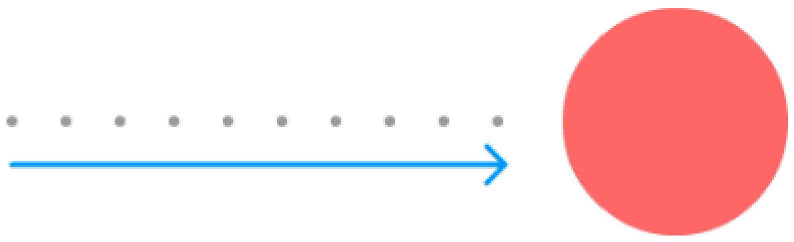
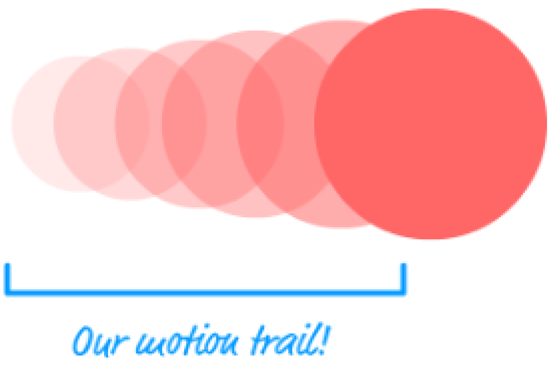
The Basic Approach

The way motion trails work is pretty simple. Let's say that we have a circle that is moving from left to right.

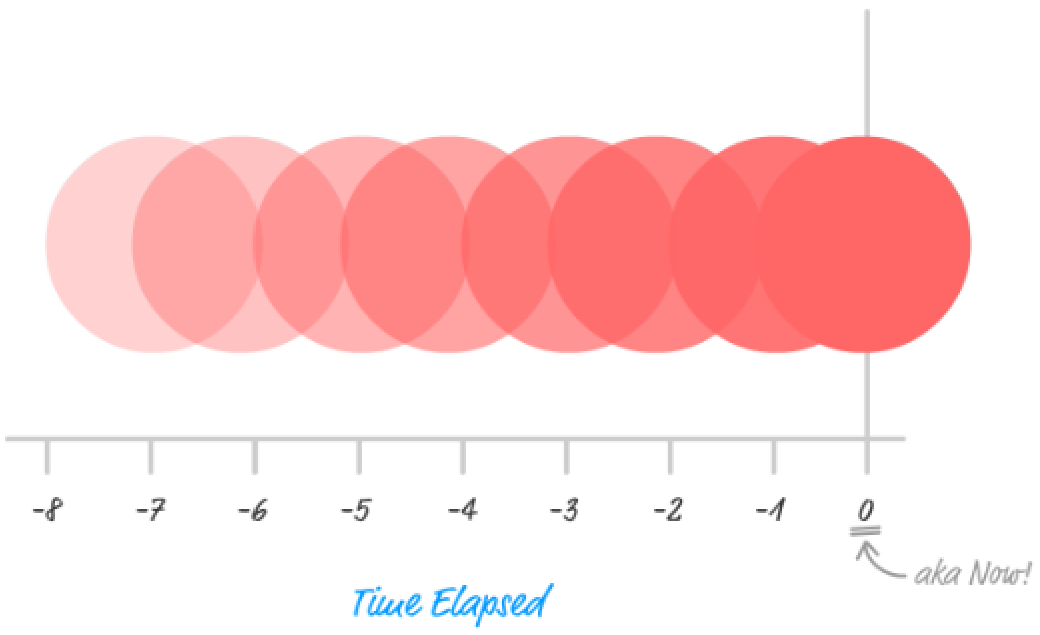


Motion trails exaggerate the direction of movement by showing you where your object was just a few moments earlier. For this circle, you may see something like this:



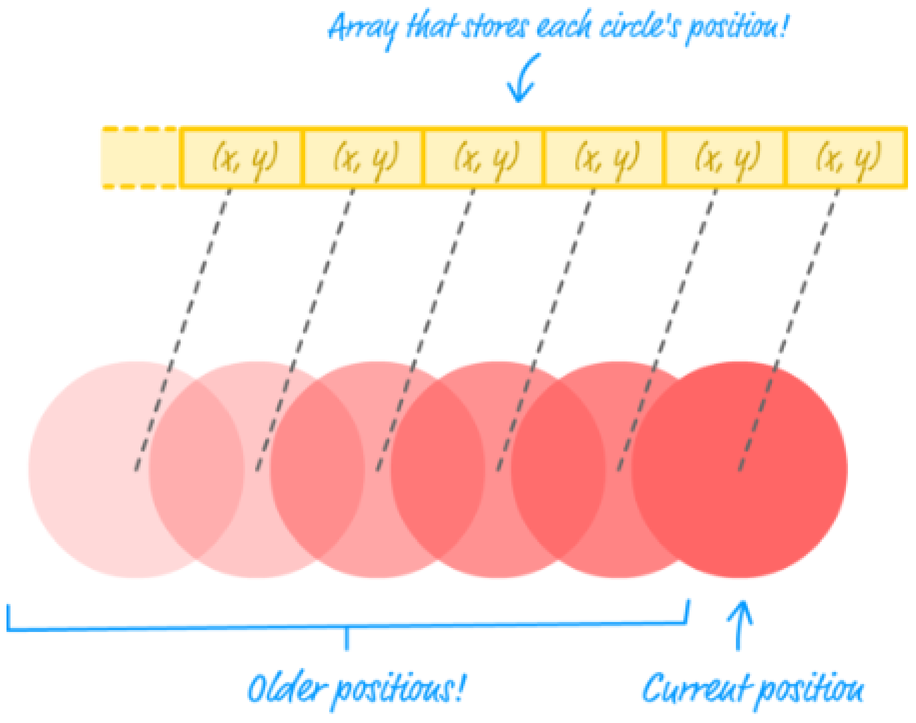
How far back your motion trails go and what it looks like are all things under your control. In this case, our motion trail gets smaller and more faded the further away from the source that you go.

Thinking about motion trails is easy. Visualizing them is pretty easy. Implementing them is pretty easy as well...once you understand how exactly they work. The main thing to note is that your motion trails show where your moving object **has been in the past**. To look at that more precisely, take a look at the following diagram:



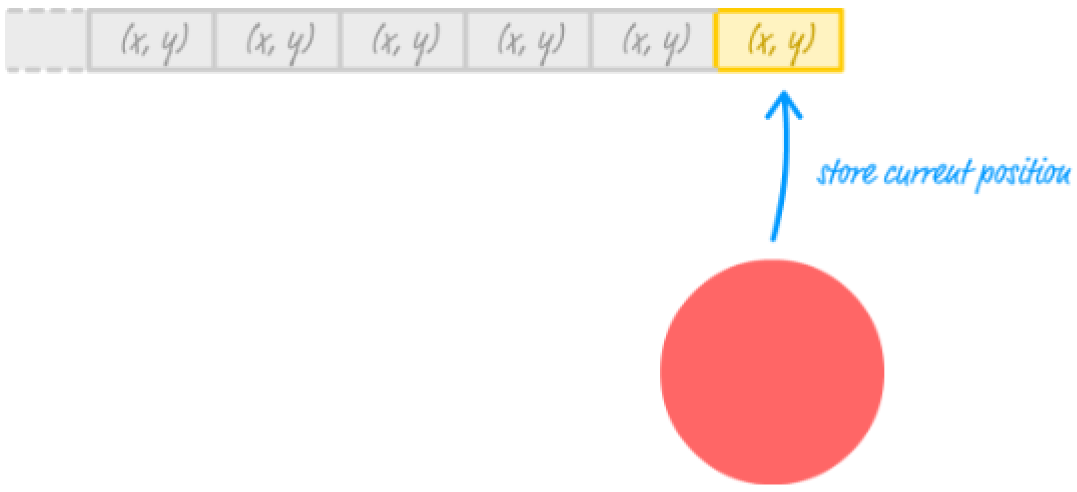
Looking at each part of our motion trail as a slice of time, you can see that at time 0, we have the **source object** we are moving. At each slice of time prior to that, we show where our source object was at that moment in the past. How can we implement such a thing? The answer lies in the **array**!

We can easily use an array to store where our source object was in the past:

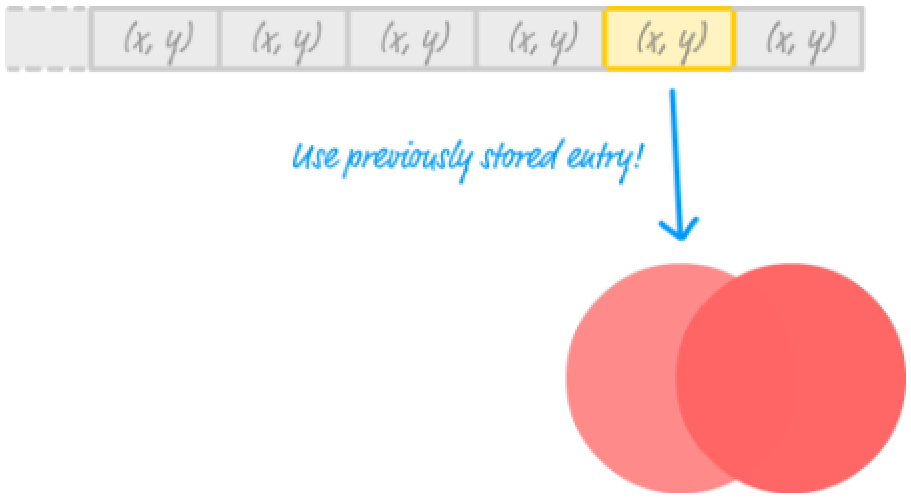


The last item in the array will be our source object's current position. Every entry prior to that represents our source object's earlier positions. All of this may sound a bit confusing at first, so let's ignore motion trails for a moment and just focus on how we populate this array.

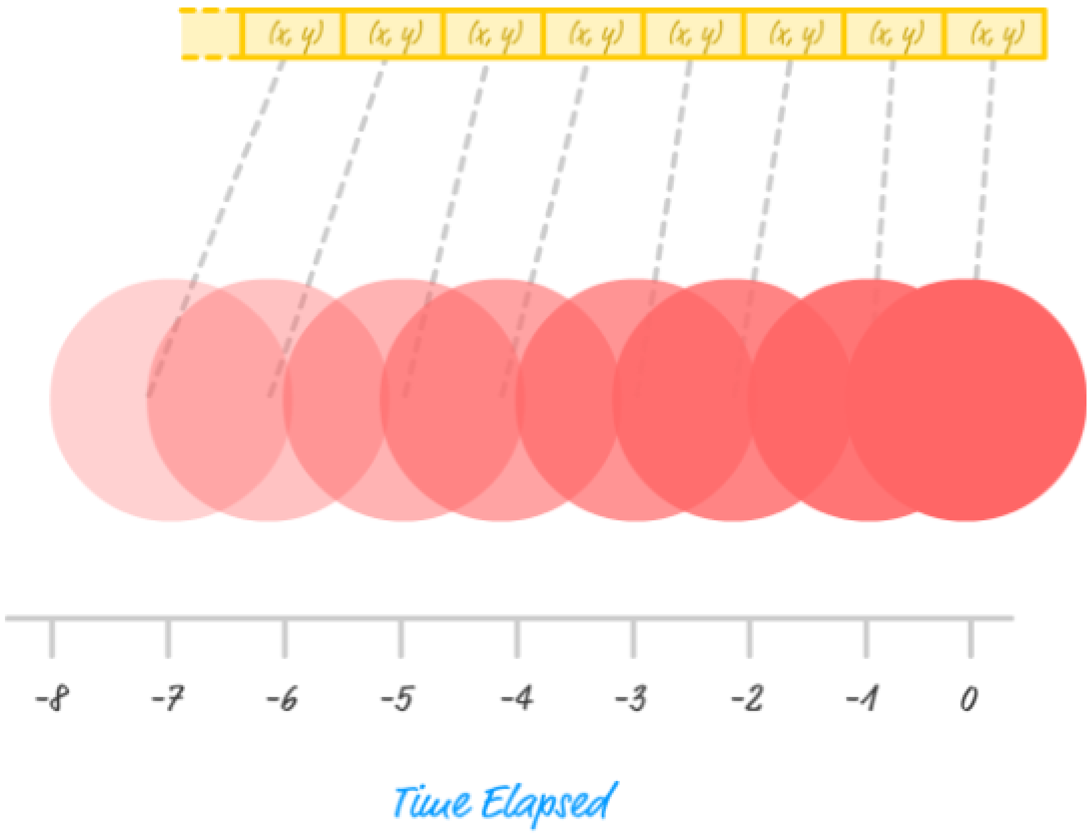
The way we populate this array is pretty simple. Every few moments, we store our source object's current position at the end of our array:



Over a period of time, our array is going to be filled with position values that map to where our source object was at the moment the array entry was populated. Our motion trail reads from this array to draw the historical snapshot of our source object:

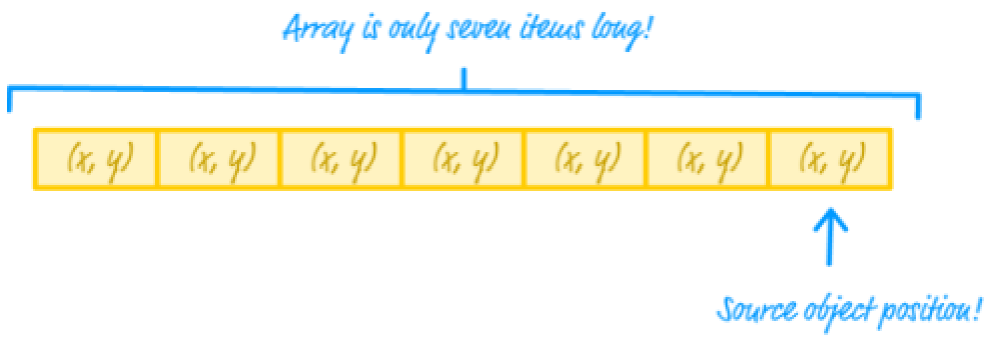


By re-drawing our source object and placing it at the points referenced by each entry in our array, we create our motion trail:



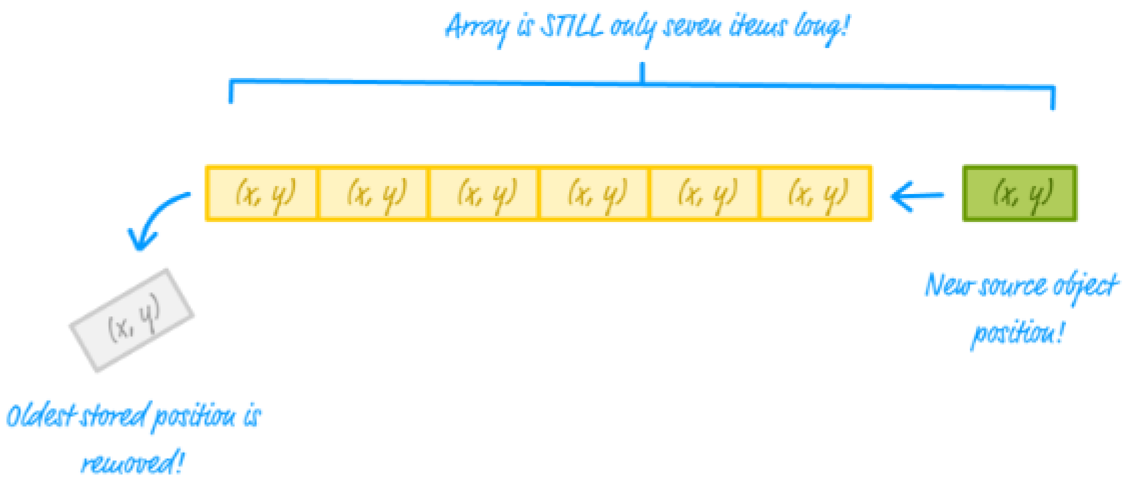
The last thing we are going to focus on is the array itself. Right now, I have given you the impression that our array has no fixed size. It just keeps growing with each new entry storing our source object's position. Unless you want a motion trail that is made up of every single place our source object has ever been, that isn't desirable. You want to restrict how big your array grows.

The size of your array determines how long your motion trail is going to be. There is nothing clever or complex about how we handle that. Let's say we want our motion trail to be made up of just seven elements. This means, we are going to limit our array's size by just seven elements as well:



Each time we add a new entry to store our source object's position, we get rid of the oldest entry:

of the oldest entry:


Array is STILL only seven items long!
(x, y)
(x, y)
(x, y)
(x, y)
(x, y)
(x, y)
(x, y)
(x, y)
New source object position!
Oldest stored position is removed!

This allows us to maintain a constant size array of seven items where the last item is our source object's position and every preceding item shows where our source object was earlier. Using an array in this way isn't anything new. What we've just described is known as a **queue** or a **first-in first-out (FIFO)** system.

Phew! By now, you probably have a really good idea of how a motion trail works. It's time to turn all of these English-looking words into JavaScript our browser can understand.