

# JQuery

# What is it?

- A library whose purpose is to make it easier to manipulate the DOM
  - traversal
  - event handling
  - animation
  - Ajax

# Syntax

```
$(selector).action()
```

Construct a jQuery object with an element or set of elements

```
$() == jQuery()
```

```
$(document).ready()
```

```
==
```

```
jQuery(document).ready()
```

# CSS Style Selectors

// select the element with ID header'

```
$ ( '#header ' ) ;
```

```
$ ( ' li ' ) ; // select all list items
```

```
$ ( ' ul li ' ) ; // select list items that are in  
unordered lists
```

```
$ ( ' .person ' ) ; // select all elements with a class of  
'person'
```

# Many selectors!

// select all elements with href attribute'

```
$ ( ' [ href ] ' ) ;
```

// select elements with a target attribute with value  
\_blank

```
$ ( ' a [ target = ' _ blank ' ) ;
```

```
$ ( ' tr : even ' ) ; // select all even <tr> elements
```

```
$ ( ' this ' ) ; // select the current element
```

# actions (functions)

- **Getters** - read the selection
  - operate on the first element in the selection
- **Setters** - change the selection
  - operate on all elements in the selection (implicit iteration)
  - return the selected element so they can be chained.

# Create new HTML

```
$( '<p>Hello!</p>' );
```

```
$( '<p>', {  
  html: 'Hello!', 'class': 'greet'  
} ).appendTo( 'body' );
```

# Working with Lists

```
// get the first list item
// on the page
var $listItem = $( 'li' ).first();
```

```
// get the siblings of the list
item var $listSiblings =
listItem.siblings();
```

```
// get the next sibling of
// the list item
var $nextSibling = listItem.next();
```

```
// get the list item's parent
var $listParent =
listItem.parent();
```

```
// get immediate children
var list = $( 'li' );
$listItems = list.children();
```

```
// get ALL list items in the list,
// including nested ones
```

```
var $allListItems = list.find( 'li' );
```

```
// find all ancestors of the list
// item that have a class of "module" var
$modules = listItem.parents( '.module' );
```

```
// find the closest ancestor of the
// list item that has a class of "module"
var $module =
    listItem.closest( '.module' );
```



# style

- If you want to change the style/presentation of elements, use classes combined with CSS rules, and only use jQuery to add and remove those classes as necessary.

```
$( 'li' ).addClass( 'hidden' );
```

```
$( 'li' ).eq( 1 ).removeClass( 'hidden' );
```

Or...

```
$( 'li' ).eq( 1 ).toggleClass( 'hidden' );
```

# Moving and Removing

- see `jquery_Example2.html`

# Events

- Selects all list items on the page, then binds a handler function to the click event of each list item using jQuery's `.click()` method.

```
$( 'thingToAffect' ).click();  
$( 'li' ).click(function( event ) {  
    console.log( 'clicked', $(this).text() );  
});
```

# Dynamically Created Events

- What if you interact with items that didn't exist when the DOM was loaded? (dynamically added)
- Use the event handler `.on()`. You can think of `.on()` as a general handler that takes the event, its selector, and an action as inputs.

```
$(document).on('event', 'selector', function() {  
    //Do something!  
});
```

```
$(document).on('click', '.item', function() {  
    $(this).remove()  
});
```

- When an event is triggered, the event handler function receives one argument, an event object that is normalized across browsers.

```
$( document ).on( 'click', function( event ) {  
    console.log( event.type );  
    console.log( event.which );  
    console.log( event.target );  
    console.log( event.pageX );  
    console.log( event.pageY );  
} );
```

Event handling function gets access to raw DOM element that initiated the event.

```
$( 'input' ).on( 'keydown', function( event ) {  
    // Change input element's background to red  
    // if backspace was pressed, otherwise green.  
  
    $( this ).css( 'background',  
                    event.which === 8 ? 'red' :  
                    'green' );  
});
```

# Nested elements

- What happens when you click on an `<a>` element that's nested inside other elements?
- The click event will be triggered for the `<a>` element as well as for all of the elements that contain the `<a>` — all the way up to the document and the window.

# Event delegation

Event delegation (to parent elements) allows us to bind fewer event handlers than we'd have to bind if we were listening to clicks on individual elements.

```
$( '#my-other-list a' ).on( 'click', function( event ){  
    console.log( event.target );  
    // logs the element that initiated the event  
});
```

```
$( '#my-other-list' ).on( 'click', 'a', function( event ) {  
    console.log(event.target);  
});
```