One Mixin to Rule Them All?

Retina Spriting with SASS and Compass

Jen Zajac, @jenofdoom

with props to Bryn Heveldt

SASS

- How many folks here use SASS or another CSS preprocessor?
- http://sass-lang.com/

Compass

- Compass is a framework for SASS, which gives you lots of prewritten mixins, a compiler and other neat stuff
- http://compass-style.org/

Sprites

- Sprites are good because they reduce HTTP requests
- Just one sprite rather than a bunch of seperate images
- Use background-position to show only the part of the sprite file we care about

Spriting with Compass

- One of the best features of Compass is its built in spriting
- You just drop a bunch of PNGs into a folder and Compass can import them, create a sprite, and give you the co-ordinates for the various images

```
/////// SCSS

@import "mySpriteFolder/*.png";

$mySpriteFolder-spacing: 15px;

.my-selector {
    @include mySpriteFolder-sprite(
        myImageName,
        $offset-y: 5px,
        $offset-x: 0px
    );
}
```

```
////// COMPILED CSS

.my-selector {
    background: url("mySpriteFolder-s0b5333a.png")
        no-repeat
        scroll
        0 0
        rgba(0, 0, 0, 0);
}

.my-selector {
    background-position: 9px -3312px;
}
```

Retina images

- We want higher resolution images for newer displays, smartphones etc
- We can use a sprite file still, but we'll have to reduce the size of the image using background-size so it still takes up the same amount of physical space

Adding a retina sprite

- We can import an additional folder for highres images
- The tricky part is: we need to know the width of the *generated* sprite file in order to halve it
- Compass gives us a nice function for this, `image-width(sprite-path($myHighresSpriteFolder-sprites))`

Why can't we just use %?

- If you give background-size a % attribute it will be a percentage of the container you are setting the property on, not a percentage of the image you are supplying

Wouldn't it be nice…

… to have one mixin that sorted out both the regular and retina sprite image at the same time?

```
$myHighresSpriteFolder-spacing: $mySpriteFolder-spacing * 2;
$sprites-media-query: "(min-device-pixel-ratio: 1.2)";

@mixin bg-sprite($name, $offset-x: 0, $offset-y:0, $highres-version:
false, $highres-offset-x: $offset-x, $highres-offset-y: $offset-y )
{

  background-image: sprite-url($mySpriteFolder-sprites);
  background-position: sprite-position($mySpriteFolder-sprites,
$name, $offset-x, $offset-y);
  background-repeat: no-repeat;

  @if $highres-version == true {
    $highres-offset-x: $highres-offset-x * 2;
    $highres-offset-y: $highres-offset-y * 2;

    @media #{$sprites-media-query} {
      background-image: sprite-url($myHighresSpriteFolder-sprites);
      background-position: round(nth(sprite-
position($myHighresSpriteFolder-sprites, $name, $highres-offset-x,
$highres-offset-y), 1) / 2) round(nth(sprite-
position($myHighresSpriteFolder-sprites, $name, $highres-offset-x,
$highres-offset-y), 2) / 2);
      @include background-size(round(image-width(sprite-
path($myHighresSpriteFolder-sprites)) / 2) auto);
    }
  }
}
```

Um, let's break that down

```
$myHighresSpriteFolder-spacing:
    $mySpriteFolder-spacing * 2;

$sprites-media-query:
    "(min-device-pixel-ratio: 1.2)";
```

```
@mixin bg-sprite(
    $name,
    $offset-x: 0,
    $offset-y: 0,
    $highres-version: false,
    $highres-offset-x: $offset-x,
    $highres-offset-y: $offset-y)
{
    background-image:
        sprite-url($mySpriteFolder-sprites);

    background-position:
        sprite-position(
            $mySpriteFolder-sprites,
            $name,
            $offset-x,
            $offset-y);

    background-repeat: no-repeat;
```

```
@if $highres-version == true {

$highres-offset-x: $highres-offset-x * 2;
$highres-offset-y: $highres-offset-y * 2;

@media #{$sprites-media-query} {

  background-image:
    sprite-url($myHighresSpriteFolder-sprites);

  @include background-size(
    round(image-width(
      sprite-path
        ($myHighresSpriteFolder-sprites)
    ) / 2)
    auto
  );
```

```
background-position:
    round(
      nth(sprite-position(
          $myHighresSpriteFolder-sprites,
          $name,
          $highres-offset-x,
          $highres-offset-y
      ), 1)
    / 2)

    round(
      nth(sprite-position(
        $myHighresSpriteFolder-sprites,
        $name,
        $highres-offset-x,
        $highres-offset-y
       ), 2)
    / 2);
```

Now we can call it

```
@include bg-sprite(my-icon, 5px, 6px, true);
```

Check it out on github:

github.com/jenofdoom/retina-spriting-demo

Other things I didn't have time to cover:

- Debug mode
- Multiple background images with gradients