jQuery Study Guide : ASD - Week 1

While watching the videos, take notes on the methods and syntax for jQuery, then refactor your JavaScript from VFW. The most commonly used jQuery methods needed to refactor your projects are listed below. Pay special attention to selectors, traversal, events and manipulation techniques in addition to those methods. For more information, don't forget about the jQuery docs at http://docs.jquery.com/

The methods listed in the parentheses below are not an exhaustive list! Spend some time going through the docs, http://dochub.io/#jquery/ or http://woorkup.com/wp-content/uploads/2011/12/jQuery-17-Visual-Cheat-Sheet.pdf for a printed cheat sheet.

The jQuery Factory

Using the $ next to open brackets, e.g. $();, is called the factory. It allows us to grab and select elements within the DOM.

Selectors

(single element, parent-child, siblings, multiples, filters)

basic selectors:

$("#myid") $(".myclass") $("p")

$("#myid > .topnav:first a[href][rel]"}

To descend through:

$("#myid ul li") spaces indicate where to go in the code. It will go through the my id and find any ul and find any li inside that ul.

Children and parents:

Right bracket is children selector.

$("div.hilite > a") div.hilite grabs the div with the class of hilite and the > a grabs any child anchors of that div

Siblings:

using + will select all siblings that appear immediately after the first selector.

ex: $(“div + p”);

Using ~ will select all siblings before or after the selector that match
ex: $(“div ~ p”);

You can select multiple selectors by separating them with commas. The comma acts as an or operator.
Ex. $(“div, p”);

Attribute selector:
Using square brackets, you can target a specific attribute. Ex. $(“img[title]”)
Using ^ = selects text at the beginning of an attribute’s value:
Ex. $(“a[href=^http://]”)

Use $= to select text at the end of an attribute
ex. $(“img[src$=.jpg]”)
Match text anywhere, use *=
$(“img[src*=puppy]”)
match where text is not inside by using !=
$(“img[src!=kitten]”)

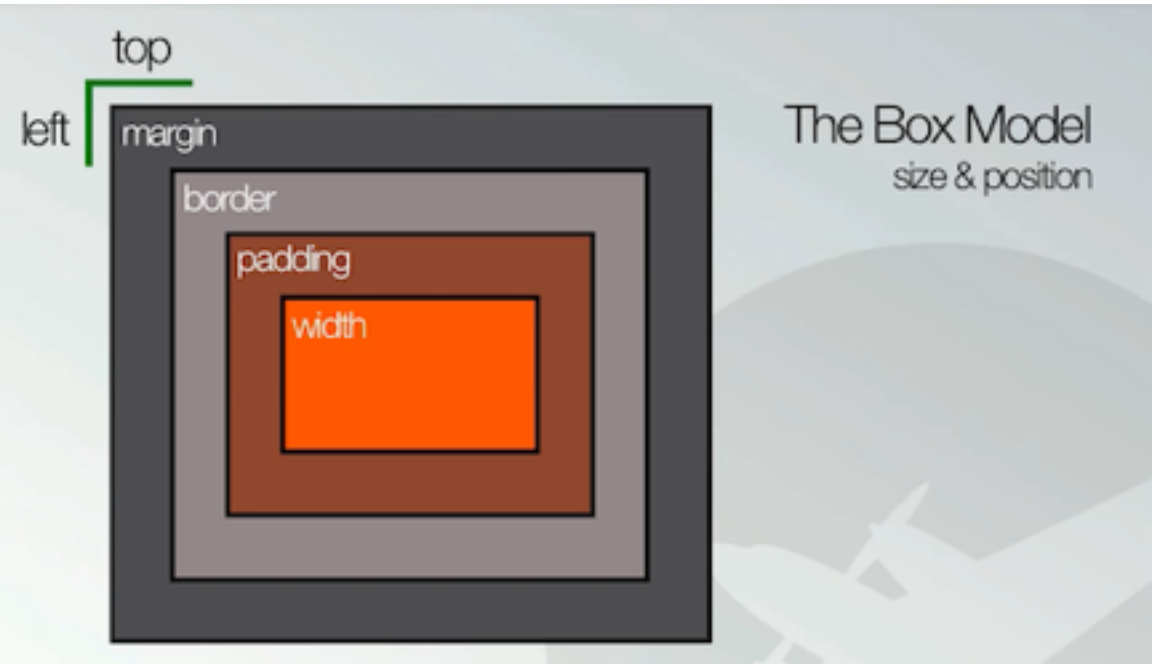
Filters:
Use a colon, no space.
ex. $(“li:first”) $(“li:even”)

jQuery CSS, Classes and Attributes
(.css .addClass .removeClass .toggleClass .attr)

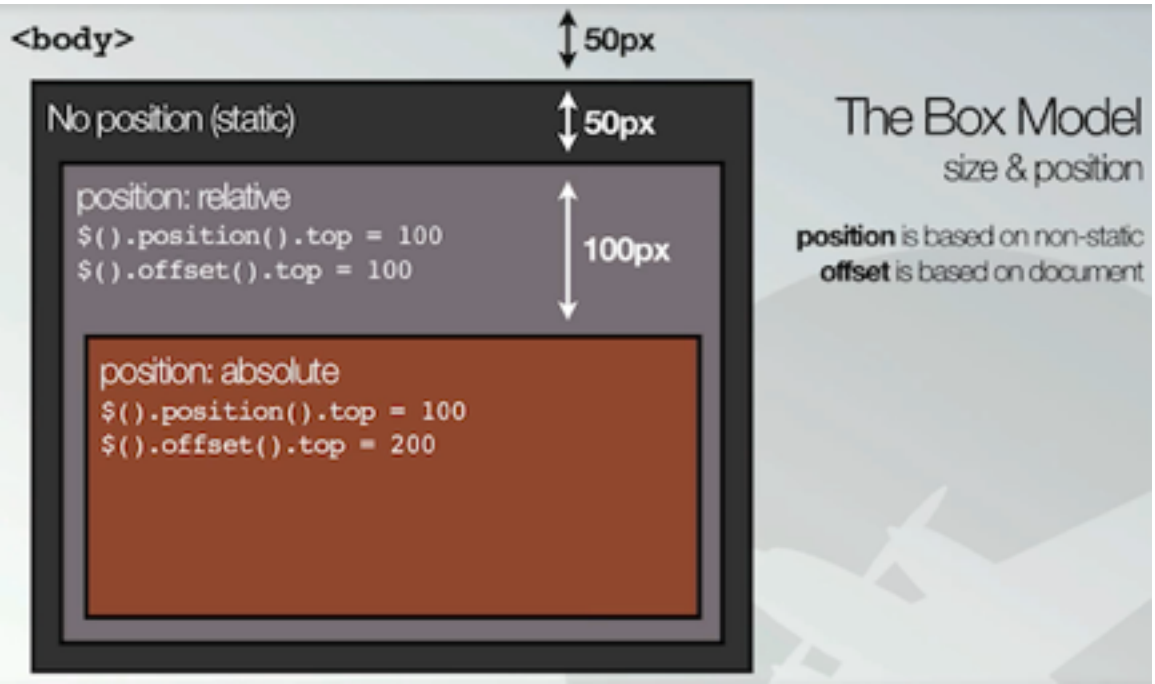
```
$().css(string)
$().css(object)
ex.
var padbot = $(‘#nav’).css(“paddingBottom”);
$(‘#nav’).css(“backgroundColor”, “#ff0000”);

$(‘#nav’).css({
        backgroundColor: “#ffffff”,
        height: 100,
        padding: “5px”
});

$().addClass(string)
```

$().attr(string)
ex.
var navalt = $('a').attr("alt");
$('a').attr("href", "http://google.com");

$('a').attr({
	title: "link",
	href: "http://google.com",
	alt: "Google"
});



$().height()
$().width()
$().position()
$().offset()



$().scrollTop()
$().scrollLeft()

target.add(selector
target.find(selector)

Traversing

(.find .children …and many other filters. May want to write down the ones you think will be useful for your apps.)

Traversing Methods

Method	Operation
.contents( )	Returns a set of the children of the previous set *(including text nodes) Also works on IFrames*
.children(s)	Returns a new set of the *direct children* of the previous set
.siblings(s)	Returns a set of all the siblings of the previous set
.closest(s)	Searches up the Ancestor tree for the *selector. Reduces the set to 1.*
.parent(s)	Matches the *direct* parent of the matches, with optional selector
.parents(s)	Matches *all* ancestors of the matches, with optional selector
.offsetParent( )	Finds the parents of the matches that are not position static
.parentsUntil(s)	Matches *all* ancestors of the matches, *up to but not including* the given selector.

target.map(function(i,elem){})
ex.

```
$(“div”).map(function(i, elem){
        if( $(elem).css(“position”) ===”absolute”){
                return elem;
        };
});
```

Manipulation

(.html .text .append .appendTo .wrap .insertAfter .empty)

appendTo will allow you to edit the new element you made, append will allow you to edit the entire element that your new element is being appended to.

appendTo can also be used to move existing html elements into a new place.

Useful to make siblings instead of children

Used to wrap content in html markup

target.empty()
target.remove()
target.detach()
use detach to retain jQuery data instead of destroying everything. Useful for moving.

target.clone(boolean)

Events

****Note that although the JQM docs and the videos are still using .bind(), jQuery has been updated to use .on(). I recommend using .on()!**
(.on .off e.preventDefault .toggle)
(jQM has its own events - see http://jquerymobile.com/demos/1.1.1/docs/api/events.html
“tap” can be used instead of “click”, but both will work)

return false should be used almost every time to prevent any unwanted bubbling effects.

Don’t use mouseover or mouseout, instead use mouseenter and mouseleave

Do not use focus and blur, use instead focusin and focusout

Tips for success: Document Setup

As Mike says in the video, when using jQuery you want to check to make sure the DOM is ready to receive the instructions you're going to give. Normally with jQuery, we check for this with:

```
1  $(document).ready(function(){
2      // Your code here
3  });
```

When using jQuery with jQuery Mobile, we want to check to make sure each jQuery Mobile page is ready when the page is initialized. To do this, you're going to want to check for the 'pageinit' event before running any code needed for that page:

```
1  $("#homepage").on('pageinit', function(){
2
3      // Code needed for homepage goes in this function
4
5  });
```

You will need to listen for the pageinit event for each jQM page, and then add the code needed for each particular page inside of the appropriate pageinit function. For example, the page that contains your form should have its own pageinit check, with the JavaScript code needed to handle the form inside:

```
8  $("#myForm").on('pageinit', function(){
9
10     // Form code goes here
11
12 });
```

Check out http://jquerymobile.com/demos/1.1.1/docs/api/events.html for more information.