Hit regions and accessibility

« Previous
Next »

The `<canvas>` element on its own is just a bitmap and does not provide information about any drawn objects. Canvas content is not exposed to accessibility tools like semantic HTML is. In general, you should avoid using canvas in an accessible website or app. The following guidelines can help to make it more accessible.

Fallback content

The content inside the `<canvas> ... </canvas>` tags can be used as a fallback for browsers which don't support canvas rendering. It's also very useful for assistive technology users (like screen readers) which can read and interpret the sub DOM in it. A good example at html5accessibility.com demonstrates how this can be done:

```
<canvas>
  <h2>Shapes</h2>
  <p>A rectangle with a black border.
  In the background is a pink circle.
  Partially overlaying the <a href="http://en.wikipedia.org/wiki/
  Partially overlaying the circle is a green
  <a href="http://en.wikipedia.org/wiki/Square" onfocus="drawSqua
  and a purple <a href="http://en.wikipedia.org/wiki/Triangle" or
  both of which are semi-opaque, so the full circle can be seen
</canvas>
```

See the video how NVDA reads this example by Steve Faulkner.

ARIA rules

Accessible Rich Internet Applications (ARIA) defines ways to make Web content and Web applications more accessible to people with disabilities. You can use ARIA attributes to describe the behavior and purpose of the canvas element. See ARIA and ARIA techniques for more information.

```
<canvas id="button" tabindex="0" role="button" aria-pressed="false
```

Hit regions

Whether the mouse coordinates are within a particular area on the canvas, is a common problem to solve. The hit region API allows you to define an area of your canvas and provides another possibility to expose interactive content on a canvas to accessibility tools. It allows you to make hit detection easier and lets you route events to DOM elements. The Path2D has the following three methods (which are still experimental in current web browsers; check the browser compatibility tables).

`CanvasRenderingContext2D.addHitRegion()`
Adds a hit region to the canvas.

`CanvasRenderingContext2D.removeHitRegion()`
Removes the hit region with the specified `id` from the canvas.

`CanvasRenderingContext2D.clearHitRegions()`
Removes all hit regions from the canvas.

You can add a hit region to your path and check for the `MouseEvent.region` property to test if your mouse is hitting your region, for example.

```
<canvas id="canvas"></canvas>
<script>
var canvas = document.getElementById('canvas');
var ctx = canvas.getContext('2d');

ctx.beginPath();
ctx.arc(70, 80, 10, 0, 2 * Math.PI, false);
ctx.fill();
ctx.addHitRegion({id: 'circle'});

canvas.addEventListener('mousemove', function(event) {
  if (event.region) {
    alert('hit region: ' + event.region);
  }
});
</script>
```

The `addHitRegion()` method also takes a `control` option to route events to an element (that is a descendant of the canvas):

```
ctx.addHitRegion({control: element});
```

This can be useful for routing to `<input>` elements, for example. See also this codepen demo.

Focus rings

When working with the keyboard, focus rings are a handy indicator to help navigating on a page. To draw focus rings on a canvas drawing, the `drawFocusIfNeeded` property can be used.

`CanvasRenderingContext2D.drawFocusIfNeeded()`
If a given element is focused, this method draws a focus ring around the current path.

Additionally, the `scrollPathIntoView()` method can be used to make an element visible on the screen if focused, for example.

`CanvasRenderingContext2D.scrollPathIntoView()`
Scrolls the current path or a given path into the view.

See also

- Canvas accessibility use cases
- Canvas element accessibility issues
- HTML5 Canvas Accessibility in Firefox 13 – by Steve Faulkner
- Best practices for interactive canvas elements

« Previous
Next »

Tags: Canvas, Graphics, Tutorial
Contributors to this page: peterj35, stephaniehobson, nmve, onurtemizkan, baker1797, fscholz
Last updated by: peterj35, Jan 16, 2018, 8:13:05 AM