jQuery & Mobile Dev Continued

AIM Code School By: Vanessa Kasun

Traversing

jQuery traversing, which means "move through", are used to "find" (or select) HTML elements based on their relation to other elements. Start with one selection and move through that selection until you reach the elements you desire.

The image below illustrates an HTML page as a tree (DOM tree). With jQuery traversing, you can easily move up (ancestors), down (descendants) and sideways (siblings) in the tree, starting from the selected (current) element. This movement is called traversing - or moving through - the DOM tree.

At times the general CSS selectors alone don't cut it and a little more detailed control is desired. Fortunately jQuery provides a handful of methods for traversing up and down the DOM tree, filtering and selecting elements as necessary.

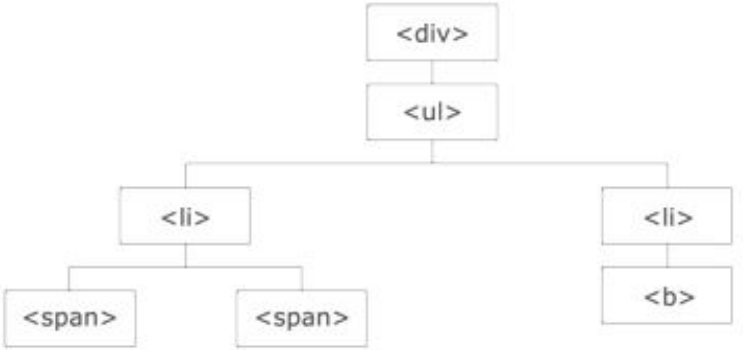
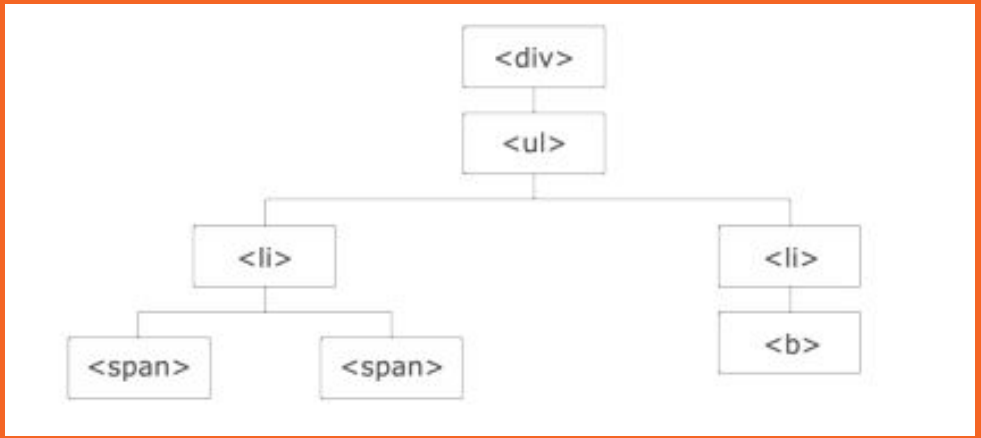


Illustration explained:

- The element is the parent of <ul>, and an ancestor of everything inside of it
- The <ul> element is the parent of both <li> elements, and a child of
- The left <li> element is the parent of , child of <ul> and a descendant of
- The element is a child of the left <li> and a descendant of <ul> and
- The two <li> elements are siblings (they share the same parent)
- The right <li> element is the parent of <b>, child of <ul> and a descendant of
- The <b> element is a child of the right <li> and a descendant of <ul> and



To get started with filtering elements inside the DOM a general selection needs to be made, from which will be traversed from relatively. In the example the original selection finds all of the div elements in the DOM, which are then filtered using the **.not()** method. With this specific method all of the div elements without a class of type or collection will be selected.

```
$('div').not('.type, .collection');
```

—

Miscellaneous Traversing

- .add()
- .end()
- .andSelf()
- .contents()

Filtering

-
- .eq()
- .has()
- .map()
- .filter()
- .is()
- .not()
- first()
- last()
- .slice()

DOM Tree Traversal

- .children()
- .next()
- .offsetParent()
- .parentsUntil()
- .prevUntil()
- .closest()
- .nextAll()
- .parent()
- .prev()
- .siblings()
- .find()
- .nextUntil()
- .parents()
- .prevAll()

Traversing Methods

jQuery has quite a few traversing methods available to use. In general, they all fall into three categories, filtering, miscellaneous traversing, and DOM tree traversing.

[See traverse method definitions here]

Manipulation

Mwahahahaha

Manipulation

Selecting and traversing elements in the DOM is only part of what jQuery offers, one other major part is what is possible with those elements once found. One possibility is to manipulate these elements, by either reading, adding, or changing attributes or styles.

Additionally, elements may be altered in the DOM, changing their placement, removing them, adding new elements, and so forth. Overall the options to manipulate elements are fairly vast.

Getting & Setting

The manipulation methods to follow are most commonly used in one of two directives, that being **getting** or **setting** information. **Getting** information revolves around using a selector in addition with a method to determine what piece of information is to be retrieved. Additionally, the same selector and method may also be used to set a piece of information.

```
// Gets the value of the alt attribute
$('img').attr('alt');

// Sets the value of the alt attribute
$('img').attr('alt', 'Wild kangaroo');
```

Attribute Manipulation

One part of elements able to be inspected and manipulated are attributes. A few options include the ability to add, remove, or change an attribute or its value. In the examples the **.addClass()** method is used to add a class to all even list items, the **.removeClass()** method is used to remove all classes from any paragraphs, and lastly the **.attr()** method is used to find the value of the **title** attribute of any **abbr** element and set it to ***Hello World.***

```
$('li:even').addClass('even-item');
$('p').removeClass();
$('abbr').attr('title', 'Hello World');
```

Attribute Manipulation Methods

- .addClass()
- .prop()
- .removeProp()
- .attr()
- .removeAttr()
- .toggleClass()
- .hasClass()
- .removeClass()
- .val()

jQuery Attribute API. Please take a look!

Style Manipulation

Along with attribute manipulation, the jQuery library also include various methods to manipulate style properties and CSS class of DOM element(s).

- .css()
- .innerWidth()
- outerWidth()
- scrollTop()
- .height()
- .offset()
- .position()
- .width()
- .innerHeight()
- .outerHeight()
- .scrollLeft()

jQuery Style Properties API.
Please take a look!

The height, width, or position methods all default to using pixel values, however other units of measurement may be used. As seen below, to change the unit of measurement, identify the value then use a plus sign followed by the quoted unit of measurement.

```
$('h1 span').css('font-size', 'normal');
$('div').css({
  fontSize: '13px',
  background: '#f60'
});
$('header').height(200);
$('.extend').height(30 + 'em');
```

DOM Manipulation

Lastly, we are able to inspect and manipulate the DOM, changing the placement of elements, adding and removing elements, as well as flat out altering elements. The options here are deep and varied, allowing for any potential changes to be made inside the DOM.

Each individual DOM manipulation method has its own syntax but a few of them are outlined below. The **.prepend()** method is adding a new **h3** element just inside any section, the **.after()** method is adding a new **em** element just after the link, and the **.text()** method is replacing the text of any **h1** elements with the text ***Hello World.***

```
$('section').prepend('<h3>Featured</h3>');
$('a[target="_blank"]').after('<em>New window.</em>');
$('h1').text('Hello World');
```

DOM Manipulation Methods

- .after()
- .before()
- .empty()
- .insertBefore()
- .remove()
- .text()
- .wrapAll()
- .append()
- .clone()
- .html()
- .prepend()
- .replaceAll()
- .unwrap()
- .wrapInner()
- .appendTo()
- .detach()
- .insertAfter()
- .prependTo()
- .replaceWith()
- .wrap()

jQuery DOM API. Please take a look!

***There are 5 different tabs for DOM manipulation. Please take a look at all of them.**

Event Methods Overview

jQuery provides quite a few methods, all of which are based around registering user behaviors as they interact with the browser. These methods register quite a few events, most popularly, but not limited to, browser, form, keyboard, and mouse events.

Here is a full list of all event methods. Please take a look!

Effects Overview

Next to events, jQuery also provides a handful of customizable effects. These effects come by the way of different methods, including event methods for showing and hiding content, fading content in and out, or sliding content up and down. All of these are ready to use methods and may be customized as best see fit.

Here is a full list of effects. Please take a look!

Let's Demo!

Open Codepen and make sure to save these demos for future reference!

1. Create a ** ** tag with the class attribute of ***panel***
2. Inside of the previous ** ** tag, place another ** ** tag with the class attribute of **panel-stage** *(this will not have any content so nothing will go between the opening and closing tag)*
3. Underneath that ** ** still inside the original ** ** element with the class ***panel;*** create an **<a>** tag with the **href** attribute of **#** and a **class** attribute of ***panel-tab.***
4. The content that will go with the <a> tag will read as the following:

```
Open <span>&#9660;</span>
```

The HTML code will look like this:

```
<div class="panel">
  <div class="panel-stage"></div>
  <a href="#" class="panel-tab">Open <span>&#9660;</span></a>
</div>
```

For the JS File...

Continue to next page to for last step once completed with step 4.

1. Using the correct jQuery syntax, select the **panel-tab** class and tell the code that when the user **clicks on** it, to **prevent** the **default** action from occurring. This **function** will be **anonymous** and will pass in **(event)** as its parameter.

2. In the next function statement select the ***panel-stage*** class and apply the **slideToggle** effect. Set the speed at which this effect will take place as **slow,** while also calling back the **anonymous function** created in step one (**don't forget the parameter)**. But this time the function body will contain an ***if/else*** statement.

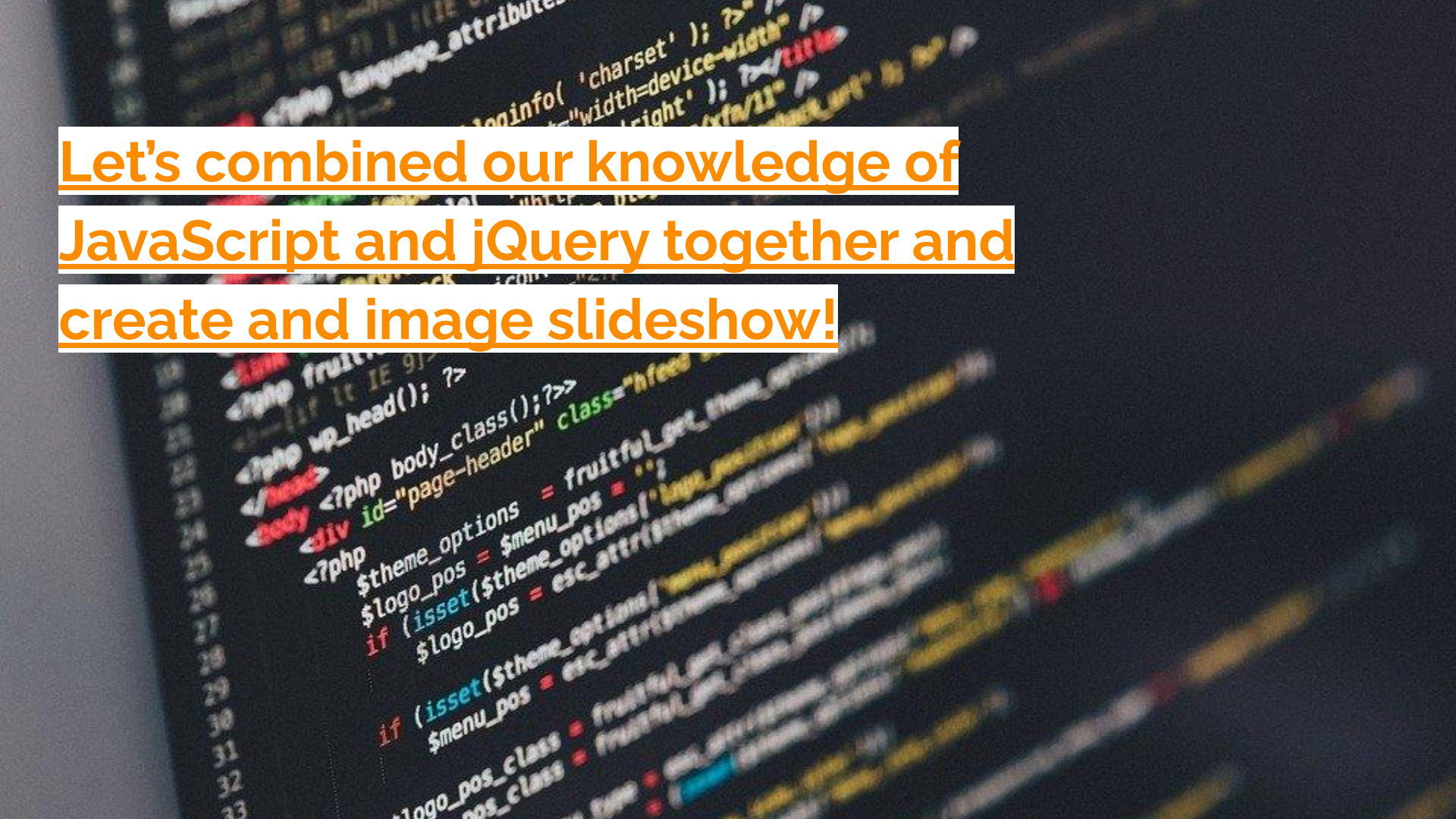
3. The ***if*** conditional statement will select the element specified by the user's click and be made **:visible (this step may be a bit tricky. If you're stuck, think about what we did in our javascript to-do-list and calculator examples to target the specified element the user clicks on)**

4. The ***if*** body statement will select the ***panel-tab*** class and apply the **attribute manipulator** that **sets** the HTML content of the ***panel-tab*** "element" with the string **'Close #&9650;'**

For the JS File...

5. In the **else** portion of the statement, select the ***panel-tab*** Class again and repeat step four. **EXCEPT**, this time the String content with read **'Open ▼'**

Here is a link to the css for this practice assignment


Let's combined our knowledge of JavaScript and jQuery together and create and image slideshow!