Getting the Exact Mouse Position

Following up on a thread we left unexplored a few moments ago, the `clientX` and `clientY` properties don't give you the exact mouse position in many situations. Understanding the details of why is something that I cover in this article, but the gist of it is that the `clientX` and `clientY` properties don't account for your `canvas` element physically being pushed around by all of its ancestors. A margin or padding here and there can cause your positioning inside the `canvas` element to go out-of-sync. It's really quite sad.

The solution is to pair up your `clientX` and `clientY` properties with the following code (taken from the Get an Element's Position Using JavaScript article) that takes into account all of the various position-related shenanigans that your `canvas` might be affected by:

```
// Helper function to get an element's exact position
function getPosition(el) {
  var xPos = 0;
  var yPos = 0;

  while (el) {
    if (el.tagName == "BODY") {
      // deal with browser quirks with body/window/document and page scroll
      var xScroll = el.scrollLeft || document.documentElement.scrollLeft;
      var yScroll = el.scrollTop || document.documentElement.scrollTop;

      xPos += (el.offsetLeft - xScroll + el.clientLeft);
      yPos += (el.offsetTop - yScroll + el.clientTop);
    } else {
      // for all other non-BODY elements
      xPos += (el.offsetLeft - el.scrollLeft + el.clientLeft);
      yPos += (el.offsetTop - el.scrollTop + el.clientTop);
    }

    el = el.offsetParent;
  }
  return {
    x: xPos,
    y: yPos
  };
}
```

Here is an example of this code in action inside our `canvas` element:

```
var canvas = document.querySelector("#myCanvas");
var context = canvas.getContext("2d");
var canvasPosition = getPosition(canvas);

canvas.addEventListener("mousemove", doSomething, false);

// take into account page scrolls and resizes
window.addEventListener("scroll", updatePosition, false);
window.addEventListener("resize", updatePosition, false);

function updatePosition() {
  canvasPosition = getPosition(canvas);
}

function doSomething(e) {
  // get the exact mouse X and Y coordinates
  var mouseX = e.clientX - canvasPosition.x;
  var mouseY = e.clientY - canvasPosition.y;

  // print it to the console
  console.log("The mouse position is: " + mouseX + ", " + mouseY);
}
```

The `canvasPosition` variable stores the object returned by the `getPosition` function, and the `mouseX` and `mouseY` variables store the exact position once you combine `canvasPosition`'s result with the `clientX` and `clientY` value:

```
var mouseX = e.clientX - canvasPosition.x;
var mouseY = e.clientY - canvasPosition.y;
```

For completeness, we even listen to the `window` **resize** and **scroll** events to update the value stored by `canvasPosition`:

```
// take into account page scrolls and resizes
window.addEventListener("scroll", updatePosition, false);
window.addEventListener("resize", updatePosition, false);

function updatePosition() {
  canvasPosition = getPosition(canvas);
}
```

We do this to ensure that our mouse position values remain accurate even if users scroll the page or resize the browser window. All of the code you see here might be going away in the future, though. There is a scheme underway to add an `offsetX` and `offsetY` property to the `MouseEvent` object that automatically takes care of all this.