Scroll Based Animation

<https://michalsnik.github.io/aos/>

In our script we will be referencing both the window object and the collection of elements we want to animate.

//Cache reference to window and animation items

var $animation\_elements = $('.animation-element');

var $window = $(window);

Notice the dollar sign in front of the variables. This is a convention to indicate that they hold a jQuery object, or collection of objects.

### Hooking into the Scroll Event

Next, we create our event handler that listens for the scroll event. This will fire when we scroll the page. We pass it a reference to our check\_if\_in\_view function (which we’ll get to in a minute). Every time the scroll event is fired, this function will be executed.

$window.on('scroll', check\_if\_in\_view);

### Handling Resizing

Because we are calculating heights and widths we need to factor in orientation changes along with general resizing.

We can update our event handler to listen for both the scroll and resize events. This will enable our detection function to work when we resize or change orientation.

$window.on('scroll resize', check\_if\_in\_view);

In addition, we also use the jQuery trigger method to trigger a scroll event as soon as the DOM is ready. We do this so that if any of the elements which should be animated are within the viewport, they will be detected as in view and the animation applied as if we had scrolled.

$window.trigger('scroll');

### Scroll Position Detection

The actual detection portion of this example comes from the following script.

function check\_if\_in\_view() {

var window\_height = $window.height();

var window\_top\_position = $window.scrollTop();

var window\_bottom\_position = (window\_top\_position + window\_height);

$.each($animation\_elements, function() {

var $element = $(this);

var element\_height = $element.outerHeight();

var element\_top\_position = $element.offset().top;

var element\_bottom\_position = (element\_top\_position + element\_height);

//check to see if this current container is within viewport

if ((element\_bottom\_position >= window\_top\_position) &&

(element\_top\_position <= window\_bottom\_position)) {

$element.addClass('in-view');

} else {

$element.removeClass('in-view');

}

});

}

Lets break down what is happening here.

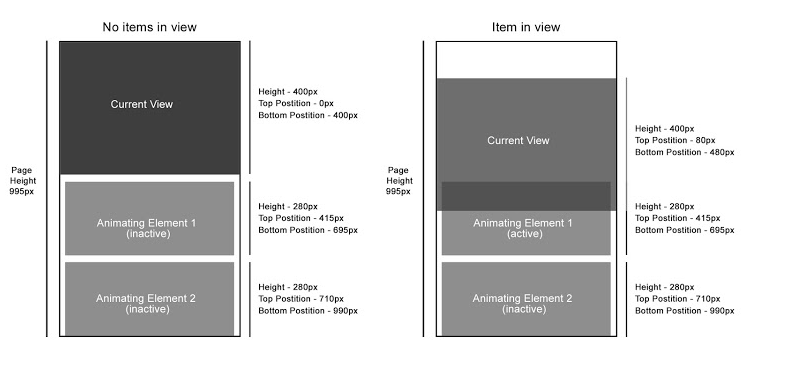
The check\_if\_in\_view function is called initially when the DOM is ready and then every time we resize or scroll.

We get the current height of the window, along with its top and bottom position so we know what area we are looking at.

We go through and look for all items that will be animating in (saved in the $animation\_elements variable). For each of these elements we collect its height along with its top and bottom position (so we know where it lives on the page).

We compare each item to see if its bottom position is greater than the top position of the window but also that the item’s top position is less than the bottom position of the window.

Here is a visual example

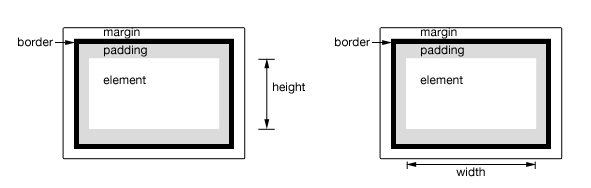


### Calculating the Height and Width

In our detection function we need to get the heights and positions of various elements to calculate things correctly, this is where we have used jQuery’s height functions. It’s important to have a breakdown of how these height functions work

#### height() and width()

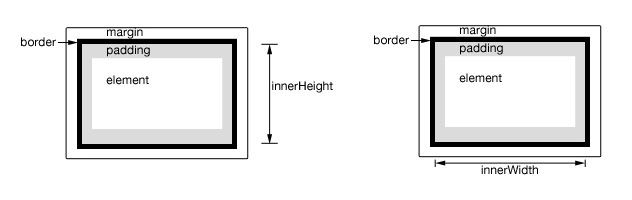
The height() and width() functions return the height or width of an element. They exclude all padding, borders and margins.



For a full breakdown visit the [height](http://api.jquery.com/height/) or [width](http://api.jquery.com/width/) documentation.

#### innerHeight() and innerWidth()

The innerHeight() and innerWidth() functions return the height or width of the element including its additional padding (however it excludes both borders and margins)

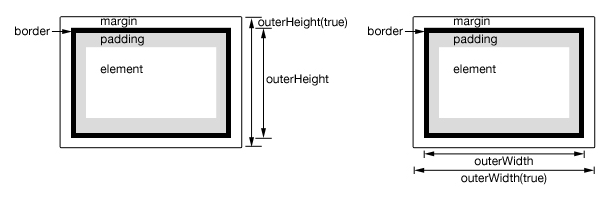


For a full breakdown visit the [innerHeight](http://api.jquery.com/innerHeight/) or [innerWidth](http://api.jquery.com/innerWidth/) documentation.

#### outerHeight() and outerWidth()

The outerHeight() and outerWidth() functions return the height or width of the element and include its padding and border.

In addition you can also specify to include its margins by passing a value of true to the function.



For a full breakdown visit the [outerHeight](http://api.jquery.com/outerHeight/) or [outerWidth](http://api.jquery.com/outerWidth/) documentation

## Scroll Animation Examples

Listed below are a series of animations that use the basics of what we have discussed. These examples will look for animation elements and apply the active in-view class when they’re within the viewport.

Elements that you want to move should all have a standard class such as animation-element that sets its position to be relative or absolute. In addition, if you are going to create multiple effects you can create corresponding classes such as slide-left which can be combined with the in-view class. You should then apply the transformation to a class such as animation-element.slide-left.inview

### Slide in from Left

For our first example we will be sliding in elements from the left when they enter the viewport. We achieve this by using a translate3d on our elements x axis.

### Fade in from Bottom

This time we will be fading our elements from the bottom upwards as the user scrolls. We achieve this via a translate3d on the element’s y axis.

### Multi-Step Bouncing Animation

For our final example we’ll use a multistage animation. To do this, we’ll define custom keyframe animations that combine a rotation with a translation. This type of animation can help showcase areas of your website (for this example we are showcasing staff member profiles).