, if used.
namespace	string: The custom namespace, if used.

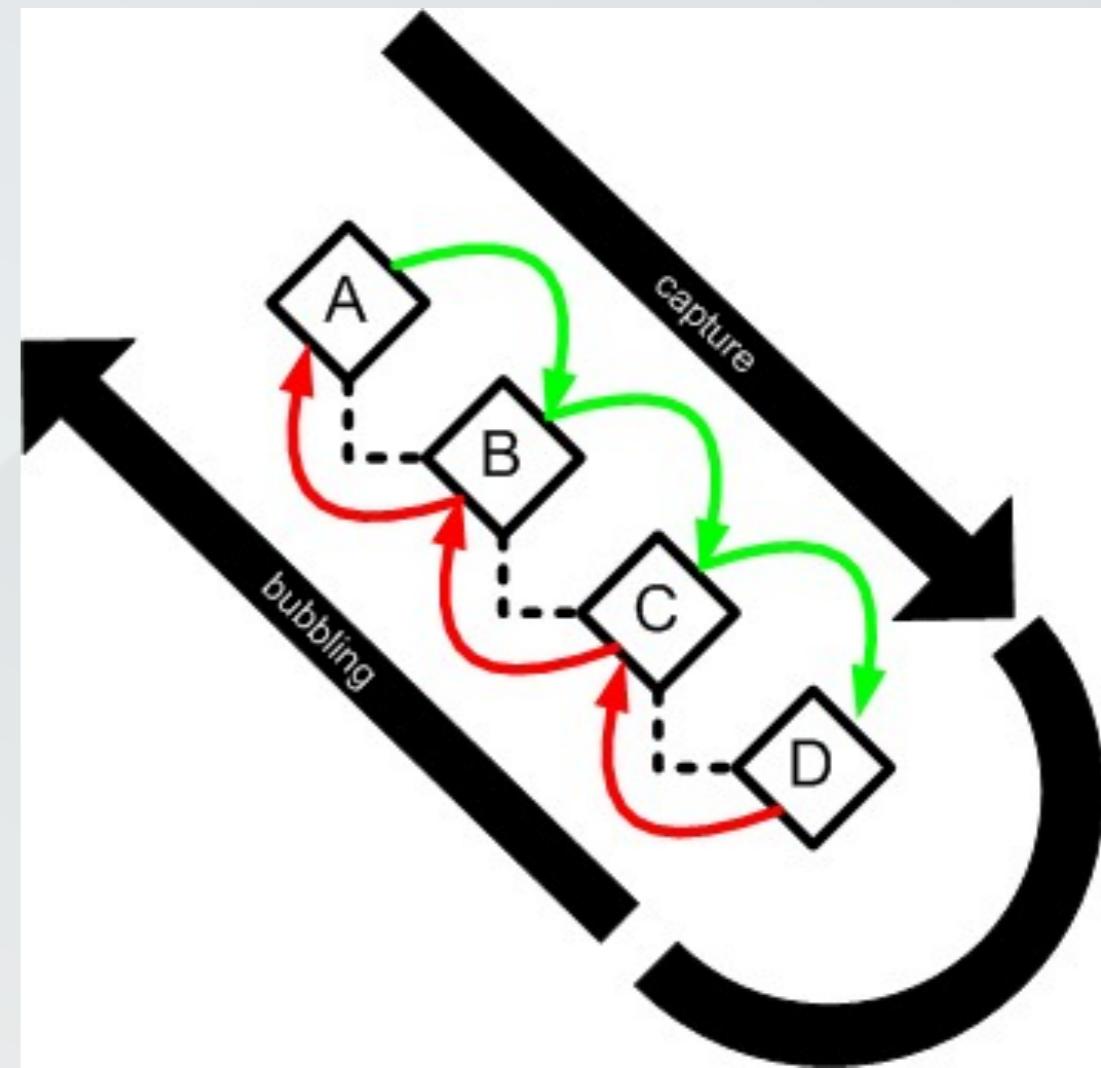
web design and development
bachelor of science degree program



Capture/Bubble DOM Model

1. Event is triggered (*click, keyup, mousemove, etc.*)
2. Window creates an event object about the event.
3. Browser captures matching event listeners
down to the firing element.
4. In reverse order, Browser fires each found event handler. Each handler fully executes before the next does.
5. Browser activates a default action unless instructed not to.

There are 2 exceptions to this model: focus & blur



Event Object Methods

e.stopPropagation();	Calling this method from inside an event handler will prevent the <i>Bubbling Phase</i> from triggering other events.
e.preventDefault();	Calling this method from inside an event handler will prevent the browser's default action (such as following an href or the <form> action)
return false;	By simply using the statement "return false;" jQuery will try to use <i>both</i> stopPropagation and preventDefault.

- Unless you specifically want *bubbling* to occur, using **return false** is the safest bet.

Event Context

- ▶ In week 1, we learned that all functions have a **context**, which is the object that the function was assigned to.
- ▶ In events, the context of our function is the **element that fires the event**.
- ▶ Context is an object, called **this**

```
$("#box").on("click", function() {  
  console.log( this );  
  return false;  
});
```

Saving the Context

- ▶ A common trick is to create a variable called **that**, with the **this** jQuery object
- ▶ Reduces factory calls and creates a localized store

```
$("a:first").on('click', function() {  
    var that = $(this);  
    that.css({background: 'red'});  
    return false;  
});
```

web design and development
bachelor of science degree program



❖ Let's look at a few other notable event types:

Event Method	Description
<code>mouseover(fn)</code>	Triggers when the cursor enters the element's area or enters the area of a child element (<i>this event may trigger multiple times</i>).
<code>mouseout(fn)</code>	Triggers when the cursor leaves the element's area or the cursor leaves a child element (<i>this event may trigger multiple times</i>).
<code>mouseenter(fn)</code>	Triggers only once when the cursor enters the element's area, not including any children elements. Only exist in jQuery. Replaces <code>mouseover(fn)</code>
<code>mouseleave(fn)</code>	Triggers once only when the cursor leaves the element's area. Only exist in jQuery. Replaces <code>mouseout(fn)</code>

- ❖ Let's look at a few other notable event types:

Event Method	Description
<code>focusin(fn)</code>	This is a fixed version of <code>focus</code> , to include bubbling and child detection. This method is a shortcut for <code>.on('focusin', handler)</code> .
<code>focusout(fn)</code>	The <i>blur</i> version of <code>focusin</code> .
<code>load(fn)</code>	Can be used on any element to detect when that element has been rendered to the page (<i>useful for images, scripts, iframes</i>)

.off()

<code>\$(target).off()</code>	No arguments, this will remove <i>all</i> events from target
<code>\$(target).off(type)</code>	Removes the specified event <i>type</i> from target
<code>\$(target).off(type, handler)</code>	If a named function was used, you can unbind just that handler by passing its name

```
var hn = function(e){  
    return false;  
};  
$("a").on('click', hn);  
$("a").on('click', function(){});
```

```
$("a").off();  
  
$("a").off("click");  
  
$("a").off("click", hn);
```

Binding Multiple Events

```
$(target).on( type, data, function )
```

- ▶ You can bind multiple events to the *same function* by using **space(s)** in the **type string**

```
$("#link").on("mouseenter mouseleave", function(e) {  
    return false;  
});
```

web design and development
bachelor of science degree program

23



Binding Multiple Events

`$(target).on(object)`

Object with events as keys, paired with function handlers

- ▶ Using an object, you can bind multiple individual events at the same time.

```
$("#box").on({  
    click: function(e){},  
    mouseenter: function(e){},  
    mouseleave: function(e{})  
});
```

Custom Event Namespaces

- ▶ Add a class name to turn on and off a bind, by name

```
$(target).on( type.namespace, data, function )
```

```
$("#box").on("click.topmenu", function(e){  
    return false;  
});
```

```
$("#box").off("click.topmenu");
```

.one()

- ▶ Exact same as .on, except this handler will self-destruct after 1 use

```
$(target).one( type, data, function )
```

Binds an event handler *function* to *event* as normal, except the handler is automatically unbound after the event is triggered once.

```
$("#link").one("click", function(e){  
    return false;  
});
```

web design and development
bachelor of science degree program

26



.toggle()

- ▶ A specialized “click” listener, alternates between multiple functions automatically

```
$(target).toggle( oddFn, evenFn )
```

oddFn: function fires for odd *n*th clicks (1st, 3rd, etc)
evenFn: function fires for even *n*th clicks (2nd, 4th, etc)

```
$("#link").toggle(  
    function(e){                  // odd function handler  
    },  
    function(e){                  // even function handler  
    }  
) ;
```



15 mins.

/Lecture Activities/day4/activity1

lecture activity: tooltip

/Lecture Activities/day4/activity1

Target all the “.tooltip” anchors (these will trigger our tooltip)

Bind mouseenter, mouseleave, and mousemove events on them

On mouseenter:

- target the “.tipbox” sibling of the `$(this)` target using `.next()`
- set these css properties on tipbox:
 - “top” to mouse’s `e.pageX+3` and “left” to `e.pageY+3`
 - “display” to “block”

On mousemove:

- update the “top” and “left” just like in mouseenter

On mouseleave, hide the tipbox

jquery delegated events

Event Delegation

`$(window).on(target, type, function)`

Binds the event listener to the global `window` object, and delegates to the `target`

```
$ (window) .on ('#nav a' , 'click' , function(e){});
```

web design and development
bachelor of science degree program

31



- ▶ Additionally, the delegated **on** events cannot be removed normally, will need use **.off**

```
$(window).off( target, type )
```

Unbinds all instances of the specified delegated “.on” event type for the *target selector*.

```
$(window).off('#nav a', 'click');
```

jquery animation

<http://jsfiddle.net/sfw2/zV3qX/>



❖ jQuery Effects

- ❖ Animation effects are one of the biggest draws of designers and developers to most language libraries.
- ❖ Remember, **usability** should always be a concern. Effects should be kept at a minimum, meant for enhancing the user experience, not to clutter it.
- ❖ *Most animations should be under a second. **jQuery's default is 400 milliseconds.***



❖ jQuery Effects

- ❖ **Effects** are jQuery methods that change **css** over time.
- ❖ So, the most important thing to note about any of these methods, is what css changes does it do?



❖ jQuery hide/show

- ❖ Often, one of the most desired effects to simply hide or reveal an element. For this:

<code>target.hide()</code>	With no arguments, immediately hides the element by setting <code>display:none</code> . If the element was already hidden, it stays hidden.
<code>target.show()</code>	No arguments, immediately reveals element by reverting the <code>display</code> property to its original state.
<code>target.hide(speed, function)</code> <code>target.show(speed, function)</code>	Same as hide and show but with twists: <code>speed</code> can be provided in milliseconds, and the <code>function</code> will work as a callback after the animation finishes. More specifically, it will animate the <code>width</code> , <code>height</code> , <code>margin</code> , <code>padding</code> , and <code>opacity</code> of the element simultaneously and evenly.



❖ jQuery toggle

- ❖ Also provided is a method to toggle between the **show** and **hide** effects:

<code>target.toggle()</code>	Determines the current <i>show/hide</i> state, and calls the opposite.
<code>target.toggle(boolean)</code>	If <i>boolean</i> is <i>true</i> , this will call <i>hide</i> , while <i>false</i> will call <i>show</i> .
<code>target.toggle(speed, function)</code>	Same as <i>toggle</i> with no <i>boolean</i> , automatically switches between <i>show</i> and <i>hide</i> , using <i>speed</i> and allowing <i>function</i> as a callback of the effect.



❖ jQuery Effect Callbacks

- ❖ Every animation effect has 2 ***optional*** arguments, **speed** and **callback**.
 - ❖ hide/show/toggle are unique in that with no speed argument, they instantly change.
 - ❖ For the next few effects, a default speed of 400 milliseconds is applied unless otherwise specified.
 - ❖ With the exception of **toggle**, each of these effect methods can also include a **callback function** as a single argument, or as the second argument. The function you specify will be called only after the successful completion of the animation.



❖ jQuery fadeIn / fadeOut / fadeTo

target.fadeIn(speed, function)

Animates the *opacity* style of the element to full. If the element was already visible, no effect occurs. The element is considered “*:visible*” after this.

target.fadeOut(speed, function)

Animates the opacity style of the element to 0. If the element was already hidden, no effect occurs. The element is considered “*:hidden*” after this.

target.fadeTo(speed, opacity, function)

Same as fadeIn and fadeOut, but accepts a new argument *opacity*.

opacity: Number: From 0 to 1, animates the opacity of the element for *opacity*100%*

```
$ ("#mydiv") .fadeTo( 600 , 0.5 );
```



❖ jQuery slideUp / slideDown / slideToggle

`target.slideDown(speed, function)`

Reveals the element by animating the *height, vertical margins, and vertical paddings* of the element to their full display.

The element is considered “`:visible`” after this effect ends.

`target.slideUp(speed, function)`

Hides the element by animating the *height, vertical margins, and vertical paddings* of the element `0px`.

The element is considered “`:hidden`” after this effect ends.

`target.slideToggle(speed, function)`

Automatically determines the element’s current visibility and fires either `slideUp` or `slideDown` accordingly.



❖ jQuery Effect Queues

- ❖ jQuery will automatically create **effect queues**
- ❖ Animations are added to the queue in the order they're called out.
- ❖ For example, chaining together 3 animation effects will cause them to queue, occurring one after the other.

```
$("li").click(function() {
    $(this).fadeOut().slideDown().slideUp();
});
```



❖ jQuery Effect Queues

- ❖ The below code block would have the same effect.
- ❖ A queue is created whenever an animation is already happening, and you attempt to call a new animation. The current running effect is not added to the queue, only the ones that are waiting.

```
$("li").click(function() {  
    $(this).fadeOut();  
    $(this).slideDown();  
    $(this).slideUp();  
});
```



❖ jQuery Effect Queues

- ❖ Also note that **ONLY** *animation effects* are entered into the queue. Any other methods are still executed immediately.

```
$("li").click(function() {
    $(this)
        .slideUp()
        .html("I'm changed!")
        .slideDown();
});
```

- ❖ The correct solution is to use a callback on the *slideUp* animation.



❖ Animation stop

- ❖ Once a queue of jQuery effects has been set on an object, we can call a **stop** method to cease the animations and even remove others from the queue.

`target.stop(bool, bool)`

First Boolean: If **true**, removes all animation queues from the item.
Second Boolean: If **true**, stops the current animation at its max value.
Empty: Stops only the current animation, next in queue fires.

```
$("#box").slideUp();  
$("#box").stop(true, true);
```



❖ Animation delay

`target.delay(ms)`

Targets the animation queue on the selector set and delays the next animation by a *millisecond* number.

```
$ ("#box")  
  .slideUp()  
  .delay(500)  
  .fadeOut()  
;
```



❖ jQuery animate

- ❖ Instead of providing a suite of effects, jQuery gives us the **animate** method.
- ❖ *All of the effects we've just seen use the animate method internally.*
- ❖ The animate method operates by letting us specify a set of CSS properties to tween.
animate will tween from the current values to the specified values, using *duration*.
- ❖ For example, if I wanted to recreate the fadeIn effect:

```
$ (“#box” ) .animate( {opacity:0} );
```



❖ **jQuery animate**

```
target.animate(properties, duration, easing, callback)
```

properties: required: *object*
duration: optional: *number*
easing: optional: *string*
callback: optional: *function*

- ❖ The **animate** method takes an ***object literal*** of css properties, and animates each simultaneously, over the *duration*, and using the *easing* provided.
- ❖ jQuery provides 2 built-in easing options: “linear” or “swing”
- ❖ As with all other effects, we can provide a *callback function* as the last argument.

```
$ (“#box”).animate({opacity:0}, 400, function(){});
```



- ❖ jQuery easing plugin

- ❖ For easing, jQuery only provides “swing” and “linear”.
- ❖ For more effects, the most popular easing plugin can be found at:

<http://gsgd.co.uk/sandbox/jquery/easing/>
google search “jquery easing”



❖ jQuery animate

- ❖ There are a couple other important notes to make about **animate**,
- ❖ CSS camelCase can be used here for the *keys*. For example, to animate the css property of “border-width” is:

```
$("#box").animate({borderWidth:10}, 400);
```

- ❖ Also notice that the value can be either a string, or a number. jQuery will automatically determine the correct value (*and uses pixels by default unless otherwise stated*)

```
$("#box").animate({borderWidth:"10px"}, 400);
```



❖ jQuery animate

- ❖ The next important note is that jQuery will also accept **em** or **%** as a value:

```
$ (“#box”).animate({width:”100%”}, 400);
```

- ❖ And finally, jQuery will also accept strings that utilize **+ =** or **- =**

```
$ (“#box”).animate({width:+=200}, 400);
```



❖ jQuery animate

- ❖ Also keep in mind that the css properties here is an *object literal*. We can specify as many as we want by using literal syntax, and **all** of these properties are animated **simultaneously**.

```
$("#box").animate(  
{  
    opacity: 1,  
    height: "+=100",  
    width: "+=300"  
,  
400,  
"swing",  
function(){ $(this).remove(); }  
);
```



❖ jQuery easing animation

```
$("#box").animate(  
{  
    opacity: 1,  
    width: "+=300",  
    height: 200  
},  
"easeInOutCubic",  
400,  
function(){ $(this).remove(); }  
);
```



❖ jQuery easing animation

```
$("#box").animate(  
{  
    opacity: [1, "swing"],  
    width: ["+=300", "easeOutBounce"],  
    height: 200  
},  
"easeInOutCubic",  
400,  
function(){ $(this).remove(); }  
);
```



lecture activity 2: animation

❖ Custom “Drop” Animation

- ❖ Let's explore creating a slightly advanced custom animation.
- ❖ Instead of having an element hide or slideUp, we'd like to have the element appear to “drop off” the page, indicating to the user that it is being removed, and then destroy the item from the DOM.
- ❖ To do this, we'll need 2 animation effects... opacity and position.
- ❖ Begin by creating a new html document and js file.



❖ jQuery “Drop” Animation

❖ GOAL: bounce the element up and drop off the page.

1. Target the .box items.
2. Setup click event.
3. Set a position .css relative position property.
4. Have the element go up by 25px before dropping off the page by.
5. Animate the box dropping off the page using some of the techniques we learned in lecture earlier today.



Lab 4 *(no turn-in today)*

lab begins 1hr after break

❖ **Next Milestone:** Project Prototype *(due end of Week 2)*

- ❖ **ALL** html/css markup completed, **no** javascript in turn-in
- ❖ filler content (**no** lorem ipsum) must be used inside html to test your design
- ❖ **ALL** components of your appl as html... example: (*landing.html, addproject.html*)
- ❖ *only 1 stylesheet file for the entire project*
- ❖ *each html page should look like it would when live*