Using rotationAngle and touchRadius

By Paul Kinlan

Paul is a Developer Advocate

A relatively small change to the Touch API in Chrome landed in Chrome 39, which introduced a working version of the `webkitRotationAngle` attribute on the TouchEvent object. Now in Chrome 45 (Beta in July 2015), it is unprefixed as `rotationAngle`, bringing our implementation more inline with the TouchEvent Spec and Firefox.

Although it's been around for a while, it is worth explaining what `rotationAngle` is as it opens up some more interesting usage of touch gestures especially on mobile devices.

Technically, the rotation angle is the number of degrees (between 0 and 90) of the contact area ellipse defined by Touch.radiusX and Touch.radiusY. Err, cool, right? (It should be noted that I only found out that Chrome on Android doesn't lock the `radiusX` and `radiusY` values to 64px but instead varies it depending on the contact size of the screen).

What does it actually mean?

Think of it as a way to accurately represent the size, shape and orientation of the user's finger on a screen. Users don't always tap the screen with the nib of their fingertip, but rather frequently press the screen like they are giving the police a fingerprint. Without the `rotationAngle` you would simple get how wide and how tall the touch gesture was. With the `rotationAngle`, you get 90 degrees of rotation (0 being vertical and 90 being horizontal). Why only 90 degrees? You only need the 90 degrees because as you move past those angles the `radiusX` and `radiusY` will change to accomodate.

Another cool thing about this is that the contact area of the user's finger changes as they vary the degree of pressure of their finger on the screen. It is not a direct replacement for `force`, but you can distinguish between light brushes on the screen because they will have a smaller surface area than a harder press.

How can I use it?

Firstly you need a device that can detect this. A Nexus 10 will work fine. A great example is to look directly at Rick Byers paint example. Not to be outdone though here is a way to use it without canvas.

```
var touchHandler = function(e) {
  e.preventDefault();
  var touches = e.changedTouches;
  var touch = touches[0]; // only concerned about first touch.

  var rotationAngle = touch.rotationAngle || touch.webkitRotationAngle || 0;
  var radiusX = touch.radiusX || touch.webkitRadiusX || 1;
  var radiusY = touch.radiusY || touch.webkitRadiusY || 1;
  var force = touch.force || touch.webkitForce || 0;

  // Use the rotationAngle to rotate the 'p' element.

  p.style.width = radiusX * 2 + 'px';
  p.style.height = radiusY * 2 + 'px';
  p.style.transform = "rotate(" + rotationAngle + "deg)";
};

document.documentElement.ontouchstart = touchHandler;
document.documentElement.ontouchend = touchHandler;
document.documentElement.ontouchmove = touchHandler;
```

Where would you use this in practice?

There are some obvious places where this would be an immediate benefit to the user:

- Painting and image manipulation web apps for example could use this information to alter the stroke or effects applied to the canvas. You could use the touch radius to alter the size of the brush and you can combine in the rotationAngle to vary the angle of contact of the brush on the canvas.
- Enhanced gesture recognition: If you understand the rotationAngle you can create a single finger gesture to make an object swivel.

Does every device support this?

No. It is not hugely common... yet. If you have a Nexus 10 you will see something like the following,

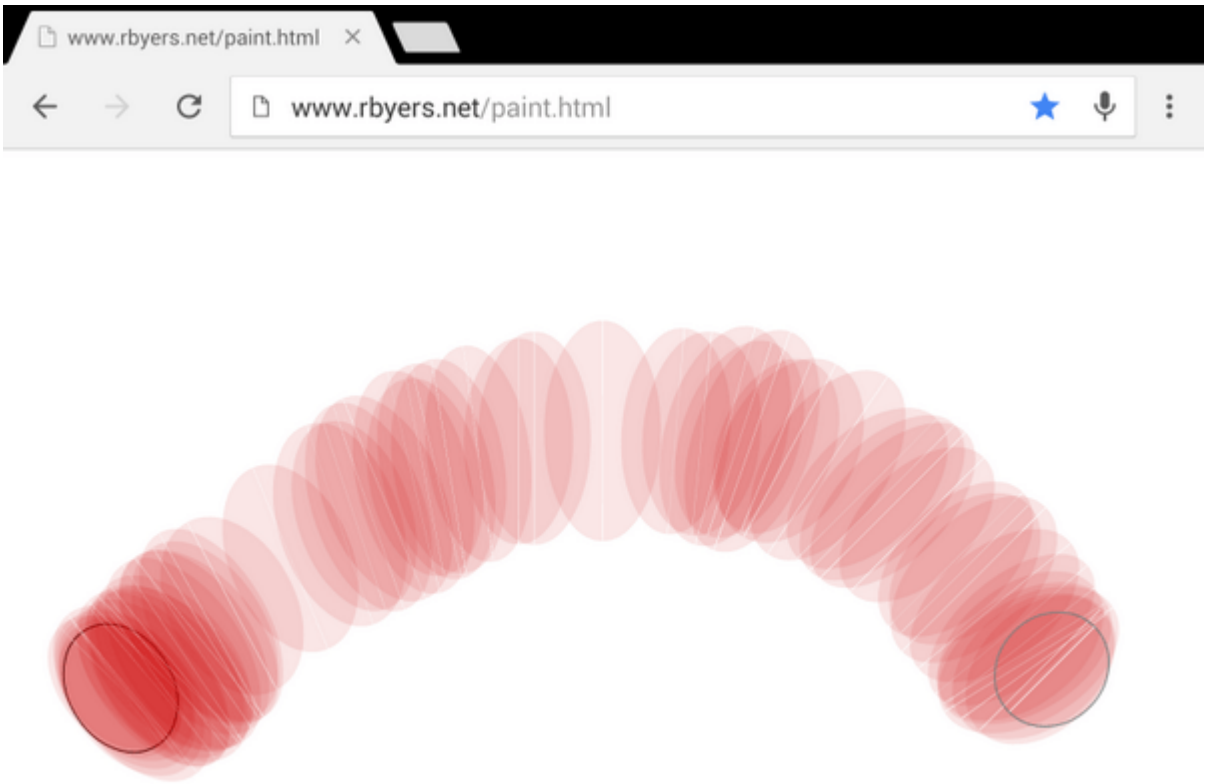

Image credit to Rick Byers.

When a device can't understand angle of rotation of a touch, the `rotationAngle` will be 0 and the `radiusX` and `radiusY` values will be equal (but may vary depending on the current touch surface area).

Why bother?

Good question. Like many features on the web, this is an additive experience.
Touch events will work when supported, and if the radii and rotation values are available you can enhance your application to give the user a more capable experience. Not every user has a Wacom tablet but when present many painting applications will take advantage of it.

What about Pointer Events?

This is really just about making sure we have a fully fleshed out touch event API for the developers who rely on it. See how I dodged the question a bit... More seriously though, if you are interested in following along Blink's PointerEvent implementation you can star Issue 471824 and read Rick Byers' tracking doc.

See Also

- Precision Touch Getures
- TouchEvent spec with rotationAngle

Last updated July 2, 2018.