FULL SAIL
UNIVERSITY

scripting for web applications

2

jQuery events and animation

due.**Dates**

Item	Due Dates
Branding / Logo	11/25/13 - After Lab on the First Day
Project Pitch	12/02/13 - Before Lecture 2
Creative Brief - Finished Document	12/4/13 - Before Lecture 3
Site Prototype (html/css)	12/09/13 - Before Lecture 5
Development Milestone (javascript)	12/13/13 - Due End of Lab 7
Inclusion of 5 media center items	12/20/13 - Last Day of Class After Lab
Aesthetics & Usability (finished site)	12/20/13 - Last Day of Class After Lab
Functionality (finished site)	12/20/13 - Last Day of Class After Lab
Professionalism	The duration of the course
Class Participation	The duration of the course

target.Review

```
$(“#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)

target.Review

```
<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"/>
]
```

manipulation.**Review**

```
var html = '<a href="">Link</a>';
```

```
$(html)
    .appendTo('#nav')
    .animate()
;
```

```
$('#nav')
    .append(html)
    .animate()
;
```

append
appendTo
prepend
prependTo

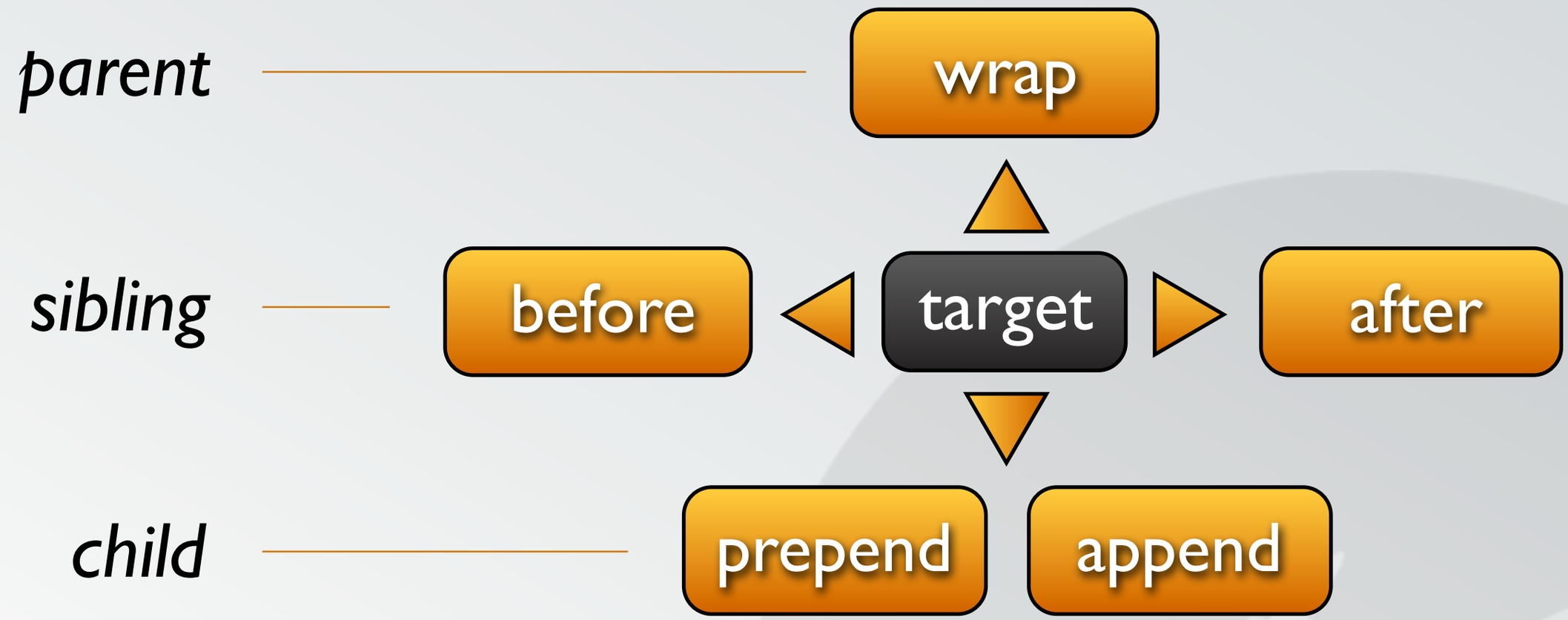
after
insertAfter
before
insertBefore

replaceWith
replaceAll
clone

wrap
wrapAll
wrapInner

remove
empty

manipulation.**Review**



jquery.**Events**

jQuery.Events

Working with Events

- Using jQuery Events and see them applied:

in FSO's Reference Tab:
Course Material Follow Along Files
'Goal4.zip'

jQuery.Events

DOM Event Model

- In PWA1 you explored the basic **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:

jQuery.Events

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

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**

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
click(*fn*)	$(“a”).click(*function(){}*);
mousemove(*fn*)	$(“a”).mousemove(*function(){}*);
mouseup(*fn*)	$(“a”).mouseup(*function(){}*);
mousedown(*fn*)	$(“a”).mousedown(*function(){}*);
keyup(*fn*)	$(“input”).keyup(*function(){}*);
etc...	

jQuery.Events

.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;
});
```

jQuery.Events

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;
});
```

jQuery.**Events**

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. *(if a child element is the source, it will be the trigger)*
currentTarget	*object:* DOM reference to the current element in the bubbling chain. Note: *currentTarget is equal to the value of* ***this***
relatedTarget	*object:* DOM reference for mouse event issues
timeStamp	*number:* Date timestamp of when the event was triggered *(in ms)*
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.

jQuery.Events

Event Context

- You 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;
});
```

jQuery.Events

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

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

jQuery.Events

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

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

jQuery.Events

.off( )

$(target).off()	No arguments, this will remove *all* events from *target*
$(target).off(*type*)	Removes the specified event *type* from *target*
$(target).off(*type, handler*)	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);
```

jQuery.Events

Binding Multiple Events w/ one Handler

$(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;
});
```

jQuery.Events

Binding Multiple Events w/ Multiple Handler

$(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){}
});
```

jQuery.Events

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”);
```

jQuery.Events

.one( )

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

*$(target).*one*(type, data, function)*	Binds an event hander *function* to *event* as normal, except the handler is automatically unbound after the event is triggered once.

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

jQuery.Events

.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
    }
);
```

lecture.**Activity** (tool tip)

in FSO's Reference Tab: Course Material - Lecture Activities 'Activity for Goal4-1'

- 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

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){});
```

event.**Delegation**

- 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****.*
- **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.**Effects**

jQuery hide/show

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

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

jquery.**Effects**

jQuery toggle

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

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

jquery.Effects

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.**Effects**

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.Effects

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.Effects

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.

jquery.Effects

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.Effects

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);
```

jquery.Effects

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 fadeOut effect:

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

jquery.**Effects**

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.Effects

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.Effects

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.Effects

jQuery easing animation

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

jquery.Effects

jQuery easing animation

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

lecture.**Activity** (drop box animation)

in FSO's Reference Tab: Course Material - Lecture Activities 'Activity for Goal4-2'

- 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.

lecture.**Activity** (drop box 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.

assignment.Goal4 (see schedule for due dates)

Next Milestone: Project Prototype

- **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*