THE ANATOMY OF

jQuery Functions

```
$('nav').addClass('visible');
```

Selector   Function   Parameter

THE BASICS

The above presents the most basic example of a jQuery functions which would actually cause a change to happen. It is composed of a **selector**, wherein the contents of the quotes function exactly like a CSS selector; a **function**, which is defined in jQuery and applies a specific action to the selector; and a **parameter**, which tells the function what value to use.

In this case, the **addClass** function adds the class **visible** to the **nav** element.

MAKING IT PRACTICAL

Generally, a function like that above needs to be triggered by user interaction. For example, clicking on a button can add a class to an item to make it viewable – useful for toggling visibility. This requires an extra level of depth to the function. Below is an example of triggering an action based on the clicking of a button.

Selector of Trigger Item   Function   Empty Parameters

```
$('.toggler').click(function(){
$('nav').addClass('visible');
});
```

```
<a class='toggler' href='#'>Toggle</a>        <nav>...</nav>
```

The clicking of this link would add the class visible to the nav.

THE ANATOMY OF jQuery Functions

```
$('.toggler').click(function(){
$('nav').addClass('visible');
});
```

BREAKING IT DOWN

Above, we are embedding one function inside another. **click()** is a function which must be attached to an element selector (**.toggler** in this case) because it must know what must be clicked to kick off the function. In between the braces, you may place any number of actions (in the form of functions) to be performed upon clicking of the first element.

Upon click, we are creating a* ***function()* *which houses the actions we want performed. The parentheses are left blank because we're not provided other information; that is, everything will be provided between the braces that follow. There are cases when parameters are necessary here, but we'll rarely if ever use them in jQuery.*

```
$('section').click(function(){          start a function when a section is clicked
$('section').removeClass('visible');    take away the visible class from all section elements
$(this).addClass('visible');            take away the visible class from all section elements
});                                     end the function(){ and the click() function.
```

this *is a javascript variable which refers to the item that triggered the action. Variables used as selectors are not surrounded by quotes.*

HTML
```
<section>
<h2>Interesting Title</h2>
<p>...</p>
</section>
<section>...</section>
<section>...</section>
```

CSS
```
section p {
display: none;
}
section.visible p {
display: block;
}
```

The result of this code would be that upon page load, you would see only the h2 elements, and then when any section was clicked, the following: All sections would have the visible class removed, and the one that was clicked would have the class added to it. We do it in this order so that only one section is “open” at a time.

THE ANATOMY OF

jQuery Functions

Putting Them to Use

```
$(document).ready(function(){
  $('.toggler').click(function(){
  $('nav').toggleClass('visible');
  return false;
  });
});
```

When all page elements are loaded, initiate jQuery functions

Since we're using a click function, we tell the browser not to obey the href value so the page doesn't jump. Look this up!

WHAT'S GOING ON

So before we start running the fun stuff, we have to wrap all our functions which are to come in one which first checks that all our page elements are loaded: **$(document).ready**. This ensures that we can actually manipulate items because they are loaded and styled first.

HTML

```
<nav>
<a class='toggler' href='#'>
Toggle Nav</a>
<ul>...</ul>
</nav>
```

CSS

```
nav ul {
display: none;
}
nav.visible ul {
display: block;
}
```

Skipping presentation, the above jQuery, HTML and CSS are the basic requirements needed for a togglable navigation list.

Notice two key things:

1) We are changing the class of the <nav> element, then styling its ul based upon its context (whether its nav has a class).

2) We leave the toggler button outside the ul, so that when the ul disappears, the nav still has content in it and therefore still shows up on the screen. This would be a good reason to apply a background color to the nav rather than just the nav > ul.